Capítulo 1

Geração Procedural de Conteúdo Aplicada a Jogos Digitais Educacionais

Wendell Oliveira de Araújo, Eduardo Aranha e Charles Madeira

Abstract

Digital educational games have proved to be a great area of research that has been increasing over the years in the international scene. This fact has been due to the potential that the games have of fun, immersion and stimulation to the learning of natural and personalized way. However, one of the major challenges in this area is the costs involved and the difficulty in balancing entertainment with learning. In this sense, the procedural generation of content has been presented as an area that can aid the development of educational games. The procedural content generation (PCG) deals with the automatic creation of content through algorithms. Thus, this chapter will present techniques and a strategy for applying PCG in educational games.

Resumo

Jogos digitais educacionais têm se mostrado uma grande área de pesquisa que vem aumentando com o decorrer dos anos em todo o cenário internacional. Esse fato tem se dado pelo potencial que os jogos têm de diversão, imersão e estímulo à aprendizagem de maneira natural e personalizada. Entretanto, um dos grandes desafios encontrados nessa área são os custos envolvidos e a dificuldade em balancear o entretenimento com o aprendizado. Nesse sentido, a geração procedural de conteúdo tem se apresentado como uma área que pode auxiliar o desenvolvimento de jogos educacionais. A geração procedural de conteúdo (PCG) trata da criação automática de conteúdo por meio de algoritmos. Assim, este capítulo apresentará técnicas e uma estratégia para aplicação da PCG em jogos educacionais.

1.1. Indústria de jogos digitais e seu potencial para melhoria da educação

A indústria de jogos digitais vem crescendo consideravelmente nos últimos anos e, segundo a consultoria Newzoo (2018), continuará crescendo na ordem de 11% ao ano até atingir o faturamento de US$ 180 bilhões em 2021 (ver Figura 1), o que consiste em um montante maior que a soma das indústrias do cinema e da música. Essa evolução acelerada da indústria de jogos também tem impulsionado os investimentos em pesquisas que buscam novas formas de fazer uso dos jogos nas mais diversas áreas tais como educação, saúde, simulação, entre outras. Devido ao grande número de pesquisas já efetuadas tendo como objetivo o uso de jogos na educação, é possível constatar o grande potencial que os jogos têm para contribuir com a melhoria da área. Savi e Ulbricht (2008) apresentam diversos benefícios que os jogos digitais educacionais podem promover, como socialização, engajamento, desenvolvimento de habilidades cognitivas, aprendizado por descoberta, facilitação do aprendizado, entre outros. Dessa forma, os jogos educacionais se mostram aptos a serem utilizados como uma ferramenta de ensino. Portanto, eles têm ganhado mais espaço nas escolas, tornando as aulas mais lúdicas e agradáveis para os alunos, sendo uma ótima forma de motivá-los e estimulá-los na superação dos desafios impostos nos ambientes virtuais simulados [Lara 2003].

Figura 1.1 Evolução do faturamento da indústria de jogos digitais durante no período de 2012 a 2021. De 2012 a 2017, os valores são reais. De 2018 a 2021, os valores correspondem a previsões baseadas em uma taxa de crescimento de 11% ao ano.

Sendo assim, os jogos se constituem como uma ferramenta importante no processo de ensino-aprendizagem dos estudantes. Contudo, para serem efetivos, os jogos precisam ter objetivos de ensino-aprendizagem bem definidos que proporcionem boas estratégias para a prática de conteúdos e o desenvolvimento de habilidades [Savi e Ulbricht 2008].

1.2. Desafios para o desenvolvimento de jogos digitais, especialmente os educacionais

Um dos grandes desafios para o desenvolvimento de jogos digitais são os custos envolvidos com a sua produção. A produção de um jogo é uma tarefa multidisciplinar que demanda o trabalho conjunto de várias áreas de conhecimento (programação, artes, design, música) para poder conseguir entregar um produto de boa qualidade de acordo com os padrões atuais de mercado. Os custos envolvidos têm uma relação direta com a complexidade do jogo que está em produção, influenciando principalmente, e de forma considerável, a etapa de produção de conteúdo (partes que envolvem um jogo tais como cenários, objetos, personagens, narrativas, texturas, músicas, etc.). Com isso, muitas vezes os desenvolvedores são levados a desistir do projeto durante o seu desenvolvimento, enquanto que alguns outros, que trabalham com equipes menores, tentam buscar alternativas para reduzir os custos de produção. Apesar da forte taxa de crescimento da indústria, ainda há uma carência de ferramentas que auxiliem os designers e artistas durante o processo de produção dos jogos, o que o torna bastante demorado e, consequentemente, custoso.

O "sucesso" de um jogo depende, na grande maioria das vezes, da experiência que ele proporciona aos jogadores. Muitas pesquisas apontam que a experiência proporcionada por um jogo depende muito do nível de desafio que é imposto ao jogador, adaptando-se às suas capacidades. Contudo, com o aumento do número de jogadores habilidosos que conseguem superar com muita facilidade os desafios propostos, os desenvolvedores passaram a se preocupar com a criação de desafios adequados para esses jogadores, visando que os mesmos não percam o interesse no jogo por não se sentirem desafiados [Bauckhage et al. 2012]. Assim, é necessário que sejam buscadas alternativas ao processo de produção padrão com o intuito de diminuir os custos, gerando rapidamente desafios adequados aos níveis dos jogadores.

Quando tratamos especificamente de jogos educacionais, além dos custos envolvidos com o processo de produção, a proposta do jogo precisa estar bem definida com os objetivos de ensino que deve ser passado, se ele deve ser usado para ensinar o aluno ou somente como uma ferramenta para prática de conteúdo, ajudando no desenvolvimento de certas habilidades [Savi e Ulbricht 2008]. Na realidade, o que geralmente acontece é que os projetos voltados ao desenvolvimento de jogos educacionais dispõem de financiamentos reduzidos, o que leva-os na maioria das vezes a dispor de equipes despreparadas que constroem produtos com foco quase que exclusivo no conteúdo a ser ensinado. Por esta razão, a experiência do jogador, que é um fator essencial do sucesso dos jogos comerciais, se torna negligenciada, fazendo com os jogos educacionais sejam reconhecidos pelos alunos como jogos simples e chatos, ou seja, que proporcionam uma experiência desinteressante.

Com o objetivo de melhor elaborar os jogos educacionais a fim de levá-los a um outro patamar, precisamos de alternativas que facilitem e reduzam os custos do processo de desenvolvimento, muito mais ainda do que no contexto dos jogos comerciais devido aos financiamentos reduzidos. Portanto, é de fundamental importância que tenhamos em mente a necessidade de gerar uma boa experiência para o jogador, proporcionando uma nova visão dentro da equipe de desenvolvimento focada no engajamento do jogador. Neste sentido, as técnicas de geração procedural de conteúdo (PCG) [Togelius *et al*. 2011]

se mostram como uma excelente alternativa pois permitem alavancar o processo de geração de conteúdo dos jogos, que é bastante custoso quando efetuado de forma manual.

