Design de Jogos Baseado em Componentes

Marcos Silvano Orita Almeida

TESE APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA
OBTENÇÃO DO TÍTULO
DE
DOUTOR EM CIÊNCIA DA COMPUTAÇÃO

Programa: Doutorado em Ciência da Computação
Orientador: Prof. Dr. Flávio Soares Corrêa da Silva

São Paulo, julho de 2016.

Design de Jogos Baseado em Componentes

Esta é a versão original da tese elaborada pelo candidato Marcos Silvano Orita Almeida, tal como submetida à Comissão Julgadora

Resumo

ALMEIDA, Marcos S. O. Design de Jogos Baseado em Componentes. Tese (Doutorado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2016.

O termo "elementos" é comumente encontrado em publicações relacionadas a jogos digitais. A noção de que os jogos podem ser compreendidos e discutidos a partir de suas partes formadoras é implicitamente aceita, embora nenhuma definição formal tenha sido estabelecida. Nesse sentido, designers de jogos frequentemente experimentam uma grande quantidade de títulos a fim de construir uma base de conhecimentos acerca dos elementos que os formam, com o intuito de reaplicá-los em novos projetos. Ao mesmo tempo, pesquisadores e profissionais observaram a necessidade por ferramentas mais formais de design, destacando abordagens baseadas em coleções de conceitos reusáveis de jogos. Embora algumas implementações tenham sido feitas, nenhuma sucedeu como ferramenta prática, tratando tais conceitos como abstrações de design de alto nível.

Neste trabalho acredita-se que os jogos possam ser desmembrados em seus componentes formadores e que esses podem ser estruturados, analisados e combinados a fim de apoiar a criação de novos jogos e a formação de uma base de conhecimentos universal de design. Para tanto, é proposta e demonstrada uma abordagem de Design Baseada em Componentes de Jogos, que define uma perspectiva apoiada em composição para pensar, projetar, registrar e comunicar decisões de design de jogos. A abordagem contribui para a sistematização do processo de design de jogos a medida que define elementos estruturantes para o design e abre possibilidades de aplicação de análises na composição dos jogos, a fim de permitir a descoberta de conhecimentos de design. Além disso, estabelece o alicerce conceitual para a construção de um Ambiente de Design de Jogos. Como demonstração, a abordagem é aplicada a dois estudos de casos para extrair aspectos de jogos existentes e para empregá-los no design de dois protótipos, posteriormente implementados para permitir experimentações estéticas.

Palavras-chave: Ferramentas de design de jogos; design de jogos baseado em componentes; componentes de jogos.

Abstract

ALMEIDA, Marcos S. O. Components Based Game Design. Tese (Doutorado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2016.

"Game element" is a commonly referred term in games related publications. The notion that games can be constituted comprehended and discussed from its forming parts is implicitly accepted, although no formal definition has been set. In that sense, designers commonly play a large amount of games in order to build a knowledge base over the elements that compose them, aiming to reuse these elements in new projects. At the same time, researchers and designers observed the need for more formal design tools, highlighting approaches based on collections of reusable game concepts. Although some implementations have been created, none have succeeded as a practical tool, addressing those concepts as high level design abstractions.

We believe that games can be dismembered into their forming components and these can be structured, analyzed and combined in order to aid the creation of new games and the conception of a design universal knowledge base. To that end, this work proposes and demonstrates a Components Based Game Design approach, which describes a perspective based on composition to think, design, register and comunicate design decisions of games. The approach contributes to the systematization of the game design process by defining the structural elements of the design and by openning possibilities of application of games composition analysis in order to allow the knowledge discovery in design. Furthermore, it establishes the conceptual base to the construction of a Game Design Environment. As demonstration, the approach is applied to two case studies in order to extract aspects from existing games and to apply them on the design of two prototypes, with later implementations to allow esthetic experimentations.

Keywords: Game design tools; components based game design; game components.

Sumário

Lista de Abreviaturas

GDD	Documento de Design de Jogos (*Game Design Document*)
RPG	Role-Playing Game
RTS	Estratégia em Tempo real (*Real-Time Strategy*)
XP	Programação Extrema (*Extreme Programming*)
GDC	Game Developers Conference
FADT	Ferramentas de Design Abstratas Formais (*Formal Abstract Design Tools*)
FPS	Tiro em Primeira Pessoa (*First-Person Shooter*)
MDA	Mecânicas-Dinâmicas-Estéticas (*Mechanics-Dynamics-Aesthetics*)
UML	Linguagem de Modelagem Unificada (*Unified Modeling Language*)
POV	Ponto de Vista (*Point-Of-View*)
HUD	Mostradores (*Head-Up Display*)
CASE	Engenharia de Software Apoiada por Computador (*Computer-Aided Software Engineering*)
OO	Orientação a Objetos
NPC	Personagem não controlável (*Non-Player Character*)
SDK	Kit de Desenvolvimento de Software (*Software Development Kit*)

Lista de Figuras

Capítulo 1 - Contextualização

O design de jogos é a atividade responsável pela concepção de um jogo, que ocorre comumente pelo planejamento, criação e testes de cada um de seus aspectos. É um ofício considerado parte integral e fundamental de todo time de desenvolvimento e do processo de construção de jogos como um todo (Kreimeier, 2003). De um modo simples, jogos são programados por engenheiros de software, ilustrados por artistas gráficos, sonorizados por músicos e concebidos por designers de jogos. O designer de jogos é o profissional responsável pelas ideias que dão início ao conceito de um novo jogo, bem como, pelo seu posterior minucioso detalhamento. Ele está presente durante todo o processo de produção do jogo, garantindo que suas ideias sejam concretizadas nas atividades realizadas pelos demais integrantes da equipe. Cabe ao designer de jogos fazer do jogo um software divertido e engajante, e não somente interativo.

Concomitantemente ao amadurecimento das técnicas e tecnologias que alicerçam a produção e a execução de jogos, a atividade do design de jogos certamente contribuiu de forma significativa para o sucesso e o crescimento da indústria nas últimas décadas. No decorrer de seu percurso histórico, o design de jogos foi gradativamente passando de uma atividade puramente artística, informal e de acertos casuais, para um ofício de cunho profissional (Rogers, 2010). Nesse sentido, pesquisadores e profissionais da indústria têm fomentado a busca pela constituição e o estabelecimento de técnicas e ferramentas mais padronizadas de design (Dormans, 2012). Embora esparsos, viu-se o surgimento de diferentes estudos acadêmicos e abordagens de design, que emergiram moldadas pelas experiências prévias de seus autores. Profissionais e pesquisadores passaram a apresentar diferentes visões acerca da constituição da atividade do design de jogo, seja pela definição das regras do jogo, pela concepção da narrativa que exerce ou pela criação experimental (Neil, 2012). Para tanto, eles buscaram em outras áreas do conhecimento, notadamente da produção de software, cinema e literatura, por conceitos, técnicas e ferramentas que lhes permitissem discutir a constituição de um instrumental de design de jogos que padronizasse e simplificasse a atividade (Dormans, 2012). No entanto, a juventude da área e as características intrínsecas da mídia dos jogos tornam a sua concepção uma atividade dependente das habilidades pessoais do designer de jogos (Kreimeir, 2003; Neil, 2012).

O design de jogos é um ofício idiossincraticamente apoiado nas habilidades do designer (Neil, 2012). Essas habilidades estão geralmente relacionadas a atividades como criação, pesquisa e sistematização, que podem ser caracterizadas respectivamente pela concepção de ideia dos jogos, pela busca de características interessantes em jogos e mídias relacionadas e pela organização da atividade de design como um todo. Contudo, diferentemente das outras áreas necessárias a produção de jogos, como a computação, a produção sonora e a criação visual, o design de jogos não tem o suporte de ferramentas consolidadas (Dormans, 2012). Designers comumente se apoiam em suas experiências passadas e no trabalho de outros para construir

seus jogos (Church, 1999). Nesse sentido, designers usualmente investigam jogos similares ao que estão produzindo para determinar acertos e falhas, identificando suas características e descobrindo quais podem ser reusadas de forma benéfica em um novo projeto.

A recorrência de características em jogos é, até certo ponto, algo esperado. Como exemplo, em franquias de jogos, os novos títulos tendem a manter um grande conjunto das características presentes em seus antecessores. Da mesma forma, há características que se deseja encontrar em jogos de um determinado gênero. Segundo Lindley (2003), os gêneros de jogos estabelecem uma taxonomia que define características fundamentais de gameplay aos títulos que classificam. Empregados constantemente pelo mercado e academia, os gêneros definem um conjunto hierárquico de categorias de jogos, baseando-se nas principais formas de interação encontradas nestes (Jäniven, 2008). Desse modo, é relevante ao designer de jogos estudar títulos existentes a fim de determinar características que se despontam e que podem contribuir para novos projetos (Church, 1999). A noção de que os jogos podem ser compreendidos e discutidos a partir de suas partes formadoras é implicitamente aceita e referenciada em diversos trabalhos (Costykian, 1994; Church, 1999; Björk et al., 2003, Kreimeier, 2003; Neil, 2012), embora nenhuma definição formal tenha sido estabelecida.

Church (1999), Kreimeir (2002) e Björk et al. (2003), apontam que a modificação ou combinação de conceitos de jogos existentes pode servir de ferramenta de análise e criação, auxiliando o designer na obtenção de novas ideias. Nesse sentido, não é incomum que a concepção de um novo jogo parta de uma ideia baseada em conceitos presentes em jogos existentes (Bates, 2004; Adams & Rollings, 2003). Além disso, experimentações com base na mistura de características de gêneros distintos também são empregadas para esse propósito (Lindley, 2003). Esse estilo "engenheiro" de design de jogos, no qual um jogo é tratado como uma estruturação de conceitos é por vezes discutido na literatura formal (Kreimeier 2003; Dormans, 2012; Neil, 2012; Almeida e Corrêa da Silva, 2013). Mesmo na mídia especializada, que emprega um vocabulário desprendido de formalidade, o termo "elemento de jogo" é comumente utilizado para se referir as partes de jogos. Contudo, cabe ressaltar que o termo não é unânime e tão pouco padronizado na academia e na indústria, pois as palavras "conceito", "aspecto" e "mecânica" são por vezes utilizadas como seus sinônimos. De forma geral, há um consenso de que existem características que descrevem a constituição dos jogos, mas nenhuma tentativa anterior de formalização foi bem-sucedida. Em geral, essas tentativas têm focado em estruturar e documentar conceitos e práticas recorrentes de design de jogos, dando pouca ou nenhuma atenção ao estudo e identificação das partes menores que possam formar o conceito de um jogo.

O modelo atual de design de jogos, orientado a narrativas e guiado por uma documentação intensamente textual, dificulta a padronização e a formalização do design (Neil, 2012). As tentativas de solucionar esse problema, como as discutidas em Kreimeier (2003), Neil (2012) e Dormans (2012) tem sido direcionadas à estruturação de conceitos e práticas recorrentes de design de jogos, distanciando-se da formalização das partes menores que os formam. Com isso chega-se a questão fundamental deste trabalho: Seria possível definir

elementos estruturantes do design de forma que jogos e gêneros possam ser tratados como composições de partes menores? Neste trabalho, acredita-se que a estruturação da forma de pensar, projetar, registrar e comunicar decisões de design para os jogos sob uma abordagem de componentização[1] de design pode contribuir significativamente com o processo de design de jogos como um todo. Para o autor, existe um vasto conhecimento de design embutido nos jogos já produzidos que pode ser resgatado por meio dos componentes que empregam. Nesse contexto, esta tese tem por objetivo propor e demonstrar uma abordagem de design de jogos baseada em componentes.

1.1 Componentes de Jogos

Alguns autores referenciam o termo "mecânicas" como os elementos de interação que compõem o gameplay[2] dos jogos (Järvinen, 2008; LeBlanc et al., 2004; Sicart, 2008). Outros descrevem os jogos como sistemas de regras (Salen & Zimmerman, 2003). Contudo, a singularidade de um jogo não recai somente sobre suas mecânicas ou regras. Além da obviedade de que parte da identidade de um jogo é construída sobre seu estilo artístico, há muitas outras características que devem ser consideradas. A forma como o jogador controla o personagem do jogo, seus objetivos, as características da narrativa empregada, ponto de vista, forma de progressão, objetivos, tipos de construções de fases (level design) ou mesmo seus mostradores (HUD – Heads-Up Display[3]), são alguns exemplos de aspectos que apresentam forte influência sobre a forma pela qual o jogador experiencia o jogo. Nesse contexto, é apresentada a definição de componentes de jogos:

> Um **componente** é qualquer aspecto distintamente reconhecível de jogos que possa ser identificado, desconectado do todo e apresente influência estética[4].

Cabe ressaltar que embora haja semelhança entre o termo "componente" e seu emprego em outras áreas, não há intenção de correspondência. Dessa forma, a expressão "componente de jogo" não apresenta relação à engenharia de software, mas sim, aos conceitos de composição e decomposição de partes em vista de formar um todo. Por outro lado, a abordagem se baseia fortemente no paradigma de Orientação a Objetos do desenvolvimento de software (Fowler, 2004), conforme discussão apresentada no Capítulo 3.

Componentes tipicamente apresentam recorrência em jogos, independente de pertencerem ao mesmo gênero. Essa característica ajuda a identificar os aspectos que podem ser qualificados como componentes. Nesse sentido, podem referir-se a diferentes aspectos do design de um jogo, como por exemplo, regras, entidades, mecânicas, UI (interface com o usuário), história ou quaisquer outras caracterizadas que contribuam para compor a experiência

---

[1] O termo "componentização", do inglês *componentization*, não está definido na língua portuguesa. No entanto, é empregado neste trabalho por descrever adequadamente o método de composição de design a partir de partes menores.

[2] *Gameplay*: forma pela qual o jogador interage e experiencia as situações presentes no jogo.

[3] http://tvtropes.org/pmwiki/pmwiki.php/Main/HeadsUpDisplay

[4] A estética, sob a definição de LeBlanc et al. (2004), compreende as respostas emocionais do jogador ao interagir com os mecanismos do jogo.

de *gameplay*.

Componentes representam aspectos intrinsecamente característicos dos jogos e distintamente reconhecíveis, tanto pela perspectiva do designer, quanto do próprio jogador. Como exemplo, o componente[5] [Jump][6] representa a ação de pular e é claramente identificável na grande maioria dos jogos de ação nos quais o jogador controla um personagem. Tal componente é facilmente encontrado nos gêneros Platformer[7] [8], FPS (First Person Shooter[9]), Third Person Shooter[10] e Action-Adventure[11]. Jogos que envolvem exploração, como os das séries Metroid[12] e Zelda[13], os primeiros títulos da franquia Resident Evil[14] e o primeiro Doom[15], empregam o componente [Backtracking], um conceito que denota o ato de retornar a áreas previamente exploradas para obter novos recursos ou alcançar novos locais. O componente [Health Bar] do tipo <HUD> é utilizado sob diferentes representações visuais em jogos nos quais os personagens possuem uma quantidade de energia vital consumida à medida que sofrem danos. Em jogos do gênero FPS, o controle do personagem pode ser designado pelo componente [FPS Controller], composto por [Mouse Aim], [Move Forward and Backward] e [Strafe]. Ainda nesse contexto, alguns jogos concedem ao jogador as ações de [Jump] e [Sprint].

Os componentes definem elementos estruturantes para o design de jogos. Eles agem como unidades menores pelas quais pode-se compor o conceito de um jogo, de forma similar a uma abordagem de blocos de construção, na qual a composição tem função fundamental. O objetivo é permitir ao designer registrar e comunicar suas ideias de forma sintética e padronizada. A Figura 1 apresenta a composição de um personagem baseado na Piranha Plant da série de jogos Super Mario Bros[16]. No exemplo, o personagem é representado estruturalmente pela junção de componentes de comportamento (<behavior>), ataque (<attack>) e interação (<interaction>). Ele está rotulado como um inimigo (<enemy>) e uma entidade (<entity>), que o caracterizam como uma "criatura" do mundo do jogo que tem por finalidade provocar danos no personagem do jogador, ou simplesmente, dificultar sua progressão. Outra entidade aparece relacionada na composição da [Piranha Planta], o [Jump Man], que corresponde ao personagem controlado pelo jogador. No entanto, um jogo não é somente composto por entidades, pois aspectos como [Backtracking], [Score] e [Health Bar] não constituem objetos do mundo do jogo, mas apresentam considerável relevância para o

---

[5] As representações de componentes neste trabalho serão feitas com termos em inglês a fim de manter uniformidade com a nomenclatura empregada em jogos e em trabalhos relacionados.
[6] Os componentes são sintaticamente identificados como [component]. Os rótulos, que os organizam e classificam, são definidos por <tag>.
[7] http://tvtropes.org/pmwiki/pmwiki.php/Main/PlatformGame
[8] O site TV Tropes é frequentemente referenciado para definições de jogos e gêneros. Ele contém uma ampla base de dados de conceitos de jogos que, embora informal, é rica em conteúdo e de fácil acesso.
[9] http://tvtropes.org/pmwiki/pmwiki.php/Main/FirstPersonShooter
[10] http://tvtropes.org/pmwiki/pmwiki.php/Main/ThirdPersonShooter
[11] http://tvtropes.org/pmwiki/pmwiki.php/Main/ActionAdventure
[12] http://tvtropes.org/pmwiki/pmwiki.php/Franchise/Metroid
[13] http://tvtropes.org/pmwiki/pmwiki.php/Franchise/TheLegendOfZelda
[14] http://tvtropes.org/pmwiki/pmwiki.php/Franchise/ResidentEvil
[15] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/Doom
[16] http://tvtropes.org/pmwiki/pmwiki.php/Franchise/SuperMarioBros

gameplay. Por esta razão, componentes não estão restritos a entidades.

```
[Piranha Plant] <enemy> <entity>
{
      <behavior>:
      [Hide and Show]
      {
            type = Vertical movement
      }
      [Look at Target]
      {
            target = [Jump Man]
      }

      <attack>:
      [Damage by contact]
      {
            target = [Jump Man]
      }
      [Shoot on Target]
      {
            target = [Jump Man]
            rate = one at time
            range = [Jump Man] is far
      }
      [Melee attack]
      {
            range = [Jump Man] is near
      }

      <interaction>:
      [Imune to attack]
      {
            source = [Jump Man]
      }
  }
```

Figura 1 – Representação textual do personagem Piranha Plant em componentes.

No esquema da Figura 1[17], a composição do personagem Piranha Plant é exposta em formato textual. Na sintaxe apresentada, os componentes estão entre colchetes ([componente]) e seus atributos, quando presentes, ficam dispostos em seu interior. O corpo de cada componente é delimitado por caracteres de início "{" e fim "}" e pelo emprego de endentação. Os rótulos, indicados por "<" e ">", servem de classificadores e podem ser usados para estruturar e guiar a descrição de componentes. No exemplo mostrado, os rótulos <enemy> e <entity> são usados para classificar o componente [Piranha Plant]. Já os rótulos <behavior>, <attack> e <interaction> servem de guias estruturais para a apresentação do mesmo componente. A Figura 2 descreve o mesmo componente com uma representação visual, que pode ser utilizada em forma de pôster ou cartão de design, em um estilo de documentação baseado no Design de Uma Página apresentado por Librande (2010). Nota-se nos exemplos que a composição é empregada como uma ferramenta de definição do comportamento das entidades.

[17] Várias figuras contidas nesta tese estão coloridas. Para melhor entendimento de seu conteúdo, recomenda-se a leitura da tese em sua versão digital, com as cores preservadas.

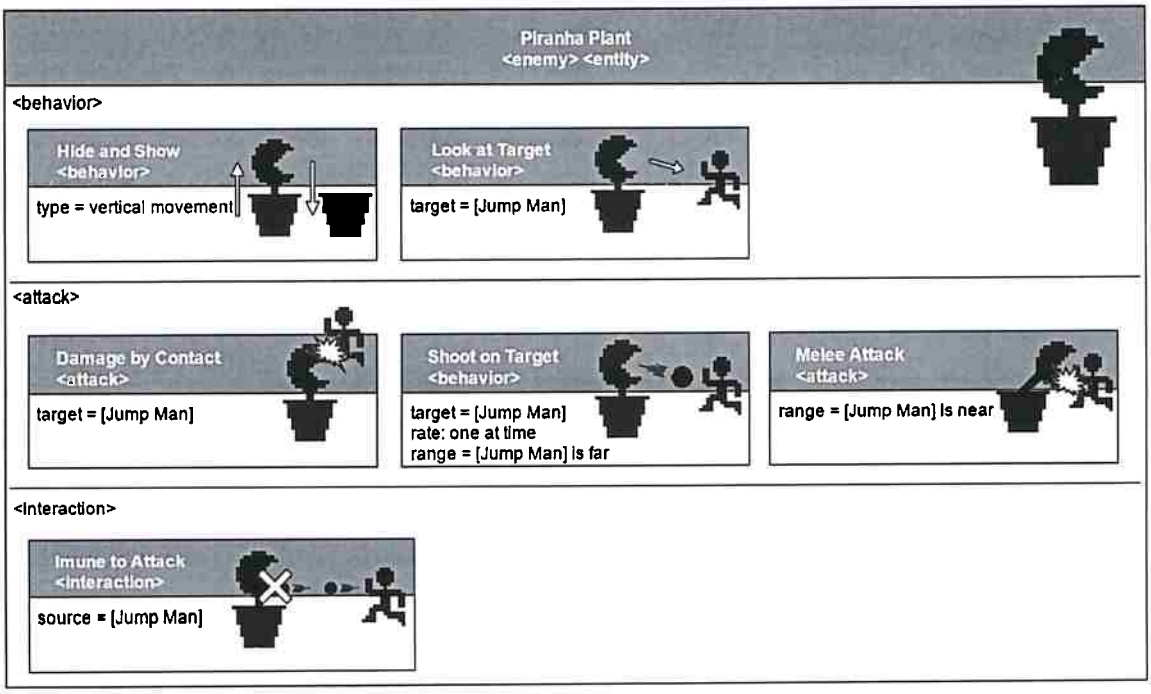

Figura 2 – Representação visual do personagem Piranha Plant em componentes.

Os esquemas apresentados nas Figuras 1 e 2 podem ser descritos textualmente da seguinte maneira: o personagem Piranha Plant é uma entidade que existe no mundo do jogo e visualmente é representada como uma planta carnívora dentro de um vaso. O comportamento dela é compreendido por expor-se e esconder-se (para dentro do vaso) periodicamente. Quando exposta, ela mantém-se voltada para o jogador (Jump Man). Seus ataques restringem-se a disparar bolas de fogo na direção do jogador, caso ele esteja longe, e investir contra o jogador tentando mordê-lo quando próximo. Além disso, o personagem do jogador sofre danos caso tenha contato com a planta. Por outro lado, ela é imune a qualquer ataque do jogador, não podendo ser destruída.

Analisando a composição da [Piranha Plant], nota-se que ela emprega componentes frquentemente recorrentes em jogos. O comportamento (<behavior>) identificado por [Hide and Show] é empregado sob diferentes apresentações visuais em diversos jogos. Como exemplo, a entidade que o executa pode esconder-se e mostrar-se em um movimento vertical ou horizontal, o que é tipicamente encontrado em inimigos do gênero Platformer. Em outro caso, jogos do gênero Tiro (Shooter), especialmente os First-Person Shooters, tipicamente apresentam personagens que podem desaparecer e reaparecer utilizando uma camuflagem que o torne invisível. Outro componente de comportamento da [Piranha Planta] recorrente em jogos é [Look at Target], frequentemente utilizado na composição de torres de ataque em Shooters.

Os componentes de ataque (<attack>) usados na descrição do [Piranha Plant] também apresentam recorrência comum em jogos. O [Damage by Contact] está presente na composição da maioria dos inimigos e obstáculos de jogos 2D, pois provê uma maneira simples de impedir que o jogador simplesmente passe por eles. Por outro lado, jogos 3D usualmente permitem que haja colisão entre o avatar do jogador e outras entidades sem que o primeiro sofra danos, mesmo contra aquelas que representam inimigos. Exemplos são jogos do tipo Fist-Person

Shooter e títulos da série God of War[18]. Já o componente de ataque [Shoot on Target] é quase unânime em jogos de tiro, tanto 2D quanto 3D. O mesmo ocorre com [Melee Attack], que representa o ato de desferir golpes de contato físico de curto alcance. Ele é usado na grande maioria dos jogos de ação 2D e 3D, especialmente aqueles pertencentes aos gêneros Beat-em-Up[19] e First-Person Shooter contemporâneos.

O exemplo da [Piranha Plant] mostra como os componentes podem ajudar a estruturar a forma de pensar, projetar, registrar e comunicar o conceito de um jogo. É possível notar que cada componente define um termo, identificando um conceito ou aspecto de jogos. Um dos objetivos da abordagem de componentes é contribuir para a padronização de um vocabulário de design. Por outro lado, diferentes designers poderão empregar nomes distintos para os mesmos componentes, guiados por sua interpretação pessoal. O registro destas informações em uma base de dados poderia diminuir tais vieses, ajudando a estabelecer uma maior uniformidade na comunicação de conceitos de design. Quanto a modelagem dos componentes em si, o objetivo da abordagem é permitir uma construção mais sintética, de fácil comunicação. Certamente, situações mais complexas do que a apresenta no exemplo certamente admitirão diferentes resultados. A modelagem é, por natureza, uma atividade de criação e não está restringida a uma interpretação absoluta. Resulta da perspectiva do designer sobre a situação tratada. Dessa forma, assim como em toda atividade de modelagem em design, existe uma inerente interpretação abstrata, uma parcialidade do fator humano nos designers que os levará a diferentes representações de componentes para uma mesma situação de design.

1.2 Desenvolvimento de jogos: uma visão geral

O desenvolvimento de jogos é o processo de desenvolvimento de software pelo qual um videogame – jogo digital – é produzido (Bethke, 2003). Sua realização envolve atividades de naturezas notadamente distintas, em especial a programação de software, a criação de arte gráfica, a sonoplastia e o design de jogos[20], que pode ser compreendido como a "concepção de jogos". Esses quatro papéis representam as atividades fundamentais à construção de um jogo digital. O atuante da área é denominado desenvolvedor de jogos e pode exercer um ou mais dos papéis citados em um projeto. Na indústria tradicional, na qual existe uma cadeia de produção financiada por grandes empreendimentos (Chandler, 2009), os papéis fundamentais são por vezes subdivididos em atividades específicas e altamente especializadas. Já no âmbito da produção independente, no qual os jogos são construídos por pequenos grupos sem o apoio financeiro significativo de empresas (Chandler, 2009), os desenvolvedores de jogos tipicamente executam vários papéis, não sendo raro a produção de um jogo por uma única pessoa. Nesse contexto, faz-se necessário apontar que existe uma distinção entre os termos "desenvolvimento" e "produção" (Bethke, 2003). O primeiro comumente refere-se ao processo

[18] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/GodOfWarSeries
[19] http://tvtropes.org/pmwiki/pmwiki.php/Main/BeatEmUp
[20] Para fins de padronização, o termo **Design de Jogos** ou **Design** é empregado como uma tradução do termo ***Game Design***. Já a atividade de produção de artefatos visuais, o ***Graphic Design***, é denominada de **Arte Gráfica**.

específico e técnico de construção de um jogo como um artefato de software. O segundo, por outro lado, denota também os aspectos mercadológicos a fim de fazer do jogo um produto. Dessa forma, a produção de um jogo envolve não somente o seu desenvolvimento, mas também a gestão financeira, de marketing e de negócio relativa ao produto que será comercializado. Cabe ressaltar que este trabalho está relacionado ao desenvolvimento de jogos.

Um motor ou *engine* é a principal ferramenta utilizada no desenvolvimento de jogos. Ele consiste de um conjunto de recursos de software que possibilita a criação e a execução de um jogo (Rogers, 2010). As principais funcionalidades tipicamente encontradas em um motor de jogos incluem o renderizador gráfico 2D/3D, o motor de simulações físicas e tratamento de colisões, o compilador, o interpretador de scripts, bem como, os gerenciadores de som, rede, animação, inteligência artificial, tarefas, memória e alocação de recursos. Alguns dos principais motores utilizados no mercado empregam uma variação do padrão arquitetural conhecido por C/ES – Sistemas de Entidades e Componentes[21]. Exemplos desses motores são as ferramentas Unity[22], Unreal[23], Cryengine[24]. Neste padrão, os objetos que compõem as cenas dos jogos são tratados como entidades, tais como personagens, obstáculos, itens, veículos, portas, armas e projéteis. As entidades funcionam como contêineres de componentes, que por sua vez, implementam os comportamentos, funcionalidades, dados e a aparência, podendo ser adicionados às entidades para lhes atribuir tais aspectos. Dessa forma, as entidades são definidas pelos componentes que utilizam. O C/ES emprega o princípio de "Favorecer a Composição sobre a Herança" (Gamma et al., 1994), segundo o qual os comportamentos e o reuso de funcionalidades são atingidos por meio da composição e não pela herança. Nesse sentido, as entidades são definidas por componentes que implementam as funcionalidades desejadas, ao invés de tornarem-se subtipos de outras entidades. Esse padrão tem se tornado gradativamente mais influente no desenvolvimento do software de jogos.

O desenvolvimento de jogos envolve métodos e processos provenientes de outras áreas, destacando-se o desenvolvimento de software. Embora a experiência de uso de um jogo passe ao usuário a impressão de forte proximidade à produção de filmes, jogos são produzidos como softwares (Bates 2004), uma vez que o software do jogo é o que permite a sua existência. Em termos do processo de desenvolvimento, é comum ocorrer o emprego de métodos ágeis, que permitem o desenvolvimento iterativo e de refinamento gradual das características do jogo, muitas vezes em mudança durante a etapa de implementação (Bethke, 2003; Bates, 2004). O processo típico de desenvolvimento de jogos envolve três etapas: Pré-Produção, Produção e Pós-Produção (Bethke, 2003). Elas são brevemente apresentadas no texto que segue.

O processo de desenvolvimento inicia com a etapa de Pré-Produção, que envolve o planejamento e a concepção do conceito do jogo, bem como, a realização de testes de conceitos e produção de protótipos. É nesta fase que a documentação de design e de projeto é produzida.

---

[21] http://entity-systems.wikidot.com
[22] http://unity3d.com
[23] https://www.unrealengine.com
[24] http://www.crytek.com/cryengine

Projetos que empregam métodos ágeis tendem a realizar uma fase de Pré-Produção mais curta e, dessa forma, o planejamento e concepção do jogo são feitos no decorrer do processo de desenvolvimento (Bates, 2004; Chandler 2009). Após concluída esta fase, inicia-se a Produção.

A etapa de Produção representa a fase mais longa do processo, se estendendo até o encerramento do projeto e com término na entrega do produto final. Durante esta etapa ocorre a efetiva construção do jogo, que compreende a produção de todos os artefatos que o compõem: visuais (ilustrações, modelos 3D, animações, vídeos, cenários, etc.), sonoros (músicas, efeitos e vozes), de software e de design (mapas, roteiros, diálogos, etc.) (Chandler, 2009). No decorrer da fase de Produção, a equipe de qualidade realiza testes sobre o jogo de forma extensiva, iniciando logo após a conclusão da primeira versão jogável. O objetivo desses testes é encontrar defeitos e erros referentes ao software, como também, fornecer suporte e feedback à equipe de design. Dessa forma é possível de identificar problemas que interfiram no *gameplay*, diminuindo a possibilidade de propagação a estágios mais avançados do projeto (Bethke, 2003). Após a conclusão do jogo, inicia-se a Pós-Produção.

A etapa de Pós-Produção define a terceira e última fase do processo de desenvolvimento e compreende essencialmente atividades de manutenção do jogo após seu lançamento e comercialização. Tais atividades referem-se principalmente a realização de correções e adaptações. Jogos online de múltiplos jogadores massivos possuem uma fase de Pós-Produção contínua, que perdura enquanto o jogo for comercializado e tem por finalidade mantê-lo funcionando adequadamente, bem como, realizar melhorias e ampliação no jogo (Chandler, 2009).

O progresso dentro de cada uma das três etapas do processo de desenvolvimento ocorre comumente de forma iterativa, à medida que o design do jogo é gradativamente estabelecido (Chandler, 2009). Em todas as etapas, há uma atenção especial da equipe ao amadurecimento do design, bem como às mudanças que venha a causar. Existe a preocupação de garantir que o conceito do jogo documentado seja corretamente implementado no resultado final, uma vez que representa a visão dos designers. Esse esforço está atrelado ao fato de que a qualidade de um jogo, aquela percebida pelos seus usuários e pela crítica especializada, não se define pelo cumprimento de uma lista de requisitos (Chandler, 2009). A experiência que emerge da interação do jogador com o jogo é o principal norteador do designer e da equipe de desenvolvimento (Bates, 2004).

Um jogo pode ser definido como um software que agrega alta quantidade de elementos audiovisuais e tem foco na definição e implementação de seu *gameplay*. O *gameplay* define as formas peculiares pelas quais os usuários interagem de forma cognitiva com os jogos, a fim de realizar atividades que os permitam vencer desafios, cumprir objetivos, conquistar recompensas e, consequentemente, experienciar emoções (Lindley et al., 2008). O gameplay gera um envolvimento interativo tipicamente associado aos videogames (Tavinor, 2009). Ele cria um vínculo entre o jogador e o jogo por meio de regras do mundo do jogo, desafios, objetivos e enredo (Salen & Zimmerman, 2003). Segundo Adams e Rollings (2003), o *gameplay* não é uma entidade singular, mas o resultado da combinação de vários elementos, como uma sinergia que

emerge da interação das diversas partes que compõem um jogo. O *gameplay* é o elemento que fundamentalmente distingue os jogos de outras mídias, como filmes, desenhos, livros, quadrinhos e música (Oxford Dictionary, 2008; Oxland, 2004). Sua concepção é a principal responsabilidade do designer de jogos (Salen & Zimmerman 2003; Rogers, 2010).

O design de jogos é a atividade que tem por objetivo conceber, documentar e avaliar os conceitos que definem um jogo (Rogers, 2010). Mais especificamente, o designer é responsável por projetar o gameplay de um jogo, definindo regras, conteúdo, ambientes, roteiro e personagens (Salen & Zimmerman, 2003; Oxland, 2004; Moore & Novak 2010). Seu principal instrumento de trabalho é o Documento de Design, no qual são registrados todos os detalhes do jogo a ser produzido (Bates 2004). Existem diferentes vertentes sobre a forma como se dá a concepção do jogo e sobre a própria ênfase do trabalho do designer. Alguns autores advogam a ideia de que a atividade de design é centrada na elaboração da narrativa do jogo, como uma história a ser contada ao jogador (Crawford, 1984; Rouse, 2000). Outros enfatizam o uso de diretrizes e regras a serem seguidas ao se projetar um jogo, documentadas a partir de experiências prévias de designers em projetos (Fabricatore et al., 2002; Falstein & Barwood, 2002). Há ainda autores que visualizam o design de jogos como uma atividade de engenharia (Bjork et al., 2003; Church, 1999; LeBlanc et al. 2004; Zagal et al., 2005; Kreimeier, 2002) e, nesse sentido, a criação do conceito de um jogo ocorre por meio da definição de suas partes – padrões, conceitos ou "mecânicas" – e das consequências de seu emprego.

Termo comumente encontrado na mídia especializada, as "mecânicas" de um jogo podem ser compreendidas como as formas pelas quais as regras conectam as ações do jogador ao seu propósito no jogo (Sicart, 2008). LeBlanc et al. (2009) define as mecânicas como regras formais que estabelecem como o jogador interagirá com determinadas partes do jogo, que ações ele deve ou pode tomar, quais as consequências dessas ações, que condições de vitória e perda estão definidas, e qual o resultado esperado de seu emprego. Nesse sentido, um jogo é tratado como uma coleção de mecânicas (ou conceitos) que, ao serem combinadas, produzem o gameplay que é experienciado pelo jogador. A estética, sob a definição de LeBlanc et al. (2009), compreende as respostas emocionais do jogador ao interagir com a implementação das mecânicas no jogo. Este conceito de estética é largamente utilizado no presente trabalho, que também trata o design de jogos sob uma ótica de "engenharia de design".

1.3 Motivações e Justificativas

Nas publicações científicas e profissionais da área de design de jogos há um consistente e frequente discurso acerca dos problemas relacionados ao método corrente de design e à falta de ferramentas mais formais e padronizadas (Kreimeier, 2003; Dormans, 2012; e Neil, 2012). Nesse contexto, não existe um método sistematizado de concepção de jogos, mas sim, um documento de design, no qual o conceito do jogo em produção é registrado essencialmente por meio de textos, com o auxílio de figuras e esquemas. Considerado o emprego predominante de textos no documento, é de grande relevência notar que não houve o estabelecimento de um vocabulário universal de design, outra questão frequentemente apontada como uma deficiência

da área (Neil, 2012). Ao mesmo tempo, a noção de que os jogos podem ser compreendidos e discutidos a partir de suas partes formadoras é implicitamente aceita e referenciada em diversos trabalhos (Costykian, 1994; Church, 1999; Björk et al., 2003, Kreimeier, 2003; Neil, 2012), embora não haja uma proposta especificamente relacionada ao tema.

O termo informal "elemento" é comumente empregado em publicações especializadas para referir-se às partes formadoras dos jogos. Nesse contexto, os pioneiros das publicações a discutir as ferramentas de design destacaram a importância da habilidade de se dissecar os jogos em suas partes formadoras, a fim de analisá-las, registrá-las e empregá-las na constituição de novos jogos (Costikyan, 1994; Church, 1999). Contudo, nenhum trabalho apresentado até o momento tratou objetivamente do conceito-chave abordado em tal discurso: o design de jogos por composição. Por outro lado, no campo do desenvolvimento de software o padrão arquitetural de Sistema de Entidades e Componentes tem sido empregado em diversos projetos e motores de jogos (Gregory, 2014). Nesse padrão as entidades, que correspondem aos objetos no jogo, são definidas pelos componentes que empregam. Estes, por sua vez, implementam aspectos das entidades, como comportamentos e aparência. A composição é então empregada como uma ferramenta de definição do comportamento das entidades.

A impossibilidade da experimentação das ideias empregadas em um jogo durante sua concepção é apontada como um obstáculo para a área (Neil, 2012). Protótipos de jogos são comumente produzidos durante o processo de desenvolvimento de jogos, mas há um distanciamento inerente entre design e experimentação, uma vez que esses protótipos são construídos somente após a conclusão de uma parte substancial do documento de design. Isso ocorre naturalmente porque designers não são capazes de construir os próprios protótipos de jogos e essa é uma tarefa comumente delegada à equipe de desenvolvimento. Construtores rápidos podem ser ocasionalmente considerados uma alternativa para este problema, mas restringem-se a recursos preconcebidos projetados para hobistas e não sincronizam com as ferramentas utilizadas pela equipe de desenvolvimento. Para preencher essa lacuna, seria essencial a existência de um Ambiente de Design de Jogos que, dentre outros recursos, permitisse a prototipação rápida por meio do intercâmbio direto de conceitos do jogo entre a documentação de design e a geração de protótipos de experimentação. No entanto, o uso de documentação massivamente textual para o registro das ideais do jogo e a inexistência de um elemento estruturante da composição de jogos inviabilizam tal ferramenta. Nesse contexto, acredita-se que o instrumental conceitual documentado neste trabalho poderá servir de alicerce à construção de um Ambiente de Design de Jogos (GDE - *Game design Environment*), possibilitando projetar e experimentar conceitos de jogos a partir de uma especificação de design estruturada por componentes.

A abordagem de componentes e a componentização podem contribuir significativamente com a sistematização do processo de design como um todo, a medida que definem elementos estruturantes para o design de jogos. Nesse sentido, atividades como a análise de jogos existentes, a pesquisa de conceitos relevantes ao projeto em desenvolvimento, o registro de ideias e a concepção de novos jogos, comuns à função do designer de jogos, podem ser

orientadas ao estudo de componentes e de suas relações. Ademais, ao tratar os jogos como uma composição de partes e ao documentar tal informação em uma base de dados, torna-se possível empregar técnicas de descoberta de conhecimento que podem auxiliar a designers e pesquisadores. Como exemplo, pode-se analisar o nível de similaridade entre jogos, assim como, a popularidade, raridade ou sucesso de determinados componentes em jogos ou gêneros, a partir do cruzamento de dados de mercado e crítica especializada. Também se torna possível gerar "árvores genealógicas" que relacionem jogos, seus componentes e gêneros, de forma a representar a evolução cronológica de títulos e as relações existentes entre suas constituições (ver Figura 16 para exemplo conceitual). Outras possibilidades incluem a construção de assistentes de design – a partir da análise da constituição do jogo e sua relação com componentes típicos de seu gênero ou jogos similares – e a descoberta do grau de influência de determinados componentes na aceitação ou rejeição de jogos ao longo dos anos. Dados esses exemplos de aplicações, é possível notar que a abordagem de design de jogos baseada em componentes pode abrir um considerável leque de possibilidades futuras para aplicações de técnicas da computação que provenham suporte a área de do design de jogos.

Discutidas as possibilidades de aplicações para a abordagem de componentes, faz-se necessário incluir considerações a respeito do escopo negativo desta tese. Nesse sentido, cabe ressaltar que este trabalho não possui a pretensão de definir um instrumental para elevar a qualidade de jogos, nem tampouco de apresentar um estudo sobre a narrativa ou a estética a fim contribuir com a definição de um "bom jogo". De forma similar, não objetiva-se apresentar um método definitivo de design, uma técnica de produção mais eficiente ou que permita criar jogos melhores. Nesse sentido, a abordagem proposta não pretende substituir a documentação tradicional de design, mas sim, complementá-la. O foco deste trabalho está em propor, projetar e demonstrar uma abordagem estruturada baseada em componentes para o design de jogos, sincronizada aos discursos de Costikyan (1994) e Church (1999). Espera-se que esta abordagem possibilite futuramente a realização de análises sobre a constituição dos jogos e que fundamente a construção de um ambiente computacional de design de jogos.

1.4 Formulação do Problema

Kreimeier (2003), Dormans (2012) e Neil (2012) apontam a falta de ferramentas de design e o uso extensivo de recursos textuais e narrativos na documentação como os dois maiores problemas que a área enfrenta. O principal instrumento de design – o Documento de Design de Jogos (GDD) – é considerado muito restritivo. Segundo Neil (2012), diversos autores apontam-no como a principal barreira à definição de uma linguagem universal de design, à transferência de conhecimento entre gerações de designers e à evolução da área como um todo. Embora alguns advoguem o uso do GDD em sua forma mais tradicional – textual e extensa – outros o renegam por completo. Problemas como a produção de documentação excessiva, a falta de padronização no documento e na linguagem utilizada, a dificuldade em mantê-lo e consultá-lo à medida que cresce, e a falta de modelos visuais que facilitem a comunicação do conceito do jogo à equipe, são comumente citados como os principais fatores negativos do GDD (Neil, 2012). Isso significa que, embora os documentos de design devessem registrar e transmitir o

conhecimento adiante para futuros projetos, na prática isso é feito por meio dos conceitos presentes nos jogos produzidos. Não há uma linguagem para descrição de conceitos de design que suporte uma base de conhecimentos estruturada (Church, 1999).

Na busca pelo estabelecimento de ferramentas padronizadas e formais de design, pesquisadores e designers têm discutido abordagens centradas na constituição de um vocabulário unificado de design e na criação de linguagens visuais de modelagem de jogos (Almeida e Corrêa da Silva, 2013). Nesse contexto, "formal" não se refere à adoção de modelos matemáticos, mas sim a métodos e ferramentas padronizadas e estruturadas que forneçam suporte ao processo de design de jogos, em posição à ideia de baseá-lo puramente em descrições textuais. Nota-se nesses trabalhos um direcionamento para abordagens baseadas em coleções de conceitos ou diretrizes de design. Um problema identificado reside no alto nível de abstração pelo qual esses conceitos são tratados, impedindo a descrição de jogos por meio de sua composição e tornando-os de aplicação vaga e imprecisa. Em sua maior parte, esses conceitos representam diretivas abstratas de design, não havendo uma preocupação em torná-los ferramentas de uso concreto. Ademais, não são apresentadas demonstrações de como esses conceitos podem ser empregados no design de novos jogos. Dessa forma, ficam restritos a discussões teóricas que, embora pertinentes, não colaboram diretamente para a estruturação de ferramentas computacionais de apoio ao design de jogos, frequentemente apontadas por diversos autores como uma das necessidades da área (Neil, 2012).

1.5 Objetivos

O presente trabalho documenta uma abordagem proposta com o objetivo de amenizar o principal problema que aflige a área do design de jogos: a falta de ferramentas e métodos estruturados. Nesse contexto, o principal objetivo deste projeto é propor e demonstrar uma abordagem de Design de Jogos Baseada em Componentes. Estes dois passos tornaram-se guias para a definição dos objetivos secundários deste trabalho:

- **Propor** um modelo estrutural que permita descrever as partes menores formadoras dos jogos: os componentes de jogos. A partir disso:
    - Investigar uma forma de empregar tal abordagem para analisar jogos existentes, a fim de desmontá-los em componentes;
    - Compreender como estes componentes se relacionam e de que forma contribuem para a estética dos jogos;
    - Investigar uma forma de empregar a mesma abordagem para projetar novos jogos, montando-os pela junção de componentes.
- **Demonstrar** a aplicação dos componentes em projetos de teste. Para isso, foi preciso:
    - Aplicar a análise pela decomposição em jogos existentes a fim de obter aspectos significativos de jogos populares para identificá-los como componentes;

- Aplicar o design pela composição a partir dos componentes selecionados, combinando-os e modificando-os para projetar novos jogos, que possam demonstrar clareza na aplicação desses componentes;
- Construir os protótipos dos jogos projetados a fim de permitir uma melhor percepçao dos componentes empregados sob a perspectiva do jogador.

Cabe ressaltar que a abordagem de componentes de design faz parte de um plano mais amplo, que guiará trabalhos futuros do autor na área. Nesse sentido, esta tese estabelece a base conceitual que viabilizará a realização desses trabalhos. A abordagem de componentes foi inicialmente idealizada como um conjunto de três partes, que compreende não somente os estudos a fim de estabelecê-la, como também, as ferramentas de software que trarão suporte ao seu emprego. São elas:

1. A definição e demonstração de um modelo estrutural e uma linguagem de componentes para modelagem de jogos e gêneros. Esta parte estabelece a base conceitual para as demais e compreende o objeto de estudos desta tese.
2. A definição e implementação de uma base de dados para gerenciamento de componentes e modelos de jogos, armazenando o conhecimento de design e possibilitando o emprego de técnicas de descoberta de conhecimento.
3. A definição e implementação de um Ambiente de Design de Jogos, que faça interface com a base de conhecimentos e permita a prototipação de jogos a partir da modelagem por componentes.

1.6 Metodologia

O princípio que deu origem a este trabalho, de que jogos podem ser compreendidos como a junção de partes menores que representam seus aspectos, surgiu a partir das experiências anteriores do autor como usuário e desenvolvedor de jogos. A abordagem de componentes apresentada nesta tese se baseia na observação dos jogos e do conhecimento de design embutido nestes. Foram as experimentações em projetos de jogos realizadas em atividades profissionais e acadêmicas do autor que motivaram este trabalho. Dessa forma, pode-se afirmar que a concepção da abordagem de componentes fundamentou-se em construção empírica e em estudos teóricos. Nesse sentido, surgiu de experiências prévias do autor como usuário e desenvolvedor de jogos, e foi delineado com base no levantamento do estado da arte das ferramentas e métodos de design.

Figura 3 – Ciclo iterativo de execução do trabalho e construção da abordagem de Componentes de Jogos.

O processo utilizado durante a realização deste trabalho pode ser resumido em três etapas iterativas: concepção, construção e aplicação (Figura 3). A metodologia empregada teve como alicerces a observação e a experimentação:

- A **concepção** da abordagem foi baseada em experimentações prévias do autor no desenvolvimento e uso de jogos;
- A **construção** da abordagem foi baseada em experimentações de ideias, tanto na tentativa de analisar jogos existentes sob a ótica de componentes de design, quanto ao conceber novos projetos, ou mesmo, em exemplos para disciplinas previamente ministradas. Posteriormente, foi delineada por estudos bibliográficos;
- A **aplicação** da abordagem foi baseada na observação de componentes em jogos existentes e sua experimentação em projetos de teste. Estes compreendem tanto aqueles construídos especificamente para demonstrá-la, quanto projetos menores para outros fins.

Durante a realização deste trabalho o autor teve a oportunidade de ministrar disciplinas de desenvolvimento de jogos na instituição em que atua. Além disso, trabalhou em projetos acadêmicos de desenvolvimento de jogos simples. Ambas atuações contribuíram para o amadurecimento da abordagem de componentes, à medida que pode ser aplicada informalmente em tais cenários: na produção independente de protótipos e no processo de treinamento de aprendizes sem conhecimento prévio na área (ver seção 5.3).

1.7 Contribuições

A principal contribuição deste trabalho é apresentar uma perspectiva estrutura de design, com qual os jogos podem ser pensados, analisados, registrados e projetados por meio de suas partes formadoras. Para tanto, a abordagem documentada estabelece elementos estruturantes para os conceitos dos jogos, intitulados neste trabalho por componentes. Essa abordagem possui similaridades com o padrão arquitetural de software de Sistemas de Entidades e Componentes à medida que ambos empregam a composição como uma ferramenta de

definição do comportamento das entidades participantes da cena de um jogo.

Os componentes de jogos definem uma linguagem estruturada que favorece a constituição de um vocabulário comum pelo qual designers podem registrar e comunicar suas ideias de forma padronizada. Dessa forma, a abordagem pode contribuir para a constituição de uma base de conhecimentos universal na área, auxiliando não somente seus praticantes, como também a transferência de conhecimentos entre gerações de designers. Nesse sentido, acredita-se que existe um conhecimento de design embutido nos jogos já produzidos que pode ser resgatado e estruturado pelos dos componentes que empregam.

Diferentemente do modo corrente de pensar e documentar o design de jogos, a compreensão de que os jogos podem ser tratados como uma associação de partes estruturadas admite a aplicação de análises em sua composição, permitindo a descoberta de conhecimentos de design. Nesse sentido, a abordagem de componentes define a base conceitual para a construção de um Ambiente Design de Jogos, uma ferramenta computacional que permita ao designer pesquisar, analisar, projetar, construir protótipos e experimentar o resultado de suas ideias.

1.8 Organização do trabalho

Este capítulo apresentou uma visão geral e introdutória dos diversos aspectos e questões envolvidas na fundamentação e na elaboração deste trabalho. O restante do documento está organizado como segue. O Capítulo 2 apresenta um levantamento bibliográfico acerca dos trabalhos relacionados a este projeto a fim de constituir sua fundamentação na literatura. Nessa seção são abordados os pontos fundamentais da área do design de jogos, as ferramentas em uso corrente e as principais propostas de ferramentas alternativas e complementares de design. Em especial, ela apresenta o curso de trabalhos publicados que tratam da falta de ferramentas e métodos de design e que propõem abordagens a fim de minimizar tal problema. Além disso, discute as contribuições destes trabalhos e de que forma eles se alinham com a abordagem de componentes de design.

O Capítulo 3 apresenta em detalhes a abordagem de Design de Jogos Baseada em Componentes. São apresentados conceitos e discutidos pelo uso de exemplos com aspectos de jogos conhecidos. A demonstração de aplicação da abordagem está documentada no Capítulo 4, que contempla as narrativas de design de dois protótipos de jogos, construídos especificamente para este trabalho. O texto do capítulo discute o processo de construção dos protótipos, partindo da análise de componentes de jogos populares e apresenta o design por composição dos protótipos. Em vias de complementar o processo de demonstração, possibilitando a experimentação dos conceitos empregados nos jogos, protótipos jogáveis foram implementados e disponibilizados. Ao final, o Capítulo 5 apresenta considerações finais e possibilidades de trabalhos futuros.

Capítulo 2 - Fundamentação

Neste capítulo é apresentada uma discussão acerca de três pontos chave para a contextualização deste trabalho:

- Os aspectos fundamentais de design de jogos e seus instrumentos correntes (seção 2.1);
- As fronteiras da área e as discussões que nortearam os trabalhos de novas abordagens e instrumentos de design de jogos (seção 2.2);
- A abordagem de componentes e sua relação com trabalhos similares (seção 2.3).

2.1 Design de Jogos: Ofício, Ferramentas e Fronteiras

Desde o surgimento do entretenimento digital, a criação de jogos tem sido um processo intrinsecamente fundamentado nas habilidades criativas do designer de jogos (Neil, 2012). Diversão não é um aspecto garantidamente alcançável por meio de um estrito processo formal e uma quantia considerável de recursos financeiros. Altos orçamentos não são uma certeza de bons jogos (Daniels, 2015). Estúdios de grande porte encerraram suas atividades nos últimos anos (Campbell, 2012; Dyer, 2011), ao passo que projetos independentes, usualmente de baixo orçamento e com foco em *gameplay*, tem apresentando um forte crescimento (Morris, 2015; Reynolds, 2015). Por outro lado, embora não haja garantias de que bons jogos possam ser concebidos por meio de um processo formalizado repetível, o emprego de padrões estruturados e uma base de conhecimentos unificada de design podem trazer benefícios a concepção de jogos.

No processo de concepção, a inspiração para novos jogos é tipicamente influenciada por outras mídias, como livros, filmes e músicas, bem como, por elementos do cotidiano e da cultura popular. Além disso, jogos existentes tipicamente servem de fundação para novas ideias. Church (1999) enfatiza essa ideia ao afirmar que nenhum designer de jogos "trabalha no vácuo". "Como designers, nós construímos nossas ideias sobre nossas experiências e as ideias passadas de outros" (Church, 1999, p.1). Nesse sentido, designers usualmente experimentam jogos existentes na busca de aspectos que possam contribuir com seus projetos. Da forma similar, experimentações são constantemente realizadas sobre o jogo em projeto a fim de se analisar as características positivas e negativas da experiência de *gameplay* que emerge dos aspectos empregados. De um modo geral, a experimentação apresenta-se como parte essencial do processo de design de jogos, que pode ser sumarizado em três etapas: design, documentação e prototipação.

2.1.1 O Documento de Design

O principal artefato produzido pelo designer de jogos é o Documento de Design (GDD). Ele serve de local definitivo para registrar todos os detalhes do conceito do jogo e comunicar a visão do designer ao time de desenvolvimento (Rouse, 2000). Comumente conhecido como uma

"bíblia" para o design de um projeto, ele age posteriormente como guia para a etapa de Produção no processo de desenvolvimento do jogo (ver seção 1.1). Por essa razão, é considerado por muitos profissionais como o método corrente de design de jogos (Kreimeier, 2003). Nesse sentido, entende-se que escrever o documento é fazer o design. De forma prática, não existe de fato um método sistematizado de design de jogos, mas sim, um documento que registra o planejamento do jogo. Em tese, tal documento deveria servir de fonte de informação única para os demais integrantes da equipe de desenvolvimento. No entanto, diversos autores têm apresentado questionamentos sobre a prática do GDD (Neil, 2012).

Considerado o artefato definitivo e fundamental de design de jogos, o GDD é alvo de diversas discussões (Neil, 2012). Esforços têm sido despendidos a fim de estabelecer formatos e conteúdos estruturados e padronizados, mas nenhum resultado significativo foi alcançado. Embora alguns autores advoguem o uso do documento em sua forma mais extensa, outros defendem o oposto (Kreimeier, 2003). Dormans (2012) afirma que o principal inibidor da criação de uma metodologia universal de design é a falta de padrões nos documentos de design. Kreimeier (2003) relata que o documento frequentemente torna-se muito extenso e os integrantes dos times sentem-se pouco motivados a lê-lo, sendo raramente utilizado em estágios avançados do projeto e servindo apenas para fins contratuais. Para ele, o tamanho e o formato do documento são fatores que levam a tal prática.

Diversos autores concordam que, em relação à produtividade da equipe, o GDD não evoluiu da mesma forma que outras ferramentas de desenvolvimento, tais como motores de jogos e modeladores 3D. Para eles, o documento é ainda usado por razões culturais e contratuais. Costykian (1994) advoga que, mesmo com o emprego de recursos visuais, tais como sketches (ilustrações conceituais) e *storyboards* (sequências de eventos ilustradas), o GDD não é suficiente para descrever um jogo de forma produtiva. Ademais, a natureza estática do documento é frequentemente destacada como um problema. Mudanças no design de um jogo são comuns durante o processo de desenvolvimento devido à natureza idiossincrática da concepção de jogos. Contudo, o estilo de documentação massiva do GDD torna sua atualização improdutiva à medida que o projeto progride (Dormans, 2012). Para tratar dessa questão, Demachy (2003) sugere o uso de documentação "leve", enfatizando o emprego de diagramas no lugar de textos no intuito de melhorar a responsividade da documentação a mudanças no design. Ele então discute a aplicação de princípios de Programação Extrema (XP), um método ágil de desenvolvimento de software (Beck, 2004), na produção de jogos. Keith (2010) também aborda o emprego de métodos ágeis para o mesmo fim.

