Curso de Ciência da Computação
Universidade Estadual De Mato Grosso Do Sul

Desenvolvimento de um Jogo 3D como Proposta de Auxílio no Ensino de Matemática Básica

Wesley de Faria Cazelli

prof[a]. Doutora Mercedes Rocío Gonzales Márquez (Orientadora)

Dourados - MS
2019

Desenvolvimento de um Jogo 3D como Proposta de Auxílio no Ensino de Matemática Básica

Wesley de Faria Cazelli

Este exemplar corresponde à redação final da monografia da disciplina Projeto Final de Curso devidamente corrigida e defendida por Wesley de Faria Cazelli e aprovada pela Banca Examinadora, como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação.

Dourados, 28 de novembro de 2019

prof[a]. Doutora Mercedes Rocío Gonzales Márquez (Orientadora)

Curso de Ciência da Computação
Universidade Estadual De Mato Grosso Do Sul

Desenvolvimento de um Jogo 3D como Proposta de Auxílio no Ensino de Matemática Básica

Wesley de Faria Cazelli

Outubro de 2019

Banca Examinadora:

Profª. Drª. Mercedes Rocío Gonzales Márquez (Orientadora)
Área de Computação - UEMS

Profª. MSc. Raquel Marcia Müller
Área de Computação - UEMS

Profª. Drª. Maria Aparecida Silva Cruz
Área de Matemática - UEMS

AGRADECIMENTOS

Agradeço primeiramente aos meus pais pela oportunidade.

Agradeço à minha orientadora Professora Doutora Mercedes Rocío Márquez, que me permitiu desenvolver esse trabalho e me apoiou do início ao fim.

Agradeço a todos os meus amigos que me apoiaram e a todas as pessoas que direta ou indiretamente me ajudaram ao longo do caminho.

E por fim, um agradecimento especial à coordenação, professores e alunos da escola Vereador Venancio Gonazaga Ramos que aceitaram colaborar com esse trabalho.

RESUMO

É comum alunos terem dificuldade em aprender novos conceitos, com sorte não precisamos ficar presos à apenas um método de ensino. Novas tecnologias e mídias provaram ser ferramentas poderosas para o ensino, elas podem transmitir informação de maneiras inovadoras e podem colaborar para o ensino de qualquer área. Graças a popularização dos jogos eletrônicos um modelo novo de jogos eletrônicos surgiu, os jogos eletrônicos voltados para o ensino. Também conhecidos como *serious games*, esses jogos usam a interatividade oferecida pela tecnologia para entreter e ensinar ao mesmo tempo.

Esse trabalho consiste na criação de um jogo eletrônico 3D com uma proposta educacional para o ensino de matemática para alunos do primeiro ano do ensino fundamental, utilizando a ferramenta Unity3D. Após o desenvolvimento, foi realizado uma pesquisa de campo na escola municipal Vereador Venâncio Gonzaga Ramos para analisar o desempenho do jogo e dos alunos e examinar o interesse dos alunos em jogos eletrônicos com propostas educacionais.

Palavras-Chave: Jogos Eletrônicos, Educação, Matemática.

ABSTRACT

It's commom for students to have diffculty learning new concepts, luckily we don't need to stay stuck in one single teaching medium. New technologies and media have proven to be powerful tools for teaching, they can transmit information in innovative ways and can colaborate in the teaching of any area. Thanks to the popularization of the eletronic games a new model of eletronic games has emerged, the eletronic games aimed for education. Also know as serious games, this games use interactivity offered by the technology to entertain and teach at the same time.

This work consist in the creation of a 3D eletronic game with an educational purpose for the teaching of math to students of the first grades, using the tool Unity3D. After the development, a survey was conducted in the municipal school Vereador Venancio Gonzaga Ramos to analyse the performance of the game and the interest of the students on eletronic games with an educational purpose.

Keywords: Eletronic Games, Education, Math.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

GDD *Game Design Document (Documento de Projeto de Jogo).*

C# *C-Sharp.*

HUD *Head-Up-Display* (Tela de Alerta).

IDE *Integrated Development Enviroment (Ambiente de Desenvolvimento Integrado).*

UI *User Interface* (Interface de Usuário).

LISTA DE *SCRIPTS*

1 INTRODUÇÃO

Jogos são uma das ações mais antigas e influentes na cultura humana. Como exemplos temos os jogos olímpicos na Grécia antiga, os jogos de tabuleiro de xadrez e damas. Kapp afirma que um jogo é uma abstração de um desafio, definido por regras, interatividade e *feedback* (KAPP, 2012).

Os jogos eletrônicos tiveram um começo modesto na indústria e foram evoluindo junto aos computadores pessoais, de tal forma que hoje em dia podem ser encontrados em diversos meios diferentes e de maneiras diferentes. O mercado de jogos eletrônicos é um dos maiores atualmente, segundo dados do BNDES (BNDES, 2018): no ano de 2017, um faturamento total de 1,3 bilhões de reais foi gerado nas empresas brasileiras e um faturamento em escala mundial de cerca de 121.7 bilhões de dólares (NEWZOO, 2018).

Jogos eletrônicos possuem um potencial enorme para passar informações de tal forma que muitas empresas usam simuladores e jogos para treinar seus funcionários. O exército dos Estados Unidos por exemplo, desenvolveu o jogo *America's Army* para treinar soldados. Esse ramo dos jogos eletrônicos foi categorizado como "jogos sérios", poia têm como objetivo ensinar e entreter ao mesmo tempo. A área de jogos sérios voltados para a educação escolar (também conhecidos como jogos educacionais) ainda é pouco explorada no Brasil e no mundo.

Segundo o relatório de olho nas metas (GIFE, 2017), apenas 42,9% dos alunos chegam ao quinto ano do ensino fundamental com nível de aprendizado adequado, um quadro desconfortável para o ensino público brasileiro.

Nesse contexto, este trabalho propõe-se a desenvolver de um jogo eletrônico 3D com uma proposta educacional voltada para o ensino de matemática para alunos do primeiro ano do ensino fundamental.

1.1 Objetivo do projeto

O objetivo principal desse trabalho é desenvolver um jogo eletrônico tridimensional com uma proposta educacional para o ensino de matemática para alunos do primeiro ano do ensino fundamental.

1.1.1 Objetivos específicos

Entre os objetivos específicos pode-se citar:

- Investigar o interesse do público alvo em jogos eletrônicos com propostas educacionais.
- Testar o jogo desenvolvido com o público alvo e receber o *feedback*.

1.2 Metodologia

O desenvolvimento deste trabalho iniciou com uma pesquisa bibliográfica sobre uso de jogos eletrônicos na educação, livros que guiariam o desenvolvimento, ferramentas disponíveis para o desenvolvimento de um jogo e conteúdo matemático aprendido no primeiro ano do ensino fundamental. Diversas pesquisas foram encontradas abordando o uso de jogos na educação como "A utilização dos jogos como recurso didático no ensino-aprendizagem da matemática"de (GALLEGO, 2007). O livro escolhido para auxiliar no desenvolvimento do jogo foi o "Desenvolvimento de *Games*"de (NOVAK, 2010). A ferramenta de desenvolvimento principal foi o Unity3D (UNITY3D, 2019) e o conteúdo que os alunos do primeiro ano aprendem foi verificado no documento Base Nacional Comum Curricular (BNCC, 2019)

Após o levantamento de informações foi escrito um *GDD* baseado no trabalho de alunos da Unicamp (LATANSIO, 2011). O desenvolvimento do jogo eletrônico educacional se iniciou utilizando o motor de jogos Unity3D (UNITY3D, 2019).

Por fim, os resultados foram obtidos através do desenvolvimento do jogo e da pesquisa de campo através de uma entrevista fechada realizada em uma escola municipal no município de Campo Limpo Paulista, SP buscando *feedback* do público alvo baseado no trabalho de Vilarinho (VILARINHO, 2015).

1.3 Organização do texto

No capítulo 2 é feita uma revisão bibliográfica revisando conceitos, ferramentas e trabalhos correlatos utilizados nesse projeto.

No capítulo 3 é apresentado o *Game Design Document* (*GDD*) um documento criado antes do início da produção do jogo para guiar o seu desenvolvimento.

O capítulo 4 discorre sobre o desenvolvimento do jogo e suas fases.

O capítulo 5 apresenta os resultados e conclusões do trabalho.

2 REVISÃO BIBLIOGRÁFICA

2.1 Jogos

Huzinga diz:

> "O jogo é mais do que um fenômeno fisiológico ou um reflexo psicológico. Ultrapassa os limites da atividade puramente física ou biológica. É uma função significante, isto é, encerra um determinado sentido. No jogo existe alguma coisa "em jogo"que transcende as necessidades imediatas da vida e confere um sentido à ação. Todo jogo significa alguma coisa"(HUIZINGA, 2000).

Um jogo pode ser definido por qualquer atividade em que exista a figura de pelo menos um jogador e que possua um ou mais objetivos e regras. Para ilustrar alguns exemplos de jogos temos:

- Xadrez: Jogo de tabuleiro em que o objetivo é capturar o rei do jogador adversário, usando as movimentações específicas que cada peça contém.
- *Sudoku*: Jogo de papel e caneta baseado na colocação lógica de números de 1 a 9 em uma célula vazia, numa grade de 9x9, que contém sub-grades de 3x3. O objetivo é completar todas as 81 células sem repetir os números numa mesma linha, coluna ou grade 3x3.
- *Blackkack* (Vinte-e-um): Jogo de cartas em que cada carta representa um valor (pontuação) e o objetivo é ter mais pontos que o adversário, mas sem ultrapassar o valor de vinte e um.
- tênis: Jogo esportivo que é jogado em uma quadra dividida em duas por uma rede. O objetivo é fazer com que a bola toque na quadra do adversário, sem que este consiga realizar a devolução da bola para a quadra do outro jogador.

2.2 Jogos Eletrônicos

Jogos eletrônicos são jogos que fazem uso de tecnologias digitais para fornecer novas experiências. Nos jogos eletrônicos, o jogador interage com aspectos eletrônicos através de uma entrada de dados (*input*) por meio de um artefato físico (teclado de computador, controle, etc.) e a resposta dos dados que o jogador fornece vem através de uma saída de dados (*output*), também por meio de um artefato físico (televisão, monitor, etc.).

Os jogos eletrônicos surgiram em departamentos de pesquisa de universidades, laboratórios, instalações militares e por fornecedores de produtos de defesa (NOVAK, 2010). Para muitos, o primeiro jogo eletrônico foi o *SpaceWar!*, de 1962 idealizado por um grupo de estudantes do *Massachusetts Institute of Technology* (MIT). O jogo utilizava um supercomputador da época, chamado PDP-1. Na década de 70, com a chegada de microchips, foi possível a criação do primeiro console de *vídeo game*, o Magnavox Odyssey, mas a popularização dos consoles de videogame ocorreu após o lançamento do Atari 2600, que possuía cartuchos removíveis. A partir disso, com o progresso natural das tecnologias e do processamento, os jogos eletrônicos foram evoluindo, novas possibilidades foram abertas para os jogos eletrônicos, e esses começaram a ficar mais "complexos", com novos tipos de interações, chegando ao patamar de serem considerados expressões artísticas. Pode-se citar alguns jogos eletrônicos de importância através das décadas:

- OXO (1952): Um jogo de três em linha (jogo da velha) onde o jogador jogava contra o computador, foi o primeiro jogo a apresentar uma noção de inteligência artificial (SILVA, 2015). Muitos não o consideram como o primeiro jogo eletrônico feito pela falta de movimentação e atualização gráfica em tempo real.

- *SpaceWar!* (1962): Considerado por muitos como o primeiro jogo eletrônico, o jogo consiste em uma batalha entre duas naves, cada uma sendo controlada por um jogador.

- Tennis (1972): Um dos jogos do Magnavox Odyssey, o jogo simula uma partida de tênis entre dois jogadores.

- *Star Fire* (1979): Um jogo em que o jogador tem que atirar em naves espaciais para aumentar a pontuação. O jogo guardava a pontuação do jogador e três letras que o jogador escolhesse para identificação, pela primeira vez um jogo guardava algum dado do jogador (SOLARI, 2011).

- *Chess* (1991): um jogo de xadrez para os computadores *Apple I*, foi o primeiro jogo a possibilitar partidas *on-line* onde a conexão era feita através de linhas telefônicas convencionais.

- *Wolfenstein 3D* (1992): Um jogo de tiro em primeira pessoa e um dos primeiros jogos a usar computação gráfica 3D.

- *Wii Sports* (2006): uma coletânea de jogos esportivos, que faz o uso de controles de movimento.

- *Pokemon GO* (2016): Um jogo que faz uso do sistema de posicionamento global (GPS) e de realidade aumentada cujo objetivo é encontrar (através do GPS) e capturar uma criatura ficcional (através da realidade aumentada).

2.3 Jogos Sérios (*Serious Games*)

A maioria dos jogos eletrônicos tem como objetivo principal entreter o usuário, mas isso não quer dizer que todos os jogos devem ter isso como propósito principal. Uma classe de jogos tem como objetivo principal ensinar algo ao jogador ou treiná-lo, esses jogos são chamados de jogos sérios. Abt diz:

> "Jogos podem ser jogados de maneira séria ou casual. Nós estamos preocupados com os jogos sérios no sentido de que esses têm que ser cuidadosamente e explicitamente feitos para o propósito educacional e não com a intenção primária de diversão. Isso não significa que jogos sérios não são, ou não devem ser divertidos"(ABT, 1987).

O uso de jogos sérios inicia-se na guerra fria. Durante esse período muito do dinheiro investido era em tecnologias militares e não demorou muito para os militares começarem a criar "jogos de guerra"para treinamento de soldados (DAMIEN ALVAREZ JULIAN, 2011). Como exemplo temos o jogo Hutspiel (1955): esse jogo permite que dois jogadores façam experimentos com o impacto de armas nucleares no campo de batalha. A partir da década de 60 vários jogos sérios foram criados pelo exército dos Estados Unidos da América, mas poucas informações foram liberadas para o público. Nessa época, devido à dificuldade de se ter um computador, os jogos sérios nunca foram realmente explorados em outras esferas da sociedade, essa expansão viria décadas mais tarde com a popularização dos computadores.

Jogos sérios podem ser divididos em subcategorias, são elas:

- Jogos militares: Esses jogos são voltados para a vida militar, são usados para treinar recrutas e simular operações.
- Jogos Educacionais: Esses jogos têm como objetivo o ensino de alguma matéria específica, como história, literatura, matemática, etc.
- Jogos de cuidados com a saúde: Através desses jogos pode ser ensinado como doenças funcionam e os cuidados a serem tomados para precavê-las.
- Arte e cultura: São jogos em que a expressão artística do desenvolvedor é mais importante que qualquer outro aspecto.
- Corporativo: tem como propósito realizar o treinamento de algo para o jogador que vá a ser útil no mundo corporativo.

2.4 A escolha do 3D no jogo

Jogos tridimensionais começaram a ser tornar populares no começo do século XXI, e desde então se tornaram a categoria principal do mercado de jogos eletrônicos.