As pesquisas realizadas na área de PCG têm sido, em grande parte, voltadas para o processo de geração de fases de jogos comerciais. Esse processo faz uso de técnicas tanto para criar fases completas de um jogo quanto para criar partes de fases que podem ser completadas pelos designers. Elas são, em sua maioria, voltadas para gêneros específicos de jogos, como os de plataforma, corrida, masmorra e aqueles envolvendo puzzles físicos. Assim, essas técnicas vêm sendo aplicadas em jogos já consagrados no mercado como *Super Mario Bros* (Nintendo), *Angry Birds* (Rovio), entre outros. No entanto, percebe-se que a aplicação de técnicas de PCG ainda se apresenta em estágio embrionário no contexto da área de jogos educacionais [Rodrigues 2017]. Assim, há uma grande carência de pesquisas envolvendo PCG e jogos educacionais, sistematizando sua aplicação e considerando um aspecto adicional que é o objetivo pedagógico.

1.3. Geração procedural de conteúdo

De acordo com Togelius *et al*. (2011), a definição de geração procedural de conteúdo (PCG) é a criação algorítmica de conteúdo do jogo com entrada de usuários limitada ou indireta. Em outras palavras, a PCG pode criar um conteúdo com o auxílio de um usuário ou por conta própria.

Os primórdios do uso da PCG se deu por volta do ano de 1980, uma vez que os jogos precisavam evoluir e a capacidade de armazenamento das máquinas era limitada. Assim, *Elite* [Brabel e Bell 1984] e *Rogue* [Toy et al 1980] foram alguns dos primeiros jogos a utilizarem PCG (ver Figura 1.2). *Elite* para gerar proceduralmente 8 galáxias, cada uma delas com 256 planetas com propriedades únicas, e *Rogue* com níveis sendo gerados aleatoriamente a cada início de partida.

Com o passar dos anos, os produtores de jogos comerciais também começaram a fazer o uso da PCG em seus jogos como *Diablo* [Blizzard 1996], gerando mapas e alocando itens e inimigos no mapa de forma procedural. Outros exemplos são *Spore* [Maxis 2008], no qual o jogador desenvolvia novos personagens e as animações dos mesmos eram criadas usando técnicas de animação procedural, *Minecraft* [Persson 2009], sendo um dos mais populares por gerar o mundo inteiro e o conteúdo deste mundo, *Spelunky* [Yu 2008], para gerar automaticamente variações dos níveis do jogo, também sendo um jogo bastante utilizado em pesquisas na área.

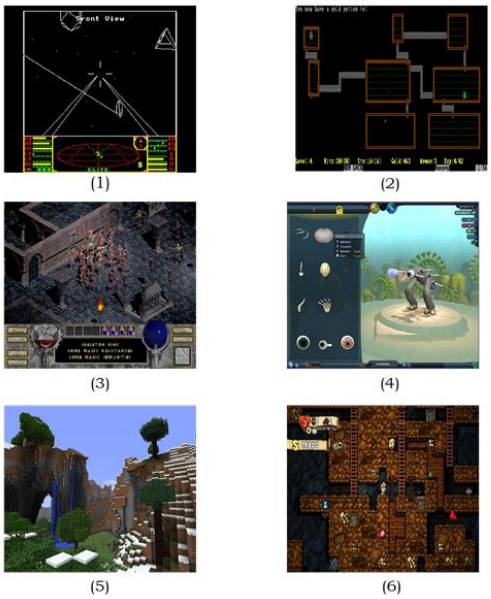

Figura 1.2. Exemplos de jogos que fizeram uso de PCG no processo de desenvolvimento: (1) Elite (2) Rogue (3) Diablo (4) Spore (5) Minecraft (6) Spelunky.

O uso de PCG oferece diversos benefícios como a redução do tempo de produção do jogo, e consequentemente dos custos, visto que o computador é capaz de gerar o conteúdo de maneira rápida e automática. Por outro lado, a facilidade da PCG gerar novos conteúdos aumenta o fator de *replay* do jogo (alonga o ciclo de vida pois o jogador é propenso a retornar a jogar). Além disso, no contexto dos computadores da atualidade não existe mais uma forte restrição quanto ao poder de processamento e a capacidade de memória conforme existia nos anos 80, o que era um dos principais entraves da PCG.

Quanto às limitações que ainda persistem da PCG, temos o fato de que o conteúdo gerado é, na maioria dos casos, específico, como por exemplo no caso de um gerador de fases. Uma dificuldade da sua aplicação consiste na falta de controle sobre o conteúdo gerado, não permitindo garantir que o mesmo sempre tenha solução ou utilidade dentro do jogo. De maneira similar, também é difícil avaliar a qualidade do conteúdo gerado pelo algoritmo, já que são necessários estudos mais elaborados com usuários e que os interesses variam de jogador para jogador.

1.3.1. Tipos de conteúdo gerado

A PCG pode ser destacada da seguinte forma: os termos geração e procedural servem para indicar que estão sendo utilizados algoritmos e procedimentos para criar algo; O termo conteúdo passa a se referir às partes de um jogo que podem ser geradas, como mapas, regras do jogo, texturas, objetos, personagens, fases, entre outros. A partir dessas definições podemos considerar como exemplos de PCG:

- Uma ferramenta de software que cria masmorras (*dungeons*) para um jogo como *The Legend of Zelda* sem qualquer interferência humana, sendo um novo nível criado a cada vez que a ferramenta é executada;
- Um sistema que cria novas armas em um jogo de tiro espacial em resposta ao que o coletivo de jogadores faz, de modo que as armas apresentadas são versões evoluídas de armas que outros jogadores acharam divertidas para usar;

- Uma ferramenta de design gráfico que permite que um usuário crie mapas para um jogo de estratégia, enquanto avalia continuamente o mapa projetado para suas propriedades de jogabilidade e sugere melhorias no mapa para torná-lo mais equilibrado e mais interessante.

Por outro lado, não pode ser considerado como exemplos de PCG:

- Um editor de mapas para um jogo de estratégia que simplesmente permita ao usuário colocar e remover itens;
- Um jogador artificial para um jogo de tabuleiro.

Na área acadêmica, vários pesquisadores apresentam suas perspectivas e métodos para enfrentar os problemas que a PCG possui, como garantir que o conteúdo gerado seja jogável, controlar a geração do algoritmo, entre outros. Assim, resultando em uma série de novos métodos, variações e combinações já desenvolvidas, muitos ainda precisando dar continuidade às pesquisas para poderem ser aplicados no contexto de jogos reais.

A PCG dispõe de diversas formas nas quais conteúdos podem ser gerados, apresentando bastante espaço para novas pesquisas. Normalmente, o desenvolvimento ocorre para um tipo de conteúdo específico, sendo alguns exemplos deles citados a seguir:

- **Níveis/fases:** PCGs focados em fases são aqueles que se preocupam em criar fases completas de jogos, desde a adição de plataformas, inimigos, bônus para o jogador e também um dos tipos de conteúdo mais utilizados nas pesquisas. Um exemplo deste tipo de conteúdo é o trabalho de [Moghadam e Rafsanjani 2017];
- **Mapas/terrenos:** São focados somente em criar as estruturas dos mapas, muito comum em jogos 3D para criação das montanhas, ambientes de navegação conforme pode ser visto no trabalho de [Raffe et al. 2015];
- **Quest/história:** Bastante comum em jogos de RPG para criação de missões secundárias para o jogo, que ajudam o personagem a aumentar o nível de experiência. Um exemplo pode ser visto no trabalho de [Kybartas e Verbrugge 2014];
- **Elementos de Cenário:** Partes de estruturas que compõem um cenário. No trabalho de **[**Stephenson e Renz 2016**]** são geradas estruturas complexas para o jogo *Angry Birds;*
- **Armas:** Mais comum em jogos de tiro para o balanceamento das armas e criação de armas com um visual que agrada ao jogador. Um exemplo deste de tipo de conteúdo pode ser visto no trabalho de [Gravina e Loiacono 2015];
- **NPCs:** Normalmente são criados agentes inteligentes para interação com o jogador e até mesmo inimigos para o mesmo, um pouco sobre isso pode ser visto no trabalho de [Anand e Wei 2016];
- **Puzzles:** São focados na criação de quebra-cabeças e enigmas que precisam ser resolvidos para que as fases sejam concluídas. Um exemplo pode ser visto no trabalho de [Shaker et al 2013];
- **Jogos de tabuleiro:** Focados na criação de tabuleiros para jogos como Banco Imobiliário [Halim 2014], normalmente jogos que envolvem o uso de dados e movimentação de um tabuleiro;