O emprego de documentação sintética de design tem sido um importante objeto de discussões e estudos por designers. Librande (2010), durante uma apresentação na Conferência de Desenvolvedores de Jogos de 2010 (GDC – *Game Developers Conference*), enfatizou que pessoas envolvidas na produção de um jogo apresentam pouca motivação em consultar o documento de design. Para ele, linguagens visuais são mais expressivas e compactas quando comparadas a descrições textuais de aspectos do jogo. Em sua apresentação, ele discute o emprego de tais linguagens em suas atividades como designer. Mais recentemente, Cerny

(2013) também apresentou relatos de sua atuação profissional no GameLab 2013, narrando a experiência de transição do uso de documentos de design extensos e imutáveis para uma documentação mais objetiva, leve e ágil. Na academia, Neil (2012) e Kreimeir (2003) discorrem sobre diversas tentativas de definição de abordagens mais estruturadas e sintéticas para a documentação de design.

2.1.2 Experimentação e a Prototipação de Jogos

A construção de protótipos jogáveis tem papel fundamental durante o desenvolvimento de um jogo (Fullerton et al., 2004). Se por um lado, as atividades do designer são centradas em projetar o sistema que dá vida ao jogo – regras, enredo, mecanismos de interação e mundo –, por outro, ele fundamentalmente persegue as características estéticas: as respostas emocionais que emergem da interação do jogador com o software (LeBlanc, 2004). É a estética que constrói o conceito positivo ou negativo de um jogo. No entanto, não é possível determiná-la por meio da documentação de design: é necessário realizar experimentação. A natureza dinâmica e interativa dos jogos torna inviável a avaliação pura e simples de conceitos registrados estaticamente no GDD. Para tanto, usa-se a prototipação em software.

Projetos de desenvolvimento de jogos fazem uso frequente da prototipação. Protótipos jogáveis são usualmente criados durante estágios preliminares do projeto, após a conclusão de parte significativa do GDD. Eles contêm versões iniciais das características fundamentais da interação do jogo, demonstrando um intenso foco em *gameplay* e desconsiderando a apresentação artística. São comumente utilizados como prova de conceitos e como ambiente de experimentações, permitindo avaliar e evoluir o *gameplay* projetado no GDD (Salen & Zimmerman, 2003). Nesse sentido, LeBlanc (2004) afirma que os jogos são sistemas complexos e manifestam comportamentos que não podem ser previstos pela leitura dos documentos que os descrevem.

Designers de jogos são unânimes quanto à valia da experimentação por meio da prototipação, tida como parte crítica do processo de desenvolvimento de jogos e considerada o único meio confiável de verificar a qualidade do design (Neil, 2012). Contudo, designers não estão geralmente habilitados para construir os protótipos de software por si. Embora construtores rápidos de jogos (*game makers*), populares entre iniciantes e entusiastas, sejam eventualmente usados para criação de protótipos, eles são considerados muito restritivos. Os mecanismos facilitadores que empregam tendem a restringir as opções do designer a um conjunto limitado de construções preconcebidas para hobistas, o que sacrifica a liberdade inerente ao processo criativo. Outro recurso de prototipação rápida conhecido são os chamados "protótipos analógicos". Construídos como jogos de tabuleiro, eles provêm uma representação simples de algumas regras do jogo, abordando especificamente aquelas baseadas em operações numéricas. Geralmente podem ser elaborados pelos designers por meio de recursos acessíveis, como papel, cola e tesoura. Contudo, embora sejam úteis na análise do balanço numérico de determinados aspectos, tais como o balanço entre ataque e defesa de personagens em jogos de

estratégia (*RTS – Real-Time Strategy*[25]) ou sistemas de batalha de RPGs (*Role-Playing Games*[26]), são ineficazes para jogos com mecanismos de interação fundamentalmente baseados em ações de tempo real de controle direto do jogador, como Super Mario e Doom (Sigman, 2005).

Considerando a inexistência de ambientes de prototipação rápida específicos para designers, a construção dos protótipos de software é uma tarefa comumente delegada à equipe de desenvolvimento. Nesse sentido, engenheiros de software e artistas gráficos os produzem sob a orientação dos designers. No entanto, os protótipos são construídos tardiamente em relação à concepção do jogo, uma vez que são iniciados após uma parte substancial do documento de design ter sido concluída. Esse atraso cria uma lacuna entre a definição dos conceitos do jogo, que são concebidos durante a escrita do documento, e o processo de experimentação que os atesta (Neil, 2012). Nesse sentido, a prototipação acaba tornando-se um processo um custoso e tardio, à medida que infere na alocação de pessoal especializado e está distante do controle direto do designer durante a concepção do jogo.

A prototipação permite avaliar a estética dos aspectos projetados no design de jogo e promove a realização de mudanças corretivas ou evolutivas tão logo sejam percebidas. Por outro lado, o custo de sincronização do documento de design para refletir tais modificações pode ser bastante alto, implicando em considerável retrabalho, ou mesmo, no completo reinício do design. Nesse sentido, aspectos-chave do *gameplay* podem já estar diretamente vinculados àqueles que serão removidos ou modificados. O emprego de um modelo de documentação mais sintética e estruturada, que possa estar sincronizada a uma ferramenta de prototipação rápida acessível ao designer de jogos, pode diminuir consideravelmente o custo de mudanças necessárias ao jogo durante sua concepção. Sem uma lacuna entre a concepção e o teste de conceitos, o design do jogo poderia evoluir concomitantemente ao GDD, que refletiria o resultado de experimentações de *gameplay* desde o início de sua produção. No entanto, o uso de documentação massivamente textual no registro das ideais do jogo e a inexistência de padrões e formalização no design dificultam a concretização de tal ferramenta. Neil (2012) e Kreimeier (2003) apresentam uma síntese analítica de diversos trabalhos publicados com o objetivo de contribuir para o estabelecimento de instrumentos e abordagens estruturadas de design de jogos. Um relato cronológico desses trabalhos é abordado na próxima seção. O final do capítulo apresenta uma discussão da abordagem de design de jogos baseado em componentes frente aos trabalhos relacionados.

2.2 Em Busca de Novas Abordagens

Designers de jogos e pesquisadores desenvolveram um discurso acerca do problema tratado nesta tese, a falta de ferramentas estruturadas e padronizadas de design. Eles consideram a principal ferramenta corrente de design – o documento de design – como muito restritiva, criando uma barreira à evolução da área como um tudo (Neil, 2012). Em 1994, Costikyan publicou uma discussão acerca da necessidade por maior formalismo no design de

[25] http://tvtropes.org/pmwiki/pmwiki.php/Main/RealTimeStrategy
[26] http://tvtropes.org/pmwiki/pmwiki.php/Main/RolePlayingGame

jogos, sugerindo a adoção de vocabulário comum de conceitos de design como uma primeira solução à falta de padronização na área. Para ele, os designers deveriam possuir uma forma de analisar os jogos, compreendê-los e identificar os elementos que os tornam bons ou ruins. Outros autores que, direta ou indiretamente ecoaram o discurso de Costikyan, fazem referência à mesma ideia. Church (1999) menciona a falta de um vocabulário comum de design como o principal inibidor da evolução dos métodos de design de jogos. Mais tarde, Fullerton (2004) expressa uma opinião similar, retomando a necessidade por um vocabulário comum de design. Nesta subseção são apresentados e discutidos os principais trabalhos que trataram do problema em foco – a falta de ferramentas e abordagens estruturadas de design – e que propuseram instrumentais conceituais de design a fim de mitigá-lo. Com o objetivo de prover uma visão geral desses trabalhos e de suas inter-relações, foi criado um mapa visual que sintetiza os trabalhos nos quais esta tese se fundamenta. Este mapa está disposto na Figura 4 e foi publicado em Almeida & Corrêa da Silva (2013). Ele organiza os trabalhos de propostas de abordagens e ferramentas de design em dois grandes grupos: Vocabulários Compartilhados e Linguagens Visuais.

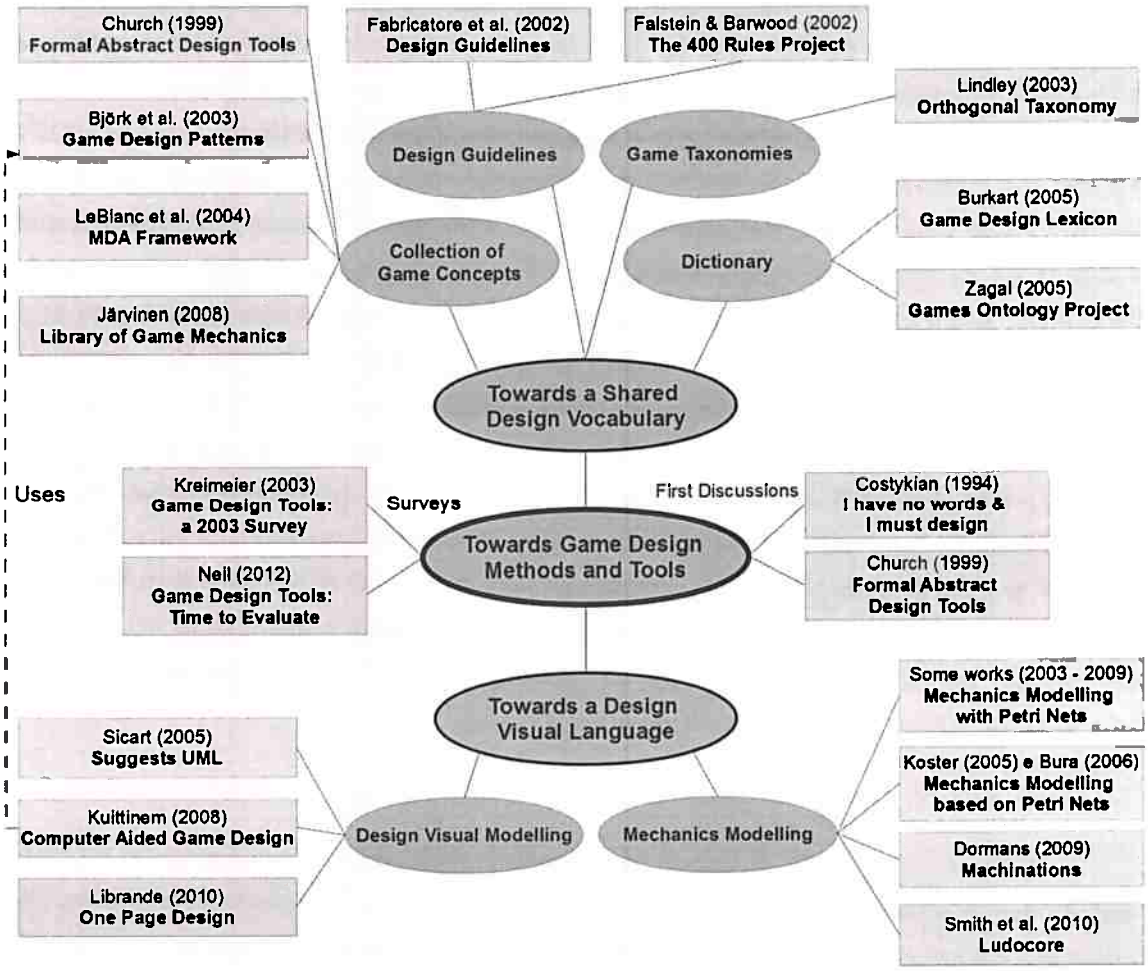

Figura 4 – Mapa de métodos e ferramentas de design de jogos (Almeida & Corrêa da Silva, 2013).

2.2.1 Vocabulários Compartilhados de Design

Diversos pesquisadores e designers consideram que um vocabulário unificado e compartilhado pode trazer benefícios significativos à área (Neil, 2012). Com esse objetivo eles discutiram e propuseram diferentes abordagens foram enquadradas em quatro grupos.

- Coleções de Conceitos de Design;
- Diretrizes de Design;
- Dicionários de Termos;
- Taxonomias de Jogos.

Os quatro tipos de abordagens identificadas com o objetivo de estabelecer alguma forma de vocabulário de design são discutidos nesta seção em ordem de importância para este trabalho e estão dispostos na parte superior do mapa da Figura 4.

2.2.1.1 Coleções de Conceitos de Design

A primeira tentativa de estabelecer uma ferramenta de design de jogos baseada em uma coleção de conceitos de design foi a FADT – Ferramentas de Design Abstratas Formais – publicada por Church em 1999. Church tinha como propósito possibilitar aos designers dissecar um jogo, identificar e separar suas partes formadoras, compreender como essas partes se relacionam e balanceiam e, dessa forma, analisar quais beneficiam ou prejudicam determinados jogos ou gêneros. Segundo ele, designers constroem suas ideias a partir de suas experiências e do trabalho de outros designers, mas não há uma forma de identificá-las, estruturá-las e documentá-las. Church foi, de fato, o principal autor a apresentar a noção de estruturação e componentização de design. Contudo, sua FADT não satisfez o seu objetivo, uma vez que representa um pequeno conjunto simplista de conceitos com alto nível de abstração. Nele, cada FADT é definida por nome e descrição, não havendo documentação de inter-relacionamentos. No total, somente 13 FADTs foram apresentadas em seu trabalho. Embora a proposta de Church não tenha se concretizado como uma ferramenta de design, certamente serviu de inspiração a outros autores.

Buscando uma descrição estrutura, Kreimeir (2002) sugeriu a adoção de uma abordagem similar aos Padrões de Projeto (*Design Patterns*) de Gamma et al. (1994) para documentação de conceitos recorrentes de design encontrados nos jogos. Björk, Lundgren e Holopainen (2003) seguiram a mesma direção e propuseram um modelo de Padrões de Design de Jogos (*Game Design Patterns*), que, embora inspirados pelo trabalho de Gamma, não seguem a abordagem de pares "problema-solução" ao estruturar os conceitos de design. A estrutura de documentação de cada padrão é definida conforme os Padrões de Projeto, sendo compreendida por nome, definição, citações de jogos em que está presente, questões envolvidas em sua aplicação, aspectos narrativos, consequências de uso e relacionamentos com outros padrões. Segundo os autores, o objetivo dos Padrões de Design de Jogos era fornecer uma ferramenta de análise design. Mais estruturado que as abordagens que o precederam, o projeto dos Padrões

de Design de Jogos teve quase 400 padrões documentos em um wiki[27] de forma colaborativa.

Dentre os trabalhos publicados, os Padrões de Design de Jogos representam uma evolução do modelo FADT e a contribuição mais significativa na tentativa de criar um banco de dados de conceitos de design. Em tese, sua estrutura inspirada nos Padrões de Projeto permitiria destacar aspectos relevantes de cada padrão e suas inter-relações. No entanto, o projeto tem pontos negativos que desencorajam o seu uso. A estrutura de documentação não é fielmente obedecida ao registrar os padrões. É comum encontrar documentação bastante incompleta, confusa e contraditória em padrões, com divergências entre nome, definição e exemplos de uso. Não há correspondência suficiente entre os padrões e os jogos ou gêneros que os utilizam de forma a permitir uma análise centrada em jogos. Com isso, torna-se difícil realizar uma pesquisa que permita inferir quais padrões levaram determinados jogos ao sucesso ou fracasso, ou mesmo, aqueles que estão relacionados a um determinado perfil de usuários. Outra dificuldade encontrada está relacionada a navegação nos padrões. Sem uma relação clara entre jogos e padrões, é preciso lê-los um a um para conhecê-los, ao invés de partir de agrupamentos relacionados a jogos e gêneros. Uma deficiência marcante é a falta de modelos gráficos que facilitem a compreensão dos padrões, bem como, a visualização das relações e da hierarquia entre padrões, jogos e gêneros. O resultado final apresenta-se como uma coleção extensa e esparsa de conceitos de design genéricos e pouco coesos, de difícil aplicação em projetos reais.

Um ano depois, os designers e desenvolvedores LeBlanc, Hunicke e Zubek (2004) teorizaram o MDA, um framework[28] para analisar jogos sob três aspectos: mecânicas[29], dinâmicas e estética. Segundo os autores, as mecânicas descrevem as regras estáticas do sistema do jogo registradas no documento de design; as dinâmicas representam o comportamento em tempo de execução das mecânicas em uma sessão de jogo; e a estética compreende as respostas emocionais desejadas dos jogadores ao utilizarem o jogo. Influenciado pelas FADTs de Church, o MDA enfatiza a importância de se considerar tanto a perspectiva do designer, o produtor das mecânicas, quanto a do jogador, que consume o jogo pela estética. Pode-se dizer que o MDA define uma abordagem de design de jogos orientado a estética. Nela, a assimilação da estética ocorre pela perspectiva do jogador e, portanto, é fundamentalmente necessário utilizar os jogos para obtê-la e compreendê-la. Nesse sentido, torna-se evidente a importância do emprego de protótipos de software para experimentação e análise da estética gerada pelos conceitos empregados. Ademais, a busca pela estética ratifica a prática da análise de jogos existentes em busca de partes que possam ser empregadas em outros projetos, de forma que produzam a estética desejada. Com isso, corrobora com a premissa deste trabalho, de que existe um conhecimento de design embutido nos jogos já produzidos por meio dos

---

[27] http://protagonist.sics.chalmers.se:1337/mediawiki-1.22.0/index.php/Category:Patterns

[28] Um framework é um ambiente de software reusável que provê um conjunto de funcionalidades para facilitar o desenvolvimento de aplicações, produtos e soluções. O framework define um esqueleto para criação de software, ditando o fluxo de controle da aplicação (Riehle, 2000).

[29] O termo "mecânica" é comumente empregado na comunidade de desenvolvimento de jogos. Alguns o utilizam como sinônimo de regra. Outros o definem como o conjunto de detalhes necessários à implementação e execução de uma regra dentro do jogo, bem como, algo com o qual o jogador interage e que ajuda a compor o *gameplay* do jogo (Dormans, 2012).

componentes que empregam. No contexto dos demais trabalhos publicados, o MDA formaliza uma estrutura em três camadas que age de forma transversal às abordagens de coleções de conceitos de design. Ele é citado em alguns desses trabalhos, sendo tratado como uma estrutura que organiza os conceitos documentados sob os três aspectos do MDA: mecânicas, dinâmicas e estética.

Ainda no contexto de coleções de conceitos de design, Järvinem (2008) apresentou um estudo acadêmico sobre teorias e métodos de design de jogos e discutiu a criação de uma biblioteca de mecânicas de jogos a um nível altamente teórico e pouco aplicado. Atualmente, os websites Giant Bomb[30] e TV Tropes[31] representam os esforços ativos mais significativos de se manter uma base de conhecimentos de conceitos de design. Embora restrinjam-se a uma estrutura bastante informal, destinada e mantida por usuários finais, eles constituem recursos promissores. Ambos apresentam não somente as descrições e ilustrações de conceitos de design, como também, discutem suas aplicações em jogos conhecidos. Ademais, essas bases de dados de design confirmam o interesse em se catalogar aspectos de jogos. Elas serão por vezes indicadas como referência para aspectos de design discutidos neste trabalho.

No contexto teórico, as abordagens citadas trazem contribuições significativas à medida que evoluem a questão inicialmente levantada por Costikyan em 1994, de que os designers deveriam possuir uma forma de analisar os jogos, compreendê-los e identificar os elementos que os tornam bons ou ruins. No entanto, como ferramentas de aplicação concreta, elas mostram uma evidente falta de maturidade e suporte computacional, tanto para experimentação quanto para adoção em cenários do mundo real.

2.2.1.2 Diretrizes de Design

Diferentemente da tentativa de se construir uma coleção de conceitos de design, Falstein & Barwood (2002), iniciaram o Projeto das 400 Regras, que teve como objetivo identificar, registrar e compartilhar experiências práticas de designers na forma de diretrizes, indicando direções a serem adotadas ou evitadas em um projeto de jogos. Do plano inicial de 400 regras, apenas 112 foram documentadas em uma planilha disponível no site do projeto[32] por meio da colaboração de vários designers. Exemplos dessas diretrizes são dispostas abaixo:

- "Incorpore Tutoriais ao Gameplay": Integre os tutoriais no jogo ao invés de uma seção separada.
- "Deixe o Jogador Desligar o Jogo": O jogador deve poder salvar do jogo e desligá-lo a qualquer momento, sem ser penalizado por isso.
- "Tenha Diversão no Primeiro Minuto": Jogos casuais devem prover diversão logo no início. Em jogos de caixa, esta regra é dispensável, embora também importante.

---

[30] http://www.giantbomb.com/concepts/ (visitado em 07/12/2015)
[31] http://tvtropes.org/pmwiki/pmwiki.php/Main/VideoGames (visitado em 03/02/2016)
[32] http://www.finitearts.com/Pages/400page.html (visitado em 07/12/2015)

Assim como no projeto dos Padrões de Design de Jogos de Björk et al. (2003), este trabalho demonstrou uma forte intenção em prover um instrumento de uso prático. Em uma abordagem similar, Fabricatore, Nussbaum e Rosas (Fabricatore et al., 2002) apresentaram um conjunto de diretrizes de design a partir de uma análise da influência de mecanismos de *gameplay* na motivação de jogadores. Independente da forma de obtenção, a documentação de diretrizes de design pode auxiliar na transferência de conhecimento entre gerações de profissionais, especialmente durante o treinamento de novos designers. Ademais, essas diretrizes possuem uma natureza intrinsecamente prática, considerando que retratam o aprendizado concreto do ofício do designer. Contudo, diretrizes por si só não constituem um instrumento de análise ou criação de jogos.

2.2.1.3 Dicionários de Termos

Em 2005, o trabalho publicado por Zagal, Mateas, Fernández-Vara, Hochhalter e Lichti descreve o Projeto de Ontologia de Jogos, visando criar um dicionário compartilhado de design a fim de prover definições inequívocas para termos de design. Kreimeier (2003) cita também a iniciativa não realizada "*Game Design Lexicon*", que teria como objetivo disponibilizar uma base de dados que armazenasse um dicionário a fim de ser embutidos em documentos de design via linguagens de marcação, tais como XML.

O Projeto de Ontologias de Zagal et al. (2005) segue um formato similar ao Projeto de Padrões de Design de Björk et al. (2003), no qual os termos estão publicados em um wiki[33]. No total estão documentadas 178 definições de termos de design e jogos, organizados sob quatro categorias: interface, regras, manipulação de entidades e objetivos. Definições de termos como "*Score*", "*Bonus Stage*" e "*Level*" estão presentes, sendo registrados por meio do nome, definição, exemplos de uso e relações com demais termos. Segundo a definição contida no site, o projeto é um "framework para descrever, analisar e estudar jogos, provendo uma ontologia que identifica os elementos estruturais importantes dos jogos". No entanto, o resultado documentado apresenta-se essencialmente como um dicionário de termos. Kreimeier (2003) afirma que dicionários são complementos necessários a qualquer método ou instrumento conceitual, pois constituem um primeiro passo para o estabelecimento de um vocabulário padronizado na área. Dessa forma, podem ser tomados como uma camada de abstração mais baixa, servindo de fundação a abordagens de design, mas não se tornando uma em si.

2.2.1.4 Taxonomias de Jogos

Uma parte do vocabulário frequentemente praticado na indústria, academia e mídia especializada resulta da taxonomia empregada em jogos (Järvinen, 2008). Idade, público-alvo, propósito e gênero são exemplos de critérios que levam a diferentes classificações em jogos. Desses, a classificação em gêneros é provavelmente o exemplo mais concreto de aplicação prática de um vocabulário. Os gêneros organizam os jogos em grupos hierárquicos segundo os principais aspectos de seu *gameplay*, ao contrário de filmes e livros, nos quais adota-se uma

[33] http://www.gameontology.com/index.php/Main_Page (último acesso em 04/12/2015)

categorização baseada em temas ou enredos (Adams, 2009). Como exemplo, o gênero "*First-Person Shooter*" refere-se a jogos nos quais o jogador interage por meio de uma perspectiva em primeira pessoa e envolve fundamentalmente a ação de disparar com armas de fogo. Hierarquicamente, os jogos do tipo "*First-Person Shooter*" pertencem ao grupo dos "*Shooters*", que por sua vez está enquadrado sob o gênero "*Action*" (Thorn, 2013). A compreensão dos jogos pelos gêneros que os definem não está limitada a designers. Quando um novo jogo é anunciado na mídia especializada, o gênero está comumente entre as primeiras informações divulgadas. Isso deve-se ao fato de que os princípios básicos que definem um gênero são compreensíveis tanto por jornalistas quanto consumidores.

A classificação em gêneros ajuda não somente a definir os elementos típicos encontrados em cada gênero, mas também, a compreender como a mistura entre gêneros ocorre. Segundo Kreimeier (2003), alguns autores sugerem que nem todos os jogos podem ser enquadrados em um sistema hierárquico de categorias e subcategorias. Entre eles, Lindley (2003) propôs uma Taxonomia Ortogonal para jogos, que define um sistema espacial de características no qual eles são posicionais de acordo com sua proximidade a essas características (Figura 5). Em seu trabalho o autor apresenta espaços de uma, duas ou três dimensões, mapeando o grau de proximidade de jogos e gêneros específicos a algumas características, incluindo ludologia, narratologia e simulação. Dentro do contexto deste trabalho, o mapeamento multidimensional de Lindley pode ser futuramente empregado para exibir o grau de proximidade de componentes de jogos a determinadas características estéticas, de mercado ou preferências de usuários, a fim de se mapear quais componentes são desejáveis a projetos de jogos que caem sobre determinada classificação de gênero.

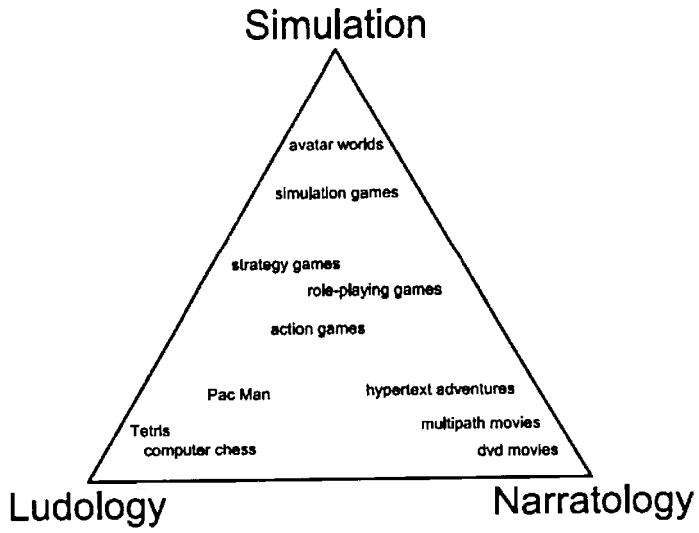

Figura 5 – Plano bidimensional de classificação ortogonal, exibindo a proximidade de jogos e gêneros a características de interesse (Lindley, 2003; obtido em Dormans, 2012).

2.2.2 Linguagens Visuais de Design

Embora de conteúdo predominantemente textual, designers de jogos comumente utilizam artefatos visuais como recursos auxiliares para descrever o conceito do jogo no documento de design (Neil, 2012). Como exemplo, eventos do jogo e comportamentos de personagens são comumente expressos por diagramas e *story boards*. Além deles, o emprego de ilustrações conceituais é uma prática amplamente usada a fim de transmitir as ideais do designer, permitindo uma pré-visualização do jogo antes de sua produção iniciar (Bates, 2004). Ademais, é comum que designers utilizem uma notação própria para expressas partes do design do jogo no qual estão trabalhando (Salen & Zimmerman 2003). Os esquemas de Design de Uma Página de Librande (2010) são um exemplo dessa prática.

Designers encontraram na comunicação visual um importante instrumento para transmitir sua visão do jogo aos demais membros da equipe de produção. Librande (2010) enfatiza que modelos visuais são mais sintéticos, naturalmente comunicativos e escalam melhor quando comparados a comunicação puramente textual. Para ele, a prática de construção de diagramas força o designer a pensar de forma mais objetiva, uma vez que é preciso extrair a essência do *gameplay* para poder representá-la em poucos elementos visuais. Na academia, outros autores sugeriram o uso de linguagens mais formalizadas existentes para a constituição dos diagramas de design, na tentativa de padronizar os elementos usados na constituição de tais mapas. Contudo, a área carece de uma linguagem visual padronizada e esforços tem sido despendidos a fim de mitigar esse problema (Neil, 2012).

Paralelamente às tentativas de desenvolver um vocabulário comum de design de jogos, a busca por linguagens visuais de design compreende o segundo grande grupo mapeada na Figura 4, localizado no segmento inferior do esquema. Duas ramificações de trabalhos foram identificadas: as propostas de linguagens visuais para descrever o conceito de jogos e as linguagens visuais para representar mecânicas ou regras de jogos.

2.2.2.1 Modelagem do Conceito do Jogo

Dentro das propostas de emprego de linguagens visuais no design de jogos, há um grupo de abordagens que tenta documentar o conceito do jogo, o design em si, utilizando alguma forma de notação visual. Algumas dessas tentativas tentam aplicar linguagens visuais advindas de outras áreas no design de jogos, enquanto outras, utilizam notações próprias. O uso de linguagens visuais para representar o design de jogos pode assemelhar-se à modelagem de requisitos e do projeto no desenvolvimento de software. Neil (2012) aponta que essa similaridade foi notada por alguns autores, que discutiram o emprego de UML para documentar o design. Dentre eles, Sicart (2008) apresenta uma perspectiva para o design de jogos apoiando-se sobre o conceito de orientação a objetos. Ele então sugere o uso de UML (Fowler, 2004) para modelagem de design, embora nenhum exemplo seja mostrado em seu trabalho. De forma similar, Demachy (2003) discorre sobre o emprego de *Extreme Programming* (Beck, 2004), um método ágil de desenvolvimento de software, na produção de jogos. Ele também se refere a UML como um possível instrumento de design para descrever os elementos do *gameplay*. No

entanto, nenhum exemplo prático de aplicação é apresentado em ambos trabalhos. Em geral, linguagens visuais são massivamente empregadas para facilitar e padronizar a comunicação no processo de desenvolvimento de software. Contudo, a aplicação de UML no design de jogos carece de estudos mais aprofundados, especialmente no que se refere a aplicação e demonstração em projetos de produção de jogos. Em outra vertente, outros trabalhos discutiram o emprego de linguagens visuais próprias para documentar o design de jogo.

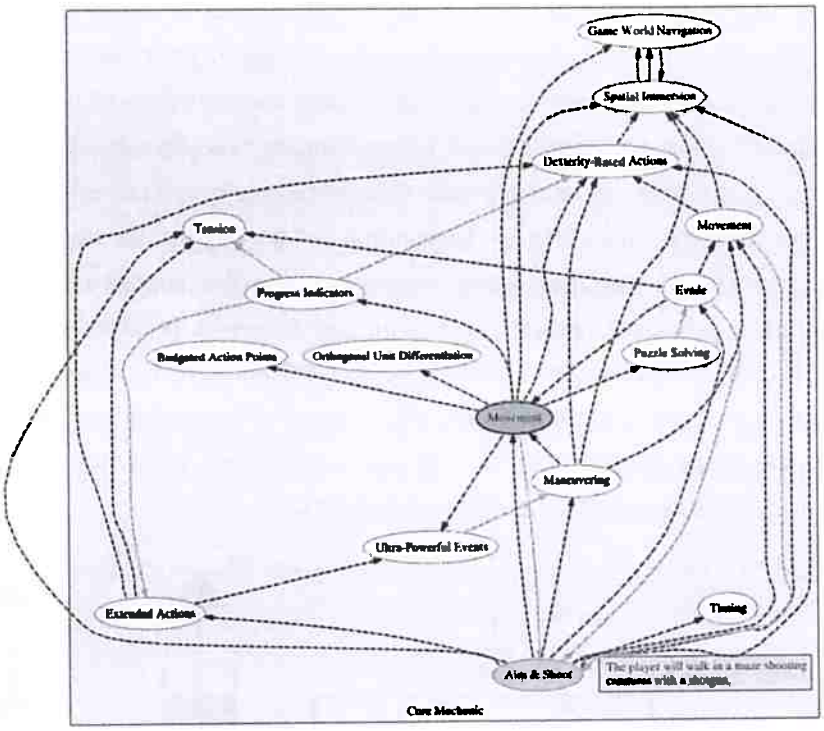

Figura 6 – Diagrama construído com o CAGE representando os comportamentos de Atirar e Mover (Kuittinen, 2008)

Em outro trabalho, Kuittinen (2008) teve como objetivo gerar modelos visuais baseados nos Padrões de Design de Jogos de Björk et al. (2003) empregados em um jogo. Ele apresentou o software CAGE – *Computer-Aided Game Design* –, um aplicativo simples de construção de diagramas. Seu objetivo era prover um instrumento de modelagem que permitisse criar um diagrama contendo os padrões de design empregados em um jogo (Figura 6). Segundo o autor, as descrições dos padrões empregados podem ser exportados para um documento de texto a fim de colocá-las no documento de design. Embora simples, o CAGE é uma das poucas iniciativas documentadas de se criar uma ferramenta computacional de design de jogos. Ele mostra-se interessante pelo fato de tentar aplicar um modelo conceitual de forma a integrá-lo ao método corrente na indústria, abordagem que também é objeto deste trabalho. Por outro lado, ele apresenta uma notação demasiadamente simplista que apenas mostra as ligações entre padrões, não tratando dos aspectos relacionados ao emprego específico de cada padrão no jogo. Como consequência, no exemplo apresentado na Figura 6 nota-se que, mesmo ao representar uma pequena parte de um comportamento bastante comum a jogos, o diagrama torna-se confuso, repleto de emaranhados de ligações entre padrões.

2.2.2.2 *Linguagens para Modelar Mecânicas de Jogos*

Em contraste com as tentativas de definir uma linguagem visual que permita a modelagem do design de jogos na forma de diagramas, abordagens acadêmicas têm focado em um nível de abstração mais baixo, tentando elaborar um vocabulário de construções lógicas que permita a representação de mecânicas de jogos. O emprego de Redes de Petri para esse propósito foi discutido por diversos autores, tais como Brom & Abonyi (2006), Araújo & Roque (2009) e Natkin & Vega (2004). Koster (2005) e Bura (2006) apresentaram variações de Redes de Petri a fim de acomodar customizações que julgaram necessárias para expressar as mecânicas de jogos. Exemplos de seus trabalhos estão apresentados nas modelagens dos jogos Damas e Vinte e Um (Figura 7). Embora demonstrem uma definição mais formal para a descrição de aspectos dos jogos, a curva de aprendizado necessária para compreender e utilizar tais linguagens é notavelmente mais alta se comparada a abordagens de documentação mais tradicionais e naturalmente intuitivas, como textos e ilustrações conceituais. Nesse sentido, Dormans (2012) nota que essas tentativas de empregar Redes de Petri sofrem de problemas similares à de outros tipos de diagramas, como aqueles contidos na UML. Usadas principalmente como linguagens de especificação de software, elas tornam-se inacessíveis aqueles que não são programadores. Ele ainda aponta que tais linguagens são muito genéricas para capturar as características peculiares necessárias ao domínio de jogos.

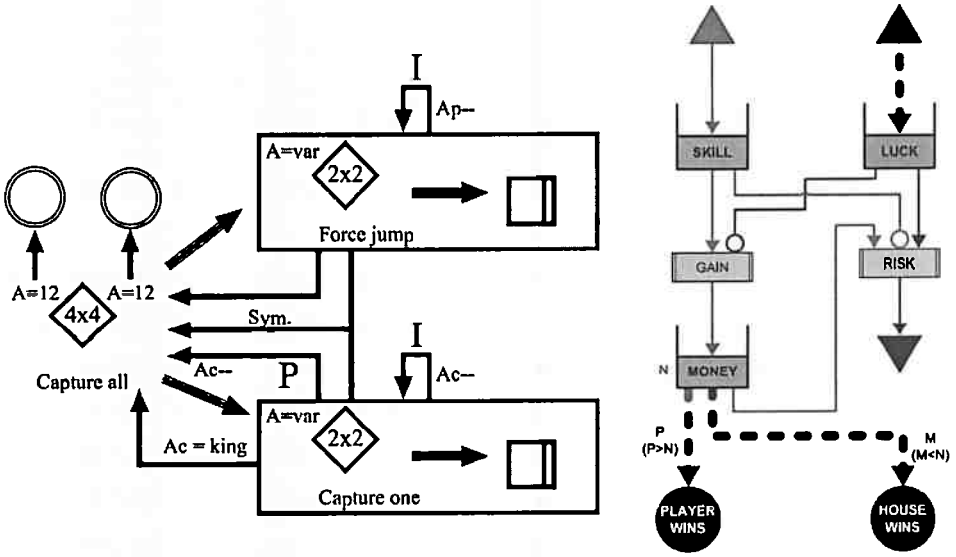

Figura 7 – Modelagens dos jogos Damas por Koster (2005), e Vinte e Um por Bura (2006) (obtidos em Dormans, 2012).

Dormans (2012) projetou e implementou a Machinations, uma ferramenta de software para especificar e testar mecânicas de jogos usando uma linguagem visual própria. Uma vez definido, o diagrama que representa as mecânicas do jogo pode ser executado por meio de uma simulação. Diferente das abordagens baseadas em Redes de Petri, Dormans (2012) considera que "*os diagramas devem mapear a economia interna que guiam o comportamento emergente nos jogos*". Nesse sentido, os diagramas do Machinations descrevem os jogos por meio do fluxo de recursos entre produtores e consumidores, bem como, pelo fluxo de informações que ocorre

entre seus elementos. Um exemplo de modelagem com Machinations é mostrado nas Figura 8. Ela contém o design de um jogo de estratégia de batalha elaborado por Kisslat (2013).

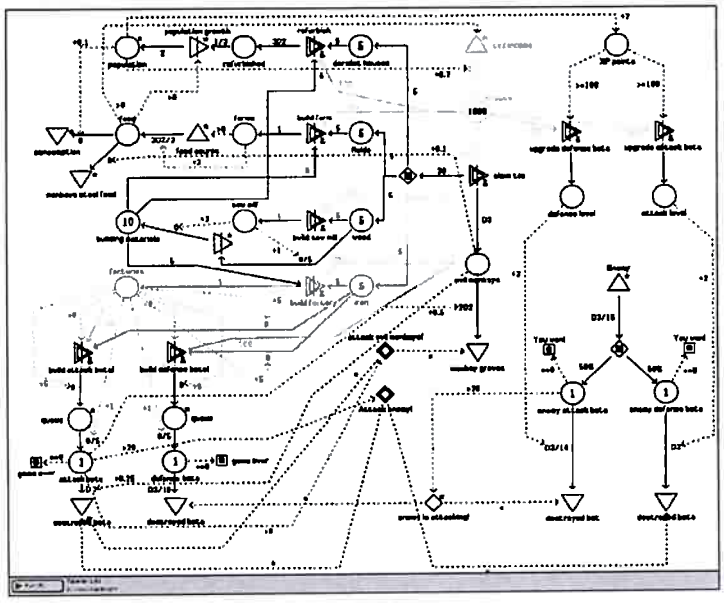

Figura 8 – Diagrama de um jogo de estratégia de batalhas (Kisslat, 2013).

Ludocore, outra ferramenta criada com um propósito similar, permite modelar o jogo como uma máquina de estados orientada a eventos e consequências. Construída e apresentada por Smith, Nelson e Mateas (Smith et al., 2010), ela empresa uma linguagem de definição textual ao invés de construtores visuais (Figura 9). Assim como a Machinations, a Ludocore permite a "execução" das mecânicas por um processo de simulação da máquina de estados definida pelo usuário.

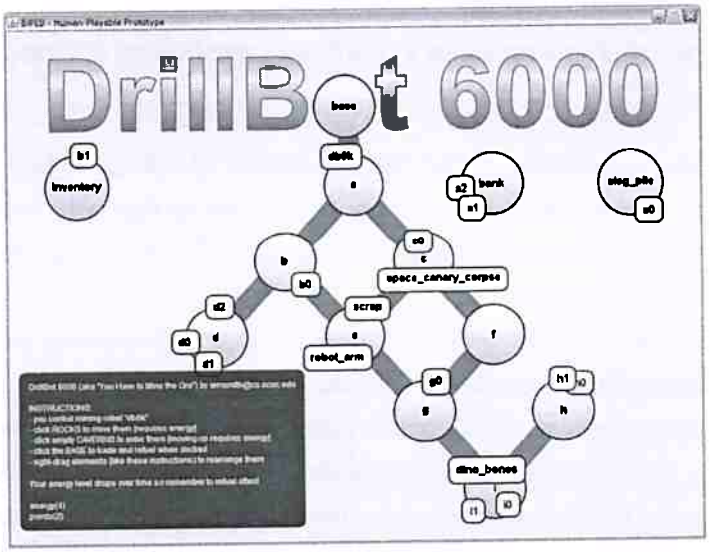

Figura 9 – Diagrama em execução na Ludocore (Smith et al., 2010)

As simulações presentes nas abordagens de Dormans (2012) e Smith et al. (2010) apresentam uma ideia interessante, mas delimitam-se a um ponto de vista bastante matemático. Nesse sentido, elas não representam a perspectiva do jogador, impedindo a experimentação dos aspectos estéticos do jogo. Jogos não são se restringem a uma simulação,

mas representam mundos virtuais nos quais seus usuários adentram para viver experiências interessantes. Por outro lado, Ludocore e Machinations assemelham-se funcionalmente aos protótipos analógicos. Dessa forma, podem auxiliar a tratar aspectos específicos de determinados gêneros, como por exemplo, no projeto e avaliação da economia de jogos de simulação, como Game Dev Tycoon[34] e Sim City[35].

De uma forma geral, nota-se que as linguagens utilizadas nas abordagens discutidas se apresentam pouco intuitivas. Assim como das tentativas anteriores de modelagem de mecânicas, que se fundamentaram em Redes de Petri, é difícil visualizar o conceito do jogo apenas olhando para os diagramas. Como discutido previamente, a modelagem restringe-se a representar aspectos numéricos do jogo, excluindo mecânicas que definem ações mecânicas de tempo real que estão presentes na grande maioria dos jogos. O problema dessas linguagens esbarra sua aplicabilidade como instrumento de comunicação da visão do designer ao time. Ao contrário de uma arte conceitual, que passa uma ideia clara de como se dará um certo momento do jogo, é muito difícil olhar para o diagrama e sequer entender que tipo de jogo ele descreve. O trabalho de Librande (2010), apresentado a seguir, documenta o resultado de sua experiência como designer e segue uma linha mais prática. Embora não possa ser caracterizado como uma tentativa de definição de linguagem formal, ele apresenta uma perspectiva realista das necessidades cotidianas dos designers em termos de documentação visual.

2.2.2.3 Designs de Uma Página

Ao lado de abordagens que surgiram de estudos acadêmicos, relatos de designers de jogos em conferências na área também corroboram com percepção de que existe uma necessidade por abordagens de documentação mais sintéticas e instantaneamente comunicativas. O designer Librande (2010), em sua apresentação na *Game Developers Conference* de 2010, fez um relato sobre a abordagem que utilizou em alguns dos projetos nos quais trabalhou.

O Design de Uma Página é um mapa sintético de design livremente composto de artefatos visuais e informações textuais curtas. O exemplo da Figura 10 mostra que é possível registrar um volume considerável de informações que cobrem diversos aspectos do jogo de forma sintética em um único local. Em especial, o exemplo mostra informações sobre o mundo do jogo, seus personagens e características, níveis do jogo, duração das batalhas e aventuras, sequências de ações e caminhos requeridas pelo jogador, dentro outras. Além disso, o autor mostra variações de pôsteres que criou para tratar das especificidades de outros projetos, bem como aqueles que possam necessitar de mais de um pôster ou que apresentar informações extras em documentos relacionados. Ainda na mesma apresentação, o autor também destaca os problemas encontrados na documentação tradicional de design, seja ela um documento, uma página web ou um wiki. Nesse sentido, ele destaca a dificuldade em atualizar o documento para manter sincronia com as ideias dos designers e o acesso moroso à informação, que desestimula

[34] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/GameDevTycoon
[35] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/SimCity

a sua leitura e favorece a comunicação verbal e informal. Segundo ele, registrar o design dos jogos em pôsteres torna-o acessível a toda equipe de desenvolvimento, garante que a equipe o acompanhará diariamente e encoraja a participação de todos na concepção do jogo, uma vez que o pôster é facilmente alterável de forma manual.

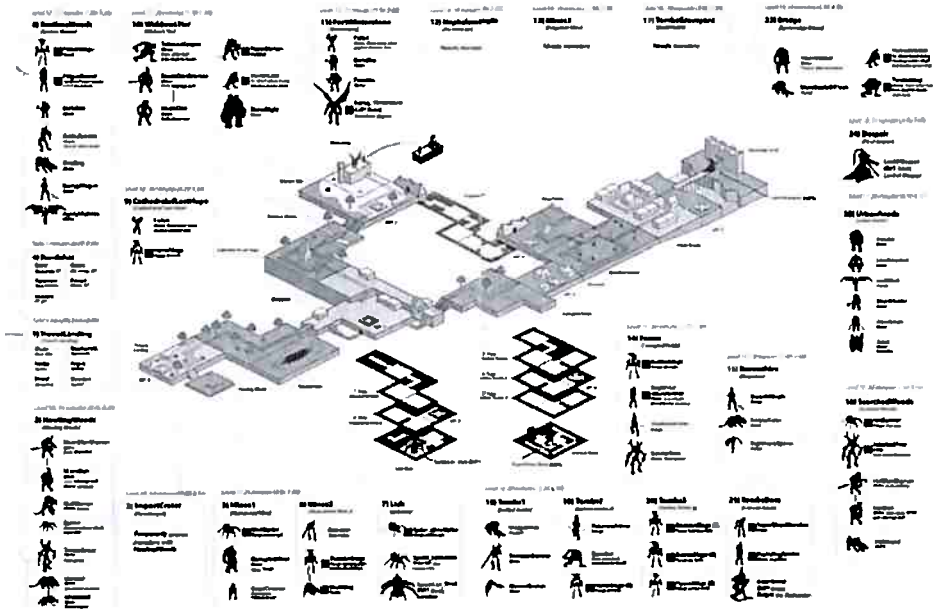

Figura 10 – Pôster de Design de Uma Página (Librande, 2010).

O designer Rogers (2010) também apresenta um exemplo da abordagem de esquemas visuais sintéticos em seu livro e cita que esta é uma prática que tem ganhado adeptos. Por outro lado, ele enfatiza que a abordagem encontra barreiras culturais que inibem a sua aplicação em estúdios mais conservadores, que temem pela perda de controle do projeto por não adotarem documentos de design tradicionais, de estilo mais contratual.

2.2.2.4 Abordagens de Poucas Aplicações

Segundo Dormans (2012), existe grande ceticismo por parte de designers profissionais acerca das abordagens propostas pela academia, que julgam esses trabalhos como desperdício de tempo e uma visão demasiadamente formalizada para uma atividade que constantemente requer criatividade. De uma forma geral, os métodos discutidos neste capítulo são propostos para analisar jogos e não para projetá-los. Esses trabalhos têm um alto teor acadêmico e são comumente desenvolvidos dentro de universidades que, ao contrário de empresas, não atuam ativamente no desenvolvimento de novos jogos. No entanto, dada a natureza da indústria de jogos, os métodos de design devem ajudar a criar jogos e não apenas analisá-los.

Os trabalhos discutidos não demonstram uma aplicação prática dos métodos que propõem, ao mesmo tempo em que não apresentam instruções de uso, fatos que corroboram para distanciá-los do emprego no mundo real. Nesse sentido, de um lado está a indústria que, motivada pelos fatores de risco financeiros relacionados ao emprego de abordagens alternativas, mostra-se cética quanto a valia e produtividade desses métodos, do outro, está a própria academia. Segundo Dormans (2012), mesmo os autores desses trabalhos são incertos quanto ao benefício real que suas propostas podem trazer se aplicadas, e dessa forma, acabam

aprofundando-se em discussões especulativas e questões teóricas. Por outro lado, os esquemas visuais de Librande (2010) demonstram uma natureza fundamentalmente prática em sua concepção. O fato de que designers profissionais utilizam alguma forma de esquema sintético que prioriza a intuitividade e a clareza na transmissão das informações indica que, ao contrário das propostas previamente discutidas, que parecem privilegiar a busca pelo vigor formal, o caminho para um instrumento praticável aparentemente reside na simplicidade e na flexibilidade de modelos mais dinâmicos.

De uma forma geral, a existência dos trabalhos discutidos e o fato de que suas propostas se aglomeram em dois tópicos bem definidos – vocabulários e linguagens visuais – são fortes indicativos de que esses são pontos importantes e necessários à área. Isso mostra que designers desejam uma forma mais padronizada e estruturada de comunicar, registrar e transmitir suas ideias, ao mesmo tempo em que precisam de uma base de conhecimentos acessível a todos os envolvidos na comunidade de design de jogos, sejam eles hobistas, pesquisadores acadêmicos ou profissionais experientes. Cabe ressaltar que, com exceção do trabalho de Kuittinen (2008), não houve uma intenção clara de criar uma interseção entre os dois temas apontados: vocabulários e linguagens de modelagem. Nesse sentido, os esquemas de design poderiam modelar um jogo por meio da inter-relação entre conceitos, que por sua vez constituíram um vocabulário dos elementos estruturantes do design. Dentro desse contexto está inserida a abordagem de design de jogos baseado em componentes, que terá aspectos discutidos em relação aos trabalhos relacionados na próxima seção.

2.3 Abordagem de Componentes de Jogos

A mídia especializada em jogos habitualmente promove revisões de títulos que são frequentemente utilizadas por consumidores e produtoras como termômetro para aferir a qualidade dos lançamentos no mercado. Web sites como IGN[36], Eurogamer[37], GameSpot[38], Game Informer[39] e Kotaku[40] mantêm revisões escritas por profissionais especializados, enquanto outros, como GameFAQS[41] e Giant Bomb[42], reúnem revisões feitas por usuários. Um fato interessante nos textos de artigos publicados nesses sites é a frequente presença do termo "elemento". Ele é utilizado para referenciar os conceitos empregados nos jogos e para identificar características fundamentais a gêneros. Como exemplo, no artigo "*The Division Has RPG and Survival Elements*" (Karmali, 2013), o autor discute a presença de elementos típicos do gênero RPG (Role-Playing Game) e Survival (sobrevivência) no conceito do jogo The Division. Em especial, ele cita elementos tradicionais de RPG como "evolução" e "gerenciamento de inventário", assim como, o elemento "escassez de itens e munição", uma restrição comumente empregada em jogos que envolvam sobrevivência. Em outro texto intitulado "*The Evolution of*

[36] www.ign.com (visitado em 20/03/2016)
[37] www.eurogamer.net (visitado em 20/03/2016)
[38] www.gamespot.com (visitado em 20/03/2016)
[39] www.gameinformer.com (visitado em 20/03/2016)
[40] www.kotaku.com (visitado em 20/03/2016)
[41] www.gamefaqs.com (visitado em 20/03/2016)
[42] www.giantbomb.com (visitado em 20/03/2016)

Single Player Co-op", Clarke (2014) discorre de forma analítica sobre exemplos de jogos de um jogador que empregam elementos comuns a modos multijogador (*multiplayer*). Além da mídia especializada, nos textos de descrições de jogos hospedados na Wikipedia[43] é também comum encontrar a menção ao termo "elemento" para descrever aspectos chave de jogos e gêneros. De uma forma geral, nota-se que o entendimento de que existem aspectos identificáveis no design de jogos, sejam eles tratados como elementos, padrões de design ou termos estruturados de um vocabulário, não se restringe a academia e indústria.

O termo "elemento de jogos" frequentemente mencionado na mídia especializada assemelha-se, mesmo que informalmente, as abordagens descritas nos trabalhos que tiveram por objetivo constituir uma estrutura para organizar um vocabulário de design, discutidos na seção 2.2.1. No meio profissional, esse conceito também está presente no cotidiano dos designers de forma implícita, seja no momento em que buscam por elementos interessantes a seus projetos em jogos existentes ou na observação daqueles que tipicamente recorrem em jogos do mesmo gênero, caracterizando-o. Nota-se que existe uma necessidade pela caracterização desses "elementos estruturantes". Embora esses elementos estruturantes façam parte de um vocabulário informal empregado na mídia, sejam objeto de discussão em diversos trabalhos já mencionados e estejam inseridos no trabalho do designer, o documento de design tradicional continua sendo o método padrão na indústria. As questões apontadas no final da seção anterior certamente corroboram para isso. Entre elas, o rigor formal utilizado nas abordagens propostas certamente dificultou suas aplicações de forma prática. Nesse sentido, esta tese documenta uma abordagem que busca responder a questões como: o que são esses elementos, como são estruturados e de que forma se relacionam para constituir os jogos? Aqui acredita-se que é possível não somente compreender os jogos existentes como uma composição de elementos estruturantes, como também, catalogar esses elementos de forma que ajudem na concepção de novos jogos. Para tanto, fundamenta-se e, especialmente, diferencia-se em uma construção não previamente explorada nos demais trabalhos: a composição.

A abordagem de componentes de jogos baseia-se no conceito de blocos de construção. Retomando a definição apresentada na seção 1.1, um componente é qualquer aspecto distintamente reconhecível do jogo que possa ser univocamente identificado, desconectado do todo e apresente influência estética. A premissa da abordagem é a de que jogos podem ser pensados, analisados, registrados e projetados como uma composição. Uma aplicação peculiar do conceito de composição reside na definição dos componentes fundamentais de um jogo, permitindo descrevê-lo sinteticamente em poucos componentes. Esses "ingredientes" principais são aqueles que caracterizam um determinado jogo ou gênero. Como exemplo os elementos-chave da franquia de jogos Super Mario Bros. podem ser resumidos em quatro componentes:

- [Jump] <action>

[43] https://www.wikipedia.org/

- [Head Jump] <attack>
- [Platform] <level construction>
- [Obstacle] <level construction>
- [Collect Items] <secondary objective>

Segundo o exemplo, os jogos da franquia Super Mario tipicamente envolvem a ação de pular (componente [Jump], do tipo <action>) e atacar inimigos pulando sobre suas cabeças ([Head Jump]); possuem plataformas[44] como principal elemento de construção dos níveis do jogo ([Platform]), acompanhadas por obstáculos ([Obstacle]), tais como buracos e espinhos; e levam o jogador a coletar itens, representados no Super Mario por moedas ([Collect Items]). Há certamente vários outros componentes na constituição do design de jogos da franquia Super Mario, mas esses quatro podem ser considerados chave. Isso pode ser evidenciado pelo fato de que nenhum dos jogos da franquia deixa de possuí-los. Ademais, os três componentes descrevem aquilo que os jogadores buscam em um jogo da franquia: desenvolver a habilidade de controlar o pulo do personagem a fim de cair sobre as plataformas, sempre coletando as moedas que estão espalhadas pelos níveis do jogo.

O exemplo apresentado acima também serve a outro propósito. É interessante notar que ele não somente define os componentes fundamentais dos jogos da franquia Super Mario, como também, ajudam a expressar o conceito do próprio gênero no qual esses jogos são classificados. Removendo os componentes [Head Jump] e [Collect Items], chegamos a representação do gênero "plataforma", que pode ser descrito pela seguinte composição:

```
[Platformer] <genre>
{
      [Jump] <action>
      [Platform] <level construction>
      [Obstacle] <level construction>
}
```

Segundo a representação acima, os jogos do gênero plataforma são tipicamente caracterizados pela ação de pular sobre plataformas e evitar obstáculos. Nesse tipo de jogo o foco do *gameplay* está em desenvolver a habilidade de controlar a trajetória e a altura do pulo para atingir os objetivos apresentados para o jogador. Sobre essa base de componentes estabelecidas para o gênero, designers tipicamente adicionam componentes a fim de enriquecer o *gameplay* de seus projetos. Nesse sentido, em alguns jogos de plataforma tem por objetivo alcançar o final do nível ([Reach End of Level] <primary objective>), tal como em Super Mario

[44] No contexto de jogos, uma plataforma é uma superfície na qual o jogador pode pisar. Nos estágios ou *levels* dos jogos, as plataformas encontram-se em diferentes alturas, de forma que o jogador tenha que desenvolver a habilidade de controlar o pulo para alcançá-las.

(jogos 2D), Sonic[45], Ducktales[46] e Crash Bandicoot[47]. Em outros, o objetivo é coletar itens ([Collect Items] <primary objective>), como as estrelas de Super Mario 64[48] e Galaxy[49], as gemas dos três primeiros títulos da série Spyro[50] e as notas musicais de Banjo & Kazooie[51].

Movimentos de ataque também são empregados no gênero e há exemplos peculiares de componentes facilmente identificáveis. Nos jogos da série Super Mario, o jogador pode usar [Head Jump] (pula sobre a cabeça dos oponentes), [Throw Object] (lança bolas do fogo) ou [Melee Attack] (socos e chutes em Super Mario 64 e rabadas em Super Mario Bros. 3). Nos jogos da franquia Sonic, o personagem ataca os inimigos com os movimentos [Spin Jump] (pula girando como uma serra) e [Spin Dash] (gira como uma serra, movendo-se rapidamente para frente). Já em Crash Badicoot, o personagem do jogador pode usar um ataque giratório ([Spin Attack]), escorregar ([Slide Attack]) ou pular sobre o oponente ([Head Jump]). Esses exemplos mostram que no processo de composição, o designer pode partir de um conceito base, tais como os componentes fundamentais de um gênero ou outro jogo, e acrescentar componentes de forma a definir uma nova experiência. Além disso, é possível notar a semelhança entre os jogos citados – Super Mario, Sonic, Crash Bandicoot, Ducktales, Spyro e Banjo & Kazooie – por meio dos componentes que empregam. Nesse sentido, todos apresentam os componentes definidos para o gênero plataforma e, portanto, pertencem a este gênero, e dessa forma, são semelhantes entre si. Também é interessante notar que, partindo-se da composição do gênero plataforma, é possível adicionar componentes de forma a alcançar um design similar aos tipos de jogos de plataforma encontrados no mercado.

