BLUCRAFT: UMA FERRAMENTA PARA A CRIAÇÃO DE JOGOS DE RPG DIGITAIS

Guilherme Paz Silva, Dalton Solano dos Reis – Orientador

Curso de Bacharel em Ciência da Computação
Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brasil

guilhermepaz@furb.br, dalton@furb.br

Resumo: *Este artigo apresenta o desenvolvimento de um editor de jogos de RPG digitais com o objetivo de facilitar a criação de histórias interativas, que podem ser utilizadas em um contexto pedagógico ou de entretenimento. O editor permite a confecção de mapas com uma variedade de sprites para a pintura de camadas e programação visual através de eventos para a criação de lógicas de jogo, utilizando conceitos como interruptores e variáveis para customização de histórias e fluxos de jogo. O editor foi desenvolvido em Unity. Os objetivos foram atingidos, resultando em um editor de jogos de RPG funcional com customização de mapas e de lógicas através de programação visual, auxiliados pelas ferramentas Unity e SimpleJSON.NET que cumpriram os papéis esperados no projeto.*

Palavras-chave*: Editor de jogos. Jogos. RPG. Unity.*

1 INTRODUÇÃO

Segundo Squire (2003, p. 1, tradução nossa), "o desenvolvimento contemporâneo de jogos, principalmente histórias interativas, ferramentas de autoria digital e mundos colaborativos, sugere novas e poderosas oportunidades para mídia educacional". Visto que jogos se mostram presentes na história de quase todas as culturas e sociedades (HUIZINGA, 1954 apud ZAGAL, 2010, p. 11), pode-se afirmar que os jogos são uma ferramenta de desenvolvimento cultural.

Os jogos de Role Playing Game (RPG) são "jogos de representação de papéis, onde a cooperação e a criatividade são seus principais elementos" (GRANDO; TAROUCO, 2008, p. 3). A tradução da sigla é "jogo de interpretação de papéis" (em tradução livre) e, em sua forma original, é comumente classificado como uma brincadeira de contar histórias (ALVES, Lynn et al, 2004, p. 5).

Visto de um espectro educacional, o engajamento de crianças no desenvolvimento de jogos "[...] pode permitir o desenvolvimento da imaginação e criatividade na infância e consequentemente das funções psicológicas superiores, como habilidades de concentração, atenção, raciocínio, memória [...]" (ALVES, 2017, p. 4). Além disso, segundo Grando e Tarouco (2008, p. 8), "as características principais que auxiliam o jogo de RPG a se tornar uma excelente ferramenta educacional são: socialização, cooperação, criatividade, interatividade e interdisciplinaridade". Desta forma, é plausível sugerir que o uso de jogos, mais especificamente do tipo de RPG, seria benéfico no ambiente pedagógico, tanto em um contexto de criação quanto de aplicação.

Diante do exposto, o objetivo do projeto foi desenvolver um editor de jogos no estilo RPG que disponibilize ferramentas para a criação de contextos e objetivos customizáveis pelo usuário. Os objetivos específicos são: permitir a criação de entidades executáveis através de programação visual; permitir a customização de cenários interconectados; executar e compartilhar os jogos criados no editor através de persistência. Portanto, este trabalho visa explorar uma versão digital dos jogos deste tipo.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção descreve os principais conceitos para o melhor entendimento do trabalho desenvolvido. A seção 2.1 descreve o que são e quais os objetivos de jogos de RPG, de sua criação até sua versão digital; a seção 2.2 apresenta a utilização da biblioteca SimpleJSON.NET para persistência de dados; e a seção 2.3 apresenta os trabalhos correlatos.

2.1 ROLE PLAYING GAMES (RPG)

A primeira versão do que viria a ser o RPG nasceu em 1974 através do jogo "Dungeons and Dragons" de Gary Gygax (GRANDO; TAROUCO, 2008, p. 4). Conforme demonstra a Figura 1, a versão analógica do jogo é composta do narrador (também chamado de "mestre" no contexto do jogo) e os jogadores que, juntos, formam uma estória interativa (PEREIRA, 1992, p. 1). Através da discussão e interpretação de ações faladas ou escritas, os jogadores narram acontecimentos enquanto o narrador descreve as consequências e desdobramentos destes acontecimentos.

Segundo Bittencourt e Giraffa (2003, p. 5), "esta mecânica refere-se a forma tradicional de jogar RPG bastante conhecida pela comunidade como 'RPG de mesa', pelo fato de comumente ser jogado em torno de uma mesa com lápis,

papel e dados". Desde sua criação, diversas outras modalidades baseadas em cima do conceito de "interpretação de personagens" surgiram. Dentre estas modalidades, o RPG digital surge em meados dos anos 70 (CRAWFORD, 1982, p. 34). Desde então, o ramo de jogos digitais cresceu vertiginosamente, com estimativas de consumo em mais de 150 bilhões de dólares no ano de 2019 (WIJMAN, 2019, p. 1).

Figura 1 - Representação de uma mesa de RPG

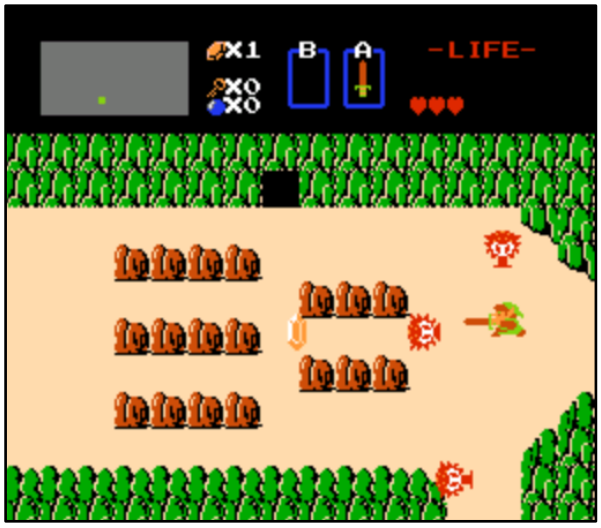
Fonte: Diacritica (2011).

Segundo Chagas (2010, p. 61), "a proliferação dos RPGs em diversas modalidades, como jogos de tabuleiro, jogos de cards [...] ou nos jogos produzidos para computadores e videogames, populariza o hábito, exercícios e conduta relacionados à construção de novas narrativas pessoais". Na modalidade digital, "o RPG continua sendo uma representação de papéis, um jogo de faz-de-conta e permitindo vivenciar mundos imaginários, só que o grupo de pessoas não se reúnem presencialmente, mas no ciberespaço" (BITTENCOURT; GIRAFFA, 2003, p. 2).