1.3.2. Técnicas existentes

Entre as técnicas de PCG mais comuns temos a *search-based PCG* (SBPCG) e a *grammar-based PCG*. A SBPCG faz uso de técnicas de computação evolutiva, algoritmos

de busca e métodos similares para se buscar um conteúdo desejado. A *grammar-based PCG* pode ser utilizada para geração de plantas, geração de fases de ação e aventura entre outras aplicações. Também podemos destacar outros métodos como geração construtiva, que busca um tipo de conteúdo específico que é a geração de masmorras *(dungeons)* e fases. Entre esses podemos destacar, o *space partitioning, grammar-based dungeon generation, cellular automata,* entre outros [Shaker et at 2016]. *Answer set programming (ASP)* segue uma abordagem para programação lógica similar a *Prolog*, sendo utilizado também em gerações de labirintos e fases, como pode ser visto no trabalho de [Smith et at 2016], bem como *Noises, fractals,* agentes para geração de paisagens, terrenos e texturas, através de métodos clássicos como o *perlin-noise* [Korn et at 2017].

1.3.2.1 Gramática

De acordo com [Prusinkiewicz e Lindenmayer 1990], as gramáticas envolvem uma técnica de reescrita, na qual objetos complexos (fases de um jogo, por exemplo) são gerados substituindo sucessivamente partes de um objeto inicial simples (estrutural geral da fases) usando um conjunto de regras ou produções de reescrita. Portanto, uma gramática é criada como um alfabeto com diversas palavras ou símbolos e um conjunto de regras que regem que palavras ou símbolos podem ser substituídos. Como exemplo podemos considerar uma simples gramática, conforme citada abaixo:

$A \rightarrow AB$

$B \rightarrow b$

Dada essa gramática pode ser gerada uma string maior a cada vez que a gramática aplicar a regra. Levando em consideração que iniciamos uma string com a letra A teríamos: A, AB, ABb, ABbb, ABbbb, etc. Dessa forma, a medida que a regra da gramática vai sendo aplicada, a *string* final é modificada.

Na geração de conteúdo para jogos, a gramática tem sido aplicada de diversas formas diferentes. Em [Font et al 2016], é apresentado um gerador de fases através de algoritmos evolutivos baseados em gramática. Ele apresenta um método em dois estágios para a criação da fase, respeitando as restrições do projeto. A ideia geral dos autores é gerar uma gramática e evoluí-la através de um algoritmo genético e, em um segundo passo, essas fases são expandidas para produzir mapas completos.

Outra aplicação pode ser vista em [Smith e Treanor et al 2009]. Este trabalho, usa PCG em jogos de plataforma baseados em ritmos (considere ritmo algo que o jogador sente com as mãos), sendo um exemplo de combinação de gramáticas. A primeira gramática é usada para gerar ritmos e a outra para gerar a geometria do cenário com base nos ritmos gerados.

Métodos baseados em gramáticas também foram usados na geração de níveis na competição *Mario AI Competition*, na qual os autores de [Shaker e Nicolau et al. 2012] evoluíam os níveis jogáveis usando gramática evolutiva. Ela permite uma codificação simples das restrições da fase e descrições compactas para fases grandes. Um exemplo de uma versão simplificada do design de fases pode ser observado na Figura 1.3. No qual, ao iniciarmos um *level*, ele pode ser substituído por um *chunks* ou *enemy*, e consequentemente após as demais execuções substituir os *chunk* por uma *platform* ou uma *coin*, de acordo com o que o algoritmo for escolhendo.

```
<level> ::= <chunks>  <enemy>
<chunks> ::= <chunk> |<chunk> <chunks>
<chunk> ::= gap(<x>,<y>, <w_g>,<w_before>,<w_after>)
  | platform(<x>,<y>,<w>)
  | hill(<x>,<y>,<w>)
  | cannon_hill(<x>,<y>,<h>,<w_before>,<w_after>)
  | tube_hill(<x>,<y>,<h>,<w_before>,<w_after>)
  | coin(<x>,<y>,<w_c>)
  | cannon(<x>,<y>,<h>,<w_before>,<w_after>)
  | tube(<x>,<y>,<h>,<w_before>,<w_after>)
  | <boxes>

<boxes> ::=  <box_type> (<x>,<y>)^2 | ...
  | <box_type> (<x>,<y>)^6

<box_type> ::= block_coin | block_powerup
  | rock_coin  | rock_empty

<enemy> ::= (koopa | goompa)(<x>)^2 | ...
  | (koopa | goompa)(<x>)^10
<x> :: = [5..95] <y> ::= [3..5]
```

Figura 1.3. Versão simplificada da gramática final para geração de nível para Super Mario Bros [Shaker et al 2012].

A gramática também pode ser utilizada para gerar diversos tipos de estruturas como grafos, *tilemaps*, formas em 2D ou em 3D. Assim, como abordado por [Shaker et al. 2016], pode ser utilizada uma combinação de gramática de grafos para gerar missões e de *tilemaps* para gerar um nível de jogo.

1.3.2.2 Abordagem baseada em busca

A computação evolutiva é muito utilizada em muitas soluções em PCG, sendo mais comum a *SBPCG* com a geração baseada em busca e também a gramática evolutiva quando fazem uma combinação geração de conteúdo com gramática e algoritmos evolutivos. Os algoritmos evolutivos podem ser definidos como um procedimento iterativo de busca (otimização) baseado nos mecanismos evolutivos biológicos. Assim, o algoritmo é utilizado para procurar conteúdo com as qualidades desejadas pelo desenvolvedor. Uma metáfora básica utilizada com os algoritmos evolutivos é que existe uma solução suficientemente boa para o problema dentro de algum espaço de soluções, e se continuarmos repetindo e aprimorando uma ou várias soluções possíveis, mantendo as alterações que compõem a(s) melhor(es) solução(ões) e descartando aqueles que são prejudiciais, eventualmente chegaremos a solução desejada [Shaker 2016]. O mesmo, apresenta os principais componentes da abordagem baseada em busca para resolver um problema de geração de conteúdo, sendo eles:

1. **Um algoritmo de busca:** É o principal mecanismo de um método baseado em busca. Algoritmos evolutivos relativamente simples funcionam bem, porém em casos específicos são necessários algoritmos mais sofisticados quando se quer levar em consideração alguma restrição ou algoritmos que são especializados em um conteúdo específico.

2. **Uma representação de conteúdo:** é o que será gerado, por exemplo, missões, fases, inimigos etc. A representação de conteúdo pode ser qualquer coisa, uma matriz de números reais, uma *string*. A representação de conteúdo define qual conteúdo pode ser gerado e determina se a busca é possível.
3. **Uma ou mais funções de avaliação:** Uma função de avaliação é uma função de uma parte individual de conteúdo para um número que indica a qualidade do que foi gerado. A saída de uma função de avaliação pode indicar, a jogabilidade de um nível, a complexidade de uma busca ou o apelo estético de um inimigo. Criar uma função de avaliação que indique com segurança a qualidade do jogo é uma das tarefas mais difíceis na criação de uma *SBPCG*.

