UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS GUARAPUAVA
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET

CAROLINA MOREIRA OLIVEIRA

SERIOUS GAME COMO OBJETO DE APRENDIZAGEM PARA PROGRAMAÇÃO DE COMPUTADORES

GUARAPUAVA
2013

CAROLINA MOREIRA OLIVEIRA

SERIOUS GAME COMO OBJETO DE APRENDIZAGEM PARA PROGRAMAÇÃO DE COMPUTADORES

Trabalho de Conclusão de Curso de graduação apresentado à disciplina de Trabalho de Conclusão de Curso II, do Curso Superior em Sistemas para Internet, da Universidade Tecnológica Federal do Paraná – UTFPR, câmpus Guarapuava, como requisito parcial para obtenção do título de Tecnólogo.

Áreas de concentração: Ensino de Programação de Computadores, *Serious Games*.

Orientador: Prof. Ms. Diego Marczal

GUARAPUAVA

2013

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Guarapuava
Curso Superior de Tecnologia em Sistemas Para a Internet

ATA DE DEFESA DE MONOGRAFIA DE TRABALHO DE CONCLUSÃO DE CURSO DO CURSO DE TSI

No dia 04 de dezembro de 2013, às 11:00 horas, nas dependências da Universidade Tecnológica Federal do Paraná Câmpus Guarapuava, ocorreu a banca de **defesa da monografia** de Trabalho de Conclusão de Curso intitulada: "**Serious Game como Objeto de Aprendizagem para o Ensino de Programação de Computadores**" da acadêmica **Carolina Moreira** sob orientação do professor **Prof. Me. Diego Marczal** do Curso de Tecnologia em Sistemas para Internet.

Banca Avaliadora	
Membro	**Nome**
Orientador	Prof. Me. Diego Marczal
Coorientador	
Avaliador 1	Prof. Me. Andres Jessé Porfirio
Avaliador 2	Prof. Me. Hermano Pereira

Situação do Trabalho	
Situação	(x) Aprovado ( ) Aprovado com ressalvas ( ) Reprovado ( ) Não Compareceu
Encaminhamento do trabalho para biblioteca	(x) Pode ser encaminhado para biblioteca. ( ) Manter sigilo para publicação ou geração de patente.

Guarapuava, 4 de dezembro de 2013.

A Folha de Aprovação assinada encontra-se na Coordenação do Curso (ou Programa).

Aos meus pais, com imenso amor!

O importante é termos capacidade
de sacrificar aquilo que somos
para ser aquilo que podemos ser.

Charles Dubois

AGRADECIMENTOS

Não se pode substituir ninguém,
porque todo mundo é uma soma
de pequenos e belos detalhes.

(Richard Linklater)

A Deus e à Nossa Senhora Aparecida, por me atenderem todas as vezes que pedi forças e sabedoria para continuar.

Aos meus pais, Murilo e Valéria, que sempre apoiaram as minhas decisões. Se hoje concluo mais essa etapa, é graças a vocês. Amo vocês!

Ao meu namorado Eleandro, por estar ao meu lado, por conhecer meu lado mais ridículo e se divertir junto comigo. Te amo!

À toda a minha família, pelo apoio e compreensão.

Ao meu querido amigo Vinicius, pela amizade sincera.

Às amizades conquistadas durante todo o tempo de faculdade. Principalmente à Simone Dominico, pelo apoio e companheirismo.

Ao professor Diego Marczal, orientador e amigo, pelo amparo e contribuição para o sucesso.

Ao professor Andres Jessé Porfirio, pelo auxilio na melhoria do protótipo.

Ao acadêmico Andre Felipe Silveira, pela colaboração no Projeto de Recursos Digitais e, principalmente, pelo apoio em todo o desenvolvimento.

Aos professores da UTFPR, por toda dedicação no seu trabalho.

À banca avaliadora pela cordialidade com que atenderam ao convite.

ÍNDICE DE FIGURAS

SUMÁRIO

RESUMO

OLIVEIRA, Carolina Moreira. Serious Game como Objeto de Aprendizagem para Programação de Computadores. 2013. 64 f. Monografia (Graduação em Tenologia em Sistemas para Internet) – Universidade Tecnológica Federal do Paraná. Guarapuava, 2013.

A evolução tecnológica tem permitido o emprego de novas abordagens no processo de ensino-aprendizagem. Os *serious games* são destacados como uma categoria especial de jogos voltada à transmissão de conteúdos e finalidades educacionais específicas. Cabe ao jogador explorar o ambiente para compreender conceitos, e então resolver os problemas propostos. No contexto do ensino de Programação de Computadores, a abordagem empregada pelos *serious games* é potencialmente benéfica na motivação do aluno. Esta abordagem pode ser utilizada para diminuir o alto nível de evasão dos cursos na área de Computação. Frente a isto, esta pesquisa objetivou estudar sobre *serious games* e desenvolver um protótipo que se insere na área introdutória de Orientação a Objetos. Contudo, ressalta-se que para o protótipo foram apenas considerados os seguintes tópicos: (1) classes e objetos; (2) atributos e métodos; (3) estrutura condicional *if-else*; e (4) estrutura de repetição *while*.

Palavras-chave: *serious game*, jogos educacionais, ensino de Programação de Computadores.

ABSTRACT

OLIVEIRA, Carolina Moreira. Serious Game as Learning Object for Computer Programming. 2013. 64 f. Monografia (Graduação em Tenologia em Sistemas para Internet) – Universidade Tecnológica Federal do Paraná.Guarapuava, 2013.

The technological evolution has allowed the use of new approaches in the teaching-learning process. Serious games are featured as a special category of games aimed at the transmisson of specific contents and educational purposes. Concerns to the player the environment exploration to understand concepts and then solve the proposed problems. In the teaching of Computer Programming, the approach employed by serious games is potentially beneficial in student motivation. This approach can be used to reduce the high evasion level in Computing courses. Because of that, this research aimed to study about serious games and develop a prototype intended for introductory Object Oriented. However, it is noteworthy to mention that, for the prototype, were only considered the following topics: (1) classes and objects; (2) attributes and methods; (3) the if-else conditional statement; and (4) the *while* repetition statement.

Keywords: serious game, educational games, teaching of Computer Programming.

1 INTRODUÇÃO

Atualmente a baixa motivação e a alta taxa de evasão entre os acadêmicos dos cursos de Computação têm sido um fator preocupante em todo o mundo (MASCHIO, 2010). Tem-se registrado altos índices de reprovação nas disciplinas destes cursos e, além disto, sabe-se que a maioria dos acadêmicos que abandonam tais cursos fazem-no durante o primeiro ano (CLUA, 2008).

Evidencia-se também que pode ser visto como desafio encontrar meios efetivos que motivem estudantes de Computação ao aprendizado, visto que tecnologias das mais variadas estão presentes no cotidiano destes alunos. Portanto, abordagens tradicionais de ensino mostram-se insuficientes para atraí-los e motivá-los ao estudo (MASCHIO, 2013).

Especialmente no que se refere à motivação, os *serious games* têm se mostrado um auxílio valioso em diferentes domínios do conhecimento. Procuram amparar o processo de ensino-aprendizagem através da contextualização de conteúdos em um ambiente de jogo digital. Para tanto, utilizam de recursos vindos da indústria de entretenimento para tornar o jogo educacional mais atraente (PRIETO et al. 2005).

Diante do exposto, acredita-se que investir na intersecção de *serious games* com a área de Computação seja uma possibilidade promissora. Desta forma, este projeto se concentra no estudo e proposição de um *serious game* destinado ao ensino de Programação de Computadores.

1.1 OBJETIVOS

1.1.1 Objetivos Gerais

O presente trabalho teve como objetivo a realização de um estudo sobre *serious games*. A partir disto, pretendeu-se elaborar e desenvolver um protótipo para o ensino dos conceitos iniciais de Programação de Computadores por meio da linguagem Java (orientada a objetos). Deseja-se despertar o interesse de alunos

pelos cursos de Computação, como também motivar acadêmicos recém-ingressados nos mesmos cursos. Foram abordados os seguintes tópicos de Orientação a Objetos: (1) classes e objetos; (2) atributos e métodos; (3) estrutura condicional *if-else*; e (4) a estrutura de repetição *while*.

1.1.2 Objetivos Específicos

Foram objetivos específicos desse projeto:

1. Levantar conceitos relacionados à construção de *serious game*;
2. Estudar tecnologias para o desenvolvimento de um *serious game*;
3. Agregar conhecimentos sobre a linguagem Java e a plataforma Unity 3D;
4. Propor meios para tornar o aprendizado de Programação de Computadores, nos níveis inicias, mais divertido e atrativo por intermédio do protótipo de *serious game* desenvolvido;
5. Apresentar, através do protótipo desenvolvido, a disciplina de Programação de Computares de uma maneira mais atrativa, tanto nas aulas inicias de um curso de Computação quanto em feiras de profissões;
6. Disseminar o conhecimento, diante das oportunidades possíveis, adquirido em eventos da comunidade científica da área, tendo em vista o amadurecimento da proponente para o ingresso em curso de pós-graduação (mestrado).

1.2 JUSTIFICATIVA

A evasão dos cursos de graduação na área de Tecnologia e, principalmente, Computação tem sido preocupante. O curso de Tecnologia em Sistemas para Internet[1] por exemplo, mantém a média de 44 alunos ingressantes e permanência de menos 10% destes no último ano do curso[2].

Estudos procuram entender o porquê isso ocorre, visto que o mercado de trabalho na área é muito aquecido e sempre existe a demanda de profissionais.

1 UTFPR, Câmpus Guarapuava.
2 Informações retiradas do próprio câmpus.

Embora não caiba à pesquisa atual encontrar onde ocorrem possíveis falhas, procura-se motivar tanto alunos candidatos quanto ingressantes na área de Computação.

Os alunos de ensino médio, primeiramente, ao fim do terceiro ano, possuem ideias e inspirações diversas que complicam a escolha por um curso superior. Também, muitos possuem impressões iniciais de determinadas áreas e, a partir disto, descartam-as precocemente no processo de decisão.

Existem, neste sentido, as feiras de profissões que procuram auxiliar nesta escolha. Entretanto, conforme se observa, na maioria destes eventos, utiliza-se muitos textos informativos e poucos recursos interativos, que realmente promovem uma imersão na área. O projeto corrente procurou possibilitar um meio que permita ao aluno um contato mais direto com uma área base da Computação. Além disto, deseja-se imergi-lo em um primeiro contato com a Computação, com uma proposta menos exigente, e que possa ajudá-lo a decidir mais naturalmente pela área.

Em segundo lugar, existe o acadêmico recém ingressante na universidade. Tal indivíduo inicia o curso com muitas expectativas, ocorrendo possível divergência com a realidade imediata. A falta de motivação inicial nos cursos da área é comum, principalmente porque as aulas iniciais são bastante teóricas. Nesta progressão, demanda tempo para que o aluno consiga construir algo atraente em termos de Computação. A evasão, por consequência, ocorre sem que muitos alunos tenham real noção do que o curso oferece.

As duas realidades apresentadas, do aluno candidato e do ingressante, já justificam a necessidade de uma apresentação melhorada da área. Acredita-se, ainda, que o primeiro contato deve evidenciar a possibilidade de desenvolver aplicativos atraentes e também expor a dificuldade associada.

Portanto, um *serious game* que atue neste nicho pode consistir em uma contribuição interessante. Trata-se de um conceito emergente que pode transmitir tanto as possibilidades quanto as dificuldades da área de uma maneira mais dinâmica e prazerosa.

1.3 ESTRUTURA DA MONOGRAFIA

Depois desta introdução, no Capítulo 2, apresenta-se o referencial teórico da pesquisa, compreendendo aspectos educacionais, jogos e *serious games*. Cabe ao Capítulo 3 expor os trabalhos correlatos que motivaram e embasaram o desenvolvimento desta pesquisa. No Capítulo 4, discute-se sobre as tecnologias utilizadas, documentando também o motivo das referidas escolhas. O formalismo da proposta é relatado no Capítulo 5, seguido de detalhes do processo de implementação do *serious game* (Capítulo 6). As dificuldades enfrentadas ao longo do projeto se encontram no Capítulo 7. O Capítulo 8 relaciona os resultados e as limitações da pesquisa. Por fim, no Capítulo 9, apresentam-se as considerações finais. As referências bibliográficas encerram este documento.

2 REFERENCIAL TEÓRICO

O referencial teórico desta pesquisa toma por base os aspectos educacionais envolvidos (Seção 2.1), a influência dos jogos educacionais (Seção 2.2) e, por fim, os *serious games* propriamente ditos (Seção 2.3).

2.1 ASPECTOS EDUCACIONAIS

2.1.1 Ensino de Programação

Em cursos de Computação um dos objetivos está em torno da capacidade do aluno apresentar soluções para diversos problemas. Dentro disto, a área de maior evidência onde se concentram uma grande quantidade de pesquisas é a Programação de Computadores.

A Programação de Computadores incide em organizar um conjunto de instruções que a máquina deve executar para cumprir uma tarefa específica. Ao conjunto da-se o nome de algoritmo (ZANINI, 2013). Elaborar um algoritmo consiste em obter os passos que solucionem a tarefa determinada. Os passos são sequencializados em instruções de uma linguagem formal (de programação) a fim de comunicar ao computador o que deve ser executado.

A Programação de Computadores é, portanto, uma disciplina presente em cursos nas áreas da Computação. Em caráter introdutório, preocupa-se em fornecer aos alunos as bases necessárias para o desenvolvimento da lógica de programação e, em seu estudo, representar o raciocínio envolvido através de uma sequência de instruções conexas. Segundo Pimentel e Direne (1998, p.2) “a programação de computadores é uma tarefa de alta carga para iniciantes por duas razões principais: a falta do conhecimento de princípios de programação e consequentemente a falta de perícia”.

Durante o processo de ensino-aprendizagem de Programação de Computadores nota-se que grande parte dos alunos apresenta dificuldades em assimilar as abstrações envolvidas (PIMENTEL; DIRENE, 1998 e MASCHIO, 2013).