Utilizando-se um processo de design por composição, é possível partir de um conceito mais simples, como a composição de um gênero ou os componentes fundamentais de um jogo, e adicionar gradativamente novos componentes até se alcançar o design desejado. Considerando o exemplo anterior, sobre a composição básica do gênero plataforma ([Jump] + [Platform]), pode-se adicionar componentes de "perspectiva" (<pov>) e "controle" (<control>) para se alcançar o design preliminar de diferentes variações de jogos do gênero plataforma. Com esse intuito, as Figuras 11 a 14 ilustram quatro diferentes composições de design concebidas sobre o modelo básico do gênero plataforma. Nelas, o jogador é identificado por uma esfera de cor laranja e as plataformas, suspensas no ar ou sobre apoios, são representadas pelos pisos de madeira e os blocos de pedra. É interessante notar que essas quatro composições se assemelham a diversos jogos pulicados e bem conhecidos no mercado.

A Figura 11 representa um jogo típico de plataformas 2D, no qual emprega-se visão lateral ([Side View] <pov>) e o personagem do jogador tipicamente contém o componente de controle [Move in Two Directions] <control>, restrito ao eixo horizontal. Por meio deste, o

[45] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/SonicTheHedgehog1
[46] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/DuckTales
[47] http://tvtropes.org/pmwiki/pmwiki.php/Franchise/CrashBandicoot
[48] http://tvtropes.org/pmwiki/pmwiki.php/Videogame/SuperMario64
[49] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/SuperMarioGalaxy
[50] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/SpyroTheDragon1998
[51] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/BanjoKazooie

personagem pode ser movido lateralmente, mudando a direção imediatamente para a aquela apontada pelo jogador (esquerda ou direita). O controle vertical ocorre pelo componente [Jump] <action>, que permite ao jogador explorar o cenário. Exemplos de jogos consolidados no gênero que utilizam os mesmos princípios são Super Mario Bros. 3 e Sonic. Títulos mais contemporâneos incluem Little Big Planet[52] e Donkey Kong Country Tropical Freeze[53].

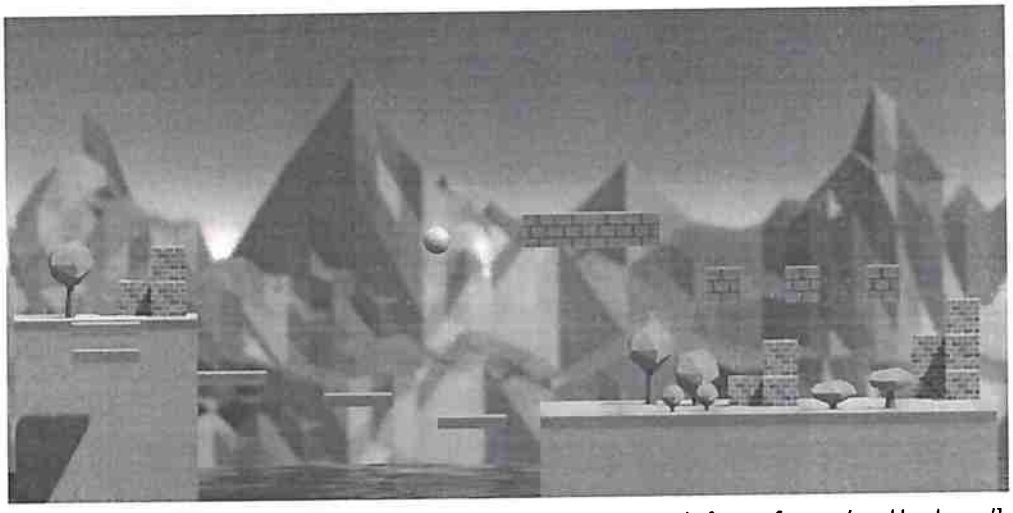

Figura 11 – Jogo de plataformas com perspectiva [side view] e controle [move forward and backward].

A Figura 12 apresenta o mesmo jogo do exemplo anterior visualizado por uma perspectiva superior ([top view] <pov>). Nele, o componente de controle [move freely] <control> permite que o jogador se mova livremente em qualquer direção com um joystick de controle analógico. Por outro lado, se utilizasse um controle digital em formato de cruz, o componente [move in eight directions] <control> se mostraria mais adequado. Neste exemplo, o [jump] <action> tem um papel diferente do jogo de visão lateral, pois a percepção do eixo vertical pelo jogador fica restrita. Dessa forma, esse componente limita-se a permitir que o jogador alcance plataformas suspensas ou suba em blocos acima do nível do chão. Embora seja raro encontrar jogos de plataforma que utilizem esse formato, jogos de outros gêneros tendem a embuti-los. Exemplos disso são os títulos 2D da séria Zelda.

Figura 12 – Jogo de plataformas com perspectiva [top view] e controle [move freely].

[52] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/LittleBigPlanet
[53] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/DonkeyKongCountryTropicalFreeze

A Figura 13 demonstra o mesmo jogo de plataformas sob uma perspectiva isométrica ([isometric view] <pov>). Os componentes de controle seriam os mesmos do jogo na visão superior. Contundo, o [jump] <action> provocaria no gameplay o mesmo impacto do jogo em visão lateral. Com ele, o jogador alcançaria diferentes níveis do cenário. De forma semelhante ao exemplo anterior, jogos de plataforma isométricos são raros. Um exemplo significativo é o jogo Sonic 3D Blast[54].

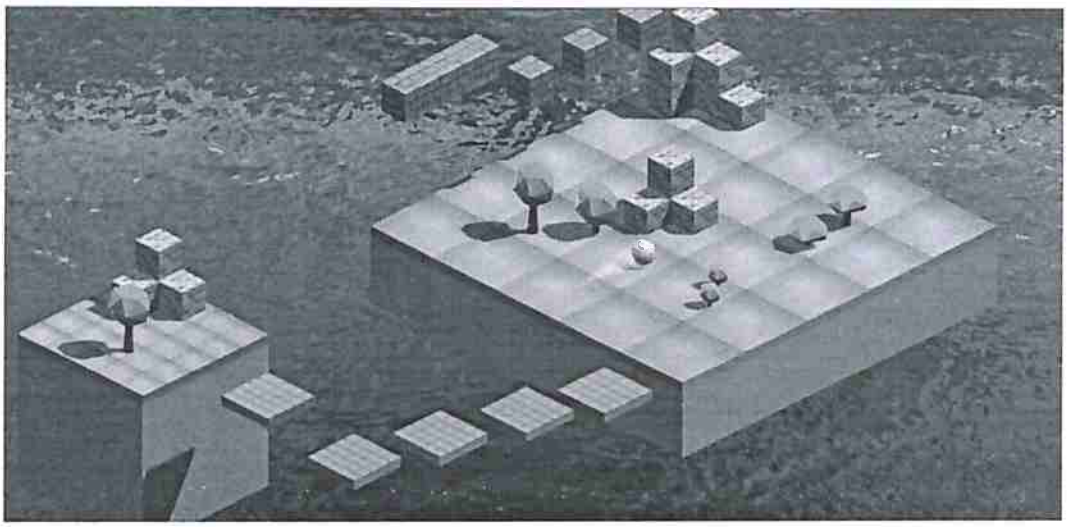

Figura 13 – Jogo de plataformas com perspectiva [isometric view] e controle [move freely].

Ao final, a Figura 14 ilustra o mesmo jogo de plataformas em versão 3D, no qual o jogador passa a ter controle sobre a câmera, que flutua livremente ao redor do seu personagem ([free camera] <pov>]. O controle mantém-se por [move freely] <control>. Contudo, as direções tornam-se baseadas na câmera, o que permite que o jogador se oriente de forma mais natural. Portanto, há o emprego do componente [camera based movement] <control>, que foi originalmente empregado no design do jogo Super Mario 64 e tornou-se massivamente popular em jogos de câmera livre que o sucederam. Exemplos de jogos no formato apresentado incluem

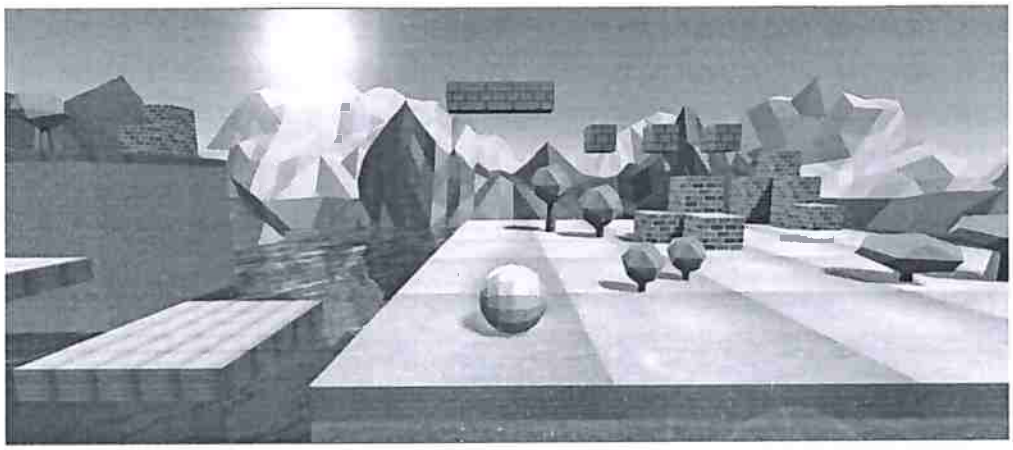

Figura 14 – Jogo de plataformas com perspectiva 3D por [free camera]. O jogador controla o personagem pelos componentes [move freely] + [camera based control].

[54] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/Sonic3DFlickiesIsland

Super Mario Galaxy, Banjo & Kazooie, Gex Enter the Gecko[55] e Yooka-Laylee[56].

O objetivo dos quatro exemplos propostos foi demonstrar que, embora representem diferentes tipos de jogos de plataforma, os componentes fundamentais ao gênero se mantêm. Dessa forma, eles podem ser enquadrados no gênero por conterem tais partes em sua composição. O mesmo pode ser observado em títulos do mercado. Diversos jogos conhecidos utilizam o mesmo formato do jogo apresentado na Figura 11. Entre eles, estão os jogos 2D das franquias Super Mario e Sonic, e títulos como Ducktales, Castle of Illusion[57], Super Meat Boy[58], Outland[59], Rayman[60] e Little Big Planet. Além destes, jogos como Trine[61], Fez[62], Limbo[63] e Braid[64], considerados na mídia especializada como jogos de plataforma e quebra-cabeças, também empregam os mesmos componentes básicos do gênero plataforma, embora as características de quebra-cabeças que possuam influenciam significativamente o seu *gameplay*, ao ponto de distanciá-los de títulos mais tradicionais, como Super Mario e Sonic. O mesmo ocorre com Guacamelee[65], Cave Story[66] e a série Metroid (somente jogos 2D), que mantém a estrutura de jogos de plataforma, mas enfatizem a exploração em um mundo aberto e a busca por recursos e habilidades. Embora esses dois últimos grupos de jogos não sejam considerados jogos "puros" de plataformas, é interessante notar que todos eles mantêm a composição fundamental ao gênero "plataforma" apresentada, o que contribui para atestar o conceito de composição.

2.3.1 Análise Frente a Outras Abordagens

O mapa apresentado na Figura 4 organiza as principais discussões e abordagens que tiverem como objetivo buscar alternativas ou complementos ao método corrente de design. Dentro desse mapa, a abordagem de componentes de jogos desta tese pode ser enquadrada na parte superior, como uma tentativa de contribuir com a definição de um vocabulário compartilhado de design. Nesse contexto, os componentes de jogos não somente agem como os elementos estruturantes para o design, como também definem um vocabulário a ser utilizado. Cabe ressaltar que a abordagem de componentes também contribui com a busca por linguagens visuais de design (parte inferior do mapa), uma vez que permite a modelagem tanto textual de componentes (Figura 1), quanto na forma de cartões ou pôsteres (Figura 2). Dessa forma, esta seção apresenta uma análise da abordagem de componentes frente aquelas com as quais apresenta maior relação.

As abordagens discutidas na seção 2.2.1 tratam da estruturação de aspectos aplicáveis

---

[55] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/Gex
[56] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/YookaLaylee
[57] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/CastleOfIllusion
[58] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/MeatBoy
[59] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/Outland
[60] http://tvtropes.org/pmwiki/pmwiki.php/Videogame/Rayman1995
[61] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/Trine
[62] http://tvtropes.org/pmwiki/pmwiki.php/Videogame/Fez
[63] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/Limbo
[64] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/Braid
[65] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/Guacamelee
[66] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/CaveStory

ao design dos jogos, sejam eles tratados como conceitos, padrões de design, termos ou diretrizes. Contudo, eles não se fundamentam na premissa da composição. Nesse sentido, não é possível definir um jogo pela composição de FADTs, Padrões de Design ou termos do projeto de ontologias. Para exemplificar a ideia, tomemos como exemplo a "Consequência Perceptível", um conceito documentado como um FADT e um Padrão de Design, que tem por definição "uma reação clara do mundo do jogo à ação do jogador". Não há como considerar esse conceito como um bloco de construção, isto é, uma peça que possa ser agrupada a outras para montar um todo, que neste caso corresponderia a um jogo. Por outro lado, pode-se visualizá-lo como uma sugestão de caminho a ser seguido, como uma diretriz genérica e abstrata. Nesse sentido, sua realização no conceito de um jogo pode se dar de várias formas. Um exemplo é a resposta da colisão do personagem do jogador com outro personagem, objeto ou parte do mundo do jogo. Em outra situação, pode ser a emissão de um som como resposta negativa à tentativa do jogador em danificar outro personagem ou abrir uma porta que está trancada. A quantidade de situações em que um jogo pode emitir alguma forma de resposta ao jogador como consequência de uma ação sua é muito vasta. O conceito de uma Consequência Perceptível é algo completamente transversal, que passa através de toda a interface entre jogo e jogador. Dessa forma, ele não ajuda o designer a montar um jogo, mas como diretriz, o lembra da necessidade ou importância de prover respostas ao jogador, a partir de suas ações, em todas as partes do jogo em que isso ocorre.

Em outro exemplo, o padrão de design intitulado "Percepção Exagerada de Influência" define que os jogadores devem acreditar que podem influenciar o resultado do jogo, independentemente dessa possibilidade existir de fato. Esse conceito define uma característica estética do jogo de aplicação ampla, podendo referir-se tanto a apresentação gráfica quanto a narrativa do jogo. Os dois exemplos apresentados estão centrados em tratar o resultado da experiência de interação do jogador com partes do jogo, ao invés de tratar dessas partes em si. Há Padrões de Design que tratam de partes de jogos, mas o fazem de forma muito genérica. Por outro lado, a abordagem deste trabalho dirige-se especificamente a essas partes, a partir de um nível mais baixo de granularidade estrutural que, ao serem combinados para formar o *gameplay* dos jogos, promovem aspectos estéticos. Dessa forma, o conceito de componentes força o designer a compreender o jogo pela perspectiva do jogador, por meio da influência que cada pequena parte possui direta ou indiretamente no *gameplay*.

Nesta tese entende-se que a diferença entre os jogos está nos detalhes de cada componente e na forma como se relacionam. Como exemplo, a "Câmera Sobre o Ombro" – [Over the Shoulder Camera] <pov> – é um componente de Ponto de Vista (POV - point-of-view) característico dos jogos contemporâneos dos gêneros de Ação (Action), especialmente no subgênero Tiro (Shooter), e Aventura (Adventure). Originado em Resident Evil 4[67], ele tornou-se padrão nos jogos que o sucederam, pois contribui para uma maior imersividade se comparado a câmera em terceira pessoa tradicional, com uma perspectiva mais afastada por trás do jogador – [Behind Player Camera] <pov>. Somado a isso, o [Over the Shoulder Camera] pode ser

[67] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/ResidentEvil4

considerado um ponto de vista intermediário entre a [First Person Camera] e a [Behind Player Camera], trazendo consigo os benefícios de ambos componentes, como um maior campo de visão e a visualização das ações do personagem controlado, tais como pular ([Jump] <action>), escalar ([Climb] <action>) e abrigar-se ([Take Cover] <action>). Além disso, permite ao jogador observar o personagem controlado em alto nível de detalhes, valorizando a produção artística do jogo. Por essas razões, tornou-se padrão em jogos ação e aventura, como os da série Resident Evil, Uncharted[68] e Batman Arkham[69]. Ainda no contexto do jogo Resident Evil 4, é curioso notar a ausência de [Strafe[70]] <movement>, mesmo que os jogos de tiro (Shooter) de sua época o contenham. Tal componente permite que o jogador se mova lateralmente a fim de desviar de disparos e ataques inimigos, bem como, facilita o posicionamento da mira para efetuar disparos. A ausência do componente no jogo tem o objetivo claro de aumentar a tensão do jogador ao enfrentar oponentes a curta distância ou mesmo nas ações evasivas a ataques recebidos de longa distância. A ausência proposital de um determinado aspecto no jogo é chamada de **componente negativo**.

A relação entre componentes e os aspectos estéticos que gera não se restringe à perspectiva do jogador. Dentre os componentes de ações, o "Pulo Duplo" ([Double Jump] <jump>) é amplamente empregado em uma grande variedade de jogos do gênero plataforma, especialmente quando deseja-se prover ao jogador a possibilidade de realização de manobras áreas de grande flexibilidade e respostas rápidas. Ele é utilizado tanto como recurso de manobra de evasão em combates (ex: Guacamelee), quanto para permitir maior exploração em jogos com mundos vastos (ex: Metroid). Em contrapartida, o "Pulo de Queda Suavizada" ([Smoothed Fall Jump] <jump>) torna o ritmo do jogo mais lento e permite ao jogador um maior tempo de resposta para tomada de decisões, além de alcançar locais mais distantes. Registrar as partes de jogos simplesmente como "Pulo", "Combate" ou "Combo" não permite capturar as particularidades do design de cada jogo e as peculiaridades de seus detalhes. Nesse sentido, o padrão de design "Combate" (ver wiki[71]) torna-se muito amplo, uma vez que existem centenas de tipos diferentes de combate, que podem variar amplamente, mesmo em jogos dentro de um gênero. Da mesma forma, o padrão de design "Combo" é claramente distinto em jogos como Virtua Fighter[72], Final Fight[73] e Street Fighter II[74]. Por fim, o conceito de "Pulo" não é sequer documentado como padrão de design. Em uma alusão ao desenvolvimento de software, as abordagens de coleções de design previamente discutidas apresentam características transversais às partes formadoras dos jogos, de forma similar ao paradigma de aspectos da programação de computadores (Kiczales, 1997). Por outro lado, a abordagem de componentes toma por base a criação de jogos pela associação de suas partes formadoras, tratando-as como "coisas" e, dessa forma, assemelhando-se ao paradigma de orientação a objetos (OOP) descrito

---

[68] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/Uncharted
[69] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/BatmanArkhamSeries
[70] http://www.giantbomb.com/strafing/3015-520/
[71] http://protagonist.sics.chalmers.se:1337/mediawiki-1.22.0/index.php/Category:Patterns
[72] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/VirtuaFighter
[73] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/FinalFight
[74] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/StreetFighterII

por Fowler (2004). As similaridades com o modelo de OOP são discutidas na seção 3.2.

2.3.2 Conceitos da abordagem

A abordagem de design de jogos baseado em componentes é uma proposta alinhada às discussões preliminares acerca da necessidade por instrumentos de design de jogos publicadas por Costikyan (1994) e Church (1999), que expuseram a necessidade de analisar e criar jogos por meio de suas partes formadoras, compreendendo quais dessas os beneficiariam ou os prejudicariam. A abordagem se apoia na experimentação e análise prática dos jogos existentes, em contraste a uma investigação teórica dos fundamentos que eles possam empregar. Neste trabalho, acredita-se que existe uma extensa base de conhecimentos de design não formalizada e pouco explorada embutida na vasta biblioteca de jogos já publicados e, especialmente, nas partes constituintes desses jogos e em suas relações. Essa biblioteca representa o resultado concreto e consolidado da aplicação da experiência e conhecimento de milhares de designers de jogos. Também acredita-se que o resgate desse conhecimento pode ser realizado por meio da identificação e reúso dos componentes que empregam. Esses conceitos e outros são apresentados nesta seção.

2.3.2.1 Design de Jogos e o Reuso de Componentes

Ao observar os jogos contemporâneos, é possível notar que muitas de suas características foram emprestadas ou modificadas de jogos anteriores, sejam eles do mesmo gênero ou não. O reuso de partes de jogos em novos títulos é uma prática comum na indústria. Criar um jogo não significa necessariamente conceber algo absolutamente novo, mas muitas vezes modificar conceitos conhecidos ou misturá-los de uma forma inusitada. Jogadores esperam encontrar aspectos familiares de *gameplay* em gêneros e jogos de sua preferência. Dessa forma, a recorrência de características em jogos é, de certo modo, algo naturalmente esperado. No ponto de vista do jogador, os gêneros existem para definir características fundamentais aos jogos que classificam. Assim, jogadores que têm por preferência o gênero "Luta" ou "Corrida", esperam que certas características estejam presentes nos jogos desses tipos.

Sob a perspectiva de quem concebe os jogos, o reuso serve de base para novas ideias. Designers comumente se apoiam em suas experiências passadas e no trabalho de outros para construir seus jogos. Church (1999) enfatiza essa ideia ao afirmar que nenhum designer de jogos "trabalha no vácuo". Nesse sentido, designers comumente realizam pesquisas em jogos para determinar características que se despontam e que podem ser usadas de forma benéfica em seus projetos. Como exemplo, se um designer deseja criar um jogo do gênero plataforma, ele certamente experimentará os principais e os mais peculiares jogos do gênero a fim de identificar as partes que têm em comum, as que os tornam únicos e aquelas que os beneficiam ou prejudicam. A partir dessa base de conhecimentos, o designer passa a ter um instrumental que o ajudará a realizar combinações conceituais de partes de jogos, servindo de fundação para a geração de novas ideias. Essa abordagem de pesquisa e reuso é facilmente perceptível nos jogos já publicados no mercado, embora não seja explicitamente discutida.

Desenvolvedores de grande sucesso ocasionalmente discorrem sobre os títulos anteriores que os inspiraram e as ideias que emprestaram desses jogos. No entanto, esse tipo de relato não é comum na indústria, provavelmente porque os grandes os estúdios temem por denegrir a originalidade do novo jogo perante seus consumidores. Na GDC de 2007, o designer Cliff Bleszinski fez uma apresentação na qual comenta sobre os jogos que o inspiraram a criar Gears of War[75], um título de grande sucesso na época (Thorsen, 2007). Ele cita três jogos como os formadores dos alicerces do *gameplay* de seu projeto: o ritmo e a câmera sobre o ombro de Resident Evil 4 ([Over the Shoulder Camera]), o sistema de abrigar-se atrás de proteções nos tiroteios de kill.switch[76] e a forma como o jogador ascende nas plataformas de Bionic Commando[77], reutilizada no eixo horizontal para permitir ao jogador avançar entre coberturas. Ao se jogar Gears of War é fácil perceber o reuso das ideias citadas, mesmo que não se tenha conhecimento prévio do relato mencionado acima. O mesmo pode ser dito sobre diversos outros títulos do mercado, nos quais nota-se uma clara recorrência de ideias. No próximo parágrafo são apresentados exemplos de reúso de componentes em títulos de grande popularidade.

Dois jogos de sucesso que definiram as bases do gênero "plataforma", Super Mario Bros. e Super Mario Bros. 3, servem de alicerce para diversos que os sucederam. Lançamentos posteriores como Kid Chameleon[78], Castle of Illusion e Crash Bandicoot, ou mesmo títulos mais recentes, como Braid, Spelunky[79] e Super Mario Galaxy 2, apresentam recorrência de vários componentes dos dois precursores. Exemplos notáveis desses componentes estão enumerados abaixo:

- [Jump] <action>: pulo;
- [Head Jump] <attack> <jump>: pulo com pisão na cabeça;
- [Pickup and Hold Object] <action>: pegar e segurar objeto, como cascos de tartarugas em Super Mario e barris em Castle of Illusion;
- [Collectible] <item>: coletáveis, como moedas, anéis e gemas;
- [100 Coins to Life] <rule>: 100 moedas para uma vida;
- [Temporary Invincibility] <status>: invencibilidade temporária;
- [Checkpoint] <level construction>: ponto de retorno;
- [Weak Platform] <platform>: plataforma frágil;
- [Moving Platform] <platform>: plataformas móveis;

---

[75] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/GearsOfWar
[76] http://www.giantbomb.com/killswitch/3030-1152/
[77] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/BionicCommando
[78] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/KidChameleon
[79] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/Spelunky

- [Balanced Lifting Platforms] <platform>: plataformas de elevação equilibradas, interligadas por um cabo em roldanas, funcionando de forma similar a uma gangorra (Figura 15);
- [Trampoline] <jump enhancer> <level construction>: trampolim (Figura 15);
- [End of Level Marker] <level construction>: marcação de final de estágio;
- [Lethal Obstacle] <level construction>: obstáculo letal, como espinhos, poços de lava e serras;
- [Object with Items] <level construction>: objetos com itens, como os candelabros e adereços de Castlevania[80] e Ninja Gaiden[81], os barris e caixas de Resident Evil 4 e 5, ou os blocos de Super Mario Bros (Figura 15).

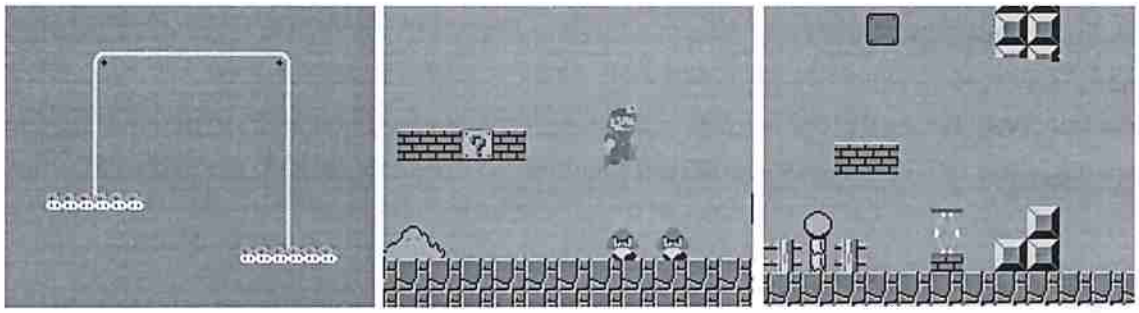

Figura 15 – Componentes [balanced lifting platforms], [object with items] e [trampoline].

O icônico ataque do personagem Mario, o componente [head jump] <attack> <jump>, reaparece com algumas modificações em alguns dos jogos citados. Em Ducktales e Castle of Illusion, o ataque requer o pressionamento de um botão para ser realizado. Little Big Planet, por outro lado, permite o uso do mesmo componente de ataque somente em alguns estágios. Jogos de outros gêneros, como Retro City Rampage[82], um jogo de ação-aventura de mundo aberto, e a série de luta Street Fighter, também utilizam o [head jump] de Mario. A razão pela qual o reuso ou adaptação dos componentes citados ocorre provavelmente deve-se ao grande sucesso dos dois jogos citados e à experiência de *gameplay* que eles promovem. De uma forma geral, o estudo da constituição dos jogos existentes em busca de componentes de valor representa uma importante ferramenta de criação de jogos e pode ajudar a aprimorar o próprio ofício do design.

2.3.2.2 Componentização de Design

Os componentes de jogos agem como elementos estruturantes no design, descrevendo jogos como uma coleção inter-relacionada de partes menores. Utilizando esse princípio, designers passam a pensar, registrar e projetar os jogos por meio de sua constituição. A componentização é uma abordagem de design intrinsecamente ligada ao conceito de abstração de complexidade. Componentes de jogos são projetados, reusados, modificados e, ao final,

[80] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/Castlevania1
[81] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/NinjaGaiden
[82] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/RetroCityRampage

combinados para criar novos componentes "maiores". Quaisquer componentes podem fazer parte da constituição de componentes mais complexos. Esse processo é realizado de forma sucessiva, culminando na formação do design do jogo. Esse mecanismo torna-se simples e flexível: jogos inteiros podem ser representados como componentes, permitindo que novos projetos possam ser descritos como um agrupamento de outros jogos ou como uma combinação de alguns de seus componentes. Por meio da componentização, designers podem descrever não somente jogos, mas também gêneros, como um grupo inter-relacionado de componentes. A abordagem age tanto como um instrumento para o design de novos jogos quanto para a análise de jogos existentes.

A **análise por decomposição** representa um método de engenharia reversa "top-down" pelo qual designers desmontam jogos em seus componentes. Nesse processo, identifica-se os componentes recorrentes de outros jogos bem como aqueles peculiares ao jogo sob análise. O intuito é compreender a constituição de um jogo e produzir uma documentação de design que registra esse conhecimento, de forma que possa ser futuramente empregado em novos projetos. Os componentes são documentados conforme uma estrutura estabelecida. O processo de engenharia reversa de jogos representa, por si, um importante exercício de design, pois força designers a compreender a influência que cada parte menor possui no resultado final. Ademais, leva a uma melhor compreensão do trabalho de outros designers, em especial, seus acertos e falhas.

O **design por composição** representa um processo análogo à análise, no qual designers "montam" o conceito de um jogo a partir da inter-relação de suas partes. Nesse processo, os componentes são projetados, reusados, modificados e combinados a fim de criar novos componentes. Esses, por sua vez, também podem vir a fazer parte de componentes mais complexos, até formar o design do jogo. Há duas abordagens para o design por composição: "bottom-up" e "top-down". No design por composição "top-down" o jogo é projetado quebrando-se consecutivamente seu conceito em partes menores inter-relacionadas, da mesma forma que na análise por decomposição. O reconhecimento de recorrência de componentes ocorre à medida que os mesmos são identificados de forma genérica, em níveis mais baixos de abstração de composições. Por outro lado, na abordagem "bottom-up" de design por composição, o conceito do jogo surge da combinação de componentes de jogos previamente analisados com aqueles especificamente concebidos para o projeto. Nesse sentido, o conceito do jogo emerge a partir da junção de suas partes menores.

2.3.2.3 Descoberta de Conhecimentos de Design

Church (1999) afirma que a base tecnológica dos jogos evolui constantemente, mas os métodos e as ferramentas de design não o fazem no mesmo ritmo. Para ele, a falta de uma base de conhecimentos compartilhada de design representa um entrave à evolução da área como um todo: "*Como designers, nós construímos nossas ideias sobre nossas experiências e as ideias passadas de outros. Com raras exceções, ninguém trabalha no vácuo. No entanto, não temos como identificá-las, estruturá-las e documentá-las.* " Nesse contexto, a abordagem de

componentes define o registro de ideias de design sob a forma de componentes e suas inter-relações. Dessa forma, o conhecimento gerado nas atividades de análise por decomposição e de design por composição pode ser persistindo uma base de dados. A partir disso, torna-se possível o emprego de técnicas de pesquisa e mineração de dados para análise da composição dos jogos e gêneros. Como exemplo, é possível averiguar a similaridade entre jogos com base nos componentes que empregam. Além disso, pode-se descobrir a relação hierárquica em um grupo de jogos de forma a construir "árvores genealógicas". Essas árvores podem relacionar jogos, componentes e gêneros, de forma a representar a evolução cronológica de títulos e as relações existentes entre suas constituições.

A Figura 16 ilustra o conceito de árvore genealógica para o jogo Uncharted, enfatizando uma visão cronológica dos jogos que o influenciaram. Um outro exemplo de aplicação é a visualização aspectos mercadológicos ligados aos componentes dos jogos. Nesse sentido, empregando um cruzamento de dados de mercado e de crítica especializada, torna-se possível encontrar a popularidade, raridade ou sucesso de determinados componentes em jogos ou gêneros, de forma a responder a questões como: "Quais são as aplicações de maior sucesso de determinados componentes?", "Quais aquelas que tiveram insucesso?", "Quais as combinações peculiares que envolvem determinado componente?" ou "Quais os componentes comuns a um determinado grupo de jogos ou gêneros?". Ao tratar o conhecimento de design como diretrizes de alto nível de abstração, as abordagens prévias a este trabalho não puderam prever a realização de tais análises. É importante ressaltar que essas possibilidades de aplicações compreendem trabalhos futuros do autor desta tese, não estando inclusas em seu objeto de estudos.

2.3.2.4 Ambiente de Design de Jogos

A principal aplicação da abordagem de componentes prevista neste trabalho é a construção de um Ambiente de Design de Jogos. Fundamentando-se nos componentes de jogos como elementos estruturais para o design, espera-se que seja possível futuramente implementar um ambiente computacional que possa assistir na concepção de jogos, em especial nas atividades de análise, design e prototipação. O objetivo é ambiciosamente assistir o designer em seu ofício. Mais especificamente, espera-se permitir que o designer explore uma base de conhecimentos de componentes, estude a composição de jogos e gêneros, utilize ferramentas de análise para identificar relações com mercado e crítica, projete novos jogos e faça experimentações em protótipos gerados. Tomando como fundamento a abordagem de componentes de jogos desta tese, vê-se a possibilidade de construir um conjunto de recursos

de assistência ao design de jogos.

Como assistente de design, o ambiente poderá prover guias para auxiliar na concepção de novos jogos. Nesse sentido, o designer pode montar jogos com componentes registrados na base de dados, iniciar um novo projeto sobre *templates* de composições de gêneros ou gerações (ex: "jogo de plataforma dos anos 90"), ou mesmo, realizar combinações das composições de jogos existentes. Em um segundo momento, o designer poderá gerar um protótipo de

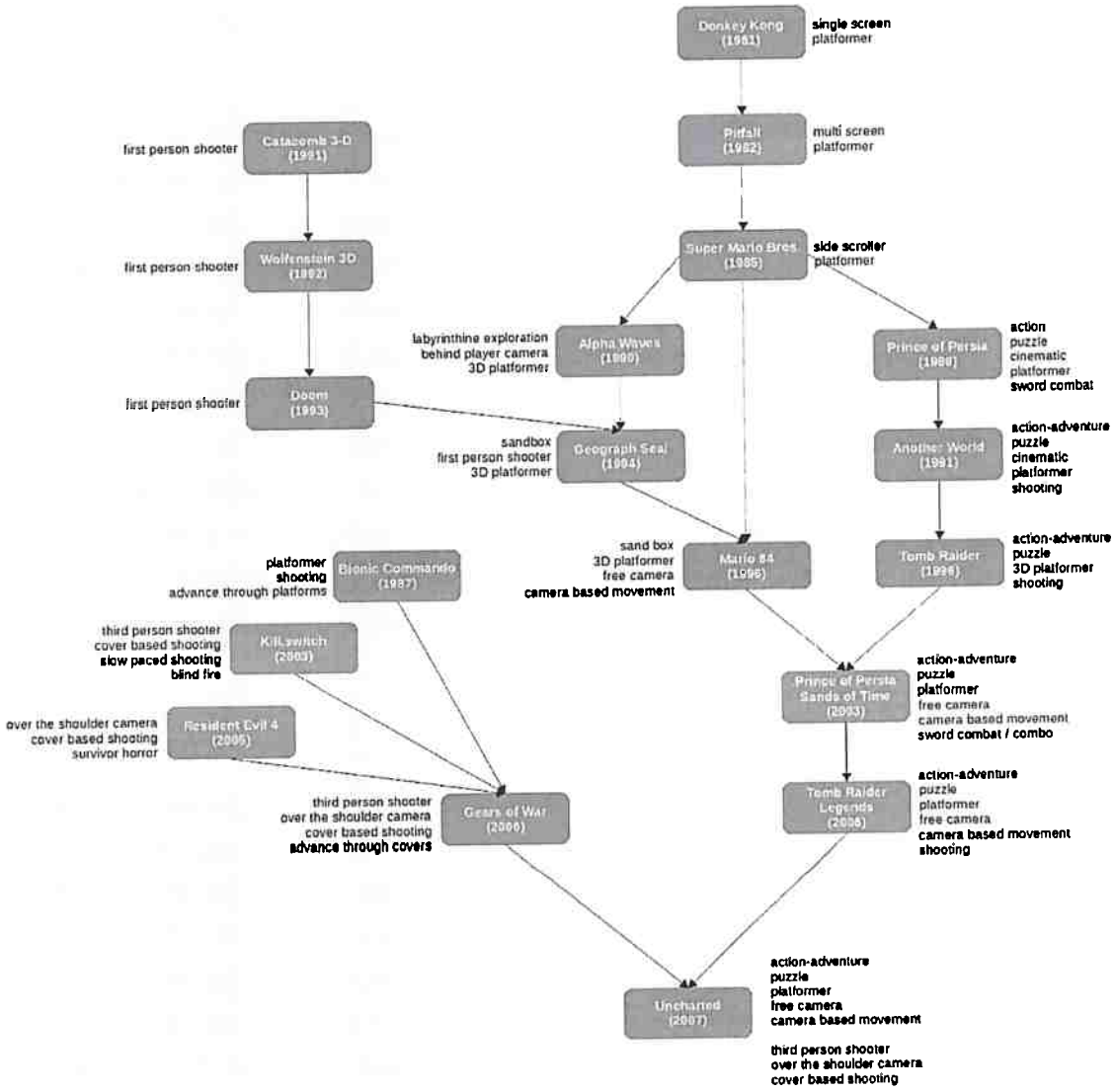

Figura 16 – Conceito de "árvore genealógica" da cronologia de jogos, baseada em seus componentes.

experimentação. Com o protótipo em execução, o ambiente pode registrar o status e as ações do jogador, além de outras métricas que possam fornecer retorno sobre a composição do jogo. Esse processo iterativo de design e testes estéticos facilita a investigação de diferentes combinações de componentes e o design incremental. Outra possibilidade refere-se ao emprego de assistentes inteligentes que forneçam sugestões com base na "mistura" de componentes do projeto.

O ambiente de design de jogos pode analisar os componentes utilizados pelo designer no conceito do jogo e sugerir alterações ou melhorias. Como exemplo, a partir da similaridade da composição do projeto com gêneros e outros jogos, o ambiente pode realizar sugestões de novos componentes para o projeto. Nesse sentido, ele pode sugerir componentes típicos aos gêneros detectados, componentes comumente relacionados aos que estão no design, ou mesmo, componentes presentes em títulos similares que tenham sucesso no mercado. De forma similar, pode advertir quanto à potenciais conflitos ou raridade nas composições realizadas. Além disso, com base em uma análise de padrões frente a composição de jogos similares, pode indicar características como falta ou excesso de desafios, itens, movimentos ou habilidades.

O leque de possibilidades de aplicações computacionais sobre a abordagem de componentes mostrasse bastante vasto. O conceito do Ambiente de Design de Jogos representa o resultado maior que poderá ser alcançado com base na conclusão do presente trabalho e torna-se a fusão entre design de jogos e técnicas de computação. Cabe ressaltar que o emprego de assistentes computacionais não é algo único deste trabalho e tem sido alvo de algumas pesquisas, notadamente no Grupo de Criatividade Computacional[83] da Goldsmiths University of London.

2.3.2.5 Sistematização do Processo de Design de Jogos

O modelo corrente de pensar e criar jogos é fundamentado na produção do Documento de Design (GDD). O processo de design é comumente direcionado para o cumprimento de cada uma das seções deste documento (Kreimeier, 2003). Nesse contexto, a abordagem de componentes definir elementos estruturantes para o design e adotar o princípio da componentização para a concepção de jogos. Assim, acredita-se que a abordagem não somente apresenta uma forma estrutura de documentação de aspectos de design, como também, pode ajudar a sistematizar o processo de design de jogos. Nesse sentido, ele passa a ser guiado pela análise, registro e modelagem de componentes. As quatro partes constituintes da abordagem de componentes, a serem apresentadas na seção 3.1 – estrutura, representação, conhecimento e ambiente –, juntamente como as práticas de análise por decomposição e design por composição podem ajudar a estabelecer os alicerces para uma sistematização do processo de design de jogos (Figura 17). Cabe ressaltar que a figura também esquematiza o funcionamento do Ambiente de Design de Jogos, conceituado na seção anterior.

[83] Computational Creativity Research Group, http://ccg.doc.gold.ac.uk/ (visitado em 09/02/2014)

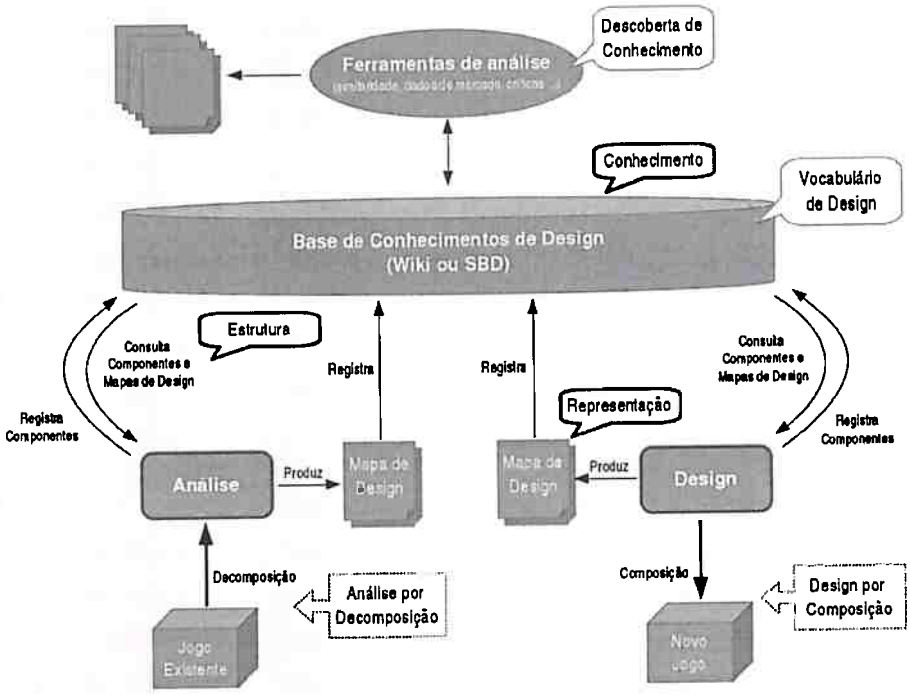

Figura 17 - Representação da sistematização do processo de design.

Na Figura 17, nota-se que os processos de análise e design são análogos quanto à sua realização, mas contrários em termos de suas entradas e saídas: a análise é feita sobre jogos existentes, ao passo que o design tem por objetivo construir o conceito de novos jogos. Em ambos, o designer registra e consulta o conhecimento na base sob a estrutura de componentes. A representação dos jogos ocorre pela linguagem e diagramas de componentes. Esse conhecimento registrado na base de dados servirá de matéria-prima para a descoberta de conhecimentos de design, realizada por meio do emprego de técnicas de mineração de dados.

Os processos de análise e design por componentização seguem um roteiro iterativo incremental de identificação e emprego de componentes. No design por composição, a definição do jogo segue um fluxo orientado "abstração → concretização", no qual parte de um conceito geral e segue à definição detalhada de seus componentes. Tal fluxo está representado na Figura 18. Utilizando um processo de refinamentos sucessivos, o designer de jogos trabalha por camadas gradativas de abstração. Em um primeiro momento, ele define os componentes abstratos que representam o conceito geral do jogo. Como exemplo, o designer pode partir de um esboço de jogo de plataforma, que pode ser caracterizado por um jogador [Player] que pula [Jump] sobre plataformas [Platform] e evita obstáculos [Obstacle]. Os componentes abstratos referem-se aos conceitos mais genérico de cada característica do jogo e representam os super tipos de cada hierarquia de componentes. Nesse momento, o designer não sabe ainda qual o tipo específico de [Jump] ou [Obstacle] a ser utilizado. Essas especificidades definem a identidade do jogo, que encontra-se nesse ponto como um título genérico de plataforma. À medida que o fluxo progride, as características do jogo passam a ser gradativamente detalhadas pelo designer, que então define o tipo concreto dos componentes abstratos previamente empregados e os relaciona para constituir o conceito do jogo. Esse processo iterativo serve tanto para o design de novos jogos quanto para a análise por decomposição de jogos existentes. No

entanto, a sequência de passos e o resultado final de cada iteração difere.

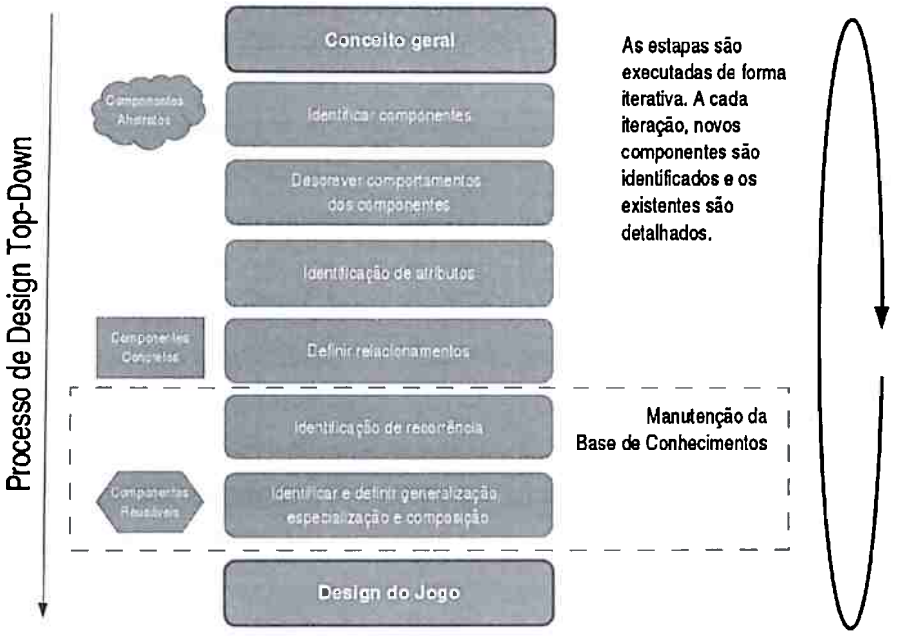

Figura 18 - Representação do fluxo iterativo do processo de design por composição.

Os processos de design por composição "Top-Down" e "Bottom-up" são análogos em termos do fluxo de execução e sua distinção ocorre no resultado de cada passo e iteração (Figura 18). No modelo "Top-Down", a cada etapa apresentada no esquema acima o designer concentra-se em descrever o conceito inteiro do jogo por meio de componentes, partindo de componentes abstratos e rumando à definição de detalhes que os tornem concretos e permitam relacioná-los. Dessa forma, a medida que o fluxo progride, as características dos componentes abstratos são detalhadas incrementalmente. Ao final de cada iteração, têm-se um conjunto de componentes concretos que descreve o jogo. O modelo "Top-Down" mostra-se mais adequado à construção gradativa em camadas do conceito do jogo. Já no modelo "Bottom-up" de design por composição, a cada iteração o designer concentra-se em definir um novo componente por completo, iniciando pelos componentes-chave no conceito do jogo. Nesse modelo, existe um maior foco na investigação do impacto do emprego do componente na constituição do jogo-alvo, bem como, da influência que exerce sobre outros jogos que o utilizam. Se comparado ao modelo "Top-Down", possui caráter mais analítico e relacionado ao design experimental de jogos. O resultado final de ambos modelos é o mesmo, mas diferem no foco do design: a construção do conceito do jogo ocorre por meio de uma combinação e experimentação gradual de componentes, investigados profundamente ("Bottom-up"); ou pela definição gradativa do jogo como um todo ("Top-Down"). O processo de análise pode decomposição é análogo a ambos modelos de design por composição. Como o jogo já existe e os detalhes de cada componente podem ser prontamente observados, o designer pode descrever gradativamente o conceito do jogo por meio de componentes concretos.

O nível de detalhamento dos diagramas de design pode variar conforme as necessidades e a cultura da equipe ou da organização. Em um contexto de metodologias ágeis, pode-se apenas descrever o conceito jogo por meio dos componentes-chave do *gameplay* (*core gameplay*) de

forma a documentar sinteticamente as características determinantes do jogo, em especial, aquelas diretamente relacionadas à interação do usuário com o jogo. Demais componentes podem ser documentados à medida que forem necessários. Nesse sentido, os diagramas de design podem estar organizados em camadas que agrupam componentes de acordo com sua influência no conceito do jogo e definem prioridades de implementação para o time de desenvolvimento. Dessa forma, a simples separação entre componentes fundamentais (*core gameplay components*), componentes secundários ou componentes contextuais (que ocorrem em determinados contextos ou situações) pode auxiliar desenvolvedores e artistas gráficos a organizar os recursos do jogo a serem construídos posteriormente. Ademais, os componentes do diagrama de design podem ser mapeados para definições de classes na arquitetura de software do projeto. Nesse contexto, o emprego da componentização de design contribui para o emprego de um vocabulário unificado na comunicação no time de desenvolvimento. Ao trata o design de jogos sob perspectiva da componentização, a abordagem acaba por sistematizar a criação de jogos como um processo guiado pela análise, registro e design de componentes.

2.4 Síntese da Abordagem

De forma resumida, a abordagem deste trabalho destaca-se das demais pelos pontos abaixo. Eles são apresentados e discutidos nos próximos capítulos:

- Definir os elementos estruturantes no design de jogos como componentes;
- Apresentar uma perspectiva de componentização para pensar, analisar, registrar e comunicar o design de jogos. Para tanto:
    - Segue o preceito de que os jogos podem ser tratados pelas suas partes constituintes, inter-relacionáveis e geralmente recorrentes em jogos independentemente de seu gênero;
    - Define o design por composição: jogos são "montados" a partir de componentes.
    - Analogamente, define a análise por decomposição;
- Estruturar, mapear e registrar o conhecimento de design embutido nos jogos existentes.
- Contribuir com a formação de um vocabulário compartilhado de design;
- Estimular a prática do design experimental de jogos, por meio de combinações de componentes;
- Apresentar uma abordagem com equilíbrio entre formalidade e praticidade, representando um elo entre as propostas de coleções de conceitos de design e as abordagens de mapas visuais;
- Definir uma linguagem para documentação de componentes e, por conseguinte, de jogos e gêneros:

  - A linguagem deve ser simples, intuitiva e apresentar uma estrutura sintática bem definida para possibilitar uma futura implementação no Ambiente de Design de Jogos;
  - Deve permitir uma mesclagem com documentação visual, na forma de mapas sintéticos de design;
- Estimular o registro e reúso de conhecimentos de design na forma de componentes e mapas de design de jogos;
- Criar possibilidades de descoberta de conhecimentos de design a partir da análise da constituição dos jogos e cruzamento de dados;
- Contribuir com a sistematização do processo de design;
- Demonstrar o emprego da abordagem em dois estudos de casos:
  - Utilizar análise por decomposição para extrair componentes de jogos existentes;
  - Utilizar design por composição para montar os conceitos de dois protótipos de jogos;
  - Construir protótipos dos jogos para demonstrar a aplicação dos componentes e possibilitar experimentações estéticas destes;
- Prover o instrumental conceitual para a criação de um Ambiente de Design de Jogos.

Capítulo 3 Design de Jogos Baseado em Componentes

A abordagem de design de jogos baseada em componentes é fundamentada no conceito de componentização, que trata o design de um jogo como uma soma de elementos estruturantes. Esses elementos – os componentes – são identificados pela perspectiva do jogador, apresentam inter-relações e são passíveis de recorrência em diferentes jogos. Nesse sentido, os componentes servem de blocos de construção para representação de jogos novos e existentes, além de gêneros. Dessa forma, o conhecimento de design presente nos jogos pode ser armazenado em uma base de conhecimentos estruturada de design.

A abordagem de componentes pode ajudar a sistematizar o processo de design de jogos à medida que define uma forma estruturada de pensar, analisar, registrar e projetar jogos. Para tanto, ela representa a compreensão do design de jogos como um processo baseado em engenharia, ao contrário de abordagens que enfatizam o emprego da narrativa. Nesse sentido, a abordagem de componentes de design inspira-se fortemente na **Orientação a Objetos** (Fowler, 2004), utiliza uma documentação estruturada para os componentes que se assemelha aos **Padrões de Projetos** (Gamma et al., 1994) e emprega a composição como construção-chave para a definição das partes dos jogos, assim como no padrão arquitetural de **Sistemas de Entidades e Componentes (C/ES)** (Leonard, 1999). Contudo, cabe ressaltar que em nenhum momento houve a intenção específica de ser trazer essas três construções ao contexto do design de jogos. O curso desta tese seguiu um rumo oposto, partindo de uma ideia embasada nas experiências prévias do autor e delineando-se pela influência de trabalhos relacionados a medida que seus conceitos se intersectaram.

O objetivo deste capítulo é apresentar, discutir e exemplificar cada um dos conceitos e construções utilizados na abordagem de design por componentes.

3.1 O escopo maior da abordagem de componentes

A abordagem de componentes foi inicialmente idealizada dentro de um escopo maior, como a união de quatro partes fortemente relacionadas: estrutura, representação, conhecimento e ambiente. Essas partes foram organizadas em dois grupos, que correspondem a definição do instrumental conceitual dos componentes e as ferramentas de software que podem apoiar designers de jogos a utilizá-los. Elas foram organizadas e definidas da seguinte forma:

1. Instrumental conceitual de componentes:
    a. Uma **estrutura** para descrição dos componentes de jogos e seus relacionamentos;
    b. Um mecanismo de **representação** dos componentes, jogos e gêneros por meio de linguagem textual e/ou visual definida especificamente para a abordagem.

2. Ferramentas de software:

    a. O gerenciamento do **conhecimento** de design por meio do registro dos componentes em uma base de dados, a fim de permitir acesso compartilhado, reuso e aplicação de técnicas de descoberta de conhecimentos;

    b. Um **ambiente** de design de jogos que forneça suporte computacional ao emprego da abordagem desta tese, permitindo definir e experimentar conceitos de jogos com base em componentes.

A fim de viabilizar a realização deste trabalho, o grupo de "Ferramentas de Software" não constituiu um objeto de estudos desta tese. Nele, o **conhecimento** e o **ambiente** definem implementações de ferramentas de software que estão fora do escopo deste trabalho e foram identificados como estudos futuros. Seus conceitos já foram apresentados e discutidos na seção 2.3.2. Dessa forma, esta tese trata especificamente do instrumental conceitual que fundamentará tais implementações, sendo eles a **estrutura** dos componentes e sua **representação** em jogos e gêneros. Eles são apresentados respectivamente nas seções 3.5 e 3.6.

3.2 Inspirando-se em classes e suas relações

O paradigma de Orientação a Objetos (OO) fundamenta-se na abstração de entidades do mundo real em elementos autônomos chamados de objetos (Fowler, 2004). Objetos gerenciam seus dados, realizam ações e comunicam-se uns com os outros. Para a OO, um programa é uma composição de objetos que interagem entre si, cada qual com uma responsabilidade específica (Lewis & Loftus, 2011).

As classes definem os tipos estruturais para os objetos. Elas possuem atributos, que as qualificam, e métodos, pelos quais podem executar ações e realizar trocas de mensagens. Os objetos são as instâncias de classes criadas quando o programa entra em execução. Nesse instante, os atributos dos objetos passam a assumir valores específicos que são utilizados por seus métodos. Por essa perspectiva, os objetos podem ser descritos como aplicações das classes em situações específicas. Dessa forma, por meio do paradigma de orientação a objetos, os programas são planejados, projetados e construídos como agrupamentos de classes e executados por suas instâncias. No contexto deste trabalho, um componente é uma parte de jogos que apresenta influência estética, isto é, sobre a experiência de *gameplay*. Da mesma forma que em um programa, nesta abordagem um jogo é uma composição de partes que interagem entre si. Nota-se, portanto, que há características comuns entre os conceitos de componentes de jogos e classes da OO. De fato, o modelo estrutural da abordagem de componentes inspira-se, mas não se limita, no paradigma de OO, apoiando-se essencialmente sobre o princípio de "Favorecer a Composição sobre a Herança" de Gamma et al. (1994), segundo o qual os comportamentos e o reuso de funcionalidades são atingidos por meio da composição de classes.

No contexto do paradigma de OO, as classes representam as estruturas genéricas dos objetos, que por sua vez, são as instâncias das classes criadas pelo programa quando entra em

execução. Uma classe pode ter um número indefinido de instâncias. Já no contexto deste projeto, um componente de jogo representa uma meta classe, uma generalização estrutural de uma classe, que se encontra fora do contexto de qualquer jogo. Quando utilizado em um design, o componente torna-se uma aplicação e passa a apresentar características específicas ao contexto do jogo. Em uma analogia à OO, a execução do jogo projetado representa uma instância desse design (Figura 19). Dessa forma, o design por componentes pode ser compreendido como uma descrição estrutural e comportamental do conceito de um jogo.