Mesmo com a popularização dos jogos eletrônicos tridimensionais a maioria dos jogos eletrônicos educacionais ainda continuam sendo bidimensionais. Esse trabalho visa não só criar um jogo eletrônico educacional, mas também trazer esse aspecto tridimensional que é ainda pouco explorado em jogos eletrônicos educacionais.

Entre as vantagens do uso da tridimensionalidade podemos citar segundo Kapp (KAAP, 2009):

- Sentimento de "estar lá"pelo jogador.
- Jogadores podem explorar mais possibilidades do que em um universo bidimensional.
- Os jogadores ficam mais focados.
- Jogadores identificam-se melhor com o avatar e com a situação atual.

2.5 Ferramentas de desenvolvimento do jogo

O desenvolvimento de um jogo é um exercício multidisciplinar, sendo assim seria um esforço enorme e fatigoso desenvolver as próprias ferramentas para o desenvolvimento.

As ferramentas utilizadas para o desenvolvimento foram escolhidas através de sua completude, popularidade e não possuírem nenhum tipo de taxa para utilizar.

2.5.1 Blender V2.79

O Blender é um *software* livre (*open source*) feita para criação de objetos tridimensionais. Nele é possível fazer modelagem, criação de esqueleto para o objeto 3D, animação, simulação, renderização e rastreamento de movimento (BLENDER, 2019). Blender se tornou um *software* livre em 2002. Desde então a Blender *Foundation* continua desenvolvendo e melhorando o Blender junto à comunidade.

Curiosamente o Blender possuía um motor de jogos próprio programável em Python, mas foi removido na sua versão 2.8.

Blender é um *softwate* multiplataforma, ou seja, opera de maneira igual em sistemas operacionais Linux, Windows e Macintosh. A interface do Blender utiliza OpenGL, uma API livre, utilizada em computação gráfica para desenvolvimento de aplicativos gráficos.

No desenvolvimento do jogo o Blender foi utilizado para a criação de alguns objetos tridimensionais, como a pirâmide da Figura 1.

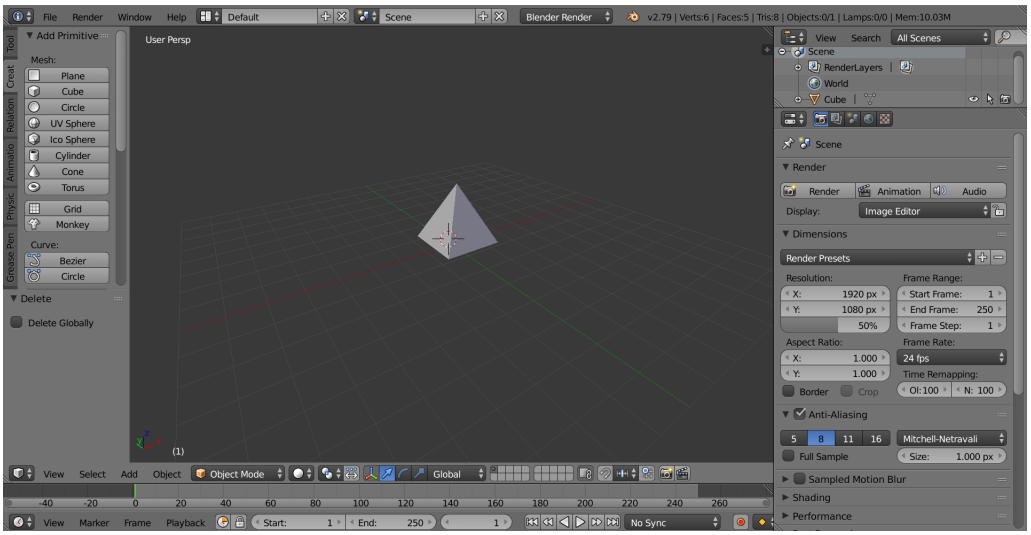


Figura 1 – Pirâmide criada com o Blender

2.5.2 Unity3D V2018.3.13f1

Unity3D é um motor de jogos, uma biblioteca, um pacote de funcionalidades que são disponibilizados para facilitar o desenvolvimento de um jogo (CLEINA, 2011). Um motor de jogos é utilizado para que um desenvolvedor não gaste seu tempo desenvolvendo funções que já foram criadas, agilizando o processo de criação do *design* do jogo.

O Unity3D se apresenta como um motor de jogos extremamente completo e vasto em funções que já estão implementadas. Todos os aspectos de objetos tridimensionais, imagem, som e programação foram gerenciados pelo Unity3D.

Disponível para Windows, Mac e Linux, inclui uma gama de ferramentas artísticas para o design de experiências imersivas e mundos de jogos, além de um conjunto poderoso de ferramentas de desenvolvedor para a implementação de lógica de jogos e jogabilidade (UNITY3D, 2019).

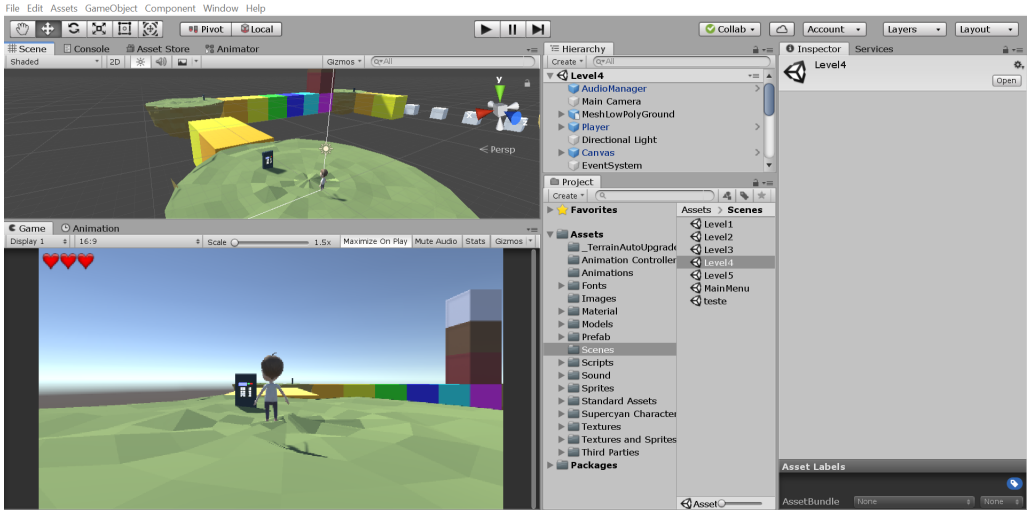


Figura 2 – Interface: Unity3D

2.5.2.1 Sobre funções do Unity3D

Está seção irá explicar brevemente como funciona um *script* no Unity3D. Quando um jogo está sendo executado, a cada segundo que passa, um número variável de quadros é executado. Durante a execução desses quadros ocorrem as chamadas de funções. No seguinte exemplo *Script* Exemplo 2.1, *Start* é a primeira função a ser executada quando o jogo inicia (é executado no primeiro quadro), *Update* é executado a todo quadro (após *Start*). No *Script* Exemplo, a variável x, é declarada com valor zero; ao iniciar o jogo, a função *Start* modificará o valor de x para um e por fim, *Update* incrementa mais um ao valor de x a cada quadro de execução do jogo eletrônico.

Script 2.1 – *Script* Exemplo

```
1  public class Teste : MonoBehaviour {
2
3      private int x = 0;
4
5      void Start()
6      {
7          x = 1;
8      }
9
10     void Update()
11     {
12         x++;
13     }
14
15 }
```

2.5.2.2 Sobre componentes do Unity3D

Componentes são comportamentos já desenvolvidos pelo Unity3D. Por exemplo, é impossível criar um objeto no Unity3D sem o componente *transform*: esse componente define a posição, a rotação e a escala do objeto no ambiente (UNITY3D, 2019).

Além do componente *transform* e alguns outros triviais, esse projeto utilizou frequentemente, devido a sua natureza tridimensional, o componente *collider*, o qual realiza a verificação de colisão entre objetos através de três funções:

- OnTriggerEnter: É executado quando ocorre a colisão.
- OnTriggerStay: É executado enquanto a colisão estiver ocorrendo.

- OnTriggerExit: É executado quando a colisão termina.

2.5.3 *Visual Studio Community 2019 V16.2.1*

Os códigos que regem o jogo foram escritos no ambiente de desenvolvimento integrado (IDE) Visual Studio Community 2019 Figura 3, uma IDE é um *software* que facilita o desenvolvimento do código, diminuindo a ocorrência de erros nas linhas de código. O uso de uma IDE serviu para agilizar a criação de códigos e encontrar erros de maneira mais rápida.

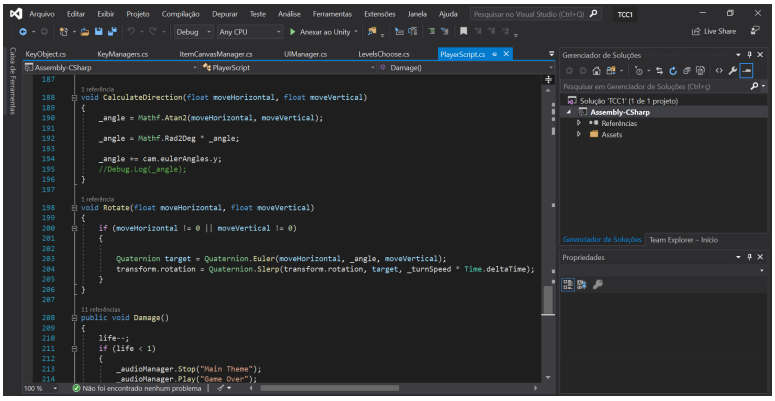


Figura 3 – Interface: Visual Studio Community 2019

2.5.4 *C-Sharp*

C-Sharp ou C# é uma linguagem de programação multiparadigma, desenvolvido pela Microsoft no ano 2000. C# é usado como linguagem de *scripting* pelo Unity3D. *Scripting* diz ao seu objeto como se comportar. Os *scripts* e componentes associados aos objetos e suas interações compõem o *gameplay*. Unity é executado em um grande *loop* e lê todos os dados de uma cena de jogo. Todas as informações sobre iluminação e comportamentos são capturadas e processadas a cada quadro (UNITY3D, 2019).

O Procedimento de dano *Script* 2.2, escrito em C#, é executado quando o jogador comete algum erro. Toda vez que esse procedimento é executado, o jogador perde um ponto (ou uma vida, simbolizada por um coração), caso ele acabe errando várias vezes seguidas e perca todos os pontos a tela de *Game Over* irá aparecer.

Script 2.2 – Procedimento de Dano

```
1  public void Damage()
2      {
3          life--;
4          if (life < 1)
5          {
```

```
6                 _audioManager.Stop("Main Theme");
7                 _audioManager.Play("Game Over");
8                 _uiManager.GameOverPanelOn();
9             }
10            _uiManager.HeartDown(life);
11        }
```

2.5.5 Gimp V2.10.4

Gimp é um *software* livre de edição de imagens multiplataforma disponível para Linux, OS X e Windows (GIMP, 2019). O *software* foi utilizado para criar ou alterar certas texturas no jogo como mostrado na Figura 4.

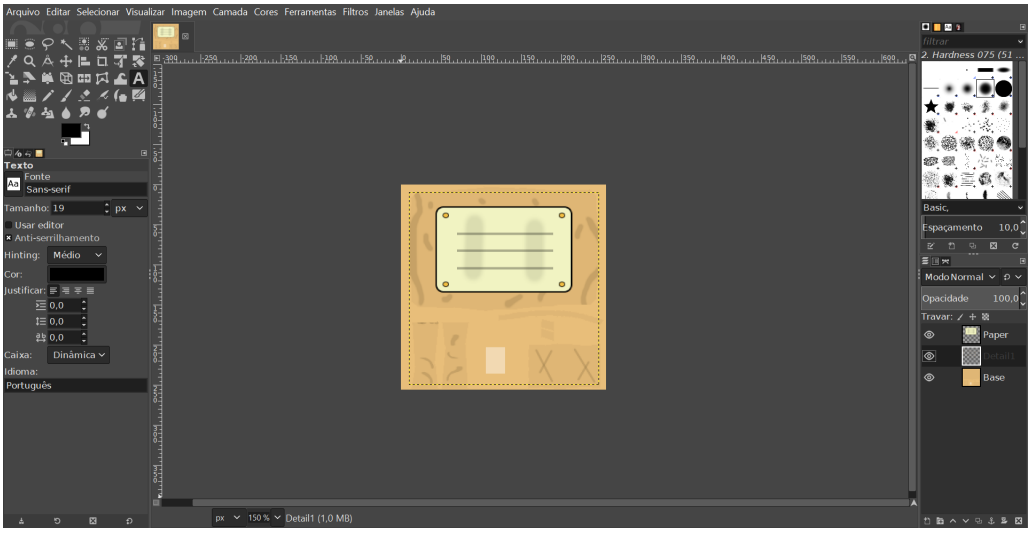


Figura 4 – Interface: Gimp V2.10.4

2.6 Trabalhos correlatos

Esta seção aborda trabalhos acadêmicos que analisam o uso de jogos na educação. Paula (PAULA, 2015) reflete sobre as características específicas de um jogo digital, bem como a integração entre jogos digitais e educação, suas potencialidades e dificuldades, chegando a conclusão que nosso meio de ensino ainda é muito semelhante aquele que surgiu no século XIX, ocorrendo poucas mudanças. No trabalho, é defendido que os jogos eletrônicos podem ser usados como artefato de educação mas que isso não significa que eles são a única mudança necessária para a melhoria da educação.

Gallego (GALLEGO, 2007) verifica se a intervenção através de jogos (não eletrônicos) podem trazer resultados positivos para o ensino de aritmética em crianças da terceira série. Através de testes, foi percebido que crianças tiveram uma melhora no desempenho a partir do uso de material dourado.

Em seu trabalho, Pedro (PEDRO, 2016) foca sua pesquisa no uso de ambientes virtuais para o ensino, suas dificuldades e o mal uso destes pelos alunos (como a mecanização da resolução de atividades). Propõe um sistema em que alunos possam trapacear menos e realizar as atividades de maneira menos mecânica, fazendo com que o conteúdo a ser aprendido seja levado em conta e a atividade não se torne apenas uma repetição assim que a solução é descoberta.

Melo (MELO, 2012) investiga o uso de jogos digitais na educação através do *software* GCompris (uma coletânea de jogos educacionais). A tese verifica se os conhecimentos obtidos no GCompris podem ser passados para a sala de aula. Os resultados obtidos foram positivos, mas é ressaltado a importância do acompanhamento de um mentor pedagógico no uso de *softwares* educacionais.

2.7 Jogos eletrônicos correlatos

Esta seção aborda jogos eletrônicos educacionais testados antes do inicio do desenvolvimento do jogo eletrônico deste trabalho. Gcompris é um *software* educativo gratuito, composto de um conjunto de aplicativos contendo uma ampla quantidade de atividades (GCOMPRIS, 2019). O propósito do GCompris é fornecer uma plataforma de desenvolvimento para atividades educacionais. Nele é possível encontrar atividades educacionais como: funcionamento do computador, uso do *mouse* e teclado, conhecimentos gerais, leitura, escrita, idiomas estrangeiros, álgebra e outras atividades.

