Victor Augusto Zunho

Desenvolvimento de um Jogo Eletrônico Educativo para Ensino de Algoritmos

Passo Fundo

2021

Victor Augusto Zunho

Desenvolvimento de um Jogo Eletrônico Educativo para Ensino de Algoritmos

Monografia apresentada ao Curso de Ciência da Computação Instituto Federal Sul-rio-grandense, Câmpus Passo Fundo, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE - CÂMPUS PASSO FUNDO

Orientador: Prof. Dr. João Mário Lopes Brezolin

Coorientadora: Ma.Carmen Scorsatto

Passo Fundo

2021

Agradecimentos

Quero agradecer a meus pais e a meu irmão por seu apoio durante esse período, principalmente por me incentivarem a continuar e nunca desistir.

Ao meu orientador João Mário Lopes Brezolin, por sua paciência e apontamentos durante as orientações. A professora Carmen Vera Scorsatto, por sua colaboração e apoio.

Também gostaria de agradecer aos professores do curso de Ciência da Computação e aos meus colegas por sua cooperação durante toda a duração do curso.

Por fim, gostaria de agradecer a todos que de alguma forma colaboraram para tornar esse trabalho possível.

Resumo

A complexidade do conteúdo da disciplina de algoritmos traz a necessidade de qualificar o ensino. Por motivar o aprendizado e a busca novos conhecimentos fora da sala de aula, jogos demonstram-se ferramentas eficazes no ensino. Apesar disso, vários jogos de ensino de algoritmos na verdade ensinam apenas a utilização e a sintaxe de uma linguagem de programação. Por isso, foi proposto o desenvolvimento de um jogo eletrônico para auxiliar no processo de aprendizagem de algoritmos, usando como base a Teoria Construcionista de Papert.

Palavras-chave: Jogos; Aprendizagem; Algoritmos

Abstract

The complexity of the contents of the Algorithms makes it necessary to qualify the teaching methods. As games motivate learning and the search for knowledge outside classroom, they can be useful tools in learning. Even so, many of the Algorithms teaching games actually teach the use and the syntax of a programming language. For that reason, the development of an electronic game to assist in the process of learning Algorithms, based on Papert's Constructionist Theory.

Keywords: Games; Learning; Algorithms

Lista de ilustrações

Figura 1 – Usando a referência: Cubo.transform.Translate(2, 0, 0). 21
Figura 2 – Usando a simplificação: TRANSLATE(2, 0, 0). 21
Figura 3 – Diagrama representando o fluxo do jogo, indicando o ciclo básico das atividades realizadas pelo jogador. 24
Figura 4 – Puzzle inicial I. 25
Figura 5 – Puzzle inicial II. 26
Figura 6 – Puzzle inicial III. 27
Figura 7 – Troca de Operações. 28
Figura 8 – Situação-problema final 29
Figura 9 – Hierarquia da interface, a partir do menu inicial do jogo. 30
Figura 10 – Menu inicial do jogo. 31
Figura 11 – Menu de pause do jogo. 31
Figura 12 – Menu de opções do jogo. 32
Figura 13 – Tela inicial do jogo. 33
Figura 14 – Tela de entrada de instruções do jogo. 34
Figura 15 – Tela de conclusão do jogo. 34
Figura 16 – Diálogo do tipo 1, ao aproximar-se do GUIA. 35
Figura 17 – Diálogo do tipo 2, ao iniciar o primeiro problema. 35

Lista de abreviaturas e siglas

3D	três dimensões
DLL	(*Dynamic Link library*), biblioteca de vínculo dinâmico
GDD	(*Game Design Document*), Documento de *Design* de Jogo
IFSUL	Instituto Federal de Educação, Ciência e Tecnologia Sul-rio-grandense
MCS	Mono CSharp (compiler)

Sumário

1 INTRODUÇÃO 9
1.1 OBJETIVOS 9
1.1.1 Objetivos Específicos 9
1.2 ORGANIZAÇÃO DO TRABALHO 10

2 REFERENCIAL TEÓRICO 11
2.1 Disciplina de algoritmos 11
2.2 Aprendizagem através de jogos de computador 12
2.3 Ferramentas utilizadas no desenvolvimento de jogos 13
2.3.1 Motor de desenvolvimento de jogos 14
2.3.2 Modelagem 3D 14
2.3.3 Documentação dos jogos 14
2.4 Trabalhos relacionados 15

3 DESENVOLVIMENTO DO APLICATIVO PROPOSTO 19
3.1 Tecnologias utilizadas 19
3.1.1 Unity 3D 19
3.1.2 Blender 19
3.1.3 MCS 19
3.2 Compilação em tempo de execução 20
3.3 Descrição do jogo 22
3.3.1 Puzzles 22
3.3.2 Situações-Problemas 23
3.3.3 Fluxo do jogo 23
3.3.4 Problemas propostos no jogo 24
3.3.4.1 Problemas Iniciais 24
3.3.4.2 Problemas de trocas de operações 27
3.3.4.3 Problema final 28
3.3.5 Interface do Jogo 29
3.3.5.1 Menus do jogo 30
3.3.5.2 Telas de jogo 32
3.3.5.3 Telas de diálogo 34

4 VALIDAÇÃO DO SISTEMA 36

5 CONSIDERAÇÕES FINAIS 38

REFERÊNCIAS 39

APÊNDICE A – PRÉ-TESTE 41

APÊNDICE B – PÓS-TESTE 42

1 Introdução

A disciplina de Algoritmos ocorre no primeiro semestre do curso de Ciência da Computação do IFSUL no Campus Passo Fundo e é o primeiro contato com os conceitos de lógica e linguagens de programação no curso. A complexidade do conteúdo da disciplina, causada pela sua relação intrínseca com todas as áreas avançadas da ciência da computação (CURRICULA; SOCIETY, 2013; ARAUJO et al., 2017), demanda a necessidade de se pensar em alternativas pedagógicas para qualificar o ensino.

A imersão dos alunos no ambiente dos jogos digitais motiva o aprendizado e abusca por novos conhecimentos fora da sala de aula. Nesse sentido, os jogos mostram-se ferramentas eficazes no ensino. Porém, vários dos jogos que se propõem ao ensino de algoritmos na verdade ensinam a utilização e a sintaxe de uma linguagem de programação. Por isso, eles não apresentam ao jogador a maioria dos conceitos básicos de algoritmos dentro do jogo, impedindo que o conteúdo seja apresentado dentro do contexto dele e perdendo a principal vantagem dessa abordagem (JR; SILVEIRA, 2016; GUNTER; KENNY; VICK, 2008). Essa situação pode ser alterada desenvolvendo-se o jogo desde o princípio tendo em vista o aprendizado de algoritmos, seguindo um método de ensino apropriado e de forma a permitir que o aluno construa seu aprendizado a partir dos conceitos estudados na disciplina de Algoritmos.

Assim, neste trabalho propõe-se o desenvolvimento de um jogo eletrônico para auxiliar no aprendizado de Algoritmos, usando como base a Teoria Construcionista de (PAPERT, 1980).

1.1 OBJETIVOS

Desenvolver um jogo eletrônico para auxiliar no aprendizado da disciplina de Algoritmos.

1.1.1 Objetivos Específicos

- Pesquisar teorias de aprendizagem que podem ser adequadas ao ensino de Algorítimos;
- Avaliar os conteúdos e exercícios que são trabalhados na disciplina de algoritmos;
- Abordar dentro do jogo as principais dificuldades dos alunos em relação ao conteúdo da disciplina;

- Avaliar as ferramentas computacionais que poderão ser utilizadas no desenvolvimento do jogo;
- Realizar o desenvolvimento do jogo proposto;
- Aplicar métodos de ensino de algoritmos em um jogo eletrônico;
- Realizar a validação do jogo junto aos alunos da disciplina de Algoritmos.

1.2 ORGANIZAÇÃO DO TRABALHO

O presente trabalho é constituído por capítulos, o que possibilita uma melhor organização e compreensão dos temas abordados.

No Capítulo 2 são apresentados os conceitos utilizados para o desenvolvimento do aplicativo proposto nesse trabalho. Inicialmente é feita uma revisão bibliográfica sobre a disciplina de algoritmos, aprendizagem através de jogos de computador e as ferramentas utilizadas no desenvolvimento de jogos.

O Capítulo 3 aborda o desenvolvimento do aplicativo proposto, começando pelas tecnologias utilizadas no desenvolvimento, então partindo para a explicação sobre como a compilação em tempo de execução foi utilizada e depois apresentando uma descrição do jogo.

O Capítulo 4 apresenta a validação do sistema, apresentando os testes realizados e os resultados apresentados.

Por fim, o Capítulo 5 aborda as considerações finais do presente trabalho.

2 Referencial teórico

Nessa seção são apresentadas as definições e conceitos necessários para a compreensão do trabalho, incluindo disciplina de algoritmos, aprendizagem através de jogos de computador e ferramentas utilizadas para o desenvolvimento de jogos.

2.1 Disciplina de algoritmos

A disciplina de algoritmos fornece as conceitos e habilidades necessários para projetar, implementar e analisar algoritmos utilizados na resolução de problemas, sendo estes essenciais em todas as áreas avançadas da ciência da computação (CURRICULA; SOCIETY, 2013). Entende-se algoritmo como "uma sequência de passos computacionais que transformam a entrada na saída" (CORMEN et al., 2012, p.3), sendo a "entrada" qualquer condição inicial apresentada e a "saída" qualquer condição final desejada. Dessa forma, pode-se afirmar que a disciplina de algoritmos tem como objetivo principal desenvolver as habilidades necessárias para compreender, criar e utilizar procedimentos ou um conjunto de etapas bem definidas que transformem um valor de entrada em uma saída para resolver problemas relacionados à computação.

O conteúdo abordado nas primeiras disciplinas relacionadas a algoritmos depende dos objetivos da instituição onde o curso é ofertado (ROCHA et al., 2005; ARAUJO et al., 2017; CURRICULA; SOCIETY, 2013). Além do conteúdo abordado, a forma como ele é ensinado também depende da instituição (ARAUJO et al., 2017; CURRICULA; SOCIETY, 2013). Porém, "quando aplicável, deve-se empregar metodologias ativas, de forma que o aluno passe mais tempo em atividades nas quais seja protagonista no processo de ensino e aprendizagem"(ARAUJO et al., 2017, p. 37). Assim, a instituição deve escolher a ordem e de que forma o conteúdo é ensino no curso, mas deve também estimular a independência dos alunos.