A *SBPCG* foi aplicada em jogos de diferentes formas. [Loiacono et al. 2015] desenvolveu uma ferramenta de geração de espadas, o algoritmo evolucionário utilizado para gerar os parâmetros das espadas é uma versão com poucas modificações de um algoritmo genético simples, sendo o processo de seleção feito pelo usuário.

[Ferreira e Toledo 2014] fizeram uma abordagem baseada em busca para a geração de fases para o jogo *Angry Birds.* Eles usam algoritmo genético e com essa abordagem conseguiram gerar fases estáveis para o jogo.

[Dahlskog e Togelius 2014] também trabalharam com um gerador de fases, porém para jogos de plataforma. Eles fizeram o uso de micro-padrões que são "fatias" de fases de já desenvolvidas por humanos e fazendo busca por meso-padrões que são abstrações de padrões comuns vistos nas mesmas fases. Assim, esse método conseguiu gerar fases similares em estilo aos níveis com os padrões originais que foram extraídos.

1.4. Aplicação da PCG para jogos educacionais

A utilização de PCG em jogos educacionais ainda é bastante escassa, dificilmente se encontra trabalhos que faça a relação de PCG com jogos educacionais, pois grande parte desses trabalhos estão voltados para melhorias na área de inteligência artificial e no uso de jogos de entretenimento. Contudo, PCG tem muito a acrescentar para auxiliar no uso desse tipo de jogo, podendo ajudar com a adaptação do conteúdo visto pelo jogador, tornando mais fácil ou mais difícil de acordo com o desempenho do mesmo. Outra contribuição, são as possibilidades de tornar o jogo "infinito" permitindo assim a prática constante e como os conteúdos são criados proceduralmente dificilmente o aluno repetiria o mesmo conteúdo aumentando assim a variedade de conteúdos e sempre apresentando novas experiências ao jogador.

1.4.1. Desafios

Os jogos educacionais precisam ser bem definidos no momento que está sendo produzido. Uma das grandes dificuldades em produzir esses tipos de jogos está na dificuldade em balancear o entretenimento com o aprendizado que o jogo deve apresentar para o aluno. Apesar do aluno estar jogando, o jogo deve contribuir com o aprendizado do mesmo e realizar este procedimento de maneira automatizada se torna um pouco complicado. Na utilização da PCG quando voltada para os jogos educacionais é necessário ter bastante cuidado no controle do que está sendo gerado, pois caso a PCG não esteja bem planejada e definida, ela pode produzir coisas muito difíceis para o aluno ou muito fáceis e por vezes criar coisas impossíveis de serem resolvidas.

1.4.2. Estratégia

1.4.2.1. Geração em etapas

Uma das possibilidades para melhorar o uso da PCG em jogos educacionais é dividir sua criação em etapas, assim como em outros trabalhos voltados para o entretenimento, como o trabalho de [Dormans 2010]. Ele propõe uma abordagem de PCG baseada em gramática generativa para jogos de aventura com calabouços, no qual o mesmo divide a geração em duas etapas, desenvolvendo inicialmente as missões que são divididas por salas e em seguida com base nessas missões o mesmo criar uma sala para cada missão que é gerada disposta em formato de grafos. Assim, dividindo todo mapa gerado em várias missões simples.

Uma proposta que pode ser utilizada para jogos educacionais seria dividir a geração em três etapas em caso de cenários, sendo elas:

1. **Estrutura:** A geração da estrutura base da fase envolvendo chãos e paredes.
2. **Objetivos:** A inserção do objetivo pedagógico que a fase pretende abordar.
3. **Elementos Adicionais:** A adição de elementos adicionais da fase como inimigos e elementos decorativos.

O intuito dessa geração por etapas busca tornar mais livre a inserção do objetivo na fase do jogo, como somente a estrutura base do cenário é criada inicialmente todo o espaço disponível e trafegável pelo jogador pode ser utilizado para colocar o objetivo, e com um mesmo gerado a cena pode ser completada com o restante dos elementos visuais e objetos que possam atrapalhar o jogador de concluir o objetivo. Assim, priorizamos que a fase sempre tenha um objetivo para ser realizado. Uma visão geral dessa divisão em etapas pode ser vista na Figura 1.4.

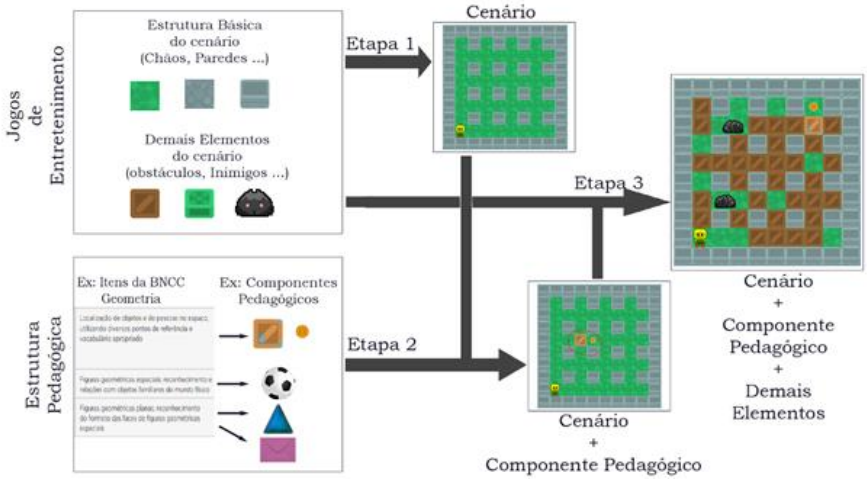

Figura 1.4. Visão geral da abordagem dividida em três etapas.

1.4.2.1.1. Modelo de jogos de entretenimento

Os jogos de entretenimento como o próprio nome sugere tem o objetivo de entreter os jogadores. A ideia por trás da abordagem visa trazer a união de mecânicas de jogos clássicos e consagrados no mercado com aspectos pedagógicos para melhorar o interesse dos estudantes pelos jogos educacionais, pois os mesmos muitas vezes são taxados como jogos desmotivantes.

Assim, estes jogos podem servir como inspiração para as gerações de fases tal como fazer o jogador passar mais tempo no mesmo. Dessa forma, propomos uma geração incremental das fases, logo para construção da mesma é feita uma divisão dos objetos que compõem o cenário por completo para assim formarmos a estrutura básica do cenário e a adição dos demais elementos do cenário.

A estrutura básica do cenário é composta pelos objetos que compõem o chão e paredes (se houver), sendo assim para gerar uma cena no qual o jogador possa se mover livremente e os objetos que compõem essa estrutura não podem fazer interação com o personagem. O objetivo dessa estrutura é criar a base para o nosso cenário e poder usar todo o espaço disponível para a adição do objetivo do jogo. Na Figura 1.4 temos o exemplo de uma estrutura base do cenário atual. Com isso temos disponível toda a área trafegável para a adição de objetivos para o jogador.