Além do GCompris diversos outros jogos educacionais sobre matemática foram encontrados em sites que possibilitam jogar pelo *browser* do computador. Mas assim como o GCompris, esses jogos são apresentados de forma bidimensional e apesar de apresentarem cenários lúdicos, agem de maneira semelhante a uma atividade que poderia ser feita com papel e caneta.

3 GDD (*GAME DESIGN DOCUMENT*)

O processo de desenvolvimento de um jogo é complexo e muitas vezes pode parecer um labirinto. A criação de um *GDD* pode ser considerado um mapa do labirinto de desenvolvimento. Um *GDD* é um documento escrito para expressar as ideias a serem desenvolvidas e evitar retrabalhos no desenvolvimento. Outro ponto de uso de um *GDD* é a organização do desenvolvimento quando se trabalha em equipe. Tendo isso em mente, um dos requisitos de um *GDD* é ter uma linguagem de fácil acesso para qualquer pessoa que leia o documento.

O *GDD* é um documento vivo, ou seja, ele muda e deve ser atualizado conforme ocorrem as mudanças (OLIVEIRA, 2019).

*GDD*s não possuem um padrão exato, cada desenvolvedor escolhe um modelo a seguir, a unica restrição de um *GDD* é que devem ser objetivos e claros. No desenvolvimento de um jogo, a criação de um *GDD* não é uma obrigação, mas sua criação é com certeza fundamental para o desenvolvimento do jogo.

Este capítulo apresenta o *GDD* criado para o desenvolvimento do jogo desse trabalho, o *GDD* foi baseado em diversos modelos encontrados sendo os principais que serviram como inspiração foram o *DOOM Bible* do jogo eletronico *Doom* (HALL, 1992), e o utilizado por alunos da Unicamp no jogo As Aventuras de Jackie e Tony (LATANSIO, 2011).

3.1 Visão Geral

Este documento visa analisar e descrever os aspectos do jogo e seu desenvolvimento.

3.1.1 Tema

O jogo está sendo desenvolvido para ajudar na formação matemática infantil, então ele deve ser educativo, mas ao mesmo tempo ele deve ser interessante o suficiente para que o jogador continue engajado.

3.1.2 Mecânicas básicas de jogabilidade

O jogador poderá:

- Movimentar o personagem no estilo de jogos *adventure*3D;
- Mover a câmera orbitando o personagem;

- Interagir com objetos específicos do cenário.

3.1.3 Plataformas

O jogo estará inicialmente disponível nas seguintes plataformas:

- Sistemas Windows;
- Sistemas Linux.

3.1.4 Modelo de monetização

Não haverá métodos de monetização.

3.1.5 Público alvo

O público alvo almejado são alunos do primeiro e segundo ano do ensino fundamental, ou seja, crianças entre seis e oito anos.

3.1.6 Escopo do projeto

Esta seção apresenta uma previsão sobre prazos e custos.

3.1.6.1 Custo e Prazo

Não haverá custo monetário no desenvolvimento do jogo. O cronograma criado leva em conta o tempo para aprender a usar as ferramentas de desenvolvimento. O cronograma previsto é o seguinte:

- Estudos das ferramentas: 2 meses;
- Desenvolvimento das mecânicas principais: 1 mês;
- *Design* de níveis e produção do jogo: 3 meses;
- Testes e melhorias (fase Beta): 1 mês;
- Entrega do jogo e recepção de *feedback* pelo público alvo: 1 mês.

3.1.6.2 Equipe

A equipe é composta por um único membro. Apesar de alguns aspectos artísticos usados no jogo serem de terceiros, como os modelos tridimensionais da quinta fase, toda a programação será feita pelo autor deste Trabalho de Conclusão de Curso.

3.1.7 Gênero e Inspirações

O jogo foi pensado para ser um *adventure/plataforming* 3D. Isso significa que o jogador para passar de fase ou acessar alguma área nova precisa interagir com objetos que tornem isso possível.

Os jogos eletrônicos tridimensionais mais famosos e bem-sucedidos desse estilo surgiram entre 1994 e 2001. Como exemplo de jogo eletrônico dessa época temos o Super Mario 64 (Figura 5), considerado um marco evolutivo entre a geração bidimensional para a tridimensional. O jogo apresenta várias fases tridimensionais distintas, em que o objetivo é capturar um item (uma estrela); para conseguir prosseguir, os controles permitiam o personagem (Mario) andar, correr e saltar para qualquer direção e a câmera é em terceira pessoa, orbitando o personagem.

Os jogos desse período são as maiores referencias para o desenvolvimento. Outras referências relevantes são:

- *The Legend of Zelda: Ocarina of Time* (1998);
- Banjo-Kazooie (1998);
- *Spyro The Dragon* (1998);
- Donkey Kong 64 (1999).

Figura 5 – Super Mario 64 - 1996

3.1.8 Descrição do projeto

Esse projeto visa criar um jogo educacional tridimensional baseado nas referências de jogos *adventure* 3D do fim do século XX e início do século XXI. No jogo serão abordados conceitos de números, aritmética e figuras geométricas, divididos em cinco fases, cada uma contendo uma mecânica única. As fases terão uma duração curta e um ambiente tridimensional pequeno; o *design* das fases será simples, pois o público alvo é muito jovem.

As tecnologias usadas serão:

- **Motor de jogos**: Unity3D.
- ***Software* de modelagem 3D**: Blender.
- **Ambiente de desenvolvimento integrado**: *Visual Studio Community*.
- **Editor de imagens**: Gimp.

Modelos de terceiros serão retirados dos seguintes *sites*:

- *Unity asset store*: <https://assetstore.unity.com/>
- *Open Game Art*: <https://opengameart.org/>

3.1.9 *Gameplay*

O personagem pode se mover num espaço 3D, ou seja, pode correr em qualquer direção. A câmera será controlada pelo *mouse*, mas sua posição de origem é atrás do jogador. O personagem poderá interagir com alguns objetos.

A missão do jogador será resolver alguma tarefa para passar de fase, cada fase deve conter atividades matemáticas que o jogador deve resolver.

3.1.10 Diferencial do projeto

Existem vários jogos educacionais, em grande maioria todos são desenvolvidos no universo bidimensional, poucos se aventuram no ambiente tridimensional e esse número é ainda menor no ambiente de jogos voltados para o ensino escolar. Esse projeto deseja analisar as dificuldades da implementação de um jogo eletrônico bem como as dificuldades que podem ser apresentadas na jogabilidade pelo público alvo.

3.2 Controle

Nessa seção será abordado as entradas de dados do jogador e como o jogo deve responder a elas.

3.2.1 Entradas de dados

O jogo utiliza teclado e *mouse* para receber dados e realizar as ações do personagem.

3.2.2 Teclado

O teclado é utilizado para controlar o personagem e interagir com objetos.

- **W** ou **flecha para cima**: Movimenta o personagem para frente (eixo z) em relação à câmera.
- **S** ou **flecha para baixo**: Movimenta o personagem para trás (eixo z) em relação à câmera.
- **A** ou **flecha para esquerda**: Movimenta o personagem para a esquerda (eixo X) em relação à câmera.
- **D** ou **flecha para a direita**: Movimenta o personagem para a direita (eixo X) em relação à câmera.
- **Z**: Aproxima a câmera ao jogador.
- **X**: Distancia a câmera ao jogador.
- **E**: Interage com certos objetos no jogo.
- **P** ou **Esc**: Pausa o jogo e abre o menu.

3.2.3 *Mouse*

O *mouse* é utilizado para movimentar a câmera e interagir com certos objetos.

- **Clique botão esquerdo**: Interage com certos objetos do jogo.
- ***Scroll***: Aproxima ou distância a câmera do jogador.
- **Leitura óptica**: Movimento da câmera.

3.3 Fases e seus conceitos

3.3.1 Fase 1

Esse será o primeiro contato do jogador com o jogo, então a fase deve ser simples em um cenário lúdico e espaçoso para o jogador se acostumar com os controles.

O objetivo da fase será mostrar os conceitos de contagem de números.

3.3.2 Fase 2

Uma outra fase simples para que o jogador ainda se acostume com os controles com um cenário ainda lúdico e espaçoso, mas já pode ser implementado um sistema de penalização ao jogador caso ele erre na tarefa da fase.

O objetivo da fase será mostrar o conceito de quantidade de objetos.

3.3.3 Fase 3

Aqui acredita-se que o jogador já tenha se acostumado com os controles do jogo, fases mais objetivas podem ser feitas a partir daqui.

O objetivo da fase será mostrar o conceito de desigualdade.

3.3.4 Fase 4

Aqui deve-se seguir a linha de uma fase mais objetiva ao jogador.

O objetivo da fase será mostrar conceitos aritméticos (soma e subtração).

3.3.5 Fase 5

Por ser a última fase um grau de dificuldade maior possa ser aplicado aqui.

O objetivo da fase será mostrar conceito de figuras geométricas (círculos, triângulos e quadrados).

3.4 Mecânica

A mecânica principal consiste na exploração da fases pelo jogador e a resolução de pequenos *puzzles* (quebra-cabeça) para conseguir ir para próxima fase.

3.4.1 Mecânicas específicas

Cada fase contará com um tipo de mecânica específica que o jogador deve resolver para conseguir ir para próxima fase. As seguintes mecânicas foram idealizadas para o jogo:

- **Fase 1**: Apresenta nove placas tridimensionais em que o jogador deve interagir (clicando perto da placa); cada placa representa um número (de um a nove) e ao interagir com uma placa, uma contagem ascendente ocorrerá, começando do número um até o número da respectiva placa. A fase termina após interagir com todas as nove placas.

- **Fase 2**: Aqui é testada a habilidade do jogador de reconhecer quantidades. Áreas no ambiente 3D devem ser reconhecidas, segundo elas sejam ocupadas por certas

quantidades de objetos. Ao reconhecer a área correta, o jogador precisa clicar em um pedestal para indicar que ele descobriu qual a área certa. Para apresentar um nível de dificuldade, o jogador perderá um ponto de vida ao errar; assim, ao perder três pontos de vida, a fase deve ser iniciada novamente.

- **Fase 3**: A fase 3 treinará o conhecimento de operadores de comparação maior e menor. Novamente aqui, o jogador precisa escolher a opção correta ao pressionar botões que representam um valor (de um a nove) o qual deve ser menor ou maior que um valor ao lado. A fase termina depois de escolher todas as opções corretas.

- **Fase 4**: Essa fase aborda aritmética (soma e subtração). O jogador precisará mover-se de um local ao outro no cenário 3D, através de construção de pontes. Para construir essa ponte, será necessário adicionar ou remover blocos adequadamente. Assim como nas duas fases anteriores, haverá situações em que uma resposta estará certa para continuar, mas também terá situações em que o jogador poderá construir pontes que não contém a mecânica de "resposta certa/errada".

- **Fase 5** a última fase apresentará conceito de formas geométricas. No primeiro ano do ensino fundamental figuras geométricas são trabalhadas em formas bidimensionais, um aspecto que se diferencia desse trabalho que usa um ambiente tridimensional. Para resolver essa solução foi decidido um sistema de correlação entre um objeto tridimensional e o bidimensional. [Exemplo: existe correlação de forma geométrica entre um círculo e uma esfera etc. Será necessário realizar a correlação de três figuras geométricas para terminar a fase].

3.5 Som

Som é uma das partes mais importantes do jogo, ele ajuda a criar climas de terror, suspense, ação e até descontração, e serve também como ferramenta de *Design* para guiar o jogador.

3.5.1 Músicas

Músicas devem ser utilizadas para quebrar a monotonia da falta de som. É preferível que a música seja animada e alegre, podendo ser repetível em algumas fases. Já que a duração das fases são curtas, a música contribuirá para que o jogo não se torne enfadonho.

3.5.2 Efeitos sonoros

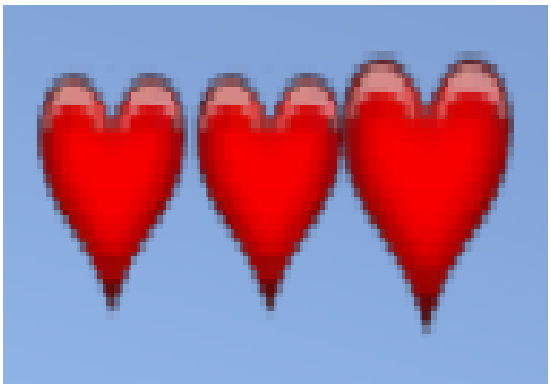
Efeitos sonoros devem ser utilizados para sinalizar acertos e erros do jogador, ajudando-o na resolução de desafios. Outros efeitos sonoros devem ser usados também para criar uma maior ambientação.

3.5.3 Arte visual

Nesta seção são abordados os aspectos de arte visual do projeto.

3.5.3.1 HUD *Head-up-display*

HUD ou monitor de alertas em português, deve conter somente informações necessárias para não poluir visualmente a tela de visualização. Na tela de alertas sempre estarão presentes, no canto superior esquerdo, três corações indicando os pontos de energia do jogador (Figura 6a). O coração pulsante indica que o coração está ativo e sempre que o jogador levar um dano a opacidade do coração ativo será diminuída e ele será desativado, passando a tarefa para o mais a esquerda (Figura 6b), a menos que esse seja o último.



(a) Três pontos de energia ativos

(b) Dois pontos de energia ativos

Figura 6 – Pontos de energia (corações)

Além disso, contadores em fases específicas estarão localizados no canto superior direito, servindo de indicadores de progressão da fase. E alguns objetivos serão descritos por texto no centro inferior da tela.

3.5.3.2 Arte na ajuda do *Game Design*

Além de adornar o jogo, a arte visual pode ser usada para ajudar os jogadores a cumprirem suas atividades. Por exemplo, uma seta tridimensional pode indicar o caminho ou o objetivo para o jogador, além de outros padrões visuais que podem ser adotados, como vermelho que representa uma escolha errada e verde, uma escolha certa.

3.5.3.3 Telas de menu

As telas de menu serão telas simples com botões distribuídos de maneira igual, na vertical, cada um com uma cor diferente para ajudar na distinção entre eles.

3.5.4 Personagem

Os personagens de jogos eletrônicos do gênero *adventure* infantis são em sua grande maioria personagens lúdicos como o urso Banjo (Figura 7) do jogo Banjo-kazooie.

Retirado de:
<https://www.pastemagazine.com/articles/2019/06/banjo-kazooie-is-coming-to-super-smash-bros-ultima.html>

Figura 7 – Urso Banjo e sua amiga Kazooie

Considerando isso, o personagem deve ser algo com o que os jogadores possam se identificar ou algo com um teor mais lúdico como imaginado na *artwork* a seguir:

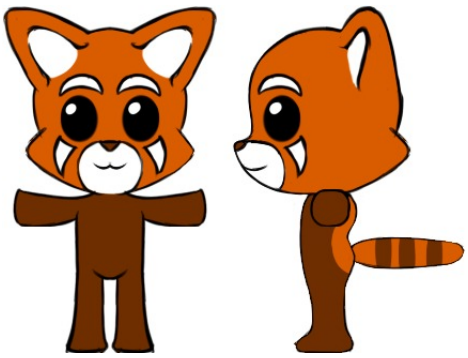


Figura 8 – Esboço de um possível personagem

3.6 História

No presente trabalho optou-se por não desenvolver uma história para o jogo, por dispormos de um prazo curto para a tarefa complexa de realizar animações tridimensionais e dublagem.