Ademais, a disciplina detém um dos maiores índices de reprovação em todas as instituições de ensino brasileiras (ZANINI, 2013). Trata-se, assim, de um ponto de reflexão por parte dos professores preocupados com a melhoria da qualidade no processo de ensino. Além disto, também ratifica a necessidade de alterações didáticas e metodológicas tanto de representação quanto de exposição dos conteúdos.

2.1.2 Lógica de Programação: Dificuldade no Aprendizado

Conforme antedito, os cursos nas áreas da Computação, possuem como uma das fundamentais a disciplina Programação de Computadores. Geralmente, essa disciplina é ofertada nos primeiros anos da graduação e recebe diferentes denominações, conforme a grade curricular, como Algoritmos, Programação de Computadores, Introdução à Lógica de Programação, entre outras (ZANINI, 2013).

Nessas disciplinas, o processo de aprendizagem de algoritmos é difícil para a maioria dos alunos, independente da metodologia abordada pelo professor, ou o nível de exigência da matéria (ZANINI, 2013). Entretanto, a dedicação dos acadêmicos é de grande valia, visto que os conteúdos embasam, praticamente, toda a graduação na área.

Neste sentido, percebe-se o papel fundamental do professor, onde diferentes abordagens se fazem necessárias para estimular os alunos a buscarem conhecimento (por vezes, extraclasse) e, mais basicamente, a quererem aprender. Desta forma, Zanini (2013, p.12) apud Beeker e Parker (2005) corrobora com o que se afirma e também justifica o contexto da corrente pesquisa, a saber:

> Fizeram um estudo com os alunos ingressantes no curso de Ciência da Computação, na universidade onde atuam, verificando que de 65% a 75% dos alunos ingressantes foram em busca do curso porque jogam videogame. Os autores afirmam que os problemas eram melhores compreendidos quando relacionados aos jogos. Esse estudo aponta os jogos como um contexto interessante para ser trabalhado em sala de aula, além de promover motivação, é um elemento conhecido dos alunos e pode facilitar a conexão entre o que se deseja ensinar e às estruturas cognitivas já existentes no aluno.

Segundo Zanini (2013), o método de ensino mais utilizado em disciplinas de Programação é a exposição de conteúdos seguida de exercícios. Cabe aos professores fornecerem enunciados e, a partir destes, os alunos alcançarem a solução algorítmica. A resolução pode ser feita inteiramente pelo professor, ou ainda pelos alunos com ou sem assistência docente. Através deste processo, torna-se evidente a dificuldade de interpretação de enunciados propostos, como também a baixa motivação para cumpri-los. Tudo isto, geralmente, incorre na resolução incorreta dos problemas fornecidos.

> Zanini e Raabe (2012) constataram que os enunciados apresentados nos livros didáticos adotados no Brasil, geralmente, são objetivos, não apresentam indícios do processo de resolução, não possuem exemplos e o contexto é puramente matemático, tornando-os mais abstratos. Conforme Jesus e Raabe (2010), os enunciados, comumente, resumem-se a “calcule isto” e “resolva aquilo” e o resultado do cálculo, da resolução, não é utilizado para algo que faça sentido dentro de um contexto maior. (ZANINI, 2013, p.13)

Com isso, percebeu-se que as pesquisas apontam que as dificuldades encontradas na resolução de problemas está diretamente relacionada à estrutura e/ou contexto do enunciado. Exemplo disto ocorre quando, muitas vezes, os alunos ficam estagnados por conta da falta de domínio no contexto específico do enunciado, não dos conteúdos.

Entretanto a pesquisa proposta presume que as dificuldades extrapolam a estruturação e contextualização dos enunciados. Acredita-se que muitos dos obstáculos advém da pobre motivação que os acadêmicos tem para estudar os conteúdos devidos.

2.2 JOGOS

Uma competição física ou mental, interpretado de acordo com regras específicas, com o objetivo de divertir ou recompensar os participantes, é o que se conhece por jogo (ZYDA, 2005). Os jogos digitais, por sua vez, destinados a videogames e computadores, correspondem a um dos setores que mais crescem na indústria do

entretenimento, pois conquistaram um lugar importante na vida das pessoas, independente da idade.

O faturamento do mercado de jogos digitais, em 2012, girou em torno de US$ 52 bilhões, segundo a empresa de consultoria DFC Intelligence, e ainda tem perspectiva de crescimento para US$ 75 bilhões até 2017 (MACHADO, 2013). Com tamanho faturamento, os jogos digitais assumiram um papel de destaque na cultura moderna. Sendo até compreensível, por diversos pesquisadores, que tal apelo seja o porquê dos referidos jogos serem tão atraentes na vida das pessoas.

O publico alvo acaba sendo seduzido pelo contexto e pelos gráficos que os jogos digitais oferecem. Assim, permanece longos períodos empenhados nos desafios propostos e passa a impressão de que nada é capaz de desconcentrá-lo. Também existe um viciante sistema de recompensas que progressivamente torna o jogador mais poderoso no contexto. Entretanto, dada a atraente imersão proporcionada, os jogadores terminam investindo horas e, com isso, ocupando tempo que poderia ser aproveitado por outras atividades mais produtivas. Por exemplo, o estudo.

Trata-se de um dos principais atritos em residências e instituições de ensino. Pois tanto os pais quanto os professores desejam que os adolescentes possuíssem o mesmo nível de interesse e de comprometimento dos jogos nos estudos.

Transferir a atenção que os estudantes dão aos jogos para as atividades educacionais está se tornando um desafio. Com isto, houve aumento no número de pesquisas que procuram encontrar formas de unir o ensino e a diversão, principalmente com o desenvolvimento de jogos educacionais (PRIETO et al. 2005).

Os jogos educacionais, por utilizarem de uma prática inovadora onde o aluno tem chance de aprendizado mais dinâmica, podem ser peças importantes no processo de ensino. Para atingirem finalidades educacionais, é primordial que os jogos tenham objetivos bem definidos e associados aos conteúdos das disciplinas envolvidas. Desta forma, é possível ajudar a promover o desenvolvimento de habilidades e estratégias a fim de ampliar a capacidade intelectual dos alunos.

Considerando que os jogos educacionais são aplicativos que visam atender necessidades vinculadas à aprendizagem, Prieto (2005) realiza importantes apontamentos. Primeiramente, salienta que os jogos educacionais devem possuir

objetivos pedagógicos bem definidos. Em segundo lugar, cita que a utilização deve estar inserida em um contexto e em uma situação de ensino. Além disto, o processo precisa ser orientado por uma metodologia que, através da interação, da motivação e da descoberta, facilite a aprendizagem de um conteúdo.

Neste sentido, os jogos podem receber diferentes nomenclaturas quando destinados ao contexto educacional. As denominações mais comumente utilizadas pela literatura da área são: jogos educacionais, jogos de ensino ou *serious games*.

2.3 SERIOUS GAMES

Hoje, observa-se o uso crescente de jogos como ferramenta de suporte ao processo de ensino e aprendizagem de crianças e adultos. Tais jogos, que se destinam a ensinar aspectos específicos de disciplinas ou que treinem habilidades operacionais e comportamentais, são denominados jogos sérios, do inglês *serious games*. Embora não se encontre uma definição específica e unívoca na literatura, sempre se fundamenta os *serious games* como jogos que ultrapassam o entretenimento e que levam a conteúdos e/ou benefícios específicos (MACHADO et al. 2011).

Pode-se afirmar que o surgimento dos *serious games* ocorreu nos anos 1980, com os simuladores desenvolvidos pelos Estados Unidos para área militar (ZYDA, 2005). Com o objetivo de treinamento, os *serious games* são aplicados para simular situações críticas que envolvam algum tipo de risco, tomada de decisões e, também, o desenvolvimento de habilidades específicas. Simulam situações, para fins de ensino-aprendizagem, em que o uso de um conhecimento seja necessário para a evolução no jogo proposto (PRIETO et al. 2005).

Um *serious game* procura combinar as etapas de ensino e de treinamento. Imediatamente depois, exige que o conhecimento adquirido seja utilizado na própria simulação oferecida. Desta forma Machado et al. (2011), quando voltado ao ensino, define a finalidade destes jogos em três categorias: conscientização, construção de conhecimentos e treinamento.

Os *serious games* utilizam estratégias vindas da indústria de entretenimento digital para tornar os jogos mais atraentes. Ao mesmo, tempo oferecem atividades que incentivam a construção de conceitos e a estimulação de funções psicomotoras.

O objetivo principal é verificar se o jogador conhece o assunto abordado, como também se sabe identificar ou propor novas soluções, realizando atividades dependentes do conhecimento abordado. Assim Machado et al. (2011) enfatiza que o termo passou a ser utilizado para identificar os jogos que possuem um propósito específico e que excedem a ideia de entretenimento. Dentro disto, oferecem outros tipos de experiências, como aquelas voltadas ao aprendizado e ao treinamento.

A fundamentação de que um *serious game* vai além da proposta de um jogo convencional é reforçada por (ZYDA, 2005 e MACHADO et al. 2011). Conforme Figura 1, observa-se que um *serious game* estende-se pelo embasamento pedagógico, ausente na maioria dos jogos comerciais. Entende-se como pedagogia, no âmbito dos jogos, como a atividade de educar ou instruir, dando o conhecimento ou ajudando a desenvolver uma habilidade (ZYDA, 2005).

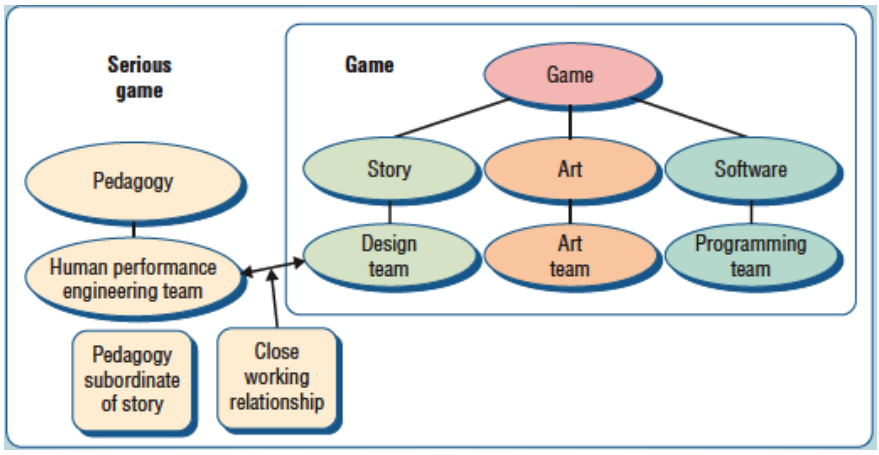

Figura 1: Do jogo para o *serious game* (ZYDA, 2005 p 26)

Acrescenta-se que, em um *serious game*, o jogador deve utilizar o raciocínio para superar obstáculos relacionados a um problema, ou ainda buscar possibilidades de minimizá-los. A construção de conhecimento, ainda que exija pré-requisito, amparam e aprimoram o raciocínio na área abordada.

Conclui-se que *serious games* são jogos que possuem como principal característica ensinar aspectos específicos, ou treinar habilidades operacionais e comportamentais. Surgiram da união entre a diversão dos jogos digitais e o embasamento teórico pedagógico. Ajudam a desenvolver os sistemas de avaliação e de planejamento para a tomada de decisões em domínios específicos. Todas as

características expostas ampliam as possibilidades de atuação destes jogos, com isso chamando a atenção na utilidade em diversas áreas do conhecimento.

3 TRABALHOS CORRELATOS

Diante da pesquisa bibliográfica sobre trabalhos correlatos ao proposto por esta pesquisa, observou-se poucas pesquisas que utilizassem jogos como elementos didáticos para o paradigma Orientado a Objetos. Conforme relatado nas seções deste capítulo, todos os ambientes foram introduzidos depois da década de 2000.

Uma primeira justificativa que se constata é do paradigma abordado ser recente na grade curricular da maioria das instituições de ensino. Em segundo lugar, remetendo-se à hipótese deduzida por (MASCHIO, 2013), os ambientes de ensino, quando inicialmente abordados pela Ciência da Computação, tratavam quase que exclusivamente da tutoria de Programação de Computadores. O direcionamento consequente ocorreu porque os especialistas envolvidos tinham a programação como área de estudo e trabalho. Com a evolução dos ambientes de ensino, atentando-se aos aspectos pedagógicos, se observou que o aprendizado de programação intermediado por ambientes de ensino era tão difícil quanto o de qualquer outro domínio de conhecimento. Com isso, as pesquisas de tais ambientes foram levadas às mais diversas áreas. O ensino de Programação de Computadores, por consequência, foi sendo abandonado, como objeto de estudo, por esses pesquisadores.

No início da década de 2000, entretanto, nota-se um reavivamento de pesquisas sobre ambientes de ensino para programação. Diante do cenário contemporâneo, é natural que os citados ambientes promovam o aprendizado do paradigma Orientado a Objetos, até mesmo pela própria dificuldade dos docentes mais maduros voltarem a antiga didática ao recente paradigma.

Com isso, é oportuno o uso dos recursos tecnológicos mais atraentes possíveis para promoverem a interação e a aprendizagem nos novos ambientes projetados. Dessa forma, uma direção promissora consistia no desenvolvimento de ambientes que utilizassem jogos como recurso didático. Alguns dos ambientes foram além e se fundamentaram, não no uso, mas na própria construção de jogos em referência à habilidade de programar.

Merecem destaque, atualmente, as ferramentas educacionais Alice[1],

1 Site oficial: http://www.alice.org

Greenfoot[2], Scratch[3], Belesminha e CodeSpell4[4]. Embora tenham sido projetados em diferentes contextos, estes cinco ambientes partilham de características semelhantes. Todas são visuais, com objetivo de promover o envolvimento e estimular o interesse em estudantes pré-universitários para a Programação de Computadores (FINCHER et al. 2010).

