Vicente Corrêa da Silva Neto

Uma plataforma de jogos JRPG destinada à educação com entretenimento

Dissertação de Mestrado

Departamento de Informática
Programa de Pós–graduação em Informática

Rio de Janeiro
Setembro de 2016

Pontifícia Universidade Católica
do Rio de Janeiro

Vicente Corrêa da Silva Neto

Uma plataforma de jogos JRPG destinada à educação com entretenimento

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática do Departamento de Informática da PUC-Rio.

Orientador: Prof. Waldemar Celes Filho

Rio de Janeiro
Setembro de 2016

Pontifícia Universidade Católica
do Rio de Janeiro

Vicente Corrêa da Silva Neto

Uma plataforma de jogos JRPG destinada à educação com entretenimento

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Waldemar Celes Filho
Orientador
Departamento de Informática — PUC-Rio

Prof. Bruno Feijó
Departamento de Informática — PUC-Rio

Prof. Esteban Clua
UFF

Prof. Márcio da Silveira Carvalho
Coordenador Setorial do Centro Técnico Científico — PUC-Rio

Rio de Janeiro, 22 de setembro de 2016

Vicente Corrêa da Silva Neto

Possui graduação em Sistemas de Informação pela Pontifícia Universidade Católica do Rio de Janeiro (2012). É aluno de Mestrado da Pontifícia Universidade Católica do Rio de Janeiro e membro do instituto TECGRAF, onde desenvolve pesquisas na área de computação gráfica e engenharia de software.

Ficha Catalográfica

Neto, Vicente Corrêa da Silva

Uma plataforma de jogos JRPG destinada à educação com entretenimento / Vicente Corrêa da Silva Neto; orientador: Waldemar Celes Filho. — Rio de Janeiro : PUC-Rio, Departamento de Informática, 2016.

82 f: il. (color.); 29,7 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2016.

Inclui bibliografia.

1. Informática – Teses. 2. Jogos Educativos;. 3. Entretenimento;. 4. Jogos;. 5. RPG;. 6. Matemática.. I. Filho, Waldemar Celes. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Agradecimentos

Ao meu orientador, Prof. Dr. Waldemar Celes Filho, por vir contribuindo com este trabalho mesmo antes de sua concepção, por ter me apoiado profissional e academicamente e pela maneira formidável com a qual gerencia nossa equipe.

Ao instituto e aos amigos do TECGRAF, por oferecerem um ambiente de trabalho agradável, produtivo e inspirador do qual temos tanto orgulho e privilégio de fazer parte.

À minha família, em especial aos meus pais, Walter Francisco Gomes e Dejanira Maria da Silva Gomes, por todo amor, fé e dedicação com que me criaram e educaram e ao meu irmão Helcio Carlos Gomes por ter comprado nossos primeiros video-games.

À minha esposa Bianca Goulart por todo o apoio, compreensão e por ser a melhor esposa do mundo.

Ao Prof. Dr. Bruno Feijó e ao Prof. Dr. Esteban Clua por aceitarem participar da comissão julgadora desta defesa de mestrado, dedicando parte de seu precioso tempo para melhoria e conclusão deste trabalho.

Resumo

Neto, Vicente Corrêa da Silva; Filho, Waldemar Celes. **Uma plataforma de jogos JRPG destinada à educação com entretenimento**. Rio de Janeiro, 2016. 82p. Dissertação de Mestrado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Neste projeto, inspirados pelas áreas de Pedagogia e Entretenimento, buscamos criar uma plataforma de desenvolvimento de jogos eletrônicos, cujo o objetivo é facilitar a criação de jogos educativos do sub-gênero JRPG (Japanese Role Playing Games), mais interessantes do que a maioria dos jogos educativos disponíveis no momento. O gênero RPG é, por definição, baseado em contação de histórias e interpretação de papéis, identificadas pela literatura como importantes ferramentas cognitivas capazes de estimular a imaginação dos estudantes, envolvê-los emocionalmente e despertar seus interesses por tópicos do currículo escolar tradicional. O sub-gênero JRPG, por sua vez, representa uma categoria especial de RPGs eletrônicos que, herda essas mesmas características educativas, mas possuem delimitações claras acerca de mecânicas de jogo e identidade artística. Tais delimitações são positivas no sentido em que funcionam como uma espécie de guia para que o desenvolvedor se oriente durante o processo de criação de jogos desta natureza.

Palavras–chave

Jogos Educativos; Entretenimento; Jogos; RPG; Matemática.

Abstract

Neto, Vicente Corrêa da Silva; Filho, Waldemar Celes (Advisor). **A JRPG game platform with the purpose of education and entertainment**. Rio de Janeiro, 2016. 82p. MSc. Dissertation — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

In this project, inspired by the fields of Pedagogy and Entertainment, we aim to develop a digital games development framework in order to facilitate the creation of educational games of the sub-genre JRPG (Japanese Role-Playing Games), more interesting than the majority of educational games available for now. The RPG genre is, by definition, based in storytelling and role-playing principles, identified by the literature as important tools that stimulates the students imagination, engage them emotionally and arouse their interests for the traditional educational program. The subgenre JRPG, in turn, represents a special category of eletronic RPGs that inherit those same educational principles, but have well defined delimitations in respect of game mechanics and artistic identity. These delimitations are positive in a sense that they work as guidelines for the development process of this kind of games.

Keywords

Edutainment; Entertaiment; Games; RPG; Mathematics.

Sumário

Lista de figuras

1
Introdução

Este trabalho consiste no desenvolvimento de uma plataforma de jogos eletrônicos destinada à criação de jogos eletrônicos educativos do sub-gênero JRPG (Japanese Role-Playing Games). Tivemos como principal motivação para a realização deste trabalho, uma série de artigos compilados no relatório "A Literature Review of Gaming in Education"[1], que expõe alguns aspectos obsoletos do sistema educacional fundamental e médio com relação à realidade dos tempos modernos, e mostra o quanto os jogos eletrônicos podem contribuir para a melhoria desse cenário.

Apesar da argumentação inicial de que os jogos eletrônicos seriam fortes candidatos a despertar o interesse dos alunos, observamos que existe uma carência muito grande no que diz respeito a qualidade dos jogos educativos disponíveis. Buscamos suprir essa carência oferecendo uma ferramenta que permita desenvolver jogos educativos mais interessantes, onde os estudantes assumem um papel ativo no processo de aprendizado e se divertem enquanto aprendem e são avaliados.

Para o desenvolvimento deste trabalho, realizamos uma extensa pesquisa em trabalhos da área de Ciência da Computação e buscamos inspiração nas áreas de Pedagogia e Entretenimento. Com base nesses estudos, indentificamos que o sub-gênero JRPG (Japanese Role-Playing Games) seria bastante adequado para a nossa proposta e optamos por ampliar um motor de jogos já desenvolvido pelo presente autor[2].

Além do aperfeiçoamento desse motor de jogos de propósito geral, desenvolvemos um meta-motor específico para a criação de jogos do sub-gênero JRPG. Este meta-motor permite incorporar mais facilmente todos os elementos identificados durante a pesquisa como fundamentais para o desenvolvimento de jogos educativos mais interessantes.

A ferramenta foi integrada a um IDE (*Integrated Development Environment*) dedicado à linguagem Lua, que além de organizar hierarquicamente os projetos, oferece funcionalidades bastantes convenientes como *syntax highlight*, auto-completação de código e, o mais importante: suporte a depuração de scripts utilizando *breakpoints*, inspeção de valores, navegação pela pilha de execução, tudo em *run-time*.

Para demonstrar a plataforma proposta, construímos um pequeno jogo como prova de conceito. Neste jogo procuramos traduzir alguns exemplos do uso de contação de histórias no ensino da matemática para o universo de jogo eletrônico.

Este trabalho é organizado da seguinte forma: No Capítulo 2, é apresentada em mais detalhes a motivação para a realização deste trabalho. No Capítulo 3, apresentamos alguns trabalhos relacionados. No Capítulo 4, apresentamos algumas ideias advindas das áreas de Entretenimento e Pedagogia que inspiraram e conduziram a realização deste trabalho: os aspectos que fazem com que bons jogos sejam inerentemente educativos, técnicas cognitivas de narrativa e contação de história que contribuem para um melhor aprendizado, o porquê de o gênero RPG ser tão compatível com esses conceitos e o porquê de optarmos por especializar nossa plataforma no sub-gênero eletrônico JRPG (Japanese Role-Playing Games). No Capítulo 5, apresentamos as decisões técnicas (i.e., adoção do motor de jogos próprio), as melhorias e funcionalidades adicionadas ao motor de jogos adotado e a arquitetura da plataforma de desenvolvimento que criamos. No Capítulo 6, apresentamos um pequeno jogo que construímos utilizando a plataforma proposta com o intuito de demonstrar seu potencial. Por fim, no Capítulo 7 apresentamos as conclusões deste trabalho e sugestões para trabalhos futuros.

2
Motivação

A principal motivação deste trabalho advém de uma série de artigos compilados no relatório "A Literature Review of Gaming in Education" [1], cujo o conteúdo é resumido a seguir em tradução livre:

O rápido avanço das tecnologias em todas as facetas da sociedade está causando mudanças significativas em como, quando e onde trabalhamos, como indivíduos, companhias, e até mesmo como nações se organizam e também como sistemas educacionais devem ser estruturados para preparar efetivamente os estudantes para a vida no século XXI.

Crianças em idade escolar de todo o planeta estão crescendo em um mundo sempre contectado, imersas em mídia e informação abundante. Discussões sobre a necessidade de reformar os sistemas de ensino com o intuito de adequá-los à esta nova realidade tem sido levantadas por políticos, educadores, pais e demais interessados[3].

Continuar oferecendo o mesmo tipo de educação enquanto o mundo muda constantemente não é uma boa estratégia. Como disse Bill Gates em 2005 durante sua participação no *National Educational Summit on HighSchools*, "Treinar a mão de obra do amanhã com as escolas de hoje é como tentar ensinar computação a crianças utilizando mainframes de 50 anos atrás. É a ferramenta errada para os dias atuais."

Uma abordagem bastante promissora para este problema consiste na adoção de jogos eletrônicos educativos como ferramenta de ensino nas escolas. Defensores dos jogos educativos citam a capacidade destes jogos de ensinarem e reforçarem habilidades importantes para as profissões do futuro tais como colaboração, resolução de problemas e comunicação. Enquanto no passado educadores se mostravam relutantes em utilizar jogos eletrônicos nas salas de aula, atualmente há um crescente interesse da classe no uso de jogos digitais como ferramenta de ensino e avaliação. Em 2005, a *Federation of American Scientists*, a *Entertainment Software Association* e a *National Science Foundation* reuniram cerca de 100 *experts* com o objetivo de pesquisar maneiras eficazes de desenvolver a próxima geração de jogos educativos. Eles descobriram que muitas das habilidades necessárias para alcançar o sucesso nos jogos eletrônicos, tais como raciocínio, planejamento, aprendizagem e competência técnica,

também eram características procuradas por potenciais empregadores[4].

A aposta que eles fazem é a de que os jogos eletrônicos são fortes candidatos para a melhoria do ensino e do aprendizado e que também oferecem maneiras mais efetivas e menos intrusivas de avaliar os alunos em comparação com o sistema tradicional.

2.1 A promessa dos jogos digitais

Os jogos digitais são considerados o maior segmento e também o de mais rápido crescimento no multi-bilionário mercado da indústria de entretenimento. O mercado global vale bilhões de dólares[5] e os custos de desenvolvimento, receita e público deste setor são comparáveis (e frequentemente ultrapassam) os da indústria cinematográfica[6]. Com 97% dos adolescentes dos Estados Unidos jogando regularmente algum tipo de jogo eletrônico não é nenhuma surpresa o aumento de interesse na aplicação desses jogos em educação. No ultimo século, houve um grande esforço nos Estados Unidos para utilizar o poder da tecnologia na melhoria do ensino[7]. Um fluxo constante de tecnologias que vão desde vitrolas, projetores, rádios e televisões até computadores, internet e afins, tem sido empregado com o intuito de aumentar o engajamento dos alunos, melhorar a eficiência da sala de aula, resolver a falta de professores e, de maneira geral, consertar o sistema educacional[8]. Muitas das previsões que se fazem sobre a habilidade dessas tecnologias de mudar a educação são sem dúvida exageradas, mas talvez haja um bom motivo para isso. Acompanhando esta tendência, os jogos eletrônicos vem levantando as esperanças e realizando promessas ousadas para o setor. Neste documento, vamos examinar evidências teóricas e empíricas dessas cinco promessas:

2.1.1 Jogos são construídos sob a estética do aprendizado.

Brincar é um importante elemento para o crescimento sadio de crianças[9], inclusive para o desenvolvimento do aprendizado. Crianças aprendem através de brincadeiras de faz de conta[10][11][12]. Uma vez que os jogos digitais oferecem uma oportunidade para brincar em ambientes simulados, esses jogos não são necessariamente uma distração para o aprendizado, pelo contrário, podem ser parte integral deste processo e do desenvolvimento intelectual[13] . Nós raciocinamos e entendemos melhor as coisas quando podemos trazer uma situação para o campo da imaginação e, desta forma, decidir como agir. Jogos apresentam um cenário ideal para este propósito, pois através da simulação de um problema temos a oportunidade de pensar, enten-

der, nos preparar e agir. Um importante elemento da experiência com jogos é que eles nos fornecem a possibilidade de praticar continuamente, pois as consequências negativas não costumam ser necessariamente associadas à falha. Ao invés disso, os erros servem como uma parte importante da experiência de aprendizado[14][15][13][16]. Isso encoraja os jogadores a melhorarem através de prática insistente, seja avançando para uma nova fase ou repetindo fases já conhecidas. Falhar sem sofrer consequências graves, ter a sensação de identidade e de escolha são elementos fundamentais inerentes aos jogos eletrônicos. Os jogos também são projetados com objetivos claros e apresentam feedback rápido sobre as ações[17]. Isto permite que os jogadores ajustem suas estratégias com o objetivo de melhorar sua performance e atingir o sucesso. A idéia de *feedback* imediato também é predominante em bons processos de avaliação. Os estudantes melhoram seus trabalhos quando recebem *feedback* construtivo[18]. Pode ser difícil para os professores transformar a performance dos estudantes em *feedback* construtivo ou planejar suas aulas de forma a incorporar desafios que requeiram ações subsequentes[19]. Este ciclo de *feedback*, no entanto, é inerente aos jogos bem projetados.

2.1.2 Jogos fornecem oportunidades de aprendizado personalizadas

A idéia de que a educação deve alcançar os estudantes "onde eles estiverem" não é nova, mas ela possui muitas variações: ensino diferenciado[20], *whole-person learning* (abordagem que leva em conta a pessoa como um todo, suas emoções e problemas pessoais)[21], instruções individualizadas e aprendizado personalizado. Aprendizado personalizado é descrito como a forma como as escolas "ajustam a educação para garantir que cada aluno atinja o mais alto padrão possível" segundo o relatório da OECD[22], que sugere a adoção desse método nas escolas através de 5 processos:

1. Conhecer os pontos fortes e fracos dos estudantes
2. Desenvolver estratégias de ensino e aprendizado com base nas necessidades dos alunos
3. Adotar alternativas curriculares
4. Organizar a escola para que seja prestativa
5. Considerar comunidade, instituição local e serviço social

No entanto, o ensino personalizado não precisa ocorrer apenas na escola. Os jogos oferecem uma oportunidade de personalizar o aprendizado para os

estudantes, alcançando pelo menos os três primeiros processos, estejam onde estiverem. Os pontos fortes e fracos dos alunos podem ser inferidos com base nas ações do jogador enquanto joga. No decorrer do jogo ELEKTRA, um projeto financiado pela *European Commission*, informações sobre as ações dos jogadores (e.g., ligar ou não o interruptor de uma lâmpada) são continuamente monitoradas para traçar um perfil atualizado de suas competências[23]. Os jogos também podem ser adaptados conforme as necessidades do aluno. Uma evolução adequada pode ser obtida utilizando diferentes níveis de dificuldade, começando pelos mais fáceis e progredindo gradualmente até os mais difíceis conforme o jogador se torna mais habilidoso.

Na sala de aula tradicional, um aluno que não aprende um conceito fundamental pode ser prejudicado futuramente no aprendizado de conceitos mais complicados. Em contraste, os jogos digitais, inerentemente forçam o jogador a dominar um conceito antes de permitir que ele avance para o próximo nível (e.g., aprender a utilizar um pulo duplo com impulso no meio do ar para cruzar o poço de lava). Os jogadores são capazes de repetir o mesmo cenário até que dominem este conceito. A mesma filosofia pode ser estendida na adoção de jogos digitais educativos. Um aluno não pode, em essência, desbloquear álgebra até que domine conceitos prévios como a aritmética.

Estes cenários também implicam que o aluno tem algum controle sobre suas escolhas curriculares. Ter um sentimento de identidade e de autonomia é algo muito importante para os alunos. O erro mais comum nas atividades de educação *online* consiste na incapacidade de dar ao estudante uma sensação apropriada de identidade e autonomia. Identidade, neste contexto, se refere a habilidade de o aluno interagir com o material, com o sentimento de fazer parte do todo e também com o apoio sócio-emocional que recebe naquela situação[24]. Dalton reporta que 56% dos estudantes que participaram de cursos online sentiram falta de interatividade; eles não se viam como aprendizes ativos, com liberdade de escolha[25]. Jogos bem projetados, no entanto, encorajam os estudantes a se adaptarem e desenvolverem um estilo próprio de aprendizado ao qual melhor se adequem, o que por sua vez leva à um papel mais ativo no processo de aprendizagem.

2.1.3 Jogos oferecem engajamento para o aluno

A escola tradicional é frequentemente taxada como entediante por muitos alunos. De fato, cerca de metade dos alunos que abandonam as escolas de ensino médio nos Estados Unidos dizem que uma das principais razões para o abandono da escola é a falta de aulas interessantes e 70% diz não terem

sido motivados ou inspirados a trabalhar com dedicação[26]. Os professores há muito tempo vem usando diferentes abordagens de ensino, incluindo mídias contemporâneas e arte, para aumentar o engajamento e a motivação nas salas de aulas. Talvez a principal contribuição dos jogos eletrônicos seja justamente a sua habilidade de sustentar o engajamento e a motivação dos alunos ao longo do tempo, em particular com atividades mais desafiadoras e sem a necessidade de que o professor seja um *"Superstar"*[27][28]. Jogos digitais podem ser mais engajantes que atividades tradicionais de sala de aula[29][30]. Embora engajamento possa ser apenas um de diversos componentes, Kirkpatrick destaca que "Reações positivas podem não ser garantia de aprendizado, mas reações negativas certamente reduzem a possibilidade de que ele ocorra" [29].

2.1.4 Jogos ensinam habilidades do século XXI

Desenvolvedores de jogos e acadêmicos argumentam que os jogos capturam a atenção do jogador e os impulsiona em direção ao pensamento e a resolução de problemas[30][27]. Por exemplo, jogos requerem o tipo de raciocínio que nós precisamos ter no século XXI porque eles utilizam o próprio processo de aprendizado como mecanismo para avaliação[27] . Eles testam não somente o conhecimento e a técnica atuais do aluno, mas também preparam o aluno para aprendizados futuros. De acordo com outro estudo, jogos imersivos de alta qualidade requerem que os jogadores pensem sistematicamente e considerem relações ao invés de eventos ou fatos isolados[31]. A abundância de opções e possibilidades de decisão dos jogos forçam os jogadores a não somente aplicarem seus conhecimentos, mas também adaptá-los a situações diferentes. Eles devem pensar de maneira abstrata, porque estão jogando de forma abstrata. Isto os ajuda a desenvolver suas habilidades em tomada de decisões, inovação e resolução de problemas[32].

2.1.5 Jogos oferecem um ambiente para avaliação autêntica e relevante

É importante notar que, por definição, jogos são inerentemente formas de avaliação. Jogos e avaliações tradicionais têm características importantes em comum que representam formas de medir conhecimento e habilidade. Os dois são tecnologias complementares que podem ser combinadas para criar modelos mais refinados do conhecimento, habilidades e comportamento dos estudantes. Por exemplo, os jogos oferecem maneiras de representar de forma prática idéias que muitas das vezes são sub-representadas em avaliações tradicionais [33]. Nos jogos, a avaliação ocorre conforme o sistema interpreta as ações do jogador e

fornece *feedback* imediato. Os jogadores progridem ou não; eles avançam para o próximo nível ou tentam novamente.

3
Trabalhos Relacionados

Existe um grande número de trabalhos sobre ferramentas de desenvolvimento de jogos eletrônicos e sobre a aplicação de jogos eletrônicos como ferramentas educativas. Este trabalho pode ser correlacionado com essas duas categorias.

3.1
Ferramentas de Desenvolvimento de Jogos Eletrônicos

Neste trabalho, utilizamos como ponto de partida o motor de jogos *CppPlay*. O *CppPlay* é a versão C++ aprimorada de um outro motor de jogos, o *JavaPlay*, que fora concebido como objeto de estudos no livro "Introdução a Ciência da Computação com Jogos"[34]. O desenvolvimento do *CppPlay* teve início no laboratório *VisionLab* e posteriormente foi continuado pelo presente autor, responsável por importantes avanços em sua arquitetura, tais como: a adoção de uma camada de *scripts Lua*[2] e a adoção do *Pipeline* programável *OpenGL*[35]. Em relação ao motor *CppPlay*, gostaríamos de destacar os seguintes trabalhos relacionados:

- **löve2d** löve2d[36] é um *framework* para criação de jogos 2D que possui código aberto e utiliza Lua como linguagem de scripts. Conta com alguns recursos avançados, tais como:

  - **Suporte a física de corpos rígidos**
  - **Suporte a shaders**

  A ferramenta tem uma comunidade bastante ativa e uma boa quantidade de projetos disponíveis para download.