No caso do IFSUL, Campus Passo Fundo, o currículo da primeira disciplina de algoritmos é focado em familiarizar o aluno com a ideia de algoritmos e como eles são utilizados, ensinando junto a eles conceitos básicos sobre linguagens de programação. Na parte prática o aluno utiliza conceitos como variáveis e expressões aritméticas na própria linguagem de programação e onde aprende e utiliza os conteúdos de formais, como no caso dos testes de mesa e outras representações de algoritmos (Fluxograma, Português estruturado, etc.) (BRASIL, 2017). Observa-se,dessa forma, uma tentativa de equilibrar aquilo que é visto pelo aluno como um uso "prático" dos conteúdos vistos na disciplina, como a criação de programas utilizando linguagens de programação, com conceitos mais teóricos, como a realização de testes de mesa utilizando português estruturado.

2.2 Aprendizagem através de jogos de computador

A aprendizagem através de jogos de computador consiste na utilização de jogos de computador no ensino para apresentar uma parte do conteúdo a ser estudado e facilitar a aprendizagem do aluno. Os jogos usados na aprendizagem são chamados educacionais e são especificamente feitos para ensinar pessoas sobre um assunto, expandir conceitos, reforçar o desenvolvimento, auxiliá-los em aprofundar-se em um assunto, aprender uma habilidade ou buscar uma mudança de atitude enquanto jogam (ABT, 2002 apud BATTISTELLA; WANGENHEIM, 2016). De acordo com Ribeiro et al. (2015), os jogos educacionais podem ser divididos em dois tipos: aqueles que foram planejados desde o início com fins educacionais e se destinam a um uso específico para determinada disciplina ou aqueles comerciais que o professor consegue estabelecer uma relação entre o conteúdo a ser trabalhado e o conteúdo do jogo. Apesar de ser possível utilizar ambos os tipos de jogos no ensino, Gunter, Kenny e Vick (2008) afirmam que jogos educacionais precisam ter um design que suporte atividades cognitivas de forma prolongada e que sejam firmemente integradas para serem efetivos, algo que possui uma menor chance de ocorrer em um jogo comercial que não foi planejado para ser utilizado como uma ferramenta de ensino.

Para um design de aprendizagem efetivo de jogo educacional é necessário encontrar o balanço entre jogabilidade, diversão e design de aprendizagem sólido, que alinhe os resultados com avaliações dentro do jogo ou como parte da experiência (FREITAS, 2018). De acordo com Gunter, Kenny e Vick (2008), o design de um jogo educacional precisa ser fundado em dois requisitos: o conteúdo precisa ser introduzido e reutilizado de forma hierárquica e, se o jogo tiver o objetivo de ensinar conteúdo acadêmico por si só, então é necessário que o conteúdo alvo esteja intrinsecamente unido com o conteúdo da fantasia ou história do jogo. De acordo com eles, isso faz com que faz com que o jogador aplique as habilidades e conhecimentos desejados enquanto joga o jogo, o que faz com que o jogo e o conteúdo não sejam dois elementos separados. Isso possibilita retornar *feedback* ao jogador sem impedi-lo de tentar descobrir a resposta correta e permite que ele aprenda usando os conhecimentos anteriores em um passo a passo, pois uma resposta incorreta não impede o jogo de continuar (GUNTER; KENNY; VICK, 2008). Isso faz com que o conteúdo receba um novo contexto por estar inserido no cenário do jogo, o que o torna mais significativo para o aluno e, consequentemente traz uma melhoria no aprendizado do aluno (JR; SILVEIRA, 2016; GUNTER; KENNY; VICK, 2008).

Estando inserido no jogo como parte de seu sistema, o conteúdo é apresentado de forma mais contextualizada. Além disso, com o conteúdo inserido em sua fantasia, o jogo permite que o aluno relacione os conceitos a serem aprendidos com os objetos que aparecem no jogo, como personagens, itens, textos ou locais. Ao possibilitar que o jogador aplique os conceitos do conteúdo a ser ensinado nesses objetos para testá-los, torna-se possível um ambiente de aprendizado onde o próprio indivíduo constrói seu conhecimento

de acordo com as relações geradas por ele entre o próprio conhecimento e o que se pretende aprender. De acordo com a ideia do construcionismo de Papert (1980), essa construção do conhecimento ocorre em duas etapas: relacionar o novo conceito a ser aprendido ao que já se sabe e apossar-se do novo conceito, utilizando-o, construindo com ele. Essa possibilidade de construção do aprendizado é facilitada por vários dos conceitos apresentados por jogos de aprendizado, como o retorno de *feedback* ao jogador, não considerando o erro como um problema, mas como algo para se construir sobre até atingir a resposta correta.

Dessa forma, pode-se criar um jogo de aprendizagem eficaz ao tornar o conteúdo a ser ensinado como um elemento do jogo que o jogador utiliza. Isso possibilita apresentar as ideias centrais do conteúdo ao jogador e treiná-lo nelas antes mesmo de apresentar o conteúdo em si. Ao fazer isso, o jogador terá mais facilidade para compreender o conteúdo. Após isso, pode-se treinar o conteúdo em si, com maior formalidade para que o aluno/jogador saiba como aplicar os conceitos do conteúdo no contexto necessário. Isso significa apresentar as ideias antes do conteúdo. Ao ensinar algoritmos de ordenação, por exemplo, pode-se apresentar cada algoritmo com uma descrição mais informal e possibilitar que os alunos utilizem os algoritmos para ordenar objetos antes de aprenderem como construir o algoritmo em uma linguagem de programação, que possui outras formalidades, como uso de memória, sintaxe e outros, que podem dificultar a compreensão das ideias de cada algoritmo. Após compreender a ideia central por trás do conteúdo, o aluno terá uma maior facilidade para formalizar essa ideia, pois ele terá algo para relacioná-la (PAPERT, 1980), nesse caso o contexto do jogo.

2.3 Ferramentas utilizadas no desenvolvimento de jogos

Existem diversas etapas no desenvolvimento de um jogo, incluindo física (se houver), plataforma em que o jogo rodará, criação do cenário, criação dos modelos, arte, história (se houver), etc Novak (2011). Ao longo do desenvolvimento do jogo essas etapas podem tornar-se complexas e, por isso, é comum utilizar ferramentas para auxiliar no desenvolvimento do jogo, que auxiliam nas etapas onde podem haver mais dificuldade.

No caso de um jogo 3D as etapas que apresentam maior complexidade e podem gerar mais problemas são a física e a criação do cenário e modelos. Além disso, devido a complexidade e ao fato de o projeto de um jogo evoluir durante o tempo, é dificultada a utilização de ferramentas de documentação comumente utilizadas, como a UML. Isso sendo necessário o uso de ferramentas para auxiliar no desenvolvimento da física e a criação dos cenários e modelos 3D no jogo e de técnicas de documentação especificas para jogos.

2.3.1 Motor de desenvolvimento de jogos

Um Motor de Jogo (do inglês *Game Engine*) é um *software* feito com base em um jogo ou um gênero de jogos que pode ser usado como base para criação de diferentes jogos diferentes sem grandes modificações em seu código original Gregory (2018), mas que não especifica a lógica ou comportamento do jogo ou o ambiente do jogo (modelos, sons, etc) (LEWIS; JACOBSON, 2002, p. 28). Podendo ser considerado, assim, um código ou software que possui um conjunto de recursos padrão usados para a construção de um tipo específico de jogo. Esses recursos são geralmente relacionados ao tipo do jogo ou a plataforma a qual ele é destinado, sendo a qualidade do jogo dependente do quão apropriado o motor de jogo usado é para esses requisitos Gregory (2018). Exemplos de motores de jogos 3D são Unreal Engine, Godot ou Unity, que possuem diversos recursos que facilitam a utilização de modelos, simulação de física e efeitos utilizados na criação de um jogo 3D, tanto para Windows quanto para Linux. Porém, eles não possuem tantos recursos direcionados para criação de jogos 2D. Assim, para obter a maior qualidade possível ao criar um jogo 3D que será executados em computadores com sistema operacional *Windows*, deve-se utilizar um motor de jogo feito para a criação de jogos 3D para computadores, com suporte ao sistema operacional *Windows*.

2.3.2 Modelagem 3D

Modelagem 3D consiste na criação modelos de personagens, enfeites, ambientes e estruturas em 3 dimensões a partir de desenhos 2D (NOVAK, 2011). Esses modelos contêm as informações sobre o formato e aparência do objeto, além das animações e metadados usados pelo jogo e propriedades visuais, como cores e refletividade (GREGORY, 2018).

Os modelos 3D são geralmente criados usando pacotes de *software* de modelagem 3D (NOVAK, 2011; GREGORY, 2018), como Maya, 3DS Max ou Blender. Porém, antes de utilizar os modelos gerados por um pacote de modelagem, é necessário extrair os dados deles e armazená-los em um formato compatível com o motor de jogo no qual eles serão utilizados (GREGORY, 2018). Os pacotes de *software* de modelagem geralmente fornecem a opção de exportar para formatos semi-padrão, que podem ser utilizados com motores de jogos, mas que não são totalmente adequados para o desenvolvimento de jogos (GREGORY, 2018).

2.3.3 Documentação dos jogos

Não há um padrão de documentação rigoroso na indústria de Jogos, apenas guias gerais do que deve ser indicado na documentação (NOVAK, 2011). A documentação completa de um jogo costuma incluir etapas de conceito e proposta, que tem como objetivo vender a ideia do jogo e o GDD (*Game Design Document*), que tem como objetivo

ser o guia de referencia para o processo de desenvolvimento (NOVAK, 2011). O GDD apresenta a jogabilidade, história do jogo, personagens, a interface e regras do jogo, com detalhes suficientes para que seja possível joga-lo sem um computador (NOVAK, 2011). O GDD apresenta, assim, os detalhes necessários para guiar o desenvolvimento do jogo, apresentando todas as informações sobre o jogo, incluindo desde o contexto do jogo até o resultado final esperado.

2.4 Trabalhos relacionados

Existem diversos trabalhos sobre o uso de jogos (eletrônicos ou não) no ensino. Vários trabalhos apresentam foco no ensino de algoritmos, porém, grande parte deles não utilizam nenhuma técnica de ensino, sendo esperado que o aluno aprenda melhor apenas pelo fato do conteúdo ser inserido em um jogo.