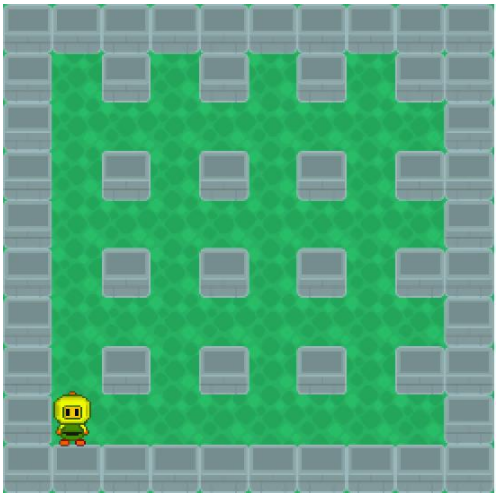

Figura 1.5. Estrutura básica do cenário do jogo.

Os demais elementos do cenário que são adicionados na última etapa da geração são elementos que podem servir incrementar o cenário como objetos de decoração, inimigos (se houver) para atrapalhar o jogador e que estes inimigos não façam parte do objetivo do jogo. Assim, com a finalização de todas as etapas a representação da Figura 1.5 poderia ficar conforme a Figura 1.6 sendo um possível cenário após toda a geração.

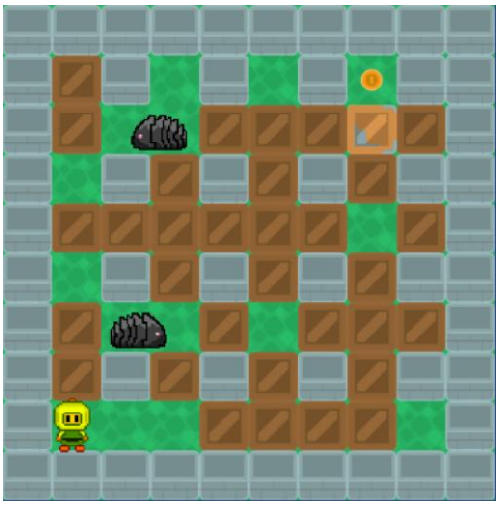

Figura 1.6. Cenário completo do jogo.

1.4.2.1.2. Estrutura pedagógica

A estrutura pedagógica é composta com base nos objetivos pedagógicos, competências e habilidades curriculares, que o desenvolvedor pretende atender. Assim, o mesmo deve planejar um objetivo para o jogo que cumpra algum requisito pedagógico. Como referência de competências e habilidades o desenvolvedor pode utilizar a Base Nacional Comum Curricular (BNCC) [Ministério da educação 2017] que apresenta as competências que precisam ser aprendidas durante a educação básica. Outro documento que pode ser levado como base é o da Sociedade Brasileira de Computação (SBC) [SBC 2017] que traz referenciais para formação de computação na educação básica.

Com essas informações o desenvolvedor pode montar mecânicas e objetivos que cumpram esses requisitos e assim montar um componente pedagógico para ser adicionado ao jogo como um objetivo a ser realizado.

1.4.2.1.3. Geração da estrutura do jogo

Uma vez especificado o modelo de jogo, a primeira etapa da PCG é responsável por definir o mapeamento inicial da fase, definindo toda a área trafegável pelo personagem durante a fase, um exemplo da mesma pode ser observado na Figura 1.4. O objetivo principal desta etapa é montar a base inicial do cenário e passar a informação para as próximas etapas dos locais disponíveis para adição de componentes pedagógicos e demais elementos que formarão o cenário do jogo.

Portanto, essa etapa irá utilizar os componentes do modelo de jogo utilizados para montar a estrutura básica do cenário, chãos e paredes, o personagem principal e outros elementos obrigatórios no cenário e com alguma função específica e importante para o jogo. Uma gramática precisa ser especificada para gerar essa estrutura básica do jogo.

1.4.2.1.4. Geração dos componentes pedagógicos

Com as áreas trafegáveis definidas, próximo passo é o de inserir componentes pedagógicos, uma vez que a fase a ser gerada tem obrigatoriamente um caráter educacional. Para isso, verifica-se as habilidades selecionadas no modelo pedagógico e os chamados componentes pedagógicos que conseguem trabalhar essas habilidades no jogo.

Supondo que foi pedido para que fosse feito um objetivo pedagógico para contagem crescente dos números, então foi desenvolvido um componente pedagógico no qual o objetivo é coletar os números no cenário em ordem crescente. Nesse caso, ação que foi utilizada foi a de coletar objetos no cenário combinado com uma competência a ser praticada pelo aluno se tornando assim um objetivo para poder vencer a fase. Levando em consideração a base estrutural da Figura 1.5, nesta etapa a geração iria dispor os objetos a serem coletados no cenário nos locais disponíveis para serem colocados os objetivos, um exemplo pode ser observado na figura 1.7.

Por fim, mesmo que nesta etapa o ideal é que o objetivo seja colocado em algum lugar disponível no cenário, caso durante a geração desta etapa não encontre um local para pôr o objetivo, a mesma pode realizar modificações na estrutura básica do cenário, como ampliar os espaços disponíveis, para que o objetivo possa ser alocado. Também é possível balancear o nível de dificuldade educacional com base nas informações

disponíveis sobre o jogador. Em termos de gramática, é necessário definir as regras de inserção de cada tipo de componente pedagógico, considerando suas especificidades.

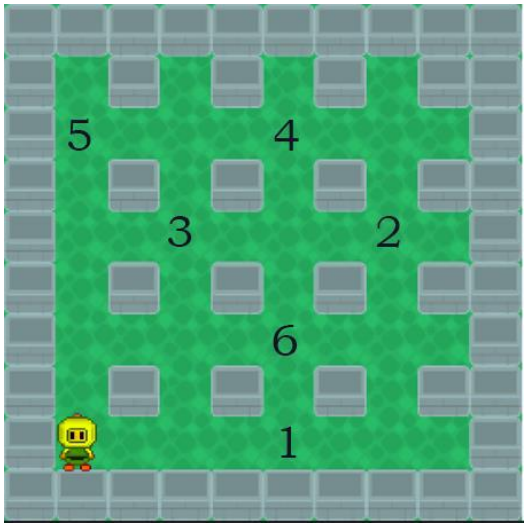

Figura 1.7. Exemplo de cenário com objetivo pedagógico.

1.4.2.1.5 Geração dos elementos adicionais

Com a alocação realizada dos componentes obrigatórios para o jogo educacional, outros componentes podem ser inseridos para se tentar garantir uma melhor atratividade para o jogo. São inseridos itens de decoração do cenário (vasos, quadros, estantes), de acordo com o modelo do jogo selecionado, além de inimigos, obstáculos, outros desafios e recompensas, características presentes em qualquer jogo de sucesso.

A PCG utilizada nesta etapa pode ainda levar em consideração as habilidades de jogo do estudante, regulando o nível de dificuldade das mecânicas de entretenimento. Esta etapa é a última, para que os elementos adicionados não interfiram na solução do objetivo pedagógico e de entretenimento proposto para que se vença a fase. Ela se torna responsável por deixar o cenário mais bonito visualmente para os jogadores, tornando assim o ambiente mais atrativo. Conforme a Figura 1.8, temos os objetos que seriam considerados como demais elementos do cenário circulados em vermelho, nesse caso as caixas e os inimigos. Com relação a gramática, é necessário definir as regras de inserção de cada tipo de componente, podendo variar de regras simples a regras mais elaboradas, envolvendo a inserção de mais de um componente em conjunto.