  A principal correlação entre löve2d e o *CppPlay* consiste na adoção de *Lua* como linguagem de *scripts*.

- **Pygame**

  Pygame[37] é um conjunto de módulos *Python* projetados para o desenvolvimento de jogos eletrônicos. Não é exatamente um motor, mas adiciona recursos multimídia aos *scripts Python* e, desta forma, possibilita que sejam desenvolvidos motores simplificados, contendo mecanismos

específicos de cada jogo ou motores de propósito geral, tais como o *löve2d* e o próprio CppPlay. *Pygame* se correlaciona com o *CppPlay* pelo fato de utilizar a linguagem de *scripts* Python, frequentemente comparada com a linguagem de *scripts* Lua, adotada pelo *CppPlay*.

3.1.1 Vantagens do motor CppPlay

O *CppPlay* foi construído tendo em mente uma API amigável e uma arquitetura orientada à componentes e eventos, procurando oferecer funcionalidades similares as encontradas em ferramentas RAD (i.e, *Rapid Application Development*) de sucesso, sejam estas destinadas a jogos eletrônicos (e.g., *Unity3D*, *Unreal Development Kit 3*) ou de propósito mais geral (e.g., *Visual Basic*, *Delphi*, *.Net Winforms*).

Esta abordagem faz com que o desenvolvimento de jogos no *CppPlay* ocorra em um nível mais alto de abstração do que nas ferramentas relacionadas, ocultando complexidade desnecessária do desenvolvedor, aumentando sua produtividade e diminuindo a curva de aprendizado. Maiores detalhes sobre essa afirmação serão apresentados na seção 5.1.

3.2 Jogos Eletrônicos como Ferramentas Educativas

Uma vez que o propósito deste trabalho consiste da construção de uma plataforma de desenvolvimento de jogos eletrônicos destinados à educação, é natural que ele também possa ser relacionado com trabalhos dessa categoria. Para citar alguns:

Em sua dissertação de mestrado, Soares (2012) utiliza uma versão em estágio inicial deste mesmo motor para desenvolver jogos educativos de aventura em Lua de conteúdo adaptável. Seu trabalho foi utilizado com sucesso por alunos e professores da escola NAVE (Núcleo Avançado em Educação) que construíram em sala de aula um jogo desse gênero[38].

Em sua dissertação de mestrado, Ignácio (2013) desenvolve um jogo de RPG utilizando a ferramenta *RPG Maker* com o objetivo de ensinar química[39].

Devlin, K. et al (2015), apontam que um problema comum à maioria dos sistemas educacionais consiste em tratar aprendizagem e avaliação como se fossem coisas distintas e argumentam que os jogos eletrônicos são capazes de fornecer uma boa oportunidade de conjugar de maneira atrativa e envolvente esses dois processos em um só[40].

3.3 Diferencial do nosso trabalho

Um grande diferencial do nosso trabalho consiste no fato de que utilizamos tecnologias que nós mesmos desenvolvemos. Além de nos garantir total conhecimento sobre as funcionalidades e limites das ferramentas que dispomos, isso também nos dá a vantagem de poder (e saber) alterá-las conforme novas necessidades possam surgir (e.g., desenvolvemos um componente *WebBrowser* para acessar e renderizar tecnologias Web dentro do jogo de maneira indistinguível). Como não precisamos nos preocupar com limitações desconhecidas ou mesmo com a curva de aprendizado de ferramentas de terceiros, pudemos dedicar o tempo que seria empregado nessas questões da área de Ciência da Computação na busca por conhecimento das áreas de Pedagogia e Entretenimento. Naturalmente, isso não nos dá a autoridade acadêmica de pedagogos ou a "bala de prata" do entretenimento, mas acreditamos obter vantagem em relação a trabalhos que não puderam se dar ao luxo de dedicar tanto tempo a essas áreas, tão importantes quanto Ciência da Computação, em trabalhos dessa natureza.

4
RPG e Aplicações em Ensino

Embora este seja um trabalho na área de Ciência da Computação, ele é fortemente inspirado pelas áreas de Entretenimento e Pedagogia. Desta forma, antes de apresentar o trabalho que realizamos enquanto que cientistas da computação, precisamos apresentar algumas das idéias que inspiraram e conduziram este trabalho, em especial, o porquê de optarmos pelos RPGs eletrônicos e especificamente pelo sub-gênero JRPG (Japanese Role-Playing Game) como categoria dos jogos educativos que desejamos facilitar o desenvolvimento.

4.1
O que são RPGs?

RPG é o acrônimo para Role-Playing Games (jogos de interpretação de papéis ou jogos de representação em português) que é um tipo de jogo onde os jogadores encarnam personagens, tais como guerreiros, magos, curandeiros, ninjas, lutadores, hackers e outros heróis em uma aventura. RPGs tradicionais são conhecidos como RPGs de lápis e papel (RPGs de mesa é outro termo utilizado) e não requerem o uso de qualquer dispositivo eletrônico: de maneira similar à uma peça de teatro improvisada, os jogadores interpretam seus personagens decidindo quais ações irão tomar na história conforme a situação que estiverem enfrentando e as habilidades e personalidade do avatar que assumiram. Tipicamente os RPGs contam com um elaborado conjunto de regras que tomam livros inteiros (e.g., o conjunto básico do jogo *Dungeons&Dragons* consiste de três livros conforme pode ser visto na Figura 4.1) e determinam se as ações dos jogadores foram bem sucedidas ou não, mas a palavra final sobre esses resultados cabe a um jogador especial, o mestre.

Normalmente é o mestre quem cria a história que será jogada em uma seção de RPG. O mestre desenha os mapas, arquiteta os ambientes e escolhe os inimigos que serão enfrentados. A principal função do mestre é narrar essa história para os jogadores enquanto mantém o jogo divertido e equilibrado para todos e, exatamente por isso, é conveniente que o mestre esteja acima das regras. O trecho[1] a seguir exemplifica como funciona uma sessão de RPG:

[1] Apresentar um trecho de uma sessão de RPG é a melhor estratégia e a fórmula adotada em manuais clássicos como o de *Dungeons&Dragons* para explicar seu funcionamento.

Figura 4.1 – Caixa e conjunto básico dos livros de regras para o jogo *Dungeons&Dragons*.

Mestre: *Vocês acordam em uma sala tétrica. O chão, o teto e três paredes são feitos de rocha sólida, mas uma quarta "parede" é composta de grossas barras de ferro. Vocês estão presos! O que pretendem fazer?*

Barab (guerreiro): *Vou me levantar e ver se há mais alguém preso conosco.*

Mestre: *Você vê uma jovem desacordada próxima à grade. Ela está deitada sobre uma poça de sangue...*

Adam (clérigo): *Vou me dirigir até ela e lançar a magia "curar ferimentos graves". -Vita mortis careo!!!*

Mestre: *Um raio verde sai de suas mãos e envolve o corpo da jovem. Aos poucos ela começa a recobrar a consciência e... VOCÊS SÃO SURPREENDIDOS POR UMA IMENSA CRIATURA QUE ARREBENTOU A PORTA QUE VEM DE UM CORREDOR E PAROU ENFURECIDA EM FRENTE A CELA, ESTALANDO UM ENORME CHICOTE NO CHÃO.*

Jerj (Hobgoblin NPC): *Imbecis, por que fazer isso? Seu jantar não querer ser jantar se estar vivo MuaHaHaha!!!* ***(Rosnou a criatura cheia de ódio e sarcasmo em português desarticulado.)***

Mestre: *Jerj está tentando chicotear a jovem e (****é possível ouvir o som do mestre rolando um dado onde não pode ser visto****)... ele acerta*[2]*! A jovem mal começou a acordar e desmaiou de novo... Vocês viram uma pequena esfera de vidro caindo do bolso dela após ter sofrido o ataque do chicote.*

Barab: *Vou tacar uma pedra na cabeça desse monstro desgraçado!!!*

[2]É provável que algum jogador conteste como seria possível a criatura chicotear alguém através das barras de ferro; dúvida para a qual um mestre habilidoso sempre tem a resposta perfeita: *Magia... É um chicote semi-etéreo do plano do pavor!*

Mestre: *OK, role um dado D20 para ver se consegue acertá-lo.*

Barab: *Tirei 13 e eu tenho um bônus +1 devido à minha destreza.*

Mestre: *Certo! Você conseguiu acertá-lo bem no meio da testa. Role um D4 para ver quanto consegue causar de dano.*

Barab: *4!! Uhuuul!!!*

Mestre: *A criatura urra de dor e esbarra em uma das tochas que está sobre a parede fazendo com que seu pêlo fique em chamas!!!*

Adam: *Vou aproveitar e lançar prender monst... (**O mestre interrompe**)*

Mestre: *FLASHIWSHH!!! Antes que você completasse a ação um imenso clarão tomou conta do ambiente... vocês não conseguem manter os olhos abertos, pois a luz é muito intensa...*

Toda essa narrativa acontece de forma improvisada e aleatória devido ao lance de dados e a criatividade dos jogadores. Se a história fosse narrada para um outro grupo, a cena muito provavelmente iria acontecer de uma forma completamente diferente da descrita acima. Justamente essa natureza imprevisível e o grau de liberdade comum às sessões de RPG que aliados a uma ambientação fantástica, tornam este tipo de jogo tão envolvente.

4.2 Como os RPGs se relacionam com a pesquisa de James Paul Gee

Jogos de RPG atendem aos 16 princípios de aprendizagem relacionados pelo pesquisador James Paul Gee como fundamentais ao aprendizado e inerentes aos bons jogos[31]:

1. **Identidade**: Os jogadores assumem totalmente o papel de um personagem

2. **Interação**: Os jogadores podem interagir com o cenário, outros jogadores ou personagens do jogo que, ao contrário dos livros, respondem de volta.

3. **Produção**: Os jogadores não só consomem, mas participam como co-autores do jogo atribuindo-lhes diferentes formas conforme suas decisões particulares.

4. **Assumir riscos**: Bons jogos minimizam as consequências quanto a falhas. Isto encoraja os jogadores a assumirem riscos e experimentarem diferentes estratégias.

5. **Customização**: Os jogadores podem, de uma maneira ou outra, adequar o jogo ao seu próprio estilo de aprendizagem e de resolver problemas.

6. **Ser agente**: Graças aos princípios anteriores, o jogador experimenta uma sensação real de responsabilidade e posse sobre o que está experimentando, sentindo-se parte do todo.

7. **Problemas bem ordenados**: Bons jogos apresentam os desafios de forma nivelada, onde os mais fáceis são apresentados primeiro e o conhecimento adquirido possa ser aplicado em problemas posteriores mais difíceis.

8. **Desafio e consolidação**: Bons jogos oferecem repetidamente ao jogador um conjunto de desafios até que ele internalize as soluções e as considere triviais. Então os jogos apresentam novos problemas que desafiam as estratégias até então desenvolvidas forçando o jogador a obter um novo aprendizado e integrá-lo ao conhecimento anterior.

9. **"Bem na hora" e "Sob demanda"**: Pessoas tem uma capacidade limitada de lidar com um monte de palavras fora de contexto; por isso livros texto podem ser tão ineficientes. Os jogos quase sempre fornecem as informações no exato momento em que são necessárias ou quando o jogador sente que precisa da informação, deseja a informação, está pronto para a informação e pode fazer bom uso dela.

10. **Significado situado**: Pessoas tem dificuldade em aprender o significado de palavras quando tudo de que dispõem é uma definição de seu significado em termos de outras palavras. Estudos recentes sugerem que as pessoas só aprendem realmente o significado de uma palavra quando conseguem associá-la ao tipo de experiência ao qual ela se refere, ou seja, ao tipo de imagem, ação ou diálogo aos quais a palavra se relaciona (e.g., choque elétrico).

11. **Frustração prazerosa**: Graças a muitos dos princípios acima, bons jogos permanecem dentro dos limites de competência do jogador, mas bem próximos à borda. Quando um aprendiz se depara como uma situação que ele sabe que é possível resolver mas considera desafiadora, ele entra em um estado de alta motivação.

12. **Pensamento sistêmico**: Os jogos encorajam o jogador a pensar em relações, não somente em eventos, fatos ou habilidades isoladas. Se o jogador se depara com uma situação onde ele precisa decidir entre atacar o inimigo ou curar um outro integrante do time, ele precisa avaliar o impacto de cada decisão: "*Se eu atacar o inimigo agora ele pode ser derrotado nesse instante, mas se ele sobreviver e matar o meu colega,*

na próxima rodada eu estarei enfrentando-o sozinho e ele certamente irá concentrar todos os seus ataques em mim".

13. **Explorar, pensar perifericamente, reavaliar metas**: Aprendi na escola, assim como muitos *baby boomers*, que ser inteligente é avançar o mais rápida e eficientemente possível até alcançar um objetivo. Jogos encorajam uma atitude diferente. Jogos encorajam os jogadores a explorarem bem as opções antes de avançarem, a pensarem perifericamente e não linearmente e a utilizarem esta exploração e pensamento periférico para reavaliar as metas de tempos em tempos. Soa exatamente como o que ambientes de trabalho globalizados e de alta tecnologia procuram.

14. **Ferramentas inteligentes e conhecimento distribuído**: Os personagens virtuais que o jogador controla em um jogo, bem como outros aspectos do mundo simulado, são na verdade *ferramentas inteligentes*. Tais agentes possuem conhecimentos próprios que eles emprestam ao jogador. Por exemplo, no jogo *Full Spectrum Warrior*, os soldados que o jogador controla sabem como se mover e tomar diferentes formações de batalha. Este conhecimento, portanto, não é algo que o jogador precisa conhecer. O jogador só precisa saber onde e quando ordenar cada formação de forma que os soldados possam se movimentar em segurança. O conhecimento necessário para jogar o jogo é então distribuído entre o jogador e os soldados.

15. **Times multi-função**: Quando os jogadores jogam um jogo colaborativo online como World of Warcraft, eles normalmente jogam em times onde cada jogador possui um diferente conjunto de habilidades (e.g., Mago, Guerreiro, Druída). Cada jogador precisa dominar sua especialidade, mas conhecer suficientemente as habilidades dos outros personagens a fim de integrar e coordenar as ações. É comum nesses times as pessoas serem classificadas primeiramente pela sua função e não por sua raça, tipo de personagem, etnia ou gênero. As missões são projetadas de forma que diferentes tipos de personagem sejam necessários para alcançar o sucesso. Por exemplo, pode ser necessário um *tanker* (personagem que possui bastante energia vital e defesa) para ficar na linha de frente recebendo os ataques mais fortes de uma criatura, enquanto um *healer* fica encumbido de restaurar, de tempos em tempos, sua energia vital e um *mage* permanece de longe lançando poderosos ataques contra a criatura. Novamente, este tipo de trabalho em equipe é muito requerido em ambientes de trabalho modernos; infelizmente, não nas escolas.

16. **Prática antes da competência**: Bons jogos operam sobre o princípio inverso ao da maioria das escolas: prática antes da competência. Os jogadores podem praticar antes de obter competência, amparados pelo design do jogo, os agentes inteligentes que o jogo oferece e, muito frequentemente, por jogadores mais experientes (em jogos multi-jogadores, salas de bate papo, ou presencialmente). É assim que o aprendizado de linguagem ocorre, embora nem sempre nas escolas, que normalmente requerem que os alunos obtenham competência antes de serem capazes de praticar sob o domínio que estão estudando.

4.3 Como os RPGs se relacionam com a pesquisa de Chandra Balakrishnan

Chandra Balakrishan (2008) realiza um excelente trabalho ao levar para sala de aula elementos de contação de histórias com o objetivo de ensinar matemática[41]. Sua pesquisa é baseada nos trabalhos de Egan (2005)[42] e Zazkis & Liljedahl (2008)[43], por ele sintetizados com a máxima: "Se quisermos engajar os alunos com o material do currículo matemático, então precisamos envolver suas imaginações –e consequentemente suas emoções".

A título de ilustração, apresentamos a seguir a metodologia utilizada pelo pesquisador para ministrar uma de suas aulas (uma das quais utilizaremos em nossa prova de conceito):

- **Ferramentas cognitivas por grau de maturidade**

  A fim de identificar as ferramentas cognitivas apropriadas ao grau de maturidade dos alunos, o pesquisador utiliza a seguinte classificação desenvolvida por Egan (2005)[42]:

  - **Entendimento mítico (entre 3 e 7 anos / a partir do desenvolvimento da fala)**
    - **História**: O indivíduo é capaz de acompanhar a estrutura e compreender a moral de histórias.
    - **Metáfora**: O indivíduo é capaz de entender idéias através do uso de figuras de linguagem (e.g., João era uma tartaruga indo para a escola, mas um coelho voltando para casa).
    - **Binários opostos**: O indivíduo é capaz de identificar conceitos opostos (e.g., bem/mal, fogo/água, forte/fraco, lento/rápido) e entender como tais conflitos conduzem o desenvolvimento de uma história.
    - **Rima, ritmo e reconhecimento de padrões**: O indivíduo atribui carga emocional e desenvolve particular interesse por

histórias contadas através de rima, ritmo e outros padrões (e.g., Os Três Porquinhos, onde o lobo utiliza repetidamente seu sopro para derrubar as casas).
    * **Faz de conta**: O indivíduo é capaz de imaginar e se colocar em situações fictícias (e.g., brincar de mocinho e bandido).
    * **Mistério**: O indivíduo desenvolve particular interesse por histórias que contenham segredos e outros mistérios que exercitem sua curiosidade e imaginação.
  - **Entendimento Romântico (por volta dos 7 anos)**
    * **Senso e limites da realidade**: O indivíduo possui entendimento sobre os extremos e os limites da realidade (e.g., adimiração e fascínio por atletas de ponta ou por marcas apresentadas no *Guinness Books*) .
    * **Senso do fantástico**: O indivíduo é capaz de entender que alguns fenômenos só existem no mundo da imaginação (e.g., super-força do He-Man, velocidade da luz de poderosos Cavaleiros do Zodíaco).
    * **Associação com heróis**: O indivíduo estabelece vínculo com os heróis que mais se identifica (e.g., sou inteligente como o Batman, quero ser rápido como The-Flash, sou ágil como a Cheetara)
    * **Entendimento de narrativa**: O indivíduo é capaz de compreender e desenvolver histórias dentro de uma estrutura narrativa, antecipando e interligando fatos que foram apresentados outrora de maneira isolada e desordenada.
  - **Entendimento Filosófico (pré-adolescência/adolescência)**
    * **Senso de agência**: Quando o indivíduo se reconhece como parte de algo maior (e.g., a família, a escola, o mundo) e como agente capaz de modificar e contribuir para a harmonia desse meio.
    * **Repertório de conhecimento geral, idéias e anomalias**: Quando o indivíduo, através da experiência adquirida em seu crescimento, possui um repertório de conhecimento geral, idéias comumente aceitas e situações que pareçam contradizer esses fatos.

- **Arcabouço de planejamento**

  O pesquisador adequa então essas ferramentas cognitivas ao seguinte arcabouço de planejamento, desenvolvido por Zazkis & Liljedahl (2008)[43]:

1. **Identificar o tópico de interesse**: Neste exemplo, o autor deseja introduzir o conceito de divisão por zero em relação a simples expressões racionais e porque esta operação é indefinida.

2. **Identificar as dificuldades/peculiaridades sobre o tópico**: O problema aparece quando abordamos os valores não-permissíveis para variáveis de uma expressão racional, por exemplo:

$$\frac{x^2 + 5x + 6}{(x + 2)(x - 5)}$$

Ao explicar para os estudantes que os valores não-permissíveis desta expressão são:

$$x = -2 \text{ ou } x = 5$$

Eles costumam confundir valores não permissíveis com valores não permitidos (proibidos).

3. **Elaborar uma história interessante para apresentar o tópico**: *O Rei William II foi o Rei mais honrado e justo que já reinou em Lichtenstein. Ele amava os seus súditos e fazia tudo que estivesse ao seu alcance para ganhar seu respeito e lealdade. Sua generosidade era conhecida por todo os cantos e era comum vê-lo em companhia de pessoas comuns do povo, doando dinheiro e comida aos mais necessitados. No entanto, nada era mais motivo de orgulho e respeito para o Rei do que seus mais valiosos pertences - 12 preciosos diamantes. Não se tratavam de diamantes quaisquer. Eles irradiavam uma luz verde tão brilhante que certamente cegaria qualquer um que olhasse diretamente para eles* ***[Os estudantes acham curioso os diamantes conterem tal propriedade. Alguns dizem que eles são feitos de kryptonita]****. Os diamantes possuíam lasers em seu interior que, diretamente de seu centro, abriam um caminho de destruição por onde quer que acertassem* ***[Os estudantes protestam sobre o fato de alguém possuir algum conhecimento sobre lasers nessa época]****. Todos os outros lords e nobres invejavam o rei por causa destas jóias "laserosas"*[3]*. Por mais que os 12 diamantes trouxessem alegria para o Rei, seus 6 filhos lhe traziam terrível sofrimento. Eram homens maus, que sem piedade e cruelmente faziam com que todos os temessem. Seus nomes eram William III, Charles, Bartholomew, Henry,*

[3] o autor faz aqui um trocadilho com a palavra "*preciosas*" e a expressão "*à laser*".