4 DESENVOLVIMENTO DO JOGO

Este capítulo abordará o desenvolvimento do jogo e suas fases baseado no livro "Desenvolvimento de Games"de Jeannie Novak. (NOVAK, 2010).

O livro fornece uma visão geral do processo de desenvolvimento de *games* abordando, entre outros temas, os antecedentes históricos, as estratégias de criação de conteúdo, as técnicas de produção e as perspectivas futuras.

Os temas gerais relacionados ao assunto principal deste livro discutidos no *GDD* são:

- Usabilidade e controle pelo jogador como os aspectos principais do desenvolvimento de *games*.
- As características únicas das equipes de desenvolvimento de *games*.
- O "modo de jogar"como uma nova forma de narrativa.

O livro é dividido em três partes, sendo a primeira: Preparação: Construindo os alicerces, onde é descrito os contextos históricos e estrutural dos jogos eletrônicos. A segunda parte discute como os desenvolvedores de *games* criam conteúdos envolventes e a terceira parte do livro discute o enfoque da gestão de projetos e os ciclos de desenvolvimento e comercialização.

No final do livro são apresentados diferentes opiniões e previsões dos especialistas sobre o futuro do setor de *games*.

4.1 Conceito

Após a escolha do tema do jogo que é uma alternativa metodológica para o ensino da matemática para alunos de primeiro ano do ensino fundamental, iniciou-se a etapa de levantamento bibliográfico, escrita do GDD e desenvolvimento. Foi feito o levantamento literário sobre o que é ensinado nas escolas baseado no documento Base Nacional Comum Curricular (BNCC, 2019), o qual é um documento de caráter normativo que define o conjunto orgânico e progressivo de aprendizagens essenciais que todos os alunos devem desenvolver ao longo das etapas e modalidades da Educação Básica (MEC, 2019). Baseado no documento espera-se que crianças do primeiro ano aprendam o conceito de número, quantificação, aritmética (soma e subtração) e formas geométricas.

A partir das informações adquiridas o processo de escrita do *GDD* foi iniciado.

4.2 Protótipo

Após a criação de um GDD a próxima fase da criação do jogo foi criar um protótipo com as mecânicas mais simples e que serão usadas do começo ao fim do jogo, Novak diz:

> "A criação de um protótipo digital é essencial para o processo de desenvolvimento e pode ser o item mais relevante para a decisão de levar ou não o projeto para o próximo nível"(NOVAK, 2010).

O foco da prototipagem foi o desenvolvimento de dois códigos, o de controle do jogador e o de controle da câmera. O código de controle do jogador foi construído para pular e movimentar o jogador pelo ambiente 3D usando o teclado. O código de câmera controla a câmera, para que essa orbite o jogador sendo controlada pelo *mouse*. Os códigos feitos na etapa de prototipagem sofreram poucas mudanças nas etapas posteriores.

4.3 Produção

A fase de produção é a fase na qual o jogo é efetivamente desenvolvido (NOVAK, 2010), sendo a fase mais extensa e trabalhosa do desenvolvimento, principalmente pelo fato de que cada fase do jogo envolve uma mecânica diferente, resultando em pouco reaproveitamento de recursos. A fase de produção representa quase a totalidade de produção de códigos, objetos tridimensionais e sons para o jogo. Vale salientar que alguns objetos tridimensionais e sons usados no jogo foram criados por terceiros e disponibilizados para uso sob uma licença pública.

4.3.1 Controle do Jogador

O elemento principal do jogo é o controle do personagem do jogador. O *script* de movimento do jogador foi o primeiro a ser desenvolvido e o que mais sofreu alteração durante o desenvolvimento. Foi escolhido um modelo de movimentação do personagem tridimensional baseado na perspectiva da câmera que também é controlada pelo jogador.

O *Script* apresentado a seguir é a versão final do controle de jogador. O procedimento *Update*() recebe informações de entrada de dados do jogador e chama dois procedimentos que irão alterar o jogador no ambiente tridimensional. Estes são os procedimento Movimentar() e *Rotate*(), que utilizam dados adquiridos em *Update*() e na posição da câmera para realizar suas respectivas funções. Movimentar() altera a posição do jogador no ambiente tridimensional e *Rotate*() rotaciona o jogador com o auxílio do procedimento *CalculateDirection*() que converte valor radianos em graus.

O *script* também controla os pontos de vida do jogador através de dois procedimentos *Damage* e *LifeUp*. Ao perder três pontos de vida o jogador perde o jogo, *Damage*() tira

um ponto de vida do jogador e verifica se o jogador já perdeu todos os três pontos de vida e fracassou na fase, *LifeUp*() recupera um ponto de vida do jogador.

Script 4.1 – *Script* de movimentação do jogador

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class PlayerScript : MonoBehaviour
6  {
7      private float _velocidade = 6f;
8      private Vector3 _movimento;
9      private CharacterController _controle;
10     private float _gravidade = 10f;
11     private float _velocidadeDeQueda;
12     public bool teste;
13     private float _turnSpeed = 10f;
14     private float _angle;
15     public Transform cam;
16     public int life = 3;
17     private UIManager _uiManager;
18     public bool canMove = true;
19     private AudioManager _audioManager;
20
21     void Start()
22     {
23         _controle = GetComponent<CharacterController>();
24         cam = Camera.main.transform;
25         Cursor.visible = false;
26         Cursor.lockState = CursorLockMode.Locked;
27         _uiManager = GameObject.Find("Canvas").GetComponent<
               UIManager>();
28         _audioManager = GameObject.Find("AudioManager").
               GetComponent<AudioManager>();
29     }
30
31     void Update()
32     {
33         float moveHorizontal = Input.GetAxisRaw("Horizontal");
```

```
34          float moveVertical = Input.GetAxisRaw("Vertical");
35          if (Input.GetKey("escape"))
36          {
37              Cursor.visible = true;
38              Cursor.lockState = CursorLockMode.None;
39          }
40          CalculateDirection(moveHorizontal, moveVertical);
41          if (canMove == true) {
42              Movimentar(moveHorizontal, moveVertical);
43              Rotate(moveHorizontal, moveVertical);
44          }
45      }
46
47
48      void Movimentar(float moveHorizontal, float moveVertical)
49      {
50          if (!_controle.isGrounded)
51          {
52              _velocidadeDeQueda -= _gravidade * Time.deltaTime;
53              teste = false;
54          }
55          else
56          {
57              _velocidadeDeQueda = 0;
58              teste = true;
59          }
60          Vector3 camF = cam.forward;
61          Vector3 camR = cam.right;
62          camF.y = 0;
63          camR.y = 0;
64          camF = camF.normalized;
65          camR = camR.normalized;
66          _movimento = (camF * moveVertical) + (camR *
                moveHorizontal);
67          _movimento = _movimento.normalized * _velocidade +
                Vector3.up * _velocidadeDeQueda;
68          _controle.Move(_movimento * Time.deltaTime);
69          if (transform.position.y < -50f)
70          {
```

```
71              transform.position = new Vector3(0, 2, 0);
72              Damage();
73              _uiManager.HeartDown(life);
74          }
75      }
76
77      void CalculateDirection(float moveHorizontal, float
            moveVertical)
78      {
79          _angle = Mathf.Atan2(moveHorizontal, moveVertical);
80          _angle = Mathf.Rad2Deg * _angle;
81          _angle += cam.eulerAngles.y;
82      }
83
84      void Rotate(float moveHorizontal, float moveVertical)
85      {
86          if (moveHorizontal != 0 || moveVertical != 0)
87          {
88              Quaternion target = Quaternion.Euler(moveHorizontal,
                    _angle, moveVertical);
89              transform.rotation = Quaternion.Slerp(transform.
                    rotation, target, _turnSpeed * Time.deltaTime);
90          }
91      }
92
93      public void Damage()
94      {
95          life--;
96          if (life < 1)
97          {
98              _audioManager.Stop("Main Theme");
99              _audioManager.Play("Game Over");
100             _uiManager.GameOverPanelOn();
101         }
102         _uiManager.HeartDown(life);
103     }
104
105     public void LifeUp()
106     {
```

```
107             if (life < 3)
108             {
109                 life++;
110                 _uiManager.HeartUp(life - 1);
111             }
112         }
113 }
```

4.3.2 Controle da câmera

A câmera é o objeto que irá projetar o jogo eletrônico na tela do dispositivo.

A câmera pode ser considerada um objeto filho do personagem do jogo, pois ela acompanha as mudanças de posição do personagem. A decisão tomada sobre a natureza da câmera seria que ela orbitaria o jogador, sendo controlada pelo *mouse*. O *script* Controle da câmera permite que isso seja possível.

O procedimento *LateUpdate* é executado logo após todos os procedimentos *Update*() terminam de ser executados. Aqui é usado *LateUpdate*(), pois no *script* de controle do jogador, o personagem se move no *Update*(), tornando assim *LateUpdate*() a escolha óbvia para criar uma câmera que segue o jogador, pois ela vai ter a informação da posição do jogador antes de se mover. O método *CollisionCheck*() verifica se a câmera realiza alguma colisão com algum objeto tridimensional no ambiente, e impede a câmera de atravessar esse objeto, caso haja colisão.

Script 4.2 – Controle da câmera

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class CameraController : MonoBehaviour
6  {
7      public bool lockCursor;
8      private float mouseSensitivity = 2.5f;
9      public Transform target;
10     public float distFromTarget = 2;
11     public Vector2 pitchMinMax = new Vector2(-40, 85);
12     public float rotationSmoothTime = 8f;
13     Vector3 rotationSmoothVelocity;
14     Vector3 currentRotation;
15     float yaw;
```

```
16      float pitch;
17      public float moveSpeed = 5;
18      public float returnSpeed = 9;
19      public float wallPush = 0.7f;
20      public float closestDistanceToPlayer = 2;
21      public float evenCloserDistanceToPlayer = 1;
22      public LayerMask collisionMask;
23      private bool pitchLock = false;
24      public bool active = true;
25
26      private void Start()
27      {
28          if (lockCursor)
29          {
30              Cursor.lockState = CursorLockMode.Locked;
31              Cursor.visible = false;
32          }
33      }
34
35      private void LateUpdate()
36      {
37          if (active == true)
38          {
39              if (Input.GetKey(KeyCode.X))
40              {
41                  distFromTarget += 0.5f * Time.deltaTime;
42              }
43              if (Input.GetKey(KeyCode.Z))
44              {
45                  distFromTarget -= 0.5f * Time.deltaTime;
46              }
47              if (Input.mouseScrollDelta.y < 0)
48              {
49                  distFromTarget += 0.5f * Time.deltaTime * 10.0f;
50              }
51              if (Input.mouseScrollDelta.y > 0)
52              {
53                  distFromTarget -= 0.5f * Time.deltaTime * 10.0f;
54              }
```

```
55              distFromTarget = Mathf.Clamp(distFromTarget, 1.0f,
                    4.0f);
56              CollisionCheck(target.position - transform.forward *
                     distFromTarget);
57              WallCheck();
58              if (!pitchLock)
59              {
60                  yaw += Input.GetAxis("Mouse X") *
                        mouseSensitivity;
61                  pitch -= Input.GetAxis("Mouse Y") *
                        mouseSensitivity;
62                  pitch = Mathf.Clamp(pitch, pitchMinMax.x,
                        pitchMinMax.y);
63                  currentRotation = Vector3.Lerp(currentRotation,
                        new Vector3(pitch, yaw), rotationSmoothTime *
                         Time.deltaTime);
64              }
65              else
66              {
67                  yaw += Input.GetAxis("Mouse X") *
                        mouseSensitivity;
68                  pitch = 45;
69                  currentRotation = Vector3.Lerp(currentRotation,
                        new Vector3(pitch, yaw), rotationSmoothTime *
                         Time.deltaTime);
70              }
71              transform.eulerAngles = currentRotation;
72              Vector3 e = transform.eulerAngles;
73              e.x = 0;
74              target.eulerAngles = e;
75          }
76      }
77      private void WallCheck()
78      {
79          Ray ray = new Ray(target.position, -target.forward);
80          RaycastHit hit;
81          if (Physics.SphereCast(ray, 0.2f, out hit, 0.7f,
                collisionMask))
82          {
```

```
83              pitchLock = true;
84          }
85          else
86          {
87              pitchLock = false;
88          }
89      }
90
91      private void CollisionCheck(Vector3 retPoint)
92      {
93          RaycastHit hit;
94          if (Physics.Linecast(target.position, retPoint, out hit,
                collisionMask))
95          {
96              Vector3 norm = hit.normal * wallPush;
97              Vector3 p = hit.point + norm;
98              if (Vector3.Distance(Vector3.Lerp(transform.position
                    , p, moveSpeed * Time.deltaTime), target.position
                    ) <= evenCloserDistanceToPlayer)
99              {
100             }
101             else
102             {
103                 transform.position = Vector3.Lerp(transform.
                        position, p, moveSpeed * Time.deltaTime);
104             }
105             return;
106         }
107         transform.position = Vector3.Lerp(transform.position,
                retPoint, returnSpeed * Time.deltaTime);
108         pitchLock = false;
109     }
110 }
```

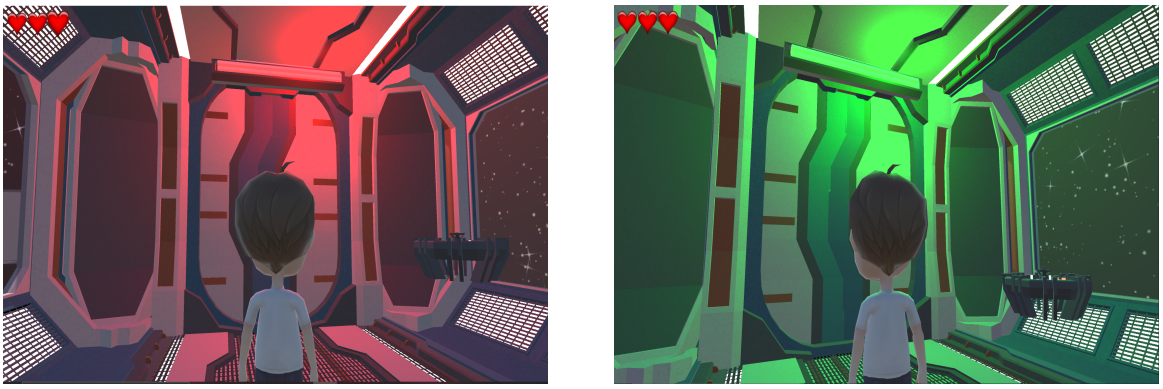
4.4 Som

A música geralmente é usada para criar climas e atmosferas (NOVAK, 2010) e no jogo foram utilizadas músicas compostas por terceiros que estão sob uma licença pública (Apêndice O.3). A música em cada fase se repete de maneira incessável. Já os efeitos sonoros são

utilizados para fornecer *feedback* e indicações ao jogador. No jogo, eventualmente o jogador irá se deparar com situações nas quais para avançar vai precisar realizar escolhas entre certo e errado. O padrão seguido do começo ao fim do jogo será que sempre que o jogador escolher uma opção errada um som grave será tocado, e quando acertar, um som agudo será tocado.