Em contramão dos RPGs *online* que permitem a conexão de diversos jogadores e foi uma modalidade popularizada principalmente pelo jogo Diablo, da Blizzard (BITTENCOURT; GIRAFFA, 2003, p. 5), uma modalidade similar mas à parte é a de jogos de RPG de jogador único, onde não existe interação com um segundo jogador, considerada a primeira forma digital de jogos de RPG.

Exemplos não faltam para o contexto de jogos, mas um dos mais ilustres e representativos foi The Legend of Zelda (Zeruda no Densetsu Za Hairaru Fantajī, no Japão), como mostrado na Figura 2. O jogo de 1986 da Nintendo apresenta Link, um personagem controlado pelo jogador através de uma visão aérea. Os objetivos do jogador são decifrar enigmas, encontrar moradores e comerciantes e enfrentar criaturas em um mundo de fantasia (NINTENDO OF AMERICA, 1986, p. 18). O jogo é considerado do gênero "Japanese RPG", uma vertende do RPG ocidental com características próprias como batalha em turnos, foco no enredo e visuais próprios derivados do estilo mangá.

Figura 2 - The Legend of Zelda



Fonte: Nintendo (1986).

Fora da indústria do entretenimento, existe uma vertente em ebulição no mundo acadêmico que explora os chamados Jogos Sérios (JS) e que abrange também os jogos educativos. Através deste conceito de Jogos Digitais Sérios do estilo RPG (RPGDS) é possível contextualizar as ações de um jogador dentro de um ambiente montado através de

uma ferramenta (GURZYNSKI; HOUNSELL; KEMCZINSKI, 2016, apud Frias, 2009, p. 618). Uma ferramenta que permite a criação destes jogos é a ferramenta RPG4ALL (ver seção 2.3).

2.2 SIMPLEJSON.NET

A biblioteca SimpleJSON.NET tem o objetivo de facilitar o `parse` e a geração de arquivos JSON em linguagens .NET através da serialização de tipos primitivos e coleções (BOLDAI AB, 2011, p.1). Na deserialização, são utilizados objetos do tipo JObject, representando a árvore do JSON que está sendo deserializado.

O formato de arquivos JSON tem como principal fundamento ser de fácil leitura tanto para humanos quanto para máquinas, utilizando somente texto e sem depender de linguagens específicas (JSON, 2020, p. 1). O Quadro 1 apresenta a estrutura de um JSON. Segundo sua documentação, o JSON se baseia em duas estruturas:

a) Coleção de pares chave/valor: representados em linguagens como um `object`, `hashmap` ou `dictionary`;
b) Lista ordenada de valores : representados em linguagens como um `array`, vetor ou lista.

Quadro 1 - Exemplo da estrutura de um JSON

```
...
"entities": [
  {
    "name": "",
    "location": "9;10",
    "states": {
      "Padrao": {
        "variableCheck": false,
        "name": "Padrao",
        "execution": "0",
        "image": "Master_Tileset;0;352;16;16",
        "passable": false,
        "events": [
          {
            "event": "Assets.Source.Events.MessageEvent",
            "message": "-> Cidade da Luz\n-> Castelo Hirulan"
          }
        ]
  ...
```

Fonte: elaborado pelo autor.

Dentro da biblioteca SimpleJSON.NET, as classes `JSONEncoder` e `JSONDecoder` tem as funções de serializar e deserializar objetos, respectivamente. Além disso, é possível verificar erros de `parse` através da classe de exceção `ParseError`. Aliando estas ferramentas com o encapsulamento proporcionado pela orientação a objetos, é possível criar objetos que se serializam e deserializam com os métodos e `JObjects` trazidos pela biblioteca. O Quadro 2 apresenta alguns exemplos de chamada de método na biblioteca e seus respectivos retornos.

Quadro 2 - Exemplos de uso na biblioteca SimpleJSON.NET

```
JSONEncoder.Encode(new[] { 1, 2, 3}); // retornará [1,2,3] no JSON
JSONEncoder.Encode(new Dictionary<string, object>
                    {
                          {"key", "value"},
                          {"other", 123}
                    }); // retornará {"key":"value","other":123} no JSON

...

JObject obj = JSONDecoder.Decode("[1,2,3]"); // retornará um objeto JObject com três
números
var numero = (int)(obj[1]); // objeto com valor 2
```

Fonte: elaborado pelo autor.

2.3 TRABALHOS CORRELATOS

Nesta seção, são apresentadas as características três trabalhos correlatos que abordam temas relacionados a este. No Quadro 3 são apresentadas características do EasyEdu, trabalho desenvolvido por Corso (2017) que se trata de uma ferramenta web equipada para o desenvolvimento de jogos educacionais por professores e crianças. No Quadro 4 são apresentadas características do trabalho RPG4ALL desenvolvido por Pessini, Kemczinski, Hounsell (2015) que se trata de uma ferramenta de autoria para o desenvolvimento de jogos. Por fim, o Quadro 5 apresenta o produto RPG Maker MV da empresa Enterbrain (2020), uma ferramenta para a criação de jogos de RPG.

Quadro 3 - EasyEdu

Referência	Corso (2017)
Objetivos	Desenvolvimento de jogos educacionais em ambiente web com o uso de templates.
Principais funcionalidades	Criação de jogos educacionais através do uso de templates. Compartilhamento dos jogos através da internet e QR Code. Personalização com o envio de imagens e cadastro de palavras.
Ferramentas de desenvolvimento	AngularJS, Google Drive, QR Code.
Resultados e conclusões	De acordo com o autor, os templates de quebra-cabeças e memória foram inclusos no planejamento mas não foram implementados na versão final. Ainda assim, a equipe de pedagogia responsável pela aplicação em sala de aula obteve os resultados esperados.

Fonte: elaborado pelo autor.

Com o EasyEdu é possível desenvolver jogos baseados em templates com regras pré-definidas e executar estes jogos a partir de uma galeria. Através da ferramenta, o professor pode desenvolver jogos, armazená-los no Google Drive e compartilhá-los disponibilizando acesso através do QR Code. Um ponto importante da ferramenta é a personalização dos jogos criados por parte do professor encarregado da utilização.

