2
Trabalhos relacionados

A manutenção e ampliação dos jogos educativos são importantes para manter o interesse dos alunos no jogo e as linguagens de *script* podem facilitar sensivelmente esse processo. Este capítulo mostra alguns dos trabalhos que se assemelham a esta dissertação por desenvolverem jogos utilizando um motor de jogos e uma linguagem de *sript* ou por utilizarem uma abordagem que permite o professor modificar o jogo.

2.1
Papa-letras

O jogo Papa-Letras [15] proporciona ao jogador a chance de aplicar e testar conhecimentos adquiridos, uma vez que desenvolve a interpretação de figuras e o uso dos elementos gráficos usados para representá-la. Ele é uma adaptação do Pac-Man e passa-se em labirintos onde o jogador deve esquivar-se de monstros e coletar as sílabas corretas para formar a palavra representada pelo desenho no centro da tela, como mostra a Figura 2.1.

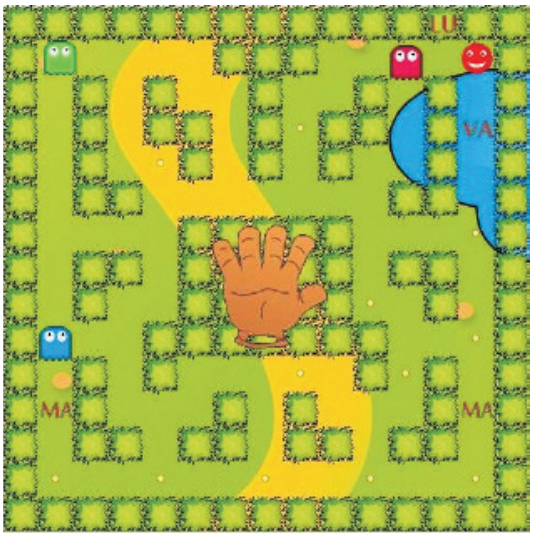

Figura 2.1: Ambiente do jogo Papa-letras. Neste labirinto o jogador precisa coletar as sílabas da palavra luva [15].

Ao tocar uma sílaba correta, isto é, que compõe a palavra apresentada, os

monstros se tornam vulneráveis, permitindo ao jogador derrotá-los temporariamente adquirindo assim mais pontos. Quando todas as sílabas necessárias para a escrita da palavra forem coletadas o jogador avança para o nível seguinte, aumentando assim a dificuldade do jogo. As primeiras fases apresentam palavras dissílabas. Em estágios mais avançados o jogo propõe palavras trissílabas ou polissílabas, compostas não apenas por sílabas canônicas (consoante-vogal), mas algumas com dígrafos, encontros consonantais, e outros elementos linguísticos, que geralmente representam dificuldade para os alunos em fase de alfabetização.

Nesse jogo, cada sílaba correta, que é coletada pelo jogador, se desloca para o canto superior direito da tela, informando ao jogador sua progressão naquele estágio. Também encontram-se espalhados pelo cenário objetos que, quando coletados, somam uma maior quantidade de pontos ao jogador.

Este jogo foi implementado utilizando o motor de jogos Love 2D e a linguagem de *script* Lua. O *framework* criado neste trabalho também segue esta ideia, ele foi desenvolvido utilizando o motor de jogos C++Play e a linguagem de *script* Lua.

2.2 Rosquinhas mágicas

No jogo Rosquinhas mágicas [16] não há interações externas, ou seja, o computador joga com ele mesmo.

Este jogo surgiu do estudo de *pathfinding*, que é basicamente a habilidade de um objeto ser capaz de mover-se de um ponto A para um ponto B. E isto é útil, por exemplo, para determinar o caminho a ser percorrido pelos personagens dos jogos. Cada vez mais podemos ver jogos de computador nos quais os personagens movimentam-se pelo mapa identificando obstáculos, e desviando deles da melhor forma possível.

Então, a ideia deste estudo, que deu origem ao jogo Rosquinhas mágicas, é construir um ambiente simulado capaz de possibilitar a realização de buscas inteligentes, bem como poder experimentá-lo de forma prática em um ambiente em tempo real.