Rajaravivarma (2005) propõe um modelo de ensino de programação utilizando jogos, onde um problema na forma de um jogo é apresentado aos alunos nos últimos 10 minutos de aula para o qual os alunos devem apresentar soluções algorítmicas na próxima aula. Os alunos descrevem os passos necessários para a execução do jogo e o processo para implementar a solução. Para a implementação do modelo, Rajaravivarma (2005) utiliza uma divisão processo de programação de um algoritmo em etapas, sendo elas: análise ou identificação do problema, projeto, implementação e teste, que devem ser diretamente introduzidas dentro do jogo. Para realizar esta tarefa é preciso escolher os problemas sobre conteúdo de algoritmos que se pretende ensinar, aplicá-los no jogo, de forma que o problema seja apresentado dentro dele e que o jogador possa produzir o algoritmo que o soluciona durante o jogo. Porém, a implementação da solução feita durante o jogo pode ficar fora de um ambiente de programação, pois o jogo não é necessariamente eletrônico no formato proposto por Rajaravivarma (2005). Assim, a fase de teste pode ser dificultada até o momento de a solução ser introduzida em um ambiente de programação.

Moreno (2010) apresenta a utilização dos recursos oferecidos pelos jogos digitais para levar experiencias da vida cotidiana para a sala de aula. O jogo proposto por ele visa ensinar conceitos básicos de programação, como sequenciamento de instruções, iteração, processos aninhados e a chamada de funções. Moreno (2010) utiliza no jogo um ciclo de aprendizagem composto pelas fases “julgamento – comportamento – realimentação”, que, de acordo com ele, traduz as características do jogo e o conteúdo instrucional incorporado em um novo conhecimento ou na melhora das habilidades. Para criar o ciclo, Moreno (2010) define seis grandes dimensões que caracterizam um jogo: Fantasia (criar um contexto imaginário), Regras e Objetivos (para que o jogador entenda seu progresso e receba uma realimentação adequada), Estímulos sensoriais (apresentar estímulos visuais e auditivos), Desafios (conter uma dificuldade adequada para incentivar o jogador a avançar), Mistério (apresentar a

informação ao estudante de forma que o mantenha interessado) e Controle (permitir um controle ativo do jogador). Utilizando características junto a conceitos de desenho de jogos, Moreno (2010) desenvolveu a ideia de um jogo onde o estudante controla um robô que participa em competições esportivas. Os movimentos do robô são determinados a partir de algoritmos que o jogador define. O jogo dá ao jogador a possibilidade de criar algoritmos utilizando um conjunto de instruções que o robô pode executar. Dessa forma, o jogo permite ao jogador internalizar os conceitos de programação sem utilizá-los diretamente em uma linguagem de programação. Após testar o jogo com alunos do cursos de engenharia, Moreno (2010) concluiu que o jogo trouxe efeitos positivos aos estudantes, principalmente no que diz respeito à motivação, sendo que vários dos alunos afirmaram que os jogos os incentivaram a esforçar-se e repensar seus modelos mentais.

Souza, Jaeger e Cardoso (2013) utilizaram jogos como alternativa para reduzir a dificuldade dos alunos dos cursos de computação na assimilação dos conteúdos envolvendo lógica. Eles afirmam que muitos alunos não conseguem elaborar a sequência lógica necessária para desenvolver um programa. Isso demonstraria, de acordo com eles, a ineficiência dos modelos atuais de ensino para o aprendizado. Os jogos seriam um meio capaz de estimular o aluno a resolver os problemas de forma lúdica e desafiadora, auxiliando a desenvolver as habilidades necessárias para construir algoritmos. Foi feita uma pesquisa quantitativa e descritiva com um grupo de acadêmicos do Centro de Educação Superior do Alto Vale do Itajaí – SC, onde foi feito um questionário com uma turma paralela de algoritmos para estudantes que não obtiveram êxito em tentativas anteriores de aprovação. Souza, Jaeger e Cardoso (2013) afirmam que o jogo facilita o ensino e aprendizagem de determinado assunto, principalmente por dar motivação. De acordo com eles, por envolver habilidades como destreza motora, a tomada de decisão e a persistência, o jogo leva a uma compreensão mais profunda do tema. Ainda de acordo com eles, no jogo são propostos desafios que exigem utilização da memória, raciocínio e atenção, podendo ser utilizados como meio de suporte na resolução de um exercício. Baseado nisso, Souza, Jaeger e Cardoso (2013) desenvolveram um jogo utilizando HTML, CSS e Javascript sobre o problema da travessia do rio e o aplicaram à turma analisada, avaliando posteriormente os impactos e benefícios da nova metodologia no aprendizado dos alunos. No jogo, o jogador deve auxiliar um fazendeiro a levar seus itens ao outro lado do rio, sendo que ele possui uma raposa, uma ovelha e uma alface e pode levar, a cada travessia, a si mesmo e mais um deles no barco. Se forem deixados sem o fazendeiro em uma mesma margem, a raposa come a ovelha e a ovelha come a alface. O objetivo é fazer a travessia de todos os elementos sem nenhuma perda. De acordo com Souza, Jaeger e Cardoso (2013), é necessária a construção de passos lógicos para resolver o problema, recebendo *feedback* a cada ação realizada. Após realizar a pesquisa com os alunos, Souza, Jaeger e Cardoso (2013) concluíram que a adoção de metodologias diferenciadas de ensino contribui para uma melhora no aprendizado dos acadêmicos.

Moser (1997), apresenta jogos como uma forma de facilitar o aprendizado de algoritmos. No artigo, ele afirma que a programação é um grupo de diversas habilidades em conjunto, sendo que várias delas são usadas ao mesmo tempo. De acordo com ele, essas habilidades são aprendidas na forma *bottom up* (de baixo para cima), aprendendo a sintaxe, a estrutura e então o estilo, o que possibilita que diversos mau hábitos se formem no processo de aprendizagem. Ainda de acordo com ele, os blocos de construção básicos da programação não são familiares aos alunos iniciais e os paralelos mais próximos que eles possuem são as disciplinas de matemática e línguas, que não se traduzem de forma equivalente para a programação. Moser (1997) afirma que a programação é aprendida em um único contexto, sendo planejadas por idade e não por habilidade e é considerada difícil por algumas pessoas. A solução proposta por Moser (1997) foi a criação de um jogo de fantasia para ensinar algoritmos, pois isso facilitaria a abstração e possibilitaria aprender conceitos de alto nível antes de partir para uma linguagem de programação. Para criar a estrutura de habilidades do jogo (organizar o conteúdo a ser aprendido), Moser (1997) afirma que é possível utilizar parcialmente sistemas para organizar interações em sala de aula no jogo para guiar o estudante através do processo de aprendizagem. Porém, de acordo com ele, isso reduz a sensação de envolvimento do jogador com o jogo por eliminar grande parte do desconhecido do jogo. Assim, Moser (1997) propôs um sistema escalonado, onde os pontos principais a serem aprendidos são colocados em cenas do enredo e agem como portões entre áreas livremente exploráveis. O jogo proposto por ele consiste em um mago que está inicialmente preso na torre de seu mestre e deve completar condições para avançar para a próxima área. No inicio, deve-se sair da torre, completando um requisito para um guardião deixado ali, depois o jogador pode explorar a cidade fora da torre, que possui um muro em sua volta e um portão guardado, onde o jogador deve completar as condições para passar para a próxima área. Esse ciclo continua durante o jogo, onde o jogador desbloqueia um local e para desbloquear o próximo deve completar o desafio daquele local.

Rapkiewicz et al. (2006) afirmam que a disciplina de algoritmos é importante nos cursos de computação e engenharia, mas diversos alunos apresentam problemas no decorrer da disciplina. Elas apontam como fator limitante o ensino instrucionista, que não deixa clara para os alunos a importância do conteúdo para sua formação. De acordo com Rapkiewicz et al. (2006), a forma como a disciplina é abordada por alguns docentes, utilizando apresentando um problema em formato textual e pedindo a solução em pseudocódigo, fluxogramas ou diagramas pode desmotivar os alunos, por adiar o contato com um recurso prático, como o computador. Elas propõe o uso de jogos numa fase introdutória do ensino para minimizar as dificuldades e promover a possibilidade do aluno construir o conhecimento por meio de ações concretas, experimentações e reflexões. Rapkiewicz et al. (2006) apresentam estratégias para serem usadas nos jogos para permitir ao professor conduzir o aluno a construir sua própria solução e representá-la na forma

de algoritmos. São elas: Problematização, por meio de um problema contextualizado; Representação lógica e construção de hipóteses; Leitura e teste dos modelos; Cooperação para a construção coletiva de uma solução e Construção de uma representação na forma de algoritmo. Elas propõe que o problema seja apresentado de forma visual através do jogo (Problematização), sendo solucionado ali e descrito no quadro pelo aluno de forma livre (Representação lógica e construção de hipótese), após isso os alunos trocam suas soluções entre si, lendo e tentando entender a solução do outro (Leitura e teste), depois é feita a construção de uma solução coletiva entre os alunos (Cooperação) e por fim é feita a representação em forma de algoritmo (Construção). Como exemplo, Rapkiewicz et al. (2006) apresentam o jogo MAGU, onde o jogador deve colocar a quantidade necessária de poção (6 litros) utilizando dois vasos com quantidades diferentes (5 e 7 litros). De acordo com elas, o jogo permite observar as ações de forma concreta e cria a expectativa de resultado, instigando o levantamento de hipóteses e sua testagem. Elas afirmam que é necessário que o professor haja como mediador, guiando o grupo de alunos para evitar que o jogo seja apenas uma divertimento já que ele é muito simples. Rapkiewicz et al. (2006) concluem dizendo que um dos aspectos fundamentais na utilização dos jogos é o fator motivador.

3 Desenvolvimento do aplicativo proposto

Neste capítulo, será detalhado o desenvolvimento do jogo proposto, incluindo as tecnologias utilizadas, compilação em tempo de execução e a descrição do jogo.

3.1 Tecnologias utilizadas

Nesta seção são descritas as tecnologias usadas para o desenvolvimento do aplicativo.