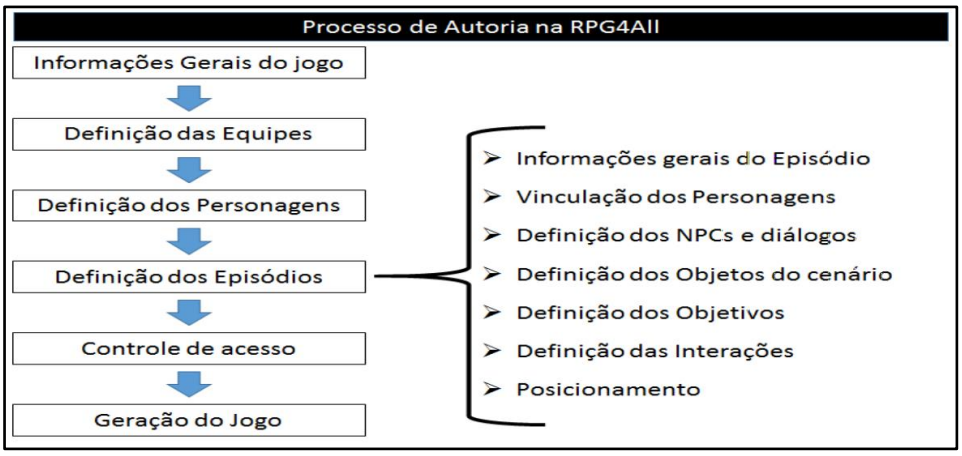
Quadro 4 - RPG4ALL

Referência	Pessini, Kemczinski, Hounsell (2015)
Objetivos	Ferramenta para especificação de Jogos Sérios por docentes sem conhecimento em desenvolvimento de jogos.
Principais funcionalidades	Importação de mapas da ferramenta Tiled, utilizada na criação de mapas. Definição de episódios, objetos e personagens. Posicionamento de objetos em cena. Reprodução dos jogos através do Módulo de Execução.
Ferramentas de desenvolvimento	Tiled Map Editor; a tecnologia dos Módulos não é discutida no trabalho.
Resultados e conclusões	A ferramenta encontrava-se em fase de homologação junto a docentes e alunos de cursos de licenciatura na época do artigo. Os autores citaram a validação das funcionalidades da ferramenta como uma etapa futura.

Fonte: elaborado pelo autor.

O RPG4ALL disponibiliza uma ferramenta para a definição de diversas etapas de um RPG em um contexto de Jogos Sérios (JS). O intuito é a utilização dentro de ambiente pedagógico e com o uso por docentes e alunos de pedagogia. A Figura 4 apresenta as diversas etapas necessárias para especificação dos JS dentro do Módulo de Autoria para que, posteriormente, o produto do processo de geração seja executado por um segundo módulo, o Módulo de Execução.

Figura 3 - Processo de Autoria do RPG4ALL



Fonte: Pessini, Kemczinski, Hounsell (2015).

Quadro 5 - RPG Maker

Referência	Enterbrain (2020)
Objetivos	Ferramenta comercial para o desenvolvimento de jogos de RPG através de programação visual.
Principais funcionalidades	Programação lógica através de eventos e programação visual. Edição de mapas e objetos. Definição de itens e personagens. Exportação para múltiplas plataformas como Windows, Mac, mobile e HTML5. Programação avançada através de JavaScript.
Ferramentas de desenvolvimento	A ferramenta é proprietária e não são divulgadas as tecnologias utilizadas.
Resultados e conclusões	A ferramenta é comercial; não foram encontradas conclusões acerca de conclusões do projeto.

Fonte: elaborado pelo autor.

Conhecido no desenvolvimento de jogos independentes, o RPG Maker é uma engine para criação de jogos de RPG, lançando sua primeira versão em 1999. O foco da ferramenta é a disponibilização de cadastros e programação visual através de eventos para permitir o design de mapas e definição de lógicas de jogo de forma acessível. Além disso, é possível utilizar JavaScript caso o usuário seja proficiente no desenvolvimento de software.

3 DESCRIÇÃO DA FERRAMENTA

Esta seção dedica-se a apresentar os detalhes de especificação, implementação e usabilidade da ferramenta. Com esse objetivo, a seção foi dividida em três partes. A primeira parte, na seção 3.1, apresenta uma visão geral da ferramenta e seu fluxo de funcionamento. A segunda parte, na seção 3.2, detalha os processos de implementação mais significativos dentro do projeto. Por fim, a seção 3.3 apresenta as características de uso da ferramenta com especificações de sua usabilidade.