Também em correlação com a pesquisa proposta, os quatro ambientes são voltados à linguagem Java e procuram apoiar a aprendizagem dos conceitos de programação de uma maneira mais divertida e atrativa ao público contemporâneo. A sequência deste capítulo detalha cada um dos ambientes mencionados.

3.1 ALICE

Alice é um ambiente de visualização, animado e interativo em 3D. Tem por finalidade orientar programadores iniciantes a construir filmes animados e jogos em 3D, investindo, assim, em maior apelo visual. O ambiente promove o aprendizado e o aprofundamento nos conceitos introdutórios de orientação a objetos. A programação dos objetos que compõem os cenários é feita através de *scripts* baseados na linguagem Python (COOPER, 2000).

Como um aspecto positivo destacado, o Alice oferece um ambiente de desenvolvimento baseado em arrastar-e-soltar componentes, conforme Figura 2, bem como no preenchimento de lacunas com opções predeterminadas. Essas restrições impedem os estudantes de cometerem muitos dos erros sintáticos recorrentes. Todavia, o Alice demanda um esforço inicial de orientação tutorial para a familiarização com o funcionamento do ambiente.

2 Site oficial: http://www.greenfoot.org
3 Site oficial: http://www.scratch.mit.edu/
4 Site oficial: http://sites.google.com/a/eng.ucsd.edu/codespells/

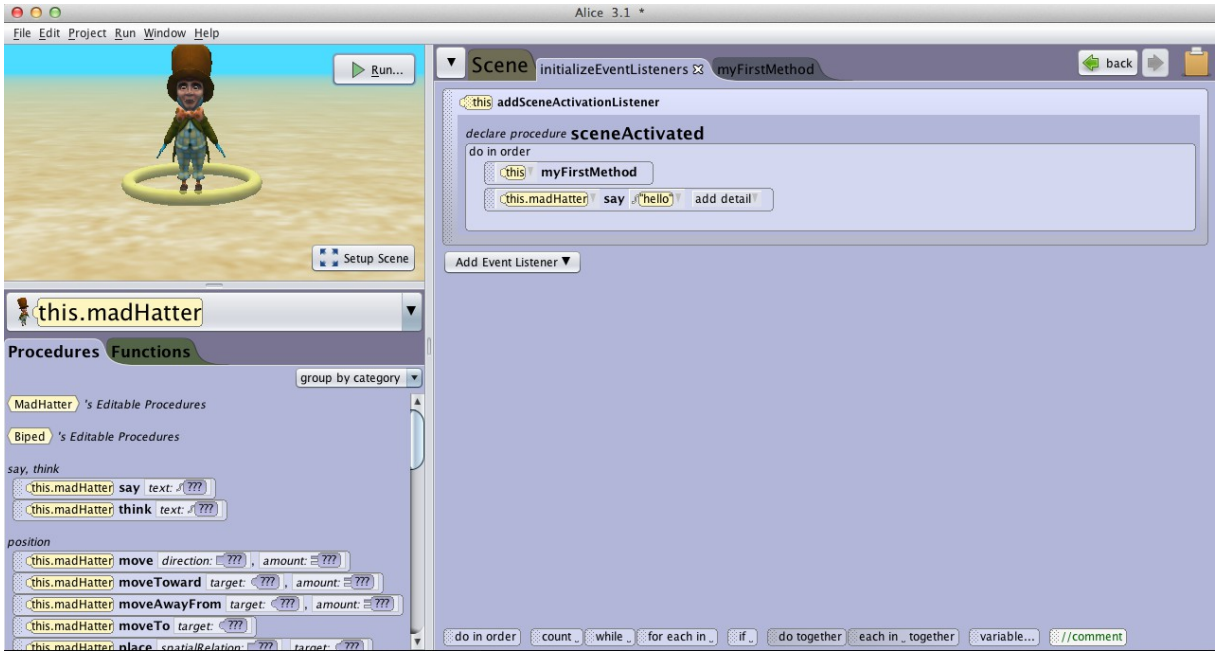

Figura 2: Interface do Alice

O software e o código fonte estão disponíveis gratuitamente no site institucional, como também material de apoio disponível para professores. A versão 3 do Alice permite o uso de códigos Java para a criação de projetos, além da interação entre a IDE Alice e alguma externa, como o NetBeans (FINCHER et al. 2010). O grupo Alice recentemente recebeu incentivo da Electronic Arts[5] (EA) e, já nessa versão, são utilizados os personagens do jogo The Sims.

3.2 GREENFOOT

O ambiente Greenfoot destina-se ao desenvolvimento interativo de pequenos jogos. Ele torna possível a construção desses jogos sem que seja necessário desenvolvê-los desde o início, pois muitos elementos já se encontram predefinidos. Desta forma, várias questões complexas da linguagem são abstraídas.

O Greenfoot (FINCHER et al. 2010) ainda utiliza a interface 2D, conforme Figura 3 para os jogos implementados, tornando-os menos atraentes do ponto de vista gráfico. Além disso, por se basear na linguagem de programação Java, a iniciação no ambiente consiste em um obstáculo para a maioria dos usuários, pois o

5 Desenvolvedora de jogos que lançou títulos como Battlefield, The Sims e Need For Speed.

desenvolvimento é fortemente acoplado à sintaxe dela.

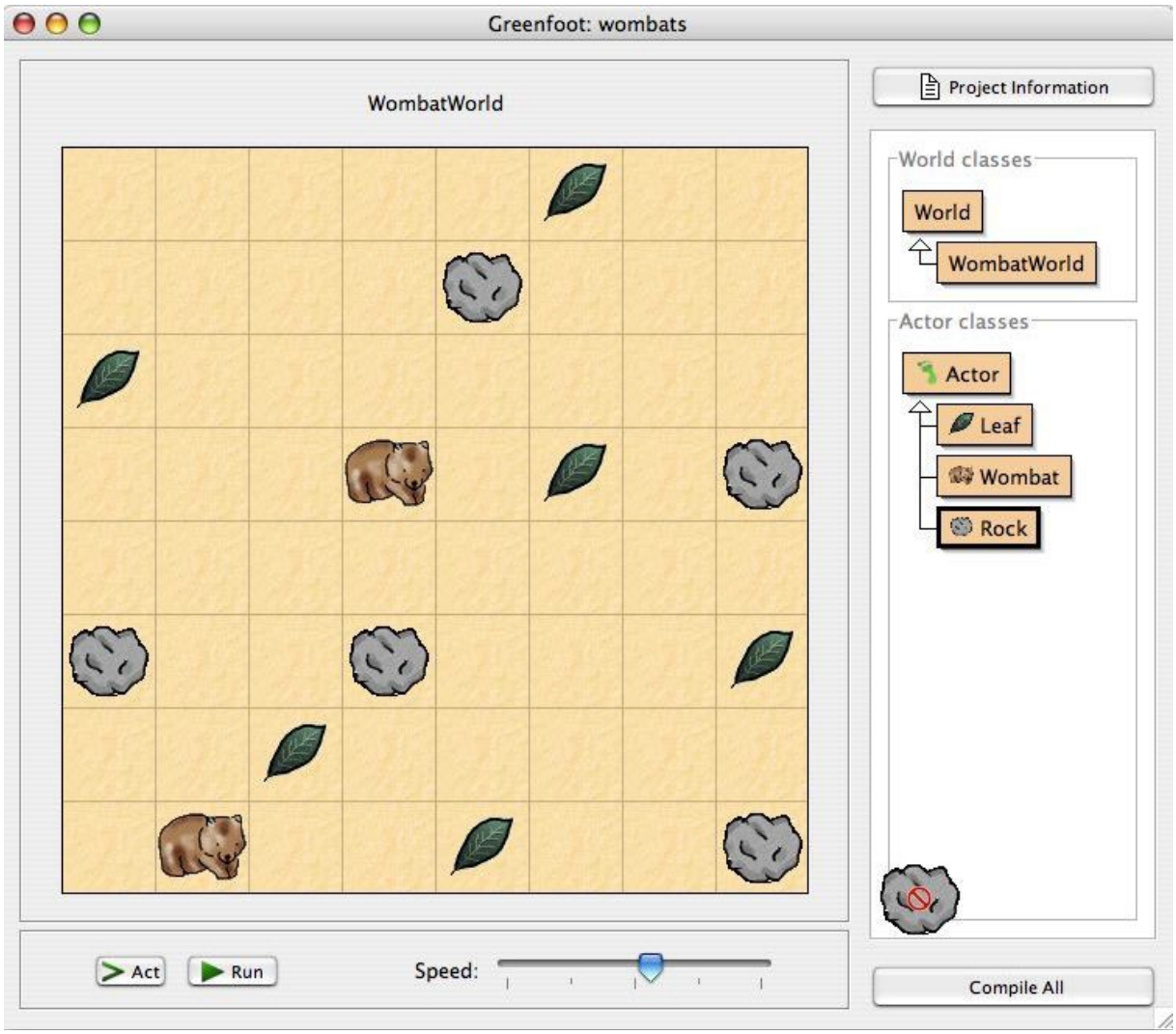

Figura 3: Interface do Greenfoot

Ainda, assim, o Greenfoot é muito utilizado para atrair o interesse em desenvolver aplicações gráficas, como jogos e simulações. E, com isso, auxilia, de maneira rápida e facilitada, o aprendizado de conceitos fundamentais da programação orientada a objetos. Como aspecto positivo, a sintaxe da linguagem Java possibilita a composição de aplicações maiores, ajudando os estudantes a se familiarizarem progressivamente com outros ambientes de programação. Em prol disso, utiliza o BlueJ[6] como ferramenta de edição (MASCHIO, 2013).

Em se tratando da dinâmica de interação, a estrutura inicial fornecida pelo

6 Site oficial: http://www.bluej.org.

Greenfoot consiste em duas classes principais: *world* e *actor*. A primeira representa o mundo em que os objetos serão distribuídos, e onde ocorrerá a execução. A segunda se refere a todos os outros objetos presentes no jogo, e que podem interagir com o mundo criado.

Assim, as atividades de interação e aprendizado no Greenfoot fundamentam-se na implementação de subclasses daquelas principais (*world* e *actor*). Nesta perspectiva, o ambiente motiva a exploração construtiva, buscando enfatizar abstrações e conceitos importantes em orientação a objetos, como a relação classe-objeto, métodos, parâmetros (MASCHIO, 2013).

3.3 SCRATCH

O Scratch foi uma iniciativa do MIT (*Massachusetts Institute of Technology*) destinada ao aprendizado extraclasse em comunidades desfavorecidas economicamente. O ambiente não se trata de um jogo propriamente dito, entretanto promove a ideia de programar através de atividades lúdicas. Ele objetiva a manipulação de mídias para a criação de histórias animadas, apresentações interativas e, nesses termos jogos simplificados.

Uma contribuição diferencial do Scratch reside na interação baseada na metáfora de blocos de instruções, conforme Figura 4, em que tais blocos são arrastados de uma paleta e encaixados para criar roteiros (procedimentos). Cada bloco é parametrizável através do preenchimento de lacunas que funcionam como canais restritos de entrada. Igualmente ao Alice, a ideia de arrastar-e-soltar componentes inibe inúmeros erros sintáticos (FINCHER et al. 2010).

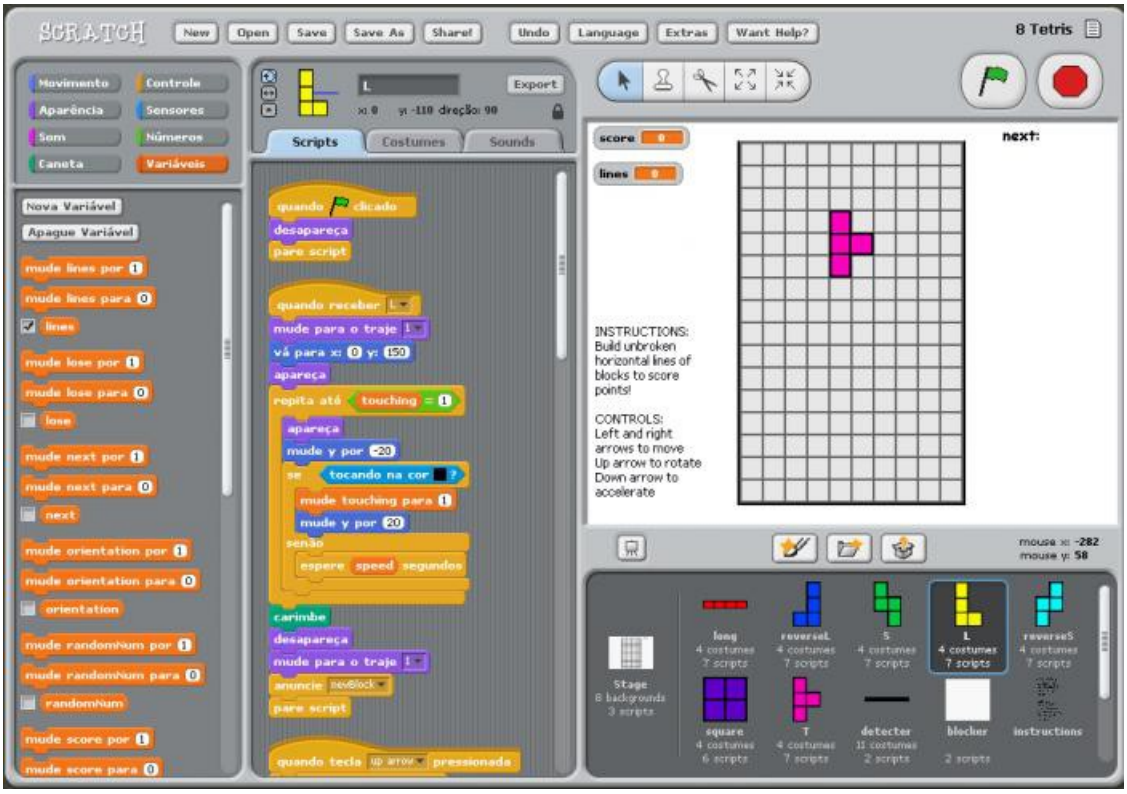

Figura 4: Interface do Scratch

3.4 CODESPELLS

O CodeSpells, consiste em um jogo propriamente dito, enquanto o Alice e o Greenfoot, diante do que se expôs, promovem a construção de jogos (THE CODESPELLS TEAM, 2013). Nesse sentido, o CodeSpells se utiliza de atividades exploratórias no ambiente para ensinar conceitos inicias de orientação a objetos, conforme Figura 5. Entretanto, se observa que as atividades propostas carecem de orientação no ambiente. Como consequência, tanto a interação quanto os objetivos didáticos se tornam confusos.