A história do jogo é a seguinte: Em um mundo distante governado por bruxas e feiticeiros, uma bruxa chamada Aurora resolveu utilizar sua magia para animar objetos. Como estava em casa resolveu procurar objetos que poderiam ser interessantes de ser animados. Foi quando olhou em sua mesa, que estava pronta para o café da tarde e viu algumas rosquinhas. Não pensou duas vezes e resolveu dar a estas rosquinhas a capacidade de poderem se locomover e atribuiu a elas os sentimentos mais básicos. Observando a reação

destas rosquinhas a bruxa pode ver que as rosquinhas possuíam uma atração natural em buscar o bule de café em sua mesa e que algumas rosquinhas tinham tanta inveja que ao invés de perseguirem o bule, perseguiam as rosquinhas que conseguiam encontrar o bule. Viu também que as rosquinhas que atingiam seu objetivo eram gananciosas demais a ponto de não notar o que acontecia a sua volta, derrubando tudo. Sem perceber a tempo a bruxa viu que sua mesa de café estava totalmente destruída pelas rosquinhas mágicas e ficou furiosa. Para punir as rosquinhas, a bruxa, enviou as rosquinhas a um mundo imaginário distante com a pena de terem que perseguir um bule de ouro eternamente e sofrendo com seus instintos básicos de inveja e ganância (Figura 2.2).

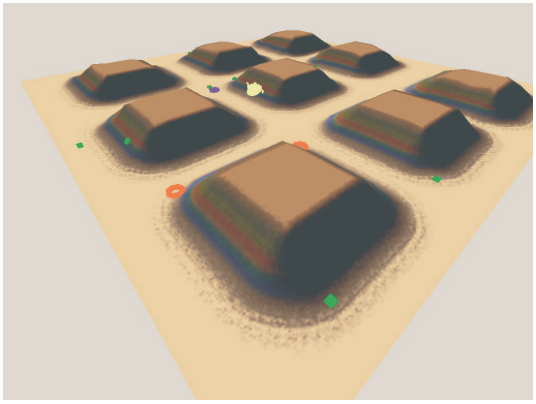

Figura 2.2: Jogo em execução: Rosquinhas em busca do bule de ouro [16].

O jogo possui as seguintes regras: rosquinhas gananciosas são penalizadas com 5 caminhos bloqueados a cada vez que um bule é encontrado; rosquinhas invejosas não suportam estar próximas entre si e atacam umas as outras, fazendo que quando isto ocorra sejam deslocadas para locais distantes de onde se encontram; toda vez que uma rosquinha gananciosa encontra um bule, ganha um ponto; toda vez que uma rosquinha invejosa ataca uma rosquinha gananciosa a rosquinha gananciosa perde um ponto e as rosquinhas invejosas ganham um ponto; o jogo acaba quando o *score* chegar a 10 pontos.

O jogo Rosquinhas Mágicas utilizou um programa hospedeiro implementado em C++ e a linguagem de *script* Lua. O *framework* desenvolvido nesta dissertação também utliza um motor de jogos em C++ e Lua. Além disso, assim como no jogo Rosquinhas mágicas, o *framework* permite que o professor modifique o conteúdo do jogo.

2.3 RPGEDU

O RPGEDU [17] é um projeto de pesquisa que tem por objetivo o desenvolvimento de um jogo educativo 3D, do tipo *Role Playing Game* (RPG).

O projeto engloba diferentes conteúdos de 5ª a 8ª série apresentando-os ao usuário em forma de atividades pedagógicas espalhadas pelo jogo.

Para o desenvolvimento deste jogo utilizou-se o motor de jogos Ogre3D e a linguagem de *script* Lua para respostas de eventos.

Dentro do ambiente do jogo existem NPCs (*Non Player Characters*), que são personagens controlados pelo computador que interagem com o usuário constantemente e eventualmente apresentam atividades pedagógicas a serem resolvidas pelo usuário.

O conteúdo pedagógico aborda várias disciplinas diferentes. Entre suas atividades pedagógicas, pode-se encontrar perguntas de múltipla escolha, tarefas de associação com mouse, entre imagem e significado, e tarefas de ordenação de imagens utilizando o mouse. Um exemplo de atividade, utilizando o conceito de associação, é apresentado na Figura 2.3 .