3.1.1 Unity 3D

A Unity é um motor de jogo e uma plataforma para criação e operação de conteúdo 3D interativo e em tempo real, providenciando ferramentas para criação de jogos para diversas plataformas (Unity Technologies, 2021). A Unity permite uma divisão entre os recursos, que consistem em imagens, modelos 3D, áudio, entre outros, utilizados no jogo e o código do jogo, que consiste em *scripts* que definem o comportamento desses recursos no jogo. Os *scripts* foram escritos em C#, que é a linguagem de programação padrão usada com a Unity, e que permite compilação de código em tempo de execução. Os jogos desenvolvidos em Unity são executados usando o Mono, que é uma implementação *open source* do .NET Framework.

3.1.2 Blender

O Blender é uma suíte livre e *open source* de criação 3D, que pode ser usada na criação de modelagem, manipulação, simulação, renderização, composição e rastreamento de movimentos, além de edição de videoclipes e animações 2D (Blender Foundation, 2021). O Blender foi utilizado na criação de modelos 3D utilizados no jogo.

3.1.3 MCS

O compilador para C# do Mono (MCS) é um compilador para C# escrito em C#, com o objetivo de fornecer uma alternativa livre de implementação da linguagem C# (ICAZA, 2009). O MCS é utilizado para realizar a compilação de código dentro do jogo, através de uma versão da biblioteca Mono.CSharp.

3.2 Compilação em tempo de execução

A compilação dentro do jogo ocorre quando o jogador clica no botão "COMPILAR", localizado no canto inferior direito da tela de entrada de instruções (mostrada na Figura 1).

O processo de compilação permite ao usuário compilar um código simples, sem modificar nenhum objeto no jogo. Mas o jogo permite também que o jogador selecione e modifique alguns objetos do jogo (chamados objetos selecionáveis). Os objetos são selecionados com um clique do mouse com o botão esquerdo sobre o objeto, sendo que o último objeto selecionável clicado ficará selecionado até o próximo objeto selecionável ser clicado. Por isso, não há uma forma de remover o objeto selecionado, sendo possível apenas trocá-lo por outro objeto. Após um objeto ser selecionado, o jogador poderá utilizar o nome do objeto selecionado para alterar o objeto, indicado no canto superior esquerdo da tela de entrada de instruções (como mostrado na Figura 1 e na Figura 2), para saber qual objeto será afetado. O nome do objeto é usado também como referência para alterar o objeto usando código.

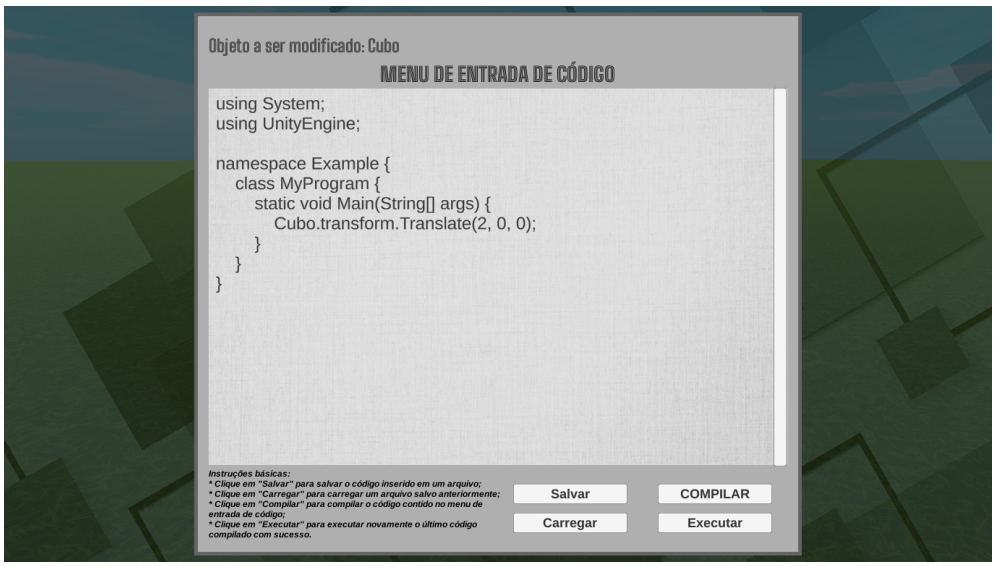
Para facilitar a utilização do menu de entrada de código, foram criadas simplificações para os comandos relacionados à Unity. As simplificações tem objetivo de facilitar o uso dos comandos no jogo e foram feitas para cinco comandos relacionados às transformações geométricas do *transform* do *GameObject*. Três deles são para obter as propriedades posição, rotação e escala (*position*, *rotation* e *localScale* na Unity) e dois deles para utilizar as funções de translação e rotação (*Translate* e *Rotate* na Unity). As simplificações feitas são indicadas abaixo:

- POSITION: atalho para nome_objeto.transform.position;
- ROTATION: atalho para nome_objeto.transform.rotation;
- SCALE: atalho para nome_objeto.transform.localScale;
- TRANSLATE: atalho para nome_objeto.transform.Translate;
- ROTATE: atalho para nome_objeto.transform.Rotate.

Assim, caso o jogador tenha selecionado o objeto de nome "Cubo"e deseje modificar sua posição em duas unidades no eixo x, ele poderia fazer isso de duas formas. A primeira seria usando a referência existente no jogo (o nome do objeto, "Cubo") para usar o comando de transladar da Unity e a segunda seria usando a simplificação fornecida. Abaixo é demonstrado como ficaria o código inserido usando cada uma das formas.

- Usando o nome do objeto como referência:

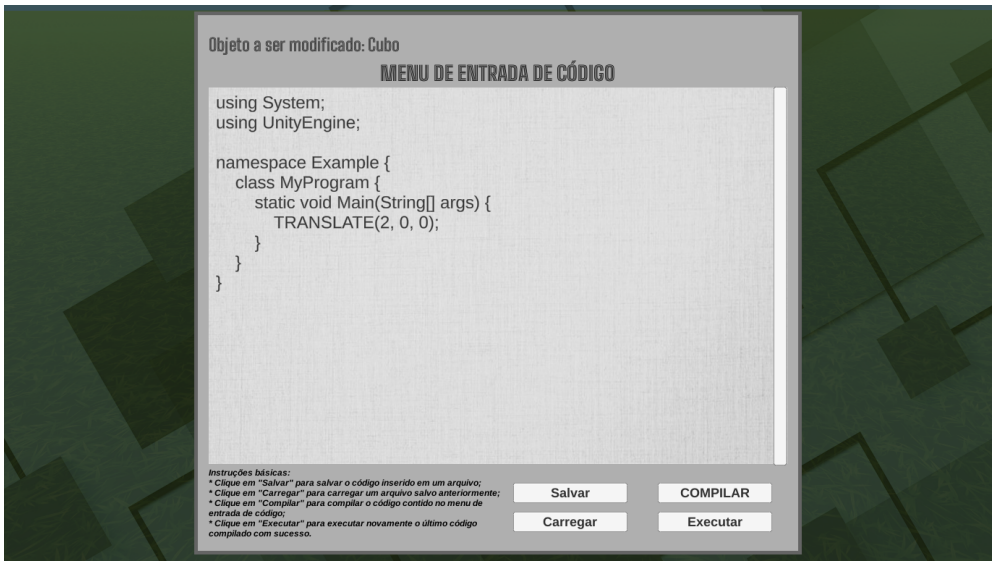
Figura 1 – Usando a referência: Cubo.transform.Translate(2, 0, 0).



Fonte: Própria, 2021

- Usando a simplificação, sem necessidade de usar nome do objeto:

Figura 2 – Usando a simplificação: TRANSLATE(2, 0, 0).



Fonte: Própria, 2021

Para compilar o código fornecido na linguagem C# dentro do jogo foi utilizado o MCS, usando a opção de geração em memória para que o código fosse compilado em uma DLL (Dynamic Link Library). As DLLs funcionam como bibliotecas do C#, sendo que as classes, funções, entre outros existentes nela podem ser utilizados em um programa referenciando essa DLL. Assim, após a compilação, pode-se executar o código dentro do jogo com o uso de reflexões (biblioteca System.Reflection). Para a utilização do compilador

MCS dentro do jogo, foi utilizada a biblioteca Mono.CSharp, fornecida pelo Mono. Porém, para que a utilização do MCS fosse possível dentro do jogo, foi necessário o uso de uma versão modificada dessa biblioteca. As modificações à biblioteca consistem em mudanças na forma em que o jogo busca o compilador MCS. A versão modificada busca o arquivo binário do compilador dentro da pasta do jogo, enquanto a versão original busca no sistema, o que faz com que o MCS (representado pelo arquivo binário mcs) não seja encontrado.

No caso do jogo, foram usadas reflexões para buscar por uma classe, dentro da DLL gerada, que contenha método com nome *Main*, cujo código será executado. Assim, o código fornecido pelo usuário deve conter uma única classe com um método *Main* nela, podendo também conter outros métodos, que só serão executados se foram usados no método *Main*.

3.3 Descrição do jogo

O jogo foi organizado na forma de "conceitos", que são os conteúdos trabalhados na disciplina. Os conceitos são trabalhos dentro do jogo na forma de problemas, divididos em *puzzles*, onde são apresentados os conceitos ao jogador e situações-problemas, onde os conceitos são formalizados usando a linguagem de programação. As situações-problemas e os *puzzles* são diferenciados pelo fato de que é necessário usar a linguagem de programação para resolver situações-problemas, mas não *puzzles*. Porém, os *puzzles* e as situações-problemas não são diferenciados dentro do jogo, sendo ambos problemas propostos para o jogador resolver.

Dentro do jogo o jogador aplica os conceitos em objetos comuns, como cubos, esferas, paredes, entre outros para resolver os problemas. Isso possibilita que o jogador compreenda como esses conceitos e objetos se relacionam e que o jogador teste e compreenda os processos do conceito a ser aprendido. Essa forma de apresentar o conceito é compatível com o construcionismo de (PAPERT, 1980), pois o jogador poderá relacionar os conceitos que ele pretende aprender com objetos que ele conhece, podendo testar os processos relacionados ao conceito através de testes com os objetos.

Na documentação do jogo, ele é dividido em vilas, sendo que cada vila apresenta os conceitos relacionados ao conteúdo de uma unidade da disciplina. Porém, os *puzzles* apresentados em uma vila não necessariamente apresentarão apenas os conceitos relacionados à unidade explorada nela, pois o jogador poderá utilizar na resolução dos *puzzles* conceitos que ainda não consegue usar em uma linguagem de programação.