Figura 5: Interface do CodeSpells

Uma contribuição do CodeSpells, na perspectiva desta monografia, concentra-se em possibilitar que o usuário construa blocos de instruções, em Java, para avançar perante os desafios do jogo. O ambiente executa o compilador Java sobre as instruções programadas e oferece retorno, e possíveis erros, conforme a exatidão léxica e sintática do que se implementou. A corretude semântica das instruções é garantida pelo próprio cumprimento dos objetivos de jogo (desafios) propostos.

3.5 BELESMINHA

Assim como o CodeSpells, o Belesminha é um jogo por si só, nacional, e tem o objetivo de contribuir para a fixação de conceitos sobre recursividade (MASCHIO, 2013). O aprendiz precisa programar instruções a fim de comandar uma lesma, que tem por tarefa coletar folhas conforme são distribuídas no caminho, conforme Figura 6.

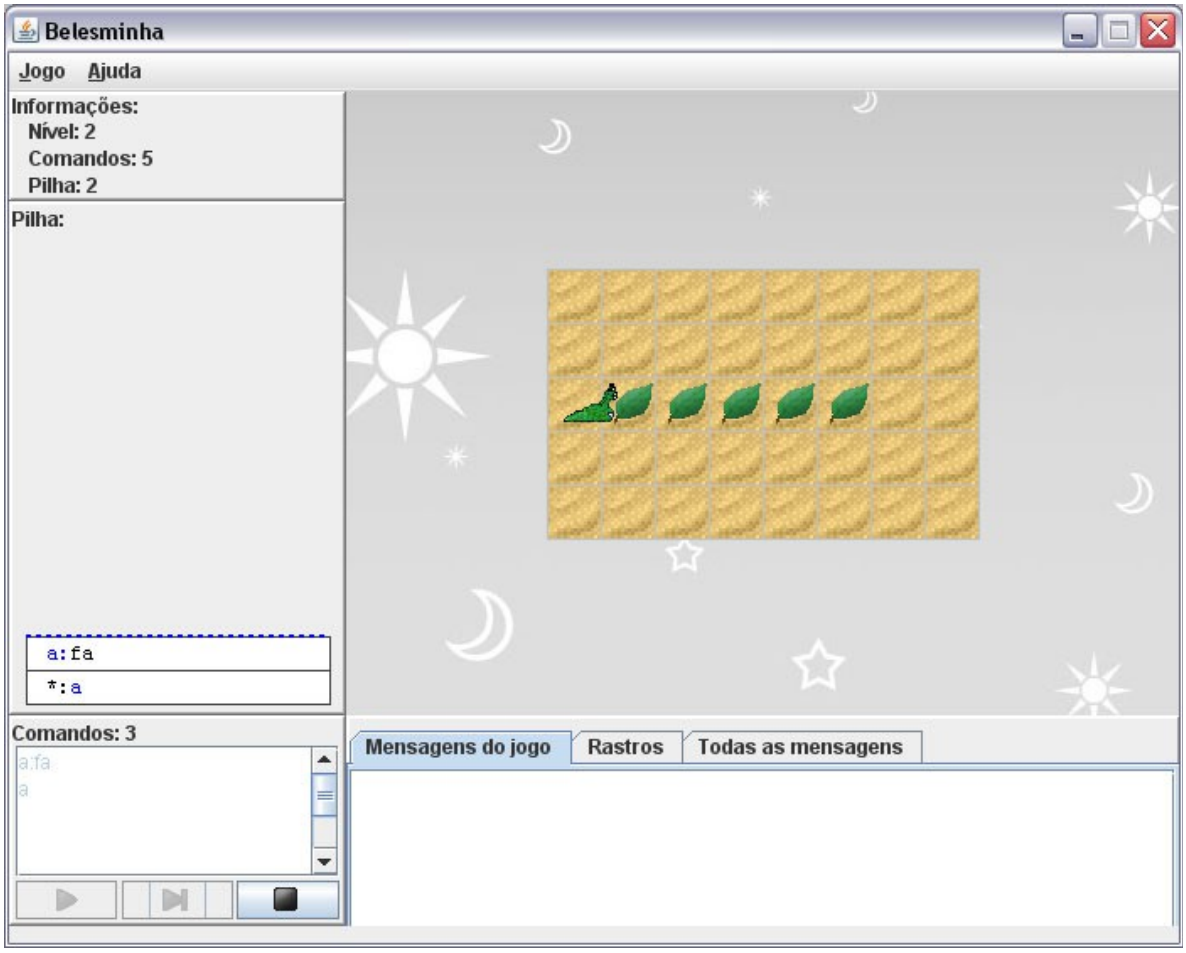

Figura 6: Interface do Belesminha

O jogo contém 10 níveis e cada um apresenta requisitos para a solução, como a quantidade máxima de instruções permitidas. A lesma pode executar três comandos básicos: virar à direita, à esquerda e avançar uma casa. Os blocos de instruções são compostos destes comandos e, à medida da progressão no jogo, admitem a modularização em funções. Para tanto, como auxílio, existem dicas sintáticas e semânticas no início de cada nível.

3.6 DISCUSSÃO

Diante do que se apresentou, os dois últimos ambientes resenhados mais se aproximam ao escopo do que se propõe para esta pesquisa. Tanto o Belesminha quanto o CodeSpells alicerçam os aspectos didáticos objetivados sobre atividades de programação para se avançar nos respectivos jogos.

O Belesminha restringe-se ao ensino do conceito de recursividade, em uma sintaxe própria definida pelo ambiente. Não aborda, portanto, qualquer conceito inicial de Orientação a Objetos. Entretanto, o jogo fornece amparo ao usuário no começo de cada nível, a fim de expressar o objetivo e os aspectos da solução requerida.

O CodeSpells, por sua vez, aproxima-se bastante do que se pretende com a pesquisa desenvolvida. Todavia, conforme se destacou, tem propósito didático pouco explícito e interação confusa, porque não transmite ao aprendiz a noção do que precisa ser feito no ambiente, embora se preste ao paradigma Orientado a Objetos e tenha interface atraente por conta dos recursos tridimensionais.

Constatado isso, há espaço para pesquisa e desenvolvimento de um jogo (*serious game*) que se destine ao ensino introdutório de Orientação a Objetos e que ofereça orientação tutorial mais presente. Os diferenciais apontados são a interface em português, mecanismos que guiem precisamente a interação e que especifiquem os objetivos didáticos relacionados com cada desafio proposto.

4 TECNOLOGIAS EMPREGADAS

4.1 A LINGUAGEM JAVA

A Linguagem Java (CASTELA, 2010) foi desenvolvida em 1990, pela organização Sun Microsystems, baseada nas linguagens C e C++. Além de ser uma linguagem de programação, Java também consiste em uma plataforma específica para execução dos aplicativos (ORACLE, 2013). Categorizada como de alto-nível, e Orientada a Objetos, é a primeira linguagem multiplataforma executada em praticamente todos os computadores (LOPES et al. 2012). Pela característica multiplataforma, consegue-se abstrair a preocupação com o *hardware* durante o desenvolvimento, construindo-se apenas componentes de *software* (ORACLE, 2013).

O nome foi escolhido pela equipe responsável pelo projeto, uma vez que eles queriam homenagear a bebida tão apreciada pelos profissionais da Computação, o café. Optaram por Java em referência a uma ilha da Indonésia, grande produtora de diversos tipos de café.

A linguagem Java tem passado por constantes alterações que visam ao seu aprimoramento, principalmente quanto ao desempenho. Tais alterações, associadas ao paradigma Orientado a Objetos, têm garantido a popularidade da linguagem, sendo, atualmente, a segunda linguagem de maior utilização no mundo (BV, 2013).

A plataforma Java é composta por três pilares (ORACLE, 2013), a saber: a máquina virtual Java (JVM), a API (*Java Application Programming Interface*) e a própria linguagem. A JVM é o principal componente da plataforma, consistindo em uma base responsável por eliminar os problemas de portabilidade, executando instruções genéricas de máquina, independentemente do hardware ou software. A API é um conjunto de componentes de software, que figura como uma vasta biblioteca de funcionalidades para os desenvolvedores.

É oportuno mencionar o processo de compilação e interpretação dos programas em Java. Primeiramente, os arquivos são escritos seguindo a sintaxe linguagem, e possuem a extensão `.java`. Tais arquivos são compilados pelo comando `javac`, que cria um arquivo de extensão `.class` para cada arquivo

`.java` (considerando a inexistência de erros sintáticos). Um arquivo `.class` não possui código de máquina nativo, mas um código intermediário que a JVM interpreta. Então, os arquivos `.class` são executados sobre uma JVM que traduz as instruções para o código de máquina da arquitetura do computador utilizado. Por isso é necessária a instalação da JVM específica para cada Sistema Operacional. Esse processo pode ser visualizado na Figura 7, abaixo:

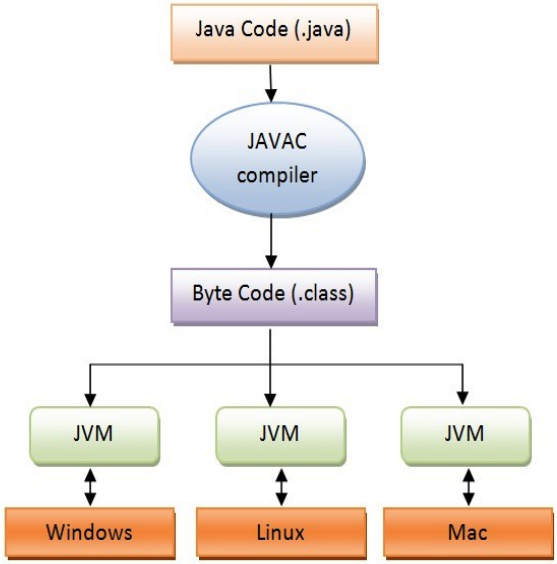

Figura 7: Processo de Compilação Java (ROMANATO, 2013)

No contexto específico da pesquisa desenvolvida, foi escolhida a linguagem Java primeiramente pela necessidade de remeter o aprendizado de Orientação a Objetos a uma situação concreta de Programação de Computadores. Além disso, a linguagem tem se mostrado adequada ao ambiente acadêmico, sendo utilizada em várias iniciativas destinadas ao ensino do paradigma citado, como, por exemplo, o Projeto Jedi[1], ambientes de ensino como o BlueJ (BARNES; KOLLING, 2009) e os próprios CodeSpells e Greenfoot, conforme trabalhos correlatos da pesquisa. Também se destaca a alta difusão de códigos escritos na linguagem e vasta quantidade materiais didáticos, corroborados pela segunda posição entre as linguagens mais utilizadas no mundo.

1 Disponível em: http://www.dfjug.org/jedi/

4.2 UNITY 3D

A Unity 3D é definida pelo próprio fabricante (*Unity Technologies*) como um ecossistema para o desenvolvimento de jogos (GOLDSTONE, 2011). A ferramenta é composta tanto por um motor de renderização 3D (*engine*) quanto por uma IDE[2] para a autoria de jogos. Atualmente, trata-se de uma das ferramentas mais utilizadas tanto para o desenvolvimento de jogos quanto para a criação de experiências interativas para a web, desktop, dispositivos móveis e consoles. O uso se estende desde o *hobby* até o emprego em grandes estúdios, passando por projetos educacionais e desenvolvedores independentes.

A Unity funciona de maneira semelhante às ferramentas que já existiam até então (Blender Game Engine[3], Unreal Development Kit[4] e CryEngine[5], por exemplo), quanto ao processo de autoria por meio de interface gráfica. Contudo, o mérito da Unity reside em ser uma opção de baixo-custo, leve curva de aprendizado e alta abstração de detalhes técnicos complexos, como a física de jogos. Assim, mesmo sendo recente, a ferramenta tem ganhado espaço, por impactar em dois fatores determinantes no desenvolvimento de jogos: o tempo e o custo de criação.

Especificamente sobre o custo, há duas versões principais: a Unity Pro, com custo de US$ 1.500,00, e a versão gratuita, simplesmente chamada Unity. A última, pode ser usada tanto para fins educacionais, quanto comerciais, e possui limitações de funcionalidade, exibição de marca d'água, e sobre o rendimento obtido com a comercialização do jogo desenvolvido. A Unity Pro pode ser testada por um período de 30 dias. As informações foram obtidas no site do fabricante e estão sujeitas a alterações[6].