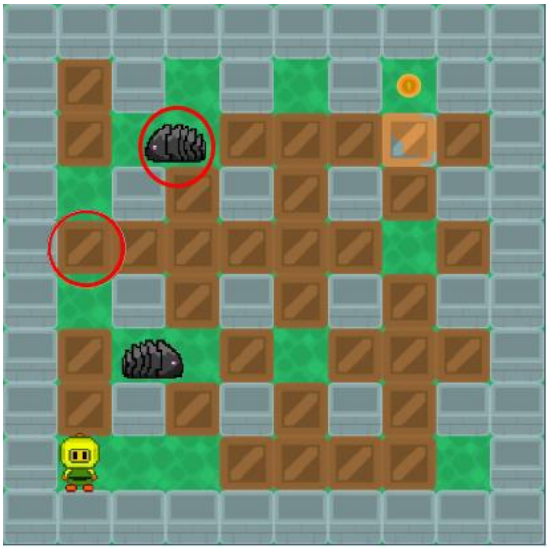

Figura 1.8. Exemplo de cenário com elementos adicionais.

1.4.3. Exemplos de aplicação

1.4.3.1. Bomberman

A saga Bomberman é um jogo desenvolvido pela *HudsonSoft* o qual teve seu primeiro título publicado em 1983. Ele é um jogo de estratégia no qual o personagem pode colocar bombas, explodindo caixas e inimigos que tentam impedi-lo de chegar ao ponto de saída. Dessa forma, para vencer é necessário que todos os inimigos sejam derrotados e que o mesmo deve chegar até o portal. A Figura 1.9 mostra uma fase do jogo.

Figura 1.9. Cenário do jogo Bomberman.

Neste exemplo, foi utilizada a habilidade matemática EF01MA14 (Identificar e nomear figuras planas (círculo, quadrado, retângulo e triângulo) em desenhos apresentados em diferentes disposições ou em contornos de faces de sólidos geométricos.) da BNCC [Ministério da Educação 2017]. Usando a estratégia da seção 1.4.2.1 foi proposto uma gramática para geração do cenário que atendesse aos objetivos do jogo *Bomberman* e a adição de um objetivo pedagógico no jogo, conforme a Figura 1.10.

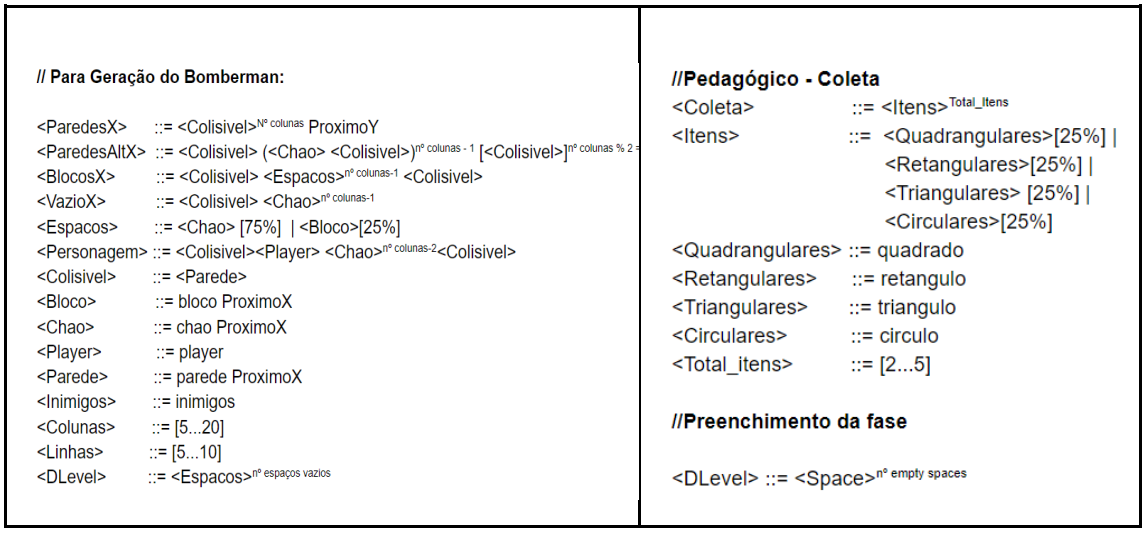

Figura 1.10. Exemplo de gramática.

Com uma gramática produzida é necessário realizar uma chamada para geração da fase conforme a estratégia da seção 1.4.2.1 fazendo uso dos elementos da gramática descrita. Assim, a chamada da gramática foi realizada em três etapas, como mostra a Figura 1.11.

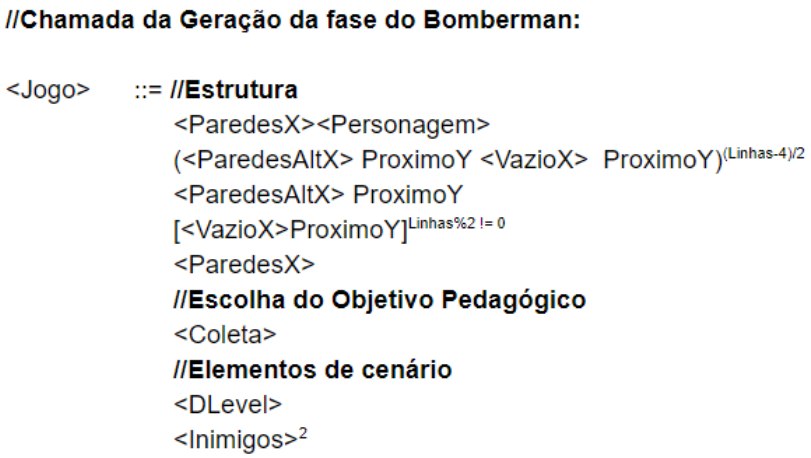

Figura 1.11. Geração em etapas da fase do Bomberman com coleta.

Ao produzir uma chamada da gramática que siga as regras da geração é possível criar fases para o jogo. Assim, resultando na aplicação de jogo de entretenimento com a adição de um objetivo pedagógico, conforme pode ser observado na Figura 1.12. Neste caso, um objeto é escolhido aleatoriamente para ser coletado pelo jogador e o mesmo deve somente coletar o item informado, caso o contrário o mesmo perde a tentativa e inicia a fase novamente.

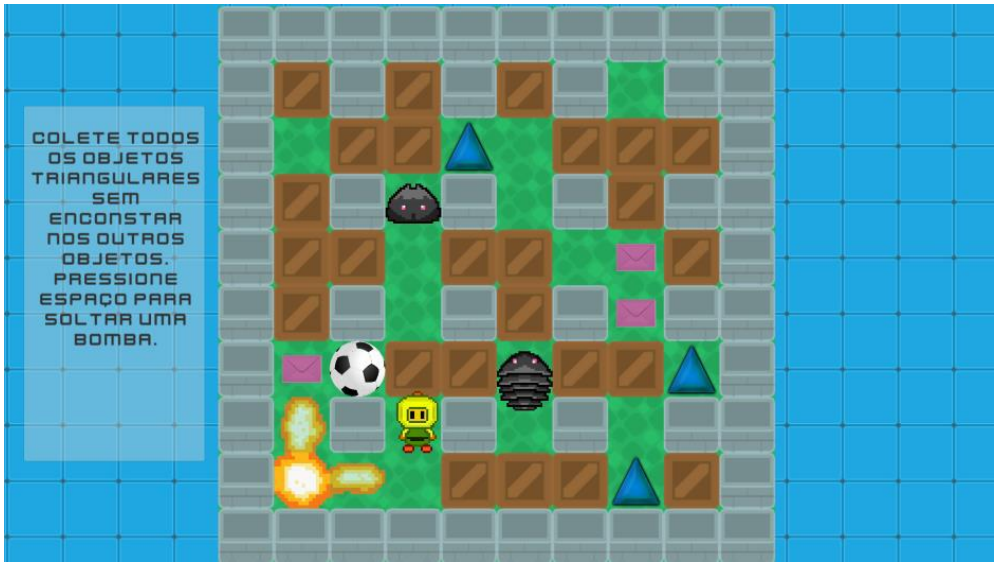

Figura 1.12. Combinação do Bomberman + Objetivo pedagógico.