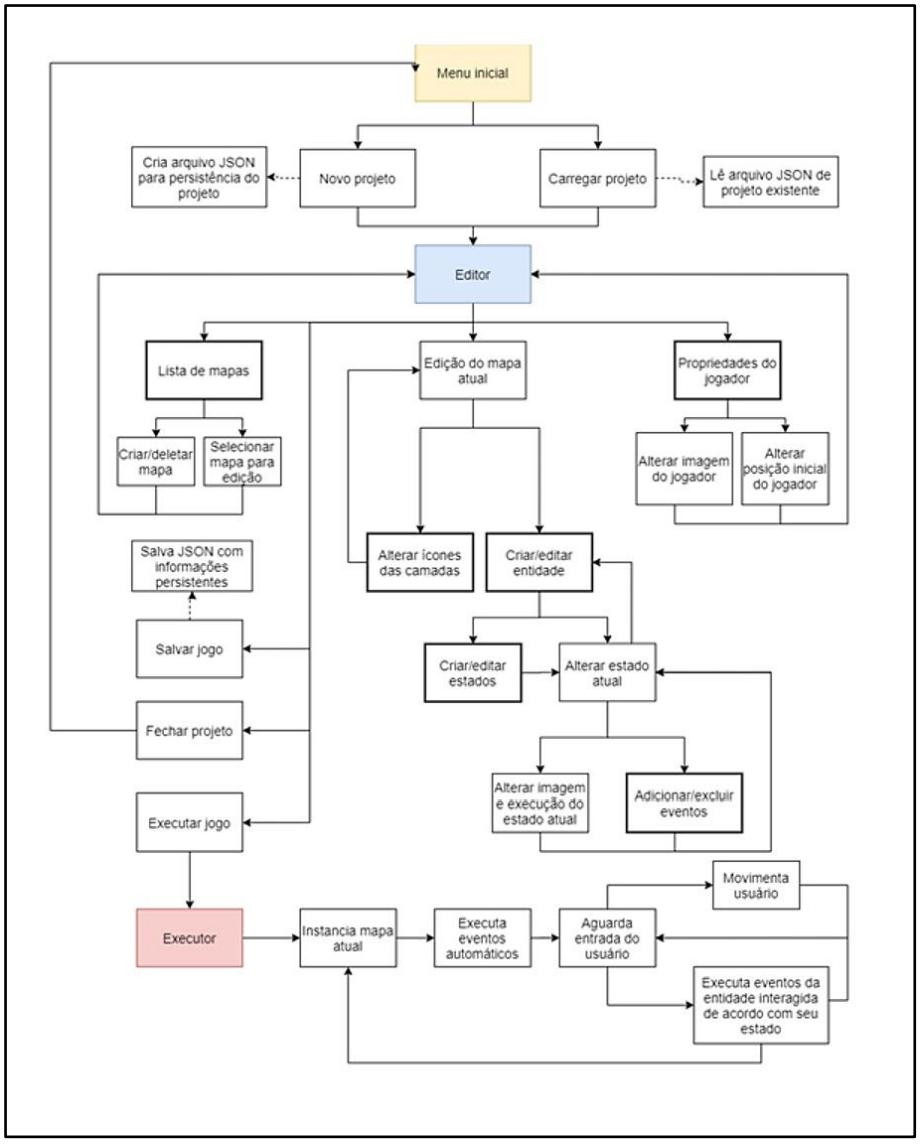
3.1 VISÃO GERAL DA FERRAMENTA

A ferramenta desenvolvida se trata de um editor de jogos com foco nas mecânicas normalmente utilizadas no gênero RPG. Para isso, foram utilizados os conceitos de mapas estáticos e entidades dinâmicas formadas de eventos, assim como programação visual através de janelas de cadastro e configuração. Assim, o fluxo da ferramenta é composto de duas partes:

c) Editor: composto por uma interface que possibilita construir o jogo, definindo mapas e atributos do jogador, bem como especificar os eventos de execução;
d) Executor: após a leitura dos dados persistidos pelo editor, realiza a execução dos mapas e eventos das entidades, recebendo as entradas dos botões de direcionais (movimentação) e interação do jogador e alterando os estados do jogo de acordo com a programação dos eventos.

A Figura 5 apresenta o fluxo de utilização do editor. Após finalizar a edição, o executor realiza o carregamento do modelo de jogo criado e utiliza um fluxo diferente para a execução do jogo. Este fluxo também sugere a hierarquia dos objetos na modelagem utilizada (vide seção 3.2).

Figura 4 - Fluxograma da ferramenta



Fonte: elaborado pelo autor.

3.2 IMPLEMENTAÇÃO

Para o desenvolvimento da ferramenta foi utilizado o motor de jogos Unity e a biblioteca `SimpleJSON.NET` (BOLDAI AB, 2011) para persistir os dados no formato `JSON`. Os ícones e imagens utilizados foram do pacote `DawnLike` (DRAGONDEPLATINO, DAWNBRINGER, 2013). Algumas imagens utilizadas foram editadas com o software Aseprite.

Como a engine Unity utiliza-se de cenas para realizar suas operações, foram criadas três cenas para melhor organizar a ferramenta:

a) `MenuScene`: cena que possui o fluxo de menu inicial, levando a criação de um novo projeto ou ao carregamento de um projeto existente;
b) `EditorScene`: cena do editor da ferramenta que possui as janelas para a programação visual e montagem do jogo;
c) `PlayScene`: cena do executor que realiza o loop de jogo conforme a programação realizada no editor.

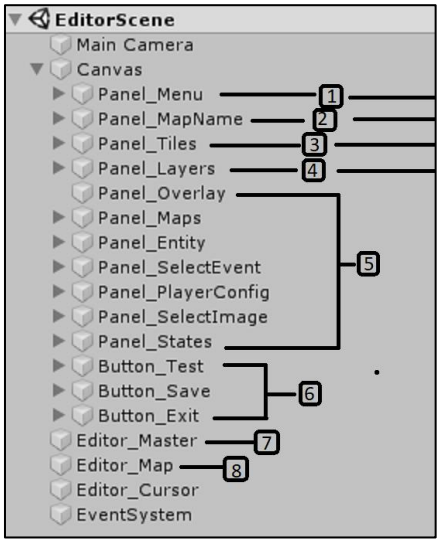
O principal ponto em comum dentro destas três cenas é a modelagem dos elementos e a persistência dos dados, ambas explicadas na seção 3.2.3.

3.2.1 EditorScene

A Figura 6 mostra a cena do editor da ferramenta sendo possível observar que é composta de elementos dentro de um `Canvas`. Em seguida, o Quadro 6 traz uma breve descrição do funcionamento de cada objeto em cena. Vale notar

que os painéis não mostrados no lado direito na lista hierárquica de objetos utilizado na cena do editor são ativados durante as interações do usuário, conforme o fluxograma da Figura 5.

Figura 5 – Hierarquia dos objetos utilizados na cena do editor



Fonte: elaborado pelo autor.

Quadro 6 - Objetos em cena no cena do editor

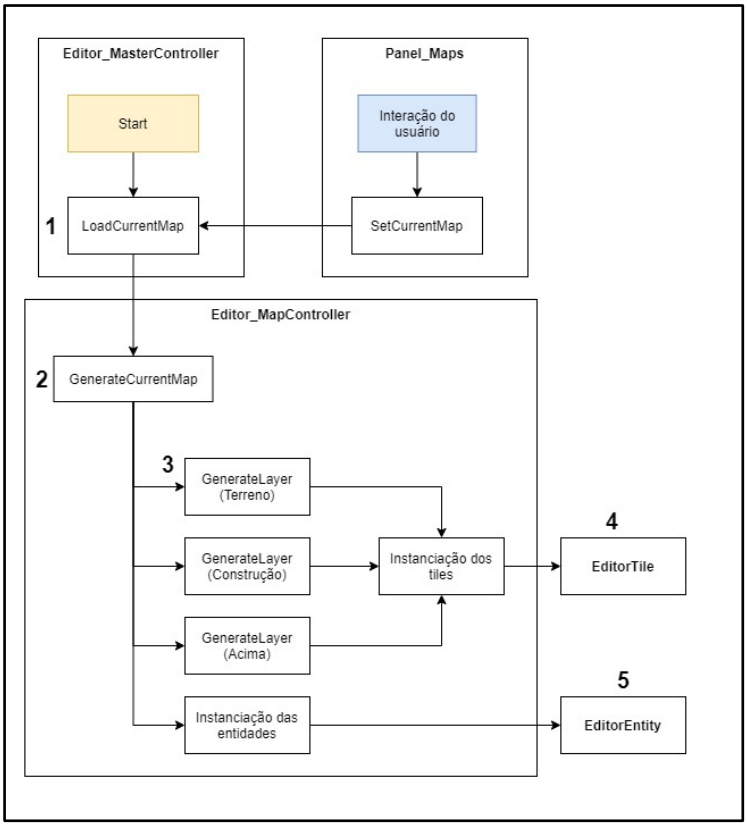
Identificador	Descrição
1	Painel contendo os botões para abertura das telas para cadastro de mensagens e configurações do jogador.
2	Painel mostrando o nome do mapa atual em edição.
3	Painel com a ferramenta de pintura do mapa atual, contendo os `sprites` disponíveis para pintura assim como opção de `Adicionar` ou `Remover` um *sprite*.
4	Painel para escolha da camada de edição, seja de pintura (camadas 1, 2 e 3) ou criação de entidades (camada 4).
5	Objetos contendo as telas de configuração disponibilizadas pelo editor.
6	Botões do fluxo de execução, salvamento e saída da ferramenta.
7	Objeto que contém o script `Editor_MasterController`, responsável por orquestrar todo o fluxo principal do editor.
8	Objeto que contém os objetos visuais e de interação com o mapa em edição atualmente, seja para a pintura de *sprites* ou para a criação de entidades.