3.3.1 Puzzles

Puzzles são resolvidos interagindo diretamente com os objetos do jogo, sem uso direto de uma linguagem de programação. Os puzzles são divididos em 3 tipos:

- puzzle do tipo A (puzzle de controlar personagem): Neste tipo de *puzzle*, o jogador pode controlar um personagem que reconhece certos comandos definidos pela situação apresentada pelo puzzle;
- puzzle do tipo B (puzzle de mover objetos): Neste tipo de *puzzle*, um ou mais objetos estão espalhados pelo terreno e o jogador deverá utilizá-los para cumprir com o que a situação do puzzle exige;
- puzzle do tipo C (puzzle envolvendo recipientes): Neste tipo de *puzzle*, o jogador deve fazer trocas entre recipientes para atingir o objetivo.

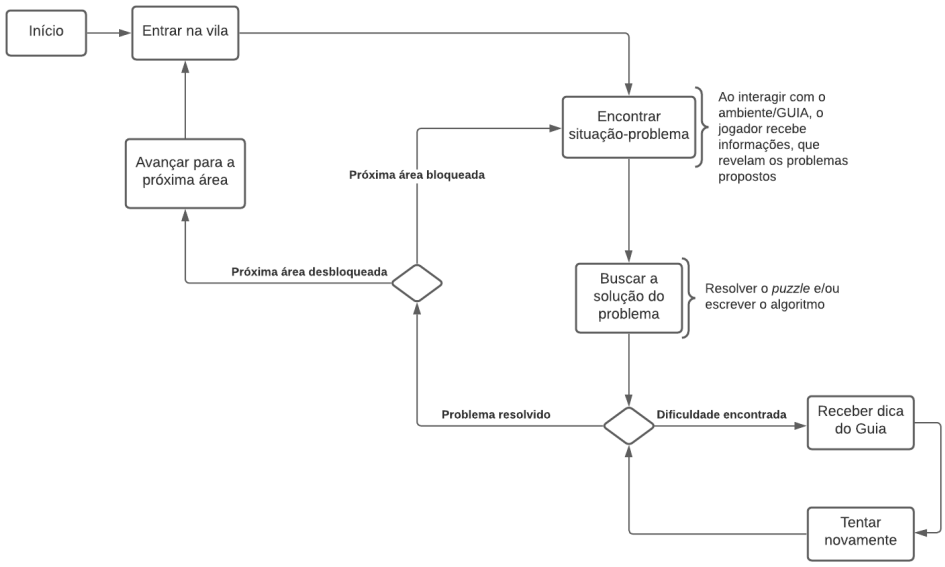
3.3.2 Situações-Problemas

As situações-problemas serão apresentadas no jogo pelo GUIA do jogo ou por algum habitante da vila com um contorno avermelhado, assim como os puzzles. Diferentemente dos puzzles, elas serão resolvidas usando uma linguagem de programação (C#) para modificar os objetos do jogo e seu comportamento ou adicionar novos objetos ao jogo. Todas as situações-problemas serão baseadas nos problemas apresentados na disciplina de Algoritmos I, com objetivo de demonstrar a utilização dos algoritmos usados nas soluções desses problemas de mais de uma forma aos alunos. As situações-problemas poderão aparecer junto à puzzles, para modificar um comportamento dos objetos que fazem parte do puzzle para que seja possível solucioná-lo ou aparecer por si só.

3.3.3 Fluxo do jogo

O fluxo do jogo (*gameflow*) é simples. O objetivo principal do jogador é interagir com o conteúdo do jogo, incluindo o GUIA e objetos presentes no jogo. Através dessa interação com o jogo, o jogador encontra situações em que não pode avançar pois o caminho está bloqueado ou em que há um desafio que ele deve completar para avançar. Nessas situações, o jogador deve resolver os *puzzles* ou situações-problemas para desbloquear a próxima área ou completar o desafio, o que permitiria descobrir novos bloqueios no caminho ou desafios. Uma representação simplificada desse ciclo é apresentada na Figura 3:

Figura 3 – Diagrama representando o fluxo do jogo, indicando o ciclo básico das atividades realizadas pelo jogador.



Fonte: Própria, 2021

O jogador inicia na primeira vila, a partir dela interagindo com o cenário e encontrando as problemas (que podem ser *puzzles* ou situações-problemas). Após isso, o jogador busca a solução para o problema e, se encontrar a solução, busca novos problemas para resolver, até que desbloqueie a próxima área. Durante o processo de resolução dos problemas, o jogador poderá pedir ajuda para o GUIA, que apresentará dicas para a solução do problema. Essas dicas são indicações gerais sobre as dúvidas mais comuns relacionadas ao problema que o jogador está resolvendo.

3.3.4 Problemas propostos no jogo

No sistema desenvolvido, foram implementados os *puzzles* e a situação-problema do primeiro nível do jogo (Vila I), que correspondem ao conteúdo da primeira unidade da disciplina de Algoritmos.

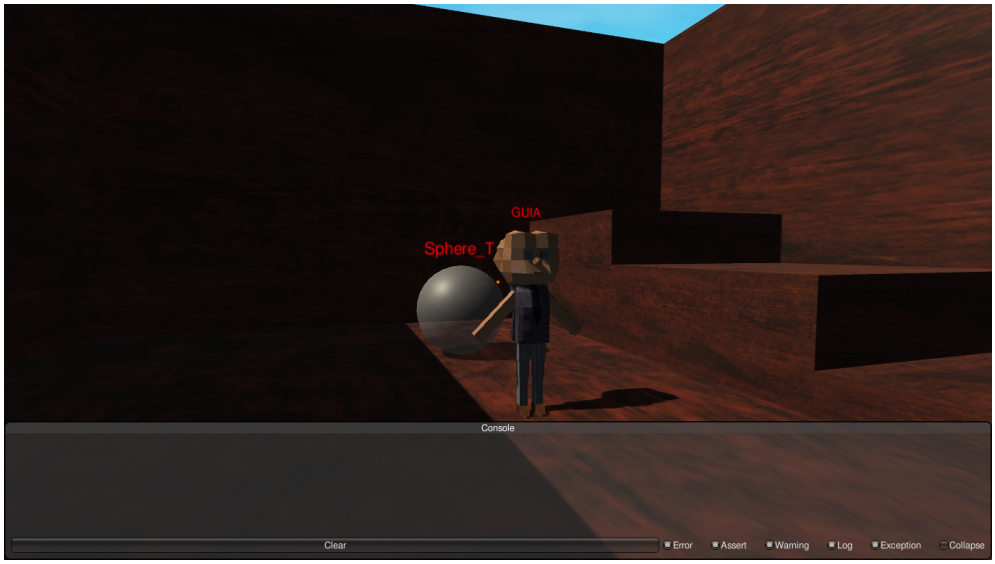
Aqui serão definidos os problemas propostos no sistema desenvolvido, em ordem de aparecimento no jogo, incluindo sua descrição, os conceitos apresentados nele e quais elementos o compõe.

3.3.4.1 Problemas Iniciais

Os problemas iniciais apresentam os conceitos iniciais, sendo divididos em três problemas, cada um indicando uma das partes básicas apresentadas na disciplina.

O primeiro problema é o Inicial I, que começa com o GUIA apresentando o ambiente do jogo ao jogador, indicando o objetivo do jogo e seus elementos principais. No fim da explicação, o GUIA pede para o jogador mover uma esfera até a parte de cima das escadas existentes no local. Após o jogador mover a esfera, o caminho para o próximo problema se abrirá. Nessa parte não é utilizado nenhum conceito relacionado à algoritmos em específico, pois ela tem como objetivo apenas apresentar o jogo.

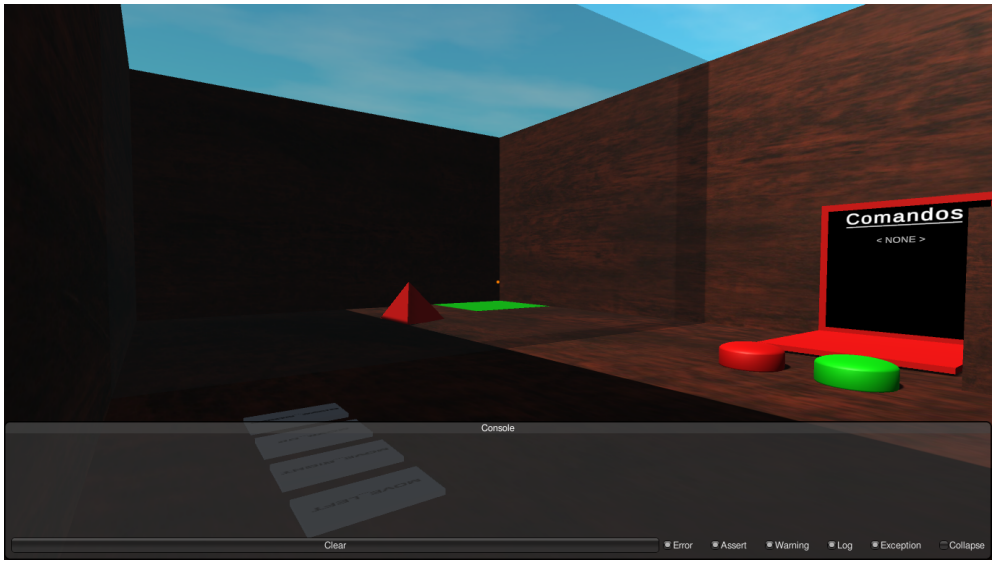
Figura 4 – Puzzle inicial I.



Fonte: Própria, 2021

O segundo problema é Inicial II, que começa após o jogador entrar no local do *puzzle*. Logo após o jogador entrar no local, um diálogo aparecerá, indicando ao jogador que ele deve utilizar as instruções existentes no local para mover um tetraedro, marcado na cor vermelha, até o objetivo, marcado na cor verde. Os comandos possíveis são MOVE_RIGHT, MOVE_LEFT, MOVE_UP e MOVE_DOWN, que fazem o tetraedro mover-se, respectivamente, para a direita, para a esquerda, para cima e para baixo. O tetraedro e o objetivo são inacessíveis ao jogador, existindo uma barreira transparente entre eles e a área do *puzzle*, como indicado na Figura 5. Após a finalização do *puzzle*, a barreira transparente e a parede atrás dela são removidas, permitindo ao jogador acessar o próximo problema. Essa parte tem como objetivo apresentar os conceitos relacionados à utilização de uma sequência de comandos para atingir um objetivo e ao uso de diferentes representações de algoritmos.