Além do exposto, abaixo são listadas outras vantagens que influenciaram a escolha da ferramenta para o projeto:

- Fluxo de trabalho intuitivo;

2 Do inglês, *Integrated Development Environment*, ou Ambiente de Desenvolvimento Integrado em português.
3 Disponível em: http://www.blender.org/
4 http://www.unrealengine.com/udk/
5 http://www.crytek.com/cryengine
6 http://portuguese.unity3d.com/

- Editor integrado de fácil utilização, associado à interface com recurso de arrastar-e-soltar;
- Escrita de *scripts* nas linguagens JavaScript, C# ou Boo[7], embora com particularidades, todas executadas na plataforma Mono (open source e compatível com a plataforma .NET);
- Compatibilidade de importação de recursos nos mais diversos padrões do mercado, como por exemplo: Blender[8], 3ds Max[9], Maya[10] e ZBrush[11].
- Alta qualidade final garantida pelo motor de jogo e pelas possibilidades de efeitos visuais e sonoros, como luz, sombra, explosões, fogos de artifício, entre outros;
- Poderosa tecnologia de animação de personagens, dispondo de movimentação natural e fluída;
- Motor de física eficiente, NVIDIA PhysX, que permite detalhes como a movimentação de roupas e cabelos no vento, vidros estilhaçando e pneus derrapando, por exemplo;
- Amparo na construção das máquinas de estado necessárias para o jogo;
- Alta performance devida à renderização somente do conteúdo capturado pela câmera;
- Multiplataforma, ou seja, um mesmo jogo pode ser exportado para diversas plataformas com pouca, ou nenhuma alteração. As plataformas suportadas são:
    - Web: Internet Explorer, Firefox, Safari, Opera, Google Chrome e Camino;
    - Desktop: Windows, Mac e Linux;
    - Dispositivos móveis: iOS, Android, Windows Phone e BlackBerry;
    - Consoles: PS3, Xbox 360 e Wii U.

7 Linguagem de programação multiparadigma que suporta orientação a objetos, programação imperativista e programação funcionalista, de código aberto e sintaticamente inspirada em Python. Foi criada pelo desenvolvedor brasileiro Rodrigo Barreto de Oliveira. Conforme página oficial: http://boo.codehaus.org
8 Site oficial: http://www.blender.org/
9 Site oficial: http://www.autodesk.com.br/products/autodesk-3ds-max/overview
10 Site oficial: http://www.autodesk.com.br/products/autodesk-maya/overview
11 Site oficial: http://pixologic.com/zbrush/

- Colaboração da comunidade[12], citada pelo próprio fabricante, como um dos grandes diferenciais da ferramenta. A versão gratuita proporciona que desenvolvedores independentes contribuam, de forma fiel, com a movimentação de fóruns e a criação de exemplos;
- Comunidade brasileira participativa[13] que disponibiliza e discute constantemente artigos, tutoriais e, inclusive, vídeos;
- Repositório de recursos que podem ser comprados ou baixados gratuitamente, permitindo que o tempo de desenvolvimento seja acelerado.

Por fim, percebe-se que a Unity está se consolidando como um padrão para o desenvolvimento de jogos, principalmente no Brasil. Investir nesse conhecimento pode trazer um diferencial de formação considerável frente a matriz curricular do curso de Tecnologia em Sistemas para Internet e ao mercado de trabalho.

12 Disponível em: http://udn.unity3d.com e http://wiki.unity3d.com.
13 http://www.unity3dbrasil.com.

5 FORMALISMO DA SOLUÇÃO PROPOSTA

Compete, ao presente capítulo, a formalização da solução proposta na resolução da problemática desta pesquisa. Foram considerados aspectos positivos e negativos dos trabalhos correlatos. Houve, adicionalmente, embasamento nas fundamentações de Ambiente Interativos de Aprendizagem (Seção 5.1), Ambiente de Descoberta Guiada (Seção 5.2) e Gamificação (Seção 5.3).

Com isso, retomou-se o enfoque da solução proposta (Seção 5.4), o conteúdo de Orientação a Objetos abordado (Seção 5.6) e, por fim, o diálogo de interação modelado para o uso da ferramenta implementada (Seção 6.6).

5.1 ASPECTOS DE AMBIENTES INTERATIVOS DE APRENDIZAGEM

Conforme Maschio (2007), um Ambiente Interativo de Aprendizagem (AIA)[1] consiste em uma ferramenta de caráter passivo, semanticamente rica, que parte de atividades de exploração, investigação e descoberta para que o aprendiz construa o próprio conhecimento. Não interfere, ou interage, ativamente ou pró-ativamente, com o aprendiz.

Nesse sentido, os micromundos (ou microcosmos), são casos bastante característicos de AIAs. Tratam-se de ambientes que utilizam da riqueza semântica como instrumento de abstração para representar situações reais. Um micromundo, portanto, disponibiliza instrumentos que atuam como um conjunto metafórico das situações representadas (PAPADOPOULOS; TEGOS, 2012).

Assim, cabe ao micromundo oferecer uma escala variada de recursos de interação ao aprendiz. Com isso, constitui-se como uma ferramenta de autoestudo que assiste à construção de conhecimento por parte do aprendiz. Diferindo-se de compiladores convencionais, um micromundo destinado a aprendizagem de Programação de Computadores, destaca representações externas de mais alto nível, com caráter exclusivamente educacional, bem como, adicionalmente, a possível correspondência entre elas.

1 Do inglês *Interactive Learning Environment* (*ILE*).

A atual proposta absorve e usufrui a concepção de AIAs, mais especificamente de micromundos, no sentido de prover uma ferramenta de estudo semanticamente rica para a condução de experimentos. De acordo com as seções subsequentes, a ideia é estendida diante dos conceitos de Ambientes de Descoberta Guiada e de Gamificação.

5.2 ASPECTOS DE AMBIENTES DE DESCOBERTA GUIADA

Bastante próxima dos Ambientes Interativos de Aprendizagem, existe a caracterização dos Ambientes de Descoberta Guiada (ADGs)[2]. Tratam-se de ferramentas educacionais cujas oportunidades de interação são conduzidas pelo próprio ambiente (MASCHIO, 2007). Enquanto nos AIAs, o auxílio tutorial humano é exigido, nos ADGs, é tido como algo suplementar ou desnecessário.

Por conta disso, os ADGs são tidos como a demarcação entre os AIAs e os ditos Sistemas Tutores Inteligentes (STIs). Estes, são projetados para incorporar técnicas de IA (Inteligência Artificial) de modo a prover tutores que saibam como ensinar, o que ensinar e a quem ensinar (WENGER, 1987).

Entretanto, o amparo fornecido por um ADG pode variar quanto à pró-atividade e inteligência. É admissível que tal ambiente guie a interação por meio de mensagens estáticas e predefinidas, como também que ocorra o suporte ao aprendiz com textos personalizados e gerados dinamicamente.

O conceito de ADG, mesmo que na forma mais básica, agrega bastante a um AIA. Como exemplo, tem-se o próprio CodeSpells, trabalho correlato citado, com carência visível de mensagens que guiem minimamente a interação. Desprovido disso, tanto o ambiente quanto o esforço depositado na respectiva construção parecem terem sido subutilizados.

A presente proposta é uma tentativa de conciliar ambas as ideias, ou seja, permitir a livre exploração conduzida por um amparo didático mínimo.

---

2 Do inglês, *Guided-Discovery Environments* (*GDEs*).

5.3 GAMIFICAÇÃO

Segundo NAVARRO (2013), compreende-se gamificação como o ato de aplicar elementos, mecanismos, dinâmicas e técnicas de jogos em contextos alheios aos jogos propriamente ditos. Incorpora, portanto, a ideia de conjugar tais recursos com situações reais do cotidiano profissional, escolar e social do indivíduo. O emprego da estratégia, contudo, independe do uso de computador.

O termo foi criado pelo programador britânico Nick Pelling, sendo mais corretamente traduzido para ludificação, em português, e se aplica a processos e aplicações visando incentivar pessoas a adotá-los, ou ainda influenciar a maneira como são usados. Destacam-se, como objetivos da gamificação (NAVARRO, 2013):

1. Tornar a tecnologia mais atraente;
2. Estimular os usuários a se engajarem com comportamentos desejados;
3. Incentivar um caminho para a autonomia;
4. Ajudar a resolver problemas específicos;
5. Aproveitar a pré-disposição humana de se engajar com jogos.

A abordagem tem se mostrado eficiente para motivar a realização de tarefas que as pessoas, normalmente, não consideram muito prazerosas. Como exemplo, pode-se mencionar a adoção, por diversas metrópoles, de intervenções urbanas como escadas cujos degraus tocam notas musicais ou cestas de lixo com aros de basquete.

Outra situação pontual (PONTES, ROSA, 2013) foi o uso pela Volkswagen, em Estocolmo, Suécia, para reduzir a velocidade dos automóveis. Nela, os radares detectavam também os veículos que cumpriam o limite de velocidade, fazendo com que os donos concorressem a uma loteria composta pelas multas de quem o excedesse. Como resultado, a velocidade média dos automóveis foi reduzida em 22%.

A Internet também tem se beneficiado da gamificação, pois se utiliza de um rico sistema de recompensas e reputação, formado por curtidas, *check-ins*, *rankings*, *scores*, contagem de amigos e seguidores, entre outros. No mesmo sentido, diversas estratégias de marketing se alicerçam no conceito de gamificação,

contando com propagandas virais, cupons de desconto, cartões de fidelidade, indicações de amigos, bonificação em milhas aéreas, para mencionar alguns exemplos. Em adição, o ambiente corporativo tem se valido de recursos semelhantes, em programas de treinamento e incentivo à produção. Entretanto, no mundo dos negócios, as ressalvas e as críticas têm se mostrado mais severas, taxando a abordagem como exploratória (CORCORAN, 2010).

Do mesmo modo, no contexto educacional, os *serious games* podem ser vistos pela perspectiva dessa abordagem. Conforme o referencial teórico apresentado pela presente pesquisa, tal categoria de jogos promove justamente as características citadas, associando-as ao ensino de conteúdos específicos. Portanto, diante do que se propõe, intenciona-se exatamente conduzir o aprendizado de programação a um território lúdico, empolgante e motivador.

5.4 ENFOQUE DA SOLUÇÃO PROPOSTA

A solução proposta por esta pesquisa se concentra na intersecção entre Ambientes Interativos de Aprendizagem, Ambientes de Descoberta Guiada e Gamificação, sendo mais especificamente:

1. Ambientes Interativos de Aprendizagem: por consistir em uma ferramenta semanticamente rica que oferece atividades de exploração para autoestudo, como abstração de situações reais;
2. Ambientes de Descoberta Guiada: pela extensão do AIA devido ao suprimento de mensagens tutoriais que conduzam o diálogo com a ferramenta;
3. Gamificação: por associar ambos, AIA e ADG, na concepção de um *serious game* que trate da problemática apresentada de forma lúdica.

5.5 CONTEÚDO DE ORIENTAÇÃO A OBJETOS ABORDADOS

Para testar e validar o protótipo como instrumento de ensino, foram escolhidos os seguintes conteúdos de introdução ao paradigma Orientado a Objetos em Java:

1. Conceitos iniciais;

2. Classes e objetos;
3. Atributos e métodos;
4. Estrutura condicional *if-else*; e
5. Estruturas de repetição.

Na continuidade deste capítulo, serão detalhados três cenários propostos e os conteúdos relacionados com cada um deles. Convém frisar que não coube à pesquisa produzir textos didáticos que abordassem tais assuntos. Com isso, os textos tutoriais da descoberta guiada provieram de Barnes e Kolling (2009).

5.6 DIÁLOGO DE INTERAÇÃO PROPOSTO

O diálogo de interação proposto para o *serious game* inicia com uma tela, mostrada na Figura 8, onde o aprendiz seleciona o tópico a ser tutorado naquela sessão de ensino, dentre aqueles disponíveis. Em adição, são mostrados o título do Trabalho de Conclusão de Curso e a respectiva autoria. Realizada a escolha do tópico, a interação é conduzida para uma segunda tela, destinada à exibição dos textos tutoriais.

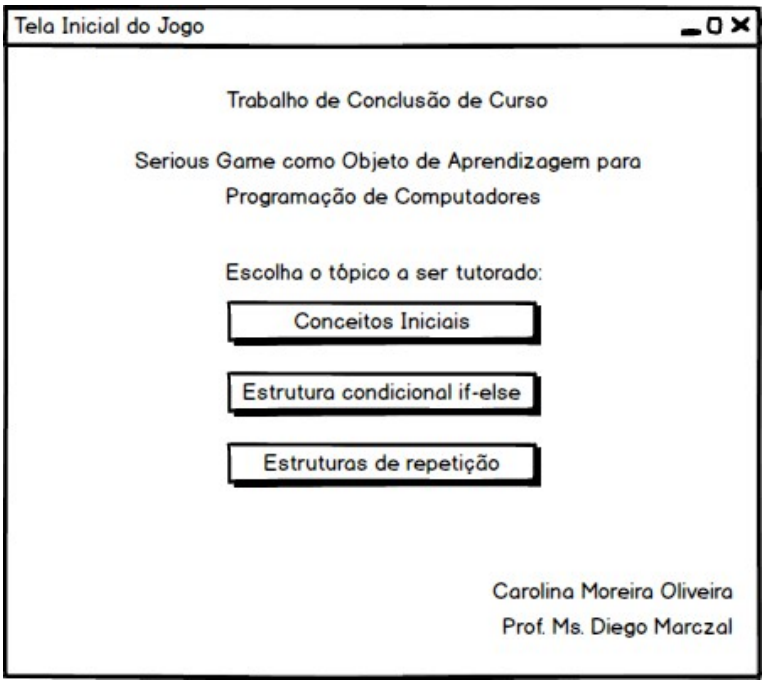

Figura 8: Tela inicial

A Figura 9 apresenta essa segunda tela, assumindo que a escolha tenha sido o tópico intitulado Conceitos Iniciais. Tal conteúdo, conforme a mesma figura, é subdividido nas seções: introdução, classes, objetos, atributos e métodos. Por fim, tem-se a proposta de um desafio a ser programado na interação com o jogo. A tela ainda disponibiliza um botão para retroceder àquela inicial e outro para prosseguir com o desafio (ou jogo).