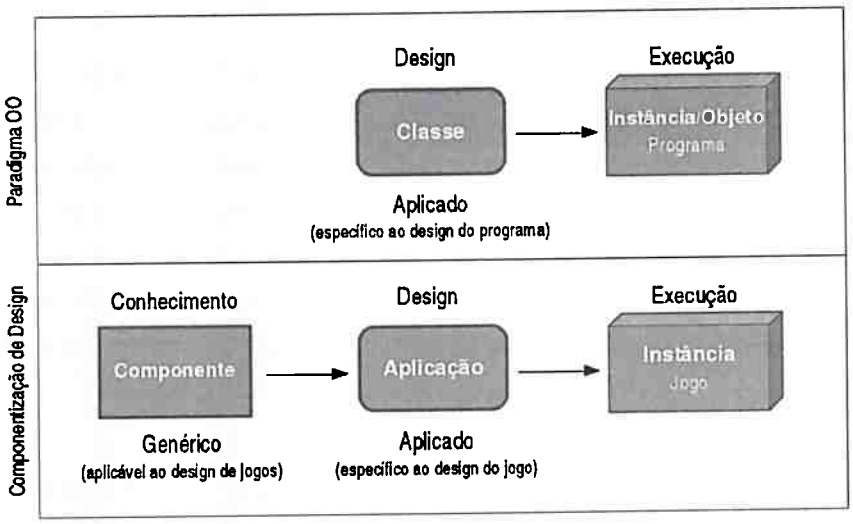

Figura 19 – Comparativo entre Classe x Objeto e Componente x Aplicação x Instância.

Um componente descreve uma parte ou característica de jogos, de forma distinta e genérica. Assim como nas classes, eles podem apresentar atributos que os qualificam, cujos valores variam de acordo com suas aplicações nos jogos. Quando empregado no design de um jogo, um componente torna-se uma aplicação específica, e sua instância ocorre quando o jogo é executado. Como exemplo, tomemos o componente [Jump], que tem sua estrutura apresentada na Figura 20. Nele, o "pulo" ([jump]) está documentado como um componente do tipo "ação" (<action>) contendo quatro atributos. Nota-se que o componente se encontra em uma forma mais genérica e livre de contexto, e seus atributos apresentam uma relação dos valores que podem assumir.

```
>> Movimento que trouxe verticalidade aos jogos 2D de visão lateral.
[Jump] <action> {
      controlable: yes/no/partial
      height: fixed/variable
      affected by motion: ground/air/both/no
      auto assisted: yes/no/prompted
}
```

Figura 20 – Representação textual do componente de ação "pulo".

Quando o componente [Jump] da Figura 20 é aplicado no design de jogos, seus atributos passam a ter valores fixos e podem ser acrescidos de considerações relacionadas ao contexto de sua aplicação nos jogos. A Figura 21 mostra um exemplo de aplicação do componente [Jump] no contexto do jogo Super Mario Bros. Contudo, em termos de aspectos de jogos, não existe uma distinção entre um "aspecto genérico" e um "aspecto aplicado". Na prática, é mais simples

comunicar um exemplo de pulo referenciando-se a um tipo específico (ex: "pulo do Mario" ou "pulo do Sonic"), em invés de tentar expressar a mesma ideia por meio de um componente de pulo genérico, detalhando os valores de seus atributos. Além disso, é conveniente descrever um conceito a partir de outro bem conhecido, como uma analogia. Dessa forma, as aplicações de componentes enriquecem o vocabulário de design e criam um recurso conveniente para a comunicação de ideias. Em termos práticos, uma aplicação é uma **especialização** (subtipo) de um componente para um determinado jogo ou situação. Por esta razão, não existe diferenciação representativa entre uma aplicação (mais específica) e um componente (mais genérico).

```
>> Mario Ele usa pulos para alcançar plataformas, atacar inimigos e
evitar armadilhas e obstáculos.
[Mario Jump] <jump> {
      [Jump] {
            controlable = partial (influenced by inertia)
            height = variable (by button press)
            affected by motion = both (by character speed)
            auto assisted = no
      }
}
```

Figura 21 – O componente [Mario Jump] é uma especialização de [Jump] para o contexto do jogo Super Mario Bros.

O exemplo da Figura 21 mostra o componente [Mario Jump], uma especialização de [Jump] para o contexto específico do jogo Super Mario Bros. Esse componente é definido a partir do mecanismo de classificação, tornando-o um tipo mais especializado (um subtipo) de [Jump]. O [Mario Jump] caracteriza-se como sendo controlável pelo jogador, que pode mover o personagem mesmo quando estiver no ar, possui altura variável definida pelo tempo de pressionamento do botão e pode ser afetado pelo movimento do personagem, o que implica que o pulo torna-se mais alto e longo se executado durante uma corrida. Os mesmos atributos são observados no pulo do personagem Sonic. Dessa forma, o [Sonic Jump] pode ser tanto representando como uma especialização de [Jump], quanto de [Mario Jump] (Figura 22), uma vez que a classificação embute implicitamente o componente tipificador. Além disso, caso desejado, pode-se descrever um novo tipo de pulo de uma a partir da alteração de atributos do [Mario Jump]. De uma forma geral, o mecanismo de classificação torna-se um recurso versátil para a organização hierárquica de tipos, assim como, para a descrição de aspectos por analogias. Ela é melhor discutida na seção 3.7.3.

```
>> Pulo do Sonic representado como uma especialização de [Jump]:
[Sonic Jump] <jump> {
      [Jump] <action> {
            controlable = partial (influenced by inertia)
            height = variable (by button press)
            affected by motion = both (by character speed)
            auto assisted = no
      }
}
>> Pulo do Sonic representado como uma especialização de [Mario Jump]:
[Sonic Jump] <mario jump> {
      [Mario Jump] <jump>
}
```

Figura 22 – Representações para o pulo do Sonic.

Um exemplo análogo ao apresentado no parágrafo anterior refere-se à aplicação do componente [Jump] nos jogos 2D das franquias Castlevania e Ninja Gaiden, representado como [Simon Jump] na Figura 23. Nesses jogos o pulo segue uma trajetória fixa e invariável, não podendo ser controlado no ar. Assim como no [Mario Jump], o [Simon Jump] não utiliza assistência de pulo, recurso que é comumente encontrado em jogos 3D que empregam plataformas.

```
>> Movimento complementar ao ataque de Simon. O pulo de Simon serve
tanto para navegar pelo level quanto para permitir ataques aéreos. Por
ser um pulo fixo e sem controle, pune o jogador por usá-lo de maneira
incorreta.
[Simon Jump] <jump> {
      [Jump] <action> {
            controlable = no
            height = fixed
            affected by motion = no
            auto assisted = no
      }
}
```

Figura 23 - Representação textual da aplicação [Simon Jump].

Como visto nos exemplos apresentados, o componente [Jump] passa a ter seus atributos definidos quando inserido no contexto de jogos específicos, tornando-se um tipo mais especializado. Classes, por outro do lado, não possuem valores de atributos definidos. Isso somente ocorre quando suas instâncias – os objetos – são criadas durante a execução do programa. Os atributos de componentes possuem um significado contextual, ao contrário do que ocorre nas classes, nas quais os atributos representam controles necessários ao funcionamento da lógica implementada nos métodos. Nesse sentido, os designers de jogos podem utilizar os atributos para descrever especificidades do comportamento do um componente quando aplicado no contexto de um do jogo ou situação. Dessa forma, a aplicação de um componente pode ser considerada o emprego do conhecimento estruturado de design.

```
[Attack Jump] <jump> <attack> {
      [Mario Jump] <jump>

      [Melee Attack] <attack> {
            repeatable: yes/no/combo
            combo: <combo>
      }
}
```

Figura 24 - Representação de [Attack Jump] utilizando [Mario Jump].

Observando os exemplos apresentados, nota-se que as representações da definição do componente [Jump] e suas aplicações em contextos de jogos [Mario Jump], [Sonic Jump] e [Simon Jump] utilizam a mesma sintaxe. Isso ocorre porque todos são considerados partes reusáveis de jogos e podem ser empregados em outros componentes. Como exemplo, a Figura 24 mostra a definição do componente [Attack Jump], que denota um movimento composto pela ação de pular e atacar. Nesse exemplo, espera-se que a ação de pulo possua o mesmo comportamento que a encontrada no jogo Super Mario Bros. e, portanto, a aplicação [Mario Jump] é utilizado. De forma análoga, o componente [Simon Jump] poderia ser utilizado para

descrever um comportamento de pulo diferente. O segundo componente utilizado no [Attack Jump] é o [Melee Attack]. Ele representa a ação de realizar um ataque de curto alcance, como um soco, chute ou golpe com arma branca. O componente está modelado com dois atributos, "repeatable", que define se o ataque pode ser repetido com frequência rápida, e a relação "combo", que pode receber um componente do tipo <combo>. Este atributo tem a função de permitir que o ataque de curto alcance seja executado de forma encadeada com outros. Por fim, uma definição alternativa para [Attack Jump] utilizada o componente [Jump] com a especificação de seus atributos (Figura 25).

```
[Attack Jump] <jump> <attack> {
      [Jump] <jump> {
            controlable = partial (influenced by inertia)
            height = variable (by button press)
            affected by motion = both (by character speed)
            auto assisted = no
      }
      [Melee Attack] <attack> {
            repeatable: yes/no/combo
            combo: <combo>
      }
}
```

Figura 25 - Representação alternativa de [Attack Jump] utilizando [Jump].

A modelagem por meio do mecanismo da composição emprega componentes que foram previamente documentados. Os **componentes internos**, aqueles usados em composições, podem ter seus atributos com valores definidos. Neste caso, não há obrigatoriedade em apenas repeti-los. De fato, componentes e atributos que sejam adquiridos como consequência da aplicação da composição ou da classificação não precisam estar explicitamente reapresentados. Isso geralmente é feito nos exemplos desta tese para facilitar sua legibilidade, uma vez que estes encontram-se fora do contexto da modelagem completa de um jogo. Como exemplo, as Figuras Figura 24 e Figura 25 explicitam os atributos do componente [Melee Attack], embora espera-se que estes já tenham sido descritos anteriormente. Por outro lado, outros exemplos presentes nesta tese poderão apresentar componentes internos de forma mais sintética, a fim de simplificar a descrição textual. Nesse sentido, a Figura 26 mostra uma representação resumida do mesmo componente [Attack Jump].

```
[Attack Jump] <jump> <attack> {
      [Mario Jump] <jump>
      [Melee Attack] <attack>
}
```

Figura 26 - Representação sintética do componente [Attack Jump] utilizando [Mario Jump].

A relação existente entre a estrutura dos componentes e o paradigma de Orientação a Objetos continuará sendo discutida na próxima seção. Como pôde ser notado nos exemplos apresentados nesta seção, os componentes apresentam relações que agem como mecanismos de construção. Nesse sentido, **composição**, **associação** e **classificação** representam recursos fundamentais para a modelagem de componentes. Essas relações demonstram uma forte influência da OO sobre a abordagem e serão discutidas em mais detalhes na seção 3.7. Esses mecanismos permitem modelar os aspectos de jogos como analogias de aplicações de

componentes no contexto de jogos específicos. Enquanto os componentes são usados essencialmente para descrever a estrutura e o comportamento das partes formadoras dos jogos, as aplicações determinam as particularidades de seu uso. Nesse contexto, a **análise por decomposição** de um jogo pode ser descrita como um processo de engenharia reversa: em um primeiro momento, o designer identifica as aplicações de componentes no jogo, para então, reconhecer sua generalização e documentá-la como um componente. No **design por composição** o processo é análogo: o conceito do jogo é formado a partir da junção de componentes, que passam a ser aplicados à medida que o conceito do jogo evolui.

3.3 Diferenças entre classes e componentes

No paradigma de Orientação a Objetos, as entidades e processos do mundo real são abstraídos em classes. As classes definem os tipos para os objetos que, por sua vez, são agrupados para formar um software. Se esse princípio fosse diretamente aplicado no contexto da componentização de design, as entidades do mundo do jogo deveriam ser necessariamente mapeadas para componentes. Exemplos dessas entidades compreendem inimigos, veículos, tiros, explosões ou mesmo partes do ambiente, como edifícios, árvores ou pedras. No entanto, nem todas essas entidades apresentam influência sobre a experiência de *gameplay* em um jogo. Por outro lado, a presença de um movimento de pulo em jogo de plataformas, um recurso de auto-mira ("auto-aim") em um jogo de tiro em terceira pessoa ou um ataque giratório, como o do personagem Link da série Legend of Zelda, são aspectos que influenciam e definem fundamentalmente o *gameplay* de um jogo. Dessa forma, diferentemente de classes e objetos, componentes são abstrações de aspectos constituintes de jogos.

O paradigma de Orientação a Objetos define que classes possuem atributos e métodos. Os atributos caracterizam e qualificam os objetos a serem instanciados a partir das classes. Os métodos contêm a lógica necessária para a realização de ações. Dessa forma, classes podem ser descritas como estruturas formadas essencialmente por atributos e ações de entidades abstraídas do mundo real. Por outro lado, os componentes de design representam aspectos de jogos e seus atributos. Esses aspectos podem compreender entidades, ações ou características que podem ser usados para compor conceitos de jogos. Além disso, no projeto de um software, a modelagem de classes está diretamente relacionada a implementação deste. Em oposição, a abordagem de componentes está preocupada em organizar e reusar ideias e conceitos de design, não tendo relação a implementação do software que regerá o jogo. Nesse sentido, os componentes representam conceitos estruturados que podem ser aplicados no design de vários jogos. Exemplos apresentando essas diferenças são discutidos no texto que segue.

No contexto da OO, um "pulo" seria abstraído como uma ação da entidade "personagem". Por conseguinte, seria provavelmente modelado como parte de um método da classe Personagem. Dessa forma, ele não existiria como uma parte independente. Semelhantemente, aspectos como "auto-mira", "vitalidade regenerativa" ou "ataque giratório" seriam também mapeados para métodos de classes, ou mais comumente, tratados sob um único método de atualização de lógica da entidade. No entanto, o "pulo" é um aspecto que

influencia profundamente a experiência de *gameplay*. Como exemplo, a ausência do movimento de "pulo" em um jogo de "terror e aventura" ("horror-adventure") leva à sensações de restrição e tensão, que são respostas emocionais desejadas no jogador neste tipo de gênero. Já em jogos de "ação" ou "plataforma", a mesma ausência pode levar à frustração do jogador, prejudicando o controle direto e imediato do personagem necessário à tais gêneros. Ademais, a remoção ou inclusão da ação de "pulo" no personagem do jogador traz consequências fundamentais para o design do jogo como um todo, influenciando em elementos como narrativa, construção dos níveis (level design) e comportamento dos personagens não jogáveis (NPC – Non-Player Character). Nesse sentido, um "pulo" é um aspecto distintamente reconhecível de um jogo que traz significado estético quando inserido ou removido em seu design. Dessa forma, diferentemente de uma abordagem OO estrita, o [Pulo] é documentado como um componente de jogo, assim como a [Auto-Mira], a [Vitalidade Regenerativa] e o [Ataque Giratório]. Ao contrário dos métodos de classes, esses componentes possuindo seus próprios atributos, tipos e definições, e agem como partes autônomas que podem ser empregadas na composição de outros componentes.

A fim de ilustrar de forma pragmática as diferenças entre os paradigmas de OO e o design baseado em componentes, as figuras Figura 27 e Figura 28 apresentam a modelagem de uma entidade de jogo sob as duas abordagens. Nelas, a entidade "Warrior" (guerreiro) modela um personagem controlado pelo jogador em um jogo de plataformas de visão lateral, que se move em duas direções, pula e ataca. A Figura 27 apresenta a definição de uma classe para essa entidade seguindo uma abordagem tradicional de OO. Nesta, os atributos representam os valores manipulados pelos métodos, que implementam a lógica de funcionamento e desenho da entidade. Nota-se que a de definição da classe é orientada para a entidade e o seu funcionamento no jogo. Além disso, existe um foco na definição de aspectos da implementação, uma vez que a classe não representa os conceitos de design, mas sim, uma estrutura que abstrai a entidade do mundo jogo que possa ser mapeada para o projeto do software. O exemplo contrário é mostrado na figura seguinte.

```
class Warrior {
      // atributos
      healthAmount: float
      attackPower: float
      jumpHeight: float
      onGround: boolean
      alive: boolean
      // metodos
      init()
      getInput() // processa entradas do jogador
      updateLogic() // usa doJump() e doAttack()
      render()
}
```

Figura 27 - Modelagem de entidade Warrior sob uma abordagem OO estrita.

A Figura 28 mostra a entidade Warrior modelagem como um componente. Este está classificado (tipificado) como uma entidade (<entity>) controlada pelo jogador (<player>), o que faz com que inclua implicitamente os componentes [Entity] e [Player]. Além destes, a

composição também contém os componentes de auto regeneração, ataque curto direto, pulo duplo e movimento unidimensional (para frente e para trás). Nota-se na modelagem apresentada que, ao contrário de uma classe, existe um foco na estruturação e representação de aspectos de *gameplay* por meio de partes menores autônomas, que podem ser usadas em outras composições. O design do personagem é descrito pelos mecanismos da abordagem: nomenclatura dos componentes, atributos, composição e classificação. Destes, a composição tem papel fundamental e é por meio dele que os aspectos de influência estética – os componentes – são agrupados para formar outros maiores. Cabe ressaltar que a modelagem de componentes é um instrumento de design e não possui relação com a implementação do software do jogo.

```
[Warrior] <entity> <player> {
      [Auto-Regenerative Health Bar] <health> {
            type = regenerate health point
            when = not damaged

            [Health Bar] <hud bar> {
                  type = health points
                  visible = yes
            }
      }
      [Straight Attack] <melee attack> {
            repeatable = yes
            combo = none
      }
      [Double Jump] <jump> {
            timed = no
      }
      [Move in Two Directions] <movement> {
            tight controls = yes
      }
}
```

Figura 28 - Modelagem de entidade "Warrior" sob a abordagem de componentes de design.

Como foi apresentado até o momento neste capítulo, o modelo de Orientação e Objetos foi usado como inspiração para a definição das construções da abordagem de componentes. Em especial, ela apoia-se no princípio da composição. Contudo, a abordagem não segue estritamente as regras da OO. Cabe ressaltar que a abordagem de componentes não partiu da intenção de aplicar o paradigma de OO ao design de jogos. A relação foi oposta. Desde o princípio deste trabalho, a intenção era estruturar um vocabulário de conceitos de design de forma a usá-lo como ferramenta de concepção de jogos. A construção de composição foi adotada de forma empírica e trouxe consigo o paradigma de objetos ao contexto do trabalho. Além da OO, o padrão arquitetural de Sistemas de Entidades e Componentes também influenciou este trabalho.

3.4 Sistemas de Entidades e Componentes (C/ES)

Em 1998 a Eidos Interactive publicou o jogo Thief: The Dark Project[84], desenvolvido pela

[84] https://en.wikipedia.org/wiki/Thief:_The_Dark_Project

antiga Looking Glass Studios[85]. A equipe de produção incluiu os designers e programadores Doug Church, o autor das FADTs (Church, 1999), e Marc LeBlanc, um dos autores do framework MDA (LeBlanc et al., 2004). Seus trabalhos já foram discutidos no Capítulo 2 e, assim como a abordagem de componentes desta tese, apresentam uma perspectiva "engenheira" para o design de jogos. O discurso original de Church, de que "*precisamos poder dissecar os jogos em suas partes, compreender como funcionam e reusá-las em novos projetos*", mostra influências no trabalho posterior de LeBlanc. Além disso, o jogo acima mencionado, Thief, é aparentemente o primeiro a empregar uma arquitetura de software de C/ES – Sistemas de Entidades e Componentes (Leonard, 1999). Nesse contexto, é importante ressaltar que o presente trabalho se alinha ao discurso de Church e teve uma coincidente semelhança com a arquitetura de C/ES.

Um modelo de Sistemas de Entidades e Componentes (Component/Entity Systems) é um padrão arquitetural de software utilizado no desenvolvimento de jogos. Publicações que apresentam discussões e implementações deste tipo de arquitetura têm sido encontradas fora do escopo acadêmico, especialmente em blogs (Lord, 2012a; Lord, 2012b; Martin, 2007; Davies, 2011), wikis[86] e relatos de desenvolvimento apresentados em conferências da área (Bilas, 2002; Church, 2002). Um C/ES segue o princípio de "favorecer a composição sobre a herança" (Gamma et al., 1994), segundo o qual os comportamentos e o reúso de funcionalidades são atingidos por meio da composição e não pela herança, ao contrário do que ocorre no modelo tradicional de OO. No modelo C/ES, os objetos que compõem as cenas dos jogos são tratados como entidades, tais como personagens, obstáculos, itens, veículos, portas, armas e projéteis. Essas entidades funcionam como contêineres de componentes. Estes, por sua vez, implementam os aspectos do jogo, como comportamentos, funcionalidades, dados, aparência e física. Os componentes são então adicionados às entidades para lhes conferir seus aspectos. Dessa forma, as entidades são definidas pelos componentes que utilizam. Em contrapartida, no modelo tradicional da OO, as entidades de jogos adquirem comportamentos tipicamente por tornarem-se subtipos de outras entidades. O padrão C/ES apresenta uma influência considerável no desenvolvimento de jogos contemporâneo. Como exemplo, três dos motores mais utilizados no mercado – Unity[87], Unreal[88], Cryengine[89]– implementam uma variação do C/ES.

O padrão C/ES surgiu para solucionar um problema comumente encontrado no desenvolvimento de jogos que possuem muitos tipos diferentes de entidades. Nesses jogos, a árvore hierárquica de classes necessária para acomodar tamanha variedade torna-se de difícil gerenciamento. Relatos do desenvolvimento de títulos como Thief (Leonard, 1999) e Dungeon Siege (Bilas, 2002) citam a necessidade da criação de milhares de tipos diferentes de entidades. Segundo esses relatos, ao empregar a herança como instrumento de diversificação de entidades, cria-se um número proporcional de classes distintas para acomodar tais variações. Dessa forma, a árvore hierárquica de generalizações e especializações entre classes torna-se

[85] https://en.wikipedia.org/wiki/Looking_Glass_Studios
[86] http://entity-systems.wikidot.com
[87] http://unity3d.com
[88] https://www.unrealengine.com
[89] http://www.crytek.com/cryengine

muito complexa, ficando repleta de conflitos de heranças múltiplas. Em contrapartida, o emprego da composição como instrumento de definição de comportamentos e características permite flexibilizar a customização de entidades sem a complexidade hierárquica de uma abordagem OO tradicional. Ademais, os motores de jogos tendem a oferecer editores e linguagens especializadas que permitem que os próprios designers manipulem as entidades do jogo por meio de seus componentes.

A Figura 29 mostra um exemplo simples das diferenças entre a modelagem de entidades utilizando herança e composição. No primeiro caso, os comportamentos estão representados nos métodos das classes. A fim de diminuir o acoplamento e facilitar o reúso de comportamentos, as classes que implementam aspectos básicos são definidas no topo da hierarquia. A medida que comportamentos mais específicos são necessários, a hierarquia vai sendo gradativamente incrementada com a criação de classes mais especializadas. Os problemas no emprego da herança começam a surgir quando se torna necessário criar combinações de comportamentos que estão inseridos em classes pertencentes a diferentes ramificações da hierarquia. Além disso, mudanças na estrutura hierárquica das classes também se tornam problemáticas, podendo causar modificações em cascata.

A parte inferior do esquema da Figura 29 apresenta dois exemplos de entidades que combinam comportamentos cruzados de outras já presentes na hierarquia. Nesse contexto, como criar entidades de Destructable Spikes ("Espinhos Destrutíveis") e Homing Asteroids ("Asteroides Perseguidores") que reutilizam os comportamentos presentes nas entidades da hierarquia? Certamente em um exemplo pequeno como o apresentado, o problema não é agravante e mudanças na hierarquia das classes serão suficientes, seja pela fusão de classes existentes ou pela criação de classes intermediárias. No entanto, em projetos de jogos com milhares de tipos diferentes de entidades, modificar a hierarquia de classes das entidades torna-se um processo crítico e custoso. Por essa razão, o padrão arquitetural de C/ES sugere que os aspectos implementados nos métodos das classes sejam definidos como componentes, a fim de que possam ser adicionados ou removidos de quaisquer entidades. Nesse sentido, o lado direito da Figura 29 apresenta a modelagem das duas entidades citadas – Destructable Spikes e Homing Asteroids – utilizando o padrão C/ES. Nesta, é possível notar que os aspectos de renderização (Renderer), colisão (Collider), bem como, o comportamento de "mostrar e esconder" (ShowHide) tornaram-se componentes que foram adicionados à entidade Destructable Spikes. O mesmo pode ser observado para os componentes que passaram a compor a entidade Homing Asteroid. Em ambos exemplos, é possível notar que aspectos de implementação do jogo ficam explícitos nos componentes.

O padrão de C/ES refere-se especificamente a uma solução arquitetural para implementação do software de jogos ou motores, não sendo discutido no contexto do design de jogos. Em uma arquitetura de jogos baseada em herança, o código é organizado em uma estrutura tradicionalmente orientada a objetos. Nesse sentido, os dados das entidades (atributos) e o código de seus comportamentos e aspectos (métodos) segue dentro das classes. Em contrapartida, a arquitetura C/ES utiliza uma abordagem orientada a dados, similar a um banco de dados. Nela, as entidades são implementadas como um simples número identificador (ex: um número inteiro). Esse número é usado para relacionar os componentes a uma dada entidade. Os componentes, por sua vez, são responsáveis por armazenar dados. O acesso aos componentes relacionados a uma entidade é feito por uma busca, utilizando o número identificador. O código que manipula os dados dos componentes para cada uma das entidades

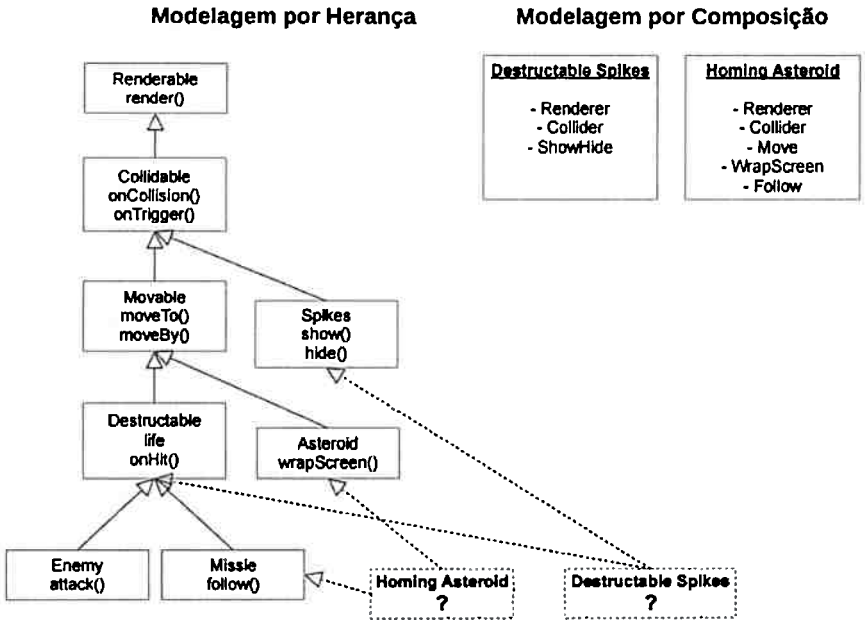

Figura 29 - Comparativo simples de modelagem por herança e composição.

fica disposto nos sistemas. Dessa forma, a arquitetura C/ES utiliza uma execução transversal: um sistema executa sobre um ou mais componentes de um determinado tipo, para todas as entidades do jogo. A arquitetura C/ES favorece a execução paralela, pois sistemas independentes podem ser processados concomitantemente em núcleos distintos da CPU (Unidade Central de Processamento).

Em linhas gerais, o foco prático da abordagem C/ES tem sido permitir que o designer tenha uma alta flexibilidade na definição dos aspectos e comportamentos das entidades dos jogos, sem que haja necessidade de modificar o código fonte do jogo. Motores de jogos populares no mercado implementam uma variação da arquitetura de C/ES a fim de permitir essa facilidade. Como exemplo, a Unity 5 permite que o designer monte as entidades dentro do ambiente do motor a partir de componentes previamente construídos em linguagem de programação, pelos programadores. De forma similar, a Unreal 4 permite que os programadores criem componentes que possam ser relacionados em "Blue Prints", uma espécia de diagrama de programação visual, utilizados pelos designers para definir as entidades do jogo. Nota-se,

portanto, um foco em desenvolvimento e não em design do jogo, em oposição a abordagem de design de jogos orientada a componentes, objeto de estudos desta tese.

A similaridade da abordagem de design por componentes com a C/ES foi uma coincidência, não tendo relação intencional. O modo de pensar os jogos como coleções de partes que podem ser desmembradas e reagrupadas em novos conceitos veio essencialmente da experiência profissional do autor desta tese. Contudo, essa similaridade mostra que a abordagem de componentes não é algo inusitado e divergente das ideias de outros desenvolvedores. Pelo contrário, alinha-se ao discurso de Church (1994), apresenta relações com o MDA de LeBlanc et al. (2004) e segue o princípio de favorecer a composição sobre a herança de Gamma et al. (1994), que serve de base para a C/ES. Contudo, a C/ES trata especificamente de problemas da engenharia do software do jogo, enquanto a abordagem de componentes, refere-se a estruturar o design de jogos, sendo desprovida de relações com a implementação do mesmo. Nesse sentido, cabe ressaltar que os problemas que a C/ES e a abordagem de componentes tentam resolver são bastante distintos.

De uma forma geral, a arquitetura C/ES tem por objetivo flexibilizar a customização de entidades do jogo, evitando a geração de uma complexa hierarquia de classes para acomodar tais variações. Em contrapartida, a abordagem de design baseado em componentes propõe um modo estruturado de pensar, projetar, registrar e comunicar o design de jogos. Dessa forma, ela pretende ajudar a padronizar termos de design, facilitar sua comunicação, estruturar e reusar o design de jogos e armazenar o conhecimento na forma de componentes e suas relações. Em termos estruturais, elas diferem pelo fato de que na C/ES, o jogo é desenvolvido como uma coleção de entidades, que utilizam os componentes para lhes conferir determinados aspectos e componentes. Já na abordagem deste trabalho todos os aspectos do jogo são tratados como componentes, que podem ser agrupados para formar outros mais complexos, estruturando uma coleção de conceitos de design. A estrutura de documentação dos componentes será apresentada na próxima seção. Em seguida, são discutidos os mecanismos de representação dos componentes.

3.5 A Estrutura de Documentação dos Componentes

Nas seções anteriores deste trabalho foram apresentados exemplos de componentes e suas aplicações. A **representação** utilizada nesses exemplos é bastante prática e sintética, e mais adequada para enfatizar a composição. Ela corresponde a um dos campos da **estrutura** de documentação de componentes, que é discutida nesta seção. Retomando os pontos apresentados na seção 3.1, a **estrutura** e a **representação** de componentes definem as duas partes constituintes do instrumental conceitual de design que é objeto de estudos deste trabalho.

Assim como no trabalho de Game Design Patterns de Björk, Lundgren e Holopainen (Björk et al. 2003), esta abordagem adota um estilo de documentação estruturada para os componentes, seguindo um formato similar ao apresentado por Gamma et al. (1994). Nesse sentido, os componentes são descritos por campos específicos, que descrevem aspectos

relevantes ao seu significado, constituição e emprego. Esta estrutura fundamentará a implementação futura de sistema de banco de dados que proverá suporte computacional ao armazenamento do conhecimento de design, documentado sob a forma de componentes. A ideia de se montar uma base de dados de conceitos de design de jogos certamente não é algo exclusivo da abordagem de componentes. Todos os trabalhos enquadrados sob o grupo de "Vocabulário Comum de Design", disposto na parte superior do mapa da Figura 4, têm a mesma intenção. No entanto, as implementações mais bem-sucedidas de uma base de conhecimentos de design de jogos recaem sobre aquelas mantidas por comunidades de usuários finais, que não demonstram preocupação em manter uma estrutura formal. Os sites Giant Bomb[90] e TV Tropes[91] mantêm bases de dados online de conceitos de jogos mantidas por usuários. Embora informal, muitos dos conceitos documentados apresentam fortes relações com componentes identificados em jogos durante a realização deste trabalho. De forma similar, a Wikipedia[92] contém a documentação de diversas características de jogos, que também apresentaram similaridades a alguns componentes identificados. Por essa razão, a estrutura da documentação de componentes contém um campo para referências ao conceito tratado pelo componente em fontes externas. Ademais, essas bases serão por vezes indicadas como referência para aspectos de design discutidos neste trabalho. Cabe reiterar que a existência dessas bases de dados online confirma a importância de se criar catálogos compartilhados de conceitos de jogos. Além disso, as relações entre conceitos catalogados nessas bases e os componentes identificados ao longo desta tese ajuda a atestar a existência do segundo como um aspecto reconhecível de jogos.

O quadro a seguir enumera os campos de documentação dos componentes, que são apresentados e discutidos na sequência.

Estrutura de Documentação do Componente

- Nome
- Aliases (outros nomes)
- Descrição
- Tipo (classificação)
- Atributos
- Ícone / Ilustração
- Exemplos de uso
- Jogos que usam
- Gêneros que usam
- Representação (Composição)
- Características estéticas
- Referências Externas

[90] http://www.giantbomb.com/concepts/ (visitado em 03/02/2016)
[91] http://tvtropes.org/ (visitado em 03/02/2016)
[92] https://www.wikipedia.org/ (visitado em 03/02/2016)

Nome: O nome de um componente deve ser claro e curto. Ele deve servir como um identificador universal para o componente, ajudando designers e desenvolvedores a identificá-lo, compreendê-lo e comunicá-lo. Idealmente, o conjunto formado pelos nomes dos componentes poderá contribuir com a formação de um vocabulário padronizado de design. Exemplos de nomes de componentes já apresentados em exemplos anteriores compreendem: [Double Jump], [Mario Jump], [Melee Attack], [Collectible], [Move Forward and Backward], [Regenerating Health] e [Auto-Aim].

Aliases: Apresenta nomes alternativos pelos quais o componente pode ser comumente referenciado. Nomear um componente para se adequar ao contexto da aplicação em um jogo ou gênero pode facilitar sua compreensão. Ex: o componente [Throw Object] pode ser melhor identificado como [Throw Shell on Mario] quando está dentro do contexto do jogo Super Mario Bros.

Descrição: O campo descrição deve conter um texto breve que discorrerá sobre as características da apresentação do componente de forma a explicitá-lo, atuando como uma complementação ao campo nome na identificação daquele.

Tipo: Lista os rótulos <tags> que classificam o componente. O mecanismo de classificação é discutido na seção 3.7.3.

Atributos: Lista e descreve os atributos definidos para o componente. Também apresenta e discute os valores possíveis para cada atributo.

Ícone/Ilustração: Este campo deve conter uma imagem que ilustre o componente. Pode-se utilizar a foto de um jogo que o contém, mostrando um instante de ocorrência do componente.

Exemplos de uso: Discorrer ou enumerar sobre alguns exemplos significativos de emprego do componente em jogos e/ou gêneros.

Jogos que usam: Apresentar lista abrangente dos jogos que empregam o componente, opcionalmente apresentando observações da aplicação. Em caso da existência de um sistema de banco de dados para gerenciar a base de conhecimentos de design sob esta estrutura, poderia apresentar links para seções de descrições dos jogos. O mesmo vale para o campo seguinte.

Gêneros que usam: Apresentar lista abrangente dos gêneros que empregam o componente, opcionalmente apresentando observações da aplicação.

Representação: Este campo deve apresentar o componente segundo a linguagem de representação (textual e/ou diagrama) definida para a abordagem. Mais informações podem ser encontradas na seção 3.6.

Características Estéticas: Este campo apresenta as consequências e características estéticas referentes a aplicação do componente em jogos e gêneros. As características podem ser

discutidas na forma de diretrizes de aplicação, indicando direções a serem tomadas ou evitadas.

Referências Externas: Neste campo são enumeradas referências ao conceito definido pelo componente em outras fontes. Exemplos dessas fontes são os wikis dos projetos Game Design Patterns[93] e Games Ontology[94], bem como, as bases de dados de conceitos de jogos disponíveis nos sites Giant Bomb e TV Tropes.

Exemplos do emprego dessa estrutura para a documentação de componentes são apresentados nas próximas seções. A seção seguinte discorre sobre a representação de componentes.

3.6 Representação de componentes

A **representação** de componentes já foi parcialmente apresentada nos exemplos de componentes presentes nas seções anteriores. Esses exemplos utilizam uma notação textual definida especificamente para a abordagem. A construção dessa linguagem foi influenciada tanto pelo paradigma de **Orientação a Objetos** (Fowler, 2004) quanto pelo padrão arquitetural de **Sistemas de Entidades e Componentes** (C/ES) (Leonard, 1999). Como visto na seção anterior, a **representação** do componente está incluída como um dos campos da **estrutura** de documentação. Contudo, ela pode ser utilizada fora desse contexto, como um instrumento de documentação e comunicação rápida. De acordo com o escopo apresentado na seção 3.1, esta seção corresponde à segunda e mais importante parte do instrumental de design definido por esta tese, uma vez que ela permite representar as composições dos componentes, suas relações e aplicações.

A representação dos componentes pode ser feita de três formas distintas, cada qual com suas particularidades. Elas estão enumeradas abaixo e são discutidas no texto que segue.

- Linguagem de Componentes;
- Diagrama de Componentes;
- Mapas de Design.

Os mecanismos de representação que serão discutidos nesta seção foram idealizados como instrumentos de modelagem de design de jogos. Eles definem o núcleo fundamental da abordagem de componentes e estão inseridos no mesmo contexto da "engenharia de design" dos trabalhos relacionados a esta tese (discutidos no Capítulo 2). Cabe reiterar que no contexto do design de jogos, o termo "formal" não se refere à adoção de modelos matemáticos que impedem a ambiguidade. Nesse sentido, as situações discutidas nas próximas seções podem apresentar diferentes soluções. A atividade de modelagem em si admite interpretações abstratas, o que implica que designers podem criar versões ligeiramente diferentes de representações do mesmo jogo. A modelagem é, portanto, uma atividade de criação e não está

[93] http://protagonist.sics.chalmers.se:1337/mediawiki-1.22.0/index.php/Category:Patterns (visitado em 03/02/2016)
[94] http://www.gameontology.com/index.php/Main_Page (visitado em 03/02/2016)

restringida a uma interpretação absoluta: é o resultado direto da visão do modelador. Não existe, nesta tese, a pretensão de se conceber uma linguagem formal que permita a geração imediata de código fonte de jogos. Assim como nos trabalhos relacionados, esta abordagem intenta prover instrumentos estruturados que forneçam suporte ao processo de design de jogos e a estruturação do conhecimento de design como um todo, em contraste à prática de fundamentá-lo exclusivamente no documento de design.

3.6.1 Linguagem de Componentes

A linguagem textual é o principal meio utilizado para representação dos componentes. Ela emprega uma sintaxe bem definida para descrição de componentes, seus tipos e atributos. Também permite representar composições e associações entre componentes. Esta forma de representação é mais adequada para descrever a constituição e a estrutura de um componente.

A sintaxe de descrição de um componente segue a estrutura apresentada na Figura 28. Os componentes são delimitados por "[ ]" e as palavras que o compõem iniciam com letras maiúsculas. À direita dos componentes são listados os tipos que o classificam. Cada tipo é apresentado entre "< >". Um componente pode ser classificado por um ou mais tipos. Um **tipo** também é um componente e, dessa forma, qualquer componente pode tipificar outros. Esse mecanismo chamado de **classificação** é usado para organizar os componentes de forma hierárquica. Mais detalhes são apresentados na seção 3.7.3.

```
[Definição do Componente] <tipos>
{
      descrição

      atributo: valores possíveis

      [Componente Interno] <tipos>
      {
            atributo: valores possíveis
      }
}
```

Figura 30 - Sintaxe da representação segundo a Linguagem de Componentes.

O conteúdo um componente é compreendido por atributos e outros componentes, que definem seus aspectos e comportamentos. A presença dos componentes internos corresponde ao mecanismo de **composição**, que também é discutido na seção 3.7. A composição estimula o designer a investigar a constituição do componente em um nível de granularidade mais baixo. Em outras palavras, o designer é levado a pensar nas possíveis partes que possam formar o conceito que ele deseja tratar como um componente. Componentes internos tipicamente são previamente definidos e apenas usados na composição de outros.

Os **atributos** dos componentes são escritos em letras minúsculas. Eles servem para explicitar detalhes que caracterizam e ajudam a identificar o emprego do componente em jogos distintos. Atributos podem apresentar um valor fixo ou listar opções de valores possíveis. A definição dos atributos estimula o designer a pensar nas características do comportamento ou funcionamento de cada componente e do aspecto que este representa.

A Figura 31 apresenta um exemplo de componente que define um aspecto comumente encontrado em jogos 2D: a visão lateral com rolagem ([Side Scroller]). Esse componente corresponde aos conceitos Side View[95] (TV Tropes), Side-Scrolling[96] (Giant Bomb) e Side-Scrolling videogame[97] (Wikipedia). A perspectiva lateral com rolagem pode ser descrita como a soma dos aspectos visão lateral e rolagem e, portanto, está representada como uma composição de [Side View] e [Scroll]. Dois atributos foram identificados. O atributo "look ahead" indica se a visão lateral de avançar à frente do jogador para aumentar seu campo de visão. O atributo "scroll type" refere-se ao tipo de rolagem deve ser empregada:

- "forwards" indica que a visão somente move-se para frente, não podendo retroceder;
- "free" indica que a visão pode mover-se para qualquer direção, podendo retroceder;
- "auto-scroll" indica que a visão move-se automaticamente para frente, forçando o jogador a progredir ou perder a partida.

```
[Side Scroller] <pov>
{
      look ahead: yes/no
      scroll type: forwards/free/auto-scroll

      [Side View] <pov>

      [Scroll] <pov>
      {
            scroll axis: vertical/horizontal/both
      }
}
```

Figura 31 - Exemplo de componente que descreve o aspecto de visão lateral com rolagem.

O componente [Side Scroller] representa uma versão mais genérica da visão lateral empregada em jogos 2D, especialmente aqueles pertences ao gênero Platformer. No entanto, vários jogos possuem um ou mais estágios em que o personagem é perseguido continuamente pela tela, devendo avançar sobre os obstáculos presentes no level para alcançar o final, sob o risco de ser esmagado pela mesma. O nome popularmente empregado para este tipo de aspecto é "auto-scrolling level". Dessa forma, dentro do contexto dos jogos que o empregam, o componente [Side Scroller] pode ser convenientemente documentado de forma especializada, gerando o subtipo [Auto-Scrolling Level]. Sua estrutura está representada na Figura 32.

```
[Auto-Scrolling Level] <side scroller>
{
      [Side Scroller] <pov> {
            moves ahead: yes
            scroll type: auto forward
      }
}
```

Figura 32 – Exemplo de especialização do [Side Scroller].

[95] Side View - http://tvtropes.org/pmwiki/pmwiki.php/Main/SideView
[96] Side-Scrolling - http://www.giantbomb.com/side-scrolling/3015-299/
[97] Side-Scrolling Videogame - https://en.wikipedia.org/wiki/Side-scrolling_video_game

O "level de rolagem automática" documentado no componente da Figura 32 é um tipo especializado de [Side Scroller] sendo, portanto, classificado como <side scroller>. Embora seja aparentemente uma redundância, a definição do componente [Auto-Scrolling Level] torna-se bastante conveniente, uma vez que emprega um termo popular na indústria e na comunidade de usuários. Dessa forma, contribui para a definição de um vocabulário compartilhado de design. Curiosamente, o componente também está documentado como conceitos nas bases Giant Bomb[98] e TV Tropes[99], assim como vários outros apresentados neste trabalho, o que ajuda a atestar o reconhecimento como aspectos identificáveis de jogos. Cabe ressaltar que uma vez que o componente [Side Scroller] foi previamente definido, não é necessário repetir sua estrutura por completo. É interessante notar que [Auto-Scrolling Level] é um componente-chave em jogos do subgênero Endless Runner (Jogo de Corrida Infinita), extremamente populares em dispositivos móveis. Exemplos são os títulos Temple Run[100], Subway Surfers[101] e Canabalt[102].

Em continuidade ao exemplo anterior, a composição e as relações hierárquica de um Endless Runner é apresentada abaixo. Nele, é possível notar que a classificação permite representação de gêneros e subgêneros de forma hierárquica. Dessa forma, o componente [Endless Runner][103] exposto na Figura 33 está classificação como um jogo "infinito" de "plataformas".

```
[Endless Runner] <platformer> <endless game>
{
      [Auto-Scrolling Level] <side scroller>

      [Distance meter] <hud>

      [Obstacle] <level object>
}
```

Figura 33 - Modelagem do subgênero [Endless Runner].

Os jogos do tipo [Endless Runner] correspondem a um subgênero dos jogos de plataforma (Platformers[104] [105]). Em termos de componentes, isto significa que o componente [Endless Runner] (Figura 33) é uma especialização de [Platfomer], que está definido na Figura 34. Em linhas gerais, os jogos do gênero se caracterizam pelo jogador controlar um personagem para realizar a ação de **pular** sobre **plataformas** e evitar **obstáculos**, a fim de atingir o objetivo definido no jogo. Esta modelagem difere levemente da apresentada na seção 2.3, na medida que define o componente [Jump] como uma ação constituinte do personagem controlado pelo jogador. Cabe ressaltar que os componentes empregados na modelagem do gênero [Platformer] são genéricos e usualmente serão substituídos por outros mais especializados, quando ocorrer

[98] http://www.giantbomb.com/auto-scrolling-levels/3015-2299/
[99] http://tvtropes.org/pmwiki/pmwiki.php/Main/AutoScrollingLevel
[100] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/TempleRun
[101] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/SubwaySurfers
[102] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/Canabalt
[103] http://tvtropes.org/pmwiki/pmwiki.php/Main/EndlessRunningGame
[104] http://tvtropes.org/pmwiki/pmwiki.php/Main/PlatformGame
[105] https://en.wikipedia.org/wiki/Platform_game

o emprego do componente.

```
[Platformer] <genre>
{
      [Jumper Character] <player> {
            [Jump] <action>
      }

      [Platform] <level construction>

      [Obstacle] <level construction>
}
```

Figura 34 - Modelagem do gênero plataforma.

Segundo a definição apresentada na Figura 33, é possível observar que, além ser um subtipo de [Platformer], o jogo do tipo [Endless Runner] também é uma especialização de [Endless Game][106] (Figura 35). Nessa modalidade de jogos, existe uma estrutura de level infinita [Endless Structure], sob a qual pode-se usar tipos especializados, como um level interminável [Endless Level], que repete com aumento gradual de dificuldade [Level Loop], ou mesmo, um conjunto de levels aleatoriamente repetidos [Random Levels Loop]. O objetivo é alcançar a maior distância possível e, nesse sentido, a medida de progressão usualmente empregada é o [Score], que é registrado em um ranking [High Score]. Usualmente, o jogo infinito tem dificuldade crescente, aumentando a velocidade do *gameplay* com o progresso do jogador [Difficulty by Acceleration][107]. Caso o jogador perca a partida, a sessão de jogo é reiniciada [Start Over]. Esses aspectos estão representados como componentes internos à [Endless Game]. Cabe notar que um [Endless Game] é comumente oposto a um [Completable Game].

```
[Endless Game] <game>
{
      >> Emprega estrutura de level sem fim: seja por
      >> repetição do mesmo level ou por estendê-lo interminavelmente
      [Endless Structure] <level structure>
      [Difficulty by Acceleration] <difficulty modifier>
      [Start Over] <game session>

      [Reach Further] <primary objective> {
            achieve: [Score]
      }

      [Score] <progress> {
            register: [High Score]
      }
      [High Score] <counter>
}
```

Figura 35 - Modelagem para o tipo de jogo [Endless Game].

O mecanismo de classificação implica em uma composição implícita. Nesse sentido, ao classificar [Endless Runner] como <platformer> e <endless game>, estes dois componentes, bem como seus componentes internos, passam a fazer parte da composição do primeiro de forma implícita. Cabe ressaltar que os componentes implícitos podem ser omitidos e, por essa razão,

[106] http://tvtropes.org/pmwiki/pmwiki.php/Main/EndlessGame
[107] http://tvtropes.org/pmwiki/pmwiki.php/Main/DifficultyByAcceleration

não estão explicitados em todos os exemplos presentes neste trabalho. Para fins demonstrativos, o componente [Endless Runner] está reapresentado abaixo (Figura 36) em sua forma mais detalhada, expondo os componentes implícitos [Plataformer] e [Endless Game]. A coloração foi empregada para facilitar a visualização dos componentes e atributos referentes a cada classificador.

```
>> Define os aspectos fundamentais aos jogos de corrida infinita,
>> subgênero dos jogos Platformers
[Endless Runner] <platformer> <endless game>
{
      [Auto-Scrolling Level] <side scroller>
      [Distance meter] <hud>
      [Obstacle] <level object>

      >> Define os aspectos fundamentais ao gênero plataforma
      [Platformer] <genre>
      {
            [Jumper Character] <player> {
                  [Jump] <action>
                  [Auto-move forward] <movement>
            }
            [Platform] <level construction>
            [Obstacle] <level construction>
      }

      >> Define os aspectos fundamentais ao "jogo infinito"
      [Endless Game] <game> {
            [Endless Structure] <level structure>
            [Start Over] <game session>
            [Difficulty by Acceleration] <difficulty modifier>
            [Reach Further] <primary objective> {
                  achieve: [Score]
            }
            [Score] <progress> {
                  register: [High Score]
            }
            [High Score] <counter>
      }
}
```

Figura 36 - Modelagem do subgênero [Endless Runner].

A Linguagem de Componentes permite que o designer utilize uma descrição sintética para enfatizar as principais características dos aspectos de jogos. Ela atua como um complemento conveniente à estrutura de descrição de componentes exposta na seção 3.5, que adota um formato inspirado nos Padrões de Projeto e permite uma descrição mais detalhada de componentes. Essa estrutura é mais apropriada à implementação de uma base de conhecimentos de componentes de design, de forma similar aos sites GiantBomb e TvTropes. Por fim, cabe ressaltar que as outras abordagens de "engenharia de design" discutidas no Capítulo 2 não provêem uma linguagem específica para a descrição dos conceitos identificados nos jogos. Em geral, elas são organizadas segundo o formato de Padrões de Projetos.

3.6.2 Diagrama de Componentes

A modelagem de componentes apresentados na seção anterior pode ser expressa de

uma maneira alternativa. Caso seja mais conveniente ao designer, os componentes podem ser representados por meio de diagramas. Considerando a similaridade existente entre a abordagem de componentes de design e o paradigma da OO, os diagramas de classe mostram-se bastante apropriados para a representação de componentes. No entanto, para acomodar as particularidades dos componentes de jogos, foram introduzidas várias modificações sobre a sintaxe original da UML. Em comparação à representação textual, os diagramas permitem visualizar mais facilmente as relações entre componentes. Em contrapartida, a descrição textual é mais ágil de ser produzida e enfatiza a organização interna aos componentes. Para fins de demonstração, o componente [Jump], previamente exibido na Figura 20, está representado na Figura 37, acompanhado pelo diagrama de componentes correspondente (Figura 38). Pelo exemplo, é possível notar que o conteúdo apresentado em ambos tipos de representações é o mesmo.

```
>> Movimento que trouxe a verticalidade aos jogos. Funciona como um
pulo real: faz o objeto subir e descer.
[Jump] <action>
{
      controlable: yes/no/partial
      height: fixed/variable
      affected by motion: ground/air/both/no
      auto assisted: yes/no/prompted
}
```

Figura 37 - Representação de [Jump] pela Linguagem de Componentes.

Os diagramas de componentes mostram-se bastante convenientes para ilustrar as relações hierárquicas do mecanismo de classificação, que permite criar uma organização de subtipos especializados. Como exemplo, a Figura 40 apresenta um diagrama das especializações do componente [Jump] para o contexto de jogos conhecidos, conforme previamente discutido na seção 3.2.

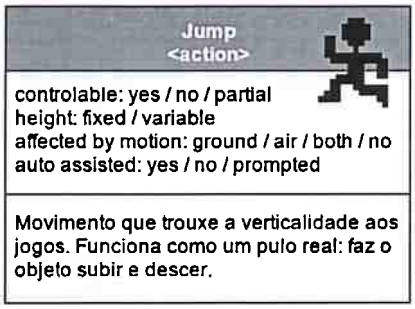

Figura 38 - Representação de [Jump] por um Diagrama de Componentes.

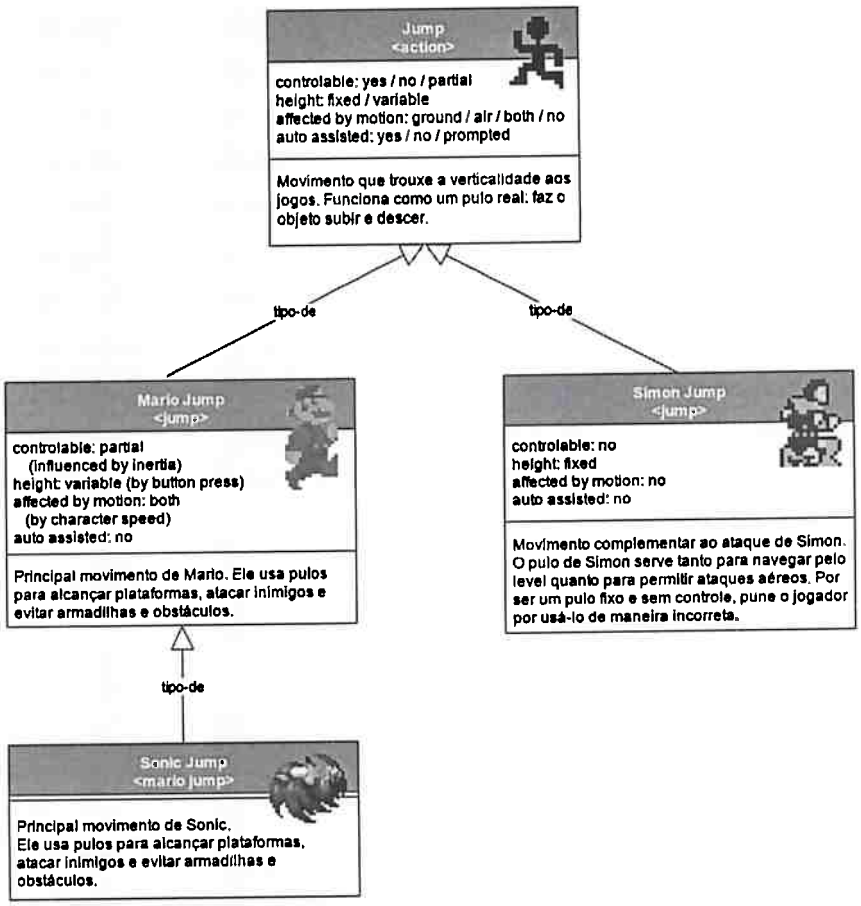

Figura 40 - Diagrama de Componentes representando especializações de [Jump].

Por fim, a Figura 39 mostra um exemplo de representação de composição para a modelagem do [Attack Jump], exposto anteriormente pela linguagem de componentes na Figura 25. Além desse, a modelagem do personagem [Piranha Plant], previamente apresentada na Figura 2, é um outro exemplo da aplicação da composição em digramas de componentes. É interessante notar que, visualmente, os diagramas de componentes que empregam a composição assemelham-se a cartões de design. Essa forma de apresentação pode ser empregada de forma física, tal qual um protótipo analógico. Assim, os componentes podem ser

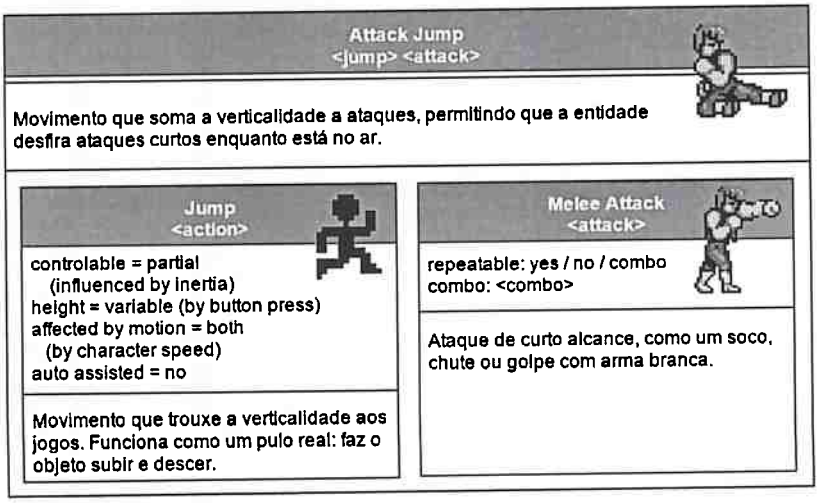

Figura 39 - Diagrama de Componentes representando uma composição em [Attack Jump].

impressos na forma de cartões para que o designer possa, convenientemente, montar e visualizar os principais seus aspectos do conceito de um jogo.