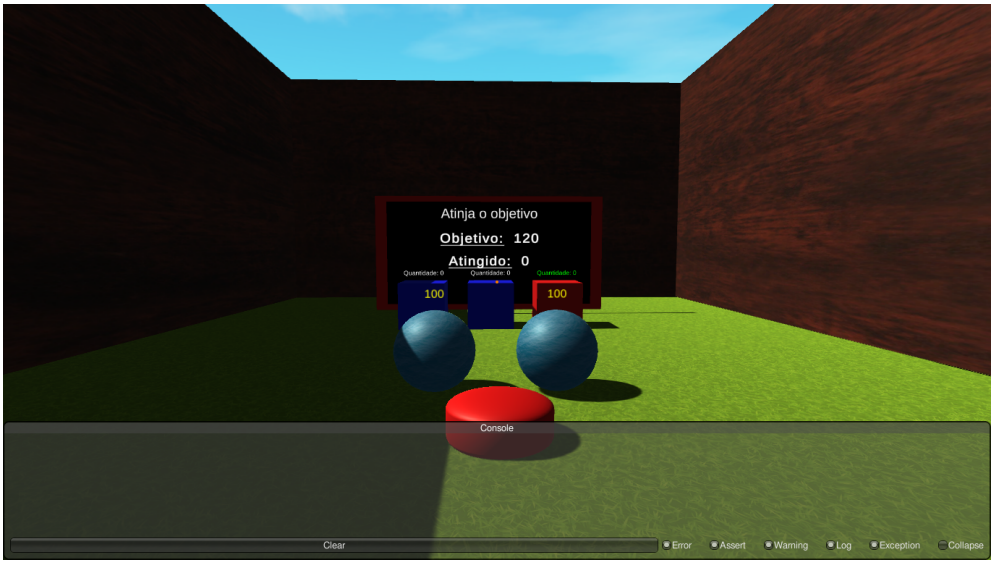
Figura 5 – Puzzle inicial II.



Fonte: Própria, 2021

O último problema inicial é o Inicial III, que começa após o jogador entrar no local do *puzzle*. Logo após o puzzle começar, um diálogo aparecerá, indicando ao jogador que ele deve utilizar os recipientes de cor azul existentes no local para fazer quantidade de material no recipiente de cor vermelha ser igual à indicada como objetivo no placar (Figura 6). É indicado também que o material é representado pelas duas esferas, que contém uma quantidade fixa dele. Depois, é indicado que o material deve ser movido entre os recipientes para atingir o objetivo e que o *puzzle* pode ser reiniciado pressionando o botão vermelho. Os itens do *puzzle* são indicados na Figura 6. Após a finalização do *puzzle*, a parede atrás do placar será removida, permitindo ao jogador acessar o próximo problema. Esse *puzzle* tem como objetivo apresentar os conceitos relacionados à variáveis e constantes.

Figura 6 – Puzzle inicial III.



Fonte: Própria, 2021

3.3.4.2 Problemas de trocas de operações

Os problemas de trocas de operações apresentam os conceitos relacionados às operações aritméticas, operações lógicas e operações de comparação em linguagens de programação.

O primeiro problema é o de Troca de Operações Aritméticas. Esse problema inicia mostrando ao jogador, logo após ele entrar no local do *puzzle*, um diálogo indicando que ele deve alterar as operações aritméticas existentes em duas expressões, para que o resultado de uma terceira expressão seja correto. Nesse *puzzle*, uma barreira transparente impede a passagem do jogador (Figura 7). Essa barreira barreira é removida após a conclusão do *puzzle*, permitindo a passagem para o próximo problema. Esse *puzzle* tem como objetivo utilizar os conceitos relacionados à operações aritméticas.

Figura 7 – Troca de Operações.

Fonte: Própria, 2021

O segundo problema é o de Troca de Operações Lógicas, que começa após o jogador entrar no local do *puzzle*. Nesse momento, um diálogo aparecerá, indicando ao jogador que ele deve alterar os valores lógicos existentes em duas expressões lógicas para que o resultado de uma terceira expressão seja correto. De forma semelhante ao problema anterior, uma barreira transparente impede a passagem do jogador (como na Figura 7). Essa barreira barreira é removida após a conclusão do *puzzle*, permitindo a passagem para o próximo problema. Esse *puzzle* tem como objetivo utilizar os conceitos relacionados à operações lógicas.

O último problema é do de Troca de Operações de Comparação, que inicia quando o jogador acessa o local do *puzzle*. Nesse momento, um diálogo aparecerá, indicando ao jogador que ele deve alterar as operações de comparação existentes em duas expressões para que o resultado de uma terceira expressão seja correto. Nesse *puzzle*, uma parede impede a passagem do jogador. Essa parede é removida após a conclusão do *puzzle*, permitindo a passagem para o próximo problema. Esse *puzzle* tem como objetivo utilizar os conceitos relacionados à operações de comparação.

3.3.4.3 Problema final

O problema final (Portão) inicia após o jogador entrar no local da situação-problema, onde um diálogo aparecerá, indicando que ele deverá resolver o problema usando uma linguagem de programação. Após isso, são dadas as explicações básicas sobre as simplificações usadas no jogo dos comandos da Unity.

Após isso, é indicado que o jogador deve utilizar os comandos para mover um

objeto do jogo, com nome Portão (indicado na Figura 8) para permitir sua passagem.

Após o jogador conseguir mover o portão, ele poderá passar pelo local, concluindo o problema. Após a conclusão do problema, aparecerá a tela de conclusão do jogo (Figura 17)

Essa situação-problema tem como objetivo apresentar a utilização dos conceitos aprendidos nos outros problemas em uma linguagem de programação.

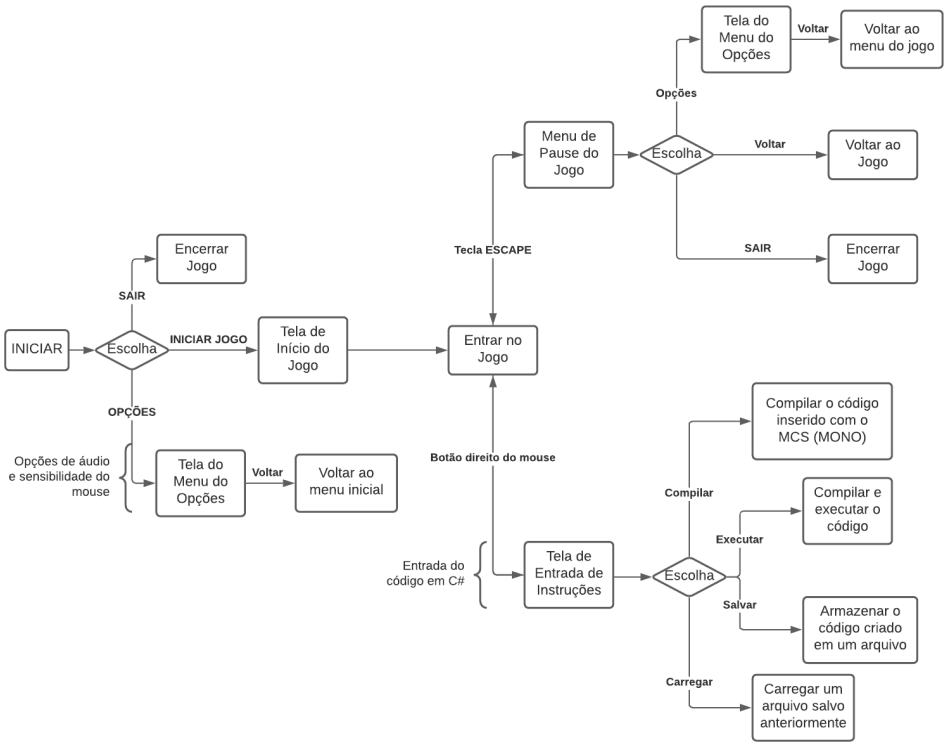
Figura 8 – Situação-problema final

Fonte: Própria, 2021

3.3.5 Interface do Jogo

A interface do jogo tem como principal objetivo ser simples e objetiva. O jogador deve ser capaz de navegar com facilidade pelos menus que fazem parte da interface, recebendo as informações básicas do jogo através dela. A hierarquia da interface do jogo é apresentada na Figura 9:

Figura 9 – Hierarquia da interface, a partir do menu inicial do jogo.



Fonte: Própria, 2021

A interface do jogo pode ser dividida entre menus, telas de jogo e diálogos.

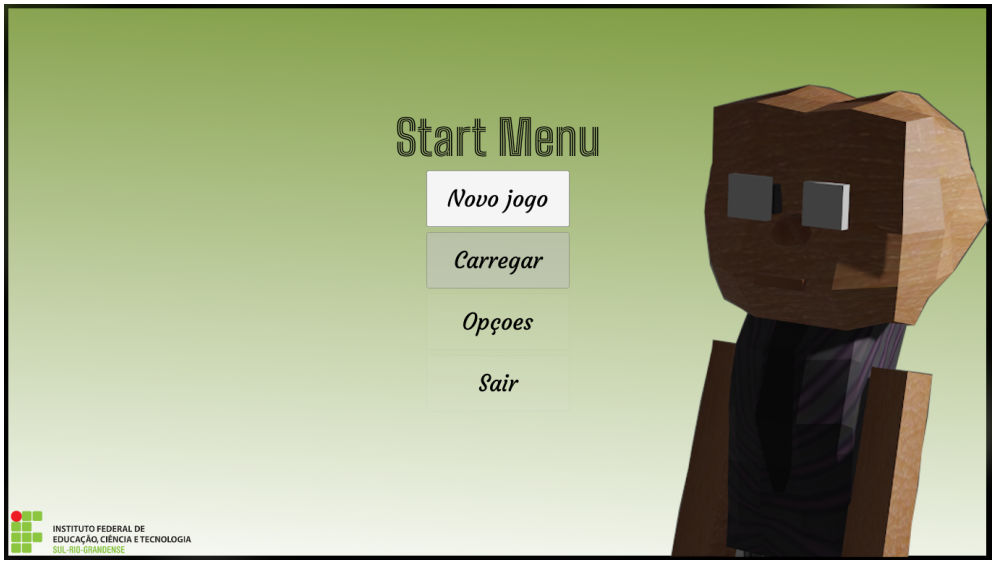
3.3.5.1 Menus do jogo

Os menus apresentados no jogo são o menu inicial do jogo, o menu de pause do jogo e o menu de opções do jogo.