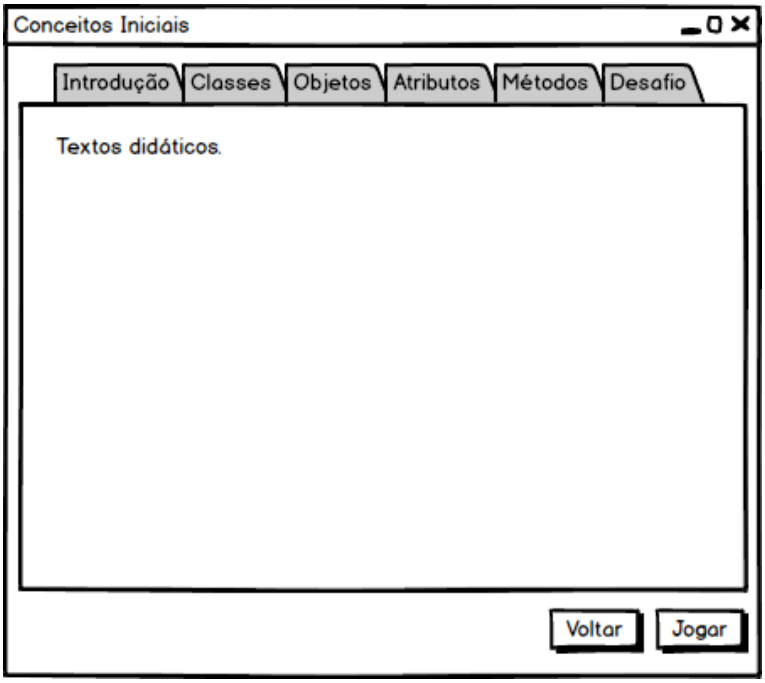

Figura 9: Tela de Conceitos Iniciais

Considerando o avanço no desafio, a interação é conduzida para a tela principal do *serious game*, exibida na Figura 10. Tal tela é dividida, verticalmente, em duas partes, uma destinada ao exercício de implementação em Java e a outra ao cenário do jogo.

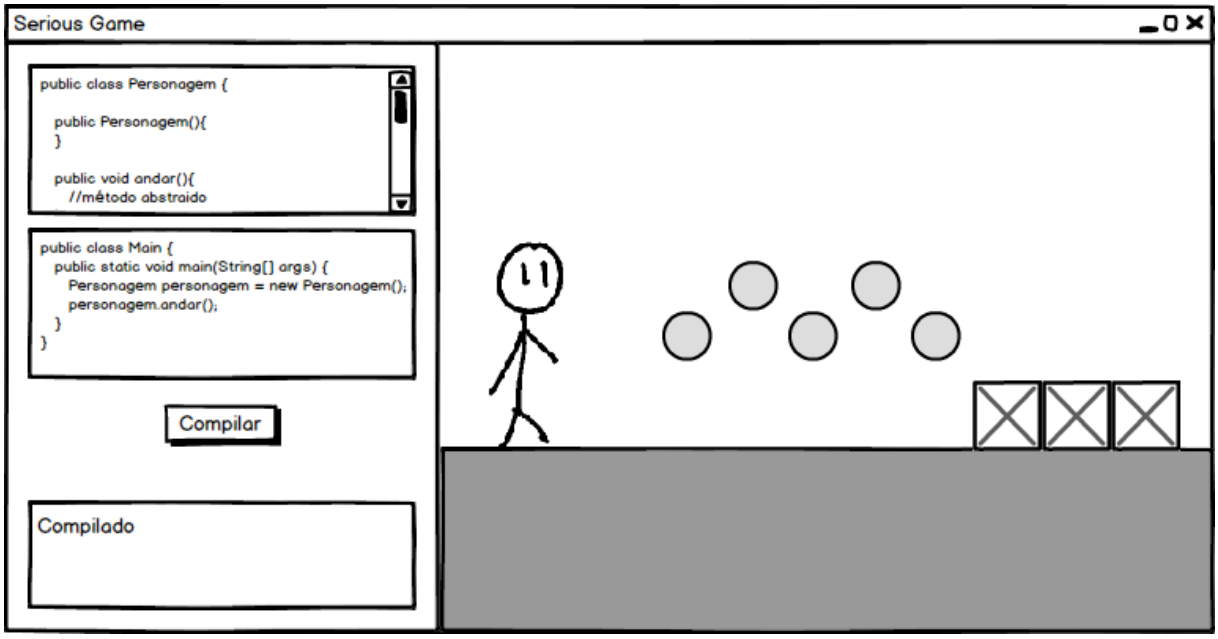

Figura 10: Tela do Primeiro Cenário

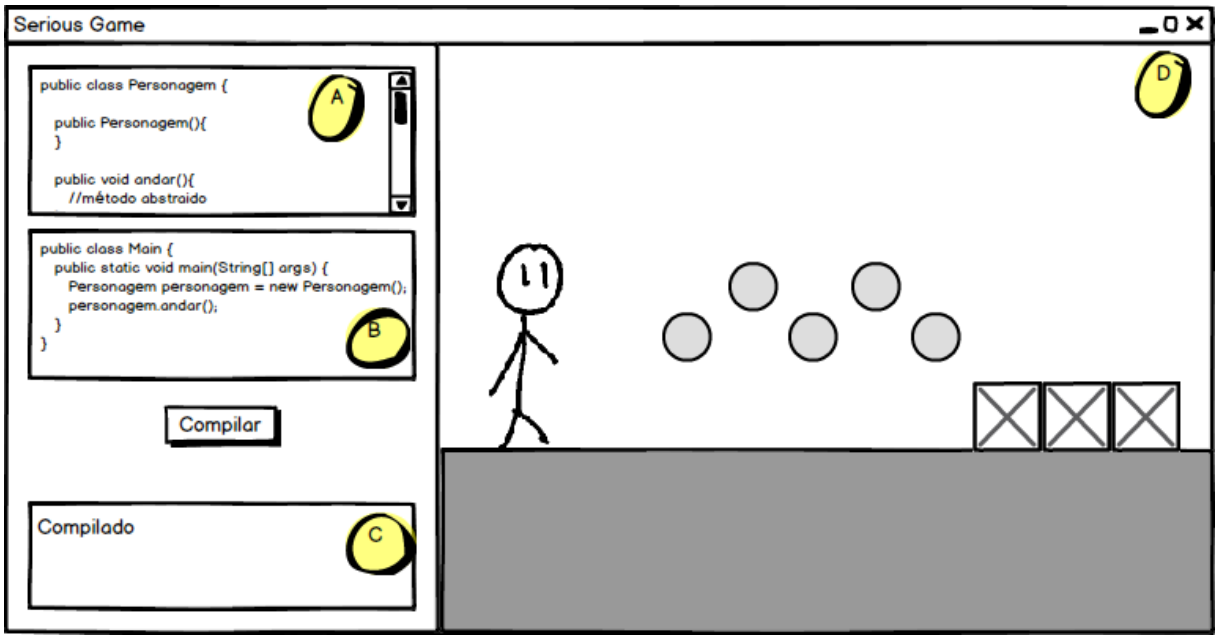

Figura 11: Tela do Primeiro Cenário com identificação

A Figura 11, ilustra a mesma tela, sinalizando as regiões envolvidas na dinâmica de utilização, que são definidas e caracterizadas abaixo:

a) Área de contextualização: refere-se ao espaço em que as classes e outros trechos de código já prontos são oferecidos ao aprendiz a fim de

contextualizar a implementação do desafio proposto;

b) Área de codificação: abrange a região onde o aprendiz digitará a solução que ele alcançou para o citado desafio. Logo abaixo do espaço destinado à entrada, tem-se o botão que se responsabiliza por compilar o trecho de código envolvido;

c) Área do console: de compilação corresponde ao espaço para feedback de compilação. Tanto mensagens de erro quanto de sucesso da compilação são exibidas nesta área;

d) Cenário do jogo: comporta a animação do jogo propriamente dito, cujo exercício de implementação determinará o seu funcionamento.

Um exemplo de interação com o desafio consiste no aprendiz ler e compreender os códigos da área de contextualização. Depois disso, ele idealiza uma solução para o desafio apresentado. Tal exercício é implementado na área de codificação. Feito isso, o aprendiz aciona a compilação do trecho de código correspondente. Em caso de erro, as mensagens próprias são exibidas na área do console e o aprendiz tem a oportunidade de alterar o código correspondente à solução. Do contrário, uma mensagem de sucesso é apresentada na mesma área e o aprendiz pode visualizar o resultado da sua solução no cenário do jogo.

Destaca-se, por fim, que, nos cenários propostos, cabe ao aprendiz apenas implementar a habilitação (ou acesso) às funcionalidades predefinidas no jogo. Ou seja, as teclas de movimentação se encontram, inicialmente, desabilitadas. Conforme os desafios, a implementação altera esse comportamento em correspondência com o conteúdo didático de cada desafio.

6 PROTÓTIPO DO SERIOUS GAME

Neste capítulo se discorre sobre a modelagem e detalhes de implementação do protótipo de *serious game* desenvolvido. Primeiramente são apresentados os três cenários que se modelou para cumprir com o conteúdo proposto. Em seguida, são trazidos aspectos de implementação da pesquisa.

6.1 CENÁRIOS MODELADOS

6.1.1 Primeiro Cenário: Conceitos Iniciais

O conteúdo abordado pelo primeiro cenário corresponde a: (1) classes e objetos; e (2) atributos e métodos. Para tanto, se propõe o desafio de instanciar o personagem (um robô), e fazê-lo andar pelo cenário, conforme mostrado na Figura 22. Cabe ao aprendiz habilitar essa funcionalidade que é acessada pelas teclas de movimentação (seta esquerda e direita) durante o jogo.

Figura 12: Primeiro Cenário

A classe que implementa o robô (`Personagem`) é mostrada ao aprendiz em caráter de contextualização. Abstraem-se detalhes de codificação na referida classe, sendo enfocada apenas a assinatura (ou cabeçalho) do método andar, justamente aquele que proporciona a movimentação do robô em resposta às teclas.

```
public class Personagem {

	public Personagem(){
	}

	public void andar(){
		//_comportamento abstraido
	}
}
```

Figura 13: Classe Personagem (contextualização)

Feita a contextualização, vide Figura 13, o aprendiz exercita o conceito de classe e de método. Com isso, ele tem conhecimento que a classe Personagem possui um método denominado `andar`.

Na área de codificação, o aprendiz se depara com um código incompleto (Figura 14) que corresponde ao cenário, ou seja, `Cenario`, a classe principal ou de execução do jogo. Isso é evidente porque a classe já dispõe do método `main`.

```
public class Cenario {
	public static void main(String[] args) {


	}
}
```

Figura 14: Classe Cenário incompleta

Conforme o desafio proposto, cabe ao aprendiz instanciar um objeto do tipo Personagem, independentemente do nome da variável utilizada. Depois disso, é necessário realizar a chamada do método `andar`, cuja assinatura já se conhece.

Implementadas essas duas instruções, isto é, a instanciação do personagem e o envio da mensagem para andar, a solução foi alcançada, conforme a Figura 15. Resta apenas ao aprendiz acionar a compilação e verificar possíveis erros sintáticos. Do contrário, uma mensagem de sucesso é exibida e o aprendiz pode interagir com o jogo e observar possíveis erros semânticos no resultado final.

```
public class Main {
      public static void main(String[] args) {
            Personagem personagem = new Personagem();
            personagem.andar();
      }
}
```

Figura 15: Classe Cenário completa

Destaca-se que este primeiro cenário, diante da prototipação, foi implementado em completude e se encontra em funcionamento.

6.1.2 Segundo Cenário: Estrutura condicional *if-else*

O segundo cenário aborda a estrutura condicional *if-else*. Consegue-se isso através do desafio de fazer com que o personagem reconheça o tipo de obstáculo em uma colisão. Conforme o contexto desse desafio, há dois tipos de obstáculos: moedas e caixas.

Os obstáculos são modelados por uma classe própria, de mesmo nome, com um método que retorna um inteiro correspondente ao tipo de obstáculo. Tal inteiro pode ser comparado com as constantes desta própria classe, *Obstaculo.MOEDA* e *Obstaculo.CAIXA*.

A classe que implementa os obstáculos é exibida para fins de contextualização. Abstraídos os detalhes de codificação, são destacados apenas o método `getTipo`, bem como a variável e as constantes recém mencionadas, de acordo com a Figura 16.

```
public class Objeto() {
      public static final int MOEDA = 0;
      public static final int CAIXA = 1;

      private int tipo;

      public Objeto(int tipo) {
         this.tipo = tipo;
         // outros detalhes de codificação foram abstraídos
      }

      public int getTipo() {
         return this.tipo;
      }
}
```

Figura 16: Classe Objeto

A contextualização retoma o conceito de classe e de método, como também exemplifica e exercita o conteúdo de constantes. Além disso, estrutura-se o pensamento do aprendiz em entender que a classe `Objeto` possui um atributo chamado `tipo`, bem como as constantes `MOEDA` e `CAIXA`, todos inteiros. Existe também um método denominado `getTipo` que retorna o inteiro que identifica o tipo do obstáculo instanciado.

o desafio proposto, informa-se que a classe `Cenario` aciona automaticamente um método do objeto de tipo `Personagem` toda vez que ocorre um evento de colisão. Na chamada do método é repassado um objeto do tipo `Obstaculo`, conforme a seguinte assinatura:

```
public void aoColidir(Obstaculo obstaculo);
```

Na área de codificação, o aprendiz se depara com um código incompleto (Figura 17) que corresponde à classe `Personagem`. São métodos abstraídos: `andar`, `pegarMoeda` e `pular`.

```
public class Personagem {

	public Personagem(){
		// comportamento abstraído
	}

	public void aoColidir(Obstaculo obstaculo){
		// sua solução
	}

	public void andar(){
		// comportamento abstraido
	}

	public void pegarMoeda(){
		// comportamento abstraido
	}

	public void pular(){
		// comportamento abstraido
	}
}
```

Figura 17: Classe Personagem do Cenário 2 (contextualização)

Cabe, então, ao aprendiz implementar o código interno do método `aoColidir` para que, conforme o tipo de obstáculo repassado em um dado momento, habilite a funcionalidade de pegar moeda ou pular caixa. Tudo isso é descrito em texto tutorial relacionado ao desafio.