Fonte: elaborado pelo autor.

Os objetos de tela (Quadro 6.5) são ativados através das entradas do usuário. Para a padronização dos painéis do editor, foi realizada a criação da interface `IEditorPanel`. Esta interface possui somente o método `DialogOpened` para o carregamento das informações de responsabilidade do painel. Os painéis são criados utilizando o `Canvas` do Unity, uma biblioteca destinada a criação de interfaces visuais para o usuário. Através do método `OpenPanel` e `ClosePanel` da classe `Editor_MasterController` é possível realizar a abertura e fechamento destes painéis, recebendo uma instância do mesmo para a configuração apropriada por parte do responsável pela abertura. Os painéis são responsáveis por receber entradas do usuário. Informações relacionadas ao uso destes painéis podem ser verificadas na seção 3.3.

O ponto principal do editor são a criação dos mapas, comandada pela classe `Editor_MapController`. Conforme mostrado na Figura 7, a classe é chamada através do método `LoadCurrentMap` (Figura 7-1), que pode ser executado na abertura do editor ou através de interação do usuário. O principal método da rotina de criação do mapa é o `GenerateCurrentMap` (Figura 7-2) que usa várias chamadas ao método `GenerateLayer` (Figura 7-3) para instanciar os `prefabs` de `tiles`. Estes `tiles` possuem o componente `EditorTile` (Figura 7-4), responsável por tratar os cliques do usuário e alterar seu `sprite` de acordo com o selecionado no editor. Posteriormente, o método `GenerateCurrentMap` faz o instanciamento das entidades cadastradas que possuem o componente `EditorEntity` (Figura 7-5) que, similar ao componente `EditorTile`, trata os cliques para a edição das entidades.

Figura 6 - Diagrama de geração dos mapas no editor



Fonte: elaborado pelo autor.

3.2.2 PlayScene

A cena `PlayScene` possui como intuito a execução do jogo criado dentro do editor. Para isso, possui o `GameMasterBehaviour` como classe principal; a classe `GameState` possui o estado do jogo atual; as classes `EntityBehaviour` e `PlayerBehaviour` para controle dos `GameObject` de entidades e jogador, respectivamente; e a classe `MessagePanel` para exibição de mensagens em tela.

A principal implementação no lado da execução são os eventos que herdam da classe `GameEvent`, presente nos modelos (ver seção 3.2.3). Cada evento deve implementar os seguintes métodos:

a) `GetNameText`: retorna o nome do evento para ser usado como informativo do usuário em tela;
b) `GetDescriptionText`: retorna uma descrição dos parâmetros para ser usado como informativo do usuário em tela;
c) `Execute`: método chamado na execução do evento;
d) `Update`: método chamado em cada loop de Update do Unity, para caso o evento seja de execução contínua.

Dentro de um evento, implementado herdando da classe `GameEvent`, existe o método `Execute` onde é realizada a definição da mensagem e exibição em tela, assim como o método `Update`, onde é possível realizar ações a todos os `frames`. A variável `finishedExecution` é utilizada para que o evento seja considerado finalizado.

Assim como o editor, a cena de execução possui uma rotina similar de instanciação de mapas dentro da classe `GameMasterBehaviour` seguindo os métodos `InstantiateMap`, `InstantiateLayer` e `InstantiateEntities`. Porém, eles utilizam menos componentes de interação visto que a lógica é ditada principalmente pela própria classe `GameMasterBehaviour` e pela classe `PlayerBehaviour`. Esta classe faz o controle de movimento e interação do jogador para a execução dos eventos supracitados. A Figura 8 mostra a comparação do instanciamento do mapa do editor com o mapa da execução do jogo.

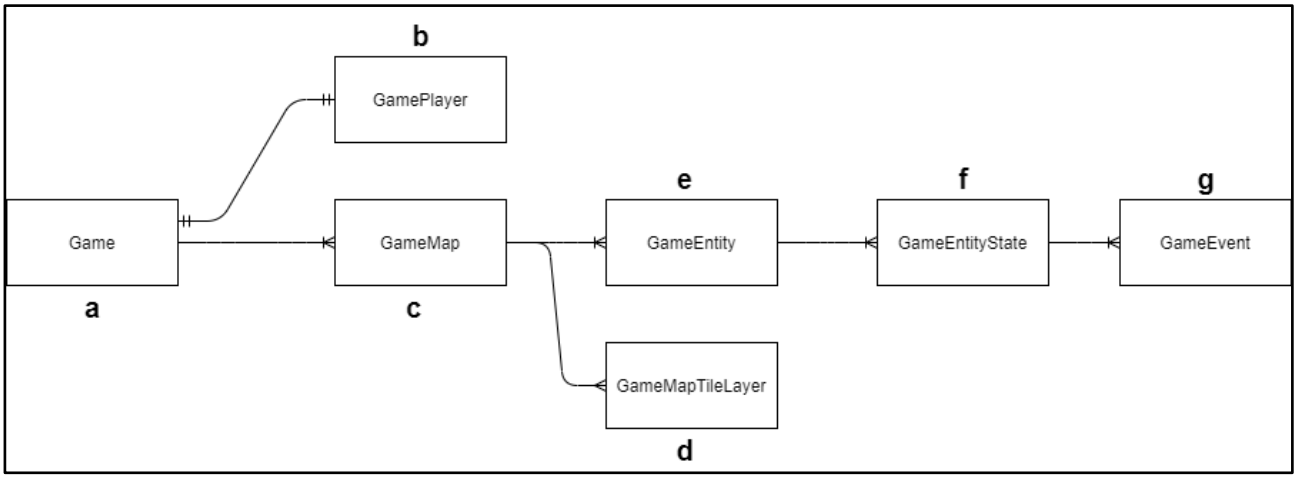
Figura 7 - Comparação do mapa de edição e mapa de execução do jogo

Fonte: elaborado pelo autor.

3.2.3 Persistência

Conforme mostrado no diagrama da Figura 9, a modelagem da ferramenta é realizada visando persistir os dados necessários para a execução do jogo de acordo com a configuração realizada através dos painéis do editor. Em seguida, o Quadro 8 descreve brevemente cada modelo utilizado e seu papel dentro do contexto da ferramenta.