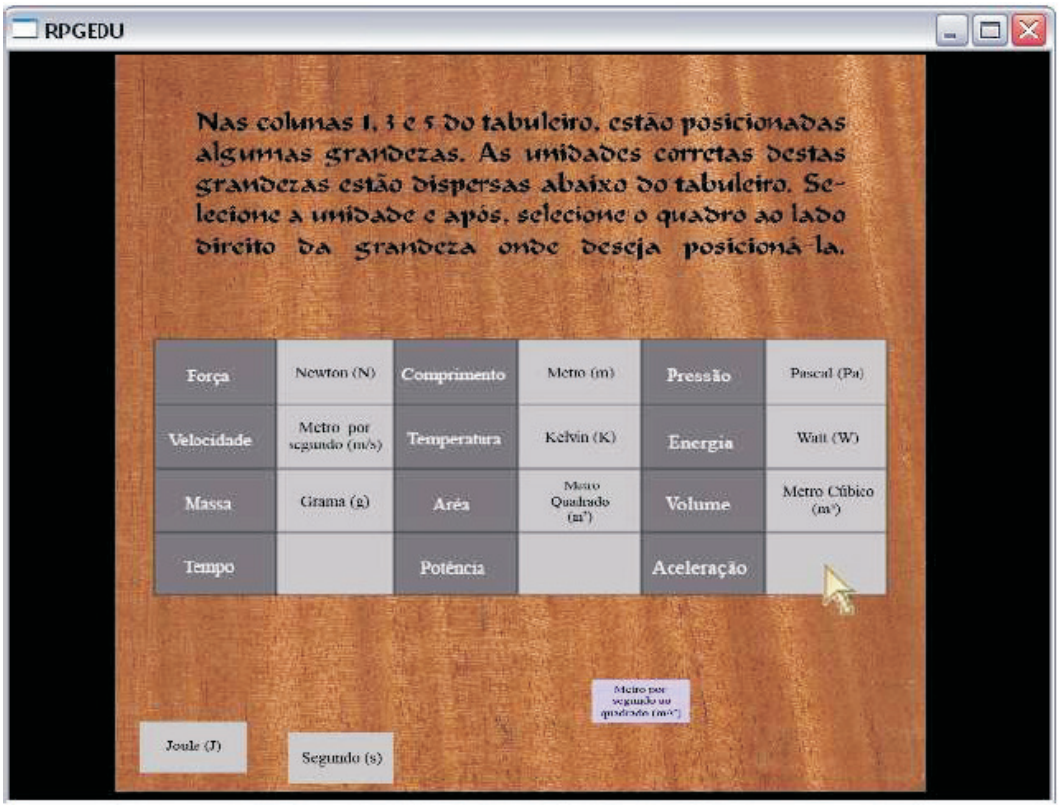

Figura 2.3: Atividade do RPGEDU utilizando o conceito de associação [17].

O jogo RPGEDU se assemelha ao *framework* desenvolvido neste trabalho pela forma que foram desenvolvidos, ambos utilizam um motor de jogos e uma linguagem de *script*.

2.4 GeoEspaçoPEC

GeoEspaçoPEC [18] é um jogo educativo que tem como intuito principal estimular a aprendizagem da geometria espacial. O jogo é composto por passatempos e possui uma estrutura de RPG (*Role Playing Game*) eletrônico, onde o jogador explora um ambiente tridimensional e através de pistas e desafios permite ao aluno utilizar seus conhecimentos de forma lúdica e divertida.

A Figura 2.4 mostra o ambiente do jogo, que é uma biblioteca com passagens secretas que serão descobertas ao longo do jogo. A biblioteca é composta por vários ambientes, como o vão central e suas salas de leitura, a catacumba e uma sala secreta. O vão central é onde se desenrola a maior parte do jogo e oferece acesso às demais áreas. O acesso à catacumba e a sala secreta será liberado à medida que o jogador transpõe e vence os desafios apresentados. O objetivo principal do GeoEspaçoPEC é encontrar um "Livro Raro" escrito por Leonardo da Vinci. O aluno obtém pistas matemáticas e passatempos para ajudá-lo a atingir o objetivo final e ganhar conhecimento de geometria espacial a cada novo nível. O personagem principal é um aluno do ensino fundamental que precisa ir a biblioteca estudar para a prova de matemática, que ocorrerá durante a semana. Ao entrar na biblioteca o aluno encontra um jornal sob a mesa com a seguinte manchete: "Ladrões de peças raras de Leonardo da Vinci atacam novamente". De acordo com o jornal a polícia desconfia que o próximo alvo seja aquela biblioteca. Após a notícia dos assaltos, o personagem encontra um aviso na mesa da ante-sala. O aviso diz que naquele lugar há um livro raro escrito por Leonardo da Vinci e, para evitar que tamanha raridade caia em mãos erradas, o personagem (jogador) decide encontrar este livro antes que os ladrões de antiguidades cheguem ao mesmo.