1.4.3.2. Jogo de plataforma baseado em ritmos

A Figura 1.13 mostra um exemplo de um sistema de geração proposto por [Moghadam e Rafsanjani 2017], para um jogo de plataforma baseado em ritmos. O sistema proposto segue a geração separa em três etapas. Diferente do que foi visto na seção 1.4.2.1 este é focado somente no entretenimento, porém segue o mesmo princípio de geração por etapas.

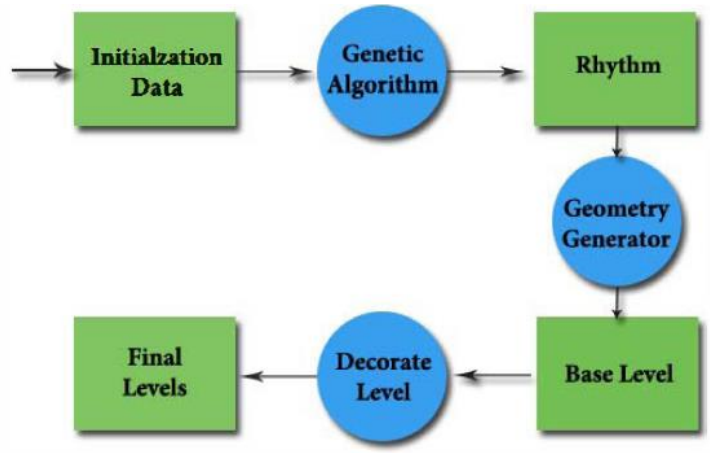

Figura 1.13. Sistema de geração de jogo de plataforma baseado em ritmo [Moghadam e Rafsanjani 2017].

Em cada etapa da geração (círculos) foi utilizado uma técnica diferente para realizar cada objetivo gerando uma informação (retângulos) a ser passada para a próxima etapa. Cada etapa foi realizada da seguinte forma.

1. **Geração dos ritmos:** É utilizado algoritmo genético para criar os ritmos do jogo (considere ritmo como uma marcação de tempo para realizar as ações, por exemplo executar pulos consecutivos no tempo certo). No qual cada ação era representada por um número (Pular = 1, Correr = 0 e Deslizar = -1). Assim, é gerado um vetor com as informações do ritmo, conforme a Figura 1.14.

2. **Geração da Geometria:** A geração das plataformas e obstáculos que o jogador enfrenta durante o jogo. Nesta etapa foi utilizada a gramática para que os elementos fossem colocados no cenário. Conforme a gramática proposta na Figura 1.15.
3. **Decoração da fase:** Nesta etapa são adicionados os elementos que não causam tanto impacto no jogo além de trazer um apelo visual. No caso deste exemplo, o autor adicionou moedas aleatoriamente com base nos locais em que acontecia cada ritmo.

0	0	1	0	0	-1	-1	0	-1	1

Figura 1.14. Exemplo de ritmo gerado [Moghadam e Rafsanjani 2017].

Run	→	Simple platform | bridge | green platform
Jump	→	Rock | angry enemy | thistle
Slide	→	Cave | tall tree hole

Figura 1.15. Exemplo de ritmo gerado [Moghadam e Rafsanjani 2017].

Assim, em cada etapa foi aplicada uma técnica diferente para geração dos elementos para finalizar uma fase. Após toda a execução é gerado uma fase, conforme a Figura 1.16 de uma fase de nível fácil.

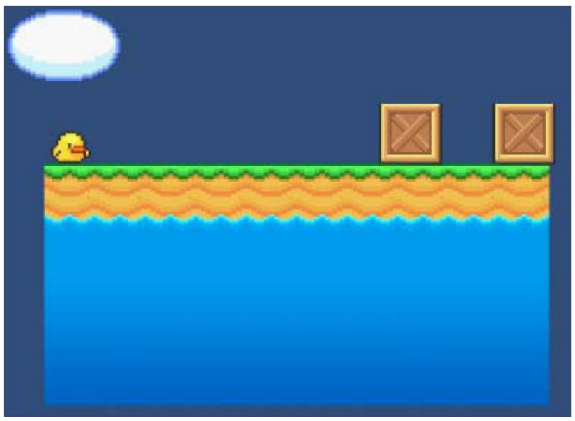

Figura 1.16. Exemplo de fase gerada do jogo [Moghadam e Rafsanjani 2017].

1.5. Considerações finais

Este capítulo buscou trazer uma abordagem geral sobre o uso de geração procedural de conteúdo aplicada em jogos digitais educacionais. Foi visto que a indústria de jogos vem crescendo no decorrer dos anos, se apresentando como um mercado bastante promissor. Os jogos educacionais têm ganhado cada vez mais espaço no que consiste a sua aplicação em sala de aula por ser uma ótima ferramenta de motivação para os alunos, tornando as aulas mais agradáveis. No entanto, um dos grandes desafios encontrados no processo de concepção desse gênero de jogos se refere aos custos de produção uma vez que, em muitos casos, conta-se apenas com um orçamento muito restrito. Além disso, também resta um outro desafio relacionado a sua coerência com os objetivos de ensino-aprendizagem, de forma que estejam bem definidos para ensinar e/ou praticar um determinado conteúdo, não deixando de lado, no entanto, de engajar o jogador. Dessa

forma, são necessárias alternativas que facilitem e reduzam os custos do processo de desenvolvimento.

A geração procedural de conteúdo (PCG) é uma ótima alternativa para contribuir com o desenvolvimento desses jogos, pois os algoritmos desenvolvidos são responsáveis por gerar conteúdos que seriam custosos caso fossem desenvolvidos manualmente. Além disso, a PCG pode trabalhar com diversos tipos de conteúdo tais como fases, agentes inteligentes, histórias, entre outros.

No presente trabalho, duas técnicas foram apresentadas para auxiliar na geração de conteúdo: baseada em gramática; e baseada em busca. Ambas são bastantes utilizadas em pesquisas que envolvem a geração de conteúdo. Neste mesmo sentido, também foi apresentada uma estratégia de geração baseada em etapas, cujo cada etapa possuía uma PCG responsável pela execução.

A PCG abre margens para o uso de adaptação de conteúdo, controlando a geração e tornando os desafios adaptados ao grau de desempenho do jogador. Assim, auxilia no processo de aprendizagem do aluno, apresentando desafios adaptados tanto às suas habilidades e competências quanto ao objetivo pedagógico que se deseja alcançar. Por fim, também pode ser utilizada para a concepção de ferramentas que auxiliem o desenvolvedor a agilizar o processo de produção do jogo.

1.6. Referências

Amato, F., Moscato, F. (2017) “Formal Procedural Content Generation in games driven by social analyses”, In: 31st International Conference on Advanced Information Networking and Applications Workshops - AINA, p. 674-679.