Figura 8 - Modelagem dos objetos para persistência



Fonte: elaborado pelo autor.

Quadro 8 - Descrição dos objetos da modelagem para persistência

Identificador	Descrição
a	`Game` contém uma lista de mapas do jogo e a referência ao objeto com configurações do jogador;
b	`GamePlayer` contém informações do jogador, como imagem e mapa inicial do jogo;
c	`GameMap` possui informações do mapa em questão e referência às camadas pintadas no editor, além da lista de entidades existentes no mapa;
d	`GameMapTileLayer` contém um array bidimensional de inteiros que representam os *sprites* pintados no editor nesta camada, utilizados para renderização dos gráficos em tela;
e	`GameEntity` possui as informações de identificador e estados de uma entidade;
f	`GameEntityState` possui as condições de execução de um estado e uma lista dos eventos a serem executados;
g	`GameEvent` é uma classe abstrata que possui as informações referentes a um evento, variando de acordo com a implementação.

Fonte: elaborado pelo autor.

Para a persistência dos dados foi criada a interface `IPersistent` para obter e definir dados de uma classe que a implementa. A classe `PersistenceData` age como um dicionário de informações genéricas mas serializáveis. E a classe `Persistor` realiza a serialização e deserialização utilizando variáveis de tipo primitivas ou `PersistenceData`.

Também foram criadas alguns métodos de extensão que convertem objetos complexos para tipos primitivos, como é o caso da `struct Vector2`. No contexto da ferramenta, os objetos da classe `GameEntity` são entidades e os eventos persistidos por `GameEvent` fazem o papel de componente (contendo dados) e sistema (executando operações), similar a arquitetura utilizada na Unity.

3.3 USO DA FERRAMENTA

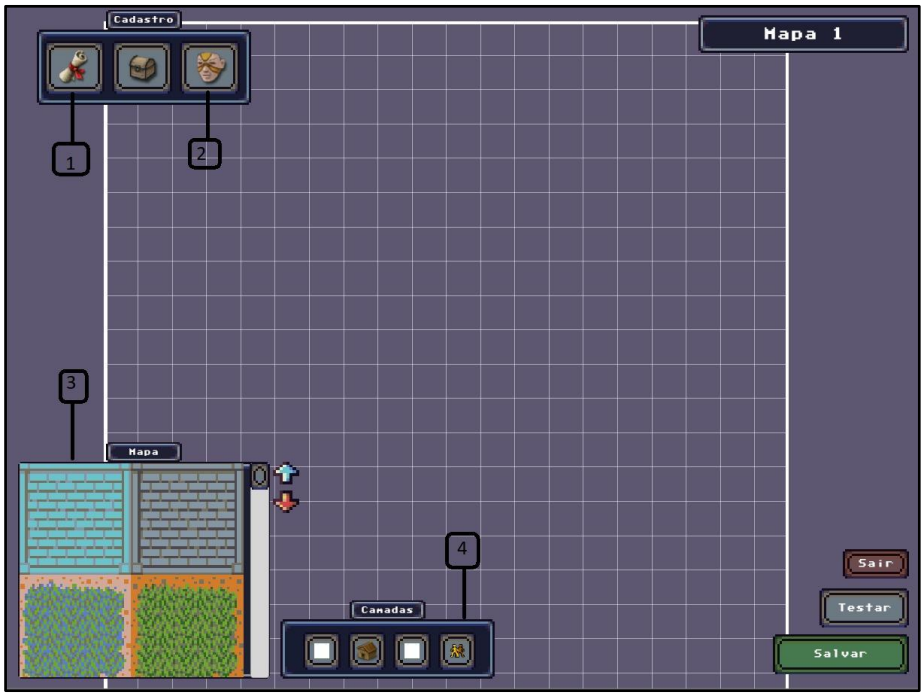
Esta seção se destina a descrever a utilização de todas as funcionalidades da ferramenta. Na primeira seção são apresentados conceitos relacionados ao fluxo e ao jogador, na segunda seção são apresentados os conceitos de entidades e seus eventos, na terceira seção são apresentadas informações de como utilizar estados, interruptores e variáveis para a criação de lógica, e na quarta seção é detalhado o processo de criação de um jogo de teste dentro da ferramenta.

3.3.1 Execução, Mapas e Jogador

O fluxo dos jogos criados pela ferramenta é baseado nos mapas criados. Para criar um mapa, deve-se clicar no botão Mapas (Figura 10-1). Através deste menu, é possível preencher nome e tamanho para o mapa criado. Após a criação, é possível selecionar o mapa para edição em tela através do botão `Selecionar mapa`. Após selecionado, o mapa pode ser editado visualmente utilizando os painéis de *sprites* e camadas (Figura 10-3 e Figura 10-4, respectivamente). O painel de *sprites* permite selecionar um ícone para ser pintado em tela com o botão esquerdo do mouse. Os botões azul e vermelho permitem adicionar e remover um ícone do mapa, respectivamente. O painel de camadas permite pintar ícones em níveis distintos.

a) Camada de terreno: esta camada será renderizada abaixo do jogador e das outras camadas;
b) Camada de construção: esta camada será renderizada acima da camada de terreno e abaixo da camada que está acima do jogador, além do jogador não poder se movimentar através dos ícones desta camada;
c) Camada acima do jogador: esta camada será renderizada acima do jogador e das outras camadas;
d) Camada de entidades: esta camada não é editada visualmente e serve para a criação de entidades, conforme seção 3.3.2.

Figura 9 - Interface da ferramenta



Fonte: elaborado pelo autor.

Além da edição de mapas, é possível editar as configurações do jogador através do botão Jogador (Figura 10-2). É possível mudar a imagem do jogador e definir qual será o mapa e coordenadas X e Y que o jogador começará o jogo.

Os controles utilizados pelo jogador são as teclas direcionais para realizar movimento e o caractere "Z" para executar a interação com entidades. Para executar um jogo criado, é possível utilizar duas opções:

a) Menu inicial: o botão `Executar jogo` leva direto ao jogo criado pelo arquivo atual;
b) Testar: o botão `Testar`, após o carregamento de um jogo, leva a execução do mesmo.

3.3.2 Entidades e Eventos

Após selecionar a Camada de Entidades (ver seção 3.3.1) é possível clicar com o botão direito em uma posição do mapa para criar uma entidade. A tela de edição da entidade aparecerá, onde será possível definir seu nome, imagem, passável/impassável, tipo de execução e eventos. O nome da entidade é utilizado como identificador e é usado por alguns eventos. O ícone é a imagem que será mostrada para representar a entidade no mapa, e pode ser deixada em branco. É possível selecionar se a entidade será "passável" (jogador pode se movimentar através da entidade) ou não-passável (jogador não poderá se movimentar através da entidade). O tipo de execução define quando os eventos serão executados, conforme a lista:

a) Interação: os eventos serão executados quando o botão de interação for pressionado junto a entidade (ver seção 3.3.4);
b) Contato: os eventos serão executados quando o jogador se movimentar em cima da entidade (a entidade precisa ser passável);
c) Automática: os eventos serão executados quando o jogador entrar no mapa atual.