Figura 2.4: O ambiente do jogo GeoEspaçoPEC [18].

Para a implementação deste jogo utilizou-se o motor de jogos Panda3D

associada com a linguagem Python, que é uma linguagem *script* de alto nível multiplataforma. Estas características também foram utilizadas no desenvolvimento deste trabalho, o *framework* foi criado utilizando um motor em C++ e a linguagem de *script* Lua.

2.5 Colisões

No jogo Colisões [19], o aluno é desafiado a realizar "experimentos virtuais" em busca da compreensão de alguns conceitos fundamentais, como velocidade, energia e momento linear, ou seja, o jogo trata principalmente dos conceitos de colisões de partículas.

O jogo Colisões possui 5 desafios, onde envolve parte dos seguintes conceitos de colisões: Velocidade linear, conservação da energia, colisão elástica e colisão inelástica. A Figura 2.5 mostra o ambiente do primeiro desafio.

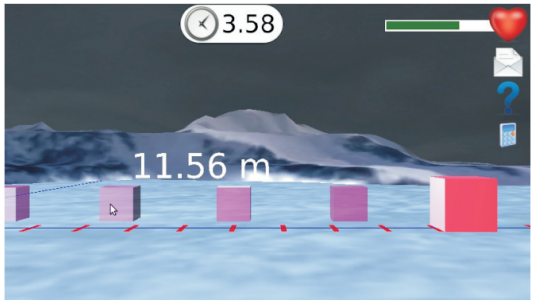

Figura 2.5: O ambiente do jogo Colisões. O Final da simulação do primeiro desafio, neste ponto o jogador deve calcular a velocidade que tem o bloco [19].

Inicialmente, o jogador se depara com o conceito de velocidade, a partir do estabelecimento de posição e tempo do bloco, num ambiente que lembra um lago congelado. A seguir, é abordado o conceito de choque elástico e inelástico, permitindo a investigação, também experimental, da conservação do momento linear e o comportamento diferenciado da energia. Finalmente, o conceito de momento linear como grandeza vetorial aparece no último nível.

Para ter acesso ao desafio seguinte, o jogador já deve ter completado os desafios anteriores e para aumentar a interatividade do jogo, o jogador possui uma vida que diminui com o passar do tempo ou caso o jogador responda errado à algum dos desafios. Para restituir a vida, o jogador deve responder corretamente aos desafios.

Para o desenvolvimento e implementação do jogo foram utilizados o motor de jogos Blender, que permite a criação de aplicações interativas em 3D, e a linguagem de *script* Python. Assim como o jogo Colisões, o *framework* desenvolvido utiliza um motor de jogos e uma linguagem de *script*.

2.6 Supermercado Virtual

O jogo Supermercado Virtual [20] busca evidenciar a aplicação prática da matemática no cotidiano explorando a habilidade dos jogadores em conversão de medidas, categorização, cálculo mental, frações e unidades de medidas. No entanto, nenhum tipo de exercício é proposto, a matemática é explorada de forma contextualizada às tarefas comuns em um supermercado como pesar verduras, verificar a validade de produtos, adequar as comprar ao orçamento, pagar e conferir o troco e assim por diante. Desta forma, diversos conceitos da matemática são trabalhados, tais como, multiplicação, cálculo mental, frações, informações monetárias e transformações de unidades. Além disso, foram priorizados ambientes abertos, ou seja, sem um roteiro único a ser seguido, permitindo assim inúmeras possibilidades de exploração.

Ao iniciar o jogo, o aluno recebe uma lista de produtos que devem ser comprados e a determinada quantidade de dinheiro que ele tem disponível para esta compra. Tanto a lista quanto a quantidade de dinheiro são configuradas pelo professor. O objetivo do aluno é encontrar os produtos da lista no mercado e efetuar a compra estando sempre atento aos prazos de validade e ao valor em dinheiro disponível. Após receber a lista de compras o aluno pode "brincar" no supermercado deslocando-se entre as estantes de produtos e conduzindo seu carrinho de compras. A figura 2.6 ilustra a visão superior do supermercado onde o aluno conduz seu carrinho e visualiza também os outros compradores, as estantes com produtos, os caixas para pagamento, entre outros objetos que compõem um supermercado na realidade.