3.6.3 Mapas de Design

As duas formas de representações mostradas nesta seção, a Linguagem de Componentes e o Diagrama de Componentes, são mais apropriadas para descrever a estrutura interna dos componentes e as relações que podem existir entre eles. Para identificar os componentes em jogos existentes e, principalmente, projetar novos jogos, pode-se empregar uma forma de documentação mais livre. Inspirando nos designs de uma página de Librande (2010), também discutidos por Rogers (2010), o Mapa de Design consiste de instrumento de associação de componentes ao seu emprego em jogos e gêneros de um modo bastante flexível. Nesse sentido, o leiaute e o conteúdo do mapa podem ser modificados, de forma a facilitar a comunicação dos aspectos determinantes de cada tipo de jogo. Como exemplo, pode-se adotar uma documentação orientada a telas ou imagens conceituais do jogo, como em um protótipo de interface de software, nas quais indica-se a ocorrência dos componentes. Também é possível utilizar um mapa de camadas de componentes, dirigido a aspectos do jogo. Dessa forma, é possível destacar visualmente (por cores ou separações em quadros) os componentes fundamentais, que definem os principais aspectos do gameplay, e aqueles que agem de forma secundária. Além desses, o mapa pode ser montado de forma dirigida ao mundo do jogo, abrangendo detalhes de locais, objetivos, quests e situações específicas de cada localidade. Por fim, também é possível organizar o mapa de design como um storyboard, explicitando a ordem de eventos do jogo. Esses e outros modelos de mapas de design são mostrados na apresentação de Librande (2010) e no livro de Rogers (2010). O objetivo dessa abordagem é adotar uma forma de documentação sintética e livre, porém abrangente, e que seja mais conveniente para expressar as particularidades do jogo em desenvolvimento.

Os designs de uma página de Librande (2010) trouxeram uma forte influência para a definição de um modelo de documentação flexível e intuitiva para a aplicação da abordagem de componentes. É interessante destacar que, desde o princípio deste trabalho, notou-se uma similaridade entre a abordagem de Librande e a forma de documentação que era empregada pelo autor durante as seções de brainstorming (Rogers, 2010) nos projetos em que participou. Nesse sentido, os primeiros protótipos de demonstração da abordagem de componentes utilizaram como documentação os mapas de design orientados a imagens do jogo. Eles foram desenvolvidos pelo autor como instrumentos didáticos, voltados as disciplinas de desenvolvimento de jogos que ministrou durante a realização deste trabalho.

No primeiro protótipo desenvolvido, chamado de Bouncing Balls, o objetivo era apresentar o comportamento de um jogo pelo uso de componentes. Por essa razão, optou-se por produzir um aplicativo simples, que continha características de tempo real de jogos, tais como entidades, movimentos e colisões, mas não permitia a interação de um usuário. Ele foi concebido e desenvolvido como um primeiro exemplo didático. No aplicativo Bouncing Balls, um grupo de entidades em forma círculos movimentam-se diagonalmente pelo mundo do jogo,

que é restrito à tela, e rebatem em qualquer um dos quatro lados desta. Novas entidades são adicionadas à tela continuamente, em posições aleatórias.

A Figura 41 mostra um exemplo do Mapa de Design empregado para decrever o aplicativo Bouncing Balls. Nele, os componentes são apresentados de forma resumida e apontados diretamente sobre a imagem conceitual do aplicativo. Nesse contexto, os mapas de design têm por objetivo prover uma forma de documentação mais sintética, que destaca os principais componentes empregados no jogo. Nesse sentido, é possível notar que não há uma preocupação em organizar e relacionar os componentes no exemplo apresentado.

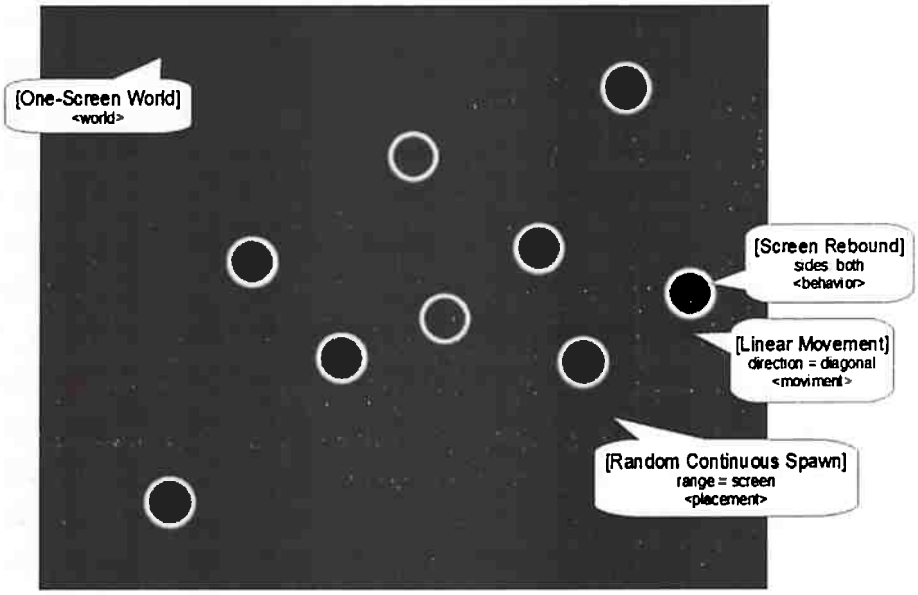

Figura 41 - Mapa de design do protótipo Bouncing Balls.

Usualmente a documentação dos protótipos produzidos nas disciplinas era elaborada de forma rústica, manuscrita diretamente em um quadro branco durante a seção de brainstorming, na qual discutia-se o significado e o impacto de cada componente empregado. Esta seção era realizada em uma aula da disciplina, seguida por aulas que tratavam da implementação do protótipo.

O segundo exemplo de protótipo trabalhado em uma das disciplinas de jogos ministradas, foi concebido a partir de alguns componentes extraídos do jogo Sonic. Na Figura 42 há uma cena do jogo durante o level Scrap Brain Zone 2, no qual há plataformas identificadas como o componente [Orbital Platform] <rotate platform>. Esse tipo especializado de plataforma rotatória possui um comportamento bastante peculiar: ele gira constantemente, prendendo à sua superfície circular toda entidade que entrar em contato consigo. Dessa forma, o personagem do jogador, ao pular e colidir com a plataforma, passa a rotacionar em torno da mesma, sendo impedido de mover-se enquanto estiver neste estado. Para desvincular-se da plataforma, o jogador deve realizar uma ação de [Jump], outro componente extraído do jogo original. O comportamento da [Orbital Platform] permite que o personagem seja arremessado para qualquer direção, em uma extensão de 360º. O objetivo deste exemplo era mostrar aos alunos que é possível utilizar uma abordagem de design "bottom-up" com os componentes de jogos.

Nela, o designer identifica e extrai componentes peculiares de jogos existentes e então inicia o processo de concepção de um novo conceito de jogo, centrando-se nos componentes identificados. A partir disto, inicia-se um ciclo iterativo de design gradual adaptativo, no qual trabalha-se incrementalmente sobre um protótipo de experimentação e na estética experimentada, de modo que a composição do jogo evolua gradativamente.

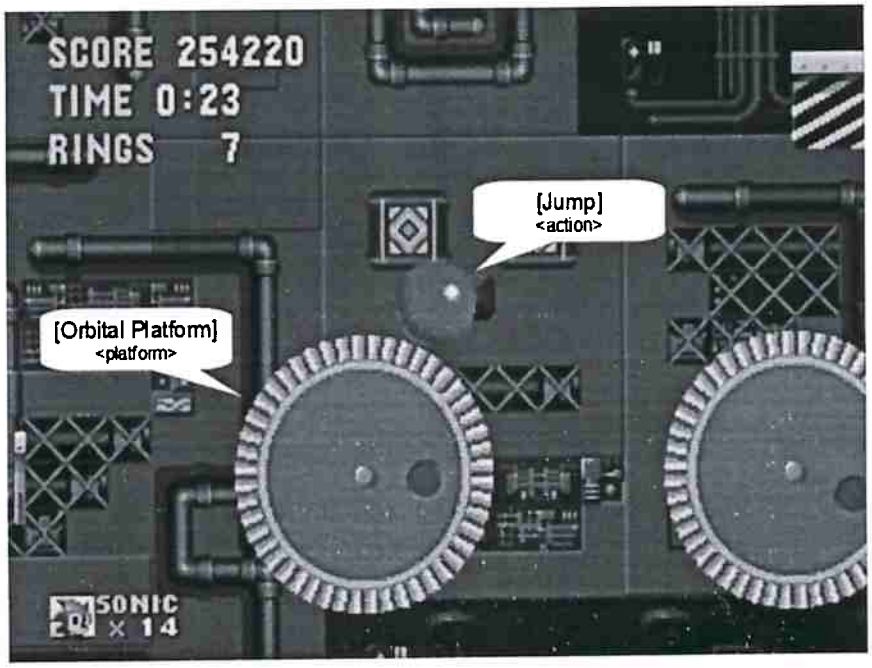

Figura 42 - Foto do level Scrap Brain Zone 2 do jogo Sonic com os componentes identificados.

Os dois componentes extraídos do jogo Sonic, [Orbital Platform] e [Jump] (Figura 22), serviram de base para a criação de um protótipo simples, chamado de Orbit Jumper. A definição do componente [Orbital Platform], bem como, uma versão reduzia da hierarquia de especializações de plataformas está representada na Figura 43. Uma modelagem mais abrangente de tipos de plataformas comumente encontrados em jogos está exposta na Figura 61.

```
>> Componente generico de plataforma. Uma plataforma e tipicamente uma
superfície na qual o jogador pode pisar. Em jogos clássicos de
plataformas, elas se encontram suspensas no ar. Plataformas geralmente
carregam os objetos sobre elas (carry objects), mas em alguns jogos, é
preciso acompanhá-las manualmente.
[Platform] <level construction>
{
      one way: yes / no
      carry objects: yes / no
}
>> Plataforma que se move, sem influência do jogador. Plataformas
podem fazer um intervalo em seu percurso (stop time) e spresentar
suavidade ao parar e reiniciar (ease type).
[Moving Platform] <platform>
{
      stop time: yes / no (continuous)
      ease type: in-out / in / out / linear
      axis: x / y / z
}
```

```
>> Plataforma que rotaciona em torno de seu eixo, dentro de uma
abrangência de ângulos (rotation angle). Pode rotacionar sempre na
mesma direção ou retroceder após parada (rewind). A parada ocorre em
um determinado ângulo (stop angle).
[Rotating Platform] <platform>
{
      rotation range: angle value
      stop angle: angle value
      rewind: yes / no
}

>> Tipo especializado de plataforma rotatoria. Usualmente em forma de
círculo (2D) ou esfera (3D), prende à sua superficie a entidade que a
tocar. O jogador tipicamente deve realizar uma ação para soltar-se. A
velocidade de rotação pode influenciar na força do lançamento aplicado
a entidade (centrifugal force). O jogador pode acelerar o movimento
centrífugo da plataforma (centriugal acceleration).
[Orbital Platform] <platform>
{
      centrifugal force: yes /no
      centrifugal acceleration: yes / no
}
```

Figura 43 - Hierarquia de especializações de plataformas para [Orbital Platform].

A Figura 44 expõe o mesmo conteúdo do exemplo anterior por meio de um diagrama de componentes. Esta forma de representação facilita a visualização das relações hierárquicas gerada pelas classificações entre os componentes de plataformas. Uma versão mais detalhada da hierarquia, contemplando diversos subtipos de plataformas está contida na Figura 61.

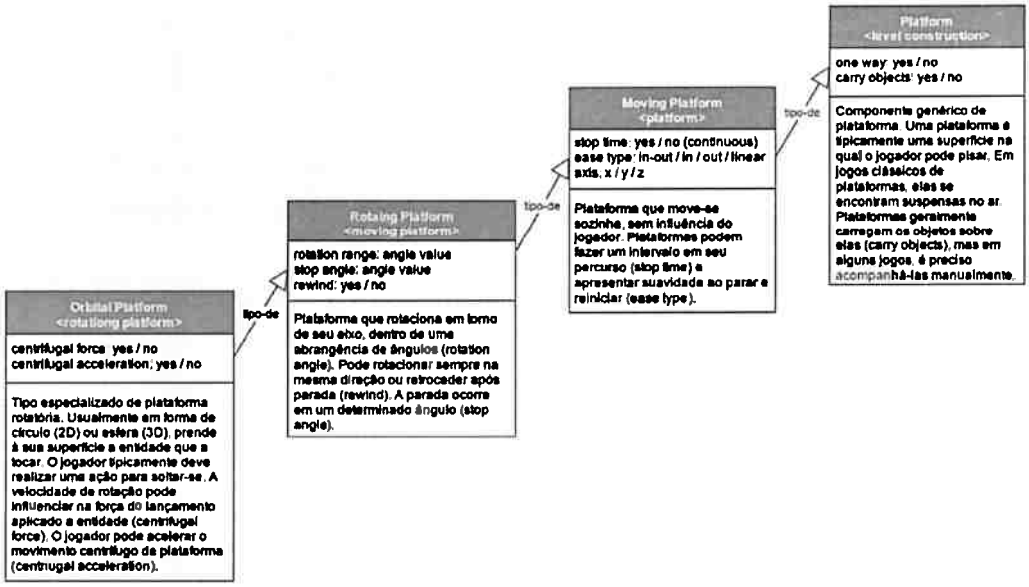

Figura 44 - Diagrama da hierarquia de especializações de plataformas para [Orbit Platform].

A primeira versão do protótipo Orbit Jumper foi construída fundamentando-se sobre os dois componentes selecionados, [Jump] e [Orbital Platform]. A partir da estética percebida pelo uso do protótipo, novos componentes foram gradativamente adicionados ao conceito do jogo, dentro de um ciclo de implementações e experimentações. Em vias de simplificar o design do jogo, a versão final tornou-se um "jogo infinito" (componente [Endless Game], Figura 35) de

plataformas (componente [Platformer], Figura 34). Nele, o jogador usa a ação de pular ([Jump]) entre as plataformas orbitais ([Orbital Platform]) para subir, com o objetivo de atingir a maior altura possível, contabilizada em pontos ([Score]). Ele deve evitar o "buraco" na base da tela, único obstáculo do jogo ([Hole]). A rolagem da tela ocorre apenas no eixo vertical ([Side Scroller] type = vertical) e suas laterais rebatem o jogador caso as toque ([Screen Rebounds]). Este último aspecto é característico de jogos de rebotes de bolas, como Pong[108], Breakout[109] e Arkanoid[110]. O mapa de design montado para o protótipo é mostrado na Figura 45.

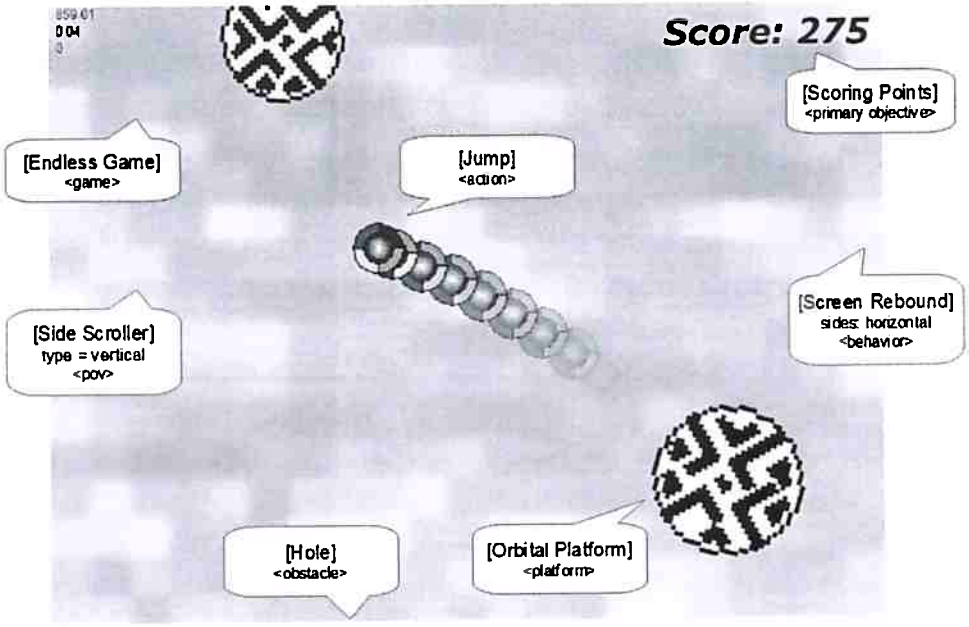

Figura 45 - Mapa de design do protótipo Orbit Jumper.

A modelagem de componentes para o jogo Orbit Jumper é mostrada na Figura 47 por meio da Linguagem de Componentes. Este exemplo demonstra um recurso interessante do mecanismo de classificação. Assim como as classes abstratas puras da OO, também conhecidas pelo nome de interfaces, a abordagem de componentes prevê a criação de *templates* (modelos). Os *templates* definem um esqueleto de estrutura a ser definida pelo componente que vierem a classificar. O esqueleto é descrito por meio de classificadores, que definem os tipos para os componentes que deverão ser preenchidos em uma futura especialização do *template*. Como exemplo, a Figura 46 mostra um *template* para um "jogo" ([Game]). Segundo o esqueleto, todo jogo deve contém uma definição de jogador, objetivo e sistema de progressão. Seu uso ocorre por meio da classificação, especializando-o para um componente específico. É importante destacar que os componentes usados na constituição de [Game] são mais genéricos e serão substituídos por outros, mais especializados, quando o *template* for empregado. Como exemplo, o componente [Orbital Jumper] (Figura 47) apresenta uma especialização do *template* [Game] para modelar o jogo em termos de seus componentes.

---

[108] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/Pong
[109] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/Breakout
[110] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/Arkanoid

```
>> Template para [Game]
[Game] <template>
{
      >> O jogo deve ter um jogador
      <player>
      >> O jogo deve ter um objetivo
      <primary objective>
      >> O jogo deve ter um sistema de progressão
      <game progression>
}
```

Figura 46 - Template para definição de um jogo.

O componente que modela o protótipo do jogo Orbital Jumper foi classificado como subtipo (ou especialização) de um jogo infinito do gênero plataforma (<game> <platformer> <endless game>). É possível notar que os componentes genéricos definidos para o gênero "platformer" foram substituídos ou modificados para se adequar ao contexto do jogo (Figura 47). A definição para os componentes [Game], [Platformer] e [Endless Game] já foi apresentada previamente. A coloração do texto é empregada para facilitar a identificação dos componentes e atributos oriundos de cada classificador.

```
>> Definição do componente que representa o jogo Orbit Jumper
[Orbit Jumper] <game> <platformer> <endless game> {

      >> Componente implícito da classificação e especializado para
      >> este jogo
      [Platformer] <genre> {

            >> O jogo deve ter um jogador
            <player>:
            [Jumper Character] <player> {
                  [Jump] <action> {
                        controlable = yes (tight)
                        height = variable by [Orbit Platform] speed
                  }
                  [Screen Rebounds] <behavior> {
                        side: horizontally
                  }
            }

            >> Usa um tipo mais especializado de plataforma
            ~~[Platform] <level construction>~~
            [Orbital Platform] <platform> {
                  centrifugal force = yes
                  centrifugal acceleration = no
            }

            >> Usa um tipo mais especializado de obstáculo
            ~~[Obstacle] <level construction>~~
            [Hole] <obstacle>
      }

      >> Componente implícito da classificação
      >> Define os aspectos fundamentais ao "jogo infinito"
      [Endless Game] <game> {
            [Endless Structure] <level structure>
            [Difficulty by Acceleration] <difficulty modifier>
            [Start Over] <game session>
```

```
            >> O jogo deve ter um objetivo
            <primary objective>:
            [Reach Further] <primary objective> {
                  achieve: [Score]
            }

            >> O jogo deve ter um sistema de progressão
            <game progression>:
            [Score] <progress> {
                  register: [High Score]
            }
            [High Score] <counter>
      }
}
```

Figura 47 - Modelagem de componentes para o design do jogo [Orbit Jumper].

O exemplo do Orbit Jumper traz alguns pontos interessantes referentes à modelagem e ao emprego de alguns componentes. Nesse sentido, ele é o primeiro exemplo a modelar um jogo por completo. Certamente, por tratar-se de um jogo mais simples, sua descrição por meio de componentes é mais sintética. Em jogos de design mais complexo, a compreensão da modelagem pode ser auxiliada por diagramas de componentes. Tratando especificamente dos classificadores empregados no exemplo, nota-se que seus conteúdos foram mesclados no corpo do componente [Orbit Jumper]. Entre eles, o componente [Platformer], que define as características fundamentais ao gênero "plataforma", teve dois componentes genéricos substituídos por outros mais especializados. Nesse sentido, [Platform] e [Obstacle], ambos representando elementos usados na construção de levels de jogos (<level construction>), foram substituídos respectivamente por [Orbital Platform], tipo especializado de plataforma (<platform>), e [Hole], tipo especializado de obstáculo (<obstacle>). Por fim, a respeito do emprego do template [Game], os requisitos estruturais feitos pelo mesmo são satisfeitos pelos componentes [Jumper Character] (<player>), [Reach Further] (<primary objective>) e [Score] (<game progression>). Este último representa a forma mais elementar e antiga de sistema de progressão empregada em um jogo.

Após a finalização do protótipo Orbit Jumper, curiosamente foram encontrados vários jogos que apresentam os mesmos aspectos e, dessa forma, empregam os mesmos componentes. Esse fato leva a duas observações importantes. Primeiramente, corrobora a prática da identificação de aspectos interessantes em jogos existentes para serem aplicados a um novo projeto, ou mesmo, o inspirarem. Em segundo lugar, ajuda a justificar a própria abordagem de componentes desta tese, uma vez que os jogos encontrados se assemelham ao Orbit Jumper. Exemplos de jogos são On the Hop[111], Planet Jumpers[112] e Little Galaxy[113] (Figura 48).

---

[111] On the Hop: https://itunes.apple.com/us/app/on-the-hop/id498238213?mt=8

[112] Planet Jumpers: https://itunes.apple.com/br/app/planet-jumpers-have-walk-among/id875758363?mt=8

[113] Little Galaxy, https://itunes.apple.com/br/app/little-galaxy-family/id576764076?mt=8

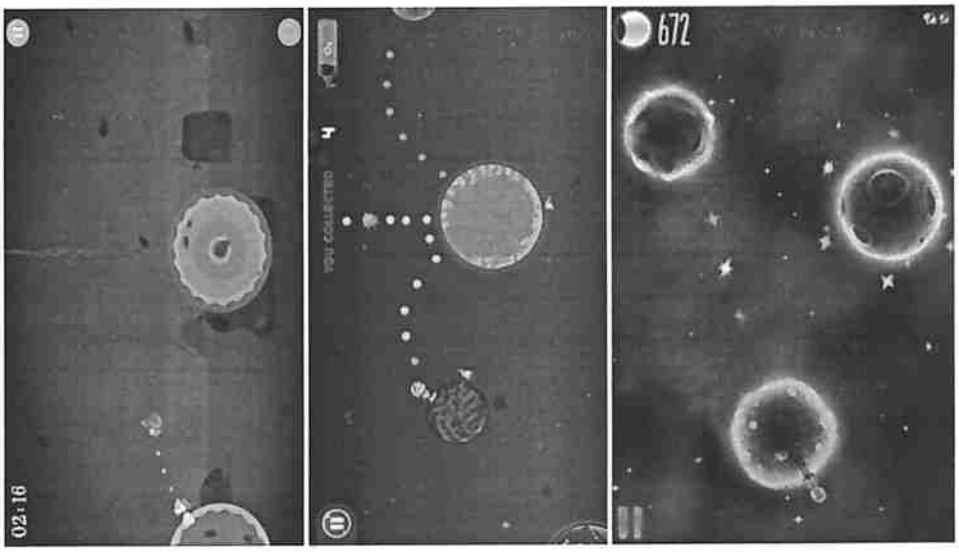

Figura 48 - Jogos para iOS: On the Hop, Planet Jumpers e Little Galaxy.

Esta seção apresentou e discutiu diversos exemplos que ilustraram a aplicação das três forma de representação da abordagem de design deste trabalho: a Linguagem de Componentes, o Diagrama de Componentes e o Mapa de Design. Elas foram concebidas para representar os componentes e suas especializações em jogos, agindo de forma sinérgica. Nesse sentido, a Linguagem de Componentes mostra-se conveniente para descrever e visualizar a estrutura interna de cada componente. Os Diagramas, por outro lado, são mais adequados para identificar as relações entre componentes. Por fim, os Mapas de Design são apropriados para comunicar os componentes dentro do contexto de jogos e gêneros, permitindo um mapeamento visual bastante prático entre componentes e suas aplicações. Essas três formas de representação são complementares e idealmente estariam implementadas na base de conhecimentos (seção 3.1). Nesse sentido, as representações textuais e os diagramas documentariam os componentes, ligados à estrutura de descrição enumerada na seção 3.5. Já os mapas de design permitiriam explicitariam as aplicações destes em jogos e gêneros.

Os mecanismos de relacionamentos de componentes – associação, composição e classificação – foram empregados em todos exemplos previamente discutidos neste trabalho. Eles constituem os elementos-chave para a modelagem de componentes e sua representação, quer seja pela linguagem, diagramas ou mapas de design. A próxima seção descreve e exemplifica as três formas de relações possíveis entre componentes.

3.7 Relações entre componentes

Como uma derivação dos princípios da Orientação de Objetos, os componentes podem apresentar três tipos de relacionamentos: **associação**, **classificação** e **composição**. A relação de associação permite os componentes tenham ligações, que podem ser determinantes para seu comportamento. A composição é a característica fundamental à abordagem, definindo a constituição dos aspectos formadores de jogos e gêneros. Por fim, a classificação age como um rotulador hierárquico de componentes. Ela apresenta similaridades à herança de classes, mas o conteúdo herdado não se refere a atributos e métodos, mas sim, à inclusão implícita do próprio componente classificador.

3.7.1 Associação

O relacionamento de associação define uma interação entre dois ou mais componentes. Ele ocorre por meio dos atributos de relacionamento nos componentes. Os atributos de relacionamento são aqueles cujos valores "apontam" para outros componentes, tal como ocorre em uma referência de objetos na OO. Como exemplo, na Figura 49 está modelada a ação de "chutar um casco de tartaruga", a qual o personagem do jogador executa no jogo Super Mario Bros. O componente [Kick Koopa Troopa Shell], uma especialização de [Throw Object], define que objeto é lançado por meio do atributo "object". De forma similar, a ocorrência da ação está condicionada à colisão entre o personagem do jogador e o [Koopa Troopa Shell], uma especialização de [Boucing Object]. Outras associações presentes no componente [Koopa Troopa Shell], mas não expressas no diagrama, estão enumeradas nos atributos "damages" e "bounce off". O primeiro indica que o objeto lançado pode danificar três tipos de componentes: jogador ([Player]), inimigos ([Enemy]) e blocos quebráveis do cenário ([Breakables]). O segundo indica que o objeto lançado rebate em paredes do cenários ([Wall]).

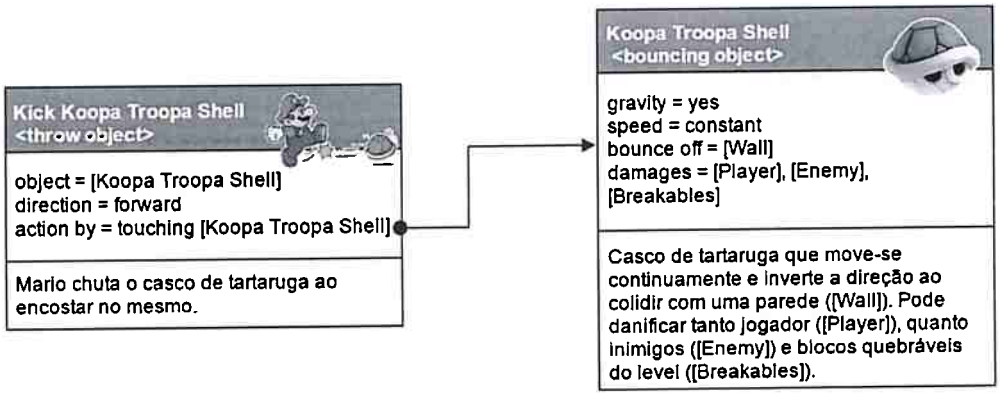

Figura 49 - A associação entre componentes ocorre pelos atributos.

Componentes genéricos (não especializados) contém a definição dos atributos e uma relação dos possíveis valores que podem assumir. Para os atributos de relacionamento, indica-se o tipo do componente que ele pode referenciar. Dessa forma, em uma especialização do mesmo componente, o atributo pode referenciar qualquer subtipo daquele que está especificado. Como exemplo, a Figura 50 expõe associações em componentes genéricos. O componente [Look at Target], muito comum em jogos de tiro (ex: canhões), é associado ao alvo por meio do atributo "target", que espera uma referência a um componente do tipo "entidade" (<entity>). Dessa forma, esse atributo pode receber qualquer componente que seja hierarquicamente classificado como <entity>, tal como os personagens dos jogos.

```
>> Comportamento que direciona uma entidade a um alvo (target). Pode
rotacionar de forma instantânea ou suavizada (smooth). Tipicamente
encontrado em inimigos do tipo "canhões".
[Look at Target] <behavior> {
      target: <entity>
      smooth: yes / no
}
```

Figura 50 - Atributo "target" espera por uma referência a um componente do tipo <entity>.

3.7.2 Composição

O relacionamento de **composição** define a característica fundamental à abordagem deste trabalho. É ela que permite a estruturação dos aspectos formadores de jogos e gêneros. Em termos práticos, a composição possibilita que os componentes sejam agrupados para formar novos, concedendo a estes novos comportamentos e características. Analogamente, componentes complexos podem ser convenientemente descritos como composições de partes menores.

A composição implica na existência de diferentes **níveis de abstração**. Há medida que a composição cresce, os componentes passam a ter descrições mais próximas dos aspectos comunicados em linguagem textual. Analogamente, à medida que se investiga a constituição de um componente, descendo-se em sua hierarquia de subtipos, observa-se gradativamente um maior detalhamento estrutural e comportamental, chegando-se a componentes mais "elementares", que representam aspectos simples. Dessa forma, pode-se dizer que ocorre um **encapsulamento de complexidade**, uma vez a mesma encontra-se oculta nos níveis mais baixos de abstração de componentes. Tal encapsulamento permite que aspectos internamente complexos sejam referenciados por um vocabulário estruturado de alto nível.

A Figura 51 mostra um exemplo do emprego da composição para definir as características e comportamentos de um personagem hipotético de um jogo 2D side scroller de plataforma e ação. O personagem identificado pelo componente [2D Action-Platformer Character] possui uma série de movimentos, que estão modelados como composições na Figura 51 e são discutidos na sequência.

```
>> Exemplo de personagem de jogo 2D side scroller de plataforma/ação.
[2D Action-Platformer Character] <player>
{
      >> Move-se para frente e para trás
      [Move in Two Directions] <movement>
      {
            tight controls = yes
      }
      >> Pulo duplo girando para atacar oponentes
      [Double Spin Jump] <jump> <attack>
      {
            timed = no
            repeatable = single

            [Jump] <action>
            {
                  controlable = yes
                  height = fixed
                  affected by motion = both (by player speed)
                  auto assisted = no
            }
            [Spin Attack] <attack>
            {
                  axis = horizontal
                  action by = button press
            }
      }
```

Movimento instantâneo e preciso.

Pulo duplo com giro horizontal de espada. A altura é fixa e o controle é preciso.

```
      >> Ataque curto
      [Straight Attack] <melee attack>
      {
            repeatable = combo
            attack while moving = no
            combo = [One-Button Combo] <attack combo>
      }

      >> Corrida com ataque, finalizando o movimento
      >> Pode também somente correr e não atacar
      [Dash Attack] <run> <attack>
      {
            chain press = no

            [Quick Run] <run>
            {
                  action by = twice forward
                  button hold = yes
                  instant move = yes
            }
            [Straight Attack] <melee attack>
            {
                  repeatable = single
                  combo = none
            }
      }
}
```

Sequência de ataques executada pressionando-se o mesmo botão.

Corre ao pressionar duas vezes a seta para frente. Pode correr e depois

Figura 51 - Modelagem de personagem de jogo 2D side scroller de plataforma e ação.

Os movimentos modelados no componente [2D Action-Platformer Character] são sumarizados abaixo:

- **Movimentos**: o personagem do jogador pode ser movimentado no eixo horizontal [Move in Two Directions]. Ele muda a direção imediatamente para a aquela apontada pelo jogador (esquerda ou direita). É preciso considerar que ele se move para frente ou para trás, mas a perspectiva do jogador é lateral (perpendicular à direção do personagem). Ele também pode correr instantaneamente ([Quick Run]) ao pressionar duas vezes o botão direcional para a direção desejada. Este comportamento está modelado no componente [Dash Attack]. Alternativamente, poderia ter sido definido um componente [Move, Dash and Attack], que encapsularia os três tipos de ações. O componente [Move in Two Directions] representa o tipo de movimentação característico a jogos 2D de visão lateral (side scrollers) e está ilustrado na Figura 52.

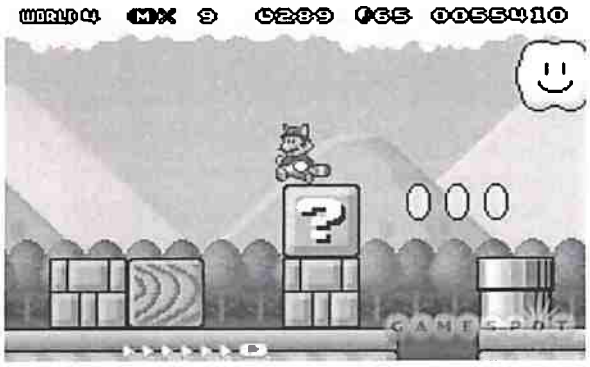

Figura 52 - Em jogos de visão lateral o jogador tipicamente move o personagem em duas direções: para frente e para trás.

- **Pulo**: o personagem pode pular, executar pulo duplo (segundo impulso no ar) e um ataque giratório enquanto pula (uma só vez). O componente é uma generalização do movimento encontrado em jogos como Ninja Gaiden, Revenge of Shinobi[114], Empire Strikes Back[115] e Megaman X4[116] (Figura 53).

Figura 53 - Vários jogos 2D de plataforma e ação empregam o pulo duplo com ataque giratório.

- **Ataques**: além do ataque giratório no pulo, o personagem pode executar um ataque curto enquanto estiver no chão. Ao pressionar o botão repetidamente, o personagem executa uma sequência de ataques [One-Button Combo]. Jogos do gênero Beat'em Up[117] utilizam comumente este tipo de combo simples de ataques. Outro movimento comum em jogos do gênero, a corrida com ataque [Dash Attack], também está presente na modelagem do personagem. Ele está ilustrado na foto do jogo Golden Axe[118], exibida na Figura 54.

[114] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/Shinobi
[115] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/SuperStarWars
[116] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/MegaManX4
[117] https://en.wikipedia.org/wiki/Beat_%27em_up
[118] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/GoldenAxe

Figura 54 - Beat'em Ups tipicamente possibilitam o movimento de corrida com ataque.

No exemplo previamente discutido, é possível notar que componentes podem apresentar diferentes níveis de complexidade. Dessa forma, não há restrições quanto ao "tamanho estrutural" de um componente, seja pelo número de componentes que o constituem ou pela quantidade de níveis de abstração que apresenta. Nesse contexto, a composição traz consigo uma relação fundamental ao conceito de blocos de construção e à abordagem "top-down" de design: componentes complexos podem ser desmembrados em outros, mais simples. Por esse mecanismo simples, jogos e gêneros podem ser descritos como composições (ver exemplo do jogo [Orbit Jumper], Figura 47). Ademais, novos jogos podem ser expressos como a união de outros existentes, seja pelo reuso completo das partes destes ou pela seleção daquelas interessantes ao novo projeto. O mesmo processo pode ser aplicado à descrição de gêneros.

3.7.3 Classificação

A classificação corresponde ao terceiro tipo de relacionamento que os componentes podem apresentar. Ela tem por função definir os tipos para os componentes, organizando-os em hierarquias de tipos e subtipos, ou similarmente, de componentes genéricos e especializados. Funcionalmente, ela age de forma similar à herança, uma construção fundamental do paradigma de OO para generalização e especialização de classes. No desenvolvimento orientado a objetos, a herança permite que uma classe possa incluir características de outras, reusando dados e comportamentos. Nessa relação, a classe herdante é conhecida como subtipo ou classe especializada, e a classe herdada, é chamada de supertipo ou classe genérica. Nesse sentido, um relacionamento "é-uma" passa a ser definido entre as classes participantes da herança, uma vez que a herdeira torna-se um tipo mais especializado da herdada.

No contexto da abordagem deste trabalho, a classificação serve como um mecanismo rotulador de componentes, organizando-os hierarquicamente. Isso ajuda a entender as relações entre componentes que identificam aspectos similares. Além disso, a classificação especializa os componentes, definindo subtipos para aplicações e contextos específicos, tal como jogos e gêneros. Dessa forma, a nomenclatura "tipo-de" torna-se mais adequada para descrever a forma

como a classificação ocorre entre componentes. Um componente classificado passa a embutir implicitamente o componente classificador em sua constituição. Este último é chamado de **componente implícito** e pode, a critério do designer, ser explicitamente descrito no corpo do componente especializado. Cabe ressaltar que um componente pode receber vários classificadores, tornando-se uma especialização destes. Além disso, em vias de simplificar a modelagem, é possível que os componentes internos a um classificador sejam diretamente representados no corpo do componente classificado.

A Figura 55 mostra dois exemplos de componente definidos sobre mais de um classificador. O "pulo duplo com ataque giratório" [Double Spin Jump] é rotulado como um tipo específico de "pulo" e "ataque". Por esta razão, ele contém os componentes classificadores [Jump] e [Attack]. De forma semelhante, a "corrida com ataque" representa um tipo especializado de "corrida" e "ataque", contendo [Run] e [Attack]. Em uma visualização hierárquica, ambos componentes estarão relacionados como tipo-de "ataque". Cabe ressaltar que além da organização de aspectos de jogos, a classificação estrutura tipos e subtipos de jogos e gêneros. Nesse sentido, o exemplo previamente documentado na Figura 47 exibiu a modelagem de componentes para o design de Orbit Jumper, que está rotulado como um tipo-de "jogo infinito de plataformas".

```
>> Pulo duplo girando para atacar oponentes. A altura é fixa, não
importando o tempo de pressionamento do botão de pulo. O controle
durante o pulo é preciso e o ataque pode ser desferido a qualquer
momento, somente uma única vez.
[Double Spin Jump] <jump> <attack>
{
      timed = no
      repeatable = single

      [Jump] <action>
      {
            controlable = yes
            height = fixed
            affected by motion = both (by player speed)
            auto assisted = no
      }

      [Spin Attack] <attack>
      {
            axis = horizontal
            action by = button press
      }
}

>> Corrida com ataque, finalizando o movimento. Permite também somente
a realização da corrida.
[Dash Attack] <run> <attack> {
      rapid press = no

      [Quick Run] <run>
      {
            action by = twice forward
            button hold = yes
      }
```

```
        [Straight Attack] <melee attack>
        {
                repeatable = single
                combo = none
        }
}
```

Figura 55 - O componente "pulo duplo com ataque giratório" é um tipo de "pulo" e "ataque".

O mecanismo de classificação oferece uma abordagem conveniente para a estruturação dos aspectos de design. Nesse sentido, variações de aspectos encontrados em jogos podem ser modelados como um componente genérico, repleto de atributos que caracterizem as diferenças sutis entre esses aspectos. Neste caso, como parte de um vocabulário cotidiano, a comunicação prática deste componente por meio de seus atributos torna-se morosa e inacessível. Por outro lado, mapear aspectos similares para componentes diferentes que sejam ligados hierarquicamente pela classificação pode estimular o enriquecimento de um vocabulário de uso comum aos designers. Nesse sentido, a relação "tipo-de" reforça que os tipos de componentes devem ser preferencialmente representados pela classificação, e não somente por diferenças em valores de atributos.

A Figura 56 apresenta um exemplo de aspectos que são convenientemente estruturados como classificações, ao contrário de um único componente genérico. A ação de "correr" é um aspecto que apresenta pequenas variações nos jogos que a empregam. Em alguns, a "corrida" é uma forma de mover-se rapidamente pelo cenário, sem limites para a realização do movimento. Em outros, emprega-se o sprint, uma corrida rápida, intensa e limitada, usualmente associada a um medidor de fôlego, que recarrega após o uso. Há também aqueles que usam o dash, um movimento horizontal no qual o personagem "desliza" rapidamente para frente, popularmente encontrado em histórias de animes e mangás. É interessante notar que, embora tenham nomes diferentes, em termos do conceito tratado, todos referem-se à ação de "correr" e suas peculiaridades podem ser representadas por meio de valores de atributos do componente [Run] <action>. Por outro lado, comunicar as variações de tipos da ação de correr por meio de atributos não é algo prático. Dessa forma, os componentes [Sprint] <run> e [Dash] <run> são representados como especializações de [Run] pelo uso da classificação, definido os valores adequados para os atributos recebidos. Assim, forma-se um vocabulário estruturado hierárquico de termos de design, organizados sob o formato de componentes. Ainda no mesmo exemplo, o aspecto de "dash aéreo" [Air Dash], foi acrescentado à hierarquia para ressaltar a versatilidade da abordagem de componentes ao estruturar um vocabulário de design uso comum. É interessante ressaltar que o processo de identificar os componentes e montar a hierarquia de termos em si torna-se um exercício de design, exigindo uma reflexão sobre os aspectos de jogos envolvidos, bem como suas peculiaridades e relações.

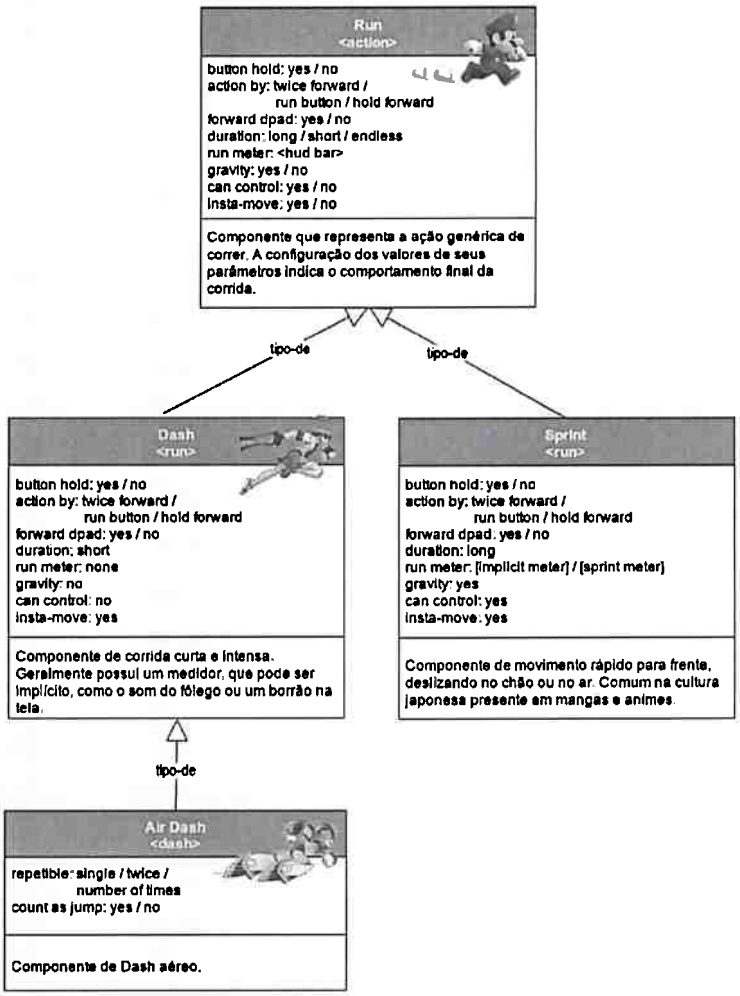

Figura 56 - Diagrama da hierarquia de componentes gerada pelas formas peculiares de movimentos de "corrida" encontrados em jogos.

A Figura 57 expressa o exemplo anterior por meio da linguagem de componentes, explicitando a estrutura interna aos componentes, bem como, os atributos e seus valores. Os componentes [Sprint] e [Dash] são especializações de [Run], apenas embutindo-o pela classificação e determinando os valores de alguns de seus atributos a fim de caracterizar as peculiaridades de cada variação do aspecto de "corrida".

```
>> Componente que representa a ação genérica de correr. A configuração
dos valores de seus parâmetros indica o comportamento final da
corrida.
[Run] <action>
{
      button hold: yes / no (precisa manter botão pressionado?)
      action by: twice forward / run button / hold forward
      forward dpad: yes / no (precisa complementar com direcional?)
      duration: long / short / endless
      run meter: <hud bar>
      gravity: yes / no (sem gravidade move-se em linha reta)
      can control: yes / no (direciona personagem enquanto corre?)
      insta-move: yes / no (corre instantaneamente ou acelera)
}
```

```
>> Componente de corrida curta e intensa. Geralmente possui um
medidor, que pode ser implícito, como o som do fôlego ou um borrão na
tela.
[Sprint] <run>
{
      [Run] <action>
      {
            button hold: yes / no (precisa manter botão pressionado?)
            action by: twice forward / run button / hold forward
            forward dpad: yes / no
            duration: long
            run meter: [implicit meter] / [sprint meter]
            gravity: yes (não consegue mover em linha reta no ar)
            can control: yes (direciona personagem enquanto corre)
            insta-move: yes (corre instantaneamente)
      }
}

>> Componente de movimento rápido para frente, deslizando no chão ou
no ar. Comum na cultura japonesa presente em mangás e animes.
[Dash] <run>
{
      [Run] <action>
      {
            button hold: no (precisa manter botão pressionado?)
            action by: twice forward / run button / hold forward
            forward dpad: no (precisa complementar com direcional?)
            duration: short
            run meter: none
            gravity: no (sem gravidade, move-se em linha reta no ar)
            can control: no (não controla personagem enquanto corre)
            insta-move: yes (corre instantaneamente)
      }
}

>> Componente de Dash aereo.
[Air Dash] <Dash>
{
      (pode executar quantas vezes no ar?)
      repetible: single / twice / number of times
      (conta como pulo, geralmente quando há pulo duplo)
      count as jump: yes / no
}
```

Figura 57 - A classificação estimula a construção de um vocabulário compartilhado de design.

O mecanismo de classificação permite que aspectos de design similares sejam organizados hierarquicamente em relações "tipo - subtipo". Dessa forma, ocorrências peculiares em jogos e gêneros podem ser acomodadas em um mapa de relacionamento sintético, que permite ascender na hierarquia para visualizar a origem de um determinado aspecto em um jogo (generalização), bem como, descender para encontrar subtipos do mesmo aspecto concebidos para outros jogos (especialização). Essa construção hierárquica favorece o enriquecimento de um vocabulário estruturado de design. O conceito pode ser ilustrado pela hierarquia de componentes exposta na Figura 61. O diagrama organiza componentes de tipos de plataformas identificados em jogos, estruturando um conhecimento de design sobre o tema.

Como pode ser notado no diagrama, a relação "tipo-de" estimula que variações de componentes devem ser representados pela classificação, em vez de diferenças em valores de atributos.

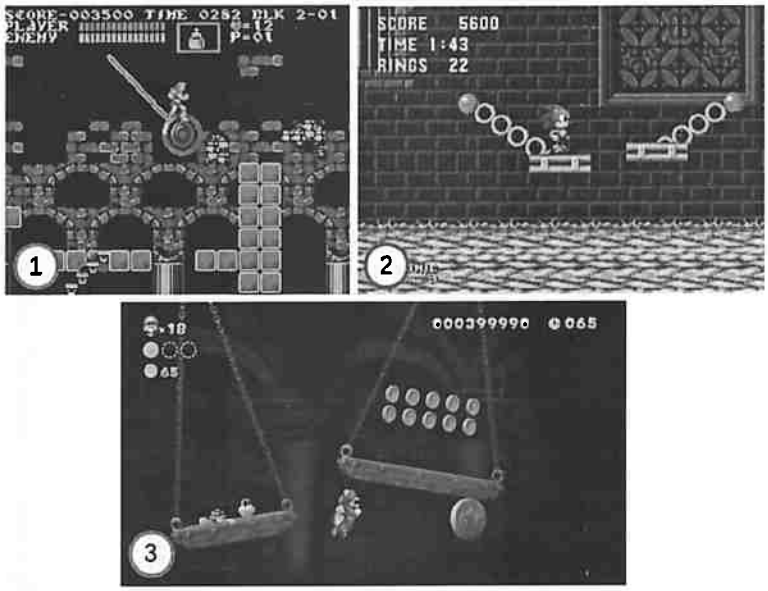

Figura 58 - Componente [Pendular Platform] empregado nos jogos Castlevania (foto 1), Sonic (foto 2) e New Super Mario Bros Wii (foto 3).

A hierarquia modelada na Figura 61 contém definições que identificam tipos peculiares de plataformas comumente encontradas em jogos. Como exemplo, a [Orbital Platform] está presente no jogo Sonic e foi detalhada nas Figuras Figura 43 e Figura 44. O componente foi empregado como aspecto fundamental na modelagem do protótipo [Orbit Jumper] (Figuras Figura 45 e Figura 47). As plataformas do tipo [Pendular Platform] podem ser facilmente identificadas em jogos como Castlevania, Sonic e New Super Mario Bros. Wii (Figura 58).

Figura 59 - Componente [Snake Platform] no jogo Super Mario Bros. Wii e [Balanced Lifting Platform] em Super Mario Bros.

Outros tipos de plataformas registrados na hierarquia de componentes da Figura 61 compreendem a [Snake Platform] e a [Balanced Lifting Platform], empregadas em jogos da série Super Mario, como mostra a Figura 59. É também na mesma série que as plataformas de caminhos pré-determinados ([Path Platform]) são amplamente utilizadas na construção dos níveis de várias gerações de jogos (Figura 60).

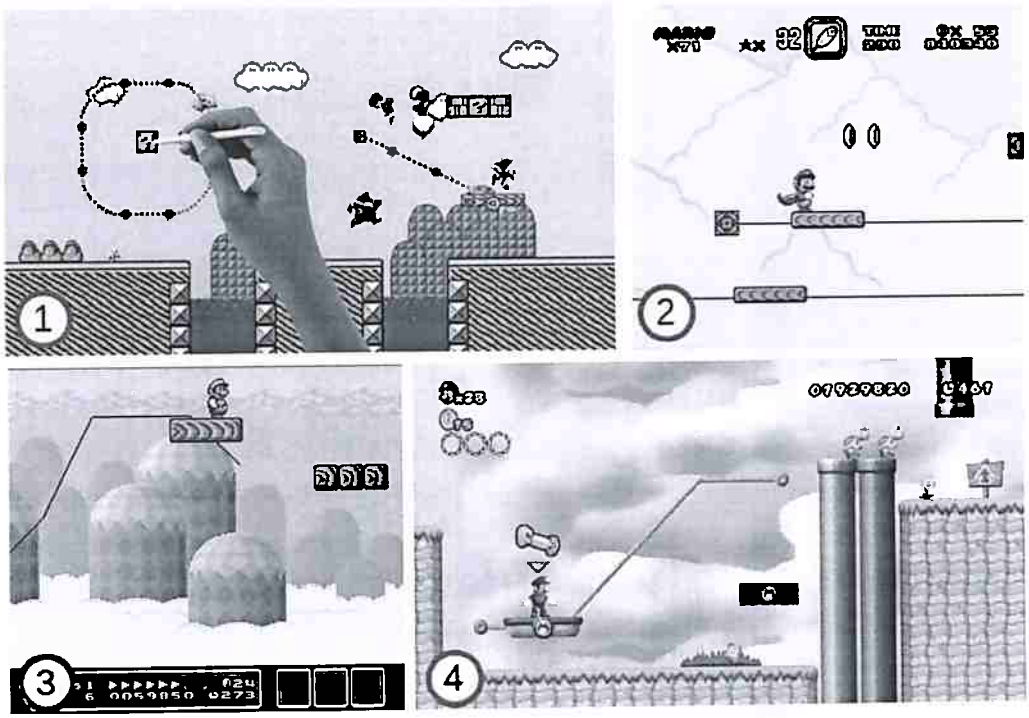

Figura 60 - Plataformas do tipo [Path Platform] são usadas na franquia Super Mario: Super Mario Maker (1), Super Mario World (2), Super Mario Bros. 3 (3) e New Super Mario Bros. Wii (4).

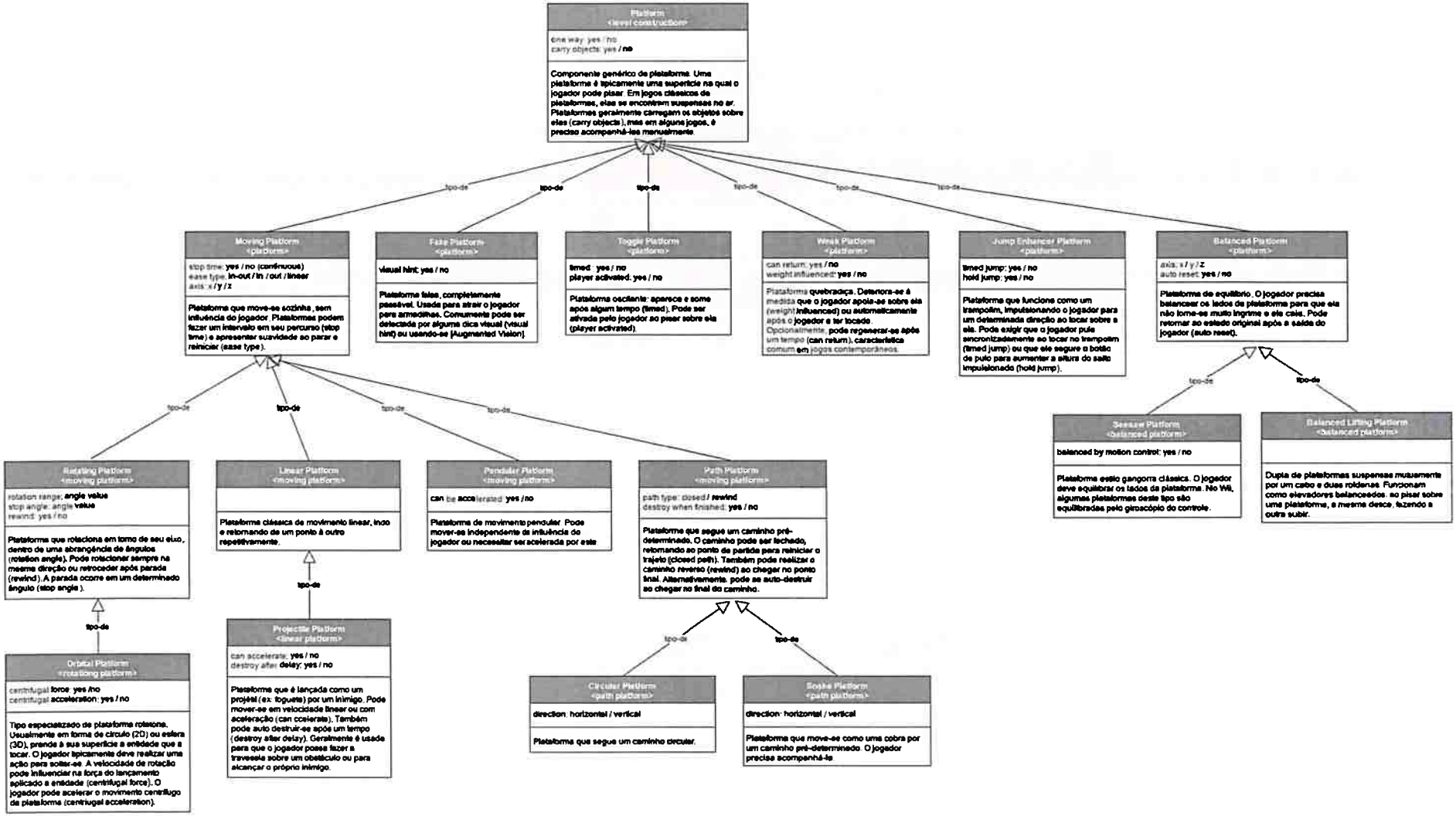

Figura 61 - Hierarquia de componentes do tipo [Platform] construída pela relação de classificação

3.8 Contribuindo com a "engenharia" de design

A natureza heterogênea das pessoas que trabalham como designers de jogos levou à diferentes compreensões sobre o que define o ofício do designer e no que ele se baseia. Durante as duas primeiras gerações de consoles, jogos eram tipicamente construídos por um ou dois programadores, que naturalmente tinham uma forma "engenheira" de pensar. Restrições tecnológicas da época impediam que aspectos como narrativa e cinematografia fossem considerados. Além disso, deve-se somar os fatos de que os jogos que eram representados por poucos pixels e produzidos em cerca de um mês por um único programador. Com o passar das gerações, os jogos cresceram em complexidade, exigindo uma equipe multidisciplinar de produção. O papel do designer passou a ser essencial, como aquele responsável não somente por planejar todos os aspectos da concepção, mas também por garantir que se concretizem durante a produção do jogo. Nesse sentido, embora alguns designers sejam programadores, outros possuem pouco ou nenhum conhecimento técnico acerca do desenvolvimento de um software, sendo originários de outras áreas, como ilustradores, animadores ou roteiristas.