James e Edward. Quando o Rei estava em seu leito de morte, seu desejo foi de que todos os 12 diamantes fossem distribuídos igualmente entre seus filhos e somente seus filhos! Cada filho deveria receber $\frac{12}{6}$ *diamantes. No entanto, cada filho queria todos os diamantes para si. Naquela mesma noite, um dos servos encontrou tanto Edward quanto James espancados até a morte em suas próprias camas. Agora, havia apenas 4 filhos. Cada filho receberia então* $\frac{12}{4}$ *ou* 3 *diamantes. Na manhã seguinte, quando Henry estava indo para sua caminhada matinal no jardim, alguém o atacou por trás e perfurou seu coração. Assim, cada filho receberia* $\frac{12}{3}$ *ou* 4 *diamantes. Mais tarde, Bartholomew foi encontrado carbonizado na lareira da sala de jantar. Logo, cada filho deveria receber* $\frac{12}{2}$ *ou* 6 *diamantes. Mais um dia se passou e William III foi encontrado afogado em uma poça de seu próprio vômito. Charles, o único filho que restou, receberia não só todos os* $\frac{12}{1}$ *diamantes mas também o título de Rei, já que seu irmão mais velho, William III, estava morto. No entanto, Charles contraiu lepra uma semana antes de sua coroação e morreu antes de receber qualquer herança. Uma vez que não havia mais nenhum filho, como poderia o juiz do testamento atender à vontade do Rei William de distribuir igualmente os 12 diamantes entre seus filhos? As instruções não faziam sentido devido ao fato de que não havia mais nenhum filho vivo. Esta situação é indefinida e portanto* $\frac{12}{0}$ *é indefinido.*

4. **Estender ou variar o problema**: Neste ítem, o pesquisador costuma pedir que os alunos elaborem suas próprias histórias em relação ao tópico para avaliar e também reforçar o entendimento.

Uma vez que narrativa e contação de histórias fazem parte da essência dos RPGs traduzir para este ambiente de jogo as técnicas e histórias que os pesquisadores citados desenvolveram (e utilizaram com sucesso na prática) é relativamente fácil.

O RPG tradicional é um recurso já utilizado em salas de aula, inclusive com o apoio do Ministério da Educação (MEC)[44] e, sem dúvidas, é uma ótima ferramenta pedagógica, no entanto, sua adoção pode representar desafios proibitivos para a maioria dos professores:

- Nem sempre os professores tem o tempo necessário para preparar aventuras, habilidade ou mesmo o interesse de fazê-lo.
- Para que uma sessão de RPG seja bem conduzida, o grupo envolvido não deve ter muito mais que 5 jogadores além do mestre, pois a partir

desse número a condução das ações e a aplicação das regras começa a ficar muito confusa.

- É comum salas de aula terem mais de 40 alunos e apenas 1 professor. Ainda que o professor consiga organizar a turma em pequenos grupos, ele ainda terá de conduzir esses grupos em um curto intervalo de aula.

4.4 Computer Role-Playing Games e JRPG

Os CRPGs (*Computer Role-Playing Games*) nada mais são do que uma adaptação do gênero RPG tradicional para plataformas eletrônicas (i.e., computadores e consoles de video game). Os CRPGs mantém grande parte dos elementos do RPG tradicional, tais como: regras de combate, sistema de evolução dos personagens, contação de histórias/narrativa predominante, liberdade de escolha e ambientação fantástica. No entanto, como consistem de jogos simulados por computador, permitem que sejam adotados como ferramenta educativa de maneira que o professor possa apenas supervisionar os alunos enquanto jogam, ou seja, sem enfrentar os desafios apresentados anteriormente com relação a adoção de RPGs tradicionais.

Além disso, os CRPGs não precisam ficar restritos às salas de aula. Nada impede que um jogo seja instalado no computador pessoal ou *smartphone* de um aluno como artefato complementar/opcional aos estudos.

4.4.1 A escolha pelo sub-gênero JRPG

Os CRPGs costumam ser classificados em dois sub-gêneros: WRPGs (*Western Role-Playing Games*) e JRPGs (*Japanese Role-Playing Games*). O curioso desta classificação é que se a analisarmos sob uma perspectiva taxonômica (i.e., de herança/generalização/especialização), perceberemos que a classe WRPG é quase tão abstrata quanto a sua super-classe CRPG. Em outras palavras, a classe WRPG não especifica tão bem as características e comportamentos que os jogos dessa categoria devem ter quando comparada com a classe JRPG. A fim de ilustrar essa afirmação, obtivemos uma lista dos 100 melhores CRPGs de todos os tempos segundo o respeitado portal de jogos eletrônicos IGN.com[45], descartamos[4] ocorrências de franquias repetidas (e.g., Diablo e Diablo II ou Final Fantasy VI e Final Fantasy VII), construímos uma tabela (Figura 4.2) contendo os 5 melhores jogos de cada sub-gênero complementando-a com colunas de classificação para os sub-sistemas comuns

[4]Com o objetivo de obter uma comparação mais justa visto que jogos de uma mesma franquia tendem a ser muito semelhantes.

aos dois gêneros, percentual de compatibilidade do jogo com o padrão de seu sub-gênero e outra coluna com sua colocação no *ranking* geral.

	SUBSYSTEMS						
Top 5 WRPGs	Inventory	Combat	Camera View	Global Map	Dialog System	Match	Rank
Mode' Implementation	Slot Max Weight Breakables	Active	Isometric	Static Bookmarks	NONE		
1 - Baldur's Gate II	Slot Max Weight Breakables	Active Pauseable Turn Based	Isometric	Static Bookmarks	Text Panel Partial Speech	60%	3
2 - Diablo II	Slot Size Puzzle Breakables	Active	Isometric	Static Waypoints	Windowed Full Speech	40%	6
3 - The Elder's Scrolls	Slot Max Weight Breakables	Active	First Person	Static Bookmarks	Predefined Categories	60%	7
4 - Mass Effect	List Based	Active Pauseable	Third Person	Static Bookmarks	Cinematic	20%	9
5 - Fallout	Stackables Max Weight	Turn Based	Isometric	Static Fog Bookmarks	Floating Text Extendable	20%	10
					Avg. Compatibility	40%	
	SUBSYSTEMS						
Top 5 JRPGs	Inventory	Combat	Camera View	Global Map	Dialog System	Match	Rank
Mode' Implementation	List Based	Turn Based	TopDown	Explorable	Windowed		
1 - Final Fantasy VI	List Based	Turn Based	TopDown	Explorable	Windowed	100%	1
2 - Chrono Trigger	List Based	Turn Based	TopDown	Explorable	Windowed	100%	2
3 - Pokemon Red/Blue	List Based	Turn Based	TopDown	Explorable	Windowed	100%	4
4 - Secret of Mana	Radial Menu	Active	TopDown	Explorable	Windowed	60%	15
5 - EarthBound	List Based	Turn Based	TopDown	Explorable	Windowed	100%	17
					Avg. Compatibility	92%	
' value that appears most often in a set of data.							

Figura 4.2 – Tabela de compatibilidade dos sub-gêneros WRPG e JRPG com relação a moda dos sub-sistemas CRPG.

Os sub-sistemas destacados em vermelho denunciam sua incompatibilidade e a necessidade de especialização com relação a moda (i.e., o tipo de sub-sistema que mais se repete na amostra) do sub-gênero ao qual o jogo foi enquadrado. Uma maneira diferente de observar esse fenômeno é através de um diagrama de classes (Figura 4.3).

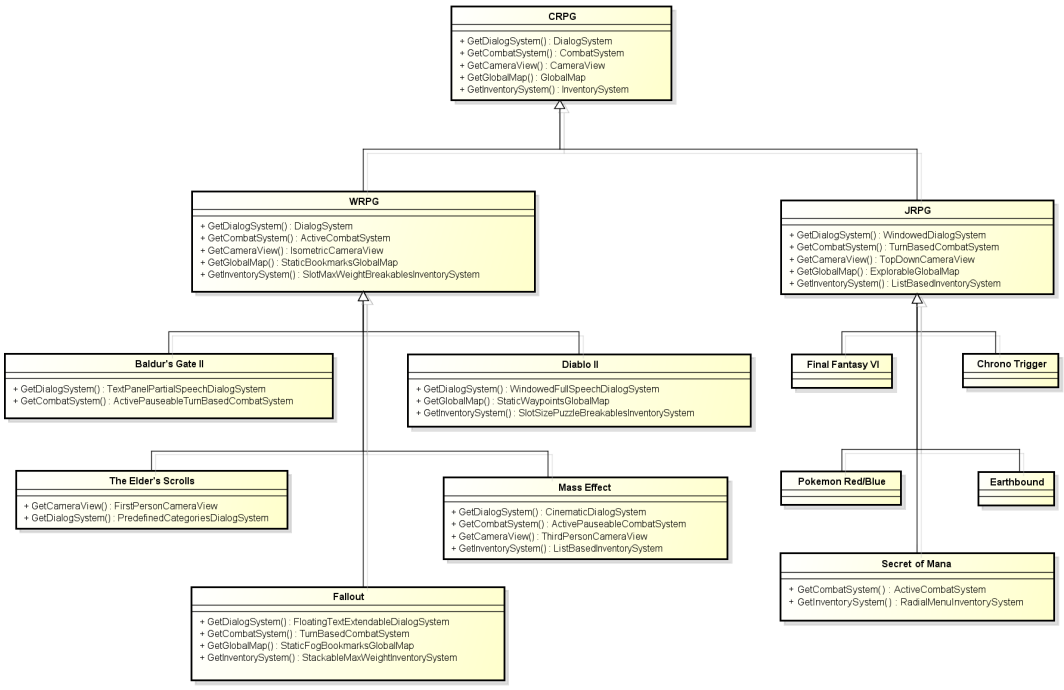


Figura 4.3 – Diagrama de classes representando as especializações que se fazem necessárias para implementação dos 5 melhores jogos de cada sub-gênero.

No diagrama, os sub-gêneros são representados pelas classes WRPG e JRPG. Essas classes implementam a moda correspondente aos jogos dos

respectivos sub-gêneros e cada folha representa um jogo que especializa apenas os sub-sistemas que se diferenciam da moda.

Com base nessa classificação, concluímos que o sub-gênero JRPG é mais padronizado, garantindo 92% de compatibilidade média entre seus jogos e sub-sistemas contra 40% de compatibilidade média do sub-gênero WRPG. Uma vez que essa padronização fica sob responsabilidade da plataforma que estamos desenvolvendo, estes percentuais se traduzem em quanto trabalho será poupado em média para o desenvolvedor que vier a utilizar nossa plataforma.

A maior padronização dos JRPGs também pode ser observada do ponto de vista artístico (Figura 4.4). Na figura, podemos observar que os diferentes jogos do sub-gênero JRPG possuem mais elementos visuais em comum do que os jogos do sub-gênero WRPG.

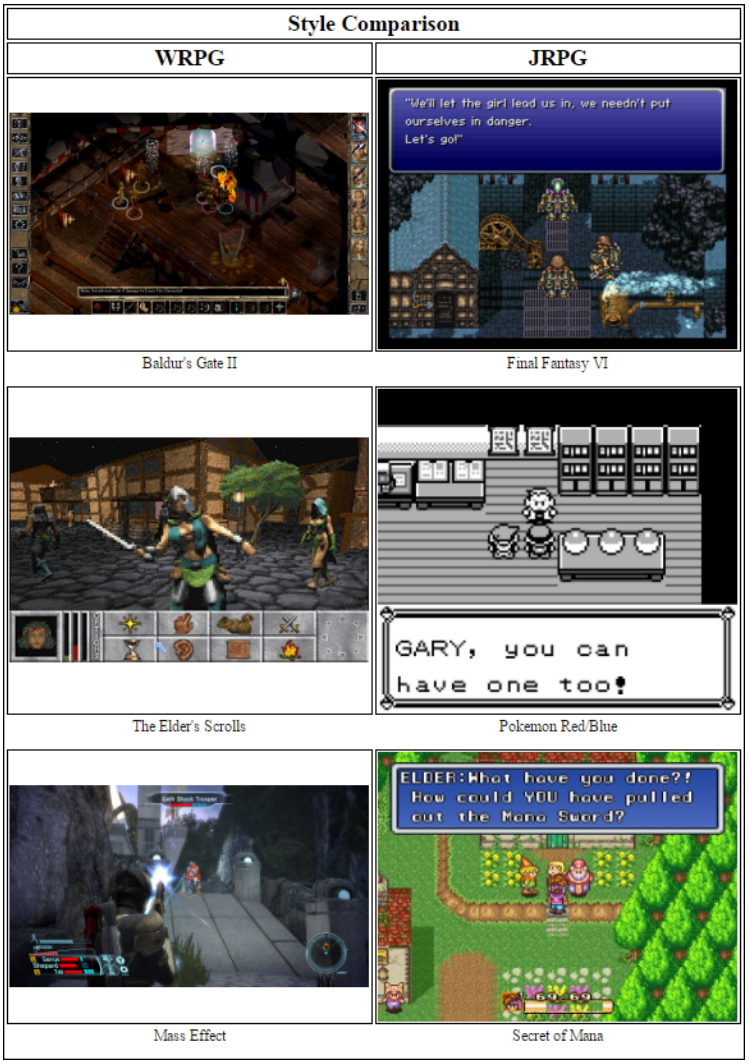


Figura 4.4 – Comparação de estilos artísticos entre jogos dos sub-gêneros WRPG e JRPG.

Embora exista uma maior padronização e identidade nos jogos JRPG, eles não precisam se prender a nenhuma forma. Se fizer sentido em uma

história de JRPG que o personagem pilote uma nave espacial ou participe de um jogo de futebol, é completamente aceitável (e bastante comum) que o jogo se transforme momentaneamente em um simulador desse outro gênero, simulando-o tão fielmente quanto o desenvolvedor julgar necessário. Um bom exemplo disso pode ser visto na Figura 4.5, onde o personagem *Cloud* (até então mercenário) pilota uma motocicleta enquanto protege seus companheiros em uma perseguição automobilística.

Figura 4.5 – *Cloud*, do *JRPG Final Fantasy VII* em uma perseguição utilizando motocicleta.

A partir dessas observações, consideramos o sub-gênero JRPG mais adequado para nossa proposta.

4.4.2
O jogo Chrono Trigger como exemplo de um JRPG

O jogo *Chrono Trigger*[46] (Figura 4.6) é sem sombra de dúvidas um dos maiores expoentes do gênero JRPG e uma grande inspiração para esse trabalho. O jogo é frequentemente descrito por desenvolvedores, jogadores e pela crítica especializada como um dos melhores jogos de todos os tempos. Gostaríamos de recorrer ao seguinte *review* do respeitado portal de jogos eletrônicos IGN.com como forma de apresentar o jogo: "*Chrono Trigger* é um dos melhores jogos de video game já criados. Um jogo de RPG amado mundialmente que teve seu primeiro lançamento no console Super Nintendo em 1995. Sua criação é tida como lendária, pois o time de desenvolvimento (conhecido como *Dream Team*) contava com os mais talentosos criadores de RPG da era de 16-bits, todos reunidos para trabalhar neste projeto épico. Seu sistema de batalhas é incrivelmente único. Você controla um grupo de até três heróis simultâneos, lutando contra inimigos no exato local onde você os encontra (em contraste com o sistema tradicional de outros jogos onde os combates acontecem em um cenário genérico). Além disso, as mecânicas

Figura 4.6 – Avatar e sistema de batalhas do jogo *Chrono Trigger*.

de combate apresentam diferentes oportunidades de trabalho em equipe. Todo personagem possui um comando de ataque básico e também pode utilizar ítems do inventário (e.g., poções de cura). Até aí, nenhuma novidade, mas a terceira opção, *Tech*, é onde a coisa fica interessante. Ao selecionar este comando você pode desferir ataques mais potentes, magias e manobras com cada um dos heróis individualmente, mas você também pode combinar movimentos de dois ou mais personagens e criar combinações ainda mais fortes"[47].

Outros motivos para o sucesso do jogo incluem:

- Um enredo cativante, com viagens no tempo, magias, tecnologias, histórias dramáticas e situações cômicas
- Uma incrível arte visual (do aclamado criador e desenhista do anime Dragon Ball, Akira Toryiama)
- Música e efeitos sonoros marcantes

5
Plataforma Proposta

Neste capítulo apresentamos o trabalho prático realizado para o desenvolvimento da plataforma que propomos com base na pesquisa teórica.

5.1
O motor de jogos CppPlay

Para o desenvolvimento da plataforma proposta, elegemos como núcleo da arquitetura o motor de jogos CppPlay[2]. O CppPlay é um motor de jogos eletrônicos 2D originalmente criado pelos laboratórios VisionLab/ICAD/IGames da PUC-Rio com o objetivo apoiar e facilitar o desenvolvimento de diversos projetos desses laboratórios bem como o de trabalhos acadêmicos (e.g., Projeto Final de Graduação, Dissertação de Mestrado). Em 2012, sob orientação do coordenador desses laboratórios, Prof. Dr. Bruno Feijó, o presente autor assumiu o desenvolvimento do motor CppPlay adicionando uma camada de scripts Lua[48] e reescrevendo totalmente sua arquitetura. Posteriormente, o motor foi utilizado como objeto de trabalho da disciplina Projeto Final de Programação[35], obrigatória deste curso de Pós-Graduação, onde realizamos a troca do antigo pipeline gráfico não-programável OpenGL pelo pipeline OpenGL moderno, programável em linguagem de *shaders* GLSL[49][50] a fim de possibilitar o emprego de avançadas técnicas de renderização em tempo real e o alcance de maior impacto visual (gráficos representam um fator importante na produção de jogos de qualidade).

O motor foi projetado utilizando diversos padrões de projeto[51] que possibilitam uma API amigável e uma arquitetura orientada a componentes e eventos, semelhante a presente em ferramentas RAD (*Rapid Application Development*) como *QtCreator*, *Delphi* e *.Net Winforms*.

No CppPlay, podemos construir as mais complexas estruturas a partir da agregação dos seguintes componentes básicos:

- **GameObject**: Representa a unidade de jogo mais abstrata da hierarquia de classes do motor. É comumente utilizada como *container* de outros componentes ou como objeto sem forma/não renderizável que necessite de atualização automática e contínua (e.g., *Timer* esperando a passagem de um intervalo de tempo para executar uma função de *callback*).

- **Sub-classes**: Especializações de **GameObject** que também podem funcionar como *container*:

  - **Sprite**: Representa um elemento gráfico construído a partir de um arquivo de imagem (e.g., bmp, png, jpeg, gif) é capaz de renderizar simples quadros ou um conjunto de animações.
  - **Sensor**: Representa uma forma geométrica capaz de detectar colisão, movimento e cliques de mouse e repassar o tratamento para a raiz da hierarquia de componentes.
  - **AutoSensor**: Representa uma forma geométrica capaz de detectar colisão, movimento e cliques de mouse e tratar diretamente o evento (i.e., sem repassá-lo para a raiz da hierarquia).
  - **SpriteSensor**: Representa uma fusão conveniente entre **Sensor** e **Sprite**, ou seja, é um sensor físico que admite imagens e animações.
  - **Text**: Permite renderizar textos de diferentes tamanhos e cores utilizando fontes *TrueType*.
  - **WebBrowser**: Permite renderizar páginas Web com a possibilidade de comunicação entre as camadas utilizando Lua ⇔ JavaScript.

Todos estes componentes podem ser organizados em uma hierarquia composta por pais → filhos de forma que: qualquer transformação (translação, rotação, redimensionamento) aplicada a componentes-pai (i.e., um componente que é composto por diversos outros componentes) afete também aos seus componentes-filhos.

A camada de scripts, por sua vez, permite que estes componentes tenham comportamento definido por eventos:

- **GameObject e Sub-classes**:

  - **OnCreate(sender)**: Ocorre uma vez que o objeto seja de fato instanciado em um **GameStateController** e passe a ser considerado um objeto ativo na cena. Nesse momento, podemos inicializar o alvo de um míssil teleguiado e/ou executar um som de disparo, por exemplo.
  - **OnDestroy(sender)**: Ocorre quando o objeto está sendo destruído. Nesse momento, pode ser interessante, por exemplo, criar um **Sprite** com uma animação de explosão.
  - **OnUpdate(sender)**: Ocorre a cada *frame*. É sem dúvidas o evento mais importante na execução de um jogo, pois é nesse momento que as entidades podem executar suas ações em resposta aos estados internos (e.g., fugir quando há pouca quantidade de energia

disponível, procurar munição quando esta estiver acabando, explodir uma bomba quando uma contagem regressiva acabar), externos (e.g., atirar em um inimigo próximo, socorrer um amigo ferido) e entradas do usuário (e.g., andar quando uma tecla de direção for mantida pressionada), enfim, é este o evento que dá vida ao jogo.

- **Sensor→Raiz, SpriteSensor→Raiz e AutoSensor**:
  - **OnClick(sender, mouse)**: Ocorre quando um **Sensor** recebe um clique de mouse. Pode ser utilizado, por exemplo, para representar botões em uma interface com o usuário.
  - **OnMouseMove(sender, mouse)**: Ocorre quando o cursor do mouse se move sobre um **Sensor**. É bastante útil quando se deseja destacar uma entidade que esteja sob o cursor do mouse (e.g., fazer um objeto de interesse brilhar para explicitar sua interatividade).
  - **OnBeginContact(sender, b)**: Ocorre quando é detectada uma colisão entre dois objetos do tipo **Sensor**. Bastante útil para identificar e impedir o acesso à lugares bloqueados, coletar ítens, disparar armadilhas ou executar *cutscenes* quando o jogador entrar em uma área pré-determinada.
  - **OnEndContact(sender, b)**: Ocorre quando uma colisão deixa de existir. Por exemplo, quando um personagem boneco de neve animado deixar uma área ensolarada demarcada por um **Sensor**), pode-se ativar um estado que o faz parar de derreter e começar a se regenerar.

- **GameStateController**:
  - **OnLoad(state)**: Ocorre quando o **GameStateController** é executado pela primeira vez pelo motor. Esse é um momento apropriado para carregar recursos a partir do disco rígido como: imagens, áudio, páginas Web e fontes de texto.
  - **OnStart(state)**: Ocorre sempre que o **GameStateController** se torna o estado ativo, ou seja, quando passa a ser executado pelo motor em seu *GameLoop*.
  - **OnUnload(state)**: Ocorre quando o **GameStateController** é destruído. Pode ser utilizado, por exemplo, para gerar uma nova cena de forma procedimental em substituição a que está sendo descarregada.
  - **OnUpdate(state)**: Ocorre uma vez a cada *frame*. Pode ser utilizado, por exemplo, para verificar se uma lista de objetivos foram cumpridos e então executar uma nova cena.