4.5 Arte Visual

A arte visual é uma parte importante do desenvolvimento, servindo para criar identidade visual para a obra. No jogo foi escolhido usar modelos tridimensionais simples, tanto para objetos como cenário tridimensional. A arte visual é um poderoso aliado também para guiar o jogador. Por exemplo, na fase 5 depois que o jogador faz a relação correta entre o item tridimensional para o bidimensional, a luz da porta muda de vermelho (Figura 9a) para verde (Figura 9b), indicando que agora é possível continuar.



(a) Luz vermelha (b) Luz verde

Figura 9 – Uso de luzes na fase 5

4.5.1 Menu Principal

O menu principal (ou menu inicial) é a primeira tela apresentada no jogo. A interação desta tela se dá por meio do *mouse*, onde o jogador pode escolher as seguintes opções:

- Jogar: Inicia o jogo na fase 1.
- Escolher level: Leva para a tela Menu de escolhas de fases.
- Sair: Encerra a execução do jogo.

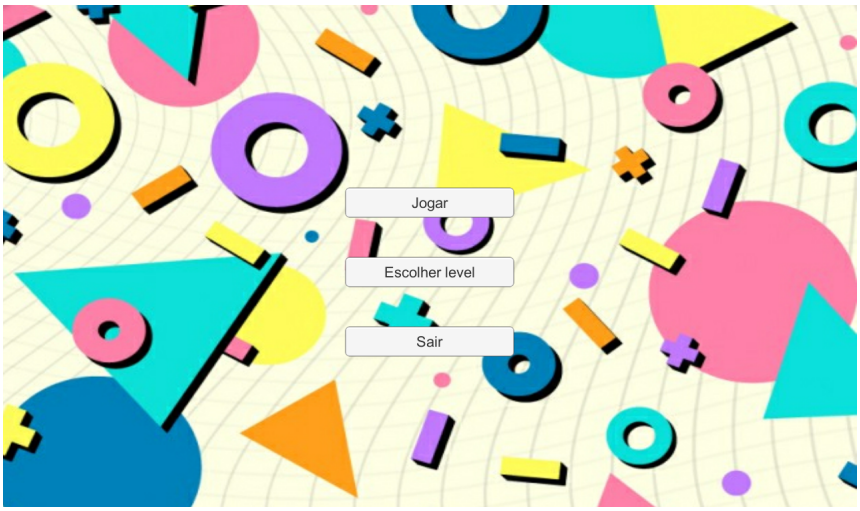


Figura 10 – Menu Principal

Sempre que o jogador passar o cursor do *mouse* por cima de um desses botões, uma cor de realce vermelho irá preencher o botão branco e um som característico irá tocar. Ao clicar em um botão a cor que preencherá o botão será verde.

4.5.2 Menu de Escolha de Fases

Esse menu apresenta a possibilidade de escolher a fase em que o jogador irá jogar. Ao passar o cursor do *mouse* em cima de um dos botões, a cor do botão será realçada e um som específico será tocado. Os botões nessa tela levam o jogador para a respectiva fase, com exceção do botão voltar, que retorna para o menu anterior.

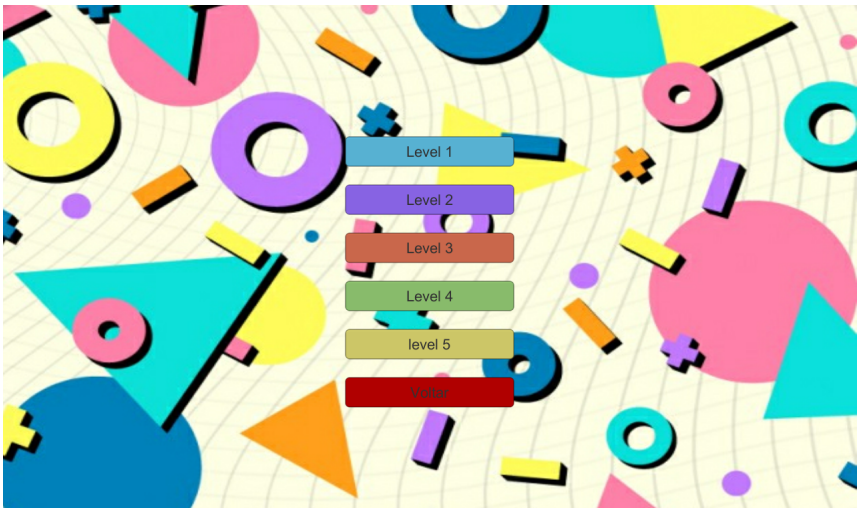


Figura 11 – Menu de Escolha de Fases

4.5.3 Menu *In Game*

Esse menu aparece e pausa o jogo sempre que o jogador apertar o botão Esc ou 'P' e não estiver na tela de menu principal ou menu de escolha de fases. Esse menu contém os

seguintes botões:

- Voltar ao jogo: Oculta o menu e retorna ao jogo.
- Escolher level: Abre o menu de escolha de fases.
- Reiniciar level: Inicia novamente a fase atual.
- Voltar ao menu principal: Retorna para a tela menu principal.

Ao passar o cursor do *mouse* sobre algum botão, uma cor vermelha irá preencher o botão para realçar e um som característico será tocado; ao clicar em um botão, a cor de realce será verde em um novo som é tocado.

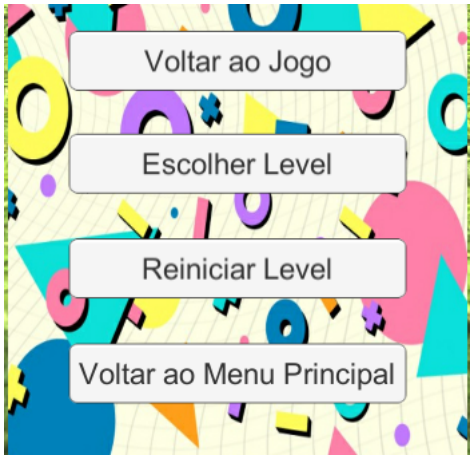


Figura 12 – Menu *In Game*

4.5.4 Tela Próxima Fase

Essa tela surge sempre que o jogador terminar uma fase, onde as opções são:

- Próximo nivel: Inicia a próxima fase.
- Escolher nivel: Leva para a tela menu escolha de fases.
- Menu principal: Retorna para a tela menu principal.

Figura 13 – Tela Próxima Fase

4.5.5 *Game Over*

A tela de *Game Over* apresenta dois botões:

- Tentar novamente: Reinicia a respectiva fase.
- Menu principal: Retorna para a tela menu principal.

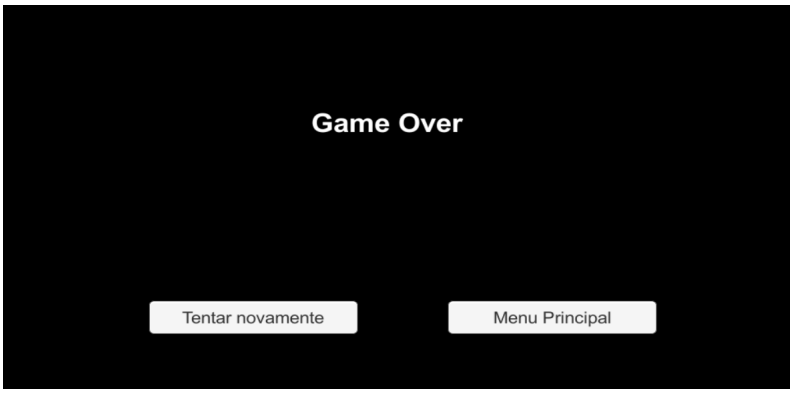


Figura 14 – *Game Over*

Ao passar o cursor sobre um dos botões, o botão será realçado com uma cor vermelha e um som característico será tocado e ao clicar, um novo som será tocado e a cor de realce irá mudar de vermelho para verde.

4.5.6 Modelo utilizado no jogo

Alguns modelos de personagem foram criados no Blender, mas nenhum ficou realmente satisfatório no quesito modelagem ou animação. O Unity3D possui uma loja de *assets* (objetos criados para usar nos jogos). Nessa loja alguns objetos podem ser adquiridos sem custo nenhum, e esse foi o caso do personagem usado no jogo e suas animações (Figura 15).

Figura 15 – Modelo tridimensional usado

4.5.6.1 Ciclo de animações do personagem

O personagem possui quatro animações durante o jogo. O diagrama de animação (Figura 16) controla as animações do personagem durante o jogo. Cada animação é considerada um estado: o estado *idle* representa a animação de "respirar parado"; o personagem começa nessa animação e só sai após ele se movimentar; quando ele para o movimento ele retorna para *idle*; *run* é o estado de corrida do personagem: essa animação ocorre sempre que o jogador se move usando as teclas de movimento do jogo; *jump-float* representa uma animação de queda: ela é constante, até o personagem tocar o chão, então o estado muda imediatamente para *jump-down*, que representa a animação de dobra de joelho após cair de uma certa altura. Se após a queda o jogador ficar parado, o estado mudará imediatamente para *idle*, mas se o jogador mover o personagem logo após tocar o chão o estado mudará para *run*.

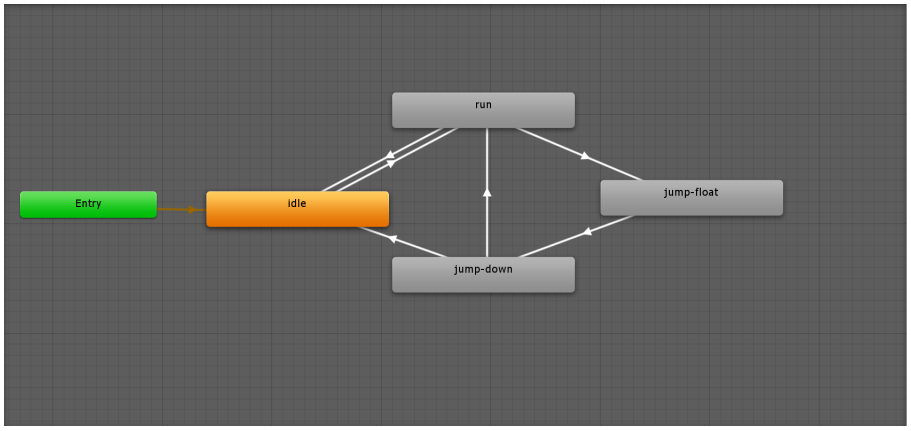


Figura 16 – Diagrama de animação

4.5.7 *HUD* (tela de alerta)

O *HUD* (*heads-up-display*) ou tela de alerta, é a representação de objetos que contém alguma informação útil e contínua, como vida, pontos etc. A única informação que é a mesma para todas as fases são os três corações localizado no canto superior esquerdo (Figura 17). Sempre que o jogador cometer um erro ele perderá um coração e sempre que ele acertar ele irá ganhar um coração. Os corações não excedem três e se o jogador perder os três a tela de *Game Over* irá aparecer.

Figura 17 – HUD

4.6 *Design* de níveis

Design de níveis é definido pela criação de ambientes, cenários ou missões em um jogo eletrônico. Novak diz sobre *design* de níveis:

> "Os níveis podem ser utilizados para estruturar um game em subdivisões eficazes, organizar a progressão e aprimorar o modo de jogar. Ao projetar níveis, considere meta, fluxo, duração, disponibilidade, relações e dificuldade. Cada nível deve ter um conjunto de objetivos compreensíveis ao jogador"(NOVAK, 2010).

Visando isso, cada fase foi projetada para abordar um tema específico. Esses temas são: números, quantidade, comparação entre números, aritmética (soma e subtração) e figuras geométricas respectivamente.

4.6.1 *Design* da Fase 1

A fase 1 (Figura 18) consiste em um ambiente aberto onde o jogador vai ter o primeiro contato com os controles. O objetivo dessa fase é interagir com todas as placas do ambiente,

cada placa representa um número. Após a interação com uma placa um áudio irá fazer uma contagem ascendente a partir do número um até o respectivo número da placa.

Figura 18 – Fase 1

Ao se aproximar da placa uma imagem indicando o botão direito no *mouse* irá aparecer no centro inferior da tela (Figura 19a). Uma contagem de placas de zero a nove está localizado no canto superior direito. A fase é concluída após interagir com todas as nove placas. Após interagir com uma placa o contador irá incrementar o contador de placas interagidas e um número tridimensional irá surgir em cima da placa (Figura 19b), representando o valor da placa e servindo como indicador visual que o jogador já interagiu com aquela placa. O conceito de contagem de números até o respectivo número da placa é apresentado ao interagir com a placa através de áudio e visualização.

Script 4.3 – Mostrar imagem do *mouse*, ativar número tridimensional e incrementar contador

```
1  //Ao se aproximar da placa o seguinte método é evocado
2      void OnTriggerEnter(Collider other)
3  {
4      if (other.tag == "Player")
5      {
6          ShowEKeyboard();
7      }
8  }
9
10 //Esse método mostra a figura de indicação do mouse
11 private void ShowEKeyboard()
12 {
13     _eKeyboard.gameObject.SetActive(true);
14 }
```

```
15
16 //Aqui o numero tridimensional é ativado após a contagem  e o
       contador é incrementado
17 private void OnTriggerStay(Collider other)
18 {
19     if (_timeBool == true)
20     {
21         _3DNumber.SetActive(true);
22         if (_alreadyVisited == false)
23         {
24             _level1Manager.signsRead++;
25         }
26     }
27 }
```

(a) Placa antes de interagir

(b) Placa após interagir

Figura 19 – Interação com as placas

Logo após o jogador clicar em uma placa seu personagem irá ficar parado por um tempo (o tempo é variável para cada placa). Nesse tempo um painel irá aparecer com imagens de frutas sem cor (Figura 20a). A cada um segundo um áudio fará uma contagem de forma ascendente e a imagem da fruta é colorido e o contador abaixo da imagem é incrementado (Figura 20b).