Anand, B., Wei, W. H. (2016) “TIṬAL – Asynchronous multiplayer shooter with procedurally generated maps”, In: Entertainment Computing, Vol. 16, pp. 81-93.

Dahlskog, S., Togelius, J. (2014) “Procedural Content Generation Using Patterns as Objectives”, In: Esparcia-Alcázar A., Mora A. (eds) Applications of Evolutionary Computation. EvoApplications 2014. Lecture Notes in Computer Science, vol 8602. Springer, Berlin, Heidelberg.

Ferreira, L., Toledo, C. (2014) “A search-based approach for generating Angry Birds levels”, In: IEEE Conference on Computational Intelligence and Games, Dortmund, pp. 1-8.

Font, J. M., Izquierdo, R., Manrique, D., Togelius, J. (2016) “Constrained Level Generation through grammar-based evolutionary algorithms”. In: Applications of Evolutionary Computation, pp. 558-573.

Gravina, D., Loiacono, D. (2015) “Procedural Weapons Generation for Unreal Tournament III”, In: Games Entertainment Media Conference (GEM).

Halim, Z. (2014) “Evolutionary Search in the Space of Rules for Creation of New Two-Player Board Games”, CoRR abs/1406.0175.

Korn, O., Blatz, M., Rees, A., Schaal, J., Schwind, V., Gorlich, D. (2017) “Procedural Content Generation for Game Props? A Study on the Effects on User Experience”, In: Computers in Entertainment (CIE) – Theoretical and Practical Computer Applications in Entertainment, Nr. 15(2).

Kybartas, B., Verbrugge, C. “Analysis of ReGEN as a Graph-Rewriting System for Quest Generation”, In: IEEE Transactions on Computational Intelligence and AI in Games, vol. 6, no. 2, pp. 228-242.

Lara, I. C. M. (2003) “Jogando com a matemática de 5ª a 8ª série”. São Paulo: Rêspel.

Loiacono, D., Mainett, R., Pirovano, M. (2015) “Volcano: An Interactive Sword Generator”, In: Games Entertainment Media Conference (GEM).

Moghadam, A. B., Rafsanjani, M. K. (2017) “A Genetic Approach in Procedural Content Generation for Platformer Games Level Creation”, In: 2nd Conference on Swarm Intelligence and Evolutionary Computation (CSIEC), Iran.

Newzoo. (2018) “Newzoo Global Games Market Report 2018”, 2018. Disponível em: https://newzoo.com/insights/trend-reports/newzoo-global-games-market-report-2018-light-version/.

Prusinkiewicz, P., Lindenmayer, A. (1990) “The Algorithmic Beauty of Plants”, In: Springer.

Raffe, W. L., Zambetta, F., LI, X., Stanley, K. O. (2015) “Integrated Approach to Personalized Procedural Map Generation Using Evolutionary Algorithms”, In: IEEE Transactions on Computational Intelligence and AI in Games, Vol. 7, NO. 2, pp. 139-155.

Savi, R., Ulbricht, V. R. (2008) “Jogos Digitais Educacionais: Benefícios e Desafios”, In: Revista Novas Tecnologias na Educação - RENOTE, v.6, n.2.

Shaker, N., Togelius, J., Nelson, M. J. (2016) “Procedural Content Generation in Games: A Textbook and an Overview of Current Research”, In: Springer. ISBN 978-3-319-42714-0.

Shaker, M., Sarhan, M. H., Naameh, O. A., Shaker, N., Togelius, J. (2013) “Automatic generation and analysis of physics-based puzzle games”, In: IEEE Conference on Computational Intelligence in Games (CIG), Niagara Falls, ON, pp. 1-8.

Smith, G., Treanor, M. et al. (2009) “Rhythm-Based Level Generation for 2D Platformers”, In: Proceedings of the 4th International Conference on Foundations of Digital Games. ACM, pp. 175-182.

Stephenson, M., Renz, J. (2016) “Procedural generation of complex stable structures for angry birds levels”, In: IEEE Computational Intelligence and Games Conference (CIG).

Togelius, J., Kastbjerg, E., Sschedl, D., Yannakakis, G. N. (2011) “What is Procedural Content Generation?: Mario on the Borderline”, In: Proceedings of the 2nd Workshop on Procedural Content Generation in Games.

Sobre os autores

Wendell Oliveira de Araújo

Doutorando em Sistemas e Computação pelo Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte (DIMAp/UFRN), mestrado em sistemas e computação pela UFRN (2018) e pós-graduado em Desenvolvimento de Jogos Digitais pela Faculdade Estácio tem experiência em Desenvolvimento de jogos (2017), atuou durante a graduação em diversos projetos de pesquisa na área de jogos pela GAMEDU. Atuou como tutor no curso técnico do Instituto Metrópole Digital no curso de Programação de jogos digitais no ano de 2016, bem como experiência no ensino de programação de jogos para estudantes do ensino fundamental através dos Clubes de Programação oferecidos pela GAMEDU em escolas privadas nos anos de 2016 e 2017. Atualmente realiza pesquisas principalmente no uso de games para melhoria da educação básica. Atua em parceria com outros centros de pesquisa, escolas do ensino básico e empresas de jogos digitais e de educação.

Eduardo Henrique da Silva Aranha

Doutor (2009) em Ciência da Computação pelo Centro de Informática da Universidade Federal de Pernambuco (CIn/UFPE), tem experiência como professor no ensino superior desde 2002, bem como experiência na gerência e desenvolvimento de software desde 1997. Atualmente é professor do Departamento de Informática e Matemática Aplicada da UFRN e coordena o Laboratório de Pesquisa em Games e Educação do Instituto Metrópole Digital da UFRN. Atualmente realiza pesquisas principalmente no uso de games para melhoria da educação básica e em engenharia de software experimental. Atua em parceria com outros centros de pesquisa, escolas do ensino básico e empresas de jogos digitais e de educação.

Charles Andrye Galvão Madeira

Possui graduação em Ciência da Computação pela Universidade Federal do Rio Grande do Norte (1998), mestrado em Ciência da Computação pela Universidade Federal de Pernambuco (2001), doutorado (2007) e pós-doutorado (2008) em Informática pela Université Paris 6 (Pierre et Marie Curie). Foi Engenheiro de Pesquisa & Desenvolvimento da empresa MASA Group (2008-2012), atuando no desenvolvimento de sistemas inteligentes para as áreas de defesa e segurança (simulação militar e gestão de crises). Atualmente é Professor Adjunto da Universidade Federal do Rio Grande do Norte, onde coordenou a criação de cursos nos níveis técnico e de graduação na área de jogos digitais, e de pós-graduação em tecnologias educacionais no Instituto Metrópole Digital (IMD). Tem atuado em diversos projetos de pesquisa e extensão nestas áreas, orientando dezenas de alunos de graduação e pós-graduação. Atualmente orienta mais de uma dezena de alunos de pós-graduação nos temas dos jogos digitais, tecnologias educacionais e inteligência artificial. Foi gerente adjunto da Inova Metrópole (2014-2015), a incubadora de empresas de base tecnológica do IMD. É coordenador do Programa de Pós-graduação em Inovação em Tecnologias Educacionais, assim como do Laboratório de Experimentação de Jogos Digitais e Realidade Virtual do IMD. Tem experiência na área de Ciência da Computação, com ênfase em Sistemas de Computação, atuando principalmente nos seguintes temas: Jogos Digitais, Jogos Sérios, Gamificação, Tecnologias Educacionais, Aprendizado de Máquina, Inovação e Empreendedorismo.