A medida que os jogos se tornaram mais complexos, o enredo deixou de ser somente um pretexto para fundamentar as ações do jogador, e tornou-se um guia para os eventos que ocorrem durante o desenrolar do jogo. Dessa forma, habilidades narrativas passaram a desempenhar um papel importante no design de jogos, que acabou se tornando um processo mais empírico e menos sistematizado (Kreimeier, 2003; Neil, 2012). No entanto, Costikyan apontava em 1994 que, embora essenciais aos jogos de seu tempo, narrativa e história não são as funções primordiais dos jogos, pois eles não seguem uma estrutura linear passiva: a história é contada através das ações do jogador e o resultado final emerge dessa experiência. LeBlanc, Hunicke e Zube (LeBlanc et al., 2004) apresentam uma perspectiva semelhante, enfatizando a natureza dinâmica dos jogos como sistemas que apresentam comportamentos emergentes gerados pelas experiências de *gameplay*. Para eles cada jogador tem uma percepção diferente, que surge a partir de suas interações com as diferentes mecânicas implementadas no jogo.

Recentemente, Albernathy e Rouse (2014) fizeram uma apresentação na GDC de 2014 na qual apresentam e discutem os resultados de uma pesquisa que tentou mostrar a relevância concreta do enredo de jogos para os jogadores. Segundo os autores, embora o enredo esteja bastante proeminente nos jogos contemporâneos, seja por narrações do personagem controlado ou por *cutscenes* mescladas ao *gameplay*, ele mostra-se irrelevante à grande maioria dos jogadores entrevistados. Nesse sentido, os resultados da pesquisa mostraram que os jogadores dificilmente se recordam da história dos jogos que utilizaram. O mesmo não ocorre para filmes e seriados televisivos, que têm seu enredo claramente descrito pelos mesmos jogadores. Contudo, é interessante notar que os jogadores se lembram nitidamente dos personagens que controlam: não pelo papel que assumem na trama, mas pelas formas de interação com o mundo do jogo. Ainda segundo a pesquisa, situações e eventos de *gameplay* também são detalhadamente descritos por jogadores. Albernathy e Rouse enfatizam que jogos se diferem essencialmente de filmes, TV e livros, nos quais o utilizador é um mero observador: os jogadores atentam à sua experiência no jogo, construindo um roteiro a partir das sequencias

de interações com o mundo jogo. Os autores concluem que os jogos contemporâneos valorizam em demasia o enredo e, por outro lado, deveriam estar mais focados em construir personagens, interações e situações de *gameplay*. Assim como Costikyan (1994) e LeBlanc et al. (2004), o trabalho de Albernathy e Rouse (2014) mostra que as experiências do jogador emergem de suas interações com os aspectos – mecânicas e características – implementados no jogo e não por sua narrativa. Essa é a premissa da perspectiva da "engenharia de design", tomada como fundamento nos trabalhos discutidos na seção 2.2. Como previamente apresentado, este trabalho enquadra-se na mesma perspectiva. Dessa forma, acredita-se que a abordagem de componentes pode ajudar na definição dos aspectos do jogo que modelam a experiência do jogador.

Como pode ser visto neste capítulo, as relações de composição e classificação assumem funções mutuamente complementares na modelagem de componentes: a classificação tem um papel estrutural e a composição age como um mecanismo de descrição comportamental. Nesse sentido, a classificação é utilizada para definir tipos e subtipos de componentes, conceituando-os e categorizando-os com rotuladores. Ela ajuda a criar um vocabulário hierárquico de design a partir de componentes que organiza. Por outro lado, a composição age na descrição do comportamento ou funcionamento dos componentes e contribui com a abstração da complexidade na descrição de aspectos de jogos. Dessa forma, permite representar conceitos em alto nível de abstração como a composição de partes menores. Os três tipos de relações apresentadas nesta seção – associação, composição e classificação – denotam as principais ferramentas à disposição do designer para expressar, registrar e comunicar o conceito do jogo sob a abordagem de componentização.

Capítulo 4 Demonstração da Abordagem de Componentes

No decorrer dos capítulos anteriores foram apresentados diversos exemplos que demonstram características da abordagem de componentes proposta. Em alguns desses exemplos, a linguagem de componentes foi utilizada para extrair e documentar componentes que representam aspectos identificados em jogos existentes no mercado. Em outros, especialmente naqueles presentes na seção 3.6.3, a abordagem foi empregada para criar o design do protótipo de um novo jogo, com base em componentes previamente identificados. Esses dois cenários – obtenção de componentes de jogos existentes e aplicação em novos projetos – compreendem o modelo de demonstração abordado neste capítulo. Além disso, os cenários são mapeados diretamente para duas práticas essenciais da abordagem: a análise por decomposição e o design por composição. No entanto, os exemplos prévios não permitiram a experimentação direta dos aspectos estéticos dos componentes documentados: essa atividade fica confiada ao conhecimento empírico do leitor. Por essa razão, este capítulo foi planejado para permitir que o leitor faça a experimentação dos aspectos discutidos.

Em contraste aos modelos de validação e avaliação aplicados nos trabalhos relacionados (seção 2.2), que empregaram métodos subjetivos e discussões teóricas, esta tese como parte de objetivos caracterizar o comportamento da abordagem em cenários práticos. Essa caracterização compreende-se pela construção dois estudos de casos, planejados como demonstrações da abordagem para a análise e o design de jogos. Mais especificamente, o processo de demonstração é compreendido por três etapas, que arremetem ao modelo de sistematização do processo de design apresentação na seção 2.3.2.5: análise -> design -> experimentação. Dessa forma, a demonstração da abordagem é compreendia pelas atividades enumeradas abaixo e pode ser visualizada no esquema da Figura 62:

1. Aplicar a análise pela decomposição em jogos ou gêneros existentes, a fim de obter aspectos significativos de jogos populares para identificá-los como componentes. O resultado dessa atividade é o design do jogo ou gênero utilizando componentes.

2. Buscar novos componentes em outros jogos/gêneros e aplicar o design por composição utilizando os componentes documentados e os agregados, combinando-os e modificando-os para projetar um novo, que possa demonstrar clareza na aplicação desses componentes. O resultado dessas atividades é o design do novo jogo utilizando componentes.

3. Construir o protótipo do jogo projetado a fim de permitir uma melhor compreensão dos componentes empregados sob a perspectiva do jogador. Nesse sentido, espera-se oportunizar ao leitor a exploração dos aspectos implementados no protótipo do jogo e a experimentação da estética relacionada aos componentes empregados. O resultado desta etapa é o protótipo jogável do novo jogo projetado.

   A respeito do terceiro item da lista acima, faz-se necessário destacar que o uso dos

protótipos pelo leitor constitui uma atividade enriquecedora, tanto para a compreensão dos aspectos modelados pelos componentes, quanto para percepção da estética que proporcionam. Em termos práticos, a implementação dos protótipos permite ao leitor explorar seus aspectos estéticos e facilita a compreensão das discussões contidas neste capítulo.

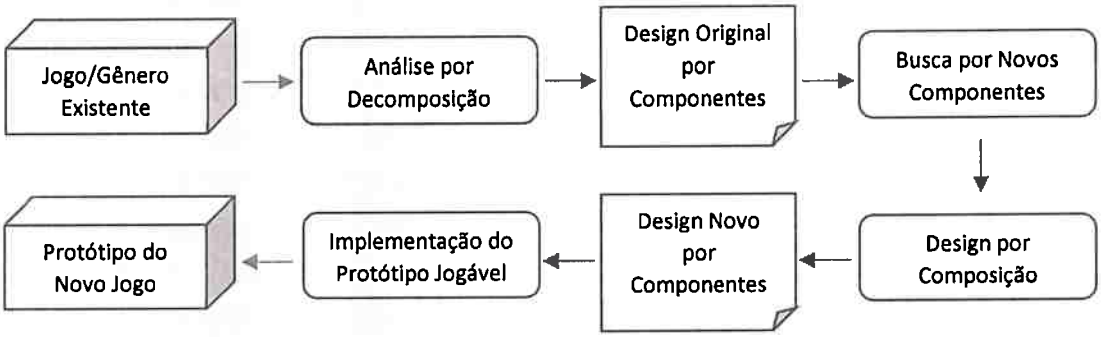

Figura 62 - Recursos e atividades do processo de demonstração.

Para a demonstração da abordagem de componentes foram realizados dois estudos de casos cujos nomes arremetem intencionalmente aos gêneros que os qualificam: Space Shooter (Figura 63) e Platformer (Figura 64). O plano inicial contemplava apenas um estudo de caso, mas concluiu-se que era importante variar as situações modeladas, permitindo ao leitor tipos distintos de experiência de *gameplay*. Dessa forma, os estudos de casos discutidos nas próximas seções foram concebidos norteados pelos seguintes requisitos:

- Demonstrar a abordagem em jogos 2D e 3D, mostrando que os aspectos de jogos independem do formato de apresentação visual empregado, assim como, do tipo de movimentação e perspectiva sobre o mundo do jogo;
- Trabalhar com gêneros distintos, que permitissem variações em aspectos de jogos, tais como tipos diferentes de ações do personagem do jogador, de objetivos primários e secundários, de comportamentos de NPCs (*Non-Player Character*) e de design de *level*;
- Selecionar gêneros populares, que possuam jogos representantes bem conhecidos no mercado, de forma a facilitar ao leitor a associação entre componentes e os aspectos que identificam;
- Aproximar os protótipos a jogos reais do mercado, especialmente em termos dos aspectos implementados e da estética que proporcionam. O foco foi deixá-los o mais próximo possível de jogos reais, não apenas tratando-os como exemplos hipotéticos. Acredita-se que essa característica seja essencial para a experimentação dos jogos pelo leitor e para a percepção e identificação do emprego de componentes extraídos de jogos conhecidos.
- Favorecer a diversidade de componentes ante a quantidade. Aumentar a quantidade de aspectos do jogo, como criar diversos levels, implementar vários tipos de inimigos e tiros, traria pouco impacto à demonstração da abordagem. Por outro lado, incrementar a diversidade corrobora a aplicabilidade da abordagem a situações variadas.

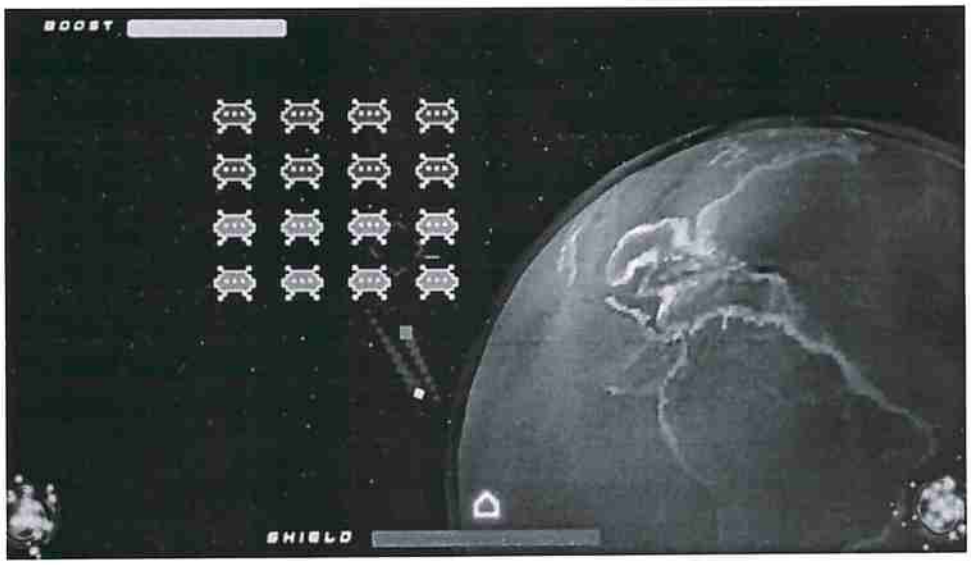

Figura 63 - Tela do jogo Space Shooter.

A estratégia empregada para a demonstração teve como foco abranger uma maior diversidade de aspectos de jogos para fundi-los em protótipos convincentes de jogos. Nesse sentido, os dois protótipos arremetem a jogos e gêneros do mercado. Em ambos os casos, o processo de desenvolvimento foi iniciado pela análise de componentes de jogos ou gêneros conhecidos. A partir desta mistura inicial, componentes de títulos clássicos e contemporâneos foram sendo extraídos e inseridos no design dos novos jogos. O primeiro protótipo (Figura 63) teve como base o jogo Space Invaders[119] e foi acrescido de vários componentes de forma a torna-lo uma versão contemporânea do primeiro. O segundo protótipo (Figura 64) foi montado sobre a composição do gênero [Platformer], com o posterior acréscimo de componentes provenientes de vários jogos, tornando-o um título 3D simplificado de plataformas em mundo aberto. A despeito da necessidade de torna-los próximos a jogos reais, como previamente exposto, não foi possível construir jogos completos. O tempo e o esforço necessário para o fazê-lo tornava a atividade inviável. Além disso, concluiu-se que não era necessário construir jogos com diversos *levels*, inimigos e situações. Nesse sentido, a abordagem é invariante quanto a quantidade de objetos no jogo, em oposição à diversidade, que é o ponto chave para a demonstração. Por essa razão, entendeu-se que protótipos de jogos eram suficientes para caracterizar a aplicação da abordagem. Assim, aplicou-se uma restrição quanto ao tamanho dos protótipos, restringindo a quantidade de inimigos, *levels*, obstáculos e itens construídos. De uma forma geral, o intuito foi apresentar os principais componentes que poderiam caracterizar os protótipos para o jogador, tanto no contexto dos jogos clássicos que se inspiram quanto nos aspectos emprestados de jogos contemporâneos.

[119] https://en.wikipedia.org/wiki/Space_Invaders

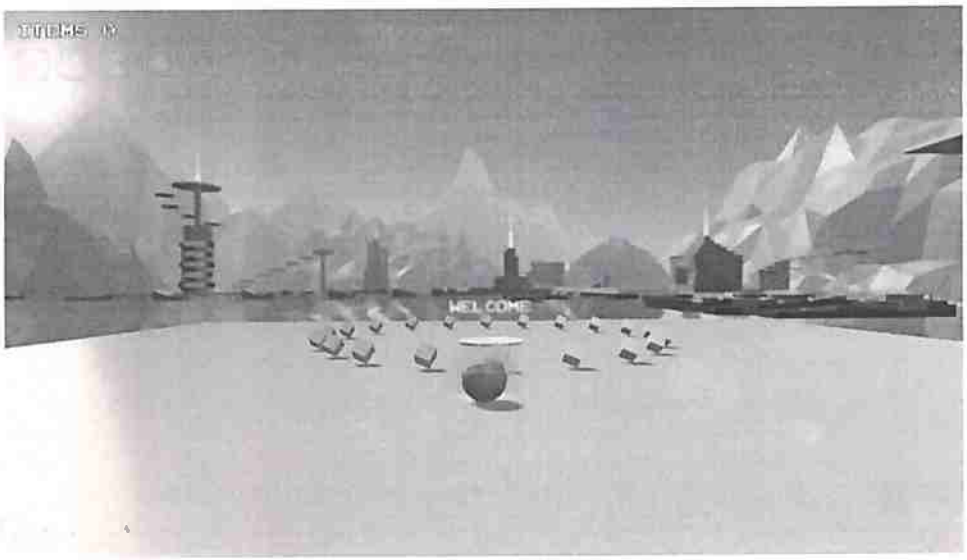

Figura 64 - Tela do jogo Platformer.

4.1 Estudo de Caso 1: Space Shooter

O protótipo Space Shooter tem por base o design do jogo Space Invaders, lançado pela Taito Corporation em 1978 (Figura 66). De forma sucinta, nele o jogador deve controlar uma nave localizada na base da tela e atirar nos NPCs inimigos para destruí-los e progredir ao próximo *level*. O jogo representa a base para os jogos de tiro (gênero *shooter*[120]), nos quais a premissa é comumente utilizar uma arma de fogo para disparar e destruir os alvos apresentados na tela. A composição do gênero shooter é mostrada na Figura 65. Em linhas gerais, em um jogo do gênero o jogador controla um **personagem** [Shooter Character] que lhe permite realizar **disparos** [Shoot] contra NPCs **inimigos** [Enemy] em via de destruí-los. Os componentes [Shoot] e [Enemy] representam tipos mais genéricos e serão substituídos por outros mais especializados no emprego de [Shooter]. Cabe ressaltar que um [Shooter] é considerado um subgênero de [Action][121], no qual comumente existe um foco em combates <attack> e o jogo ocorre em um ritmo rápido [Fast Paced]. Portanto, o componente [Action] age como componente classificador de [Shooter].

```
[Shooter] <action> <genre>
{
      [Fast-Paced] <concept>

      [Shooter Character] <player> {
            <attack>
            [Shoot] <attack>
      }

      [Enemy] <entity>
}
```

Figura 65 - Composição do gênero Shooter

[120] https://en.wikipedia.org/wiki/Shooter_game
[121] http://tvtropes.org/pmwiki/pmwiki.php/Main/ActionGame

A respeito do componente [Shooter], que modela o gênero de mesmo, os componentes internos [Shoot] e [Enemy] podem ser facilmente identificados no Space Invaders e no Space Shooter (Figura 66). O componente [Enemy] é uma entidade (ser que existe no mundo do jogo) que como função prejudir o personagem do jogador, seja atacando-o ou simplesmente existindo no cenário para ser evitado. O componente [Shoot] denota a ação de disparar tiros, tipicamente com uma arma de fogo.

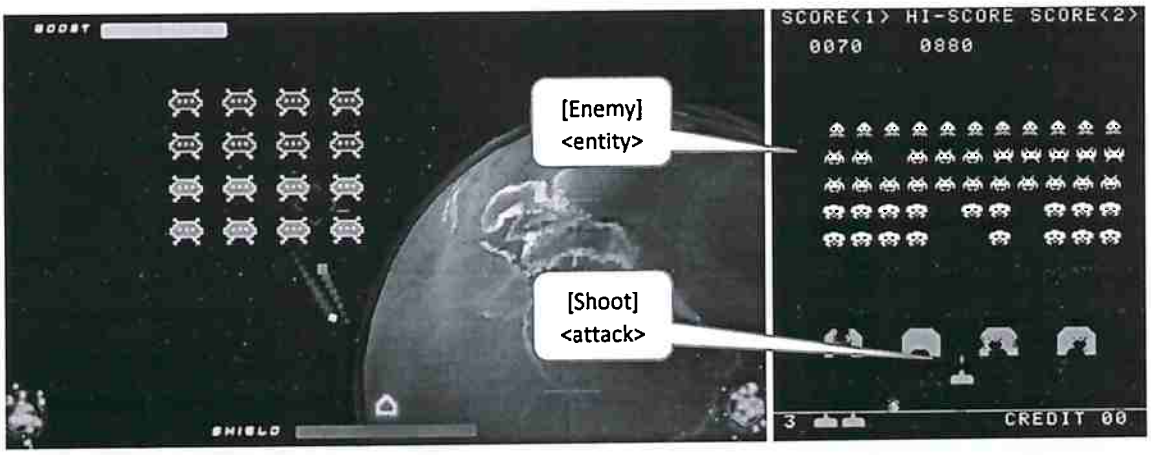

Figura 66 – Protótipo Space Shooter face ao seu originador, Space Invaders (1978).

4.1.1 Análise por Decomposição

O protótipo Space Shooter é facilmente associável ao seu originador, o jogo Space Invaders. De fato, a base utilizada para o design do protótipo, que é apresentado na próxima seção, foi o título original de 1978. O jogo em questão foi escolhido por ser considerado um dos principais representantes do gênero [Shooter][122] e precursor dos jogos Shoot'em Up[123] [124]. De forma similar, o jogo Space Shooter se enquadrada especificamente sob o subgênero [Shoot'em Up], no qual tipicamente o personagem controlado pelo jogador possui munição ilimitada [Unlimited Ammo], coleta melhorias para armas e características [Power-Up], deve destruir inimigos e desviar de um número exagerado de projéteis inimigos [Bullet Hell], que comumente cobrem toda a extensão da tela[125] [126]. Esteticamente, este tipo de jogo exige reflexos rápidos do jogador [Fast-Paced] para desviar de inimigos e seus projéteis [Bullet Hell], ao mesmo tempo em que o faz sentir-se capaz de destruir a grande quantidade de inimigos lançados na tela, ao prover-lhe munição infinita [Unlimited Ammo]. A Figura 67 mostra o design do componente [Shoot'em Up], evidenciando o conteúdo implicitamente agregado pelo componente classificador <shooter>. Pela classificação, um [Shoot'em Up] é um subgênero de [Shooter] que, por sua vez, é subgênero de [Action].

---

[122] https://www.washingtonpost.com/lifestyle/kidspost/minecraft-space-invaders-among-video-game-hall-of-fame-contenders/2016/03/30/32aaf962-e7d9-11e5-bc08-3e03a5b41910_story.html
[123] http://tvtropes.org/pmwiki/pmwiki.php/Main/ShootEmUp
[124] https://en.wikipedia.org/wiki/Shoot_%27em_up
[125] http://tvtropes.org/pmwiki/pmwiki.php/Main/BulletHell
[126] http://www.giantbomb.com/bullet-hell/3015-321/

```
[Shoot'em Up] <shooter> <genre>
{
	[Unlimited Ammo] <modifier>
	[Bullet Hell] <concept>
	[Power-Up] <item>
	[Fast-Paced] <concept>

	[Shooter Character] <player> {
		<attack>
		[Shoot] <attack>
	}

	[Enemy] <entity>
}
```

Figura 67 - Design do gênero [Shoot'em Up].

Considerando as representações prévias dos componentes [Shoot'em Up] (Figura 67), [Game] (Figura 46) e [Endless Game] (Figura 35), a modelagem de design extraída do jogo Space Invaders está apresentada em versão sintética na Figura 68. Nesse sentido, os componentes implícitos agregados ao [Space Invaders] pelos classificadores <endless game> e <shoot'em up> estão omitidos. A respeito do segundo classificador, embora o jogo seja comumente considerado o precursor no gênero [Shoot'em Up], os aspectos [Bullet Hell] e [Fast-Paced] ocorrem de maneira branda, possivelmente por limitações técnicas da época em que o jogo foi produzido. Dessa forma, a estética que o componente [Bullet Hell] gera – tensão de ver-se envolto em uma enorme quantidade de projéteis inimigos e a súbita necessidade de escapar dessa situação – é mais difícil de ser percebida pelo ritmo lento do jogo, especialmente se comparado ao estilo contemporâneo de jogos de tiro, que possuem ação rápida e exigem reflexos de seu utilizador. As limitações técnicas também impactam no componente [Fast-Paced], uma vez que o ritmo do jogo é demasiadamente lento para os padrões atuais. Cabe também ressaltar que o Space Invaders não possui [Power-Up] e, por esta razão, este último está removido de sua composição. Nesta análise, é necessário considerar a época em que o jogo foi desenvolvido (1978) o fato de ser precursor nos gêneros Shooter e Shoot'em Up.

```
[Space Invaders] <shoot'em up> <endless game> <game> {
	[One-Screen World] <game world>
	[Top-View] <pov>
	[Destructible Cover] <cover>
	[Lives] <status>

	<objectives>:
	>> jogador deve destruir todos os inimigos para progredir
	[Destroy Enemies] <primary objective> {
		achieve: [Game Level Progression]
	}
	>> jogador deve impedir que inimigos avancem à base da tela
	[Prevent Enemy Advance] <primary objective> {
		prevent: reach screen base
	}

	<game progression>:
	>> jogador percebe a progressão ao passar para o próximo level
	[Game Level Progression] <progression>
}
```

Figura 68 - Modelagem sintética do jogo Space Invaders.

Uma versão expandida da modelagem de design do jogo Space Invaders é apresentada na Figura 69. Dessa forma, os componentes implícitos de seus classificadores estão explicitamente declarados e colorizados para facilitar sua identificação e origem. A respeito de seu design, Space Invaders é um [Shoot'em Up] infinito, repetindo o mesmo estágio (*level*) e aumentando progressivamente a dificuldade. O jogo caracteriza-se pelo jogador controlar um personagem que pode ser movido lateralmente [Move Sideways] e disparar projéteis [Shoot] ilimitados [Unlimited Ammo]. O mundo do jogo restringe-se à tela [On-Screen World] e o jogador possui uma perspectiva superior [Top-View]. A ação apresenta-se em um ritmo rápido [Fast-Paced]. No cenário, encontram-se inimigos [Enemy] e coberturas destrutíveis [Destructible Cover] que o jogador pode utilizar para proteger-se. O objetivo principal é destruir todos os inimigos na tela [Destroy Enemies] para progredir ao próximo level [Game Level Progression], bem como, para aumentar a pontuação [Score]. Além disso, o jogador deve impedir o avanço dos inimigos [Prevent Enemy Advance], pois caso algum atinja a base da tela, a partida é perdida. Os dois objetivos são comuns a jogos 2D de uma só tela ([One-Screen World]). O level do jogo é único e reinicia com maior dificuldade a cada progressão [Level Loop]. O número de tentativas concedidas ao jogador é limitado [Lives] e, quando exauridas, o jogo reinicia [Start Over].

```
[Space Invaders] <shoot'em up> <endless game> <game>
{
      [One-Screen World] <game world>
      [Top-View] <pov>
      [Destructible Cover] <cover>
      [Lives] <status>

      [Shoot'em Up] <shooter> <genre>
      {
            [Unlimited Ammo] <modifier>
            [Bullet Hell] <concept>
            [Power-Up] <item>
            [Fast-Paced] <concept>

            <player>:
            [Shooter Character] <player> {
                  [Move Sideways] <movement>
                  [Shoot] <attack>
            }
            [Enemy] <entity>
      }

      <objectives>:
      >> jogador deve destruir todos os inimigos para progredir
      [Destroy Enemies] <primary objective> {
            achieve: [Game Level Progression]
      }
      >> jogador deve impedir que inimigos avancem à base da tela
      [Prevent Enemy Advance] <primary objective> {
            prevent: reach screen base
      }
      <game progression>:
      >> jogador percebe a progressão ao passar para o proximo level
      [Game Level Progression] <progression>

      [Endless Game] <game>
      {
```

```
            [Endless Structure] <level structure>
            [Level Loop] <endless structure>
            [Difficulty by Acceleration] <difficulty modifier>
            [Start Over] <game session>

            <objectives>:
            [Reach Further] <primary objective> {
                  achieve: [Score]
            }

            <game progression>:
            [Score] <progress> {
                  register: [High Score]
            }
            [High Score] <counter>
      }
}
```

Figura 69 - Modelagem geral do Space Invaders

Utilizando uma representação com Mapa de Design, o conceito previamente apresentado para o jogo Space Invaders pode ser expresso visualmente como na Figura 70. É necessário ressaltar que a solução de design apresentada expõe os principais conceitos do jogo Space Invaders, omitindo atributos dos componentes e não contemplando aspectos como comportamentos dos inimigos, mostradores e detalhes do jogador. Esta simplificação enfoca uma modelagem compacta do design, uma vez que que as omissões realizadas não afetam a caracterização do Space Invaders. Por outro lado, por uma questão de completude do estudo de caso, os componentes omitidos são apresentados na sequência.

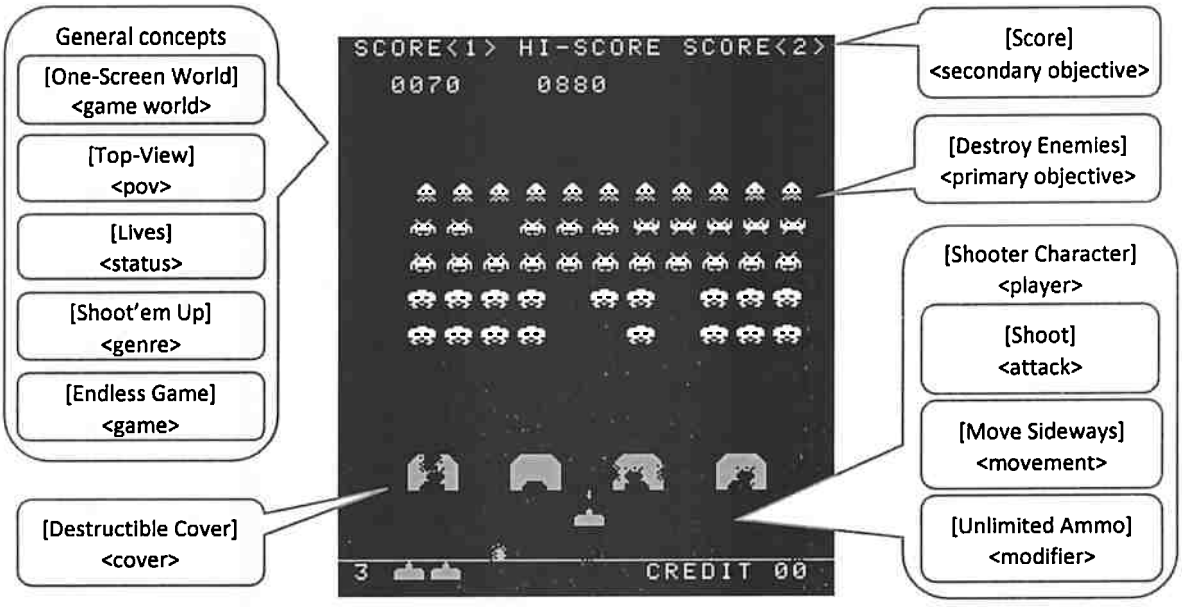

Figura 70 - Mapa de Design para Space Invaders

Na modelagem apresentada, as interações entre projéteis, inimigos e o personagem do jogador ficam implícitas ao classificar componentes como <player> e <enemy>: colisões com inimigos e seus projéteis danificam o jogador. De forma análoga, os projéteis disparados pelo jogador avariam os inimigos. Em uma análise mais detalhada, o personagem do jogador pode ser modelado como na Figura 71. O comportamento [Move Sideways and Shoot] é comumente

encontrado com algumas variações em praticamente todos os jogos Shoot'em Up 2D de visão superior [Top-View] e com mundo de uma tela [One-Screen World], nos quais o jogador controla uma nave. De um modo geral, vários dos aspectos resgistrados pelos componentes apresentados na modelagem do Space Invaders são comuns a diversos jogos de sua época. Exemplos notáveis incluem Megamania[127], Phoenix[128], Astro Blaster[129], Galaga[130], Juno First[131], Titan Attacks![132] e Super Cross Fire[133].

```
[Player Ship] <shooter character> {

      [Move Sideways and Shoot] <behavior> {
            [Move Sideways] <movement> {
                  acceleration: no
                  action by: button hold

                  [Limited to Screen] <modifier>
            }
            <attack>:
            [Straight Shoot] <shoot> {
                  direction: forward
                  damages: [Enemy]
                  rate: one at time

                  [Unlimited Ammo] <modifier>
            }
      }

      [One-Hit Kill] <concept> {
            consumes: [Lives]
      }
}
```

Figura 71 - Modelagem detalhada do personagem do jogador no Space Invaders.

A modelagem apresentada omite os mostradores (<hud>[134]) e a descrição dos comportamentos dos inimigos, que usualmente não são primordiais na caracterização de um jogo ou gênero. Em termos de mostradores, o Space Invaders emprega ícones para informar a quantidade de vidas [Lives Marker] e números para pontos [Score Marker] e recordes [High Score Marker] (Figura 72). A maioria dos jogos exibe informações ao jogador por meio de HUDs textuais ou icônicos, exibidos como uma camada 2D sobreposta à câmera do jogo. Contudo, cabe ressaltar que há determinados componentes de mostradores que podem ser empregados ou omitidos (componente negativo) a fim de estimular determinadas respostas emocionais no jogador. Alguns jogos costumam não utilizam HUDs [No HUD][135] (componente negativo) ou o empregam HUDs diegéticos [Diegetic HUD][136] a fim de enriquecer a sensação de imersão.

[127] http://www.giantbomb.com/megamania/3030-9694/
[128] http://www.giantbomb.com/phoenix/3030-23221/
[129] http://www.giantbomb.com/astro-blaster/3030-22174/
[130] http://www.giantbomb.com/galaga/3030-15784/
[131] http://www.giantbomb.com/juno-first/3030-14153/
[132] http://www.giantbomb.com/titan-attacks/3030-34488/
[133] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/SuperCrossfire
[134] http://tvtropes.org/pmwiki/pmwiki.php/Main/HeadsUpDisplay
[135] http://www.giantbomb.com/no-hud/3015-569/
[136] http://tvtropes.org/pmwiki/pmwiki.php/Main/DiegeticInterface

Exemplos de jogos sem HUD incluem ICO, Another World, King Kong e Tomb Raider (2013). Um HUD diegético é um mostrador fundido ao contexto do jogo, tornando-se parte de seu mundo. Alguns jogos do gênero [First Person Shooter] utilizam mostradores integrados ao capacete do personagem, tais como Halo, Star Wars Republic Commando e Metroid Prime. Além disso, jogos que não utilizam barra de vitalidade ([No Health Bar]) usualmente modificam o aspecto da tela para representar as sensações físicas do personagem à baixa vitalidade, fazendo-a piscar, ficar sobreposta por tons vermelhos ou ofuscá-la [Critical Health]. Em situações nas quais o jogador controla um veículo, as informações são dispostas diretamente no próprio painel do mesmo. O jogo Dead Space é um exemplo notável de emprego de tipos especializados de [Diegetic HUD]. Nele, a barra de vitalidade é localizada na coluna vertebral da armadura [Spine Health Bar], o contador de munição é projetado sobre a arma quando mirada [In-Weapon Ammo Counter] e o inventário é projetado como um holograma sobre o pulso do personagem [Hologram Inventory].

```
[Lives Marker] <hud> {
      type: icon
}

[Score Marker] <hud> {
      type: numeric
}

[High Score Marker] <hud> {
      type: numeric
}
```

Figura 72 - Mostradores usados no Space Invaders.

O jogo Space Invaders apresenta dois tipos distintos de inimigos: [Invader] e [Bonus Ship]. O primeiro é apresentado na Figura 73 caracteriza-se por mover-se lateralmente [Move Sideways], ricocheteando nas laterais da tela [Screen Rebound] e descendo na direção do jogador (base da tela) [Move Straight]. Os inimigos do tipo [Invader] encontram-se dispostos em formação de grade [Grid Formation][137] e movimentam-se como em um enxame sincronizado [Syncronized Swarm][138]. Assim como no veículo do jogador, a movimentação está restrita ao espaço da tela [Limited to Screen], aspecto comum em de jogos [One-Screen World]. O [Invader] é destruído com apenas um projétil do jogador [One-Hit Kill][139], que ao fazê-lo, recebe pontos como recompensa [Points]. O ataque consiste em um tiro direto [Straight Shot], mesmo componente empregado na constituição do personagem do jogador. Além disso, o inimigo inflinge danos ao jogador apenas por enconstar neste [Collision Damage] [140], uma forma de ataque bastante comum em jogos 2D.

```
[Invader] <enemy> {

      [Grid Formation] <enemy formation> {
            [Syncronized Swarm] <modifier>
      }
```

[137] http://tvtropes.org/pmwiki/pmwiki.php/Main/InvisibleGrid
[138] http://tvtropes.org/pmwiki/pmwiki.php/Main/SynchronizedSwarming
[139] http://tvtropes.org/pmwiki/pmwiki.php/Main/OneHitKill
[140] http://tvtropes.org/pmwiki/pmwiki.php/Main/CollisionDamage

```
      [Points] <reward> {
            when: destroyed
      }

      <behavior>:
      [Move and Rebound Towards Player] <behavior> {
            [Move Sideways and Rebound] <behavior> {
                  [Move Sideways] <movement>
                  [Screen Rebound] <modifier> {
                        side: horizontally
                  }
            }
            [Move Straight] <movement> {
                  direction: down
            }
            [Increase Speed] <modifier> {
                  with: time
            }
            [Limited to Screen] <modifier>
      }

      <attack>:
      [Straight Shoot] <shoot> {
            direction: forward
            damages: [Player]
            rate: one at time
      }
      [Collision Damage] <attack>

      [One-Hit Kill] <concept>
}
```

Figura 73 - Modelagem do componente [Invader].

O classificador [Enemy], utilizado para tipificar o componente [Invader], indica que a entidade tem o propósito de danificar o personagem do jogador, dificultando seu progresso no jogo. De uma forma geral, espera-se que um inimigo tenha um comportamento (<behavior>) e um ataque (<attack>), o que está representado no *template* [Enemy] (Figura 74).

```
>> Template para NPC inimigo
[Enemy] <template> <entity>
{
      >> inimigo possui comportamento, mesmo que seja "ficar parado"
      <behavior>
      >> inimigo possui forma de danificar jogador, mesmo por contato
      <attack>
}
```

Figura 74 - Template para um NPC inimigo em jogos.

O segundo tipo de inimigo encontrado no jogo Space Invaders é o [Bonus Ship]. Ele pode ser descrito como um [Bonus Enemy]: uma entidade colocada no jogo para estimular o jogador a destruí-la e obter algum benefício extra. Comumente encontrado em jogos do gênero [Shoot'em Up], ele pode ser visualizado nos títulos R-Type III (ref) e Contra III (ref), cujas fotos estão dispostas na Figura 75. Nos dois jogos, destruir o [Bonus Enemy] fornece um item de melhoria para arma do jogador, que precisa coletá-lo.

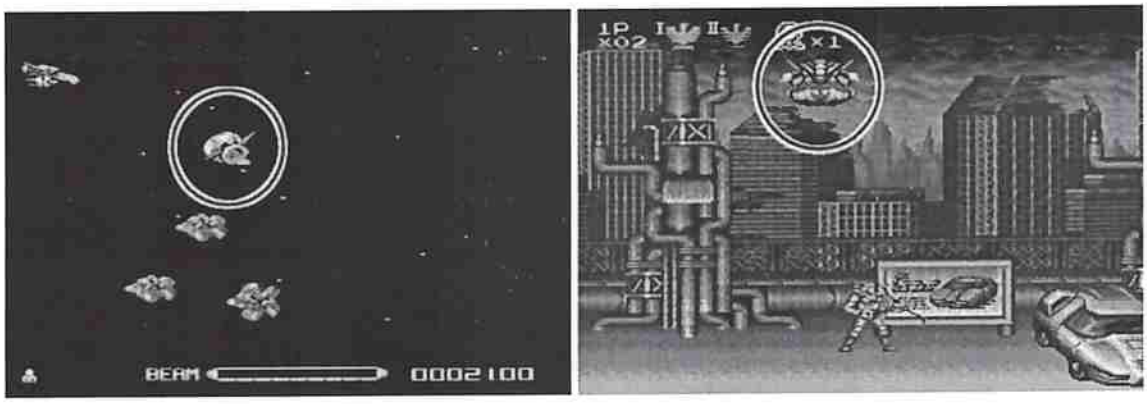

Figura 75 - R-Type III e Contra III são exemplos de jogos que empregam [Bonus Enemy].

Um [Bonus Enemy] usualmente precisa ser destruído para fornecer itens de vitalidade, melhorias para habilidades ou pontos extras, acima do fornecido por um inimigo corriqueiro. Esteticamente, este tipo de entidade aumenta o desafio do jogo ao estimular o jogador a "caçá-lo" para alcançar maiores pontuações. Dessa forma, o [Bonus Enemy] também contribui para diversificar as partidas e torná-las menos repetitivas. No Space Invaders, o [Bonus Ship] é visualmente caracterizado como um disco voador que cruza a tela e fornece pontos extras quando abatido. De forma similar, o [Bonus Enemy] empregado nos jogos R-TypeIII e Contra III também cruza a tela e fornece um item caso destruído.

```
[Bonus Ship] <bonus enemy> {
      [Points Reward] <reward>

      <behavior>:
      [Cross Screen] <behavior> {
            direction: horizontal
      }

      [One-Hit Kill] <concept>
}
```

Figura 76 - Modelagem do inimigo [Bonus Ship].

A modelagem de componentes para o inimigo [Bonus Ship] é mostrada na Figura 76. Assim como o [Invader] e o próprio [Player Ship], o [Bonus Ship] é destruído com um único disparo ([One-Hit Kill]). Seu comportamento, cruzar a tela de um lado ao outro ([Cross Screen]), é frequentemente encontrado em jogos 2D, especialmente naqueles pertencentes aos gêneros [Platformer] e [Shooter]. A Figura 77 expõe alguns exemplos de título 2D que utilizam o componente [Cross Screen] como parte do comportamento de inimigos. Nela, os inimigos zumbi e morcego de Castlevania (fotos superiores) cruzam a tela e desaparecem. Em Ninja Gaiden (fotos inferiores), ambos inimigos mostrados cruzam a tela e rebatem em algum obstáculo (buraco ou parede). Dessa forma, eles utilizam os componentes [Cross Screen] e [Rebound].

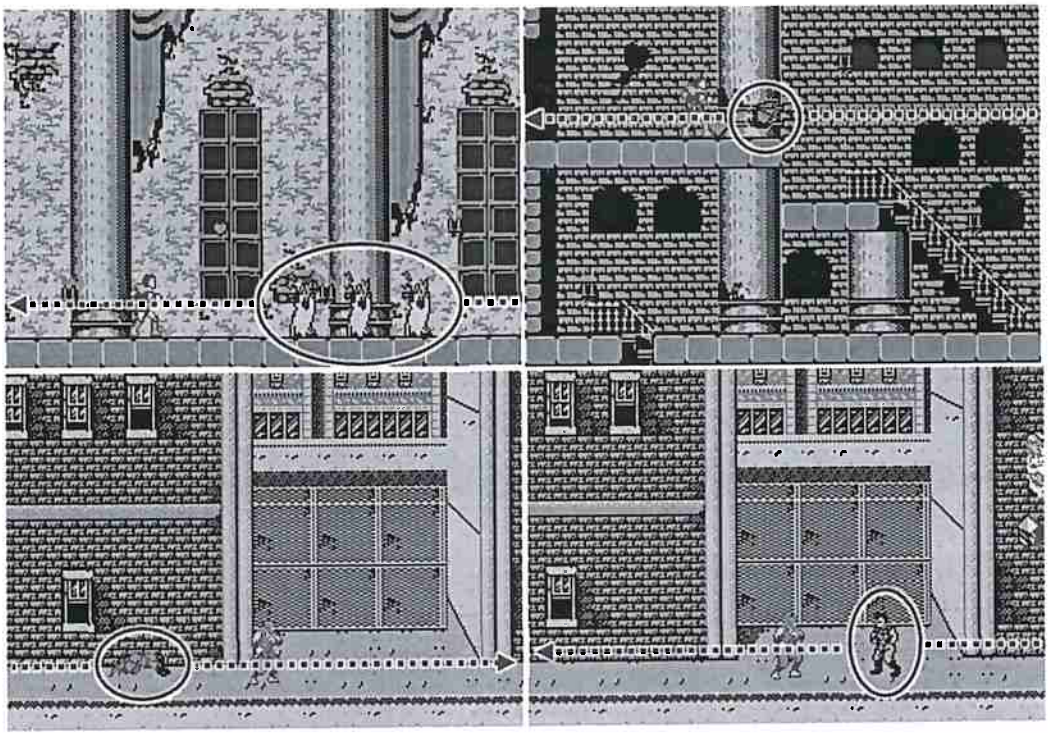

Figura 77 – Exemplos de personagens que usam [Cross Screen] como comportamento.

4.1.2 Design por Composição

A segunda etapa do estudo de caso Space Shooter compreendeu a extração de componentes de títulos populares do mercado a fim de acrescentá-los ao design original do Space Invaders de forma a transformá-lo em um novo jogo. Além disso, componentes do jogo original foram removidos ou modificados. Os gêneros que mais trouxeram influências ao design do estudo de caso foram FPS (First-Person Shooter[141]) e TPS (Third-Person Shooter[142]). Ambos são subgêneros de Shooter e, dessa forma, descendentes de Space Invaders, uma vez que este é tido como precursor do primeiro. Esses subgêneros foram escolhidos por sua grande popularidade no mercado contemporâneo (ESA, 2016) e para permitir uma mistura de componentes contemporâneos com os de jogos antigos.

Trazendo o novo ao velho

Os jogos dos gêneros FPS e TPS popularizaram a utilização conjunta de teclado e mouse[143]. Essa forma de controle tornou-se um padrão e pode ser facilmente observada em diversos títulos do mercado, como em franquias de jogos FPS – Halo, Call of Duty, Battlefield e Killzone –, bem como, em jogos TPS ou que incluem tiroteio em terceira pessoa – Gears of War, Max Payne, Uncharted e Resident Evil (5 e 6). Nesses jogos, o teclado move o personagem adiante ou o retrocede ([Move Forward and Backward]) e permite este que seja movido lateralmente ([Move Sideways], também conhecido por [Strafe]). O mouse, por outro lado,

[141] http://tvtropes.org/pmwiki/pmwiki.php/Main/FirstPersonShooter
[142] http://tvtropes.org/pmwiki/pmwiki.php/Main/ThirdPersonShooter
[143] https://en.wikipedia.org/wiki/Free_look

controla a direção da câmera ([Mouse Look]), além de ser usado para mirar e disparar ([Mouse Aim]). O componente de um personagem de jogos de FPS está modelado na Figura 78.

O uso conjunto de teclado e mouse apresenta-se como uma forma bastante peculiar de controles, permitindo ao jogador mover o personagem e definir a direção da câmera simultaneamente. Além disso, nas situações de combate é possível desviar de ataques ao mesmo tempo em que se mira e dispara. De um modo geral, a combinação de mouse e teclado confere maior liberar ao jogador para conhecer o ambiente e, por essa razão, é popularmente usada nos jogos de terceira pessoa, mesmo nos títulos que não envolvam tiroteio, tais como os do gênero Adventure. Cabe ressaltar que as mesmas funcionalidades são simuladas em controladores de videogames (*gamepads*), que contemporaneamente possuem duas alavancas analógicas, uma para a função do teclado e outra para a do o mouse.

```
[FPS Player] <player> {

      // jogador move personagem adiante ou retrocede
      // tambem pode mover-se lateralmente
      [Move and Strafe] <movement> {
            [Strafe] <movement>
            [Move Forward and Backward] <movement>
      }
      // jogador usa mouse para direcionar a visão/câmera
      [Mouse Look] <control>
      [Mouse Aim] <control>
}
```

Figura 78 - Componente de personagem básico de jogos de primeira pessoa.

Levando-se em conta que o Space Invaders é tido como precursor dos jogos de tiro e o estudo de caso Space Shooter é baseado neste, entendeu-se que era um experimento interessante trazer o esquema de controles de Shooters modernos, tais como Firt-Person Shooters e Third-Person Shooters, para seu originador. Dessa forma, o componente [Mouse Aim] foi adicionado à constituição do [Player Ship]. Com ele, o jogador passa a controlar uma mira e disparar projéteis com o mouse. Os movimentos da nave mantiveram-se restritos a [Move Sideways] e, dessa forma, o componente [Move Forward and Backward] foi descartado, uma vez que a localização de inimigos e jogador do Space Invaders foram mantidos: inimigos no centro e jogador permanece na base da tela. Ao adicionar os componentes mencionados, o componente do personagem do jogador é modificado para o mostrado na Figura 79. Segundo a modelagem apresentada, o jogador utiliza o mouse para mover uma mira dentro de espaço da tela e o botão esquerdo é utilizado para disparar. Para mover a nave lateralmente, o jogador utiliza as setas do teclado. O resultado no protótipo é apontado na Figura 80.

```
[Player Ship] <shooter character> {

      [Move Sideways and Mouse Aim] <behavior> {
            [Move Sideways] <movement> {
                  acceleration: no
                  action by: button hold

                  [Limited to Screen] <modifier>
            }
```

```
            <attack>:
            [Normal Shot] <straight shot>
                  [Straight Shoot] <shoot> {
                        direction: [Mouse Sight]
                        damages: [Enemy]
                        rate: [Rapid Fire]
                        action by: left mouse button
                  }
                  [Mouse Aim] <aiming> {
                        [Mouse Sight] <marker>
                  }
                  [Rapid Fire] <modifier>
                  [Unlimited Ammo] <modifier>
                  [Unprecise Shot] / [Shot Error] <modifier>
            }

            <attack>:
            [Precise Shot] <straight shot> {
                  [Straight Shoot] <shoot> {
                        direction: [Mouse Sight]
                        damages: [Enemy]
                        rate: [Rapid Fire]
                        action by: right mouse button
                  }
                  [Mouse Aim] <aiming> {
                        [Mouse Sight] <marker>
                  }
                  [Weapon Scope] <modifier> {
                        [Restricted Visibiliy] <modifier>
                        [Aim Down Sights] <action>
                  }
                  [Rapid Fire] <modifier>
                  [Unlimited Ammo] <modifier>
            }
      }

      [Health] <status> {
            [Regenerating Health] <modifier> {
                  amount: full health
                  action by: few seconds without getting hit
            }
            [Health Meter] <hud>
      }
}
```

Figura 79 – [Player Ship]: personagem do jogador no Space Shooter.

Jogos de tiro contemporâneos tipicamente permitem que o jogador mire a arma para obter maior precisão nos disparos. Durante esta ação, a arma é posicionada sobre a tela e o jogador passa a usar a mira física da mesma para apontá-la. Esta ação tipicamente traz consequências ao gameplay: a precisão dos disparos aumenta consideravelmente ao custo de uma redução pronunciada da velocidade de movimento do personagem e direcionamento da mira, bem como, da visibilidade da cena do jogo. Essas características são ainda mais acentuadas quando o jogador utiliza uma arma do tipo franco-atirador. Esteticamente, o ato de mirar traz dois pesos opostos ao jogador: há maior confiança em confrontos de médio a longo alcance, mas as restrições de mobilidade e visibilidade aumentam sua tensão ao perder a percepção geral

da cena. Comumente nomeado de [Aim Down Sights][144], este aspecto foi identificado e incluído no Space Shooter, como constituinte de [Weapon Scope][145]. Utilizando o botão esquerdo do mouse, o jogador dispara tiros normais ([Normal Shot]), que possuem baixa cadência baixa e uma taxa de erro ([Shot Error]). Esta desvia levemente os tiros da mira, simulando os disparos de uma metralhadora. Por outro lado, ao pressionar o botão direito do mouse o jogador usa a mira de precisão e dispara com alta cadência de tiros ([Precise Shot]), ao custo de perder a visibilidade de parte da tela.

O último componente empregado no design do protótipo, no âmbito de aproximá-lo a jogos de tiro 3D, foi a vitalidade regenerativa ([Regenerating Health]). Ela representa uma característica muito popular entre jogos de tiro contemporâneos e difundida pela série Halo[146]. A [Regenerating Health] permite que a vitalidade do personagem do jogador seja recarregada automaticamente caso fique alguns segundos sem sofrer danos. Esteticamente, este aspecto permite que o jogador realize ações mais ousadas nos confrontos do jogo e remove a procupação de buscar itens de restauração da vitalidade. Ambos componentes de shooters 3D, [Aim Down Sights] e [Regenerating Health], estão descritos no corpo do componente [Player Ship] (Figura 79). Os demais componentes são discutidos na próxima seção.

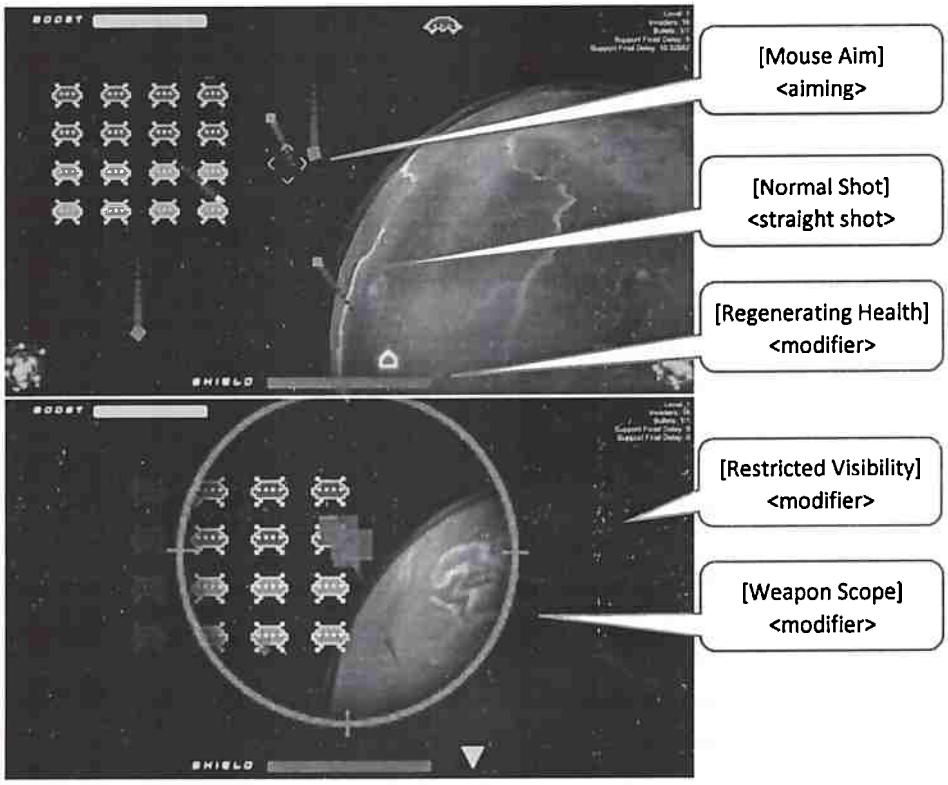

Figura 80 - Componentes contemporâneos empregados no protótipo.

[144] http://www.giantbomb.com/iron-sights/3015-567/
[145] http://tvtropes.org/pmwiki/pmwiki.php/Main/ScopeSnipe
[146] http://tvtropes.org/pmwiki/pmwiki.php/Main/RegeneratingHealth

Aumentando o ritmo do jogo

O número de inimigos e a quantidade de seus disparos foi incrementada significativamente em relação à formula original do jogo Space Invaders. Essas modificações tiveram como propósito aumentar o ritmo do jogo [Fast-Paced] e adicionar o aspecto [Bullet Hell], criando situações de tensão nas quais o jogador precisa desviar de uma grande quantidade de projéteis inimigos, ou mesmo, decidir pela situação de danos mais favorável: se não houver como desviar dos tiros, deve escolher a posição que leve ao menor número de colisões com projéteis. Jogos de tiro [Fast-Paced] tipicamente empreguem o componente [Rapid Fire][147], que representa a habilidade de disparar tiros consecutivamente ao manter o botão pressionado. Em títulos antigos, o [Rapid Fire] é comumente representado como uma habilidade temporária ([Power-Up]). Por outro lado, jogos de tiro contemporâneos permitem que o jogador utilize armas de disparo manual (pressionar para disparar) ou automáticas ([Rapid Fire]). Desta forma, em vias de adotar o estilo contemporâneo, o [Rapid Fire] foi empregado no Space Shooter de forma permanente. Somado ao componente [Unlimited Ammo], o [Rapid Fire] ajuda a criar uma atmosfera de ritmo rápido, contribuindo com o [Fast Paced].

Modificando o mundo do jogo

O mundo do jogo pouco mudou em relação ao original. Nesse sentido, o veículo do jogador mantém-se posicionado na base da tela e os inimigos acima. O cenário continua restrito a uma tela ([One-Screen World]), mas os adversários podem se mover além dos limites da mesma, reaparecendo no lado oposto ([Wrap Screen]). As coberturas destrutíveis foram removidas (~~[Destructible Cover]~~) para intensificar o aspecto [Bullet Hell], forçando o jogador a buscar constantemente por pontos de fuga dos dispartos inimigos, em oposição a simplesmente manter-se protegido sob um escudo. Conteudo, considerando que a nave do jogador não pode ser movida adiante ou retroceder (não emprega [Move Forward and Backward]), dois [Warp] foram posicionados nas laterais da tela a fim de evitar situações que deixem o jogador encurralado. Quando a nave do jogador entra em um [Warp] ela é instantaneamente transportada para lado oposto da tela, na posição do outro [Warp]. Esta solução é similar ao emprego do componente [Wrap Screen], que remove a restrição de movimentação dentro dos limites da tela ([Limited to Screen]). No entanto, os [Warps] garantem que a nave estará sempre ao alcande de um disparo inimigo, impedindo que o primeiro possa se refugiar nos cantos da tela. Por fim, a apresentação do jogo presa pela simplicidade, com desenhos pixelados e sons eletrônicos, em alusão ao título original ([Retro Style]). A modelagem geral do estudo de caso está disposta na Figura 81, assim como feito para o Space Invaders na Figura 69.