Script 4.4 – Ativa o painel de contagem, ativa cores da imagem

```
1 //Ao se aproximar da placa e ao interagir com a mesma, faz o
      painel ativar
2 private void OnTriggerStay(Collider other)
3 {
4     if ((other.tag == "Player") && Input.GetMouseButton(0))
```

```
5       {
6           if (_panelActive == false)
7           {
8               ShowPanel();
9               _playerScript.canMove = false;
10              _cameraController.active = false;
11              _timerBool = true;
12              StartCoroutine(WaitTime());
13          }
14      }
15  }
16
17  //Faz o personagem ficar parado e o painel ativo por um certo
        tempo
18  private IEnumerator WaitTime()
19  {
20      yield return new WaitForSeconds(_time);
21      _timeBool = true;
22  }
23
24  //Ativa o contador do painel da placa, realiza a contagem e
        ativa as cores das sprites
25  void Update()
26  {
27      if (_timerBool == true )
28      {
29          _timer += Time.deltaTime;
30          if ((_timer >= _timeRate) && (_timeCount < _signID))
31          {
32              _images[_timeCount].color = new Color(1f, 1f, 1f, 1f
                    );
33              _numbers.text = _timeCount+1 + "";
34              _timeCount++;
35              _numbers.text = _timeCount+1 + "";
36              _signAudioManager.PlayOneShot(soundString);
37              _timeCount++;
38              _timeRate += 1.5f;
39          }
40      }
```

```
41 }
```

(a) Placa na contagem ascendente

(b) Placa no final da contagem

Figura 20 – Contagem ascendente

O *script* Gerenciador da fase 1 controla o momento que o jogador termina a primeira fase, o procedimento UpdateLevel1UI atualiza o contador de placas que foram interagidas e EndLevelCheck verifica se todas as placas foram interagidas e caso isso seja verdadeiro o menu de próxima fase irá aparecer.

Script 4.5 – Gerenciador da fase 1

```
1  //Atualiza contador de placas na tela principal
2  public void UpdateLevel1UI()
3  {
4      _level1UI.UpdateTextCount();
5  }
6
7  //Verifica se todas as placas foram interagidas, caso sim a fase
       é terminada
8  public void EndLevelCheck()
9  {
10     if (signsRead == _goal)
11     {
12         _canvasNextLevel.SetActive(true);
13     }
14 }
```

4.6.2 *Design* da Fase 2

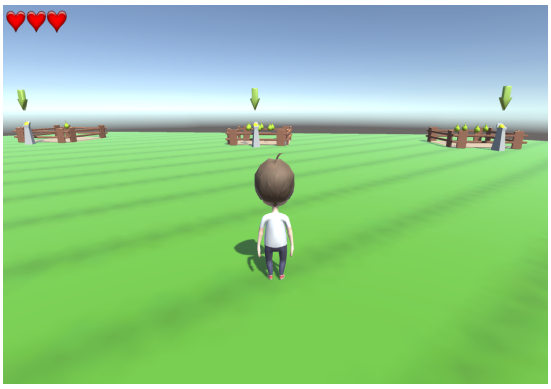
A fase 2 (Figura 21) apresenta pequenas áreas cercadas com frutas dentro (Figura 22a), essas áreas estão divididas em três seções.

Figura 21 – Fase 2

Ao se aproximar de uma seção um painel irá aparecer com um texto e imagem indicando qual área deve ser escolhida. Flechas tridimensionais indicam os pedestais que o jogador pode interagir para realizar a escolha (Figura 22b).

Script 4.6 – Painel de objetivo

```
1 //Ao se aproximar de um seção o painel de objetivo irá aparecer
2 private void OnTriggerEnter(Collider other)
3 {
4     ShowText();
5 }
```



(a) Áreas

(b) Painel com objetivo

Figura 22 – Segunda fase

Ao escolher a opção errada a flecha indicadora irá sumir, a esfera amarela irá mudar para vermelha (Figura 23a), em contra partida, se acertar, a esfera mudará para verde (Figura 23b) e todas as flechas da seção irão sumir, a fase termina após acertar a opção correta de cada seção. Esta fase apresenta o conceito de quantidade, em que o jogador deve fazer a contagem exata para passar de fase.

Script 4.7 – Interação com o objetivo

```
1  //Esse procedimento só ocorre caso o jogador esteja perto do
       pedestal
2  private void OnTriggerStay(Collider other)
3  {
4      //Caso o jogador escolha a opção correta, a esfera correta (
           verde) é ativada, flechas tridimensionais da seção
           desaparecem, contador de objetivo é incrementado.
5      if ((Input.GetKey(KeyCode.E) || Input.GetMouseButton(0)) &&
           _rightOrWrong == true)
6      {
7          _defaultSphere.gameObject.SetActive(false);
8          _rightSphere.gameObject.SetActive(true);
9          _selfArrow.SetActive(false);
10         _arrow1.SetActive(false);
11         _arrow2.SetActive(false);
12         if (_wasPressedBefore == false)
13         {
14             _level2Manager.rightChoice += 1;
15             _wasPressedBefore = true;
16             _level2Manager.EndLevelCheck();
17         }
18       //Caso a escolha seja errada, a esfera errada (vermelha) é
             ativada, a flecha tridimensional da respectiva área
            desaparece
19     } else if ((Input.GetKey(KeyCode.E) || Input.GetMouseButton
           (0)) && _rightOrWrong == false)
20     {
21         _defaultSphere.gameObject.SetActive(false);
22         _wrongSphere.gameObject.SetActive(true);
23         _selfArrow.SetActive(false);
24         //Verifica se o pedestal já foi interagido ou não
25         if (_wasPressedBefore == false)
26         {
27             _wasPressedBefore = true;
28         }
29     }
30 }
```

(a) Área errada

(b) Área certa

Figura 23 – Áreas certas e erradas

O *script* Gerenciador da fase 2 controla o momento que o jogador termina a segunda fase, o procedimento EndLevelCheck() verifica se as três áreas corretas foram interagidas pelo jogador.

Script 4.8 – Gerenciador da fase 2

```
1 public void EndLevelCheck()
2 {
3     if (rightChoice == _goal)
4     {
5         _canvasNextLevel.SetActive(true);
6     }
7 }
```

4.6.3 *Design* da Fase 3

A fase 3 (Figura 24) apresenta o conceito de operadores de comparação maior e menor. O objetivo estará escrito em uma placa e o jogador deve realizá-lo para terminar a fase.

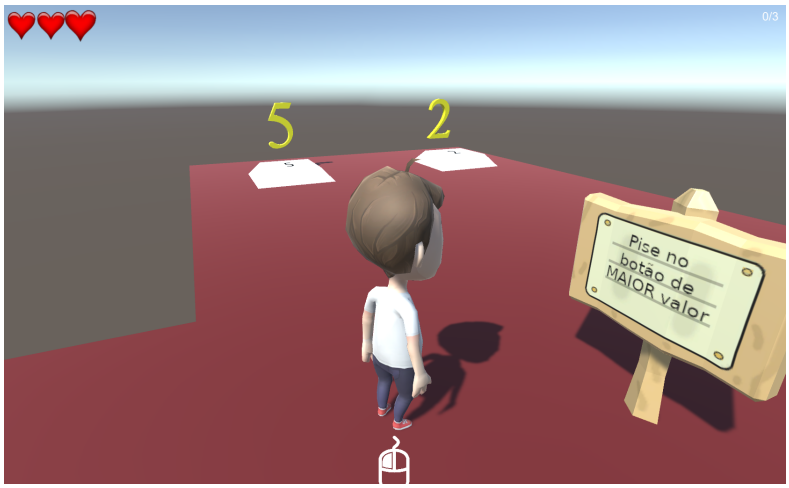


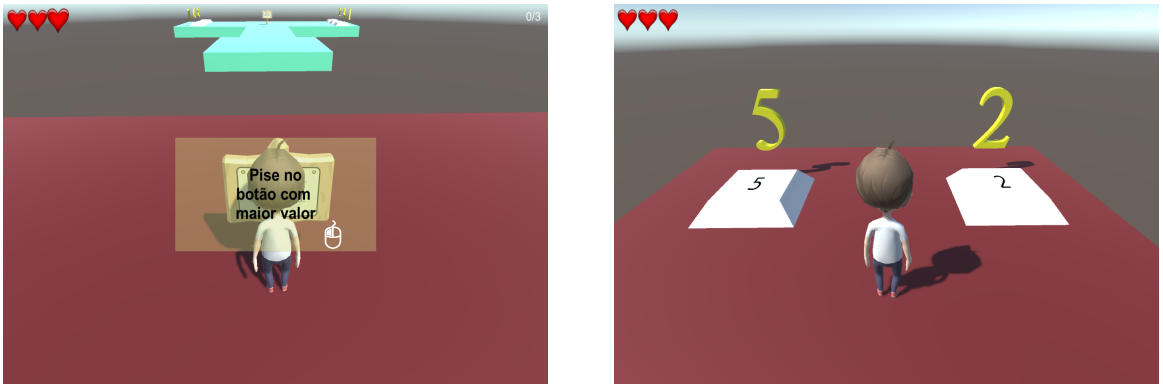
Figura 24 – Fase 3

Um painel irá aparecer ao clicar com botão direito do *mouse* quando o personagem estiver próximo a placa (Figura 25a), a partir daí o jogador deve fazer com que o personagem suba no botão que representa a resposta correta (Figura 25b).

Script 4.9 – Painel de objetivo

```
1  //Chama o método de ativar o painel, caso o painel já esteja
       ativado o painel é desativado
2  private void OnTriggerStay(Collider other)
3  {
4          //Verifica se o jogador está próximo da placa e se ele
               clicou (ou apertou o botão E)
5          if ((other.tag == "Player") && (Input.GetKeyDown(KeyCode
               .E) || Input.GetMouseButtonDown(0)))
6          {
7              //Verifica se o painel está oculto
8              if (_panelActive == false)
9              {
10                 ShowPanel();
11             }
12             else
13             {
14                 HidePanel();
15             }
16         }
17 }
18
```

```
19 //Apresenta o painel de objetivo
20 private void ShowPanel()
21 {
22     _signPanel.gameObject.SetActive(true);
23     _panelActive = true;
24 }
25
26 //Oculta o painel de objetivo
27 private void HidePanel()
28 {
29     _signPanel.gameObject.SetActive(false);
30     _panelActive = false;
31 }
```



(a) Painel com objetivo (b) Botões

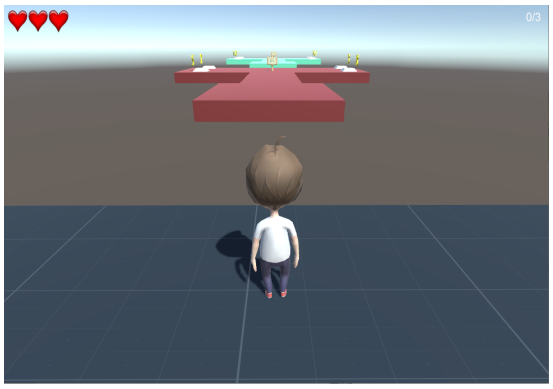
Figura 25 – Objetivo e botões

Para ir de um objetivo ao próximo é necessário acertar o objetivo da área atual. Após acertar, o número tridimensional em cima do botão ficará verde e uma ponte irá surgir (Figura 26b) e o jogador poderá prosseguir para o próximo objetivo, caso erre a número tridimensional ficará vermelho e o jogador perderá um ponto (coração). A cada nova seção da fase novos botões são apresentados aumentando a chance de erros. A fase termina ao realizar o ultimo objetivo.

Script 4.10 – Construção da ponte

```
1 //Verifica se a resposta da área foi correta
2 void TestAnswer(int answer)
3 {
4     //Caso a resposta esteja certa o procedimento para construir
          a ponte é acionado
```

```
5       if ((answer == 1) && (stage == 1))
6       {
7           stage = 2;
8           StartCoroutine(WaitTime());
9       }
10  }
11
12  //Procedimento que constroi ponte de uma área a outra
13  private IEnumerator WaitTime()
14  {
15      //Constrói um pedação da ponte a cada 0,75 segundos
16      for (int i = 0; i < 5; i++)
17      {
18          yield return new WaitForSeconds(0.75f);
19          if (stage == 2)
20          {
21              _bridge1[i].SetActive(true);
22          }
23      }
24  }
```



(a) Ponte desativada

(b) Ponte após completar objetivo

Figura 26 – Terceira Fase

4.6.4 *Design* da Fase 4

A fase 4 (Figura 27) trabalha com aritmética (soma e subtração), a fase está cheia de ilhotas flutuantes e para o jogador ir para de uma ilha a outra ele deve adicionar ou remover blocos para construir uma ponte entre as ilhas.

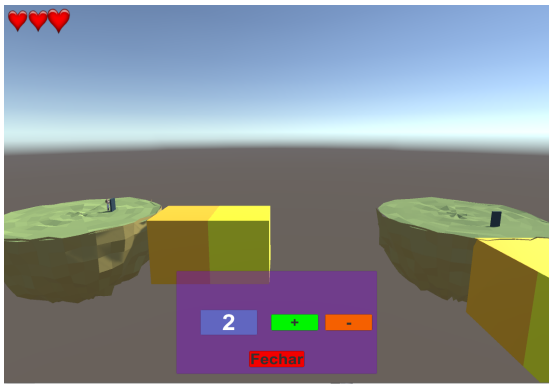
Figura 27 – Fase 4

Essa construção se dá interagindo com um painel que utiliza soma ou subtração de cubos. Nas duas primeiras ilhotas o jogador apenas precisa clicar no botão '+' ou '-' para acrescentar ou decrementar um cubo na ponte, mas a partir da terceira ilhota o jogador deve realizar um cálculo aritmético simples para criar a ponte. A fase termina quando o jogador alcança a última ilhota.

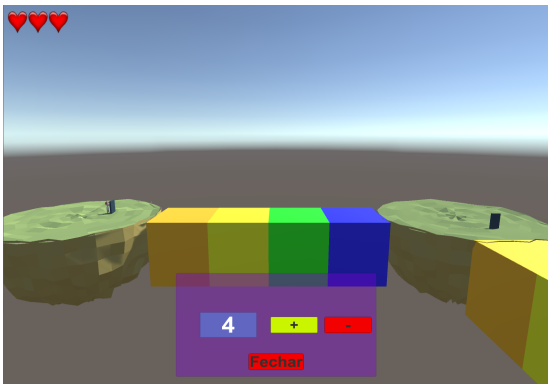
Script 4.11 – Painel de construção de pontes da primeira ilhota

```
1  //Ao clicar no botão de soma um cubo é adicionado a ponte, se
       esta estiver com menos de quatro cubos
2  public void PlusCount()
3  {
4      if (_count < 4) {
5          _count++;
6          UpdateCountText();
7          UpdateCubes();
8      }
9  }
10
11 //Ao clicar no botão de subtração um cubo é removido da ponte,
       se esta estiver com pelo menos um cubo
12 public void MinusCount()
13 {
14     if (_count > 0) {
15         _count--;
16         UpdateCountText();
17         UpdateCubes();
18     }
19 }
20
```

```
21 //Atualiza o texto de contagem de cubos
22 private void UpdateCountText()
23 {
24     _textButton.text = "" + _count;
25 }
26
27 //Método que adiciona ou remove cubos do ambiente tridimensional
       , é invocado a cada soma ou subtração
28 private void UpdateCubes()
29 {
30     for (int i = 0; i < 4; i++)
31     {
32         if (i < _count)
33         {
34             _cubes[i].gameObject.SetActive(true);
35         }
36         else
37         {
38             _cubes[i].gameObject.SetActive(false);
39         }
40     }
41 }
```



(a) Painel



(b) Ponte após adicionar cubos

Figura 28 – Ponte entre ilhotas

4.6.5 *Design* da Fase 5

A fase 5 (Figura 29) é a última fase do jogo, ela consiste em encontrar objetos tridimensionais e relaciona-los com sua contraparte bidimensional, por exemplo, uma esfera com um círculo.