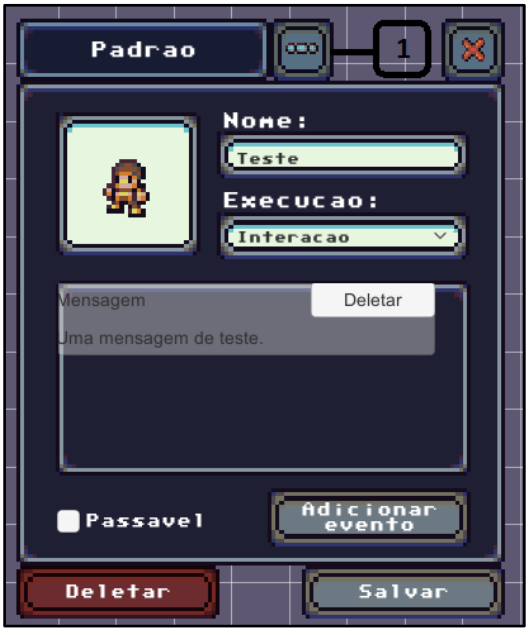
Os eventos são comportamentos que executam atividades sequencialmente. Cada estado (ver seção 3.3.3) de uma entidade pode possuir diversos eventos que serão executados sequencialmente de acordo com o seu tipo de execução. Os eventos são:

a) Mensagem: mostra uma mensagem em tela que pode ser fechada pressionando o botão de interação (ver seção 3.3.4);
b) Definir variável: define, soma ou subtrai um valor na variável informada (ver seção 3.3.3);
c) Ativar/desativar interruptor: define se um interruptor está ativo ou inativo (ver seção 3.3.3);
d) Mover entidade: move a entidade do nome informado para um local no mapa ou relativo a sua posição (utilizar o nome "Jogador" para mover o jogador);
e) Mudar mapa: realiza a transição para outro mapa;
f) Mudar imagem de entidade: altera a imagem da entidade informada.

3.3.3 Estados, Interruptores e Variáveis

Cada entidade pode possuir diversos estados, representando comportamentos de acordo com o contexto atual do jogo. Um estado pode ser ativado através de variáveis ou interruptores. Através do botão `Estados` (Figura 11-1), é possível criar novos estados definindo um nome e ativações. Uma ativação representa uma verificação de uma variável, um interruptor ou ambos. No cenário da verificação informada ser verdadeira, o estado será ativado. Para editar a imagem, tipo de execução e eventos de uma entidade, é necessário clicar no botão `Selecionar`. Estes eventos e tipo de execução serão utilizados caso este estado esteja ativo.

Figura 11 - Exemplo do painel de edição de entidades



Fonte: elaborado pelo autor.

3.3.4 CRIAÇÃO DE UM JOGO

Este seção apresenta o uso da ferramenta para a criação e execução de um RPG testando as funcionalidades implementadas e descritas conforme a seção 3.3.3. O jogo criado para realização destes testes se encontra no repositório `GIT` do projeto através do link https://github.com/guipaz/tcc. A Figura 12 apresenta a edição do mapa inicial do jogo, customizado através do menu inferior esquerdo.

Figura 12 - Edição do mapa inicial do jogo

Fonte: elaborado pelo autor.

A criação das entidades do jogo é demonstrada na Figura 13. É possível verificar que ao lado esquerdo da Figura 12 é cadastrada a entidade `Guardiao`, que mostra duas mensagens em sequência ao jogador ao executar uma interação, conforme campo o `Execução`. Ao lado direito da Figura 12, uma entidade com o `sprite` de vila realiza o evento de mudança de mapa ao detectar um contato com o jogador. Esta segunda entidade também é passável, permitindo que o jogador atravesse-a e dispare sua execução.

Figura 13 - Criação de entidades

Fonte: elaborado pelo autor.

A Figura 14 demonstra a execução da entidade Guardiao criada na Figura 13. Ao interagir com a entidade, uma tela com a mensagem cadastrada é mostrada.

Figura 14 - Mensagem executada ao interagir com entidade

Fonte: elaborado pelo autor.

A Figura 15 apresenta a tela de escolha de `sprites`, onde é possível selecionar dentre uma lista de texturas previamente cadastradas na ferramenta e, dentro da textura selecionada, um `sprite` específico. Estas texturas seguem um padrão de `16x16 pixels`.

Figura 15 - Tela de escolha de `sprites`

Fonte: elaborado pelo autor.

A lógica desenvolvida para entrar no terceiro mapa do jogo é demonstrada na Figura 16. A entidade `Anciao` executa um evento de ativação do interruptor `FalouAnciao` (Figura 16-1). Na entidade `Guardiao`, um estado é cadastrado para quando o interruptor `FalouAnciao` for ativado (Figura 16-2). Neste estado, um interruptor `LiberouPassagem` é ativado (Figura 16-3) e o Guardiao ativa seu terceiro estado, onde não está mais sendo apresentado a passagem está livre para o jogador prosseguir (Figura 16-4).

Figura 10 - Etapas na criação de estados e interruptores

Fonte: elaborado pelo autor.

4 RESULTADOS E CONCLUSÕES

Esta seção se destina a descrever os testes realizados com a ferramenta. Para validar as funcionalidades da ferramenta, foi realizada a criação de pequenos jogos de RPG utilizando todos os eventos e funcionalidades de estado disponibilizados. Os resultados foram satisfatórios do ponto de vista de implementação, visto que a ferramenta Unity se mostrou útil e capaz de realizar todas as funcionalidades necessárias. A principal utilização da ferramenta Unity foi o `Canvas` que se mostrou de grande valia para a criação rápida dos diversos painéis de cadastro necessários para a criação do editor.

A primeira implementação foi realizada utilizando a arquitetura Entity Component System (ECS); porém, tais conceitos se mostraram demasiadamente abstratos para a implementação do editor. A solução foi utilizar conceitos mais brandos e substituir a execução simultânea dos componentes e sistemas por eventos que seriam executados linearmente, similar ao trabalho correlato RPG Maker. Os eventos criados para a utilização do editor foram satisfatórios e a lógica disponível com a utilização das variáveis e interruptores é suficiente para um jogo pequeno de RPG. Conceitos interessantes de serem abordados dentro do projeto são conceitos de itens e batalhas, comuns dentro do gênero de RPG. Além disso, a movimentação pode ser melhorada utilizando um algoritmo de `pathfinding` como o `A*`.