  - **OnMouseDown(state, button)**: Ocorre quando um botão do mouse é pressionado. Pode ser utilizado, por exemplo, para definir a posição de destino de uma entidade de jogo.
  - **OnMouseMove(state, mouse)**: Ocorre quando a posição do cursor do mouse é modificada. Pode ser utilizada, por exemplo, para desenhar um cursor de mouse personalizado.
  - **OnMouseUp(state, button)**: Ocorre quando um botão de mouse pressionado anteriormente é liberado. Pode ser utilizado, por exemplo, para aplicar um *delta* em relação ao momento em que o botão foi inicialmente pressionado e assim determinar a potência de uma ação (e.g., chute para o gol, sopro de um dragão, arremesso de uma pedra).
  - **OnKeyDown(state, keycode)**: Ocorre quando uma tecla é pressionada. Pode ser utilizado, por exemplo, para solicitar a finalização de uma *cutscene* quando o jogador pressionar *Enter*.
  - **OnKeyUp(state, keycode)**: Ocorre quando uma tecla pressionada anteriormente é liberada. Pode ser utilizada, por exemplo, para começar a desacelerar um carro em movimento quando a tecla referente ao acelerador parar de ser pressionada.

- **Sprite e SpriteSensor**:

  - **OnAnimationStarted(sender, animation)**: Ocorre quando uma animação começa a ser executada. Pode ser utilizada, por exemplo, para tocar um efeito sonoro condizente com a animação: som do desembanhar de uma espada, som de explosão.
  - **OnAnimationFinished(sender, animation)**: Ocorre quando uma animação termina de executar seu último *frame*. Bastante útil para realizar a transição entre um estado e outro (e.g., Após o término de uma animação de antecipação de movimento, executar uma animação de corrida e aumentar a velocidade do personagem).

A combinação destes mecanismos possibilita criar desde um simples pixel, que acompanha o cursor do mouse na tela, até estruturas complexas como uma nave espacial alienígena dotada de inteligência artificial e composta por módulos semi-autônomos como canhões laser e escudo de plasma (Figura 5.1).

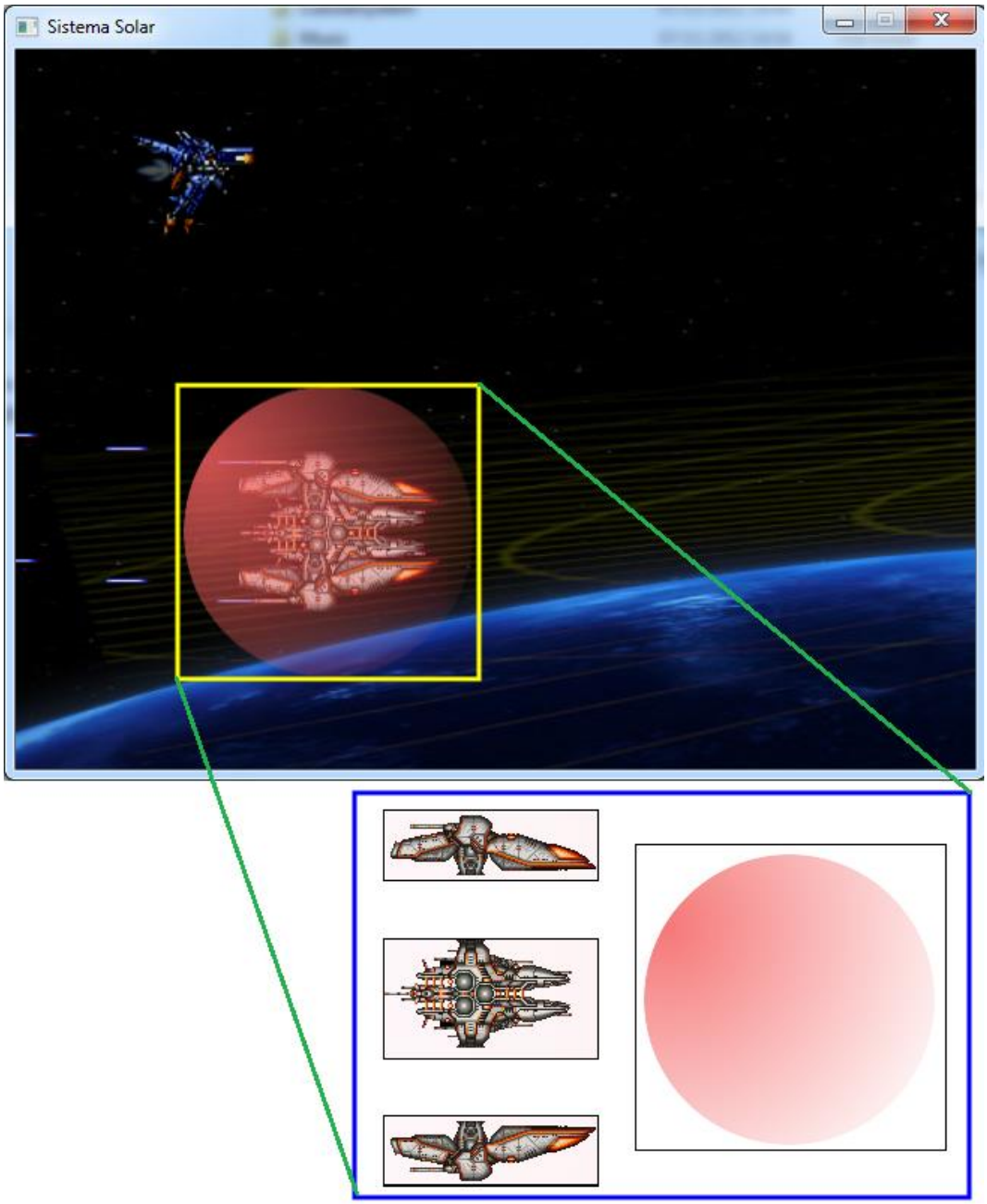


Figura 5.1 – Exemplo de uma nave construída no motor através da composição de elementos semi-autônomos.

O fluxo principal de execução (i.e., o *GameLoop*) de aplicações construídas no motor CppPlay é bastante simples e pode ser conferido na Figura 5.2.

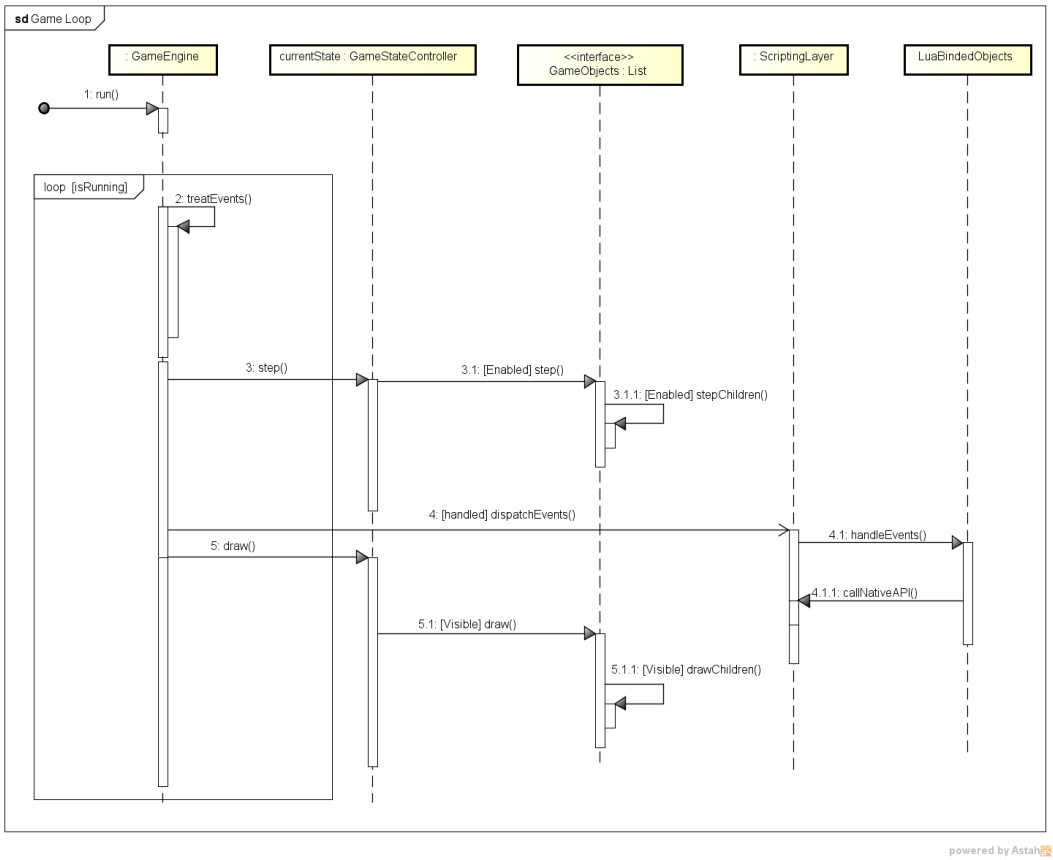


Figura 5.2 – Fluxo principal de execução de aplicações construídas no motor *CppPlay*.

Uma arquitetura simples, além de ser facilmente compreendida, permite que otimizações pontuais possam ser implementadas sem a necessidade de utilizar técnicas muito obscuras: o mecanismo apresentado na Figura 5.3 ilustra uma dessas otimizações onde pudemos observar um ganho de até 380%, em termos de *Frames-Per-Second*, simplesmente evitando que novas verificações sejam feitas para códigos de tratamento de evento já detectados como inexistentes.

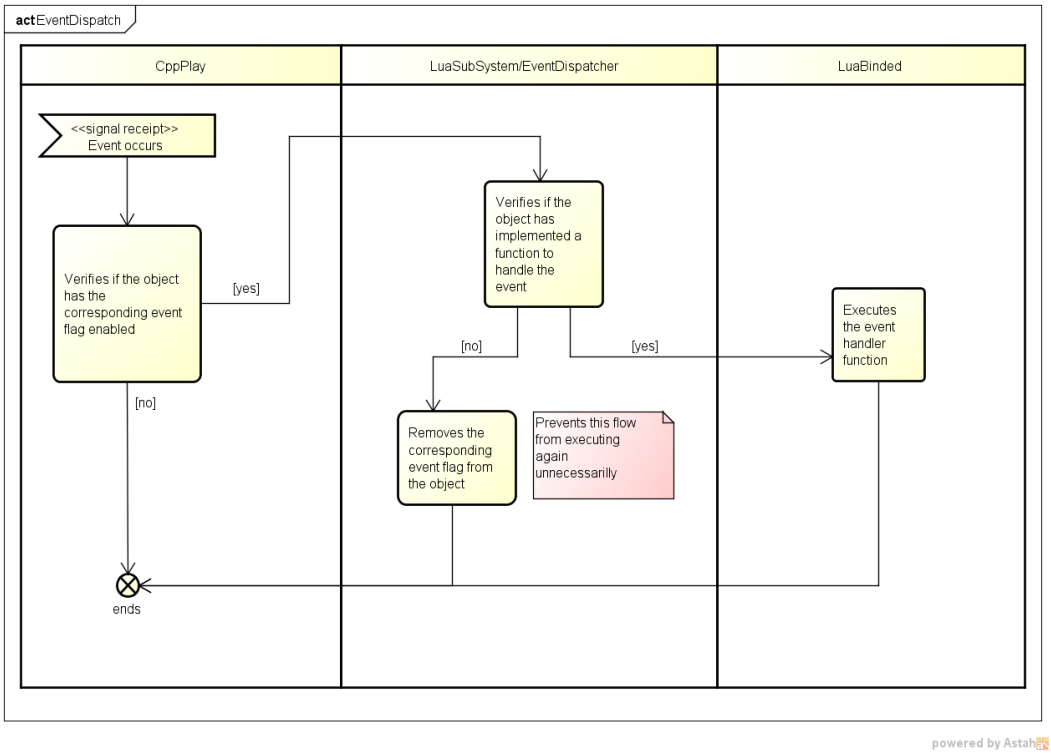


Figura 5.3 – Fluxo para o tratamento de eventos para a camada de *scripting* do motor *CppPlay*.

O motor também conta com uma extensa documentação de código compilada pela ferramenta *Doxygen*[52]. Esta documentação contém diagramas UML, prototipação e descrição de métodos e propriedades C++, além de indicadores visuais que destacam quais elementos escritos em C++ são disponibilizados para a camada de scripts Lua.

5.2 Novas funcionalidades para o motor CppPlay

Adicionamos ao motor duas importantes funcionalidades com o objetivo de aumentar a produtividade e o potencial da plataforma de jogos educativos que aqui propomos:

1. **Suporte a arquivos no formato Tiled (.tmx)**:

   Tiled[53] é uma ferramenta *open source* bastante popular que permite construir de maneira visual e intuitiva cenários para jogos 2D. Na Figura 5.4 podemos ver a ferramenta sendo utilizada para desenhar um dos cenários do nosso jogo prova de conceito, com diversas camadas de imagens, objetos demarcando locais inacessíveis e outras meta-informações que podem ser interpretadas das mais diversas maneiras pelo jogo desenvolvido (e.g., tele-transportes, objetos de interação, portas, escadas, ítems escondidos, etc).

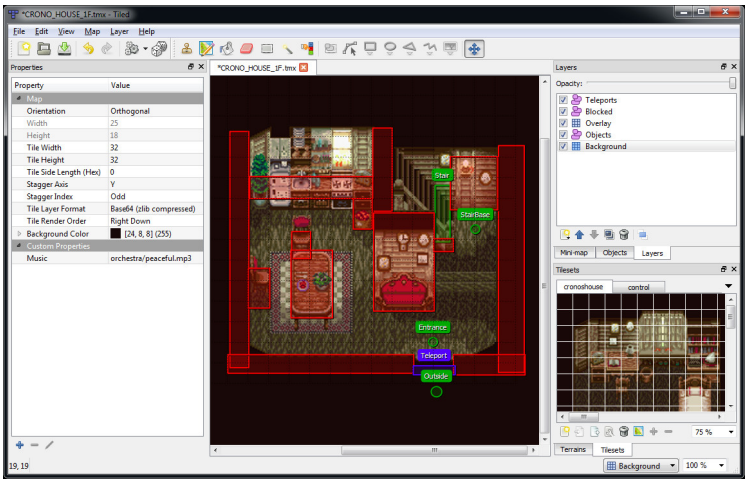


Figura 5.4 – Ferramenta *Tiled* sendo utilizada na construção de um dos cenários de nosso jogo prova de conceito.

A possibilidade de construir visualmente os cenários aumenta incrivelmente a produtividade de nossa ferramenta, pois de outra maneira o posicionamento de cada objeto e o preenchimento de suas propriedades, precisaria ser feito programaticamente utilizando o método de tentativa e erro, dependendo também da execução do jogo para visualização dos resultados.

Para incluir esta funcionalidade, a biblioteca C++ *tmxparser*[54] foi incorporada ao projeto, em seguida a classe *GameStateController* passou a conter um novo construtor capaz de receber o caminho para um arquivo *.tmx* e transformar seus componentes (imagens, sensores e objetos abstratos com meta-informação) em componentes correspondentes no nosso motor.

2. **Componente WebBrowser**:

   Desenvolvemos um componente *WebBrowser* que, com auxílio do *Chromium Embedded Framework*[55], utiliza o motor *Google Chromium*(i.e., porção *open source* do *Google Chrome*)[56] para consumir tecnologia Web. Este componente permite instanciar páginas Web dentro do jogo, utilizando qualquer tecnologia compatível com o *Browser* original, de maneira indistinguível (quando desejável) de qualquer outro elemento de jogo. Desta forma, um leque imensurável de bibliotecas e *frameworks* pode ser incorporado a nossa solução. Para citar alguns exemplos:

   - **HTML5**: Versão mais atual da linguagem HTML (*HyperText Markup-Language*) que tem por objetivo suprir as lacunas existentes

em versões anteriores da linguagem no que diz respeito ao desenvolvimento de *Web Applications*[57].

- **JavaScript**: JavaScript é a linguagem de scripts padrão para Web. JavaScript pode tornar as páginas Web mais dinâmicas, fazendo com que modificações em seu conteúdo possam ser feitas sem a necessidade de carregar uma nova versão do documento (*Dynamic HTML/DHTML*) e que dados possam ser carregados e enviados de forma assíncrona (*AJAX - Asynchronous JavaScript and XML*)[58].
- **CSS**: *Cascading Style Sheets* (CSS) representa um simples mecanismo para adicionar estilo (e.g., fontes, cores, espaçamento) à documentos Web[59].
- **WebGL**: WebGL é um padrão Web *cross-platform*, *royalty-free* para uma API gráfica 3D baseada em *OpenGL ES 2.0* e exposta através do elemento *Canvas* de HTML5. Desenvolvedores familiarizados com *OpenGL ES 2.0* conseguem enxergar WebGL como uma API bastante similar em termos de construtos e semântica. A diferença consiste apenas de algumas concessões feitas para respeitar os mecanismos de gerenciamento de memória utilizados em linguagens como JavaScript[60].
- **Streaming Multimídia**: *Streaming* possibilita que arquivos multimídia pesados (e.g., áudio e vídeo) possam ser executados sob demanda, sem que seja necessário baixar todo o arquivo antes de se iniciar a reprodução. Em outras palavras, o suporte à essa tecnologia nos permite acessar, de maneira eficiente, serviços ricos em conteúdo como *YouTube*[61] e *Coursera*[62].
- **Bibliotecas e Frameworks**:
  - **AngularJS**: *AngularJS* nos possibilita escrever *client-side web applications* como se tivéssemos um navegador mais inteligente. Além de utilizar HTML de maneira convencional para formatar documentos, *AngularJS* permite estender a sintaxe da linguagem para podermos expressar de forma mais clara e sucinta os componentes de uma aplicação. O *framework* sincroniza automaticamente os dados da interface com o usuário (*view*) com objetos JavaScript (*model*) através de uma comunicação de duas vias (*2-way data binding*). Para facilitar a estruturação de uma aplicação e torná-la mais fácil de testar, *AngularJS* ensina ao navegador como realizar injeção de dependências e inversão de controle[63].
  - **Bootstrap**: É um *framework* elegante, intuitivo e poderoso para desenvolver *front-ends* de maneira mais rápida e fácil[64].

Oferece uma padronização de estilo e componentes avançados para construção de interface com o usuário e é extremamente popular.

- **D3.js**: É uma biblioteca JavaScript para manipulação de documentos baseados em dados. *D3.js* possibilita dar vida aos dados utilizando *HTML*, *SVG* e *CSS*. Seu foco em padrões Web permite que você tenha acesso completo as funcionalidades de navegadores modernos sem depender de um *framework* proprietário, combinando poderosos componentes de visualização com uma abordagem *data-driven* para manipulação do *DOM* (*Document Object Model*)[65]. É bastante utilizada em aplicações que precisam desenhar gráficos estatísticos e matemáticos.
- **MathJax**: É um motor *open source* de visualização para *Latex*, *MathML* e *AsciiMath* que funciona em todos os navegadores modernos. Foi projetado com o objetivo de consolidar os avanços recentes de tecnologias Web em uma simples e definitiva plataforma de matemática para a rede. Não requer nenhuma configuração por parte do usuário (não depende da instalação de nenhum *plugin* ou pacote de *software*), portanto, o desenvolvedor pode criar documentos Web contendo fórmulas matemáticas e ter a confiança de que os usuários poderão visualizá-las sem qualquer empecilho[66].

Mais do que renderizar páginas, este componente permite uma comunicação bilateral entre as plataformas, isto é, as páginas Web podem executar código do motor de jogos utilizando scripts Lua e os jogos desenvolvidos no motor podem executar JavaScript nas páginas Web fazendo com que toda a lógica se alinhe. Na Figura 5.5 podemos ver lado a lado um elemento de jogo sendo executado por nosso motor e pelo navegador *Google Chrome*. Neste exemplo, as características de estilo da página Web foram mantidas propositalmente para demonstrar que a página é processada dentro do jogo como se tivesse sido aberta pelo próprio navegador. No entanto, quando criamos componentes para um jogo, podemos desejar alterar o estilo visual da página para que ela se confunda com o estilo artístico do próprio jogo, ou seja, pareça indistinguível de qualquer outro elemento não Web, como podemos ver na Figura 5.6. Repare que as coordenadas do mini-mapa correspondem a posição do herói na cena do jogo. Esta informação é atualizada dinamicamente conforme o personagem se movimenta pelo mundo.