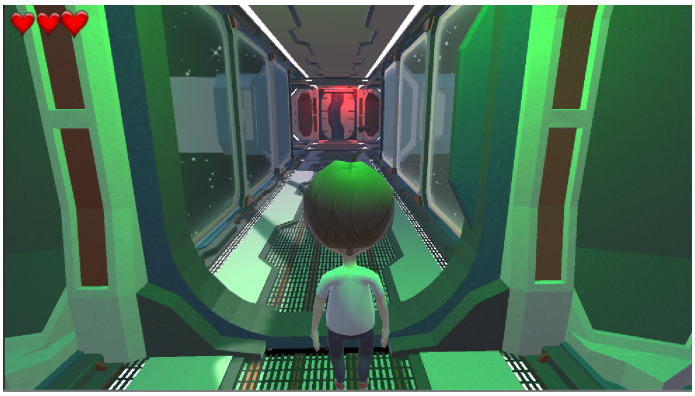
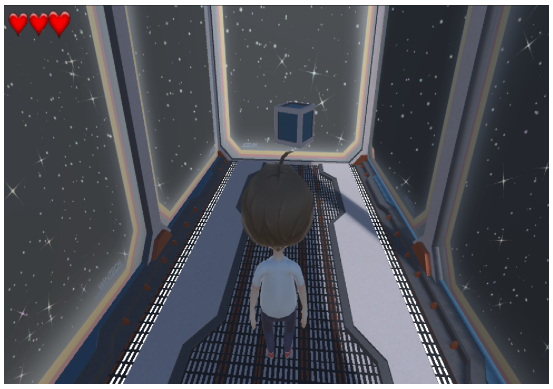


Figura 29 – Fase 5

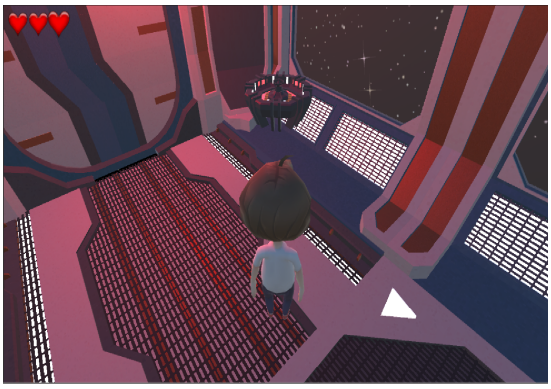
Para realizar a relação entre objetos é necessário primeiramente encontrar o objeto tridimensional no cenário (Figura 30a) e após encontrar é necessário que o jogador toque no objeto para que ele apareça no inventário. Para abrir a porta é necessário interagir com o objeto flutuante que está localizado perto da porta (Figura 30b), esse objeto abrirá um painel para realizar a relação entre objetos.

Script 4.12 – *Script* de contato com o objeto tridimensional

```
1  //OnTriggerEnter verifica se ha alguma colisão com o objeto.
2  private void OnTriggerEnter(Collider other)
3  {
4      //Verifica se a colisão ocorreu com o jogador
5      if (other.tag=="Player")
6      {
7          //Verifica qual objeto tridimensional foi tocado pelo
               jogador
8          if (this.gameObject.tag == "CubeKey")
9          {
10             _keymanagers.cubeImage = true;
11         }
12         Destroy(this.gameObject);
13         }
14 }
```



(a) Objeto tridimensional (cubo)



(b) objeto flutuante

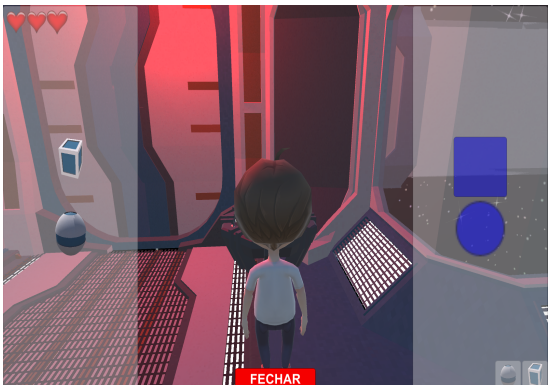
Figura 30 – Objetos da fase 5

Ao interagir com o objeto flutuante dois painéis irão abrir, um contendo imagens dos objetos tridimensionais encontrados e o outro com figuras geométricas bidimensionais (Figura 31a). Para realizar a relação entre as imagens basta clicar na imagem da figura tridimensional encontrada e arrastar a imagem para sua contraparte bidimensional (Figura 31b). Após relacionar todos os objetos a última porta abrirá, concluindo a fase.

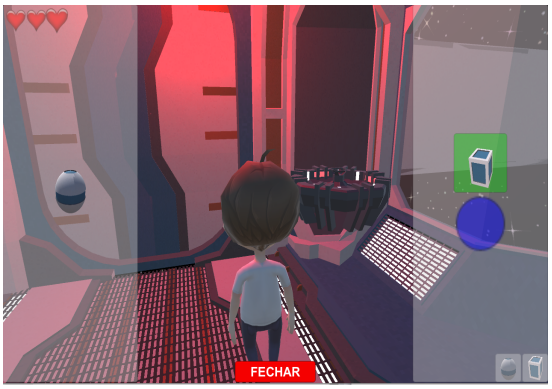
Script 4.13 – *Script* para arrastar uma imagem para um ponto

```
1  //Realiza o arraste da imagem
2  public void OnDrag(PointerEventData eventData)
3  {
4      this.transform.position = eventData.position;
5  }
6
7  //Esse método é executado quando o usuário solta o botão do
       mouse quando ele estiver arrastando imagem
8   public void OnDrop(PointerEventData eventData)
9  {
10     Draggable d = eventData.pointerDrag.GetComponent<Draggable
           >();
11     //Verifica se o mouse está arrastando realmente algo
12     if (d != null)
13     {
14         //Verifica se a imagem solta está no local certo, se
               estiver realiza um verificador para abrir a porta
15         if (slotKey == d.keyId)
16         {
17             d.parentToReturnTo = this.transform;
```

```
18                  img.color = new Color(0, 1, 0, 0.4f);
19                  OpenDooRValidator();
20              }
21          }
22      }
```



(a) Antes da relação entre objetos



(b) Após a relação entre objetos

Figura 31 – Relacionamento entre os objetos tridimensionais e bidimensionais

4.7 Beta

(NOVAK, 2010) separa as fases que ocorrem após a produção em Alfa e Beta, mas para o desenvolvimento desse projeto não houve uma separação clara entre essas fases, então serão consideradas sendo apenas a fase Beta. Na fase Beta o jogo estava jogável do começo ao fim, algumas mecânicas construídas como a possibilidade de o jogador pular, foram retiradas por não apresentar contribuição significativa para o jogo. Houve aprimoramento em alguns códigos como no código de câmera que agora fazia cálculos de colisão com paredes, aspectos sonoros e de artes visuais foram adicionados para melhorar a jogabilidade e por fim todas as fases foram testadas e os erros encontrados foram corrigidos.

4.8 *Gold*

Nessa fase foi avaliado pelo desenvolvedor que o produto está pronto para ser lançado (NOVAK, 2010). Pode-se considerar essa a versão 1.0 do jogo, essa foi a versão utilizada para realizar a pesquisa de campo.

4.9 Pós-produção

Durante a fase de pós-produção, várias versões subsequentes também podem ser lançadas para substituir e melhorar o jogo original, aumentando sua longevidade (NOVAK, 2010).

Nessa fase são corrigidos erros não encontrados na fase beta e melhorias depois do *feedback* da pesquisa de campo.

5 RESULTADOS

Como parte final desse trabalho foi realizado uma pesquisa de campo com alunos do primeiro e segundo ano do ensino fundamental, da escola municipal Vereador Venancio Gonzaga Ramos do município de Campo Limpo Paulista, São Paulo, através de um entrevista fechada.

Participaram da entrevista trinta e um alunos, dezesseis do primeiro ano e quinze do segundo ano.

O jogo foi criado baseado na grade curricular de matemática visto no primeiro ano do ensino fundamental. A alfabetização só se concretiza no segundo ano como visto na base nacional comum curricular (BNCC, 2019). Ao longo do jogo encontra-se alguns textos indicativos que ajudam o jogador a avançar no jogo, então os alunos do primeiro ano contaram com a ajuda do autor desse texto ao longo dos testes para ler os textos e dar algumas dicas.

Após testarem o jogo eletrônico, os alunos participaram de uma entrevista com doze perguntas. Além da entrevista os alunos foram analisados se conseguiam controlar o personagem de maneira agradável e resolver as tarefas sem dificuldades.

5.1 Entrevista

A entrevista foi dividida em três seções. A primeira seção analisa o histórico do aluno com jogos eletrônicos, se ele possui familiaridade com esses, onde e como costuma jogar e se ele já jogou algum jogo eletrônico educativo antes.

A segunda seção da entrevista faz o aluno analisar e avaliar o jogo. As questões desta seção foram modificadas e simplificadas do trabalho de (VILARINHO, 2015), que em seu trabalho desenvolve um questionário para análise de jogos eletrônicos no âmbito pedagógico.

A última seção recebe um *Feedback* mais direto dos alunos, em relação ao que se deve mudar no jogo eletrônicos e também verifica se os alunos possuem interesse em jogar mais jogos educacionais.

5.1.1 Primeira seção

A primeira pergunta realizada verificava se os alunos tinham o costume de jogar jogos eletrônicos regularmente, e onde costumavam jogar.

No primeiro ano seis alunos informaram que normalmente jogam em um console de *video-game*, três costumam jogar no computador, um no celular e seis alunos não costumam jogar. Desses seis alunos que não costumam jogar cinco são do sexo feminino.

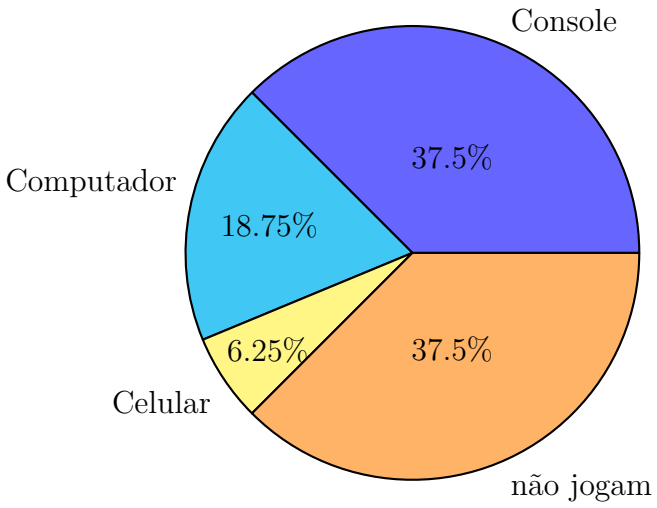


Figura 32 – Gráfico, primeiro ano: Você costuma jogar em qual aparelho?

O segundo ano apresenta um uso forte de celular para jogos, o fator provável disso é o interesse em jogos *online* de fácil acesso, sete alunos informaram que gostam de jogar no celular. três alunos informaram que preferem jogar no console de *video-game*, dois alunos costumam jogar no computador e três alunos informaram que não jogam, os três alunos que informaram que não costumam jogar são do sexo feminino.

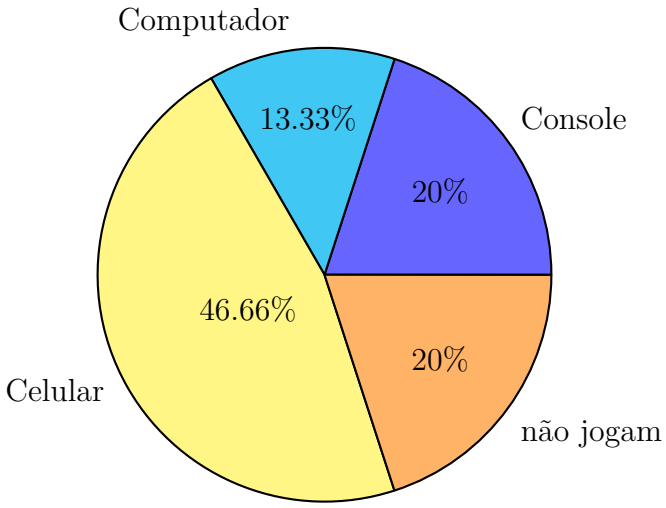


Figura 33 – Gráfico, segundo ano: Você costuma jogar em qual aparelho?

Após isso foi perguntado para aqueles que costumam jogar qual gênero dos jogos eletrônicos eles estão acostumados a jogar ou quais eles gostam mais de jogar. Tanto no primeiro ano quanto no segundo os gêneros foram variados, ambas as pesquisas não encontraram um gênero de jogos eletrônicos dominante.

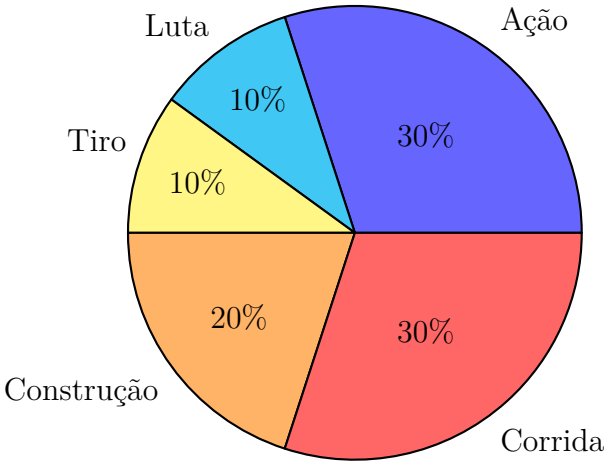


Figura 34 – Gráfico, primeiro ano: Qual gênero de jogo eletrônico você mais joga?

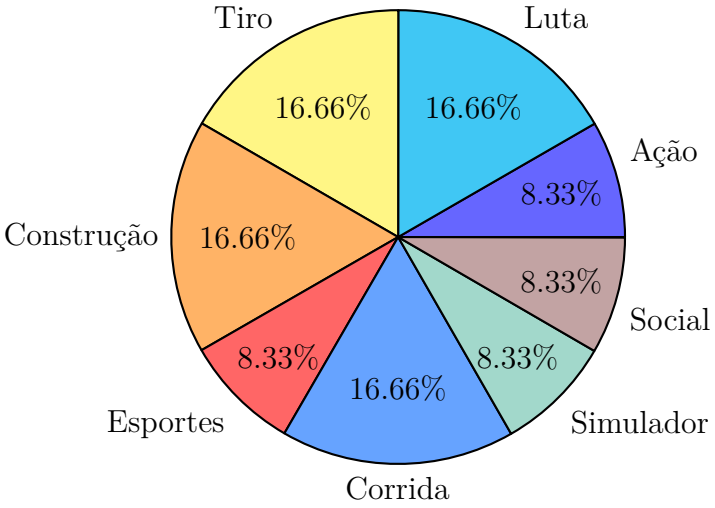


Figura 35 – Gráfico, segundo ano: Qual gênero de jogo eletrônico você mais joga?