A estrutura de progressão no Space Shooter mantém-se em levels ([Game Level Progression]) e para alcançá-la, o objetivo principal continua a ser destruir todos os oponentes que se encontram na tela ([Destroy Enemies]). A cada novo level, uma configuração diferente de invasores é apresentada, caracterizando o lançamento de ondas crescentes de inimigos ([Enemy Waves]). O jogador não possui um número restrito de tentativas, como no jogo original

[147] http://www.giantbomb.com/rapid-fire/3015-2374/

([~~Lives~~]). Ao invés, ele pode repetir o mesmo level quantas vezes desejar ([No Lives]). Também é possível que o jogador inicie uma nova partida (recomeçando do primeiro level) ou continue do último level alcançado ([New Game or Continue]). Por fim, a estrutura geral do jogo passou de jogo infinito ([~~Endless Game~~]) para completável ([Compleatable Game]), contendo doze *levels* e um inimigo maior ao final, chamado popularmente de "chefe" ([Boss][148]).

```
[Space Shooter] <shoot'em up> <~~endless game~~> <game>
{
      [One-Screen World] <game world>
      [Top-View] <pov>
      [Warp] <transporter>
      [Retro Style] <presentation>
      ~~[Destructible Cover] <cover>~~

      <game progression>:
      >> jogador percebe a progressão ao passar para o próximo level
      [Game Level Progression] <progression>
      ~~[Endless Game] <game>~~
      [Compleatable Game] <game>
      [New Game or Continue] <game session> {
            from: start of level
      }
      [No Lives] <status>
      ~~[Lives] <status>~~

      >> power-up
      [Power-Up Meter] <hud> {
            when full: activate [Hyper Mode]
      }

      [Hyper Mode] {
            [Temporary Power-Up] <power-up>
            [Invincibility] <ability>
            [Increase Fire Rate] <ability>
            [Bullet Time] <modifier>
      }

      [Shoot'em Up] <shooter> <genre>
      {
            [Unlimited Ammo] <modifier>
            [Bullet Hell] <concept>
            [Fast-Paced] <concept>

            >> componente genérico, substituído poe específico
            ~~[Power-Up] <item>~~
            [Pill Power-Up] <power-up> {
                  fills: [Power-Up Meter] <hud>
            }

            <player>:
            >> componente genérico, substituído por específico
            ~~[Shooter Character] <player>~~
            [Player Ship] <shooter character>

            [Enemy] <entity>
      }
```

[148] http://tvtropes.org/pmwiki/pmwiki.php/Main/BossBattle

```
      <objectives>:
      >> jogador deve destruir todos os inimigos para progredir
      [Destroy Enemies] <primary objective> {
            achieve: [Game Level Progression]
      }
      >> jogador deve impedir que inimigos avancem à base da tela
      [Prevent Enemy Advance] <primary objective> {
            prevent: reach screen base
      }

      <enemies>:
      [Enemy Waves] <enemies structure>
      [Boss] <enemy>
      [Distinguishable Enemy Types] <modifier>
}
```

Figura 81 - Modelagem geral do Space Shooter.

Especializando o gênero Shoot'em Up

Assim como o Space Invaders, o protótipo [Space Shooter] é classificado como um [Shoot'em Up]. Por esta razão, o segundo componente está explicitado no corpo do primeiro. No design apresentado acima, o componente genérico [Shooter Character] foi substituído por uma especialização nomeada de [Player Ship] e, dessa forma, está classificada como <shooter character>. Conforme previamente discutido, este componente contém a modelagem dos aspectos referentes ao personagem controlado pelo jogador no Space Shooter. Outra especialização significativa realizada no design do estudo de caso se refere ao [Power-Up] originalmente presente no gênero [Shoot'em Up], que foi substituído pelo componente [Pill Power-Up]. Inspirado no jogo Pac-Man, o [Pill Power-Up] pode ser coletado pelo jogador para preencher a barra [Power-Up Meter] que, ao ser completa, aciona o [Hyper Mode]. Neste modo temporário ([Temporary Power-Up]), o jogador torna-se invencível ([Invincibility]), dispara em ritmo acelerado ([Increase Fire Rate]) e a velocidade de ações e ataques inimigos diminui drasticamente, sem afetar a nave do jogador ([Bullet Time]). Esteticamente, o [Hyper Mode] provê um momento de compensação ao [Bullet Hell]. Assim como no Pac-Man, o modo torna o jogador apto a infligir danos consideráveis a uma grande quantidade de inimigos, ao mesmo tempo em que fica invulnerável.

Novos Inimigos

Na concepção de um jogo, os inimigos e o design dos *levels* ajudam a diversificar as situações e os desafios vivenciados pelo jogador. Nesse sentido, diferentes combinações de inimigos são tipicamente utilizadas para criar novas experiências, apresentadas de forma gradativa ao jogador durante sua progressão no jogo. Dessa forma, com o objetivo de prover uma experiência de *gameplay* fiel ao Space Invaders original, porém acrescida de maior diversidade, foram projetados e implementados diversos tipos de invasores.

As últimas modificações realizadas sobre o conceito original do Space Invaders referem-se aos inimigos. Ao invés de apenas um comportamento, previamente documento no componente [Invader] (Figura 73), o Space Shooter implementa nove tipos diferentes de inimigos invasores. Cada um dos doze *levels* progressivos do jogo lança uma combinação distinta

desses inimigos para que o jogador os destrua e progrida (Figura 82). Além dos invasores, o jogo também lança nove variações de inimigos de suporte, que constituem versões mais elaboradas do [Bonus Ship] (Figura 76) do Space Invaders. Após concluídos os doze *levels* do jogo, o jogador deve enfrentar o [Boss], um inimigo maior e mais complexo. Ao derrota-lo, o jogador conclui o jogo. Além de enriquecer a experiência de uso do protótipo, a diversidade de inimigos permite demonstrar diferentes situações de modelagem de design com a abordagem de componentes.

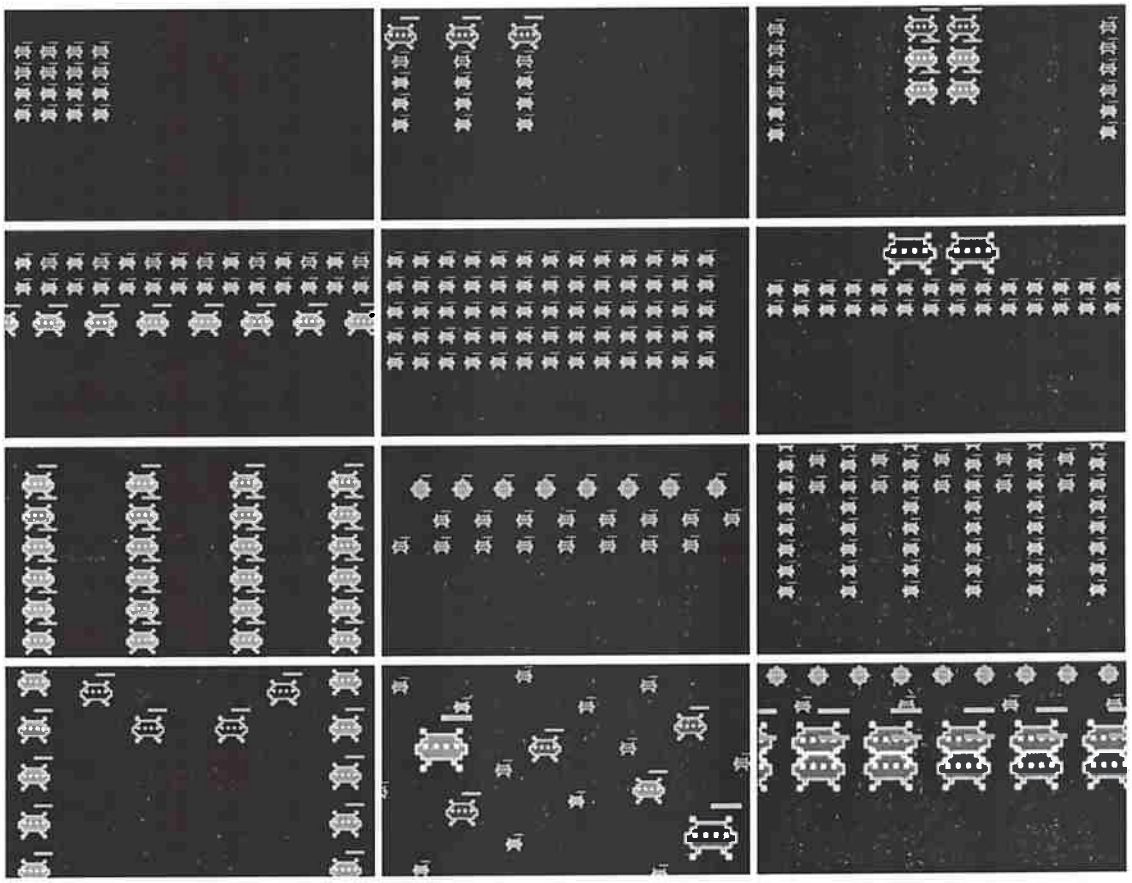

Figura 82 - Combinações de Inimigos Invasores nos doze levels.

Um aspecto característico de jogos 2D empregado no Space Shooter refere-se ao denotado pelo componente [Distinguishable Enemy Types] <modifier>. Ele indica que os tipos dos inimigos podem ser associados à sua aparência e, dessa forma, são reconhecidos pelo jogador antes mesmo de iniciar seu comportamento. Jogos de guerra que buscam fidelidade ao realismo das batalhas com armas de fogo, como os das séries Medal of Honor, Call od Duty, Battlefield e Brothers in Arms, comumente apresenta poucos tipos de soldados inimigos. Estes são reusados com bastante frequência no decorrer dos *levels* do jogo, apenas com uma mudança em sua aparência. Dessa forma, não é possível associar o comportamento à aparência do NPC.

Os inimigos concebidos para o Space Shooter podem ser organizados em dois grupos: invasores e suporte. Seis tipos de invasores representam variações do [Invader] original sendo dispostos em forma de grade (Invasores Clássicos). Além destes, dois movem-se progressivamente na direção do personagem do jogador (Perseguidores) e um ricocheteia nos

limites da tela (Ricochete). Quanto aos inimigos de suporte, há nove tipos, organizados da seguinte forma: três representam variações do [Bonus Ship] original (Figura 76), cruzando a tela de um lado ao outro; seis são agrupados e movem-se por caminhos pré-determinados em forma de "tira" e, por fim, um cruza a tela verticalmente e não pode ser destruído, representando um obstáculo a ser desviado pelo jogador. A modelagem de todos os inimigos citados é discutida no texto que segue.

Inimigos Invasores

O principal tipo de invasores projetado para o Space Shooter é nomeado de Invasor Clássico. Ele teve como inspiração o padrão de comportamento dos invasores originais do jogo Space Invaders. Dessa forma, o componente [Grid Invader] foi naturalmente modelado como uma especialização do [Invader] original e define a base para seis variações de Invasores Clássicos, nomeados de Classic, Sniper, Spread, Blocker, SpinShot e Stealth. A representação dessas entidades na linguagem de componentes está expota na Figura 83. Nela, o [Grid Invader] representa o tipo básico para os seis tipos de invasores criados para o Space Shooter. Dessa forma, os aspectos comuns a esses invasores estão modelados no componente. Os invasores, por sua vez, especializam o [Grid Invader] para definir comportamentos específicos e, portanto, o utilizam como componente classificador. Para facilitar a compreensão das modificações que levaram ao design do [Grid Invader] o esquema da Figura 83 utiliza destaques por cores. A composição implícita proveniente da classificação pelo [Invader] é indica em azul. Por outro lado, os componentes adicionados estão destacados em verde, como por exemplo, o aspecto de vitalidade ([Health]) e sua barra mostradora, que some após um tempo ([Auto Hide Health Bar]).

```
>> Componente básico para os seis Grid Invaders presentes no jogo
[Grid Invader] <invader> {

      [Invader] <enemy> {
            [Grid Formation] <enemy formation> {
                  [Syncronized Swarm] <modifier>
            }
            ~~[Points] <reward>~~

            <behavior>:
            [Move and Rebound Towards Player] <behavior> {
                  [Move Sideways and Rebound] <behavior> {
                        [Move Sideways] <movement>
                        [Position Rebound] <modifier> {
                              side: horizontally
                        }
                  }
                  [Move Straight] <movement> {
                        direction: down
                  }
                  ~~[Increase Speed] <modifier>~~
                  ~~[Limited to Screen] <modifier>~~
                  [Wrap Screen] <modifier>
            }
```

```
            <attack>:
            [Straight Shoot] <shoot>
            [Collision Damage] <attack>

            [Increase Fire Rate] <modifier> {
                  by: level
            }

            <status>:
            [One-Hit Kill] <concept>
            [Health] <status>
            [Auto Hide Health Bar] <hud>
      }
}

>> foram criados seis tipos especializados de [Grid Invader]
[Classic Invader] <grid invader> {
      [Straight Shot] <shot>
}

[Sniper Invader] <grid invader> {
      [Shoot on Target] <shot> >> ver [Piranha Plant]
}

[Spread Invader] <grid invader> {
      [Spread Shot] <shot>
}

[Blocker Invader] <grid invader> {
      [No Attack] <negative>
}

[SpinShot Invader] <grid invader> {
      [Spin Shot] <shot>
}

[Stealth Invader] <grid invader> {
      [Hide and Show] <behavior> >> ver [Piranha Plant]
      [Straight Shot] <shot>
}
```

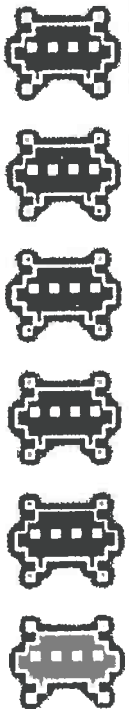

Figura 83 - Modelagem dos Invasores Clássicos do Space Shooter.

O segundo tipo de Inimigos Invasores refere-se aos Perseguidores (Figura 84). Diferentes dos invasores originais do Space Invaders, o [Follow Invader] e o [Kamikaze Invader] não disparam projéteis contra a nave do jogador. Seu propósito é persegui-lo na tentativa de danificá-los com colisões. Pata fins de demonstração, a modelagem dos perseguidores seguiu uma estratégia distinta da empregada nos Invasores Clássicos: ao invés de se utilizar um componente comum e especializá-lo para os subtipos, um dos perseguidores reusa os aspectos do outro. Nesse sentido, o [Follow Invader] define por completo o primeiro tipo de perseguidor, que é então usado como classificador para o segundo, o [Kamikaze Invader]. Este, por sua vez, reutiliza os aspectos do primeiro, apenas aplicando algumas alterações quanto ao comportamento de perseguição do jogador ([Follow Target]).

```
>> Primeiro tipo de Invasor Perseguidor: segue o jogador em intervalos
e apresenta erros aleatórios de direção
 [Follow Invader] <enemy> {
      <behavior>:
      [Follow Target] <behavior> {
            [Pause at Interval] <modifier>
            [Random Direction Error] <modifier>
      }
      [Wrap Screen] <modifier>

      <attack>:
      [Collision Damage] <attack>

      <status>:
      [Health] <status>
      [Auto Hide Health Bar] <hud>
}

>> Kamikaze Invader é similar ao Follow Invader. Contudo, ele segue o
jogador lentamente, mas sem intervalos e erros de direção
 [Kamikaze Invader] <follow invader> {
      [Follow Invader] <enemy> {
            <behavior>:
            [Follow Target] <behavior> {
                  [Pause at Interval] <modifier>
                  [Random Direction Error] <modifier>
                  [Moves Slowly] <modifier>
            }
      }
}
```

Figura 84 - Modelagem dos Inimigos Perseguidores.

O último tipo de Inimigos Invasores foi chamado de Ricochete ([Rotor Invader]). Modelado na Figura 85, ele ricocheteia nas laterais da tela e, assim como os Perseguidores, danifica a nave do jogador por colisões.

```
>> Inimigo Ricochete
[Rotor Invader] <enemy> {
      <behavior>:
      >> ver protótipo [Bouncing Ball]
      [Move and Rebound] <behavior> {
            [Move Straight] <behavior> {
                  direction: diagonally
            }
            [Screen Rebounds] <modifier> {
                  side: both (horizontally and vertically)
            }
      }

      <attack>:
      [Collision Damage] <attack>

      <status>:
      [Health] <status>
      [Auto Hide Health Bar] <hud>
}
```

Figura 85 - Modelagem do Inimigo Ricochete.

Inimigos de Suporte

O segundo grupo de inimigos projetados para o Space Shooters corresponde aos inimigos de suporte. Tomando como base o [Bonus Ship] original do Space Invaders, eles têm por função auxiliar os invasores no ataque ao jogador e prover um meio ao jogador de obter os itens [Pill Power-Up], que devem ser coletadas para acionar o [Hyper Mode]. Desse modo, os inimigos de suporte ajudam a enriquecer e diversificar a experiência de *gameplay*, uma vez que, esteticamente, tornam-se pontos de interesse ao jogador, que passa a tê-los como um objetivo secundário no jogo. Três tipos foram definidos, totalizando nove inimigos diferentes de suporte: [Flying Disk], [Strip Support] e [Asteroid]. Cabe ressaltar que o [Asteroid] representa um obstáculo a ser desviado, uma vez que não pode ser destruído ([Indestructible]). As características comuns aos [Flying Disk] e [Strip Support] foram modeladas no componente [Support Ship] (), que é posteriormente empregado como componente classificador para os mesmos. Conforme mostrado na Figura 86, as naves de suporte são lançadas com frequência constante por um [Loop Spawner] e, assim como os invasores, possuem vitalidade ([Health]) e uma barra mostradora ([Auto Hide Health Bar]). Ao serem destruídos, liberam uma [Pill Power-Up]. O componente também define um *template* para os inimigos que vier a classificar, estabelecendo a necessidade de definir posteriormente um comportamento (<behavior>) e um ataque (<attack>).

```
[Support Ship] <bonus enemy> {

      [Loop Spawner] <spawner>

      [Item Reward] <reward> {
            item: [Pill Power-Up] <power-up>
            when destroy: group
      }

      >> status
      [Health] <status>
      [Auto Hide Health Bar] <hud>

      >> um support ship possui comportamento
      <behavior>

      >> um support ship ataca o jogador
      <attack>
}
```

Figura 86 - Modelagem do [Support Ship], que define os aspectos comuns aos inimigos de suporte.

As três primeiras especializações do [Support Ship] representam uma variação direta do [Bonus Ship] original do Space Invader (Figura 76), cruzando a tela de um lado ao outro e disparando de formas variadas contra o jogador. Elas foram modeladas como [Flying Disk], [Flying Disk Burst] e [Flying Disk Fast] (Figura 87). As diferenças entre estes três inimigos de suporte referem-se ao tipo de disparo e a velocidade com a qual cruzam a tela.

```
>> cruza a tela e dispara
[Flying Disk] <support ship> {
      <behavior>:
      [Cross Screen] <behavior> {
            direction: horizontal
      }

      <attack>:
      [Straight Shot] <shot>
}

>> cruza a tela e dispara rajadas de tiros
[Flying Disk Burst] <support ship> {
      <behavior>:
      [Cross Screen] <behavior> {
            direction: horizontal
      }

      <attack>:
      [Burst Shot] <shot>
}

>> cruza a tela rapidamente e dispara
[Flying Disk Fast] <support ship> {
      <behavior>:
      [Cross Screen] <behavior> {
            direction: horizontal
      }
      [Lightint Fast] <modifier>

      <attack>:
      [Straight Shot] <shot>
}
```

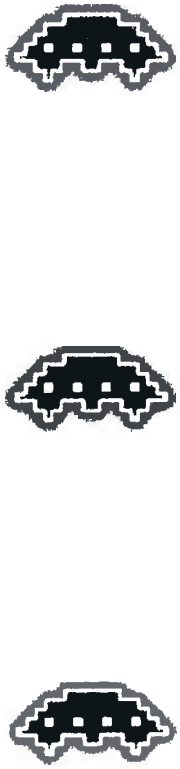

Figura 87 - Modelagem de inimigos de suporte [Flying Disk], que especializam [Support Ship].

Jogos do gênero Shoot'em Up de enredo espacial tipicamente empregam naves inimigas que se movem enfileiradamente, como uma "tira" orquestrada que realiza padrões de caminhos. Com o intuito de trazer esse aspecto ao Space Shooter, o segundo conjunto de especializações de [Support Ship] corresponde ao [Strip Support]. Todos os inimigos de suporte do tipo [Strip Support], movem-se como uma " tira" ([Strip Formation]) por um caminho pré-estabelecido ([Follow Path]) e são lançados em uma sequencia ([Strip Spawner]). Como ataque, eles disparam um tico comum contra o jogador ([Straight Shot]). Esse comportamento está modelado na Figura 88.

```
>> tira de inimigos
[Strip Support] <support ship> {
      <behavior>:
      [Strip Spawner] <spawner>
      [Strip Formation] <enemy formation>

      [Follow Path] <behavior> {
            pattern: classic/square/two loops/eights/circle/sine
      }
      <attack>:
      [Straight Shot] <shot>
}
```

Figura 88 - Modelagem dos inimigos de suporte [Strip Support], que especializam [Support Ship].

Os padrões de movimentação dos [Strip Support] foram montados com base em caminhos pré-definidos. As naves de suporte foram coloridas diferentemente, de forma a facilitar seu reconhecimento pelo jogador. Ao todo, seis padrões foram construídos, permitindo diferentes coreografias de ataque ao jogador. Elas estão apresentadas na Figura abaixo.

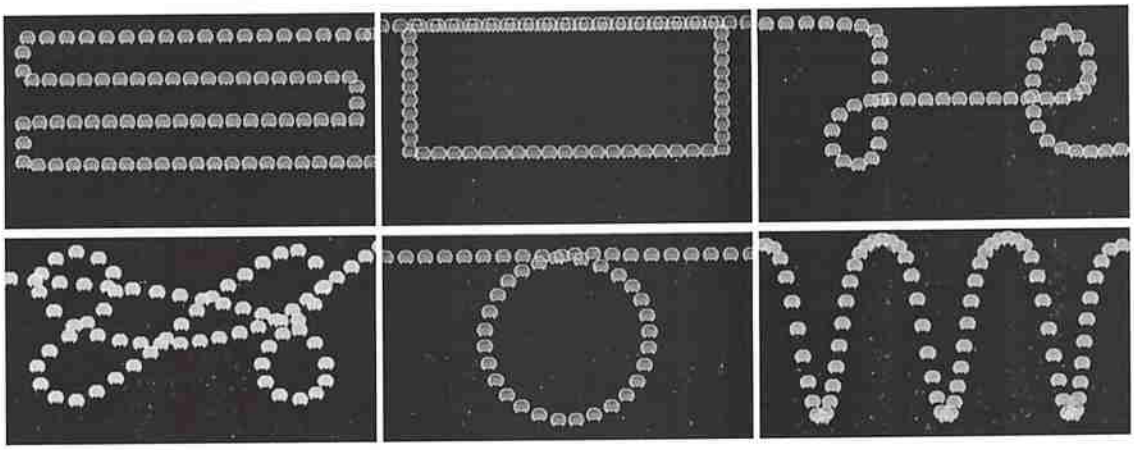

Figura 89 - Padrões de caminhos ([Follow Path]) realizados pelos inimigos [Strip Support].

O último personagem não controlado pelo jogador adicionado ao design do jogo foi representado pelo componente [Asteroid] (Figura 90). Ele é o único obstáculo adicionado ao Space Shooter e aparece nos levels finais. Seu comportamento é simples, cruzando a tela verticalmente. O [Asteroid] não pode ser destruído e representa um obstáculo a ser desviado.

```
>> obstáculo
 [Asteroid] <obstacle> {
      [Loop Spawner] <spawner>

      [Sparce Formation] <enemy formation>

      [Indestructible] <status>

      [Cross Screen] <behavior> {
            direction: vertical
      }
}
```

Figura 90 - Modelagem do obstáculo [Asteroid].

Adicionando um "Chefe" ao final

Jogos de ação tipicamente apresentam cenas de batalhas contra personagens de comportamento elaborado e vasta vitalidade. Tais personagens são popularmente nomeados de Boss[149] (Chefe). Um Boss comumente emprega ataques intensivos e repetições de padrões de comportamentos. Esteticamente, a batalha com um Boss traz ao jogador a noção de ter atingido o ponto culminante de uma etapa do jogo, mais significativa que a progressão habitual entre levels. Vencê-la significa avançar de forma significativa no jogo. Para o jogador, a complexidade da batalha com um Boss traz inerentemente a necessidade de se criar estratégias. Nesse sentido, a adoção de padrões de comportamentos no design do Boss permite que o

[149] http://tvtropes.org/pmwiki/pmwiki.php/Main/BossBattle

jogador aprenda e reconheça seus movimentos de forma a encontrar os momentos certos para evadir os ataques e infligir danos.

O chefe final do Space Shooter está modelado na Figura 91. Assim como os demais inimigos do jogo, ele possui vitalidade ([Health]) exibida em uma barra que é ocultada automaticamente ([Auto Hide Health Bar]). Ele carrega um escudo de partículas orbitais ([Orbiting Particle Shield]), que consiste de um grupo de objetos espalhados que giram ao seu redor. O [Boss] possui seis comportamentos distintos que são acionados de acordo com sua vitalidade. Para tanto, é empregado [Select Behavior by Health Status], um componente seletor (<selector>) que ajuda a estruturar os componentes internos, indicando que um deles é selecionado por algum critério. Em geral, os comportamentos são definidos por meio do reuso de componentes previamente empregados nos inimigos invasores e de suporte. Por fim, o [Boss] avisa quanto ao tipo de comportamento a ser usado mudando de cor ([Behavior Hint]), permitindo ao jogador identificá-lo. Este componente encerra a modelagem do jogo Space Shooter.

```
>> Chefe final do jogo
[Boss Invader] <boss> {
      [Orbiting Particle Shield][150] <moving shield> {
            [Indestructible] <modifier>
      }
      [Behavior Hint] <hint>
      [Health] <status>
      [Auto Hide Health Bar] <hud>

      [Select Behavior by Health Status] <selector> {
            # comportamento azul
            [Move Sideways and Spread Shoot] <behavior> {
                  [Move Sideways] <movement>
                  [Position Rebound] <modifier> {
                              side: horizontally
                  }
                  [Spread Shot] <shot> // quando azul
            }

            # comportamento verde
            [Ranged Melee Attack] <melee attack>

            # comportamento amarelo
            [Move to Collid] <behavior> {
                  [Follow Path] <behavior>
                  [Collision Damage] <attack>
            }
            # comportamento roxo
            [Move Sideways and Burst Shoot] <behavior> {
                  [Move Sideways] <movement>
                  [Position Rebound] <modifier> {
                        side: horizontally
                  }
                  [Burst Shot] <shot> // quando azul
            }

            # comportamento vermelho
```

[150] http://tvtropes.org/pmwiki/pmwiki.php/Main/OrbitingParticleShield

```
            [Spin Shot] <shot>

            # comportamento final: modo berzerk
            [Move and Rebound] <behavior> {
                  [Move Straight] <behavior> {
                        direction: diagonally
                  }
                  [Screen Rebounds] <modifier> {
                        side: both (horizontally and vertically)
                  }
            }
      }
}
```

Figura 91 - Modelagem do chefe final do jogo - [Boss].

Um novo Space Invader

O protótipo discutido nesta seção teve como propósito demonstrar uma perspectiva de aplicação da abordagem de componentes para a construção de um novo jogo sobre outro existente. Para tanto, o Space Invaders foi modelado de maneira reversa, extraindo-se o design a partir do jogo. Esse processo é chamado de Análise por Decomposição. A partir da base estabelecida nesta primeira etapa, executou-se o Design por Composição. Nesse sentido, componentes foram removidos, modificados ou adicionados, a fim de transformar o Space Invaders em um novo jogo, chamado de Space Shooter. Mais detalhadamente, os objetivos do primeiro estudo de caso, que refletem as estratégias de demonstração planejadas, podem ser relacionados como:

- Aplicar Análise por Decomposição em um título conhecido do mercado;
- Extrair componentes de jogos do mesmo gênero que possam ser compatibilizados com o design do jogo original;
- Aplicar Design por Composição, iniciando sobre o design do jogo original, modificando-o de forma a conceber um novo jogo convincente;
- Trabalhar com o escopo dentro de um mesmo gênero, explorando as possibilidades da mistura de componente clássicos e contemporâneos dos jogos de Tiro (*Shooters*);
- Tornar contemporâneo o design de um jogo antigo, adicionando aspectos comuns a jogos modernos do gênero a que pertence;
- Construir um jogo que tenha variedade de NPC's a fim de demonstrar como a linguagem de componentes pode ser usada para estruturar a descrição de entidades diferentes que mantém determinados aspectos comuns.

O design do Space Shooter apresenta um foco na diversidade de comportamentos e leiatues de inimigos. Em contrapartida, o próximo estudo de caso tem por ênfase o design dos levels, que define o ambiente que o jogador deve explorar. Dessa forma, ele representa uma perspectiva diferente de demonstração da abordagem de componentes. Ademais, diversamente a tomar por base um título específico e explorar possibilidades dentro de um

mesmo gênero, o segundo jogo emprega o Design por Composição partindo da definição genérica de um gênero e segue adicionando componentes, sem limitar-se ao mesmo.

4.2 Estudo de Caso 2: Open World 3D Platformer

O segundo estudo de caso compreende o design e implementação de um jogo de plataformas 3D de mundo aberto. O termo "mundo aberto" (Open World[151] [152]) refere-se a liberdade de explorar um amplo ambiente virtual, permitindo que o jogador acesse as áreas que desejar. Isto implica que o jogo não é dividido em levels, mas baseado em missões (*quests*) espalhadas pelo mundo. Esteticamente, o mundo aberto dá ao jogador a sensação de maior controle sobre o rumo do jogo e a composição da própria narrativa que o guia[153]. O 3D Platformer, nome dado ao segundo protótipo, tomou por base a composição do gênero [Platformer] (Figura 92), previamente discutido nesta tese.

```
[Platformer] <genre>
{
      [Jumper Character] <player> {
            [Jump] <action>
      }
      [Platform] <level construction>
      [Obstacle] <level construction>
}
```

Figura 92 - Modelagem do gênero [Platformer].

Construindo o jogo sobre um gênero base

O primeiro estudo de caso apresentou a construção gradativa do conceito de um jogo apoidano-se em modificações na composição de um título previamente existente. O segundo empregou um processo diferente, não apoiado-se em um jogo expecífico, mas direcionado por uma ideia inicial: um jogo 3D de plataformas com ênfase em exploração. Nesse sentido, o gênero [Platformer] foi utilizada como base e gradativamente incrementado para se obter a mistura de componentes que pudesse satisfazer o conceito inicial. Nesse sentido, alguns aspectos norteadores foram escolhidos para compor o protótipo com o objetivo de enfatizar a caracterização do jogo como um [Platformer], bem como, o aspecto da exploração. Esses aspectos são apresentados abaixo.

[Open World] <game world>

A apresentação de um mundo amplo e conectado, em oposição a secioná-lo em levels progressivos, favorece a exploração à medida que abre a possibilidade de movimentação a qualquer direção e o acesso a cada local do mundo do jogo. Empregado contemporaneamente em jogos de ação, o componente [Open World] foi utilizado para apresentar um contraste em relação ao primeiro protótipo, que empregou uma estrutura de mundo secionada em levels ([Game Level Progression]), que apresentavam ondas de inimigos ([Enemy Waves]). Uma visão

[151] https://en.wikipedia.org/wiki/Open_world
[152] http://tvtropes.org/pmwiki/pmwiki.php/Main/WideOpenSandbox
[153] http://www.giantbomb.com/open-world/3015-207/

panorâmica do mundo do segundo protótipo está apresentada na Figura 93.

[Collectible][154] <item>

Jogos de plataformas usualmente empregam itens coletáveis que premiam o jogador ao atingir uma determinada quantidade. O jogador sente-se motivado a procura-los e obtê-los. Dessa forma, o emprego de coletáveis estimula a exploração. As séries Super Mario e Sonic são dois exemplos notáveis de jogos de plataformas que empregam tal aspecto.

[Acquirable Ability] <ability>

O emprego de obtenção gradativa de habilidades estimula a exploração por meio de dois mecanismos simples: elas são necessárias para a progressão no jogo e, portanto, ele fará buscará obtê-las. Nesse sentido, os meios para obtê-las no 3D Platformer são os coletáveis, que estão convenientemente espalhados pelo mundo do jogo. Dessa forma, a busca pelas habilidades torna-se um objetivo secundário do jogo. Exemplos notáveis de jogos que empregam abilidades adquiríveis são os da série Metroid[155] e Legend of Zelda[156]. Cabe ressaltar que qualquer ação do jogador pode ser classificada como uma <acquirable ability>.

Ações típicas de jogos [Platformer] foram selecionadas para compor o protótipo, a fim de caracterizá-lo como um jogo do gênero e distanciá-lo de títulos de ação em mundo aberto, como os das séries Grand Theft Auto[157] e Watch Dogs[158]. Essas ações foram definidas no jogo como habilidades adquiríveis:

- [Jump][159] <action>: ação de pular, aspecto inverente aos jogos de plataformas. Utilizada para mover-se entre plataformas e navegar em levels com verticalidade.
- [Double Jump][160] <jump>: o pulo duplo permite que jogador realize um segundo salto enquanto está no ar. Permite não somente alcançar locais mais altos ou distantes, mais também, maior versatilidade de movimentação enquanto o personagem está no ar. O componente foi utilizado pela primeira vez em Dragon Buster[161] e está presente em diversos títulos 2D e 3D, incluindo Revenge of Shinobi, Castlevania, Metroid Prime[162], Ratchet & Clank[163] e Banjo & Kazooie[164].
- [Dash] <action>: movimento horizontal que permite ao jogador deslocar-se rapidamente para a direção frontal. Assim como [Double Jump], [Air Dash] e [Wall

---

[154] http://www.giantbomb.com/collectibles/3015-375/
[155] http://tvtropes.org/pmwiki/pmwiki.php/Franchise/Metroid
[156] http://tvtropes.org/pmwiki/pmwiki.php/Franchise/TheLegendOfZelda
[157] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/GrandTheftAuto
[158] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/WatchDogs
[159] http://tvtropes.org/pmwiki/pmwiki.php/Main/JumpPhysics
[160] http://tvtropes.org/pmwiki/pmwiki.php/Main/DoubleJump
[161] http://www.giantbomb.com/dragon-buster/3030-12337/
[162] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/MetroidPrime
[163] http://tvtropes.org/pmwiki/pmwiki.php/Franchise/RatchetAndClank
[164] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/BanjoKazooie

Slide], é comum encontrar o aspecto em jogos que envolvem plataformas.

- [Air Dash] <dash>: habilidade de executar um [Dash] enquanto está no ar. Combinado com [Double Jump] permite ao jogador cobrir longas distâncias facilmente.
- [Wall Slide] <action>: movimento que traz conveniência ao controle de quedas, impedindo falhas diretas por pulos malsucedidos que terminam em colisões frontas com paredes. Além disso, favorece a exploração do nível. É popularmente encontrado em jogos que empregam plataformas, notadamente nas séries Megaman X e New Super Mario Bros. Combinado ao [Duble Jump], o [Wall Slide] permite que o jogador escale construções verticais de grande altura. Este representa o uso predomintante do componente no 3D Platformer.
- [Aim and Shoot] <straight shot>: empregado tipicamente em jogos de tiro 3D de terceira e primeira pessoa. Pode ser encontrada em jogos 3D de plataformas, como os das séries Rathet & Clank e Banjo & Kazooie. O componente [Aim and Shoot] está modelado de forma similar ao [Precise Shot] do [Space Shooter].

Assim como as ações definidas para o personagem do jogador, o estilo de construção do mundo (*level design*) e presença de plataformas foram utilizados a fim de contribuir com a caracterização do jogo ao gênero [Platformer].

[Drowning Pit] <bottomless pit>

Segundo a modelagem exposta na Figura 92, jogos do gênero [Platformer] tipicamente apresentam obstáculos que o jogador deve evitar. De todas as construções que podem ser usadas como obstáculos, a preponderante no gênero é o "fosso sem fundo" ([Bottomless Pit][165]). Duas especializações usuais desse componente referem-se ao "fosso com lava" ([Lava Pit][166]) e o "fosso de afogamento" ([Drowning Pit][167]). O segundo foi o único componente de obstáculo utilizado no protótipo e ajuda a enriquecer a atmosfera do jogo.

[Platform] <level construction>

Títulos do gênero [Platformer] unanimamente possuem plataformas: superfícies que precisam ser alcançadas usualmente por meio da ação de pulo para progredir no level. Alguns tipos especializados de plataformas populares foram empregados no protótipo para enriquecer o design do level e ajudar a caracterizá-lo dentro do gênero. Eles foram previamente identificados e expostos no digrama de componentes da Figura 61. Em especial, foram empregadas duas especializações de plataformas móveis, [Linear Platform] e [Rotating Platform], e uma plataforma fraca, [Weak Returnable Platform].

[Level Construction Style] <level construction>

---

[165] http://tvtropes.org/pmwiki/pmwiki.php/Main/BottomlessPits
[166] http://tvtropes.org/pmwiki/pmwiki.php/Main/LavaPit
[167] http://tvtropes.org/pmwiki/pmwiki.php/Main/DrowningPit

Jogos de plataformas comumente demonstram forte ênfase na construção de seus levels. Esse aspecto pode ser observado em jogos populares, como os das séries Super Mario e Little big Planet, que além de apresenteram uma grande diversidade de design de levels, embutem ferramentas que permitem que os próprios jogadores construam novos levels e os compartilhem. Procurando evidenciar esse aspecto no protótipo, foram empregados estilos de construções que se assemelham à padrões comumente encontrados em jogos do gênero:

- [Climbable Tower] <climbable> <level construction>: construção vertical que deve ser escalada pelo personagem do jogador por meio de plataformas ou habilidades de escalada ([Wall Slide] + [Double Jump]). Exemplos de jogos que a empregam compreendem Metroid, Assassin's Creed e Far Cry. A Figura 93 (foto 3) ilustra uma [Climbable Tower] no protótipo.

- [Climbable Pyramid] <climbable> <level construction>: construção piramidal em que a ascenção ao topo ocorre por meio de caminhos que a contornam. O jogo Banjoo & Kazooie emprega vários elementos verticais em seus levels, incluindo construções piramidais. Na Figura 93 (foto 2) há um exemplo de emprego de pirâmide.

- [Maze] <level construction>: seção do level de um jogo apresentada na forma de labirinto, enfatizando o componente de [Exploration]. Encontrado em títulos previamente referenciados, em especial, Metroid e Banjo & Kazooie. No protótipo (Figura 93, foto 2) ele dá acesso a uma [Climbable Pyramid].

- [Retractable Bridge] <bridge> <level construction>: ponte retrátil que pode ser usada como plataforma ou caminho de acesso temporizado (Figura 93, foto 1).

- [Catwalk] <level construction>: ponte de acesso estreito. Exige que o jogador se mova com cautela para evidar a queda e o consequente reinício do percurso (Figura 93, foto 4).

- [Ramp] <level construction>: emprego em conjunto com [Dash], o componente [Ramp] (rampa) força o jogador a realizar saltos sobre obstáculos letais (ex: [Drowning Pit]). No protótipo foram empregados encadeamentos de rampas com tal propósito.

- [Platforms Sequence] <level construction>: sequencias de plataformas são, possivelmente, o estilo de construção de level mais comum em jogos do gênero. Elas tipicamente são empregadas com o propósito de criar desafios de "derrota instantânea" para o jogador. Na foto central da Figura 93 é possível notar a presença predominante de [Platforms Sequence] no protótipo.

- [Verticality] <level structure>: com o objetivo de incentivar o jogador a explorar as combinações das habilidades de [Double Jump], [Air Dash] e [Wall Slide], há verticalidade embutida em várias construções empregadas no design do mundo do jogo, em especial, nas denotadas pelos componentes [Platform], [Climbable Tower] e [Climbable Pyramid].

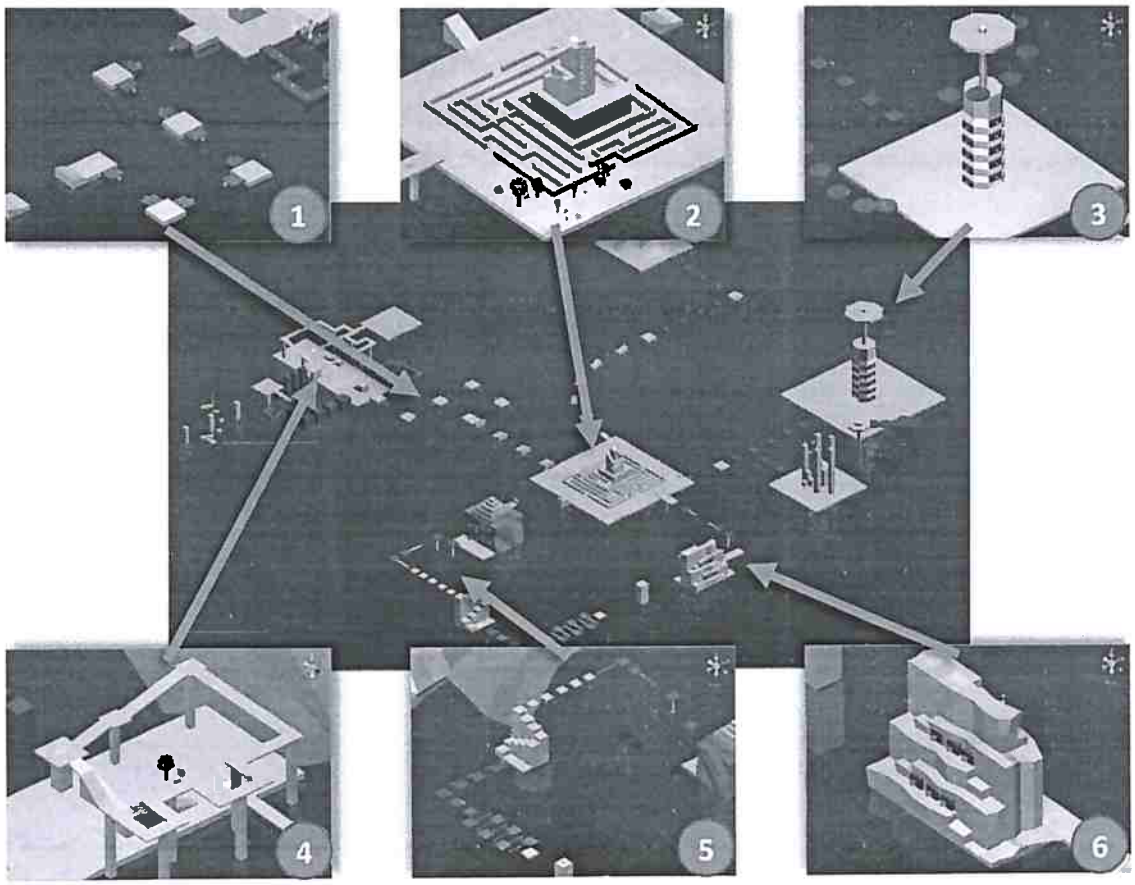

Figura 93 - Exemplos de seções do mundo do jogo que contém implementações de alguns estilos de design de levels: (1) [Retractable Bridge], (2) [Maze] com [Climbable Pyramid], (3) [Climbable Tower], (4) [Catwalk], (5) [Platforms Sequence] e (6) [Climbable Pyramid].

Os aspectos enumerados acima definiram a forma geral do protótipo. Além destes, componentes adicionais foram empregados para finalizar o design do jogo e contribuir com o *gameplay* desejado. A modelagem completa do jogo está exibida na Figura 94. Em seu formato final, é um jogo 3D de plataformas ([Platformer]) de mundo aberto ([Open World]) cercado ([World Wall]), com ênfase na exploração para coletar itens ([Collectible]) e obter habilidades ([Acquirable Ability]) a fim de completá-lo ([Compleatable Game]). A progressão no jogo ocorre pela coleta que, consequentemente, leva a obtenção das habilidades ([Mission Progression]). Ambos estão representados em mostradores fixados na interface do jogo, que também contém elementos dispostos no mundo do jogo, como placas de avisos e tutoriais ([Signs]), dicas de locais a serem descobertos ([Point Hint]) e mensagens ocultas ([Hidden Hints]), que podem ser acessadas pela visão aumentada ([Augmented Vision]). Além de elementos de interface e das construções previamente discutidas, o design do mundo emprega objetos destrutíveis ([Destructibles]), que dão acesso ao ponto final do jogo ([Final Point]). Por fim, a apresentação visual do jogo presa pela simplicidade, de estilo semelhante aos primeiros jogos 3D do mercado ([Retro Style]).

O pergonagem do jogador no 3D Platformer está modelado no componente [Platformer Player]. Assim como em outras seções do design apresentadas, ele foi concebido com o objetivo de fidelizar o jogo ao gênero alvo, especialmente as instâncias contemporâneas 3D. Nesse

sentido, o personagem é controlado de forma livre ([Move Freely]) relativamente a posição da câmera ([Camera Based Control]), tal como ocorre nos jogos Super Mario 64, Banjo & Bazooie e Super Mario 3D World. O componente indica que o personagem é movimentado para a a direção apontada pelo controle (teclado ou joystick), em oposição a esquemas de controles tipicamente empregados em jogos 3D de tiro, denotados por [Mouse Aim] + [Move Forward and Backward] + [Strafe]. Além do componente de controle e do grupo que organiza as ações associadas as habilidades adquiríveis no decorrer do jogo, o jogador pode utilizar uma visão aumentada ([Augmented Vision]). Amplamente empregada em jogos contemporâneos, ela permite visualizar elementos escondidos ou objetos através das paredes. No design do protótipo, ela está principalmente associada aos objetivos do jogo, permitndo que o jogador os visualize: [Collectible], [Acquirable Ability] e [Hidden Hint]. Exemplos de títulos que a empregam compreendem Splinter Cell, Assassin's Creed, Batman Arkham Knight, Tomb Raider, Dishonored e Far Cry.

```
[3D Platformer] <game> <platformer> {
      [Game World] <group> {
            [Open Word] <game world>
            [Exploration] <game aspect>
            [Free Camera] <câmera>
            [Compleatable Game] <game structure>
            [Start Over] <game session>
            [Collectible] <item>
            [Retro Style] <presentation>
      }
      [Game Hud] <group> {
            [Collectible Meter][168] <completion meter>
            [Ability Meter] <completion meter>
      }
      [Game Progression] <group> {
            <game progression>:
            [Mission Progression] <progression> {
                  objective: [Gain Ability]
            }

            <objectives>:
            [Collect Items] <primary objective> {
                  hud: [Collectible Meter]
                  item: [Collectable]
            }
            [Gain Hability] <secondary objective> {
                  hud: [Ability Meter]
                  ability: [Acquirable Ability]
            }
      }

      <player>:
      [Platformer Player] <group> {
            [Move Freeely] <control> {
                  [Camera Based Control] <modifier>
            }
            [Augmented Vision][169] <vision> {
                  see:  [Collectible], [Acquirable Ability],
```

[168] http://tvtropes.org/pmwiki/pmwiki.php/Main/CompletionMeter
[169] http://tvtropes.org/pmwiki/pmwiki.php/Main/XRayVision

```
                        [Hidden Hint]
                  action by: vision button
            }

            [Acquirable Ability] <group> {
                  acquired by: [Collectible]

                  [Jump] <acquirable ability>
                  [Double Jump] <acquirable ability>
                  [Dash] <acquirable ability>
                  [Air Dash] <acquirable ability>
                  [Wall Slide] <acquirable ability>
                  [Aim and Shoot] <straight shot>
                        [Straight Shoot] <shoot> {
                              direction: [Mouse Sight]
                              damages: [Destructible]
                              action by: fire button
                        }
                        [Mouse Aim] <aiming> {
                              [Mouse Sight] <marker>
                              [Aim Down Sights] <action>
                        }
                        [Unlimited Ammo] <modifier>
                  }
            }
      }
      [Level Design] <group> {
            [Verticality] <level structure>
            [Destructibles] <level constructio>
            [Final Point] <level construction>
            [Signs] <level construction>
            [Hidden Hints] <hint>
            [Point Hint] <level construction>
            [World Wall] <level construction> <restriction>

            [Level Construction Style] <group> {
                  [Climbable Tower] <level construction>
                  [Climbable Pyramid] <level construction>
                  [Retractable Bridge] <bridge> <level construction>
                  [Catwalk] <level construction>
                  [Ramp] <level construction>
                  [Maze] <level construction>
                  [Platforms Sequence] <level construction>
            }

            [Platform] <level construction>
            [Linear Platform] <moving platform>
            [Weak Returnable Platform][170] <weak platform>
            [Rotating Platform] <moving platform>

            [Obstacle] <level construction>
            [Drowning Pit] <lethal obstacle>
      }
}
```

Figura 94 - Modelagem do protótipo [3D Platformer].

Uma segunda perspectiva de demonstração

[170] http://tvtropes.org/pmwiki/pmwiki.php/Main/TemporaryPlatform

O 3D Platformer difere do Space Shooter em vários aspectos. Como jogos, são representantes de gêneros distintos e adotam estruturas opostas: o primeiro é um jogo de plataformas 3D de mundo aberto e câmera livre e o segundo, um jogo de tiro 2D de visão superior e progressão linear. A única característica explicitamente similar entre ambos é o emprego do mouse para mirar e disparar.

Como relatos de design, os dois projetos empregam rumos diferentes. O Space Shooter iniciou pela engenharia reversa de um jogo existente e seguiu com modificações trazidas por em componentes de jogos contemporâneos de tiro. O design final representa uma versão consideravelmente modificada do jogo Space Invader. O segundo protótipo seguiu por um rumo de design distinto, tendo início em um conceito geral e um gênero base. Sobre eles, foram acrescidos diversos componentes de outros títulos e gêneros de forma a satisfazer a ideia inicial do jogo. Além disso, o conteúdo referente a cada projeto trouxe variações nos tipos de componentes empregados. Como um jogo Shoot'em Up, o Space Shooter apresentou grande ênfase na diversidade de comportamentos e organizações das naves invasoras. Por conseguinte, é possível notar que o modelo que o descreve foi dirigido a representação das esquadras de naves, contendo seus comportamentos e ataques. Por outro lado, o segundo protóripo possui uma modelagem mais sintética, com ênfase nos estilos de construção de level empregados.

O principal motivo que levou a existência do segundo protótipo de desmontração foi a modelagem de um jogo completamente distinto do primeiro, variando gênero (platformer), perspectiva (câmera livre) e estrutura (3D). Nesse contexto, as estratégias de demonstração empregadas que diferem do primeiro estudo de caso podem ser sintetizadas como:

- Representar os aspectos de um personagem controlável pelo jogador que se assemelhe aos encontrados em jogos 3D contemporâneos 3D;
- Apresentar a identificação e emprego de componentes oriundos do gênero platformer.
- Realizar o design por componentes de um jogo fortemente baseado em construções de level;
- Aplicar o Design por Composição de forma livre, iniciando sobre um conceito geral e agregando componentes de forma a satisfazê-lo e, consequentemente, construir um novo jogo convincente;

Por fim, há alguns pontos peculiares referentes a estratégia de modelagem adotada. O primeiro deles refere-se a forma como os componentes estão organizados. Assim como o componente estruturador <selector> presente no [Boss Invader] (Figura 91), o segundo protótipo emprega agrupadores (<group>) para dispor os aspectos na modelagem do jogo. O segundo ponto refere-se a forma como o gênero [Platformer] foi disposto no [3D Platformer]. Em oposição a adicioná-lo de forma íntegra ao jogo, como feito com o [Shoot'em Up] no [Space Shooter], o [Platformer] foi desmontado e espalhado na composição do jogo. Por fim, o foco existente na documentação do design do mundo do jogo apresenta uma restrição da abordagem, que não é adequada a modelagem exata de formas, narrativa e sequencias de

eventos no jogo. Para tanto, o uso de recursos auxiliares, como o próprio Documento de Design, mostra-se adequado. Dessa forma, a definição concreta dos levels pode ser baseada em esquemas, plantas ou mapas visuais.

4.3 Considerações sobre Componentes e Design

O presente capítulo teve por objetivo demonstrar a aplicação da abordagem de componentes para o design de dois protótipos convincentes. O termo "convincente" refere-se ao fato dos protótipos terem sido projetados e construídos para contemplar as mesmas características de jogos reais. Pela mesma razão, eles foram enquadrados em dois gêneros de grande popularidade no mercado de consoles de videogames: atiradores (*shooter*) e ação[171] (*action*) (ESA, 2016). Além disso, aspectos tipicamente empregados em jogos contemporâneos foram incluídos nos protótipos, notadamente o controle de mira e disparos com o mouse, a câmera 3D flutuante de controle livre e a vitalidade recarregável. No contexto da abordagem de componentes de que trata este trabalho, o processo de demonstração teve dois propósitos fundamentais:

- Mostrar que é possível descrever jogos e gêneros, sejam eles existentes ou novos projetos, pelo agrupamento de componentes. Para tanto, há duas questões relevantes:
    - É possível aplicar engenharia reversa em jogos existentes e decompô-los em suas partes formadoras?
    - É possível agrupar componentes para formar o design de novos jogos?
- Apresentar narrativas de concepções de jogos utilizando a abordagem de componentes, partindo da ideia inicial, prosseguindo ao design e finalizando com a implementação e experimentação estética.

Considerando os propósitos apontados, os dois estudos de caso discutidos neste capítulo expõem relatos completos de construção de jogos convincentes (similares a produtos reais). Eles apresentam uma narrativa de design por componentes, partindo de uma ideia inicial, seguindo para a discussão dos componentes e alcançando o design final dos protótipos. Por fim, a implementação dos protótipos permite que o leitor realize experimentações e perceba a estética resultante da combinação de componentes utilizados.

Sob a perspectiva do design, os estudos de caso apresentam diferentes soluções de concepção de jogos apoiadas sobre a modelagem por componentes. Na primeira, tomou-se por base a constituição de um jogo específico, que foi contemporanizado pelo emprego de componentes de jogos de tiro 3D modernos. Além disso, o jogo teve uma forte ênfase na variedade de comportamentos de inimigos e seus ataques. Em contrapartida, o segundo protótipo teve uma ênfase no design de seus levels (suas construções) e em exploração, partindo de um gênero base e uma ideia inicial, e construído com componentes comumente

[171] Platformer é subgênero de Ação. http://tvtropes.org/pmwiki/pmwiki.php/Main/VideogameGenres

encontrados em títulos do mercado.

Os estudos de caso discutidos em detalhes neste capítulo apresentaram a modelagem completa de jogos por meio da Linguagem de Componentes. Assim, voltando ao contexto dos dois propósitos enumerados, os protótipos claramente satisfazem o primeiro. Como exemplo, o estudo de caso Space Shooter apresenta a modelagem reversa completa de um jogo existente intitulado Space Invader. Nela, os aspectos identificados no jogo foram mapeados para componentes e organizados com base nas três construções da abordagem: classificação, associação e composição. Além disso, ambos protótipos apresentam a modelagem de dois novos jogos concebidos sobre misturas distintas de componentes. De uma forma geral, foi possível identificar os aspectos como componentes, compreender suas relações e impactos no design e misturá-los para criar novos títulos inéditos.

A demonstração da abordagem não só mostra diferentes formas de aplicação da linguagem de componentes para a modelagem de design, mas também, uma perspectiva de design realmente distinta daquelas apresentadas nos trabalhos discutidos no Capítulo 2. Nesse sentido, a demonstração não discute e teoriza conceitos relacionados aos jogos, bem como, não tem a tenta representá-los sobre linguagens de diagramas existentes, como a UML ou Redes de Petri. Ela, de fato, expõe os jogos como a junção de partes menores (componentes), seja ao revertê-los de produtos finais ao design ou ao conceber novos. Essas partes menores são estruturadas, desconectáveis, apresentam significado próprio e podem ser usadas para se montar novos jogos.

Acredita-se que existe um vasto conhecimento de design embutido nos jogos já produzidos, fruto direto da experiência dos designers que os projetara. Parte desse conhecimento pode ser resgatado por meio do mapeamento de aspectos para componentes. Dessa forma, a abordagem de componentes auxilia na concepção de um vocabulário de design que colabora para a estruturação do conhecimento de design, como também, pode ser diretamente empregado como ferramenta de concepção de jogos.

Capítulo 5 - Considerações Finais e Trabalhos Futuros

O processo de desenvolvimento de jogos envolve intrinsecamente a multidisciplinaridade: são programados por desenvolvedores, ilustrados por artistas gráficos e sonorizados por músicos. Entretanto, a atividade essencial à produção de jogos não recai sobre um papel técnico. A definição da forma e do conteúdo de um jogo são de responsabilidade do designer de jogos. Fundamental na criação de jogos, o designer é tido como o profissional responsável pelas ideias que dão início ao conceito de um novo jogo, bem como, pelo seu posterior detalhamento minucioso. Cabe a ele o crédito por fazer do jogo um software divertido e engajante, e a penalização por não alcançar tal objetivo. Por outro lado, o instrumental que o profissional tem à disposição pouco evoluiu desde o início da atividade e isso tem sido considerado um fator limitante para a área (Dormans, 2012).

O principal recurso utilizado por designers para o registro de ideias e a concepção do jogo tem sido o Documento de Design. Ele serve não somente como documentação, mas também age como um guia, definido o rumo do processo de produção (Kreimeier, 2003). Contudo, diversos autores apontam problemas em seu emprego massivo e sugerem soluções alternativas, como esquemas visuais e diagramas, por vezes advindas de outras áreas (Neil, 2012). Em 1994, Costykian (1994) apresentou uma discussão sobre a necessidade de ferramentas, conceituais ou de software, que pudessem auxiliar e padronizar o ofício do designer de jogos. Após alguns anos, Church (1999) enriqueceu a discussão e incentivou uma série de publicações posteriores que propuseram abordagens alternativas e complementares de design (Almeida & Silva, 2013a).