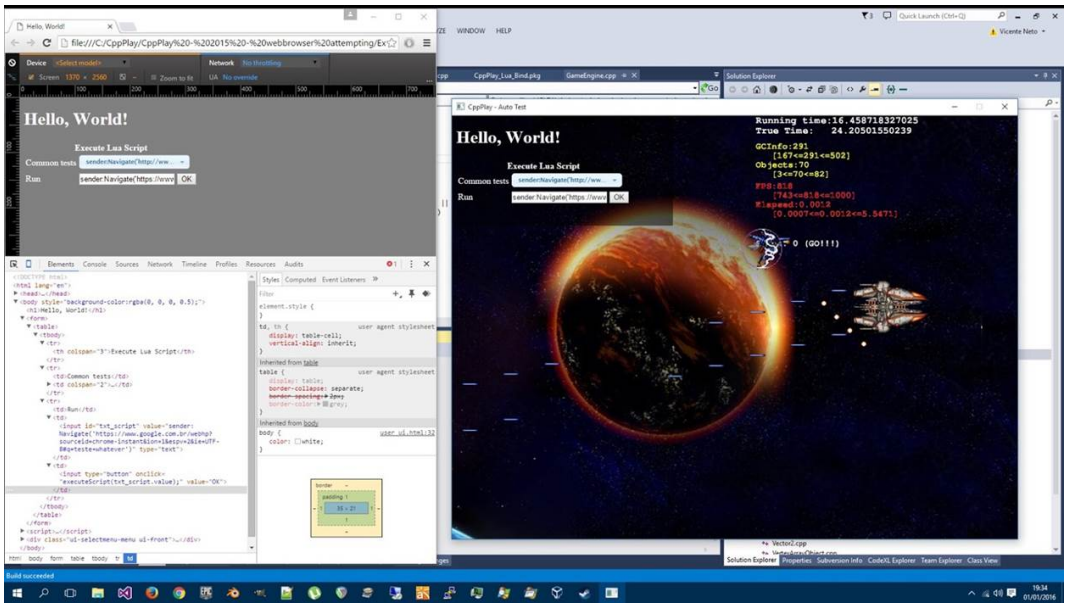


Figura 5.5 – Elemento de jogo sendo executado lado a lado pelo motor *CppPlay* e pelo navegador *Google Chrome*.

Figura 5.6 – A lista de tarefas no topo e o mini-mapa apresentado no canto inferior direito da cena, são na verdade páginas Web sendo renderizadas pelo componente *WebBrowser*.

5.3 O desenvolvimento de um meta-motor para JRPGs

A arquitetura de um motor JRPG pode ser representada por um conjunto bem definido de sub-sistemas[43] (Figura 5.7)

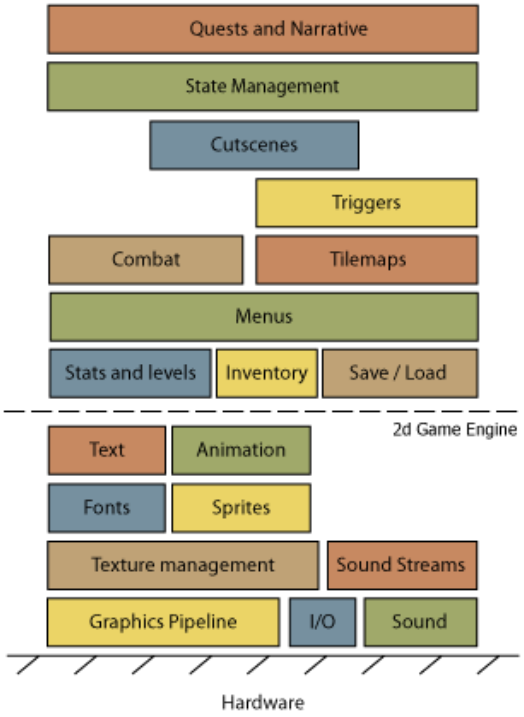


Figura 5.7 – Sub-sistemas de um motor *JRPG*.

Em nossa abordagem, decidimos implementar a fatia superior dessa arquitetura como um meta-motor (i.e., um motor executado por outro motor) escrito em Lua e apoiado por tecnologias Web. Esta é uma abordagem bastante comum no desenvolvimento de jogos: as funcionalidades de baixo-nível que necessitam de alto desempenho e poucas modificações são encapsuladas em um núcleo otimizado de código nativo (normalmente C/C++) enquanto que as funcionalidades alto-nível ficam sob responsabilidade de uma camada de scripts, capaz de comandar os mecanismos de alto desempenho através de uma interface mais simples, amigável e manutenível, oferecendo desta forma mais produtividade sem comprometimento de poder computacional. Os módulos que compreendem este meta-motor são agrupados em 3 pacotes distintos: *Algorithm*, *Controllers* e *Controlled Classes*.

- **Algorithm**: Compreende estruturas de dados e funções auxiliares (açúcares sintáticos) para o meta-motor (Figura 5.8).

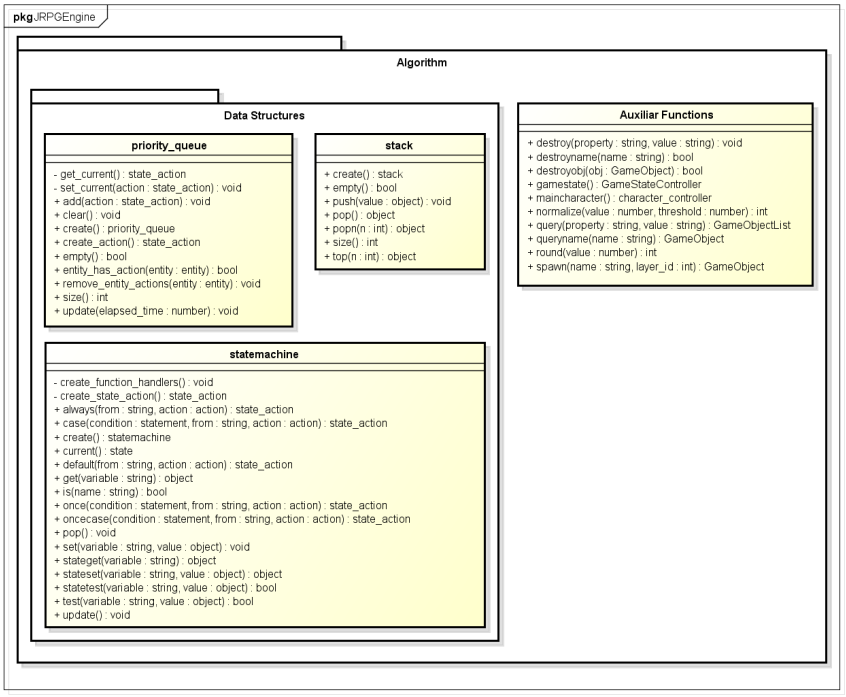


Figura 5.8 – Pacote *Algorithm* da arquitetura do meta-motor *JRPG*.

– **Data Structures**

  * **stack**: Implementa uma estrutura de pilha, útil por exemplo, para simular transições recursivas entre estados e cenas.
  * **priority_queue**: Implementa uma fila de prioridades, bastante útil durante combate baseado em turnos,onde os personagens com maior velocidade de combate recebem vantagem na iniciativa de suas ações.
  * **statemachine**: Implementa uma *FiniteStateMachine*, com regras de transição e ação, muito conveniente para implementação de inteligência artificial em *NPCs*.

– **Auxiliar Functions**

  * **destroy(property, value)**: Destrói objetos que contenham a propriedade *property* com valor igual à *value*.
  * **destroyname(name)**: Destrói objetos que contenham a propriedade *name* com valor igual à *value*.
  * **destroyobj(obj)**: Destrói um objeto verificando antes se o objeto é válido (i.e., *obj != nil*).
  * **gamestate()**: Obtém uma referência para o *GameStateController* sendo executado no nomento.
  * **maincharacter()**: Obtém uma referência para o personagem principal (i.e., personagem controlado pelo jogador) na cena em execução.

* **query(property, value)**: Obtém uma lista de objetos que contenham a propriedade *property* com valor igual à *value*.
* **queryname(property, value)**: Obtém um objetos que contenha a propriedade *name* com valor igual ao nome fornecido.
* **spawn(name, layer_id)**: Instancia um objeto construído pelo script "*scripts/entities/<name>.lua*" e o adiciona em uma camada de objetos de *z-order* igual a *layer_id*. É possível construir várias instâncias de um mesmo tipo sucedendo o parâmetro *name* com um "." seguido de uma numeração (e.g., soldier.1, soldier.2, soldier.3).
* **normalize(value, threshold)**: Realiza uma normalização de valores conforme as seguintes regras:

$$result = \begin{cases} \text{-1}, & \text{if } value <= -threshold \\ 0, & \text{if } abs(value) < threshold \\ 1, & \text{if } value >= threshold \end{cases}$$

* **round(value)**: Arredonda um número para o valor inteiro mais próximo.

- **Controllers**: Compreende controladores para os diferentes sub-sistemas do motor JRPG (Figura 5.9).

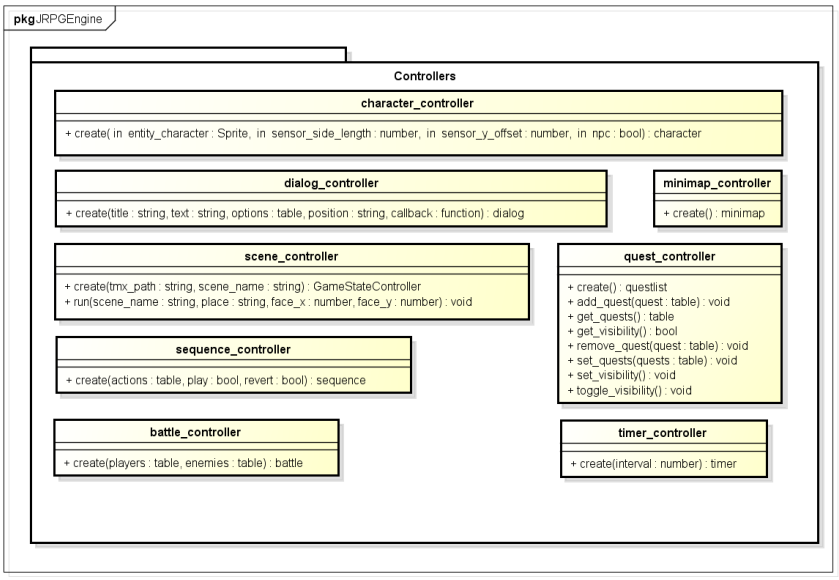


Figura 5.9 – Pacote *Controllers* da arquitetura do meta-motor *JRPG*.

  - **character_controller**: Permite construir e manipular os personagens do jogo que podem ser controlados por jogadores humanos ou pelo próprio computador (i.e., *NPCs ou Non-Player-Characters*).

– **dialog_controller**: Permite controlar diálogos no estilo JRPG: janelas típicas de texto, com a possibilidade de múltiplas escolhas e posicionamento conforme os pontos cardeais (i.e., centro, norte, sul, leste, oeste, noroeste, etc). Em nossa plataforma, as janelas de diálogo permitem utilizar: fórmulas matemáticas sofisticadas, sintaxe *HTML* e *emoticons* (i.e., ícones que expressam a emoção referente à uma fala). Na Figura 5.10, podemos ver um exemplo de diálogo construído com esse componente.

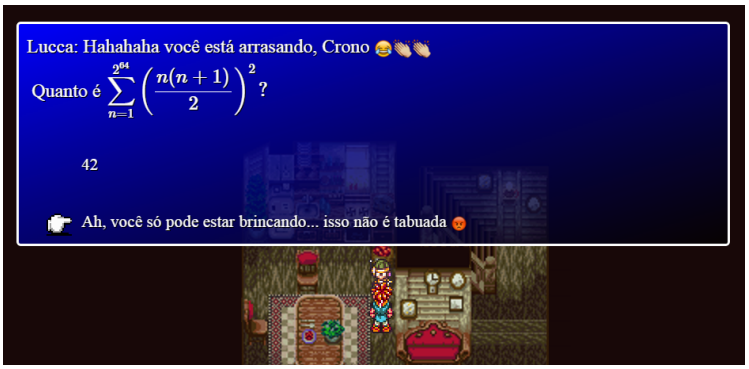


Figura 5.10 – Exemplo de diálogo contendo fórmulas matemáticas, HTML e *emoticons*.

– **scene_controller**: Permite controlar transições entre as cenas de jogo, música tema, criação automática de sensores de bloqueio (demarcação de áreas não acessíveis pelo jogador), geração e posicionamento automático de entidade. Tudo através das meta-informações fornecidas no editor de mapas *Tiled*.
– **sequence_controller**: Permite construir e controlar sequência de ações (e.g., movimentar um personagem, executar uma função, aguardar um intervalo de tempo). Tais sequências são muito úteis para implementação de *cutscenes*.
– **battle_controller**: Permite iniciar uma batalha entre os personagens do jogador e os inimigos controlados pelo computador. O componente controla o andamento da batalha em turnos, transformando os comandos do jogador e as decisões do computador em ações dos respectivos personagens e analisando se as condições de término foram atingidas: o que basicamente acontece quando todos os integrantes de um time forem eliminados. Como nossa plataforma é educacional, utilizamos, como mecânica para avaliação das ações de batalha, um mecanismo de resolução de equações: caso o jogador acerte a resposta para a equação sua ação será bem sucedida e do contrário não (Figura 5.11).

Figura 5.11 – Mecânica de batalhas utilizando resolução de equações matemáticas.

Ao término da batalha, o componente exibe um gráfico de desempenho do jogador bem como os desafios apresentados e as respostas do jogador durante o combate (Figura 5.12).

Figura 5.12 – Após uma batalha, o meta-motor apresenta uma análise de desempenho e os desafios e respostas do jogador durante o combate.

- **quest_controller**: Permite controlar uma lista de *quests* (i.e., missões que o jogador deve cumprir para avançar no jogo ou conseguir algum prêmio), possibilitando a inclusão, atualização e remoção de entradas nessa lista e também sua visibilidade na interface com o usuário.
- **timer_controller**: Permite construir e controlar componentes do tipo *timer*. Componentes *timer* são utilizados sempre que uma

determinada ação precise considerar a passagem de um intervalo de tempo (e.g., duração do efeito de uma poção de invisibilidade, tempo de espera até a chegada de um exército aliado).

– **minimap_controller**: Permite construir e controlar um componente do tipo *minimap*. Este componente serve para exibir uma versão miniatura do mapa externo do jogo, possibilitando ao jogador ter uma visualização global do mundo e saber qual a localização geográfica atual de seu personagem.

- **Controlled Classes**: Compreende as classes produzidas pelos *Controllers* que podem ser diretamente manipuladas pelo desenvolvedor a fim de construir JRPGs personalizados (Figura 5.13).

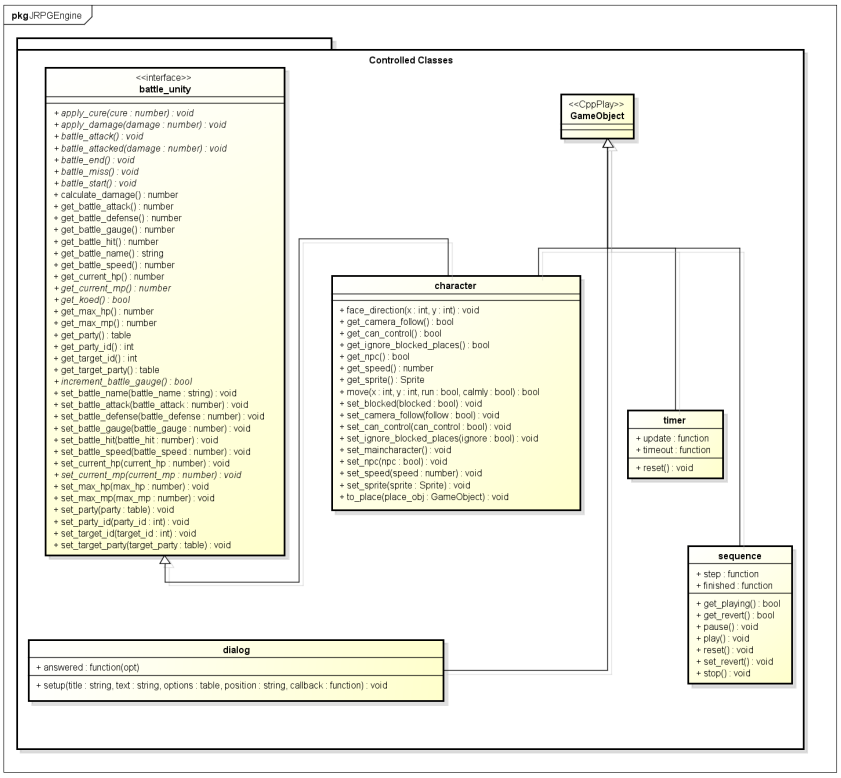


Figura 5.13 – Pacote *Controlled Classes* da arquitetura do meta-motor *JRPG*.

– **character**: Permite manipular os personagens, movimentando-os de um lugar para o outro, definindo se são controlados pelo jogador ou pelo computador, se podem ser controlados em um dado momento, se podem atravessar lugares fisicamente bloqueados, se devem ser seguidos pela câmera. Também é responsável por atualizar as animações de *Sprite* conforme o estado atual do personagem.
– **battle_unity**: Interface implementada por todo *character* que possibilita consultar e atualizar os valores de habilidades do personagem

(e.g., ataque, defesa, velocidade), atualizar estados e executar ações de acordo com o fluxo de uma batalha ativa.

- **timer**: Representa uma contagem regressiva que, ao atingir o valor 0 (zero), executa um evento *timeout*. Pode ser utilizado, por exemplo, para explodir uma bomba ou aplicar efeitos que se repetem de tempo em tempo: dano por veneno, cura por regeneração.
- **sequence**: Representa uma sequencia de ações (e.g., andar da posição atual até a posição (17, 14), correr até a posição (10, 1), fazer uma chamada de função e retornar o controle para o jogador). Sua interface permite controlar a execução da sequência e executar códigos personalizados quando a sequência for atualizada (implementando o método *update*) ou finalizada (implementando o método *finished*).
- **dialog**: Permite manipular uma janela de diálogos, atualizando o texto e opções de múltipla escolha conforme o andamento de uma conversa e respondendo as escolhas do jogador através do método *answered*.

Nesta sub-seção apresentamos o meta-motor JRPG sob uma perspectiva abstrata e, de certa forma, superficial. Por mais que tentemos simplificar uma arquitetura desta natureza, ainda teremos internamente um conjunto de sub-sistemas e funcionalidades bastante complexo e, como pudemos ver sob a ótica pedagógica desta pesquisa, apresentar uma grande quantidade de informação fora de contexto prático pode ser uma abordagem pouco didática. Por essa razão, acreditamos que uma melhor estratégia para o conhecimento da plataforma aqui proposta, consista de uma abordagem ""mão na massa , isto é, aprender a utilizá-la gradativamente com o auxílio de um ambiente integrado de desenvolvimento (*IDE - Integrated Development Environment*) e um projeto de exemplo parcialmente construído que pode ser analisado, modificado e expandido pelo desenvolvedor.

5.4 Integrando nossa plataforma em um *IDE - Integrated Development Environment*

Um ambiente integrado de desenvolvimento (i.e., *IDE - Integrated Development Environment*) oferece ao desenvolvedor uma série de facilidades que contribuem tanto em termos de produtividade quanto em qualidade do produto em desenvolvimento. Para cumprir o papel desse ambiente em nossa plataforma, escolhemos a ferramenta *LDT - Lua Development Tools*[67], que foi construída com base na arquitetura do *Eclipse* (um dos ambientes de desen-

volvimento mais populares e poderosos do mercado)[68]. Os principais motivos para a escolha deste ambiente foram a sua natureza *open source*/multiplataforma e o fato de oferecer ao desenvolvedor Lua ferramentas similares as encontradas no desenvolvimento de projetos escritos em linguagens estáticas. Para citar algumas dessas ferramentas:

- **Script Explorer**: Permite organizar o projeto em uma hierarquia de pastas e arquivos, facilitando a procura e identificação de diferentes tipos de artefatos, bem como a criação de novos elementos (Figura 5.14).

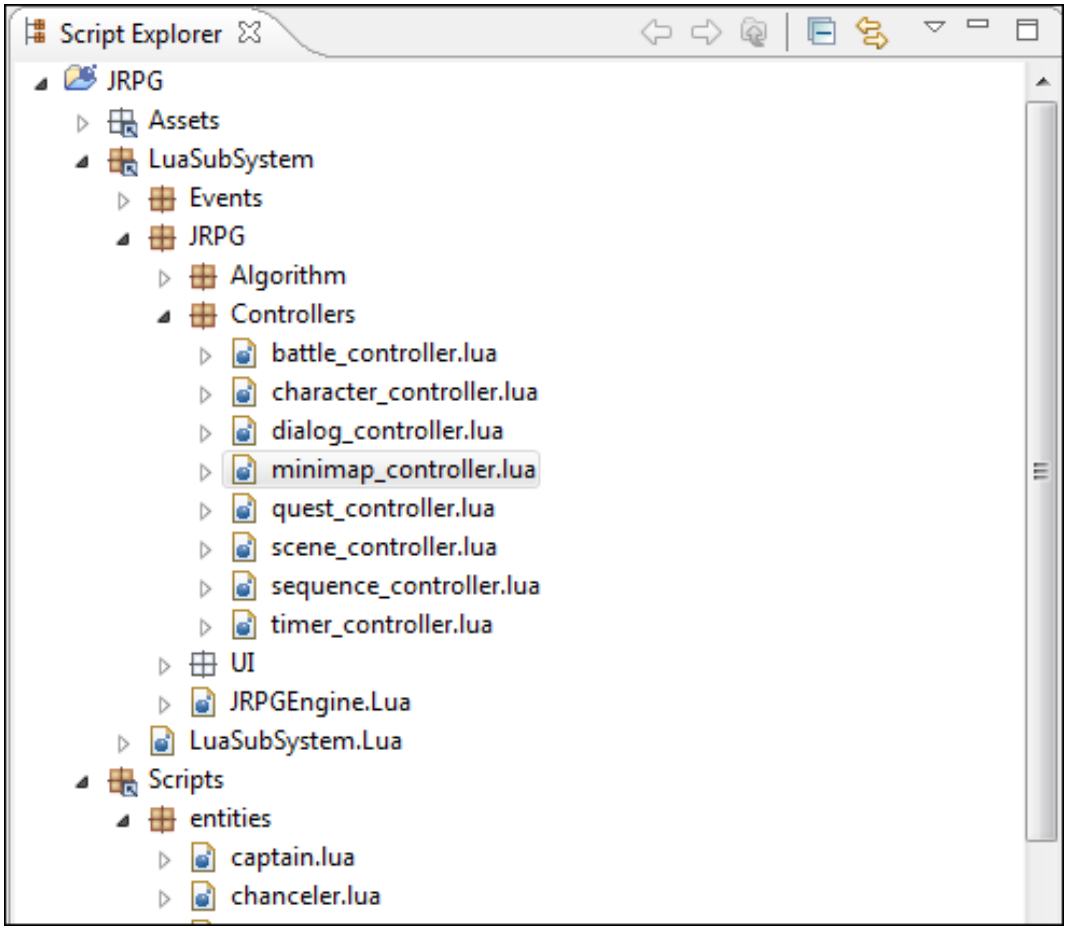


Figura 5.14 – *Script Explorer* exibindo alguns artefatos do nosso projeto prova de conceito.