A solução é alcançada com o emprego da estrutura condicional *if-else* para

guiar o acesso exclusivo ao método `pegarMoeda` ou `pular`. A Figura 18 mostra apenas esse trecho de código.

```
public void aoColidir(Obstaculo obstaculo){
   if (obstaculo.tipo() == Obstaculo.MOEDA)
      this.pegarMoeda();
   else
      this.pular();
   }
}
```

Figura 18: Solução esperada para o método aoColidir

Assim como especificado no cenário anterior, basta ao aprendiz acionar a compilação e, na ausência de erros sintáticos, interagir com o jogo e analisar o resultado que obteve.

6.1.3 Terceiro Cenário: Estruturas de repetição

Cabe ao terceiro cenário introduzir o conteúdo de estruturas de repetição, abordando a instrução *while*. O desafio tem proposta bem semelhante àquele do cenário anterior e consiste em atacar um oponente enquanto este ainda estiver vivo.

Os oponentes são modelados por uma classe específica, de mesmo nome, com um método que retorna um *boolean* indicando se ele está vivo. Dessa forma, o retorno será falso caso esse oponente tenha sido destruído.

Em caráter de contextualização, a classe que implementa os oponentes é exibida. Os detalhes de codificação são abstraídos, tendo enfoque apenas o método `vivo`, conforme mostra a Figura 19.

```
public class Oponente() {
      public Oponente() {
         // comportamento abstraído
      }

      public boolean estaVivo() {
         // comportamento abstraído
      }
}
```

Figura 19: Classe Oponente

Além de reforçar novamente os conceitos de classe e de método, a contextualização faz o aprendiz ter conhecimento de que a classe `Oponente` possui um método denominado `vivo`.

Em analogia ao desafio anterior, também se informa que a classe `Cenário` aciona automaticamente um método do objeto de tipo `Personagem` toda vez que se percebe um oponente (trata-se de um evento, acionado como o método `aoColidir`). No envio de mensagem é repassado um objeto do tipo `Oponente`, conforme a seguinte assinatura:

```
public void aoPerceberOponente(Oponente oponente);
```

Na área de codificação, o aprendiz se depara com o seguinte código incompleto (Figura 20), que corresponde à classe `Personagem`.

```
public class Personagem {

	public Personagem(){
		// comportamento abstraído
	}

	public void aoPerceberOponente(Oponente oponente){
		// comportamento solução
	}

	public void andar(){
		// comportamento abstraido
	}

	public void pegarMoeda(){
		// comportamento abstraido
	}

	public void pular(){
		// comportamento abstraido
	}

	public void atirar(){
		// comportamento abstraido
	}
}
```

Figura 20: Classe Personagem do Cenário 3

Desta vez, o aprendiz deve implementar o código interno do método `aoPerceberOponente` para que habilite a funcionalidade de `atirar` enquanto o oponente não for destruído, ou seja, estiver `vivo`. Igualmente, a descrição é informada ao aprendiz em texto tutorial do desafio.

A solução é obtida com o uso da estrutura condicional *while* para repetir a chamada ao método `atirar`. O trecho de código é exibido pela Figura 21.

```
public void aoPerceberOponente(Oponente oponente){
   while (oponente.vivo()) {
      this.atirar();
   }
}
```

Figura 21: Solução esperada para o método aoPerceberOponente

Resolvido o desafio, o processo de compilação e execução é feito da mesma maneira que nos outros dois cenários.

6.2 INTEGRAÇÃO DO JAVA COM UNITY 3D

A Unity 3D oferece suporte nativo às linguagens Javascript, C# e Boo, mas não ao Java. Com isso, houve incessante esforço para que a integração da linguagem com a ferramenta ocorresse. A sequência de tentativas sem sucesso serão documentadas na Seção 7.3.

A abordagem efetivada na solução foi auxiliada pelo professor Andres Jessé Porfirio. A alternativa, diante da falta de suporte nativo da ferramenta ao Java, consiste na capacidade do Unity executar comandos no terminal, por meio instruções próprias em C#, conforme Figura 22.

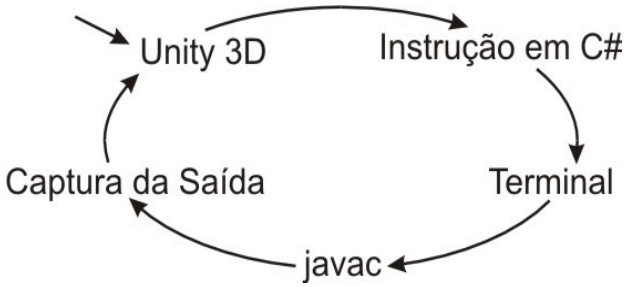

Figura 22: Interação Java com Unity

Dessa forma, pode-se ativar o compilador Java (`javac`) e, na sequência, a execução do programa resultante. Além disso, como característica da execução em terminal, é possível capturar a saída do que se executa.

A abordagem reside justamente nessa capacidade, de capturar a saída do programa resultante. Assim, as classes Java correspondentes ao desafio, contendo os trechos de código do aprendiz, são compiladas e executadas, então a respectiva

saída é utilizada para guiar o funcionamento do jogo no Unity.

A incorporação da saída no Unity é feita por um procedimento básico de análise (*parsing*), procurando por cadeias de caracteres (*strings*) específicas no retorno. A presença, ausência e ordem dessas cadeias de caracteres são utilizadas no acionamento de métodos determinados dentro do jogo no Unity.

Entretanto, outra dificuldade residiu em ocultar detalhes de implementação, dessa saída em tela, da perspectiva tida pelo aprendiz na utilização da ferramenta. Conforme previsto, o aprendiz idealiza e implementa uma solução baseada em conceitos de Orientação a Objetos para alterar o comportamento do jogo. Assim sendo, o aprendiz, obviamente, não compõe sua solução direcionada à escrita na saída padrão para interferir no jogo.

Como artifício, a maioria das classes abordadas pelos desafios possuem duas versões. Uma primeira, mostrada ao aprendiz pela ferramenta, em caráter de contextualização do que deve ser implementado. A segunda versão, é usada internamente pela ferramenta e se responsabiliza, em adição, pela escrita na saída.

Os trechos de código da segunda versão que correspondem à saída são ocultos na primeira. Portanto, usa-se a indicação de que tais trechos de código foram abstraídos e que não exigem preocupação por parte do aprendiz.

Como exemplo, serão contrastadas as classes referentes ao personagem no primeiro cenário (Seção 6.1.1). Primeiro, na Figura 23, se transcreve a classe conforme mostrada na contextualização feita ao aprendiz. Depois, a Figura 24, apresenta a classe admitida internamente como artifício no suporte à solução efetivada.

```
public class Personagem {

	public Personagem(){
	}

	public void andar(){
		//comportamento abstraido
	}
}
```

Figura 23: Classe Personagem (contextualização)

```
public class Personagem {

	public Personagem(){
		System.out.println("c1 personagem");
	}

	public void andar(){
		System.out.println("c2 andando");
	}

	public void pegarMoeda(){
		System.out.println("c3 moeda");
	}
}
```

Figura 24: Classe Personagem admitida internamente

7 DIFICULDADES ENCONTRADAS

Neste capítulo são relatadas as principais dificuldades que se encontrou no decorrer do projeto.

7.1 COMPLEXIDADE DO TRABALHO

A primeira dificuldade enfrentada na realização da pesquisa diz respeito à complexidade do trabalho em si. Obviamente, desde a idealização inicial, se imaginou que a condução do projeto exigiria esforço extra, principalmente pelas várias particularidades não contempladas de forma direta pela matriz curricular do curso de Tecnologia em Sistemas para Internet. Além disso, no decorrer do projeto, diversos obstáculos relacionados à complexidade da pesquisa foram encontrados, exemplo disto são:

- O desenvolvimento de jogos ainda é uma tarefa complicada, mesmo com o subsídio de uma IDE como a Unity 3D, que, embora abstraia detalhes como o motor de física, ainda exige muito de um iniciante na área;
- Inexperiência em trabalhar com imagens, animações 3D, *scripts* para definição de comportamentos e integração de tecnologias;
- Falta de habilidade para a sincronização dos diversos elementos que compõem um jogo;
- Peculiaridades do processo de desenvolvimento de um jogo, tais como a concepção inicial, criação de cenários e roteiros;
- Diferenciação do processo de análise e desenvolvimento de um jogo frente a outros aplicativos tradicionais;
- Dificuldade em mensurar o tempo requerido por tarefas específicas da área.

7.2 ESCOLHA DE FERRAMENTAS E DECISÕES DE PROJETO

Uma das dificuldades residiu no pouco contato acadêmico com a área e

consequente falta de parâmetros iniciais de tarefas como a escolha de ferramentas para a autoria do jogo. Ocorreu que, desde o começo do projeto, diante das pesquisas realizadas, não era claro de se visualizar os possíveis impactos dos prós e contras de cada tecnologia no resultado final do jogo desenvolvido. Um período considerável de familiarização foi exigido.

7.3 INTERAÇÃO DO JAVA COM UNITY 3D

Por não se tratar de uma linguagem com suporte nativo em Unity 3D, houve bastante esforço para que a integração com a ferramenta ocorresse. Na sequência são registradas as tentativas realizadas.

7.3.1 Primeira Tentativa: Integração por meio do Eclim com Euclid

A primeira tentativa consistiu em investigar como se dava o processo de compilação no jogo educacional CodeSpells. Para isso, houve contato com a equipe desenvolvedora do citado projeto (CodeSpells Team). Com muita demora, uma das desenvolvedoras explicou que a integração ocorria por meio da ferramenta Eclim[1], somada ao pacote Euclid[2], de autoria da própria equipe, porém disponível à comunidade.

O Eclim possibilita o acesso às funcionalidades da IDE Eclipse, tais como autocompletar, buscas, validação de código, entre outros, através da linha de comando ou mesmo conexões locais de rede. Permite também que esses recursos sejam acessíveis por um editor externo, como o Vim ou outro de própria autoria do usuário.

Por sua vez, o pacote Euclid faz a interconexão do Euclim com o Unity 3D. Tão logo, as funcionalidades citadas tornam-se disponíveis aos projetos desenvolvidos no Unity. As tecnologias citadas se relacionam conforme expressa a Figura 25, onde o Unity, por meio do pacote Euclid, se conecta com o Eclim que acessa as funcionalidades do Eclipse. Convém lembrar que o Eclipse interage com a plataforma Java através da JDK (*Java Development Kit*).

1 Disponível em http://eclim.org.
2 https://github.com/srfoster/Euclid.

Unity ↔ Euclid ↔ Eclim ↔ Eclipse ↔ SDK

Figura 25: Integração entre as ferramentas

Entretanto, a própria equipe desenvolvedora do CodeSpells alertou que já reconheceu essa abordagem como uma solução inviável. Primeiramente, porque consistia em uma exigência muito grande, para os usuários finais, manter o Eclipse instalado, e em execução prévia, para que o CodeSpells funcionasse.

A equipe ainda informou que a proposta alternativa empregava somente o Java SDK (*Software Development Kit*) para exibir as mensagens de compilação dentro do jogo, através da análise (*parsing*) do texto retornado. Contudo, a equipe não divulgou detalhes adicionais sobre isso, nem qualquer esclarecimento sobre a compilação e incorporação dinâmica de códigos Java no Unity.

Por fim, em testes realizados pela presente pesquisa, a própria integração por meio das tecnologias Euclim e Euclid se mostrou insatisfatória.

7.3.2 Segunda Tentativa: Executar JavaScript dentro do Java

A abordagem consistia em se beneficiar da capacidade que a JVM tem de executar códigos de outras linguagens, como por exemplo, Ruby, Lua, Scala e Javascript.

Convém lembrar que Javascript é uma das linguagens suportadas nativamente pelo Unity. Assim, a implementação Java poderia acessar funcionalidades de um código JavaScript incorporado ao Unity.

A tentativa se mostrou bastante válida nos testes realizados. Todavia, no momento da integração, se constatou que o Unity utiliza uma sintaxe pouco particularizada da linguagem JavaScript, resultando em incompatibilidade. Como exemplo, a própria declaração de variáveis é feita de maneira diferente, além do provimento de funções adicionais inexistentes na linguagem JavaScript aceita pelo Java.

7.3.3 Terceira Tentativa: Execução de Comandos no Terminal pelo Unity 3D

Conforme colaboração do professor Andres Jessé Porfirio, chegou-se à alternativa de executar comandos no terminal por meio do próprio Unity 3D. Com isso, foi possível que se ativasse o compilador Java (`javac`) e, em seguida, a execução do programa resultante.

A abordagem consiste na captura da saída do programa resultante e utilização disso para guiar o funcionamento do jogo no Unity. Outra dificuldade residiu em ocultar os detalhes de implementação, relacionados com a saída em tela, da perspectiva tida pelo aprendiz na utilização do jogo.

Como esta foi, justamente, a abordagem que se efetivou, os detalhes de implementação foram discutidos no Capítulo 5.

8 RESULTADOS E LIMITAÇÕES

São relacionados, neste capítulo, os resultados alcançados com a pesquisa, bem como as limitações identificadas diante do que foi desenvolvido.