Ao encostar o carrinho em uma estante, o aluno visualiza esta em detalhe, onde é possível consultar os dados dos produtos e colocá-los no carrinho com a intenção de comprá-los. A figura 2.7 ilustra a estante e a etiqueta de um produto.

Após julgar concluídas as compras, o aluno deve passar no caixa, onde os produtos serão totalizados e o valor final da compra é solicitado. A função educacional do caixa é fornecer a noção de valores monetários, pois para pagar a conta ele precisa decidir quais as cédulas serão entregues ao operador do caixa. A compra é concluída somente quando o valor que o aluno pagar for superior ou igual ao valor total dos produtos comprados. Depois de concluir a tarefa o aluno recebe um *feedback* das compras realizadas. Ele é informado se comprou produto vencido, produto que não foi solicitado e se deixou de comprar algum produto pedido na lista.

Além disso, o jogo permite que o professor defina a lista de produtos a serem comprados, a quantia em dinheiro a ser disponibilizada ao aluno e

Figura 2.6: Visão superior do supermercado [20].

Figura 2.7: Detalhe de um produto em uma estante do supermercado [20].

altere os valores dos preços dos produtos e dos prazos de validades, deixando o jogo sempre atualizado (Figura 2.8). Para facilitar a tarefa do professor em determinar a quantia de dinheiro a ser disponibilizada ao aluno para uma determinada lista de compras, o software gera automaticamente três possibilidades: o valor mínimo (selecionando as marcas mais baratas); o máximo (selecionando as marcas mais caras); e o valor médio (uma média entre as duas possibilidades anteriores). Ainda assim, se desejar o professor pode definir um valor arbitrariamente.

O *framework* desenvolvido neste trabalho também aborda esta ideia, ou

seja, no *framework* o professor pode modificar o conteúdo do jogo e criar diferentes jogos.

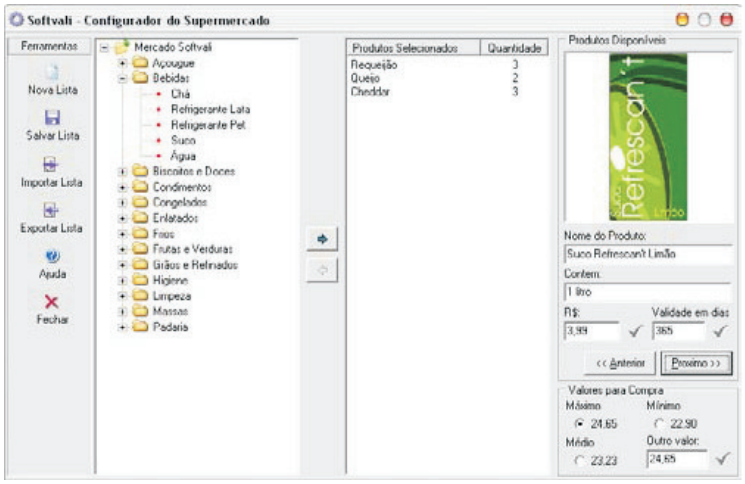

Figura 2.8: Interface do módulo do professor. O professor pode criar a lista de compras que desejar, definir a quantidade de dinheiro o jogador terá disponível e ainda pode alterar o preço e prazo de validade dos produtos [20].

Para o desenvolvimento deste jogo foi utilizada a ferramenta Flash MX. O módulo do professor foi criado usando a linguagem C++ e a integração entre os módulos foi feita utilizando XML. Todos os produtos do supermercado estão armazenados no banco de dados Firebird e são acessados somente pelo módulo do professor. Ao salvar uma lista de compras, um documento XML é gerado, bem como todas as imagens dos produtos. Ao entrar no supermercado as informações do documento XML são interpretadas e as imagens são carregadas.

2.7 AgentSheets

AgentSheets é uma ferramenta de simulação baseada em agentes, que permite que uma grande faixa de usuários finais (de crianças a profissionais) criem suas próprias simulações e jogos [21, 22]. Os agentes podem reagir a operações com *mouse*, entradas do teclado e movimentação ao seu redor podem alterar sua aparência.