- **Syntax Coloring**: Colore automaticamente o código fonte facilitando a identificação de comentários, tipos, funções e palavras reservadas (Figura 5.15).
- **Code Assistance**: Oferece auto-completação de código sensível a contexto (i.e., identifica as variáveis declaradas no escopo). Também é capaz de inferir o tipo de uma variável e sua API com o auxílio da documentação do código que a produziu ou da documentação embutida no próprio

ambiente de execução (e.g., documentação Lua 5.1 que já acompanha a ferramenta *LDT*). Para acionar esta funcionalidade basta pressionar *Ctrl+Space* no editor de código (Figura 5.15)[69].

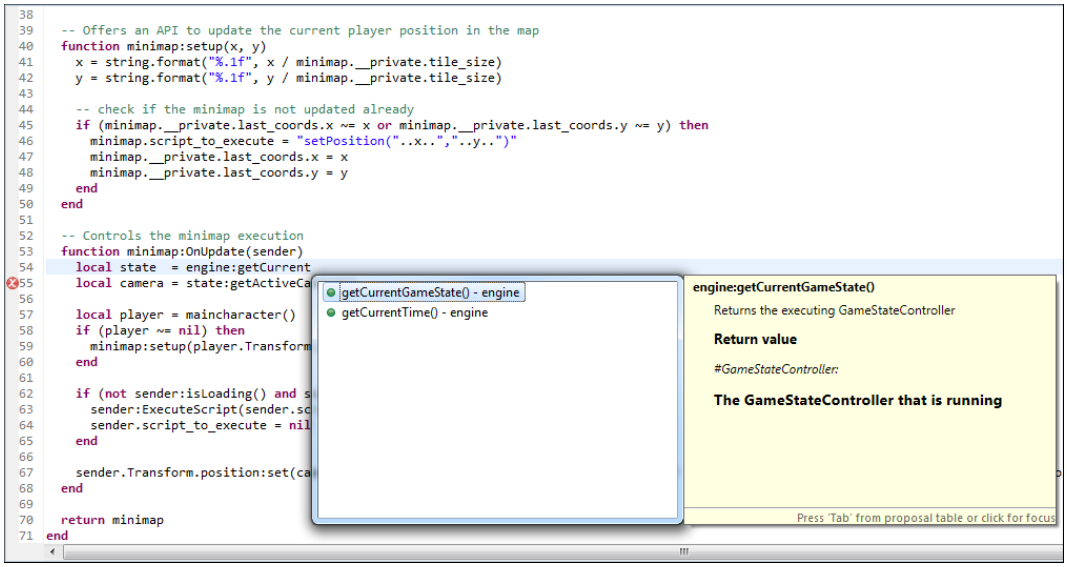


Figura 5.15 – Editor de código exibindo as funcionalidades de *Syntax Coloring* e *Code Assitance*.

- **Code Template**: Permite utilizar *templates* de código pré-configurados, bastante úteis para facilitar a "digitação" de estruturas de repetição ou de tabelas com leiautes pré-definidos. Também é acionada através do pressionamento de *Ctrl+Space* no editor de código (Figura 5.16).

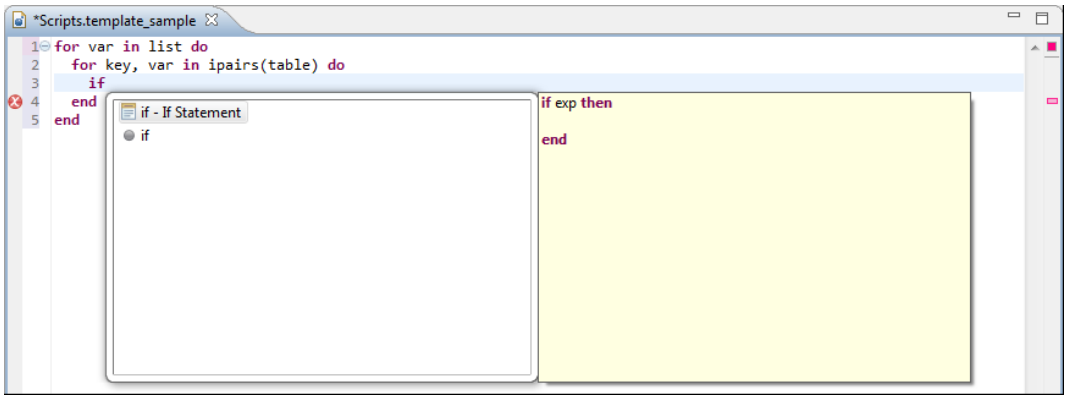


Figura 5.16 – Alguns exemplos de *Code Templates*.

- **Error Markers**: Permite a detecção automática de erros ainda em tempo de desenvolvimento. Os erros detectados através desta funcionalidade ficam demarcados no editor de código e são detalhados na aba *Problems* (Figura 5.17).

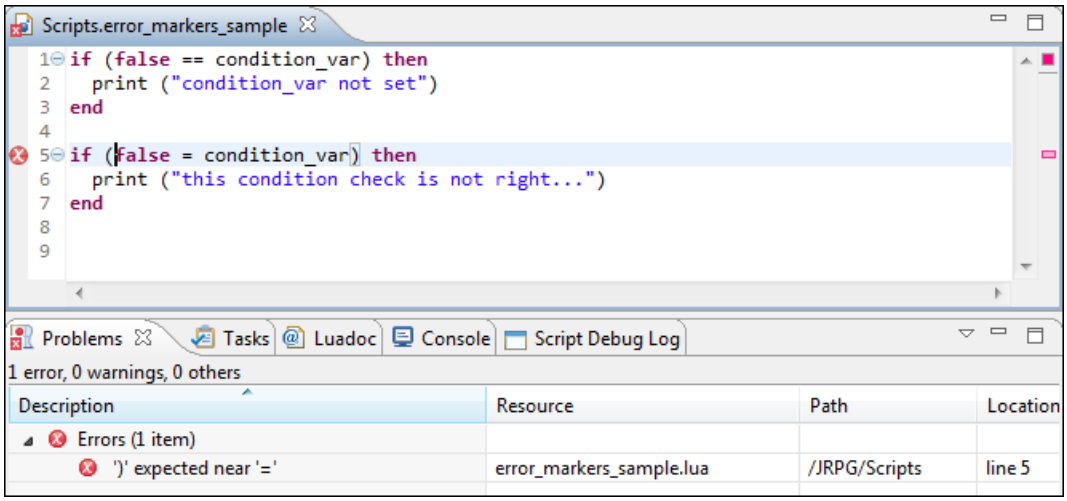


Figura 5.17 – Ferramenta *Error Marker* detectando erros ainda em tempo de desenvolvimento.

- **Debugger**: Permite executar os projetos em modo de depuração. Para que isso seja feito, basta seguir os 3 passos descritos na Figura 5.18. A ferramenta *Debugger* oferece suporte a *breakpoints* (Figura 5.19a), navegação pela pilha de chamadas (Figura 5.19b), inspeção de variáveis (Figura 5.19c), avaliação de expressões (Figura 5.19d) e console interativo (Figura 5.19e).

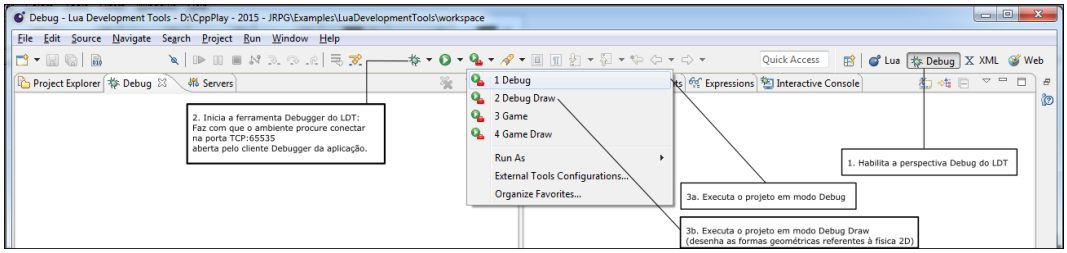
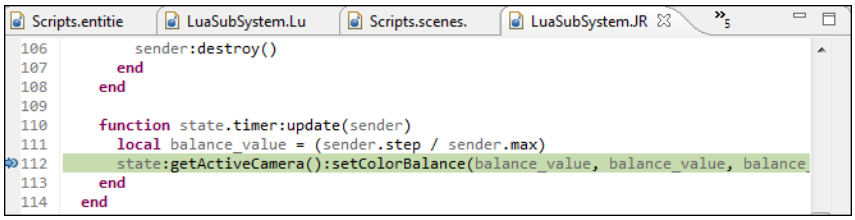
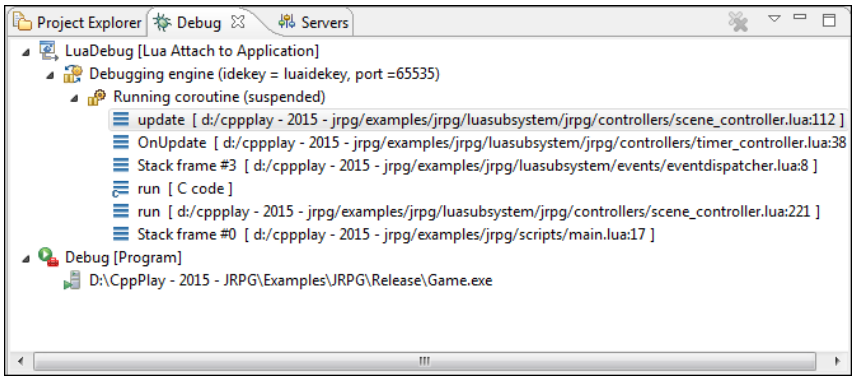


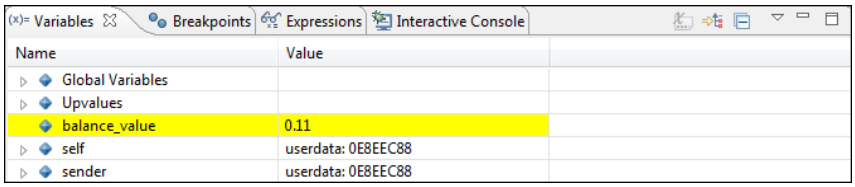
Figura 5.18 – Passos para a execução de um projeto em modo de depuração.



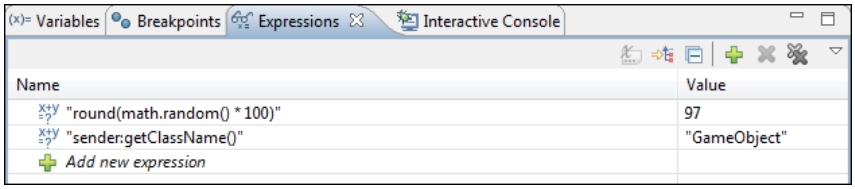
(a) *Breakpoints*.



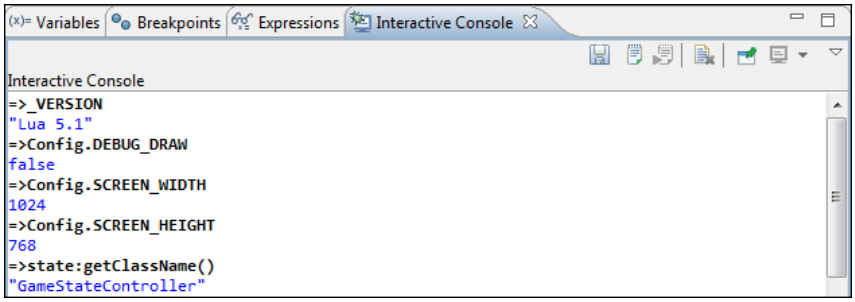
(b) Pilha de chamadas.



(c) Inspeção de variáveis.



(d) Avaliação de expressões.



(e) Console interativo.

Figura 5.19 – Funcionalidades oferecidas pela ferramenta *Debugger*.

5.5 Utilizando o editor de mapas *Tiled* como extensão de nossa plataforma

Como vimos anteriormente, o motor *CppPlay* foi modificado para dar suporte à arquivos no formato do editor de mapas *Tiled* (*.tmx)[53]. Uma

vez que o desenvolvedor tenha essa ferramenta instalada em seu computador, podemos facilmente enxergá-la como uma extensão de nossa plataforma, pois a maneira como o ambiente de desenvolvimento *LDT - Lua Development Tools* associa as extensões de arquivo aos seus respectivos editores, faz com que a ferramenta correta seja automaticamente acionada quando o desenvolvedor ordenar a abertura de um mapa deste formato incluído no projeto (Figura 5.20).

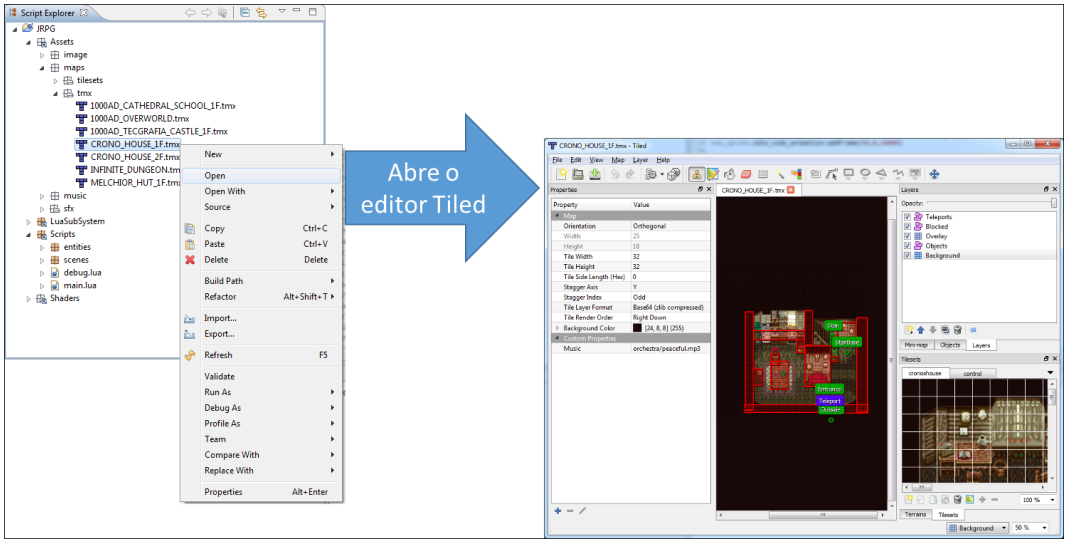


Figura 5.20 – A abertura de um arquivo de mapa utilizando a ferramenta *Script Explorer* aciona automaticamente o editor *Tiled*.

O editor *Tiled* é bastante intuitivo e através do menu *Help→Documentation* é possível ter acesso ao tutorial de introdução da ferramenta. Além disso, podemos encontrar facilmente dezenas de tutoriais, inclusive em vídeo, realizando uma simples procura em motores de busca.

Devemos apresentar, no entanto, o mecanismo que utilizamos em nossa plataforma para automatizar tarefas de programação utilizando meta-informação nos campos de propriedades do mapa. O mecanismo funciona atribuindo nomes especiais aos campos de propriedades dos objetos, diminuindo a necessidade de escrever código de programação Lua. A seguir, apresentamos as palavras reservadas que já são suportadas em nossa solução:

- **Music**: Quando esta propriedade é atribuida ao mapa, o meta-motor JRPG tocará automaticamente o arquivo de áudio informado como música tema do cenário (Figura 5.21).

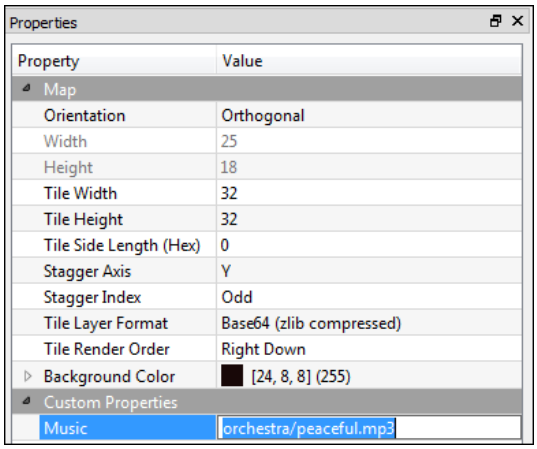


Figura 5.21 – Através da propriedade *Music* de um mapa é possível informar o caminho para um arquivo que será utilizado como música tema do cenário.

- **Blocked**: Quando esta propriedade for atribuída a um objeto, o meta-motor irá interpretá-lo como uma barreira física, ou seja, o lugar ocupado pelo objeto será bloqueado para o jogador (Figura 5.22).

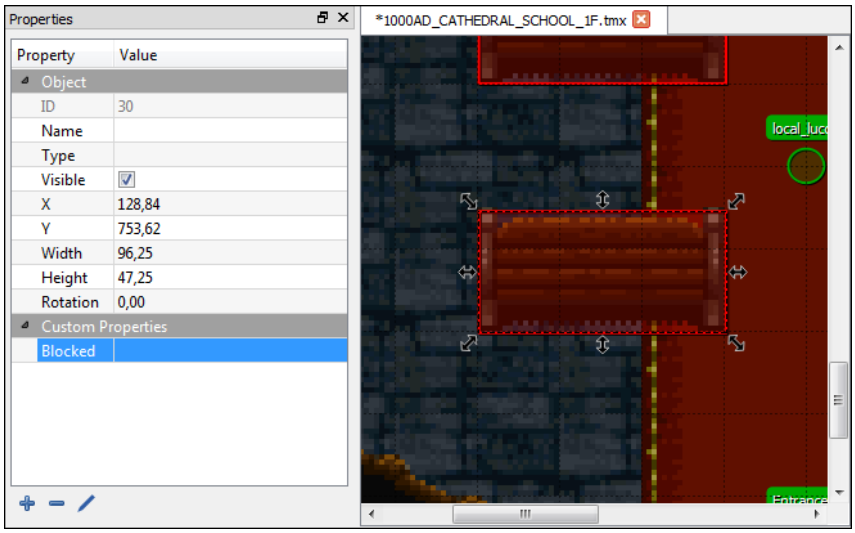


Figura 5.22 – Através da propriedade *Blocked* é possível marcar objetos como barreiras físicas para o jogador.

- **Spawn**: Quando esta propriedade for atribuída a um objeto, o meta-motor irá interpretá-lo como um *Spawner* (i.e., um objeto que será responsável por instanciar entidades automaticamente). O meta-motor age substituindo cada um desses objetos por uma entidade cujo script de construção detenha o mesmo nome que a propriedade *Name*. No exemplo, o objeto "local_crono" será substituído por uma entidade construída pelo script "scripts/entities/local_crono.lua" (Figura 5.23).

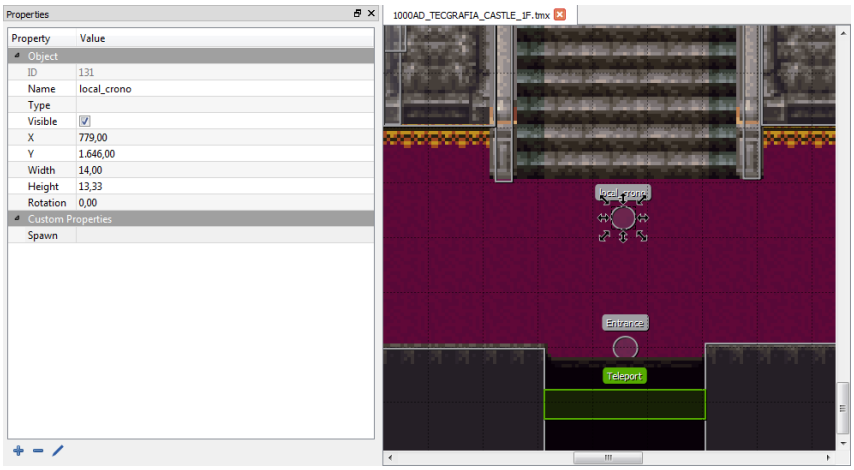


Figura 5.23 – Exemplo de um objeto Spawn configurado para instanciar automaticamente a entidade "local_crono".

O desenvolvedor pode utilizar vários *Spawners* de uma mesma entidade, bastando para isso suceder o nome do objeto com um "." seguido de uma numeração (e.g., soldier.2).

- **Action**: Quando esta propriedade for atribuída a um objeto, o meta-motor irá interpretá-lo como um objeto interativo. Para interagir com um desses objetos, basta que o personagem esteja em contato com o objeto e que o jogador pressione a tecla *Space*. Quando isso ocorrer, o script informado como valor da propriedade *Action* será executado (Figura 5.24).