O menu inicial do jogo (menu iniciar) é a primeira tela do jogo. É a partir dessa tela que o jogador acessa todas as áreas seguintes do jogo, como indicado no diagrama da Figura 9. Essa tela é composta basicamente por quatro botões, sendo eles: Novo jogo, Carregar, Opções e Sair.

O botão Novo jogo inicia o jogo a partir do início, sendo atualmente a única forma de acessar o jogo. O botão de opções permite acesso à tela de opções (Figura 12). O botão Sair encerra o jogo. Por fim, o botão Carregar está desativado.

Figura 10 – Menu inicial do jogo.

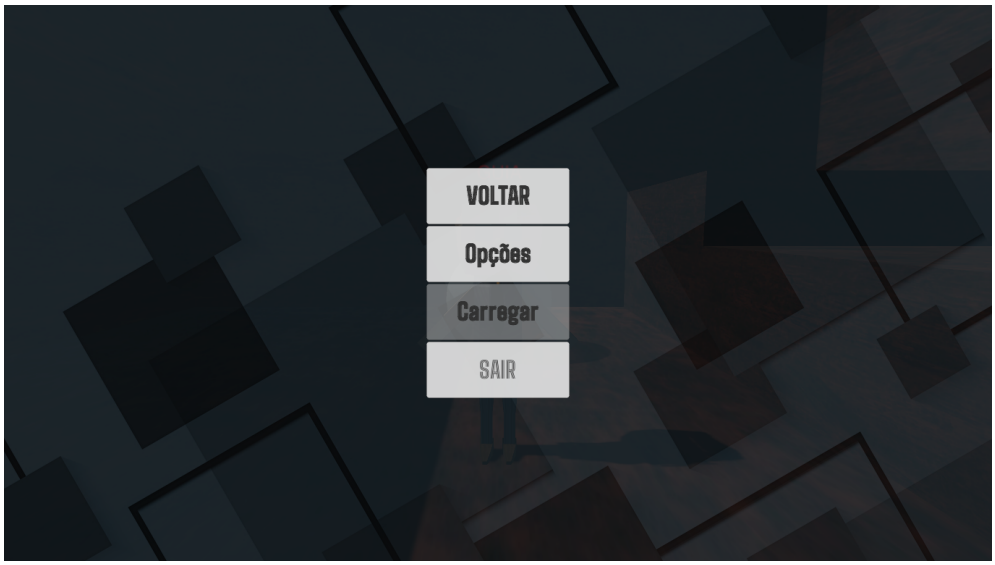


Fonte: Própria, 2021

O menu de pause do jogo é a tela mostrada ao pausar o jogo após iniciar o jogo (interface mostrada na Figura 13). Ela é composta por quatro botões: VOLTAR, Opções, Carregar e SAIR.

O botão VOLTAR volta ao jogo, "despausando"o jogo. O botão de opções permite acesso à tela de opções (Figura 12). O botão SAIR encerra o jogo. Por fim, o botão Carregar está desativado.

Figura 11 – Menu de pause do jogo.



Fonte: Própria, 2021

O menu de opções é a tela com as configurações do jogo, resumidas a áudio e

sensibilidade do mouse. Ela é composta por dois controles deslizantes e dois botões.

Os dois controles deslizantes são o controle de Volume, que controla o nível de volume do jogo e o controle de Sensibilidade do mouse, que controla o nível de sensibilidade do mouse. Os valores nesses controles são salvos ao encerrar o jogo, sendo que quando não há nenhum valor definido, o jogo usa como valor padrão metade do nível máximo.

Os dois botões presentes são o botão de Carregar, que está desativado e o botão de Voltar, que retorna à tela anterior, que, conforme a Figura 9, pode ser o menu iniciar ou o menu de pause.

Figura 12 – Menu de opções do jogo.

Fonte: Própria, 2021

3.3.5.2 Telas de jogo

As telas de jogo são a tela de início do jogo, a tela de entrada de instruções e a tela de conclusão do jogo.

A tela de início do jogo é acessada ao iniciar o jogo, sendo a interface básica durante o jogo. Ela contém um console com *logs* relacionados à acontecimentos dentro do jogo, principalmente erros e avisos. No console também são indicados os resultados da compilação e execução de código realizadas na tela de entrada de instruções (Figura 14). A partir dessa tela é possível acessar o menu de pause e a tela de entrada de instruções, além da tela de conclusão do jogo ao terminar o último *puzzle*.

Figura 13 – Tela inicial do jogo.

Fonte: Própria, 2021

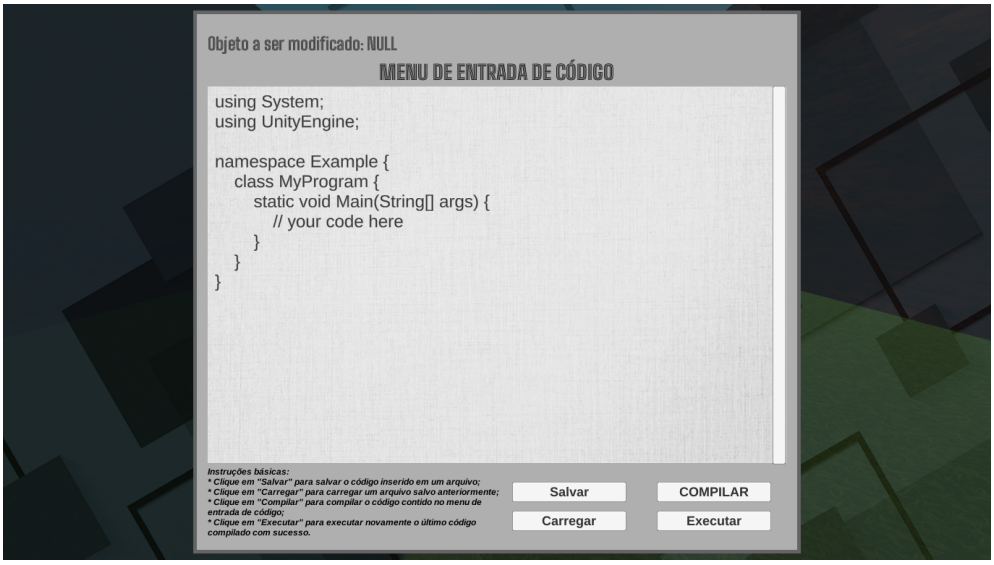
A tela de entrada de instruções é onde é feita a entrada do código em C#. Nessa tela são indicadas as informações básicas relacionadas à entrada de código, incluindo o objeto selecionado atualmente, indicado como "Objeto a ser modificado"e instruções sobre as ações que cada um dos botões realiza.

Através dos botões o jogador tem a possibilidade de escolher o que fazer com o código inserido, incluindo salvar, carregar, compilar e executar o código.

Para salvar o código inserido é usado o botão SALVAR, que abre um painel que permite a entrada do nome do arquivo em que o código será salvo e permite confirmar ou cancelar a operação. Para carregar um código salvo é usado o botão Carregar, que abre um painel que permite selecionar um dos arquivos salvos e permite confirmar ou cancelar a operação.

A compilação é feita usando o botão de compilar, que apresentará o resultado da compilação realizada no console de *logs*, indicado na Figura 13. Caso o código tenha sido compilado com sucesso, é possível executá-lo usando o botão Executar.

Figura 14 – Tela de entrada de instruções do jogo.



Fonte: Própria, 2021

A tela de conclusão do jogo é uma tela simples que indica o fim do jogo e aparece após o último problema do jogo ser concluído. Ela possui apenas um botão, SAIR, que tem a função de encerrar o jogo.

Figura 15 – Tela de conclusão do jogo.

Fonte: Própria, 2021

3.3.5.3 Telas de diálogo

As telas de diálogo são telas que indicam mensagens ou instruções dentro do jogo. Elas são divididas em dois tipos: Tipo 1 e Tipo 2.

As telas de diálogo do Tipo 1 ocorrem enquanto o jogador interage com um objeto ou se aproxima de um objeto que tenha diálogo no jogo, não interrompendo o jogador. Elas são usadas durante o jogo para indicar mensagens relacionadas à ações que o jogador pode realizar alguns objetos do jogo.

Figura 16 – Diálogo do tipo 1, ao aproximar-se do GUIA.

Fonte: Própria, 2021

Já as telas do Tipo 2 ocorrem quando há uma explicação longa a ser feita ao jogador, interrompendo temporariamente o jogador, para que ele leia a mensagem. Elas são usadas no jogo para realizar as explicações dos problemas no jogo e apresentar as ajudas relacionadas à eles.

Figura 17 – Diálogo do tipo 2, ao iniciar o primeiro problema.

Fonte: Própria, 2021

4 Validação do Sistema

O jogo foi avaliado através de uma pesquisa qualitativa realizada em duas etapas com os alunos. Primeiro, alguns alunos voluntários do curso de Algoritmos I do curso de Ciência da Computação responderam a um questionário (indicado no Apêndice A) antes de terem qualquer contato com o jogo. Após isso, eles foram convidados a testar o jogo e responder a outro questionário (indicado no Apêndice B) realizado após o contato com o jogo.

Na primeira etapa os jogadores foram questionados sobre suas expectativas em relação à jogos de aprendizagem e sobre sua experiencia anterior com estes, além do nível de conhecimento sobre a disciplina de algoritmos que o aluno considera ter. O objetivo desse questionário foi avaliar o perfil inicial dos alunos. Na segunda etapa os alunos foram questionados em relação à usabilidade e apresentação inicial dos conceitos no jogo, além da opinião que o aluno teve do jogo. O objetivo desse questionário foi avaliar se o jogo tem uma interface amigável e apresenta o conteúdo de forma adequada.

O questionário realizado antes do contato com o jogo foi respondido por 12 alunos, o que representa cerca de metade dos alunos cadastrados na disciplina (cerca de trinta alunos). As respostas desses alunos mostraram que mais da metade deles (58,4%) já tiveram contato com jogos de aprendizagem e que a maioria dos alunos considera seu conhecimento de algoritmos regular (41,7%) ou bom (25%). As respostas mostraram também que a maioria dos estudantes (58,3%) esperam que na aprendizagem com jogos seja possível aprender um conteúdo de forma interessante ou divertida, sendo que vários destes esperam que um jogo de aprendizagem ensine de uma forma diferente.