A utilização do editor se tornou mais complexa do que o necessário, levando em conta o leiaute das telas e a relação dos modelos apresentados. Como os conceitos da seção 3.3 são abstratos, existe uma necessidade de exemplos práticos para uma boa compreensão do usuário.

Diante dos resultados apresentados, conclui-se que a ferramenta é capaz de criar jogos de RPG utilizando lógicas de programação visual e customização de mapas. Parte das funcionalidades propostas para o editor foram alcançadas, ainda que sua usabilidade tenha deixado a desejar do ponto de vista de facilidade de uso. A ferramenta Unity se mostrou poderosa, embora burocrática em alguns pontos (ver seção 4.1) e a biblioteca SimpleJSON.NET foi de grande valia para as rotinas de persistência e abstraiu muito da manipulação dos arquivos JSON.

As possíveis extensões propostas para seguir a implementação são: a) adicionar novos eventos para lógicas mais complexas ou funcionalidades visuais; b) implementar uma lógica de `pathfinding` como A* para a movimentação de entidades; c) adicionar o conceito de itens conforme usado normalmente no gênero RPG; d) implementar batalhas conforme usado normalmente no gênero RPG.

REFERÊNCIAS

ALVES, Adriana G. “Eu fiz meu game”: um framework para desenvolvimento de jogos por crianças. In: CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 6., 2017, Recife. **Anais…** Itajaí: Univali, 2017, p. 1-9. Disponível em: https://br-ie.org/pub/index.php/wcbie/article/view/7353/5151. Acesso em: 05 jul. 2020.

ALVES, Lynn et al. Ensino On-Line, jogos eletrônicos e RPG: Construindo novas lógicas. In: CONFERÊNCIA ELES, 4., 2004, Bahia. **Anais...** 2004. Bahia: UNEB, 2004, p. 49-58. Disponível em: http://www.comunidadesvirtuais.pro.br/ead/artigo.pdf. Acesso em: 05 jul. 2020.

BOLDAI AB. **SimpleJSON.NET**. 2011. Disponível em: https://github.com/mhallin/SimpleJSON.NET. Acesso em: 05 jul. 2020.

BITTENCOURT, João Ricardo. GIRAFFA, Lucia Maria. Modelando Ambientes de Aprendizagem Virtuais utilizando Role-Playing Games. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 14., 2003, Rio de Janeiro. **Anais...** Porto Alegre: PUCRS, 2003. p. 2-5. Disponível em: http://www.nce.ufrj.br/sbie2003/publicacoes/paper71.pdf. Acesso em: 02 jun. 2020.

CHAGAS, Artur Alves de Oliveira. **O transbordamento do lúdico e da biopolítica em jogos Massive Multiplayer Online**: um estudo sobre World of Warcraft. 2010. 157 p. Tese (Mestrado em Educação) - Universidade de São Paulo, São Paulo, 2010.

CORSO, Felipe L. **EasyEdu**: Editor web para jogos multitoque. 2017. 93 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

CRAWFORD, C. **The Art of Computer Game Design**, Washington State University, 1982.

DRAGONDEPLATINO. DAWNBRINGER. **DawnLike**: 16x16 Universal Rogue-Like Tileset v1.81 | OpenGameArt.org. 2013. Disponível em: https://opengameart.org/content/dawnlike-16x16-universal-rogue-like-tileset-v181. Acesso em: 02 jun. 2020.

ENTERBRAIN. **Make Your Own Game With RPG Maker**. Disponível em: https://www.rpgmakerweb.com/. Acesso em: 02 jun. 2020.

GRANDO, Anita; TAROUCO, Liane Margarida Rockenbach. O uso de jogos educacionais do tipo RPG na educação. **RENOTE - Revista Novas Tecnologias na Educação**. UFRGS, v. 6, n. 2, p. 3-8, 2008. Disponível em: https://seer.ufrgs.br/renote/article/view/14403/8308. Acesso em: 05 jul. 2020.

GURZYNSKI, Cleber; HOUNSELL, Marcelo; KEMCZINSKI, Avanilde. Análise de um jogo RPG educacional produzido pelo próprio docente, auxiliado por Ferramenta de Autoria. In: BRAZILIAN SYMPOSIUM ON COMPUTERS IN EDUCATION, 27., 2016, Uberlandia. **Anais...** 2016. Joinville: UDESC, 2016, p. 617.

HUIZINGA, Johan. **Homo Ludens**. São Paulo: Editora Perspectiva, 2010.

JSON. **Introdução ao JSON**. Disponível em: https://www.json.org/json-pt.html. Acesso em: 05 jul. 2020.

NINTENDO OF AMERICA. **The Legend of Zelda Instruction Booklet**. Japão, 1986. Disponível em: https://www.nintendo.co.jp/clv/manuals/en/pdf/CLV-P-NAANE.pdf. Acesso em: 05 jul. 2020.

PEREIRA, C.K., Andrade, F.M., Freitas, L.E.R. **Desafio dos Bandeirantes – Aventuras na Terra de Santa Cruz**, GSA, 1992.

PESSINI, Adriano; KEMCZINSKI, Avanilde; DA SILVA HOUNSELL, Marcelo. Uma Ferramenta de Autoria para o desenvolvimento de Jogos Sérios do Gênero RPG. COMPUTER ON THE BEACH, 6., 2015, Florianópolis. **Anais…** 2015. Florianópolis: UDESC, 2015, p. 71-80. Disponível em: https://siaiap32.univali.br/seer/index.php/acotb/article/view/6994/3938. Acesso em: 05 jul. 2020.

DIACRITICA. **File: Role Playing Gamers – Wikimedia Commons**. 2011. Disponível em: https://commons.wikimedia.org/wiki/File:Role_playing_gamers_%28II%29.jpg. Acesso em: 05 jul. 2020.

SQUIRE, Kurt. Video Games In Education. **International Journal of Intelligent Games & Simulation**. Cambridge, v. 1, n. 1, p. 2-4, 2003. Disponível em: . Acesso em: 02 jun. 2020.

SOBEL, Jonathan M. FRIEDMAN, Daniel P. **An Introduction to Reflection-Oriented Programming**. Indiana University. p. 1, 1996. Disponível em: https://web.archive.org/web/20100204091328/http://www.cs.indiana.edu/~jsobel/rop.html>. Acesso em: 02 jun. 2020.

WIJMAN, Tom. **The Global Games Market Will Generate $152.1 Billion in 2019 as the U.S. Overtakes China as the Biggest Market**. [S.I.]: Newzoo, 2019.