8.1 RESULTADOS ALCANÇADOS

Tendo em vista os objetivos expostos na Seção 1.1, são sumarizados na sequência os principais resultados desta pesquisa:

1. Levantamento de conceitos e pesquisas atuais relacionados ao desenvolvimento de jogos educacionais, e mais especificamente de *serious games* (conforme capítulos 2 e 3);
2. Estudo das tecnologias envolvidas na construção de um *serious game*. Também foi realizada a documentação das tentativas e maneiras com que tais tecnologias foram integradas (capítulos 4 e 7);
3. Proposição de meios para tornar o aprendizado inicial do paradigma Orientado a Objetos mais lúdico e atrativo, pela interseção dos conceitos de Ambientes Interativos de Aprendizagem, Ambientes de Descoberta Guiada e Gamificação, aplicados na construção do protótipo de *serious game* desenvolvido (Capítulo 5);
4. Evolução dos estudos e da capacidade técnica da proponente perante a linguagem Java e a plataforma Unity 3D (Capítulo 6);
5. Implementação de um *serious game* funcional com um primeiro cenário que aborda os conceitos inicias do paradigma Orientado a Objetos, desde a introdução, passando por classes e objetos, bem como atributos e métodos. Outros dois cenários foram propostos, um primeiro para o ensino da estrutura condicional if-*else e* outro para aquelas de repetição (Capítulo 6);
6. Possibilidade de apresentar, por meio do protótipo desenvolvido, a disciplina de Programação de Programadores, de uma forma mais empolgante, útil para um primeiro contato nas aulas inicias e nas feiras de profissões (Seção 5.6);

7. Disseminação de conhecimento em eventos da comunidade científica da área, tendo em vista o amadurecimento da proponente para o ingresso em curso de pós-graduação (mestrado):
    1. Publicação do resumo expandido intitulado Serious Game como Objeto de Aprendizagem para Programação de Computadores, em coautoria com os professores Diego Marczal e Eleandro Maschio, na III Semana de Integração Ensino, Pesquisa e Extensão da Unicentro – SIEPE. Apresentado como painel;
    2. Publicação do resumo expandido intitulado Um Objeto de Aprendizagem para os Conceitos Introdutórios de Orientação a Objetos, também em coautoria com os professores Diego Marczal e Eleandro Maschio, no I Encontro de Computação e Matemática Aplicada – ECOMAP. Apresentado como comunicação oral;
8. Incentivo à criação e participação voluntária no Projeto de Pesquisa intitulado Produção de Recursos Educacionais Digitais, também orientado pelos professores Diego Marczal e Eleandro Maschio, com conforme Edital 21/2013 – PROGRAD, de 17 de junho de 2013;
9. Participação do Grupo de Pesquisa denominado Núcleo de Pesquisas no Ensino da Computação – BlackFace, da Coordenação do Curso de Tecnologia em Sistemas para Internet.

Por fim, enfatiza-se que o enriquecimento de formação trazido à proponente, por meio desta pesquisa, sobrepôs as dificuldades descritas no Capítulo 7. Alcançou-se uma parcela considerável de conhecimento teórico e prático, como também um diferencial frente aos conteúdos que compõem a matriz curricular do curso de Tecnologia em Sistemas para Internet.

8.2 LIMITAÇÕES DA SOLUÇÃO PROPOSTA

Nesta seção se descreve as principais limitações observadas na solução proposta. O suprimento delas será remetido aos trabalhos futuros (Seção 9.1):

1. Perante as dificuldades técnicas apresentadas, os outros dois cenários modelados, correspondentes ao conteúdo de estrutura condicional *if-else* e estruturas de repetição, não foram implementados em completude;
2. Necessidade de revisar a incorporação dinâmica de códigos Java no Unity, para que ocorra de maneira fluente e dispense a captura da saída do programa resultante a fim de guiar o funcionamento do jogo;
3. Embora não tenha sido objetivo da pesquisa, o *feedback* do protótipo se restringe às mensagens vindas do próprio compilador, não tendo ainda amparo didático; e
4. Os textos tutoriais têm caráter estático e a ordem de apresentação não é personalizada ao aprendiz, pois não havia modelagem de perfil prevista.

9 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

O Trabalho de Conclusão de Curso corrente apresentou uma abordagem, baseada no conceito de *serious game*, para a dinamização do aprendizado inicial do paradigma Orientado a Objetos. A pesquisa decorreu da constatação de carências específicas no que diz respeito ao nicho contemplado.

O referencial teórico da pesquisa foi fundamentado sobre aspectos educacionais, jogos e *serious games* propriamente ditos. Depois se levantou uma discussão sobre ambientes que se mostram correlatos ao trabalho desenvolvido, como Alice, Greenfoot, Scratch, CodeSpells e Belesminha.

Os formalismos da solução foram estabelecidos na convergência de Ambientes Interativos de Aprendizagem, Ambientes de Descoberta Guiada e Gamificação. Com isso, se alcançou o enfoque da solução e o diálogo de interação propostos.

Depois, a fim de remeter o referido formalismo à prática, foram introduzidas as tecnologias que seriam empregadas para a implementação do protótipo. Oportunamente, seguiu-se com detalhes do desenvolvimento do protótipo de *serious game* objetivado.

Em caráter de finalização, foram relacionadas as dificuldades que se apresentaram no decorrer do projeto. No mesmo sentido, demarcaram-se os resultados alcançados e as limitações observadas com a conclusão da pesquisa.

Como consideração final, acredita-se que os objetivos cumpridos podem consistir em uma contribuição relevante, em nível de graduação, para a intersecção de áreas abrangida. Além disso, pensa-se que o pioneirismo de investir esforços discente e docente neste nicho pode incentivar pesquisas futuras na instituição. Nesse sentido, espera-se que outras pesquisas se espelhem no trabalho realizado e motivem-se em desenvolver instrumentos adicionais direcionados à mesma preocupação com o ensino em Ciência da Computação.

9.1 TRABALHOS FUTUROS

À proporção do desenvolvimento desta pesquisa pôde-se perceber possíveis trabalhos futuros que venham a estendê-la e superar as limitações constatadas (Seção 8.2):

1. Concluir a implementação dos dois outros cenários modelados, que abrangem o conteúdo de estrutura condicional *if-else* e estruturas de repetição;
2. Estender a abrangência do protótipo, progressivamente, aos demais conteúdos de Orientação a Objetos, como herança, polimorfismo, interface, entre outros;
3. Prover o protótipo de cenários adicionais que reforcem o aprendizado de todos esses conteúdos;
4. Investir na produção de material didático (principalmente textos e vídeos) que permitam melhor aproveitamento, por professores e aprendizes, da abordagem proposta;
5. Pesquisar e efetivar aspectos de compilação e incorporação dinâmica de códigos Java para que sejam integrados com o Unity 3D, como alternativa à captura e análise (*parsing*) da saída em tela. Uma remota possibilidade é que o Unity passe a tratar códigos Java nativamente, em complemento às linguagens JavaScript, C# e Boo;
6. Expandir as funcionalidades do editor de código Java do protótipo, para que admita recursos como autocompletar, análise sintática em tempo real, *pretty printers* (convenções de formatação associadas à sintaxe), buscas e substituições, desfazer e refazer, numeração de linhas, entre outros;
7. Promover amparo didático às mensagens de retorno vindas do próprio compilador a fim de aperfeiçoar o *feedback* ao aprendiz;
8. Incorporar aspectos da modelagem de perfil do usuario com o propósito de adequar a interação para cada aprendiz;
9. Avaliar a efetividade do protótipo por meio da coleta de dados a partir do uso inicial. Também se mostra conveniente usar do mesmo processo para

percepção da necessidade de aperfeiçoamento das funcionalidades do protótipo;

10. Considerar a possibilidade de portar o protótipo para a *web*, visto que o próprio Unity provê suporte à plataforma.

REFERÊNCIAS

BARNES, David J.; KOLLING, Michael. **Programação Orientada a Objetos com Java:** Uma Introdução Prática Usando o BlueJ. 4. ed. São Paulo, SP: Pearson Prentice Hall, 2009. 455 p. ISBN: 9788576051879.

BV, Tiobe Software. **TIOBE Programming Community Index for November 2013**: November Headline: Programming languages of Microsoft are gaining popularity. Disponível em: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>. Acesso em: 10 nov. 2013.

CASTELA, Rodrigo Tenedini. **Introdução a Linguagem Java**. Brasil, 2010. Disponível em: <http://www.dotsharp.com.br/programacao/java/introducao-a-linguagem-java.html>. Acesso em: 17 out. 2013.

CLUA, Esteban Walter Gonzalez. **A Game Oriented Approach for Teaching Computer Science.** Anais do XXVIII Congresso da SBC, Workshop sobre Educação em Computação. Belém do Pará (PA), 2008. p. 10–19.

COOPER, Stephen; DANN, Wanda; PAUSCH, Randy. **Alice: A 3-D Tool for Introductory Programming Concepts**. Proceedings of the fth annual CCSC northeastern conference on The journal of computing in small colleges, CCSC '00, p.107-116, USA, 2000. Consortium for Computing Sciences in Colleges.

CORCORAN, Elizabeth. **The 'Gamification' Of Education.** O'Reilly Media, 2010. Disponível em: <http://www.forbes.com/2010/10/28/education-internet-scratch-technology-gamification.html>. Acesso em: 14 nov. 2013.

FINCHER, Sally; COOPER, Stephen; KOLLING, Michael; MALONEY, John. **Comparing Alice, Greenfoot & Scratch**. SIGCSE '10: Proceedings of the 41st ACM technical symposium on Computer science education, p. 192-193, New York, NY, USA, 2010. ACM.

GOLDSTONE, Will. **Unity 3.X Game Development Essencials:** Game

Development wth C# and Javascript. Birmingham, UK: Pack Publishing Ltda, 2011. 462 p. ISBN 9781849691444.

JESUS, Elieser e RAABE, André Luís Alice. **Avaliação Empírica da Utilização de um Jogo para Auxiliar a Aprendizagem de Programação**. In: XXI Simpósio Brasileiro de Informática na Educação, 2010, João Pessoa. Anais, 2010.

LOPES, Sergio; SILVEIRA, Paulo; SILVEIRA, Guilherme. **Introdução à Arquitetura e Design de Software:** Uma Visão Sobre a Plataforma Java. Rio de Janeiro, RJ: Elsevier, 2012. 257 p. ISBN 9788535250299.

MACHADO, Diogo. **Indústria de games bate Hollywood e deve arrecadar US$ 74 bi até 2017**. Disponível em: <http://www.correiodeuberlandia.com.br/entretenimento/industria-de-games-bate-hollywood-e-deve-arrecadar-us-74-bi-ate-2017/>. Acesso em: 09 dez. 2013.

MACHADO, Liliane dos Santos et al. **Serious Games Baseados em Realidade Virtual para Educação Médica**. Revista Brasileira de Educação Médica, Rio de Janeiro, v. 35, n. 2, p. 254-262, junho 2011.

MASCHIO, Eleandro. **ANEXO VII DO REGULAMENTO DE PESQUISA DA UNIVERSIDADE ESTADUAL DO CENTRO-OESTE, UNICENTRO: MODELO DE RELATÓRIO TÉCNICO FINAL DE PROJETO PQI E PQE**. Guarapuava 2010. 18 p.

MASCHIO, Eleandro. **Modelagem do Processo de Aquisição de Conhecimento apoiado por Ambientes Inteligentes**. 2013. 104 f. Tese (Doutorado em Informática) – Universidade Federal do Paraná, Curitiba, 2013.

MASCHIO, Eleandro. **Uma Abordagem Metacognitiva Através de Múltiplas Representações Externas para o Ensino de Programação de Computadores**. 2007. 107 f. Dissertação (Mestrado em Informática) - Universidade Federal do Paraná, Curitiba, 2007.

NAVARRO, Gabrielle. G**amificação: a transformação do conceito do termo jogo no contexto da pós-modernidade**. 2013. 26 f. Monografia (Especialização em

Cultura, Mídia e Informação), Centro de Estudos Latino-americanos Sobre Cultura e Comunicação, Butantã, 2013.

ORACLE. **Java Tutorials**. Disponível em: <http://docs.oracle.com/javase/tutorial/>. Acesso em: 04 nov. 2013.

PAPADOPOULOS, Yannis; TEGOS, Stergios. **Using Microworlds to Introduce Programming to Novices.** Panhellenic Conference on Informatics, 2012.

PIMENTEL, Andrey Ricardo; DIRENE, Alexandre Ibrahim. **Medidas Cognitivas no Ensino de Programação de Computadores com Sistemas Tutores Inteligentes**. Anais do Simpósio Brasileiro de Informática na Educação (SBIE), Fortaleza, Ceará, 1998.

PONTES, Felipe; ROSA, Guilherme. **Conheça a gamificação, que transforma suas tarefas cotidianas em games: Cuidar da saúde, render no trabalho e melhorar o mundo jogando**: são os jogos da vida. Disponível em: <http://revistagalileu.globo.com/Revista/Common/0,,EMI291109-17773,00-CONHECA+A+GAMIFICACAO+QUE+TRANSFORMA+SUAS+TAREFAS+COTIDIANAS+EM+GAMES.html>. Acesso em: 15 nov. 2013.

PRIETO, Lilian Medianeira et al. **Uso das Tecnologias Digitais em Atividades Didáticas nas Séries Iniciais**. CINTED-UFRGS: Porto Alegre (RS), 2005.

ROMANATO, Allan. **Entenda como Funciona a Java Virtual Machine JVM**. Disponível em: <http://www.devmedia.com.br/entenda-como-funciona-a-java-virtual-machine-jvm/27624>. Acesso em: 09 dez. 2013.

THE CODESPELLS TEAM. **Welcome to CodeSpells.** Disponível em: <https://sites.google.com/a/eng.ucsd.edu/codespells/home>. Acesso em: 09 dez. 2013.

WENGER, Etienne. **Artificial Intelligence and Tutoring Systems:** Computational Approaches to the Communication of Knowledge. Morgan Kaufmann Publishers, 1987. 486 p.

ZANINI, Adriana Salvador. **Avaliação da Influência dos Enunciados na Resolução de Problemas de Programação Introdutória**. 2013. 126 f. Dissertação (Mestrado Acadêmico em Computação Aplicada) – Universidade do Vale do Itajaí, Itajaí, 2013.

ZANINI, Adriana Salvador e RAABE, Andre. **Análise dos enunciados nos problemas de programação introdutória em cursos de Ciência da Computação no Brasil**. In: XX Workshop de Educação em Computação, Congresso Anual da Sociedade Brasileira de Computação, Curitiba, 2012.

ZYDA, Michael. **From visual simulation to virtual reality to games**. USC Information Sciences Institute, set.2005. Disponível em: <http://gamepipe.usc.edu/~zyda/pubs/zyda-ieee-computer-sept2005.pdf>. Acesso em: 18 março 2010.