Segundo Church (1999), extrair características de jogos existentes pode contribui significativamente para a concepção de novos projetos. Há uma grande carga de conhecimento e experiência de design embutida nos jogos existentes no mercado, que tipicamente servem de fundação para novos jogos. Church (1999) enfatiza essa ideia ao afirmar que nenhum designer de jogos "trabalha no vácuo". "Como designers, nós construímos nossas ideias sobre nossas experiências e as ideias passadas de outros" (Church, 1999, p.1). A noção de que os jogos podem ser compreendidos e discutidos a partir de suas partes formadoras é implicitamente aceita e referenciada em diversos trabalhos (Costykian, 1994; Church, 1999; Björk, 2003, Kreimeier, 2003; Neil, 2012), embora nenhuma definição formal tenha sido estabelecida.

O modelo atual de design de jogos, orientado a narrativas e guiado por uma documentação intensamente textual, dificulta a padronização e a formalização do design (Neil, 2012). As tentativas de solucionar esse problema, como as discutidas em Kreimeier (2003), Neil (2012) e Dormans (2012) tem sido direcionadas à estruturação de conceitos e práticas recorrentes de design de jogos, distanciando-se da formalização das partes menores que os formam. Por essa razão, a motivação deste trabalho foi norteada por uma questão levantada no início do Capítulo 1: **Seria possível definir elementos estruturantes do design de forma que jogos e gêneros possam ser tratados como composições de partes menores?** Os exemplos e

discussões apresentados no decorrer deste trabalho constituiem uma demonstração de que é possível tratar jogos como uma soma de partes identificáveis e estruturáveis.

No decorrer deste trabalho foram apresentados diversos casos concretos de modelagem de jogos e gêneros por meio de componentes. Além destes, o Capítulo 4 contém a documentação de duas narrativas de design de jogos sob a perspectiva da componentização. No trabalho, foram expostos componentes que descrevem quatro jogos por completo: [Orbit Jumper], [Space Shooter], [3D Platfomer] e [Space Invaders]. Estes e outros exemplos de modelagem de design apresentados seguiram a abordagem de componentização e utilizam predominantemente a Linguagem de Componentes, desenvolvida especificamente para tal fim. Cabe reiterar que a Linguagem é acompanhada de dois outros instrumentos também apresentados no trabalho, embora em menor ocorrência: os Diagramas e os Mapas de Componentes. A preferência pela Linguagem de Componentes ocorreu pela simplicidade e praticidade em escrevê-la, eliminando a necessidade de ferramentas visuais, de produzir ilustrações e organizar leiautes.

Em termos práticos, o principal objetivo deste trabalho foi mostrar que é possível discutir, descrever e construir jogos por meio de suas partes formadoras. Para tanto, foi proposto um modelo estrutural para descrever componentes de jogos e aplicado em várias situações. Dessa forma, os exemplos dispostos no trabalho, além daqueles tratados especificamente no Capítulo 4, constituem uma demonstração concreta da aplicação da componentização na análise e no design de jogos. Eles não somente contribuem com a validação do trabalho, como também, fornecem indícios de aplicações diversas e estudos futuros para a abordagem de componentes.

5.1 Constribuições do Trabalho

Neste trabalho, acredita-se que a estruturação da forma de pensar, projetar e registrar jogos sob uma abordagem de componentização pode contribuir significativamente com o processo de design. Para o autor, existe um vasto conhecimento de design embutido nos jogos já produzidos que pode ser resgatado por meio dos componentes que empregam. Nesse contexto, destacam-se três pontos diferenciais deste trabalho em relação aos relacionados no Capítulo 2, que são discutidos no texto que segue:

1. A caracterização de aspectos de jogos como **blocos de construção**;
2. A junção de um **vocabulário de design** com uma **linguagem de modelagem**;
3. A **demonstração de aplicação** da abordagem em projetos de jogos.

Kreimeier (2003), Neil (2012) e Almeida & Silva (2013a) realizaram um levantamento acerca das propostas de ferramentas de design. Entre elas nota-se uma ênfase na busca pelo estabelecimento de um vocabulário estruturado de design de jogos. Acima de um simples dicionário de termos, existe o consenso de que é necessário definir uma estrutura de conceitos reusáveis de design. Por outro lado, os trabalhos discutidos tratam tais conceitos como

definições esparsas, por vezes muito teóricas, que não podem ser agrupadas ou conectadas para constituir jogos e gêneros. Essa caracterização de **blocos de construção**, tratanto aspectos de jogos como componentes que podem ser organizados incrementalmente para montar jogos e gêneros é enfaticamente o ponto diferencial da abordagem desta tese e foi apresentado em todos os exemplos de modelagem de design utilizando a Linguagem de Componentes.

Paralelamente aos eforços em se estabeler um vocabulário estruturado de design, publicações discutidas no Capítulo 2 apresentam uma segunda ênfase, que corresponde a busca pela definição de linguagens visuais de design. Tentativas de aplicação de diagramas conhecidos, como UML e Redes de Petri, foram documentadas por diferentes autores, bem como, trabalhos que propõem esquemas visuais próprios. A Figura 4 apresentou um esquema que mapeia as abordagens e suas ênfases de forma hierárquica e sintética. Em especial, estão indicados os dois principais grupos de ênfases observadas: vocabulários de design e linguages visuais. Embora os resultados dos trabalhos publicados não tenham sucedido como ferramentas de uso efetivo, suas existências são fortes indicativos de que esses são pontos relevantes e necessários à área. Por outro lado, não foi observada uma intenção clara de criar uma interseção entre os dois temas apontados: **vocabulários estruturados e linguagens visuais**. Este é o segundo ponto diferencial da abordagem, que permite não somente a definição de um vocabulário de componentes, como também, possui uma linguagem própria para representá-los e modelar suas aplicações em jogos e gêneros.

Kreimeier (2003) e Neil (2012) apontam a existência de grande ceticismo por parte de designers profissionais acerca das abordagens propostas como alternativas ou complementos ao Documento de Design. Os designers consideram que essas propostas embutem uma visão demasiadamente formalizada para uma atividade que constantemente requer criatividade e flexibilidade. Nesse contexto, os métodos discutidos no Capítulo 2 são geralmente propostos para analisar jogos e não para projetá-los. Além disso, são trabalhos que não demonstram uma aplicação prática dos métodos que propõem, corroborando para distanciá-los do emprego em cenários reais. Segundo Dormans (2012), mesmo os autores desses trabalhos são incertos quanto ao benefício real que suas propostas podem trazer se aplicadas, e dessa forma, acabam aprofundando-se em discussões especulativas e questões teóricas. Privilegia-se a busca pelo vigor formal em detrimento a um instrumento simples e praticável. Em oposição a este cenário, o presente trabalho enfatizou a demonstração: todas as discussões apresentadas foram acompanhadas de exemplos que tiveram como alvo a **aplicação da abordagem em situações de design frequentes em jogos existentes** e, por conseguinte, de emprego mais realista. Essa preocupação em modelar partes de jogos e gêneros praticados pela indústria compreende o terceiro ponto diferencial deste trabalho. Além dos três pontos discutidos, há promissoras possibilidade de apliações futuras da abordagem sob diferentes perspectivas.

5.2 Perspectivas e Cenários de Usos Futuros

Designers comumente se apoiam em suas experiências passadas e no trabalho de outros para construir seus jogos (Church, 1999). Nesse sentido, eles usualmente investigam jogos similares ao que estão produzindo para determinar acertos e falhas, identificando suas características e descobrindo quais podem ser reusadas de forma benéfica em um novo projeto. De fato, é comum a ocorrência de reuso ou "empréstimo" de aspectos entre jogos, mesmo entre aqueles que não possuem relação. Como exemplo, a árvore genealógica de aspectos de jogos apresentada previamente na Figura 16 organiza hierarquicamente jogos de diferentes franquias e gêneros que apresentam relações de reuso. Ao tratar jogos e gêneros como composições de aspectos conectados, a abordagem de componentes torna possível estabelecer relações entre os conceitos que descrevem tais jogos e gêneros. Nesse contexto, surgem possibilidades de aplicações futuras que podem extender a abordagem a diferentes cenários.

A abordagem de componentes de design faz parte de um plano mais amplo, idealizado como algo complexo que guiará trabalhos futuros do autor na área. No plano, esta tese estabelece a base conceitual que viabilizará a realização de tais trabalhos. Dessa forma, a abordagem foi inicialmente idealizada como um conjunto de três partes, que compreende não somente os estudos a fim de estabelecê-la, como também, os instrumentos que a extendem e provêm suporte ao seu emprego. O plano inicial mencionado define um conjunto de instrumentos de design, compreendido por:

1. **Modelo estrutural**: a definição e demonstração de um modelo estrutural e uma linguagem de componentes para modelagem de jogos e gêneros. Esta parte estabelece a base conceitual para as demais e compreendeu o objeto de estudos desta tese.

2. **Base de Conhecimentos**: a definição e implementação de uma base de dados para gerenciamento de componentes e modelos de jogos, armazenando o conhecimento de design estruturado e possibilitando o emprego de técnicas de descoberta de conhecimento.

3. **Ambiente de Design**: a definição e implementação de um Ambiente de Design de Jogos, que faça interface com a base de conhecimentos e permita a prototipação de jogos a partir da modelagem por componentes registrada.

Como parte de um conjunto maior de instrumentos de design, é possível notar que a abordagem de componentes traz perspectivas de aplicações e contribuições a diferentes cenários, em especial, a indústria, a pesquisa e o ensino.

5.2.1 Perspectivas para a Indústria

A abordagem deste trabalho destaca-se por prover uma **Modelo Estrutural** para o design sob a perspectiva de blocos de construção. Nesse sentido, os jogos são montados a partir de de partes menores, que podem ser relacionadas para a constituição de aspectos mais complexos. Para tanto, o designer tem a disposição recursos para a modelagem de design –

Classificação, Composição e Associação –, que pode ser registrada por meio de diferentes notações – Linguagem de Componentes, Diagramas de Componentes e Mapas de Design. Do ponto de vista do designer, a combinação dos recursos mencionados contribui como um mecanismo estruturado de documentação do conceito do jogo, junto ao Documento de Design.

Tomando o design sob uma perspectiva de engenharia, os métodos de Análise por Decomposição e Design por Composição discutidos neste trabalho podem servir de guias para a execução do processo de design, especialmente ao se considerar designers iniciantes. Os componentes definem partes de jogos, blocos de construção que podem ser usados para se montar novos jogos. Logo, tratar a atividade de design sob a ótica de componentes ajuda a quebrar a concepção de um jogo em problemas menores.

A **Análise por Decomposição** representa um método de engenharia reversa "top-down" pelo qual designers desmontam jogos em seus componentes, processo demonstrado para o jogo Space Invaders. A engenharia reversa de jogos em componentes representa por si, um importante exercício de design, pois força designers a compreender a influência que cada parte menor possui no resultado final. Ademais, leva a um melhor entendimento do trabalho de outros designers, em especial, seus acertos e falhas. Analogamente, o **Design por Composição** representa um processo no qual designers "montam" o um jogo a partir da inter-relação de suas partes, seja iniciando sobre conceito maior ou sobre um componente específico. Por meio deste, os componentes são projetados, reusados, modificados e combinados a fim de criar novos. Esses, por sua vez, podem vir a fazer parte de componentes mais complexos, até gradativamente formar o design de um jogo. Nesse sentido, o conceito do jogo emerge apoiado na junção de suas partes menores.

Considerando o plano maior reiterado na seção anterior, a construção de uma **Base de Conhecimentos** permitrá o registro de componentes de design, bem como, da modelagem de jogos e gêneros de forma a favorecer a pesquisa e o reuso de aspectos de jogos. Sob a perspectiva de quem concebe jogos, o reuso serve de base para novas ideias. Nesse sentido, a **Base de Conhecimentos** pode fornecer ao designer um instrumental que o ajudará a realizar combinações conceituais de partes de jogos, servindo de fundação para a geração de novas ideias. Ademais, os componentes documentos poderão compor um vocabulário comum de design.

Por fim, o próprio conceito de um Ambiente de Design de Jogos apresenta possibilidades promissoras à profissão. Fundamentando-se nos componentes de jogos como elementos estruturais para o design, existe a perspectiva de que seja possível implementar um ambiente computacional de assistência ao designer na concepção de jogos, em especial, nas atividades de análise, design e prototipação. Mais especificamente, espera-se permitir que o designer explore a Base de Conhecimentos de componentes, estude a composição de jogos e gêneros, utilize ferramentas de análise para identificar relações com mercado e crítica, monte novos jogose e faça experimentações nos protótipos gerados. Além disso, considerando a natureza dos componentes, o ambiente poderá prover kits iniciais de jogos como esquelteos (*templates*) que podem ser preenchidos, assistindo na concepção de novos projetos e em sua experimentação,

ao gerar protótipo jogáveis. Além disso, o ambiente pode registrar o status e as ações do jogador no protótipo, além de outras métricas que forneçam retorno sobre os componentes utilizados. Esse processo iterativo de design e testes estéticos pode facilitar a investigação de diferentes combinações de componentes, favorecendo uma abordagem incremental de design.

De um modo geral, há diversas perspectivas promissoras que se apóiam sobre os instrumentos conceituais estabelecidos neste trabalho. Dessa forma, espera-se que a abordagem possa futuramente contrubuir com uma sistematização do processo de design de jogos (conceito previamente ilustrado pelo esquema da Figura 17).

5.2.2 Perspectivas para a Pesquisa

A abordagem de componentes de design não representa a aplicação de uma técnica existente da computação ou originada de outra área no design de jogos. O conceito gerador deste trabalho teve origem empírica, com base nas experiências prévias do autor como desenvolvedor de jogos. Com a definição do plano inicial, o conceito foi gradativamente evoluindo para a abordagem no estado atual, agregando recursos que permitissem satisfazer seus requisitos. Nesse sentido, o conhecimento previo do autor como profissional de computação teve grande influência para a definição da estrutura "engenheira" da abordagem. Portanto, embora a abordagem de componentes não tenha sido construída como uma especialização da computação para o design de jogos, ela torna-se indissociável da computação. Nesse sentido, espera-se tomar a abordagem como base para empregar técnicas de computação e construir ferramentas de software que auxiliem o designer de jogos.

O conceito do Ambiente de Design de Jogos representa o resultado maior que poderá ser alcançado com base na conclusão do presente trabalho e a fusão entre design de jogos e técnicas de computação. Ele traz perspectivas promissoras não somente à indústria, como abre espaço para realização e apliação de pesquisas em design de jogos. Nesse sentido, ao tratar os jogos como uma composição de partes e documentar tal informação em uma base de dados, se torna possível empregar técnicas de descoberta de conhecimento. Como exemplo, pode-se analisar o nível de similaridade entre jogos, assim como, a popularidade, raridade ou sucesso de determinados componentes em jogos ou gêneros, a partir do cruzamento de dados de mercado e crítica especializada. Também se torna possível gerar "árvores genealógicas" que relacionem jogos, seus componentes e gêneros, de forma a representar a evolução cronológica de títulos e as relações existentes entre suas constituições (a Figura 16 apresentou um exemplo conceitual).

A estrururação do conhecimento de design possibilita a construção de assistentes inteligentes que empreguem algoritmos de similaridade para fornecer sugestões com base na "mistura" de componentes de um projeto. Nesse sentido, uma ferramenta de software pode analisar os componentes utilizados pelo designer no conceito de um jogo e sugerir alterações ou melhorias. Como exemplo, a partir da similaridade da composição do projeto com gêneros e outros jogos, o ambiente pode realizar sugestões de novos componentes para o projeto. Sugestões desta natureza podem envolver componentes típicos aos gêneros detectados, componentes comumente relacionados aos que estão no design, ou mesmo, aqueles presentes

em títulos similares que tenham sucesso no mercado. De forma similar, pode advertir quanto à potenciais conflitos ou raridade nas composições realizadas. Além disso, com base em uma análise de padrões frente à composição de jogos similares, pode indicar características como falta ou excesso de desafios, itens, movimentos ou habilidades. Mesmo fora do escopo do design, informações relacionadas aos projetos dos jogos podem ser investigadas, como relações entre métodos de desenvovimento empregados, complexidade dos jogos e componentes.

Acredita-se que a abordagem deste trabalho não representa apenas uma contribuição positiva ao degisn de jogos, mas ela permite aproximar a área à computação. Nesse sentido, ela abre um considerável leque de possibilidades futuras para o emprego de técnicas e algoritmos da computação, seja para a extração de informações que aulixiem na concepção de jogos ou para identificar relações referentes aos processos pelos quais os jogos são desenvolvidos.

5.2.3 Perspectivas para o Ensino

O autor desta tese é servidor da Universidade Tecnológica Federal do Paraná, instituição em que atua como professor do curso de Ciência da Computação. Concomitantemente à realização deste trabalho, o autor teve a oportunidade de ministrar disciplinas de tema livre, nas quais pôde trabalhar com desenvolvimento de jogos e, consequentemente, com a experimentação de ideias para a concepção da abordagem de componentes. Tal experimentação foi empregada constantemente durante a produção de conteúdo para as disciplinas, em especial, na criação de estudos de casos, na implementação de protótipos e na identificação de espectos em jogos existentes. As disciplinas serviram de espaço para testes de conceitos e contribuíram significativamente para elaboração e amadurecimento da abordagem, que constuituia-se apenas de uma coleção de intensões.

O objetivo das disciplinas ministradas foi capacitar os alunos a planejar, projetar e produzir jogos, tratando-os como uma composição de aspectos. Nesse sentido, os conceitos que nortearam o desenvolvimento da abordagem – blocos de construção de design e composição/decomposição de jogos – serviram de guia para as disciplinas, organizando sua execução de forma similar a um projeto real. De uma forma geral, o planejamento e a execução das disciplinas foram norteados por três diretrizes:

- Como **aprendizes**, os alunos foram ambientados ao design e desenvolvimento de jogos por meio de aspectos;
- Como **designers**, os alunos trabalharam para analisar, discutir, planejar e projetar jogos como composições;
- Como **desenvolvedores**, os alunos implementaram os jogos guiando-se pelos aspectos fundamentais que identifacaram para seus projetos.

A partir das três diretrizes apresentadas, pode-se notar que as disciplinas de jogos não representaram apenas um ambiente de testes, mas também, a realização de uma experiência de ensino de design baseado em componentização. Nesse sentido, alunos com nenhum

conhecimento prévio na área de desenvolvimento de jogos foram capazes de pensar, registrar e produzir jogos como uma composição de partes, mesmo que realizando essas atividades de maneira bastante informal. Não há a pretensão de se propor ou provar um método de ensino, mas a experiência com as disciplinas mostrou que é possível discutir os jogos por meio de seus aspectos com aprendizes.

5.3 Resultados Relacionados

No decorrer do desenvolvimento deste trabalho foram realizadas diversas atividades que levaram ao amadurecimento da abordagem de componentes e sua fundamentação. Embora não sejam o alvo de estudos da tese, representam consideráveis resultados relacionados. Eles estão brevemente apresentados nesta subseção.

5.3.1 Publicações

Durante as atepas iniciais deste trabalho foram publicados quatro artigos envolvendo estudos que levaram a sua escrita. Dois desses artigos foram determinantes para este trabalho, pois representam explicitamente o conteúdo abordado nos Capítulos 1, 2 e 3. Além disso, receberam premiações nos eventos aos quais foram submetidos.

Towards a Game Design Patterns Suggestion Tool (Almeida et al, 2013)

Publicado no XII Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames 2013), evento oficial de desenvolvimento de jogos da SBC (Sociedade Brasileira de Computação). Além disso, é o principal e maior evento da área na América Latina. O artigo descreve um experimento do emprego de técnicas de mineração de dados e análise de similaridade de textos para sugerir os Padrões de Design de Jogos (Björk et al., 2003) que podem estar contidos em um determinado jogo. Esse trabalho teve relevância na verificação das possibilidades do emprego de análises automatizadas para encontrar similaridades no design de jogos distintos. Ele foi o primeiro contato com as publicações de propostas de ferramentas de design. O artigo pode ser acessado digitalmente pelo site do evento[172].

Requirements for Game Design Tools (Almeida e Flávio, 2013b)

Publicado no XII Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames 2013). Ele apresenta uma análise das ferramentas de design de jogos a fim de enumerar diretrizes que ajam como requisitos na a elaboração de futuras propostas de ferramentas de design. O resultado desse contribuiu para a concepção inicial da abordagem de componentes. Ele pode ser acessado digitalmente pelo site do evento[173].

Towards a Library of Game Components (Almeida e Flávio, 2013c) - **Premiado**

Publicado no XII Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames 2013). O trabalho foi premiado como segundo melhor artigo completo da Trilha de Arte e Design. Ele

[172] http://www.sbgames.org/sbgames2013/proceedings/artedesign/10-dt-paper.pdf (visitado em 10/03/2014)
[173] http://www.sbgames.org/sbgames2013/proceedings/artedesign/33-dt-paper.pdf (visitado em 10/03/2014)

descreve a proposta preliminar da Abordagem de Componentes de Jogos e deu início ao Capítulo 3 deste trabalho. O documento pode ser acessado digitalmente pelo site do evento[174].

A Systematic Review of Game Design Methods and Tools (Almeida e Flávio, 2013a) - **Premiado**

Publicado no XII International Conference of Entertainment Computing (ICEC 2013), evento de visibilidade internacional na área. Os Anais do evento foram publicados como livro da série Lecture Notes in Computer Science (LNCS) da Springer (vol. 8215) e o artigo corresponde a um capítulo do mesmo. O trabalho foi premiado como melhor artigo completo do evento. Ele apresenta uma revisão analítica do estado da arte das ferramentas e métodos de design de jogos publicadas por pesquisadores da academia e designers profissionais. O artigo contribuiu para a redação dos Capítulos 1 e 2 desta tese. Informações sobre a publicação podem ser encontradas no link do livro[175].

5.3.2 Protótipos de Jogos

Diversos protótipos de jogos foram construídos pelo autor deste trabalho. Eles serviram de espaço de testes e experimentações para a concepção, desenvolvimento e, ao final, demonstração da abordagem. Além disso, foram utilizados como exemplos nas disciplinas de desenvolvimento de jogos ministradas (seção 5.2.3). Alguns deles são apresentados e expostos nas figuras abaixo.

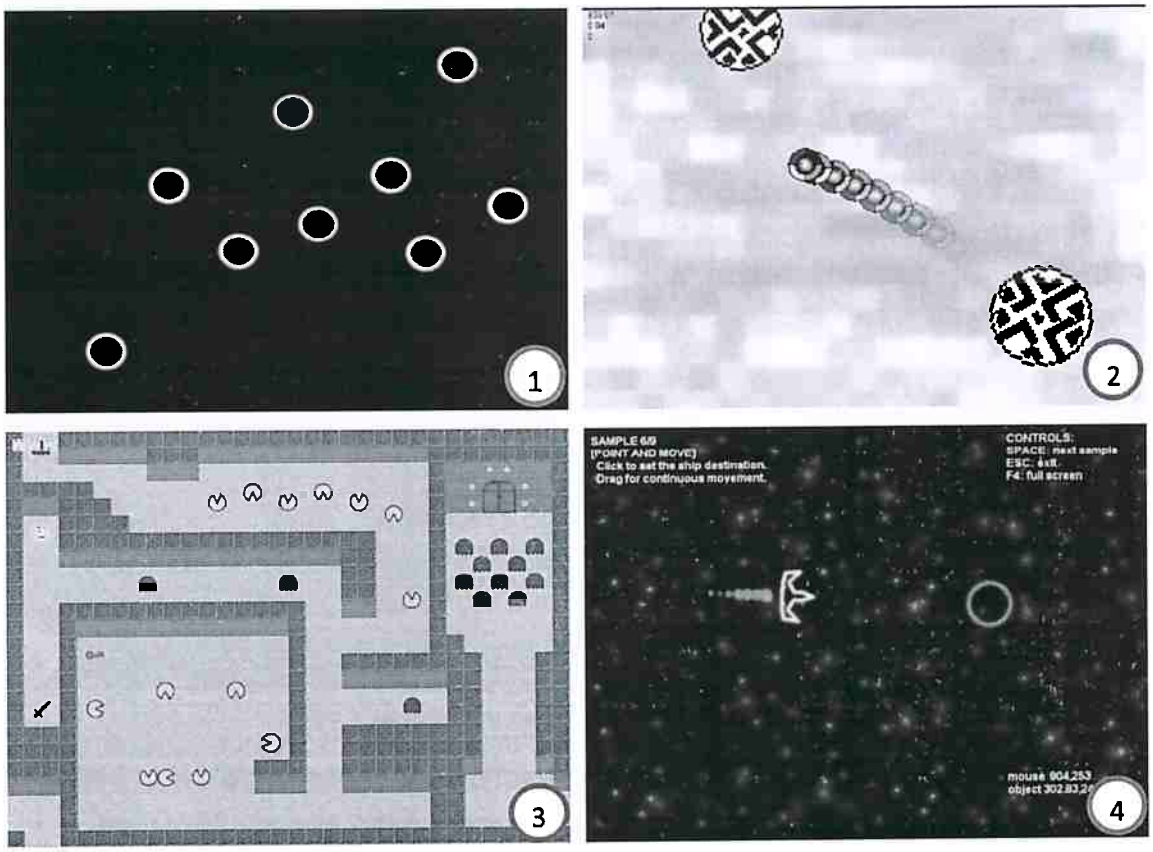

Figura 95 - Protótipos de demonstração 2D produzidos na Game Maker Studio.

[174] http://www.sbgames.org/sbgames2013/proceedings/artedesign/01-dt-paper.pdf
[175] http://link.springer.com/chapter/10.1007/978-3-642-41106-9_3

Os protótipos expostos na Figura 95 foram desenvolvidos com o motor Game Maker Studio. Eles demonstram aplicações de componentes comumente encontrados em jogos 2D.Os dois primeiros protótipos foram utilizados como exemplo nesta tese. O Bouncing Balls (1) serviu de exemplo básico na disciplina e o Orbit Jumpet foi construído em torno do componente [Orbit Platform]. O protótipo Dungeon Runner (3) teve como protósito demonstrar uma composição de jogo com essencialmente baseada em componentes encontrados em jogos de aventura com labirintos, como os populares Legendo of Zelda: A Link to the Past[176] e Adventure[177]. Embora a imagem mostre o mapa por inteiro, quando em execução ele emprega um componente limitador de visibilidade [Lamp] que escurece a tela e cria um foco de luz ao redor do personagem, criando tensão e intensificando a necessidade de exploração do ambiente. O último aplicativo da figura, Control Components Demo (4), foi construído para demonstrar a execução de diveros componentes de controle 2D, livres do contexto dos jogos que os empregam.

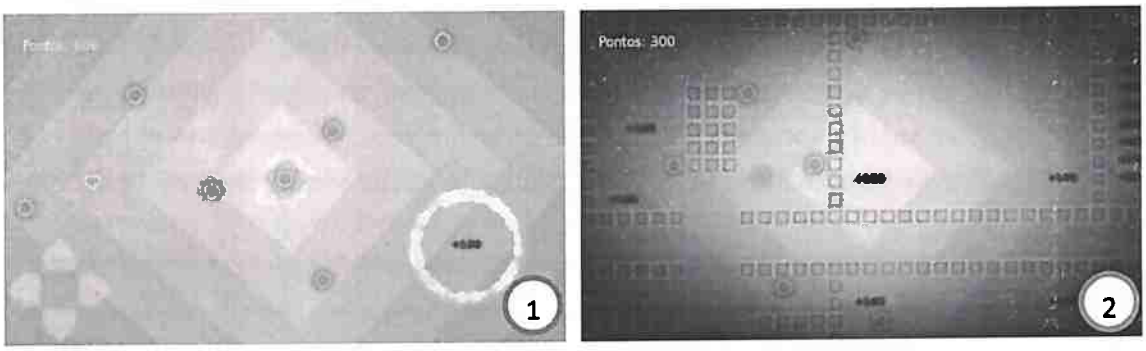

Figura 96 - Demonstrativos de jogos produzidos no motor Cocos2d-x.

A Figura 96 mostra dois protótipos desenvolvidos como demonstrativos de jogos com o motor Cocos2d-x. Nomeados Fuja (1) e Fuja+ (2), eles serviram como kits introdutórios para os projetos dos alunos e empregam boa parte dos componentes evidenciados nos protótipos anteriormente mencionados.

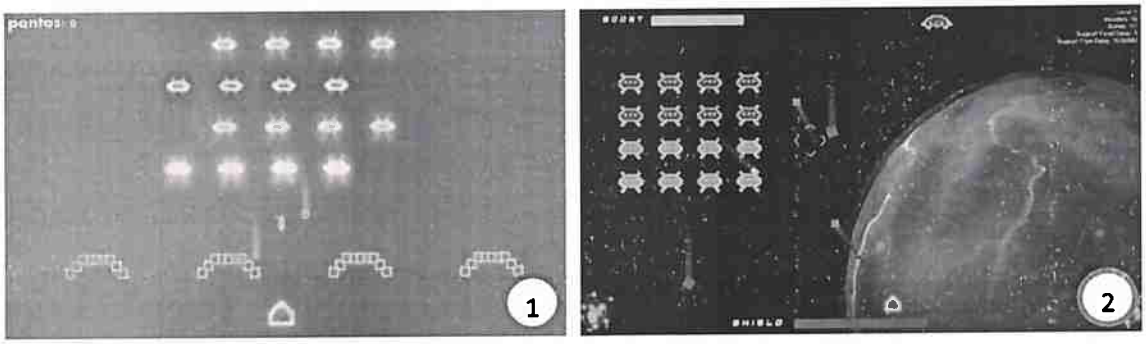

Figura 97 - Space Shooter a primeira versão (1) era fortemente baseada no conceito do Space Invaders.

Os protótipos produzidos para a demonstração da abordagem discutidos no Capítulo 4 representam as versões finais de seus respectivos jogos. Nesse sentido, eles passaram por versões interemediárias que representavam conceitos consideravelmente distintos. Como

[176] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/TheLegendOfZeldaALinkToThePast
[177] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/Adventure

exemplo, a primeira versão do Space Shooter (Figura 97, foto 1) foi construída no motor Game Maker Studio e teve um conceito fortemente baseado no Space Invaders original. Ela acabou apresentando pouco espaço para mudanças de design e implementação, o que feria o propósito para o qual estava sendo desenvolvida. Por essa razão, foi descartada e foi posteriormente substituída pela versão (2), reconstruída no motor Unity 3D e enfatizando a mistura de componentes do Space Invaders com outros populares em jogos de tiro contemporâneos.

O segundo projeto de demonstração da abordagem, 3D Platformer, também passou por versões intermediárias que foram descartadas por razões técnicas e, especialmente, por melhorias em seu design. A Figura 98 apresenta as quatro versões gradativamente desenvolvidas para o protótipo: (1) iniciado como um título tradicional 2D, (2) empregando perspectiva 2.5D, (3) como um jogo 3D usando componentes do jogo Marble Madness[178] e, em sua versão final (4), formatado para um jogo de plataformas 3D de câmera livre.

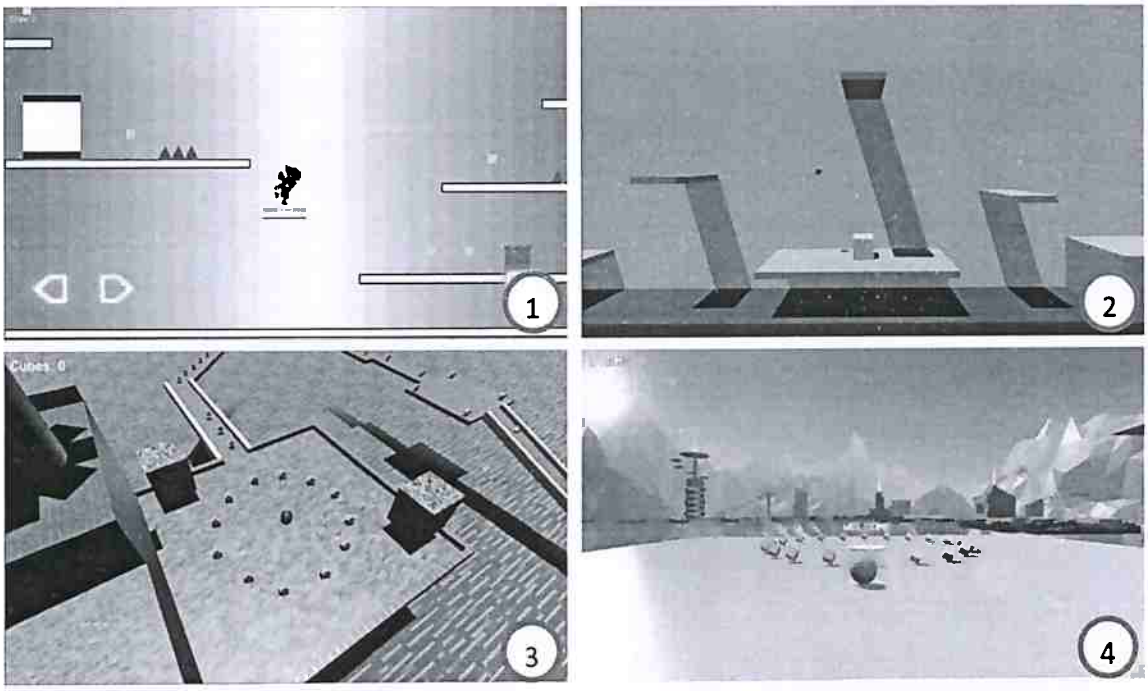

Figura 98 - O protótipo 3D platformer teve quatro versões diferentes.

5.3.3 Disciplinas Ministradas

Durante a realização das disciplinas de desenvolvimento de jogos ministradas pelo autor previamente mencionada na seção 5.2.3, os participantes construíram projetos de jogos como requisito do processo de avaliação estabelecido. Embora não houve emprego formalizado da abordagem desta tese, as disciplinas contribuíram para discussões e experimentações da mesma. Os projetos dos alunos foram baseados em componentes de títulos populares e alguns estão expostos na Figura 99: Alone In The Space (1), Andry Nerd (2), Mr. Square (3), Runner (4), Space Detour (5), As Aventuras de Tux e Fox (6), A Saga do Vegetal (7) e Gravity Jumper (8).

[178] http://tvtropes.org/pmwiki/pmwiki.php/VideoGame/MarbleMadness

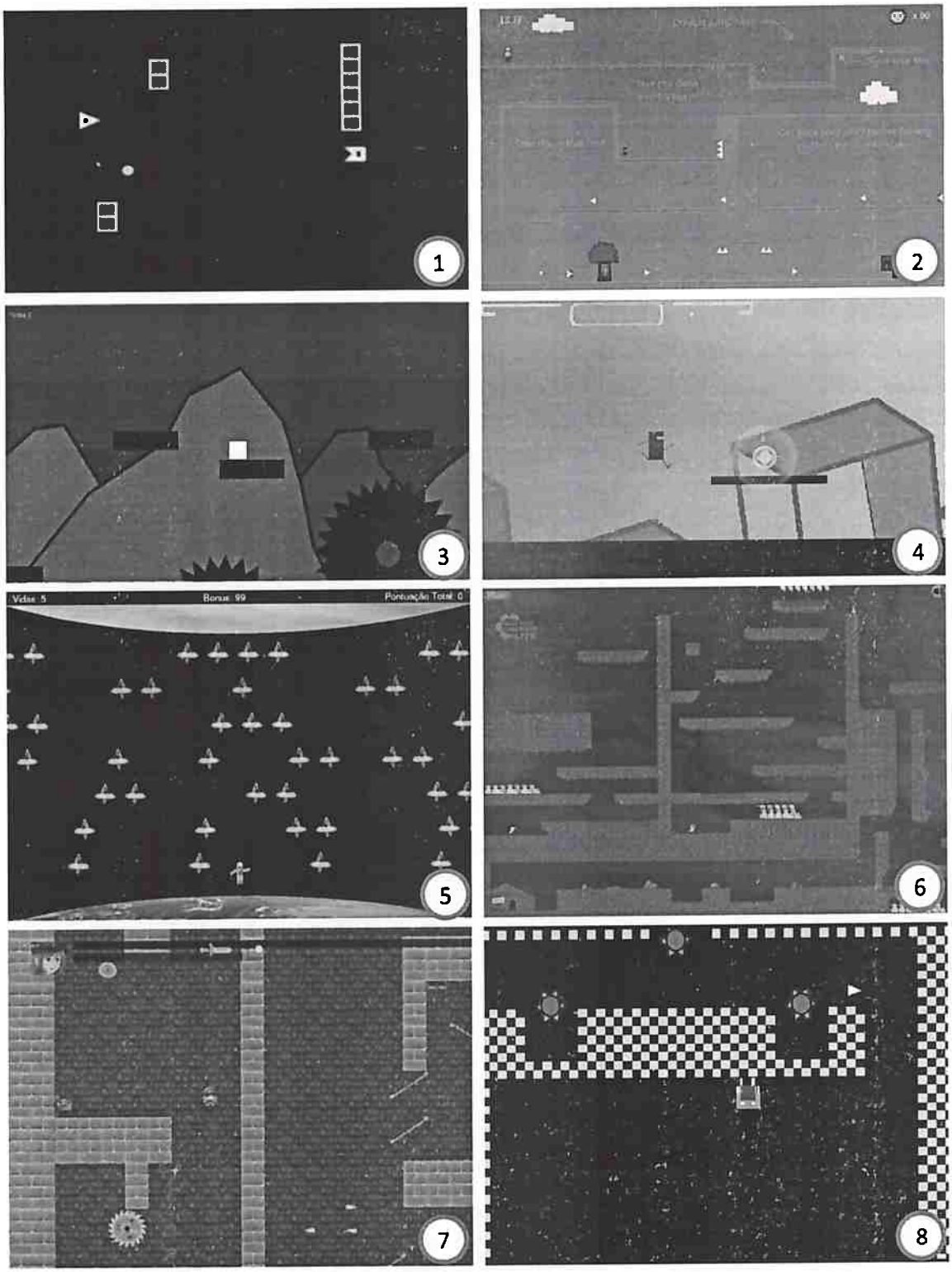

Figura 99 - Projetos de Jogos desenvolvidos pelos alunos.

5.4 Perspectivas de Estudos Futuros e Próximos Passos

O estado atual da abordagem de componentes traz instrumentos que permitem a representação do conceito de um jogo tal como uma composição de aspectos. Se comparada a forma como se dá a implementação do mesmo em um motor gráfico habitual, os aspectos encontram-se em um nível maior de abstração. Além disso, o Ambiente de Design de Jogos, um dos propósitos futuros da abordagem, prevê a geração de protótipos jogáveis com base em especificações de componentes. Logo, será necessário gerar código fonte para um motor gráfico com base na descrição de componentes. Em vias de tornar possível a tradução de um modelo em Linguagem de Componentes, como aqueles apresentados pelos exemplos contidos nesta tese, para uma descrição que permita a geração de protótipos exercutáveis, há pontos que necessitam de estudos mais aprofundados. Além disso, a própra extensão da aplicação da abordagem não pode ser delineada.

Considerando a perspectiva do Ambiente de Design, torna-se relevante investigar as consequências que a simplicidade apresentada nos exemplos de modelagem de componentes pode trazer ao processo de tradução modelo-protótipo. Quando componentes são nomeados e adicionados ao design de um jogo, há uma grande carga semântica associada ao seu uso. Como exemplo, um componente classificado como <enemy> traz consigo diversas características e funcionalidades inerentes ao papel de um "inimigo" em um jogo que são claras a designers e jogadores. No entanto, do ponto de vista da implementação, tal carga semântica não é trivial. Nesse sentido, detalhes específicos à implementação do software dos jogos são omitidos no design, tais como gerencimaneo de objetos, posicionamento de entidades no mundo do jogo, máquinas de estados e renderização. Tais aspectos foram desconsiderados porque a modelagem de componentes é um instrumento de design e não possui relação com a implementação do software do jogo. Contudo, considerando o objetivo a perspectiva do Ambiente de Design de Jogos, será preciso criar uma "ponte" entre a modelagem e a geração de protótipos jogáveis. Esta "ponte" consitui um dos alvos de trabalhos futuros.

A extensão das situações de design nas quais a abordagem de componentes é aplicável como um instrumento de design não pôde ser verificada. O esforço necessário para se decompor jogos existentes, bem como, para compor novos projetos tornou inviável a realização de vários estudos de caso. Também não é possível inferir as consequências de aplicação da abordagem considerando apenas os resultados obtidos nos estudos de casos realizados, dada a enorme diversidade de jogos e gêneros existentes no mercado. A fim de delimitar a amplitude de apliação da abordagem é inerentemente necessário trabalhar com gêneros e jogos distintos daqueles tratados nos exemplos, de forma que ajudem a criar situações de design inexploradas com a modelagem de componentes. Nesse sentido, seria relevante criar diversos estudos de casos de forma a explorar as possibilidades de aplicação da abordagem, bem como, encontrar seus limites. Estes estudos de casos constituem o primeiro passo de estudos futuros, sejam eles realizados na forma de projetos acadêmicos ou produtos. De uma forma geral, esses eforços poderão contribuir significativamente com a evolução das linguagens apresentadas e da abordagem como um todo.

A abordagem de componentes proposta e demonstrada no decorrer deste trabalho conseguiu responder positivamente à questão de pesquisa levantada. Nesse sentido, ela mostrou ser plausível definir elementos estruturantes do design de forma que jogos e gêneros possam ser tratados como composições de partes menores. Exemplos de modelagem de partes de jogos, bem como gêneros e jogos foram apresentados, tanto pela decomposição de títulos existentes quanto pela composição de novos projetos. Contudo, embora o estado atual da linguagem de componentes tenha sido suficiente para descrever o design dos exemplos e estudos de casos apresentados, a interação das representações de componentes com artefatos de documentação visual, como story boards, ilustrações de arte conceitual, ou mesmo, esquemas customizados, foram omitidas pela incapacidade do autor de produzir tais artefatos. De forma similar, a possibilidade de relacionar a abordagem de componentes com as algumas da discutidas no Capítulo 2 pode trazer melhorias a mesma. Assim como a diversidade de situações de design, estudos de aplicações conjuntas de técnicas e abordagens de design pode contribuir significativamente com evolução a componentização de design.

As principais contribuições deste trabalho referem-se à proposta e demonstração de um modo diferente de pensar o design de jogos. As construções e linguagens foram definidos para se alcançar e demonstrar o conceito de componentização de desgn e, dessa forma, descrever com sucesso os jogos por meio de suas partes menores. Contudo, não houve a intenção de se criar recursos definitivos. Nesse sentido, os recursos da abordagem podem ser modificados e melhorados, como fruto dos trabalhos e estudos futuros citados acima. De forma similar, a própria abordagem de componentes não é um recurso definitivo ou imutável e tampouco tem a pretensão de substituir os instrumentos correntes da área. Pelo contrário, ela representa um instrumento conceitual que pode ser agregado para enriquecer as soluções de design, sejam elas de uso comum aos designers ou particularmente desenvolvidas em seu cotidiano.

Referências Bibliográficas

Adams (2009) Ernest Adams. The Designer's Notebook: Sorting Out the Genre Muddle. Gamasutra, 2009. Mídia digital online. http://www.gamasutra.com/view/feature/4074/the_designers_notebook_sorting_.php. Último acesso em 18/07/2016.

Adams & Rollings (2003) Andrew Rollings; Ernest Adams. Andrew Rollings and Ernest Adams on Game Design. New Riders Publishing, 2003.

Albernathy & Rouse III (2014) Tom Albernathy & Richard Rouse III. Death to the Three Act-Structure. Apresentação na Game Developers Conference – GDC 2014, São Francisco, Califórnia, 2014. http://gamasutra.com/view/news/213337/Plot_is_overrated_Game_narrative_is_all_about_your_characters.php. Último acesso em 18/07/2016.

Almeida & Corrêa da Silva (2013) Marcos Silvano Almeida; Flávio Soares Corrêa da Silva. A Systematic Review of Game Design Methods and Tools. Em 12th ICECI: XII International Conference of Entertainment Computing, páginas 17-29. Springer Berlin Heidelberg, Outubro 2013.

Araújo & Roque (2009) Manuel E. Araújo & Licínio Roque. Modeling Games with Petri Nets. Proceedings of 2009 Digital Games Research Association Conference (DiGRA). Brunel University, 2009.

Bates (2004) Bob Bates. Game Design. 2ª Edição. Thomson Course Technology, 2004.

Bilas (2002) Scott Bilas. A Data-Driven Game Object System. Apresentação na Game Developers Conference – GDC 2002. http://gamedevs.org/uploads/data-driven-game-object-system.pdf. Último acesso em 20/01/2016.

Beck (2004) Kent Beck. Extreme Programming Explained: Embrace Change. 2nd edition. Addison-Wesley. Novembro, 2004.

Bethke (2003) Erik Bethke. Game development and production. Texas: Wordware Publishing, Inc, 2003.

Björk et al. (2003) Staffan Björk; Jussi Holopainen; Sus Lundgren. Game Design Patterns. Em DiGRA 2003: Level Up - Proceedings of Digital Games Research Conference. Utrecht University, 2003.

Brom & Abonyi (2006) Cyril Brom & Adam Abonyi. Petri-Nets for Game Plot. Proceedings of AISB, Vol. 3, 2006.

Bura (2006) Stéphane Bura, S. A Game Grammar, 2006. http://www.stephanebura.com/diagrams/. Último aceso em 18/07/2016.

Campbell (2012) Colin Campbell. 20 Game Studios We Lost in 2012. IGN, dezembro de 2012. http://www.ign.com/articles/2012/12/11/20-studios-we-lost-in-2012. Último acesso em 18/07/2016.

Cerny (2013) Mark Cerny. The Road to the PlayStation 4. Apresentação na GameLab Conference 2013. http://www.ign.com/videos/2013/06/29/the-road-to-the-playstation-4. Último acesso em 10/04/2014.

Chandler (2009) Heather M. Chandler, H. M. The Game Production Handbook (2nd ed.). Hingham, Massachusetts: Infinity Science Press, 2009.

Church (1999) Doug Church. Formal Abstract Design Tools. Gamasutra, julho 1999. http://www.gamasutra.com/view/feature/3357/formal_abstract_design_tools.php. Último acesso em 14/07/2016.

Church (2003) Doug Church. Object Systems. Apresentação em Conferência Técnica. Seul, Coréia do Sul. http://chrishecker.com/images/6/6f/ObjSys.ppt. Último acesso em 18/07/2016.

Clarke (2014) Scott Clarke. The Evolution of Single Player Co-op. IGN, janeiro de 2014. http://www.ign.com/articles/2014/01/05/the-evolution-of-single-player-co-op. Último acesso em 18/07/2016.

Costikyan (1994) Greg Costikyan. I Have No Words & I Must Design. Interactive Fantasy Magazine, número 2, 1994. http://www.rpg.net/oracle/essays/nowords.html. Último acesso em 14/07/2016.

Crawford (1984) Chris Crawford. The Art of Computer Game Design, 1984.

Daniels (2015) Shonté Daniels. Big Budget Games Are a Dying Breed. Em: Motherboard. Mídia digital online, 2015. http://motherboard.vice.com/read/aaa-games-are-crumbling-under-their-huge-budgets. Último acesso em 18/07/2016.

Davies (2011) Tom Davies. Entity Systems. Blog pessoal, 2011. http://www.tomseysdavies.com/2011/01/23/entity-systems/. Último acesso em 20/01/2016.

Demachy (2003) Thomas Demachy. Extreme Game Development: Right on Time, Every Time. Gamasutra, julho de 2003. http://www.gamasutra.com/view/feature/2827/extreme_game_development_right_on_.php. Último acesso em 18/07/2016.

Dormans (2012) Joris Dormans. Engineering Emergence: Applied Theory for Game Design. Tese de Doutorado. Amsterdam University of Applied Sciences, 2012.

Dyer (2011) Mitch Dyer. 12 Game Studios That Died in 2011. IGN, dezembro de 2011. http://www.ign.com/articles/2011/12/01/12-game-studios-that-died-in-2011. Último acesso em 18/07/2016.

ESA (2016) Entertainment Software Association (ESA). *Essential Facts about the Computer and Video Game Industry*. 2016. http://www.theesa.com/wp-content/uploads/2016/04/Essential-Facts-2016.pdf. Acesso em: 09/05/2016.

Fabricatore et al. (2002) Carlo Fabricatore; Miguel Nussbaum; Ricardo Rosas. Playability in Action Videogames: A Qualitative Design Model. Em Human-Computer Interaction, volume 17, número 4, páginas 311-368, 2002.

Falstein & Barwood (2002) Noah Falstein & Hal Barwood. More of the 400: Discovering Design Rules. Apresentação na Game Developers Conference – GDC 2002. http://www.gdconf.com/archives/2002/hal_barwood.ppt. Último acesso em 04/09/2013.

Fowler (2004) Martin Fowler. UML Distilled: A Brief Guide to the Standard Object Modeling Language. Boston, MA: Addison-Wesley, 3a ed, 2004.

Fullerton et al. (2004) Tracy Fullerton; Chris Swain; Steven Hoffman. Game Design Workshop: Designing, Prototyping, and Playtesting Games. CMP Books, San Francisco, 2004.

Gamma, et al. (1994) Erich Gamma; Ralph Johnson; Richard Helm; John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994.

Gregory (2014) Jason Gregory. Game Engine Architecture. 2ª Edição. AK Peters, 2014.

Järvinen (2008) Aki Järvinen. Games Without Frontiers: Theories and Methods for Game Studies

and Design. Tese de Doutorado. University of Tampere, 2008.

Karmali (2013) Karmali Luke. The Division Has RPG and Survival Elements. IGN, dezembro de 2013. http://www.ign.com/articles/2013/12/17/the-division-details-on-driving-black-market-and-more. Último acesso em 18/07/2016.

Keith (2010) Clinton Keith. Agile Game Development with Scrum. Addison-Wesley Professional, 2010.

Kiczales et al. (1997) Gregor Kiczales; John Lamping; Anurag Mendhekar; Chris Maeda; Cristina V. Lopes; Jean-Marc Loingtier; John Irwin. Aspect-Oriented Programming. Em 11th ECOOP: 11th European Conference on Object-Oriented Programming, páginas 220-242, Springer-Verlag, 1997.

Kisslat (2013) Carsten Kisslat. A Tutorial to Machinations Diagrams. Mídia digital online. Gamasutra, 2013. http://www.gamasutra.com/blogs/CarstenKisslat/20130814/198216/A_Tutorial_to_Machinations_Diagrams.php. Último acesso em 18/07/2016.

Koster (2005) Raph Koster. A Grammar of Gameplay. Apresentação na Game Developers Conference, San Francisco CA, March 2005. URL http://www.raphkoster.com/gaming/atof/grammarofgameplay.pdf. Último acesso em 18/07/2016.

Kreimeier (2002) Bernd Kreimeier. The Case For Game Design Patterns. Gamasutra, março 2002. http://www.gamasutra.com/view/feature/132649/the_case_for_game_design_patterns.php. Último acesso em 14/07/2016.

Kreimeier (2003) Bernd Kreimeier. Game Design Methods: A 2003 Survey. Gamasutra, março 2003. http://www.gamasutra.com/view/feature/2892/game_design_methods_a_2003_survey.php. Último acesso em 14/07/2016.

Kuittinen (2008) Jussi M. Kuittinen. Computer-Aided Game Design. Master's Thesis in Information Technology. University of Jyväskylä, 2008. https://jyx.jyu.fi/dspace/bitstream/handle/123456789/18458/URN_NBN_fi_jyu-200803181271.pdf. Último acesso em 18/07/2016.

LaMothe (2002) André LaMothe. Tricks of The Windows Game Programming Gurus. 2ª Edição. Sams Publishing, 2002.

LeBlanc et al. (2004) Marc LeBlanc; Robin Hunicke; Robert Zubek. MDA: A formal approach to game design and game research. Em AAAI-04: American Association for Artificial Intelligence 2004 - Workshop on Challenges, 2004.

Lewis & Loftus (2011) John Lewis & William Loftus. Java Software Solutions: Foundations of Programming Design. 7ª ed. Pearson Education, 2011.

Leonard (1999) Tom Leonard. Postmortem: Thief – The Dark Project. Gamasutra, julho 1999. http://www.gamasutra.com/view/feature/131762/postmortem_thief_the_dark_project.php. Último acesso em 20/01/2016.

Librande (2010) Stone Librande. One-Page Designs. Apresentação na Game Developers Conference – GDC 2010, San Francisco CA, Março 2010. http://stonetronix.com/gdc-2010/OnePageDesigns.ppt. Último acesso em 15/07/2016.

Lindley (2003) Craig A. Lindley. Game Taxonomies: A High Level Framework for Game Analysis and Design. 2003. http://www.gamasutra.com/view/feature/2796/game_taxonomies_a_high_level_.php. Último acesso em 14/07/2016.

Lindley et al. (2008) Craig A. Lindley, Lennart Nacke and Charlotte C. Sennersten. Dissecting Play

– Investigating the Cognitive and Emotional Motivations and Affects of Computer Gameplay. In: Proceedings of CGAMES 08, Wolverhampton, UK: University of Wolverhampton, 2008.

Lord (2012a) Richard Lord. What is an Entity System Framework for Game Development? Blog pessoal, 2012a. http://www.richardlord.net/blog/what-is-an-entity-framework. Último acesso em 20/01/2016.

Lord (2012b) Richard Lord. Why use and Entity System Framework for Game Development? Blog pessoal, 2012. http://www.richardlord.net/blog/why-use-an-entity-framework. Último acesso em 20/01/2016.

Martin (2007) Adam M. S. Martin. Entity Systems are the Future of MMOG Develpment. Blog pessoal, 2007. http://t-machine.org/index.php/2007/09/03/entity-systems-are-the-future-of-mmog-development-part-1/. Último acesso em 20/01/2016.

Moore & Novak (2010) Michael E. Moore & Jeannie Novak. Game Industry Career Guide. Delmar: Cengage Learning, 2010.

Morris (2015) Alex Morris. How The Rise of Indie Games Has Revitalized the Video Game Industry. Em allBusiness, 2015. Mídia digital online. http://www.allbusiness.com/indie-games-video-game-industry-101485-1.html. Último acesso em 18/07/2016.

Natkin & Veja (2004) Stephane Natkin & Liliana Vega. A petri net model for computer games analysis. International Journal of Intelligent Games Simulation 3, 2004.

Neil (2012) Katharine Neil. Game Design Tools: Time to Evaluate. Em DiGRA Nordic 2012 Conference: Local and Global – Games in Culture and Society, 2012.

Oxford Dictionary (2008) Concise Oxford English Dictionary. 11ª edição. Oxford University Press, USA. Agosto, 2008.

Oxland (2004) Kevin Oxland. Gameplay and design. Addison Wesley, 2004.

Reynolds (2015) Sam Reynolds. The Rise and Rise of Kickstarter Indie Games. Em VRWorld, 2015. Mídia digital online. http://vrworld.com/2015/04/02/the-rise-and-rise-of-kickstarter-indie-games/. Último acesso em 02/12/2015.

Riehle (2000) Dirk Riehle. Framework Design: A Role Modeling Approach. Ph.D. Thesis. Zürich, Switzerland, ETH Zürich, 2000. http://dirkriehle.com/computer-science/research/dissertation/diss-a4.pdf. Último acesso em 18/07/2016.

Rogers (2010) Scott Rogers. Level UP: The Guide to Great Videogame Design. Wiley Publication, 2010.

Rouse (2000) Richard Rouse III. Game Design: Theory & Practice. Wordware, Inc., 2000.

Salen & Zimmerman (2003) Katie Salen; Eric Zimmerman. Rules of Play: Game Design Fundamentals. Cambridge, Massachusetts: MIT Press, 2003.

Sicart (2008) Miguel Sicart. Defining Game Mechanics. Game Studies: The International Journal of Computer Game Research, volume 8, número 2. Dezembro 2008. http://gamestudies.org/0802/articles/sicart. Último acesso em 15/07/2016.

Sigman (2005) Taylor Sigman. The Siren Song of the Paper Cutter: Tips and Tricks from the Trenches of Paper Prototyping. Gamasutra, setembro de 2005. http://www.gamasutra.com/view/feature/2403/the_siren_song_of_the_paper.php. Último acesso em 18/07/2016.

Smith et al. (2010) Adam M. Smith; Mark J. Nelson; Michael Mateas. LUDOCORE: A logical game engine for modeling videogames. CIG 2010 IEEE Symposium, 91–98, 2010.

https://adamsmith.as/papers/ieeecig10_ludocore.pdf. Último acesso em 18/07/2016.

Tavinor (2009) Grant Tavinor. The Art of Videogames. Wiley-Blackwell, 2009.

Thorn (2013) Alan Thorn. Game Development Principles. Cengage Learning, maio de 2013.

Thorsen (2007) Tor Thorsen. Game Developers Conference 2007: Cliffy B disassembles Gears. GameSpot, 2007. Mídia digital online. http://www.gamespot.com/articles/gdc-07-cliffy-b-disassembles-gears-mentions-sequel/1100-6167213/. Último acesso em 18/07/2016.

Zagal et al. (2005) José P. Zagal, Michael Mateas, Clara Fernández-Vara, Brian Hochhalter, Nolan Lichti. Towards an Ontological Language for Game Analysis. In: S. de Castell & J. Jenson (Eds.), Changing Views: Worlds in Play, Selected Papers of DIGRA 2005. Vancouver, Canada, 2005.