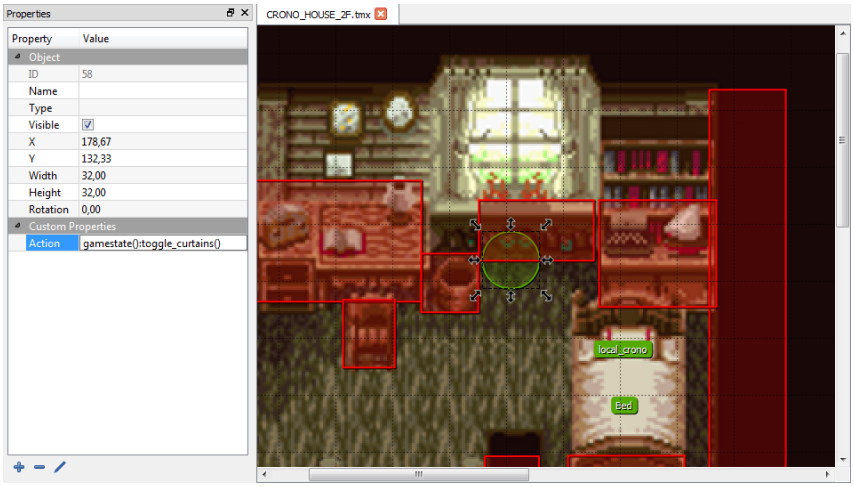


Figura 5.24 – Exemplo da propriedade *Action* possibilitando que o jogador abra e feche as cortinas.

- **AutoAction**: Quando esta propriedade for atribuída a um objeto, o meta-motor irá interpretá-lo como um objeto automaticamente interativo. Quando o personagem entrar em contato com o objeto, o script

informado como valor da propriedade *AutoAction* será executado (Figura 5.25). Esta propriedade é normalmente utilizada para representar *triggers* (e.g., armadilhas, detector de intrusos, objetos de transição de cena).

Figura 5.25 – Exemplo da propriedade *AutoAction* possibilitando que o jogador "desça as escadas" ao encostar no objeto *Stair*.

O desenvolvedor pode introduzir novas palavras reservadas em seu jogo e utilizar as funções de consulta de propriedades fornecidas pelo pacote *Algorithm* (e.g., *query*, *queryname*) para operar sobre os objetos desejados.

6
Resultados

Apresentamos neste capítulo um pequeno jogo construído em nossa plataforma com o intuito de demonstrar o potencial de nossa proposta, tanto para o jogador (i.e., quem vai consumir os jogos educativos desenvolvidos com a nossa solução) quanto para o desenvolvedor de jogos dessa natureza.

6.1
Prova de conceito

O jogo que desenvolvemos é baseado no universo *Chrono Trigger*[46] e utiliza recursos artísticos proprietários da empresa que o criou (*Square-Enix*), isto é: seu propósito é estritamente didático, o jogo não poderá ser explorado além do ambiente acadêmico, sem fins lucrativos sob as restrições de *fair-use*[70].

As sub-seções a seguir subdividem a história do jogo de maneira semelhante a pequenos capítulos de um livro:

6.1.1
Uma bela manhã de prova

O jogo se passa no reino de Tecgrafia e tem início quando nosso herói acorda e se lembra de que é dia de prova oral de matemática (Figura 6.1a). Para dar sequência a esta *quest*, o jogador deverá guiar seu personagem até a *Catedral Escola*, na parte sul do continente. Porém, quando o jogador tenta sair de casa, seu personagem é surpreendido por uma colega de classe que faz questão de que pratiquem a tabuada antes de se apresentarem para a prova (Figura 6.1b).

Embora a prática de tabuada possa ser uma atividade tediosa, acreditamos que as ferramentas pedagógicas indentificadas na pesquisa consigam apresentar a atividade de forma mais eficiente e cativante que a usual: há um contexto prático (i.e., a prova oral iminente), um apelo emocional (i.e., o acordo e a preocupação de sua colega de classe com o seu bom desempenho) e uma estética de entretenimento com doses de humor e drama para envolver o jogador.

(a) Cena dos primeiros momentos do jogo, quando o herói acorda e se lembra que deve realizar uma prova oral de matemática.

(b) Cena de quando o herói é impedido de ir realizar a prova sem antes praticar a tabuada.

Figura 6.1 – Cenas do primeiro capítulo do jogo educativo.

Uma vez que tenha completado esta tarefa, o jogador estará livre para se dirigir até a *Catedral Escola*, localizada na parte Sul do continente (Figura 6.2).

Figura 6.2 – Podemos ver nesta cena o herói do jogo posicionado no mapa global próximo à *Catedral Escola* onde deverá realizar a prova.

6.1.2 O Rei decretou recesso

Quando o jogador se apresenta para realizar a prova, descobre que o Rei decretou recesso e que, por conta disso, não haveria aula naquele dia (Figura 6.3).

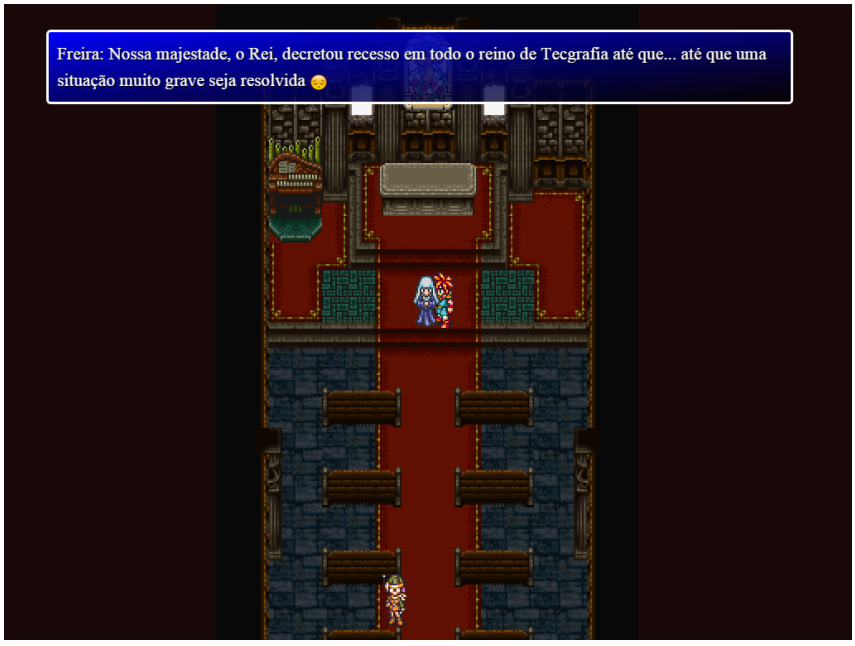


Figura 6.3 – Cena de quando a *Professora Freira* informa ao herói que não haverá mais prova devido a uma misteriosa situação que estaria preocupando o Rei.

A *Professora Freira* se mostra preocupada com a situação, explica o pouco que sabe sobre um mistério que envolve monstros e cristais, mas recomenda que o herói vá até o castelo se quiser descobrir o que realmente está acontecendo.

Neste capítulo do jogo, não realizamos nenhuma atividade matemática, mas utilizamos ferramentas cognitivas (e.g., senso de agência, introdução de mistério e do maravilhoso) para envolver o jogador em um nível emocional, exercitar sua imaginação e atribuir uma razão lógica para a realização de estudos e exercícios futuros.

6.1.3 Os 12 diamantes e seus herdeiros

Neste capítulo do jogo, apresentamos o mistério central da história através de uma adaptação do conto da divisão igualitária dos 12 diamantes e seus herdeiros (Figura 6.4).

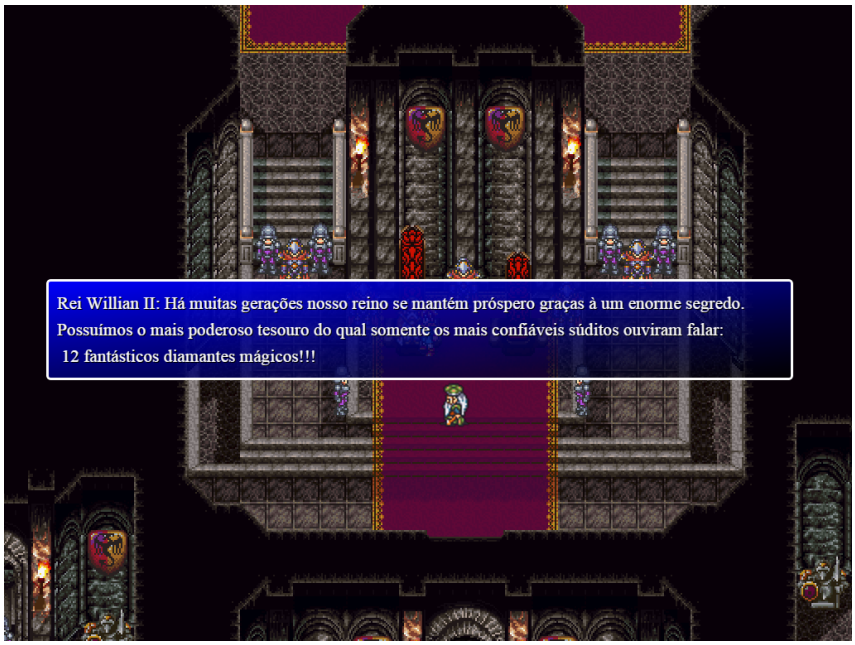


Figura 6.4 – Cena de quando o Rei descreve para o herói o problema da divisão igualitária dos 12 diamantes mágicos e seus herdeiros.

Em nossa adaptação, ao invés de todos os herdeiros morrerem (resultando em uma indefinição sobre a divisão dos diamantes por zero herdeiros), há a possibilidade de que um dos filhos do Rei, a *Princesa Nádia*, tenha sobrevivido e possa ser resgatada da *Masmorra Infinita*. Esta é uma missão que só o herói poderia realizar, pois ninguém nunca conseguira voltar da misteriosa masmorra. Não há nenhuma garantia de que a princesa ainda estaria viva, mas procurá-la é a última esperança para o reino. Esta modificação, além de

cumprir o objetivo da história original (i.e., apresentar o conceito de indefinição em uma divisão por 0), ainda possibilita que a história prossiga reforçando este conceito e apresentando novos tópicos.

6.1.4 A Masmorra Infinita

Neste capítulo do jogo, o jogador segue em busca da Princesa Nádia por uma masmorra mágica onde deverá resolver enigmas apresentados em uma bola de cristal (que pode apresentar vídeos inclusive do *youtube*). As regras, explicadas no primeiro nível pelo *Guardião da Masmorra* (Figura 6.5), são as seguintes: a cada nível será apresentado um desafio cuja a resposta correta indica em qual das portas (A, B, C, D ou E) o jogador deverá entrar para avançar no jogo. Caso o jogador escolha a porta errada, além de sofrer perda de energia vital, será automaticamente transportado para um nível anterior da masmorra ao invés de seguir em direção ao seu objetivo. O *Guardião da Masmorra* alega que está de bom humor e autoriza que sua companheira *Lucca* anote as questões a fim de que possam ser analisadas com calma (Figura 6.7). Se o jogador tiver dúvidas sobre qual é a resposta correta, poderá desafiar o monstro para uma batalha matemática (Figura 6.6) em troca do gabarito (Figura 6.8).

Figura 6.5 – Cena em que o *Guardião da Masmorra* explica as regras da masmorra através de um vídeo incorporado ao "artefato mágico".

Figura 6.6 – Cena em que o jogador batalha com o monstro do nível em troca do gabarito da questão.

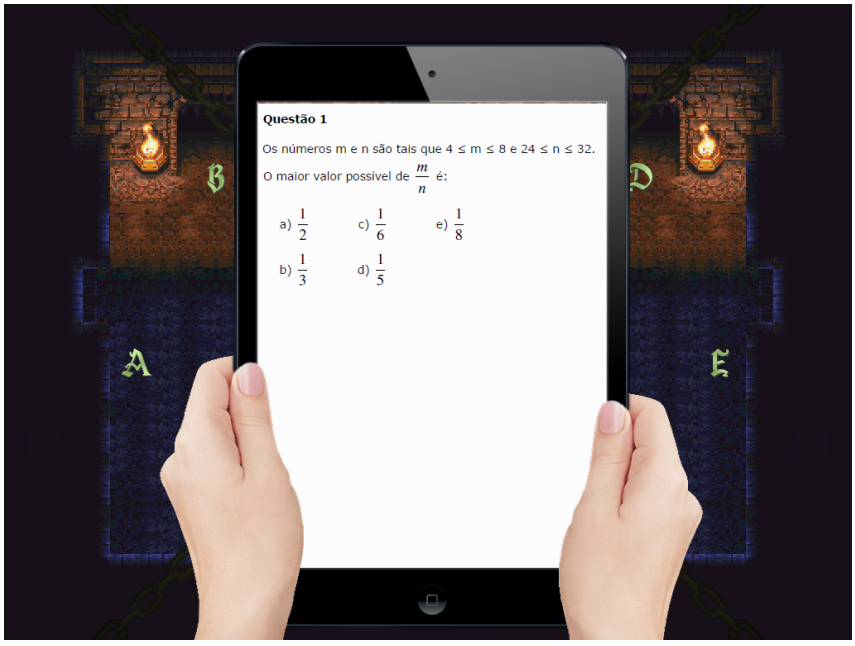


Figura 6.7 – Cena em que Lucca apresenta o desafio matemático por escrito. Curiosamente, ela o faz em um moderno *Tablet*.

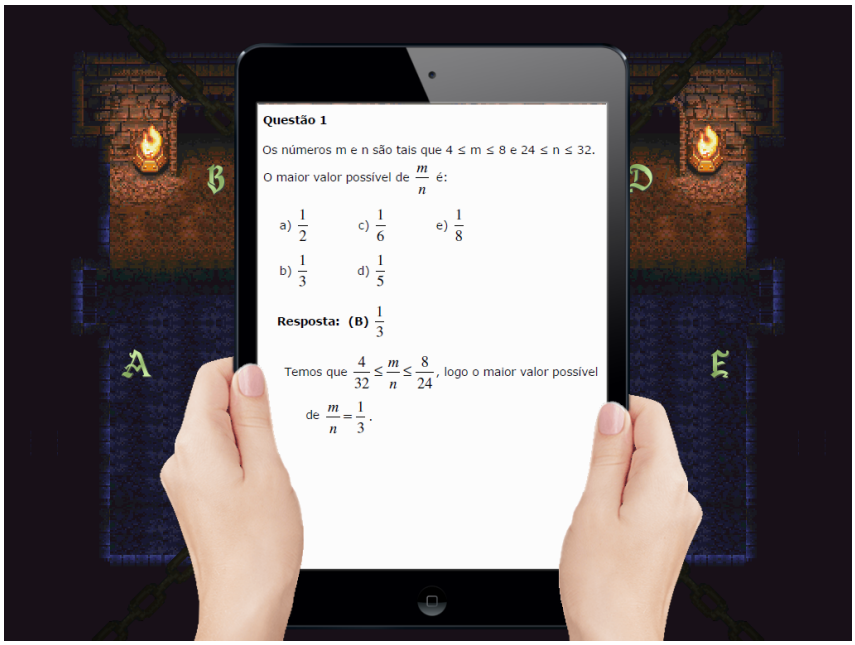


Figura 6.8 – Cena em que Lucca apresenta o desafio matemático atualizado com o gabarito fornecido pelo monstro.

Na medida em que o jogador for avançando, os desafios se tornarão mais difíceis até que, no último nível, onde supostamente a princesa se encontra, não haverá outra escolha a não ser enfrentar o poderoso *Robô Guardião* (Figura 6.9).

Figura 6.9 – Cena do jogador enfrentando o poderoso *Robô Guardião* e falhando em lhe causar algum dano.

Invariavelmente o jogador perderá essa batalha: a armadura do *Robô Guardião* é forte demais para ser avariada pela espada comum do nosso herói e, apesar de lento, seu ataque causa muito dano. O herói precisará de uma nova arma se quiser vencer este adversário.

6.1.5 Uma Espada Exponencialmente Poderosa

Quando o herói se vê impossibilitado de vencer a batalha contra o *Robô Guardião*, sua colega sugere que eles devam contar ao Rei sobre o acontecido. Durante o relato para o Rei, seu *Chanceler* recomenda que os heróis procurem *Melchior*: um mago famoso por produzir armas, armaduras e outros artefatos mágicos (Figura 6.10).

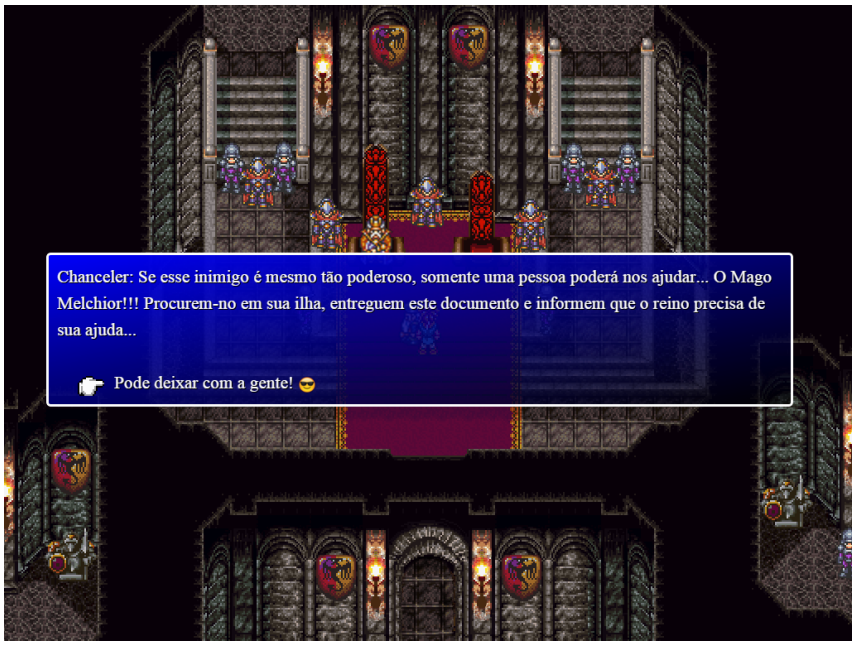


Figura 6.10 – Cena de quando o *Chanceler* recomenda que o jogador procure o *Mago Melchior* para descobrir como derrotar o *Robô Guardião*.

Quando nossos heróis encontram *Melchior*, ele insiste que não é um mago de verdade, mas sim um engenheiro: as ciências são tão fascinantes quanto magia e por isso os menos informados costumam confundir sua profissão.

Ao saber sobre a derrota contra o *Robô Guardião*, *Melchior* afirma que é possível vencê-lo utilizando uma verdadeira espada samurai, cujo processo de fabricação faz com que sua lâmina possua milhões de camadas de aço. O engenheiro explica este processo de fabricação[1] que consiste em aquecer uma peça de aço até sua temperatura de fusão, dobrar a peça ao meio com a ajuda de ferramentas e modelar até que volte a ter suas dimensões originais. O ferreiro

[1] O processo é mesmo utilizado na fabricação de espadas japonesas no mundo real.

repete este processo 22 vezes, produzindo ao fim mais de 4 milhões de camadas de aço[71]. O objetivo dessa história é explicar o porquê das espadas samurais serem tão resistentes e expandir o conhecimento do jogador sobre expoentes e crescimento exponencial (Figura 6.11).

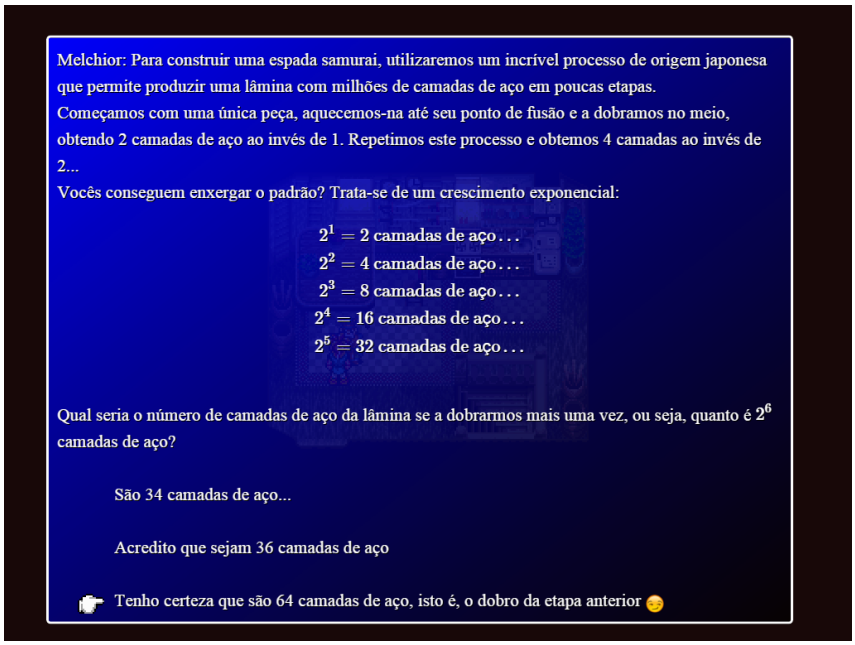


Figura 6.11 – Cena de quando *Melchior* explica para o herói como as espadas samurais se tornam tão poderosas ao serem fabricadas com uma técnica baseada em crescimento exponencial.

Figura 6.12 – Cena mostrando o herói a procura de matéria prima necessária para fabricação de uma poderosa espada samurai.

Para fabricar a espada, no entanto, *Melchior* pede que o herói consiga a matéria prima necessária (i.e., carbono , manganês, sílica, fósforo e cobre), pois com o recesso decretado pelo Rei, os fornecedores não estavam autorizados a entregar encomendas. Isso nos leva a uma adaptação de atividade realizada por Balakhrishnan (2008) para abordar o tópico *Coordenadas Cartesianas*[41]. (Figura 6.12).

Melchior entrega ao herói um pergaminho contendo a pista de onde encontrar os componentes necessários: trata-se de um sistema de equações, cuja a resolução indica as coordenadas (x, y) onde o herói deverá encontrar o material procurado no mapa global do jogo.

Uma vez que o herói tenha coletado toda a matéria prima necessária, *Melchior* construirá sua nova espada como haviam combinado. Derrotar o *Robô Guardião* agora será uma tarefa extremamente fácil (Figura 6.13).