O questionário realizado após o contato com o jogo foi respondido por 6 alunos, que representa metade (50%) dos alunos que responderam ao primeiro questionário. As respostas dos alunos indicaram achar interessante a proposta, apesar do jogo precisar de aprimoramentos. Os usuários avaliaram a usabilidade do jogo com uma nota média de 7 e a clareza das instruções com uma nota média de 7,5.

Cinco alunos (83,3% do total) indicaram acreditar que o jogo colaborou de alguma forma para seu aprendizado por apresentar características de algoritmos de forma diferente e prática dentro do jogo, e 1 aluno (16,7% do total) indicou acreditar que o jogo não colaborou por uma falha impedi-lo de avançar no jogo.

Quatro alunos (66,7% do total) indicaram que recomendariam o jogo a um amigo que estivesse aprendendo algoritmos, enquanto 2 alunos (33,3%) do total indicaram que talvez recomendassem o jogo, pois ele possui problemas que podem impedir de avançar no jogo e pode deixar confusos alunos que ainda não tiveram contato com algoritmos.

Com base nos resultados do primeiro questionário, pode-se confirmar que os alunos da disciplina de algoritmos I esperam de um jogo de aprendizado que ele seja divertido e apresente o "conteúdo"de forma diferente. Já o segundo questionário mostrou que o jogo possui problemas que podem impedir o jogador de avançar no jogo, tanto nas explicações como na usabilidade do jogo. Apesar disso, o segundo questionário mostrou também que os alunos têm interesse pela ideia de aprender através de um jogo, por apresentar o conteúdo de forma diferente e permitir a utilização dos conceitos aprendidos dentro do jogo de forma prática na resolução dos problemas.

5 CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo o desenvolvimento de um jogo para o ensino de algoritmos, com base na teoria construcionista de Papert. Na criação dos problemas durante o jogo, foi possível apresentar conceitos relacionados ao conteúdo apresentado na disciplina de Algoritmos I do IFSUL, campus Passo fundo, representando-os de diferentes formas. Para a criação do jogo foram pesquisados métodos para realização das atividades necessárias dentro do jogo, incluindo a compilação em tempo de execução e apresentação do conteúdo dentro do jogo.

Durante o desenvolvimento do jogo, o sistema inicialmente proposto foi modificado por conter muitos elementos. Por isso, as funções de salvar e carregar o estado jogo foram desabilitadas no jogo e a interação com personagens no jogo foi reduzida à interação com o GUIA.

Com os questionários foi possível avaliar o perfil dos alunos da disciplina de Algoritmos I e verificar que o jogo proposto demonstrou-se como uma forma diferente de ensino que pode colaborar com a aprendizagem dos alunos, desde que as falhas que o jogo apresentou sejam corrigidas e as explicações melhoradas.

Com isso, pode-se afirmar que os objetivos do trabalho foram alcançados, pois o jogo desenvolvido tem a capacidade de colaborar na aprendizagem de algoritmos, apesar de ter apresentado falhas e as explicações não terem sido claras o suficiente para algumas situações.

Assim, em trabalhos futuros, as falhas que o jogo apresentou devem ser corrigidas e as explicações devem apresentar mais detalhes sobre os problemas, além de adicionar a possibilidade de modificar o conteúdo do jogo, possibilitando a adição de novos problemas e desafios dentro do jogo para tornar o jogo mais efetivo e flexível como uma ferramenta de aprendizagem.

Referências

ABT, C. C. *Serious Games*. [S.l.]: University Press of America, Lanham, MD, USA, 2002. ISBN 0819161489. Citado na página 12.

ARAUJO, R. et al. *Referenciais de Formação para os Cursos de Graduação em Computação 2017*. Porto Alegre: Sociedade Brasileira de Computação (SBC), 2017. 153 p. ISBN 978-85-7669-424-3. Citado 2 vezes nas páginas 9 e 11.

BATTISTELLA, P.; WANGENHEIM, C. G. von. Games for teaching computing in higher education–a systematic review. *IEEE Technology and Engineering Education*, v. 9, n. 1, p. 8–30, 2016. Citado na página 12.

Blender Foundation. *blender.org - Home of the Blender project - Free and Open 3D Creation Software*. 2021. Disponível em: <https://www.blender.org/>. Acesso em: 20 fev 2021. Citado na página 19.

BRASIL, M. d. E. *Projeto Politico Pedagógico do Bacharelado em Ciência da Computação*. Passo Fundo: IFSUL, 2017. 39 p. Citado na página 11.

CORMEN, T. H. et al. *Algoritmos: teoria e prática*. [S.l.]: ELSEVIER EDITORA, 2012. ISBN 8535236996. Citado na página 11.

CURRICULA, A. f. C. M. A. Joint Task Force on C.; SOCIETY, I. C. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. New York, NY, USA: Association for Computing Machinery, 2013. ISBN 9781450323093. Citado 2 vezes nas páginas 9 e 11.

FREITAS, S. D. Are games effective learning tools? a review of educational games. *Journal of Educational Technology & Society*, JSTOR, v. 21, n. 2, p. 74–84, 2018. Citado na página 12.

GREGORY, J. *Game engine architecture*. 3. ed. [S.l.]: crc Press, 2018. 1240 p. Citado na página 14.

GUNTER, G. A.; KENNY, R. F.; VICK, E. H. Taking educational games seriously: using the retain model to design endogenous fantasy into standalone educational games. *Educational technology research and Development*, Springer, v. 56, n. 5-6, p. 511–537, 2008. Citado 2 vezes nas páginas 9 e 12.

ICAZA, M. de. *The Internals of the Mono C# Compiler*. 2009. Disponível em: <https://github.com/mono/mono/blob/main/mcs/docs/compiler.txt>. Acesso em: 20 fev 2021. Citado na página 19.

JR, A. T. B.; SILVEIRA, I. F. Permotive: um modelo conceitual de persuasão, motivação e engajamento para jogos educacionais. *XV Simpósio Brasileiro de Jogos Digitais e Entretenimento (SBGAMES), São Paulo*, p. 920–929, 2016. Citado 2 vezes nas páginas 9 e 12.

LEWIS, M.; JACOBSON, J. Game engines in scientific research. *Communications of the ACM*, v. 45, p. 27–31, 01 2002. Citado na página 14.

MORENO, J. Uso de juegos digitales educativos como herramienta de soporte para el aprendizaje de algoritmos. *RENOTE-Revista Novas Tecnologias na Educação*, v. 8, n. 3, 2010. Citado 2 vezes nas páginas 15 e 16.

MOSER, R. A fantasy adventure game as a learning environment: Why learning to program is so difficult and what can be done about it. *SIGCSE Bull.*, Association for Computing Machinery, New York, NY, USA, v. 29, n. 3, p. 114–116, jun. 1997. ISSN 0097-8418. Disponível em: <https://doi.org/10.1145/268809.268853>. Citado na página 17.

NOVAK, J. *Game development essentials: an introduction*. Canada: Cengage Learning, 2011. Citado 3 vezes nas páginas 13, 14 e 15.

PAPERT, S. *Mindstorms: Children, Computers, and Powerful Ideas*. USA: Basic Books, Inc., 1980. ISBN 0465046274. Citado 3 vezes nas páginas 9, 13 e 22.

RAJARAVIVARMA, R. A game-based approach for teaching the introductory programming course. *SIGCSE Bulletin*, v. 37, p. 98–102, 12 2005. Citado na página 15.

RAPKIEWICZ, C. E. et al. Estratégias pedagógicas no ensino de algoritmos e programação associadas ao uso de jogos educacionais. *RENOTE-Revista Novas Tecnologias na Educação*, v. 4, n. 2, 2006. Citado 2 vezes nas páginas 17 e 18.

RIBEIRO, R. J. et al. Teorias de aprendizagem em jogos digitais educacionais: um panorama brasileiro. *RENOTE-Revista Novas Tecnologias na Educação*, v. 13, n. 1, 2015. Citado na página 12.

ROCHA, M. d. G. B. et al. Currículo de referência da sbc para cursos de graduação em bacharelado em ciência da computação e engenharia de computação. *Relatório Técnico.*, v. 8, 2005. Disponível em: <https://www.sbc.org.br/documentos-da-sbc/send/131-curriculos-de-referencia/760-curriculo-de-referencia-cc-ec-versao2005>. Acesso em: 15 abr 2020. Citado na página 11.

SOUZA, M. d.; JAEGER, E. V.; CARDOSO, B. Ensino de algoritmos apoiado pelo uso de jogos digitais educativos. *Revista Novas Tecnologias na Educação*, v. 11, 12 2013. Citado na página 16.

Unity Technologies. *Unity Platform*. 2021. Disponível em: <https://unity.com/products/unity-platform>. Acesso em: 20 fev 2021. Citado na página 19.

APÊNDICE A – Pré-teste

Seja bem-vindo! Ao preencher o formulário a seguir você vai participar da pesquisa realizada por Victor Augusto Zunho do curso de Bacharelado em Ciência da Computação, sob orientação do professor João Mário Lopes Brezolin. Essa pesquisa tem como objetivo avaliar o jogo de aprendizagem. Os dados obtidos serão utilizados exclusivamente para fins acadêmicos, embasando a produção de conhecimento científico.

1. Você já jogou algum jogo de aprendizagem?
    - Sim
    - Não
2. O que você imagina quando se fala em aprendizagem com jogos?
3. Como você considera seu nível de conhecimento sobre algoritmos?
    - Ótimo
    - Bom
    - Regular
    - Ruim
    - Péssimo

APÊNDICE B – Pós-teste

Seja bem-vindo! Ao preencher o formulário a seguir você vai participar da pesquisa realizada por Victor Augusto Zunho do curso de Bacharelado em Ciência da Computação, sob orientação do professor João Mário Lopes Brezolin. Essa pesquisa tem como objetivo avaliar o jogo de aprendizagem. Os dados obtidos serão utilizados exclusivamente para fins acadêmicos, embasando a produção de conhecimento científico.

1. Qual foi sua percepção em relação ao jogo?
2. Em uma escala de 0 a 10, qual nota você daria à usabilidade do jogo?
3. Em uma escala de 0 a 10, qual nota você daria à clareza das instruções, dicas e ajudas apresentadas no jogo?
4. Você acredita que o jogo contribuiu ou poderia contribuir de alguma forma em sua aprendizagem?
    - Sim
    - Não
5. Justifique a resposta anterior.
6. Você recomendaria esse jogo para um amigo?
    - Sim
    - Não
    - Talvez
7. Justifique a resposta anterior.