Utiliza uma linguagem de programação visual baseada em regras, formadas por condições e ações. Para definir regras a um agente, o sistema oferece um editor de comportamento. O comportamento de um agente é definido com regras do tipo IFTHEN, contendo condições e ações. Com essa ferramenta os usuários podem criar pequenos jogos ou simulações sem a necessidade de utilizar uma linguagem de programação textual. A Figura 2.9 exibe um exemplo

de uso desse editor, onde as regras são formadas por condições que verificam se uma tecla foi pressionada, e ações que alteram o estado de um agente.

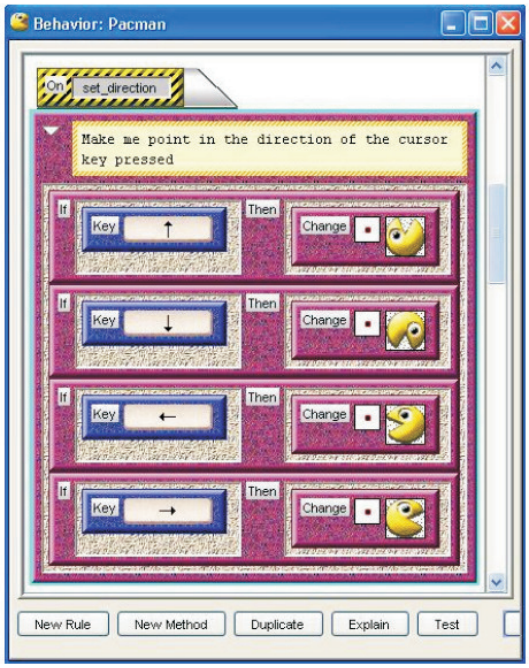

Figura 2.9: Editor de comportamento do AgentSheets [21, 22].

Ele foi criado com o propósito de ser uma ferramenta educacional exclusivamente dedicada à aprendizagem através da criação de jogos. Seu propósito é criar simulações em um ambiente puramente visual, requerendo o mínimo possível de programação em nível de código e, ao mesmo tempo, sendo uma ferramenta acessível e sofisticada em que os alunos possam desenvolver aplicações mais complexas ao desenvolver seu conhecimento.

Assim como o AgentSheets, o framework desenvolvido neste trabalho, premite que diferentes jogos sejam desenvolvidos, embora no framework o professor deva fazer as modificações via código.

2.8 Análise comparativa

Os jogos analisados possuem algumas diferenças e características em comum com o *framework* apresentado nesta dissertação. Os jogos Papa-letras, Rosquinhas Mágicas, RPGEDU, GeoEspaçoPEC, Colisões e o jogo criado neste trabalho utilizam um motor de jogos para agilizar o desenvolvimento do jogo e uma linguagem de *script* para tornar os jogos mais flexíveis. O único jogo que não é educativo é o Rosquinhas Mágicas. Na realidade, ele é um ambiente de simulação muito semelhante a um jogo, em que o computador joga contra ele mesmo. No entanto a ideia deste ambiente de

simulação é muito semelhante ao trabalho desenvolvido nesta dissertação, pois basicamente o fluxo de funcionamento dos dois baseia-se nos seguintes eventos: O usuário edita o *script* que será utilizado por um motor de jogos para executar a lógica desejada e depois, executa o motor de jogos responsável em colocar em prática todo o *script* programado. O jogo Supermercado Virtual também segue essa ideia, pois o professor pode fazer modificações e criar vários jogos. A diferença do jogo Supermercado Virtual, em relação ao jogo implementado nesta dissertação, é que o professor faz as modificações utilizando uma interface, ele não tem acesso à implementação do jogo. O AgentSheets segue essa mesma linha, esta ferramenta permite ao usuário modificar a lógica dos objetos e criar seus próprios jogos utilizando um ambiente puramente visual. Os outros jogos não abordam a possibilidade do usuário modificar o comportamento do jogo. Em relação ao conteúdo educativo, apenas o Supermercado Virtual, GeoEspaçoPEC e RPGEDU abordam a mesma matéria (matemática) que foi utilizada no jogo deste trabalho. Por fim, uma característica comum entre o jogo desenvolvido nesse trabalho e o papa-letras foi a criação de logs que capturam e registram em arquivo texto as respostas dos alunos.