Figura 6.13 – Cena do jogador enfrentando o (agora não tão poderoso) *Robô Guardião* e causando-lhe uma enorme quantidade de dano.

Quando derrotado, o *Robô Guardião* revela que a *Princesa Nádia* não se encontra na *Masmorra Infinita* e nem mesmo no castelo, mas deixa escapar sua localização estimada (Figura 6.14).

Figura 6.14 – Cena onde o derrotado *Robô Guardião* revela o possível paradeiro da *Princesa Nádia*.

6.1.6
A verdade sobre a Princesa Nádia

Neste capítulo do jogo reside a conclusão de nossa pequena aventura. A busca pela princesa está quase concluída: encontrar no mapa as coordenadas de sua localização é trivial, uma vez que o *Robô Guardião* já as forneceu. Elas levam até a *Catedral Escola* (Figura 6.15) e o resto é história[2].

Figura 6.15 – As coordenadas fornecidas pelo *Robô Guardião* levam até a *Catedral Escola*.

[2]Pelo bem do entretenimento, reservamos-nos o direito de não incluir *spoilers* da conclusão do jogo no texto.

7
Conclusão

Neste trabalho, criamos uma plataforma de desenvolvimento de jogos educativos, baseados na dinâmica e ludicidade dos JRPGs. O nosso objetivo, neste caso, é facilitar a criação de jogos educativos onde os estudantes assumem um papel ativo no processo de aprendizado e se divertem enquanto aprendem e são avaliados.

Enquanto investigávamos que características seriam importantes para a construção desses jogos, pudemos ver que bons jogos (projetados ou não com esse intuito) são inerentemente educativos. Também conhecemos ferramentas cognitivas, técnicas de narrativa e contação de história utilizadas em salas de aula que são capazes de apresentar conteúdo educativo de maneira envolvente e relevante. Neste contexto, identificamos que o sub-gênero JRPG leva vantagem sobre outros gêneros e oferece uma excelente oportunidade de empregar essas técnicas de forma automatizada, diminuindo a responsabilidade do professor no preparo de aulas elaboradas como no caso da adoção do RPG tradicional.

No desenvolvimento da presente pesquisa, também verificamos a versatilidade do motor de jogos *CppPlay*[2]. E incorporamos melhorias importantes que, além de possibilitar a implementação de características de JRPG, também podem ser reaproveitadas facilmente em outras pesquisas e projetos que venham a utilizar o motor.

Reunimos todos esses elementos em um ambiente de desenvolvimento integrado, oferecendo ao desenvolvedor uma série de facilidades para a melhoria da produtividade e qualidade de seus jogos. Por último, contruímos um jogo prova de conceito similar a um dos jogos JRPG mais consagrados de todos os tempos e, desta forma, demonstramos o potencial e a viabilidade deste ambiente.

7.1
Trabalhos futuros

Nesta seção, apresentamos sugestões de trabalhos futuros para melhoria deste trabalho como um todo:

- **Adoção de recursos artísticos próprios (ou de licença livre)**: O jogo prova de conceito, que acompanha a plataforma desenvolvida utiliza

recursos do jogo *Chrono Trigger*, pertencentes a empresa *Square-Enix*[46] e só podem ser utilizados sob as restrições de *fair-use*[70]. Para que possamos distribuir publicamente este jogo, precisaremos criar recursos artísticos próprios (i.e., contratando e/ou agregando artistas ao time de desenvolvimento) ou adotar recursos de licença livre. A decisão de utilizar recursos proprietários de terceiros nesta pesquisa se deu porque a primeira alternativa requer recursos monetários e humanos dos quais nem sempre se dispõe, já a segunda alternativa esbarra em um outro desafio: encontrar material artístico gratuito, em quantidade e qualidade suficientes para construir um jogo harmonioso artisticamente (i.e., cenários, objetos, personagens, música e efeitos sonoros que combinem entre si) é uma tarefa quase impossível. Uma sugestão para resolver essa questão seria estabelecer uma parceria com o Departamento de Artes e Design ou com outros alunos da universidade que também sejam artistas.

- **Sofisticar o meta-motor JRPG**: O meta-motor desenvolvido nesta pesquisa cumpre o propósito de apresentar a plataforma proposta, mas de maneira alguma atinge todo o seu potencial. Há espaço para diversas melhorias em seus subsistemas. O sistema de batalhas, por exemplo, poderia ser modificado para permitir múltiplos personagens do jogador e inimigos em uma mesma batalha. Assim surgiriam novas oportunidades de mecânicas de jogo, como por exemplo: incorporar ataques especiais onde a área de efeito (i.e., área onde o ataque causará dano aos inimigos) compreenda o *plot* de funções matemáticas. Múltiplos personagens também requerem um pensamento mais estratégico do jogador sobre quando e qual inimigo atacar, decidir que tipos de ataques são mais eficientes para determinados oponentes ou identificar quando curar um personagem do seu grupo é mais importante do que atacar um inimigo.
- **Abordar outras disciplinas**: Nesta pesquisa, abordamos basicamente o ensino da matemática (quase uma tradição em pesquisas relacionadas a melhoria do ensino), mas a plataforma proposta de maneira alguma está limitada a esta disciplina. Seria interessante construir jogos que abordem outras áreas presentes no currículo escolar (e.g., línguas, história, filosofia, biologia, física, etc), seja de maneira isolada ou multidisciplinar.
- **Material de treinamento para desenvolvedores**: Desenvolver jogos eletrônicos é uma tarefa multidisciplinar que requer prática mesmo nas ferramentas mais amigáveis do mercado. Oferecer material de treinamento (e.g., tutoriais passo-a-passo, vídeo aulas, cursos e *workshops*) seria uma boa forma de atrair desenvolvedores e contribuir para o sucesso da plataforma.

- **Avaliar a aplicação em campo**: Os testes com usuários que realizamos durante a pesquisa foram bastantes simples, envolvendo poucas pessoas e um ambiente controlado. Tanto a plataforma de desenvolvimento quanto os jogos que ela pode produzir precisam ser avaliados em campo, com usuários de diferentes perfis, em maior quantidade e utilizando alguma metodologia formal. A escala *Likert*[72] seria uma boa forma de medir a satisfação dos usuários e análises estatísticas, comparando o desempenho escolar de estudantes que utilizaram jogos construídos com a plataforma *versus* estudantes que não utilizaram, ou ainda, comparando o desempenho dos estudantes antes e depois de sua adoção, seriam de imenso valor.
- **Sistema Visual de Scripting**: Por mais que um ambiente integrado de desenvolvimento e uma linguagem de programação amigável como Lua facilitem o trabalho dos programadores, escrever código de programação ainda é uma tarefa complexa para os não iniciados (i.e., pessoas fora da área de computação). Para tornar mais fácil o processo de desenvolvimento de scripts, seria bom que tivéssemos uma ferramenta capaz de representar visualmente os algoritmos escritos em Lua sem exigir que o desenvolvedor domine linguagens de programação baseadas em código. Uma ferramenta que serve de modelo para esta funcionalidade é a *Blueprints Visual Scripting*[73], presente no motor *Unreal Engine 4* [74]. Neste tipo de ferramenta, os algoritmos são construídos de forma bastante intuitiva através da conexão de nós em um grafo (Figura 7.1).

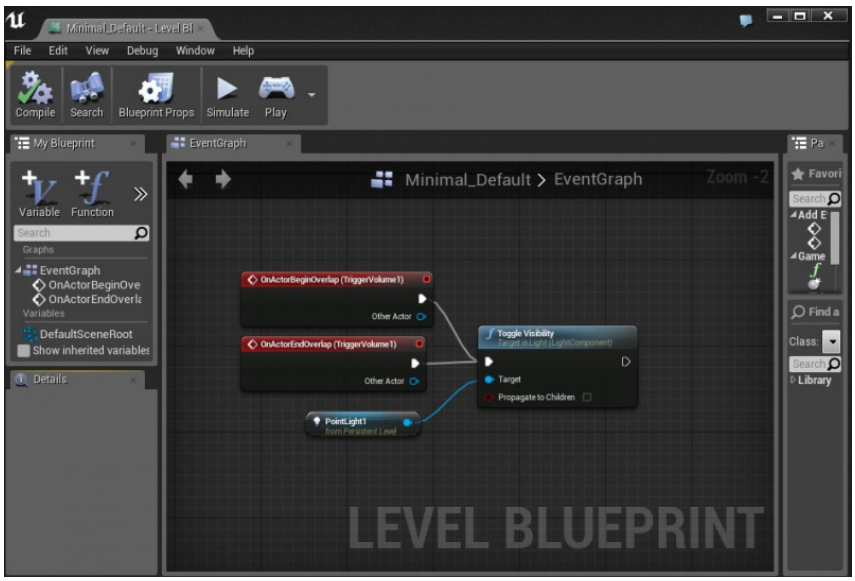


Figura 7.1 – Ferramenta *Blueprints Visual Scripting* sendo utilizada para programar um script que acende e apaga uma lâmpada conforme o jogador entra ou sai de uma sala no cenário do jogo.

Referências bibliográficas

[1] MCCLARTY, K. L.; ORR, A.; FREY, P. M.; DOLAN, R. P.; VASSILEVA, V. ; MCVAY, A.. **A literature review of gaming in education**. Gaming in education, 2012.

[2] NETO, V.. **Uma camada de Scripts Lua para o motor de jogos CppPlay**, 2012.

[3] REIMERS, F. M.. **Preparing Students for the Flat World - Education Week**. Education Week, 28(7):24–25, 2008.

[4] OF AMERICAN SCIENTISTS, F.. **Summit on educational games: Harnessing the power of video games for learning.**, 2006.

[5] KIRRIEMUIR, J.; MCFARLANE, A.. **Literature Review in Games and Learning**. URL: http://archive.futurelab.org.uk/resources/publications-reports-articles/literature-reviews/Literature-Review378, 2004. Acessado em: 01/09/2015.

[6] KIRRIEMUIR, J.. **Video gaming, education, and digital learning technologies: Relevance and opportunities**. URL: http://www.dlib.org/dlib/february02/kirriemuir/02kirriemuir.html, 2002. Acessado em: 01/09/2015.

[7] LENHART, A.; KAHNE, J. ; MIDDAUGH, E.. **Teens, video games, and civics**. URL: http://www.pewinternet.org/Reports/2008/Teens-Video-Games-and-Civics.aspx, 2008. Acessado em: 01/09/2015.

[8] FABOS, B.. **Media in the classroom: An alternative history**. 2001. Paper presented at the annual meetings of the American Educational Research Association, Seattle, WA.

[9] GINSBURK, K.. **The importance of play in promoting healthy child development and maintaining strong parent-child bonds**. URL: http://pediatrics.aappublications.org/content/119/1/182, 2007. Acessado em: 01/09/2015.

[10] BODROVA, E.; LEONG, D.. **The importance of being playful.** URL: http://www.ascd.org/publications/educational-leadership/apr03/vol60/num07/The-Importance-of-Being-Playful.aspx, 2003. Acessado em: 01/09/2015.

[11] HIRSH-PASEK, K.; GOLINKOFF, R. ; EVER, D.. **Einstein never used flashcards: How our children really learn and why they need to play more and memorize less**, 2003.

[12] ZIGLER, E.; SINGER, D. ; BISHOP-JOSEF, S.. **Children's play, the roots of reading**, 2004.

[13] KE, F.. **A qualitative meta-analysis of computer games as learning tools**. Handbook of research on effective electronic gaming in education, 1:1–32, 2009.

[14] GEE, J.. **Deep learning properties of good digital games: How far can they go?**, 2009.

[15] GROFF, J.; HOWELLS, C. ; CRANMER, S.. **The impact of console games in the classroom: Evidence from schools in Scotland.**, 2010.

[16] KLOPFER, E.; OSTERWEIL, S.; SALEN, K. ; OTHERS. **Moving learning games forward**. Cambridge, MA: The Education Arcade, 2009.

[17] DICKEY, M. D.. **Engaging by design: How engagement strategies in popular computer and video games can inform instructional design**. URL: http://dx.doi.org/10.1007/BF02504866, 53(2):67–83, 2005. Acessado em: 01/09/2015.

[18] BLACK, P. J.; BUONCRISTIANI, P. ; WILIAM, D.. **Inside the Black Box: Raising Standards Through Classroom Assessment**. Hawker Brownlow Education, 1998.

[19] BLACK, P.; HARRISON, C.; LEE, C.; MARSHALL, B. ; WILIAM, D.. **Working inside the black box: Assessment for learning in the classroom**. 2002.

[20] TOMLINSON, C. A.. **The differentiated classroom. responding to the needs of all learners.**, 1999.

[21] SNOW, R.; FARR, M.. **Conative and affective process analysis (3rd ed.)**. Erlbaum Associates, 1987.

[22] HOPKINS, D.; JÄRVELÄ, S. ; MILIBAND, D.. **Schooling for tomorrow. Personalising Education**. OECD, 2006.

[23] KICKMEIER-RUST, M. D.; HOCKEMEYER, C.; ALBERT, D. ; AUGUSTIN, T.. **Micro adaptive, non-invasive knowledge assessment in educational games**. In: DIGITAL Games AND Intelligent Toys Based Education, 2008 Second IEEE International Conference ON, p. 135–137. IEEE, 2008.

[24] JALONGO, M. R.. **Beyond Benchmarks and Scores: Reasserting the Role of Motivation and Interest in Children's Academic Achievement an ACEI Position Paper**. Childhood Education, 83(6):395–407, 2007.

[25] DALTON, J.. **Online training needs a new course: The forrester report**. 2000.

[26] BRIDGELAND, J. M.; DIIULIO JR, J. J. ; MORISON, K. B.. **The silent epidemic: Perspectives of high school dropouts.** Civic Enterprises, 2006.

[27] GEE, J.. **What video games have to teach us about learning and literacy**, 2003.

[28] RUPP, A. A.; GUSHTA, M.; MISLEVY, R. J. ; SHAFFER, D. W.. **Evidence-centered design of epistemic games: Measurement principles for complex learning environments**. The Journal of Technology, Learning and Assessment, 8(4), 2010.

[29] KIRKPATRICK, D. L.. **Evaluating training programs (3rd ed.)**. Berrett-Koehler Publishers, Inc, 2006.

[30] BARAB, S.; DEDE, C.. **Games and immersive participatory simulations for science education: an emerging type of curricula**. Journal of Science Education and Technology, 16(1):1–3, 2007.

[31] GEE, J. P.. **Good Video Games and Good Learning: Collected Essays on Video Games, Literacy, and Learning**. New York: Peter Lang, 2007.

[32] JOHNSON, L.; LEVINE. **The 2011 Horizon Report.** 2011.

[33] BEHRENS, J. T.; FREZZO, D.; MISLEVY, R.; KROOPNICK, M. ; WISE, D.. **Structural, functional, and semiotic symmetries in simulation-based games and assessments**. Assessment of problem solving using simulations, p. 59–80, 2007.

[34] FEIJÓ, B.; DA SILVA, F. S. C. ; CLUA, E.. **Introdução à Ciência da Computação com Jogos**. Elsevier, 2009.

[35] NETO, V.. **Adoção do Pipeline programável OpenGL no motor de Jogos CppPlay e as vantagens sobre o pipeline convencional**, 2015.

[36] **LÖVE2D - A framework for making 2D games in the Lua programming language**. URL: https://love2d.org/, 2016. Acessado em: 22/09/2016.

[37] **pygame - a set of Python modules designed for writing games**. URL: http://www.pygame.org/, 2016. Acessado em: 22/09/2016.

[38] SOARES, M.. **Projeto de jogos educativos 2d de aventura utilizando lua**, 2012.

[39] IGNÁCIO, A.. **O rpg eletrônico no ensino de química: Uma atividade lúdica aplicada ao conhecimento de tabela periódica**, 2013.

[40] KIILI, K.; DEVLIN, K.; PERTTULA, T.; TUOMI, P. ; LINDSTEDT, A.. **Using video games to combine learning and assessment in mathematics education**. International Journal of Serious Games, 2(4), 2015.

[41] BALAKRISHNAN, C.. **Teaching secondary school mathematics through storytelling**. PhD thesis, Simon Fraser University, 2008.

[42] EGAN, K.. **An imaginative approach to teaching**. San Francisco, 2005.

[43] ZAZKIS, R.; LILJEDAHL, P.. **Teaching mathematics as storytelling**. Sense publishers The Netherlands, 2008.

[44] OLIVEIRA, C.; LEIPZIGER, D.. **Criação de um Texto Cooperativo – RPG (Role Playing Game)**. URL: http://portaldoprofessor.mec.gov.br/fichaTecnicaAula.html?aula=12381, 2011. Acessado em: 12/09/2016.

[45] **Top 100 RPGs of All Time**. URL: http://www.ign.com/top/rpgs/, 2016. Acessado em: 12/09/2016.

[46] SQUARE-ENIX. **Chrono Trigger**. URL: http://www.square-enix.co.jp/smart/chronotrigger/, 2011. Acessado em: 12/09/2016.

[47] THOMAS, M.. **Chrono Trigger Review**. URL: http://www.ign.com/articles/2011/05/25/chrono-trigger-review/, 2011. Acessado em: 12/09/2016.

[48] CELES, W.; FIGUEIREDO, L. ; IERUSALIMSCHY, R.. **Binding C/C++ objects to Lua**. Game Programming Gems, 6:341–355, 2006.

[49] SELLERS, G.; WRIGHT JR, R. S. ; HAEMEL, N.. **OpenGL SuperBible: Comprehensive Tutorial and Reference**. Addison-Wesley, 2013.

[50] WOLFF, D.. **OpenGL 4.0 Shading Language Cookbook**. Packt Publishing Ltd, 2011.

[51] GAMMA, E.. **Design patterns: elements of reusable object-oriented software**. Pearson Education India, 1995.

[52] VAN HEESCH, D.. **Doxygen: Source code documentation generator tool**. URL: http://www.doxygen.org, 2015. Acessado em: 12/09/2016.

[53] LINDEIJER, T.. **Tiled: Your free, easy to use and flexible tile map editor**. URL: http://doc.mapeditor.org/, 2016. Acessado em: 12/09/2016.

[54] ATIAS, T.. **TMXParser: a library for parsing the maps generated by Tiled Map Editor**. URL: https://github.com/tamatias/tmxparser, 2016. Acessado em: 12/09/2016.

[55] MARSHALL, G.. **CEF: Chromium Embedded Framework**. URL: https://bitbucket.org/chromiumembedded/cef, 2016. Acessado em: 12/09/2016.

[56] CHROMIUM.ORG. **Chromium: an open-source browser project**. URL: https://www.chromium.org/Home, 2016. Acessado em: 12/09/2016.

[57] W3C - THE WORLD WIDE WEB CONSORTIUM. **HTML5: A vocabulary and associated APIs for HTML and XHTML**. URL: https://www.w3.org/TR/html5/introduction.html, 2014. Acessado em: 12/09/2016.

[58] W3C - THE WORLD WIDE WEB CONSORTIUM. **JavaScript Web APIs**. URL: https://www.w3.org/standards/webdesign/script, 2016. Acessado em: 12/09/2016.

[59] W3C - THE WORLD WIDE WEB CONSORTIUM. **CSS: Cascading Style Sheets**. URL: https://www.w3.org/Style/CSS/, 2016. Acessado em: 12/09/2016.

[60] GROUP, K.. **WebGL: OpenGL ES 2.0 for the Web**. URL: https://www.khronos.org/webgl/, 2016. Acessado em: 12/09/2016.

[61] COMPANY, G.. **Youtube**. URL: https://www.youtube.com/yt/about/, 2016. Acessado em: 12/09/2016.

[62] COURSERA.ORG. **coursera**. URL: https://www.coursera.org/about/, 2016. Acessado em: 12/09/2016.

[63] GOOGLE INC. **Angularjs**. URL: https://github.com/angular/angular.js, 2016. Acessado em: 12/09/2016.

[64] MARK, O.; JACOB, T.. **Bootstrap: a sleek, intuitive, and powerful front-end framework for faster and easier web development**. URL: https://github.com/twbs/bootstrap, 2016. Acessado em: 12/09/2016.

[65] MIKE, B.. **D3: Data-Driven Documents**. URL: https://d3js.org/, 2016. Acessado em: 12/09/2016.

[66] MATHJAX CONSORTIUM. **Mathjax: Beautiful math in all browsers**. URL: https://www.mathjax.org/, 2016. Acessado em: 12/09/2016.

[67] CONTRIBUTORS, E.; OTHERS. **LDT: Lua Development Tools**. URL: http://www.eclipse.org/ldt/, 2016. Acessado em: 12/09/2016.

[68] ECLIPSE.ORG. **Eclipse: An amazing open source community of Tools, Projects and Collaborative Working Groups**. URL: https://www.eclipse.org/home/index.php, 2016.

[69] ECLIPSE.ORG. **DocLua: Documentation Language**. URL: https://wiki.eclipse.org/LDT/User_Area/Documentation_Language, 2016.

[70] EN.WIKIPEDIA.ORG. **Fair use**. URL: https://en.wikipedia.org/wiki/Fair_use, 2016. Acessado em: 12/09/2016.

[71] PAPPAS, T.. **More Joy of Mathematics: Exploring mathematics all around you**. Wide World Pub Tetra, 1991.

[72] EN.WIKIPEDIA.ORG. **Likert scale**. URL: https://en.wikipedia.org/wiki/Likert_scale, 2016. Acessado em: 22/09/2016.

[73] EPIC GAMES. **Blueprints Visual Scripting**. URL: https://docs.unrealengine.com/latest/INT/Engine/Blueprints/index.html, 2016. Acessado em: 12/09/2016.

[74] EPIC GAMES. **What is Unreal Engine 4**. URL: https://www.unrealengine.com/what-is-unreal-engine-4, 2016. Acessado em: 12/09/2016.