A terceira pergunta verifica se os alunos costumam jogar sozinhos ou não, *online* ou *offline*. A maioria dos alunos revelou que normalmente joga com os pais.

No primeiro ano dois alunos informaram que jogam sozinhos, sete jogam com família ou amigos *offline* em casa, apenas um informou que costuma jogar *online* sozinho.

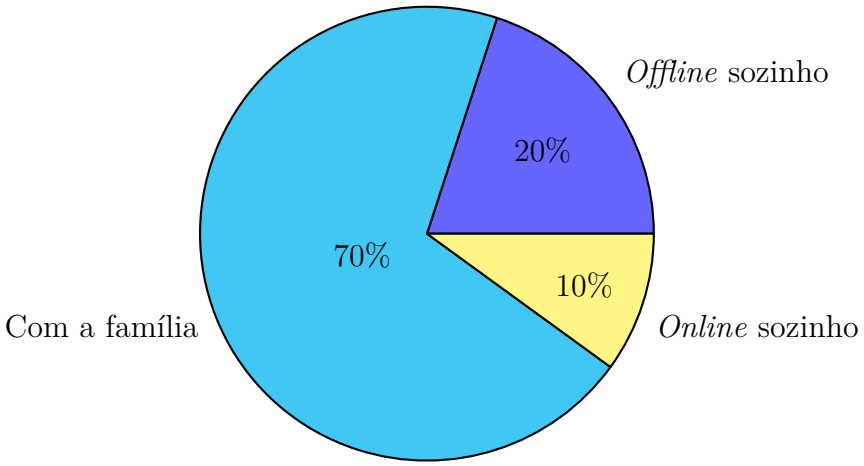


Figura 36 – Gráfico, primeiro ano: Como você costuma jogar?

No segundo ano quatro alunos informaram que jogam *offline* sozinhos, quatro jogam com *offline* com a família ou amigos, apenas um joga sozinho *online* e três jogam *online* com família ou amigos.

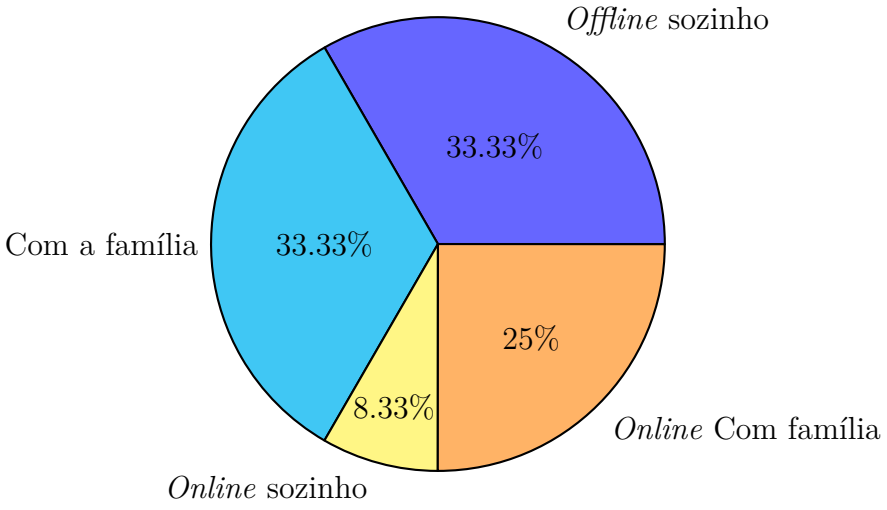


Figura 37 – Gráfico, segundo ano: Como você costuma jogar?

A última pergunta da seção verifica se os alunos já tiveram contato anteriormente com algum jogo eletrônico educacional.

A grande maioria dos alunos do primeiro ano nunca tiveram contato com jogos eletrônicos educacionais, doze alunos responderam que era a primeira vez jogando um jogo educacional e quatro informara que já haviam jogado antes.

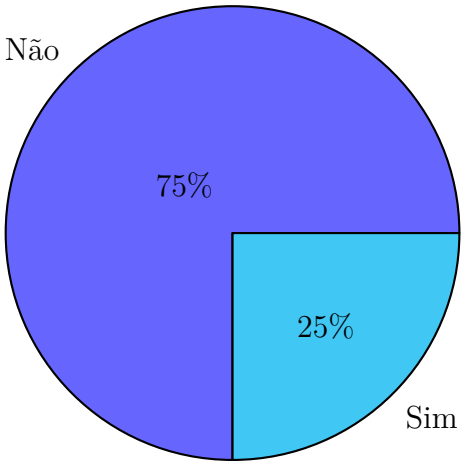


Figura 38 – Gráfico, primeiro ano: Você já jogou algum jogo eletrônico educativo antes?

A resposta do segundo ano não foi muito diferente, dez alunos nunca haviam jogado um jogo educacional antes e cinco já haviam jogado.

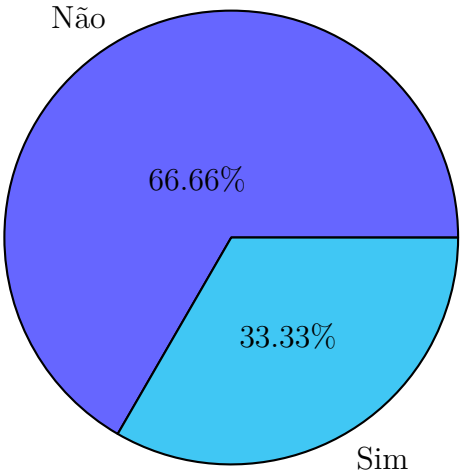


Figura 39 – Gráfico, segundo ano: Você já jogou algum jogo eletrônico educativo antes?

5.1.2 Segunda seção

A segunda parte da entrevista faz os alunos analisarem o jogo após ele ser testado. Essa etapa é um pouco complicada, então foi usado uma versão simplificado do formulário apresentado em (VILARINHO, 2015), nessa fase as respostas são dadas em formato de escala (fácil, médio e difícil). As respostas do segundo ano e do primeiro podem parecer contra indicativas, mas vale lembrar que os alunos do primeiro ano recebiam dicas para prosseguir no jogo, já os alunos do segundo ano não.

O formulário começa perguntado se os alunos acharam as tarefas envolvendo matemática no jogo foram de nível fácil, médio ou difícil.

O primeiro ano em geral achou fácil, doze alunos responderam que não tiveram dificuldades, três alunos falaram que a dificuldade estava média e um aluno achou as atividades difíceis.

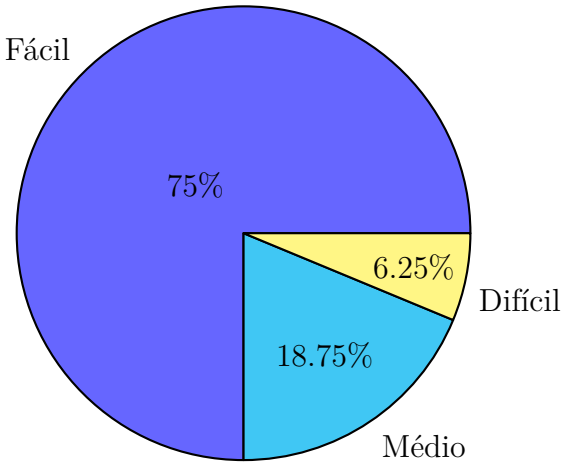


Figura 40 – Gráfico, primeiro ano: As atividades apresentadas no jogo estão em qual grau de dificuldade?

Nesse quesito apesar de estar jogando sem dicas o segundo ano não teve opiniões convergentes. Dez alunos acharam as atividades fáceis, quatro acharam a dificuldade mediana e um aluno achou as atividades difíceis.

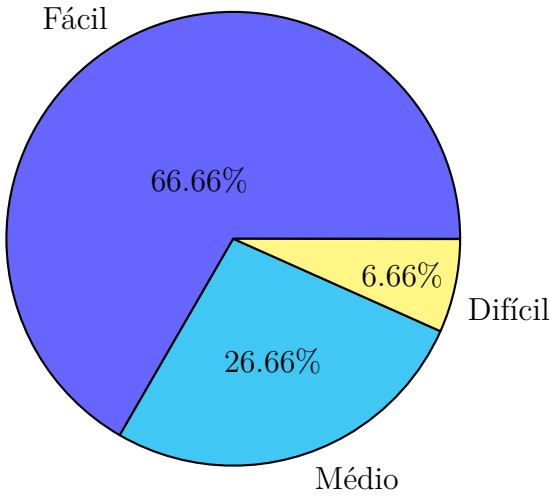


Figura 41 – Gráfico, segundo ano: As atividades apresentadas no jogo estão em qual grau de dificuldade?

A pergunta seguinte verifica se o jogo apresenta os objetivos (mecânicas das fases) de cada fase de maneira objetiva e de fácil entendimento, novamente usando a escala, fácil para objetivos claro e difícil para dificuldades em compreender o objetivo.

No primeiro ano onze alunos acharam que foi fácil compreender o objetivo da fase, três acharam médios e dois acharam difícil de compreender os objetivos

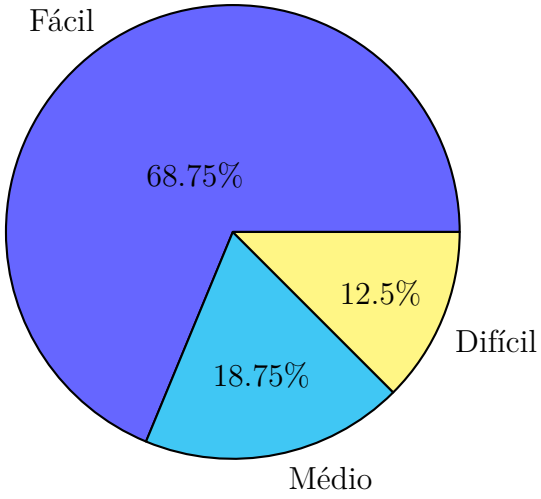


Figura 42 – Gráfico, primeiro ano: Os objetivos foram apresentados de forma clara?

Já os alunos do segundo ano apresentaram uma resposta diferente, sete alunos acharam que foi fácil compreender os objetivos do jogo e oito alunos acharam que a dificuldade de compreender os desafios foi mediana.

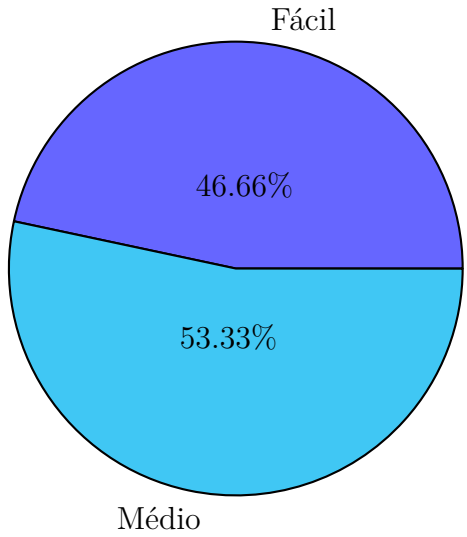


Figura 43 – Gráfico, segundo ano: Os objetivos foram apresentados de forma clara?

A questão seguinte verifica se os alunos acharam o jogo desafiador, analisando o jogo como um todo, fácil significa que foi pouco desafiador e difícil que foi muito desafiador.

No primeiro ano onze alunos acharam o jogo pouco desafiador, dois jogadores acharam a dificuldade média e três alunos acharam a dificilmente desafiador.

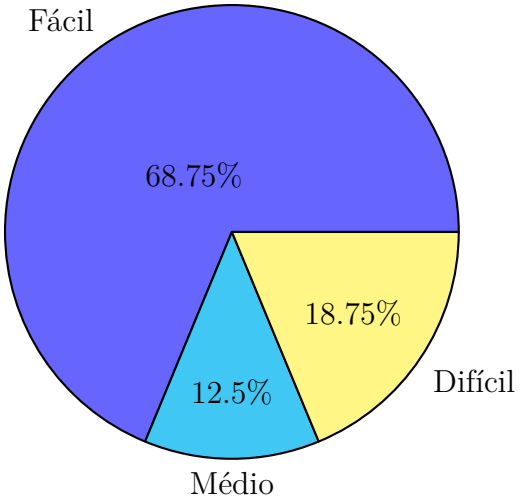


Figura 44 – Gráfico, primeiro ano: O jogo foi desafiador?

No segundo ano sete alunos acharam o jogo pouco desafiador, sete acharam medianamente desafiador e apenas um achou o jogo dificilmente desafiador.

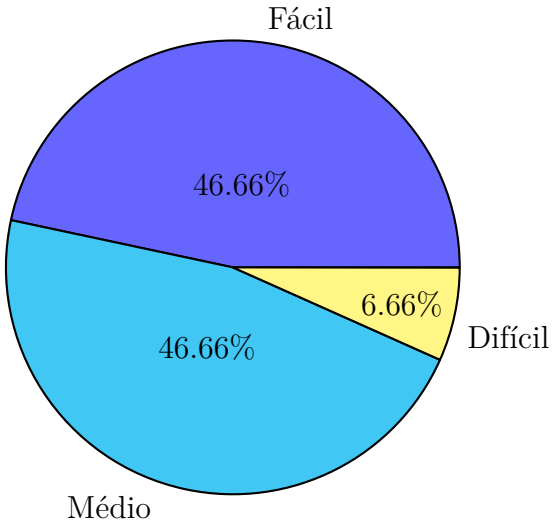


Figura 45 – Gráfico, segundo ano: O jogo foi desafiador?

A quarta e última pergunta da seção pede uma nota geral para a satisfação do jogador em relação ao jogo em si como um todo, a escala aqui é alterada para péssimo, fraco, regular, bom e excelente.

No primeiro ano dois alunos acharam o jogo regular, dois alunos acharam bom e doze alunos acharam excelente.

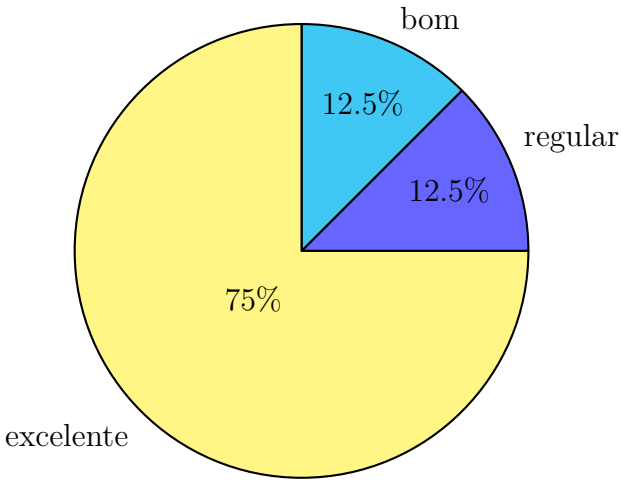


Figura 46 – Gráfico, primeiro ano: Grau de satisfação

Um aluno do segundo ano achou o jogo bom e quatorze alunos acharam excelente.

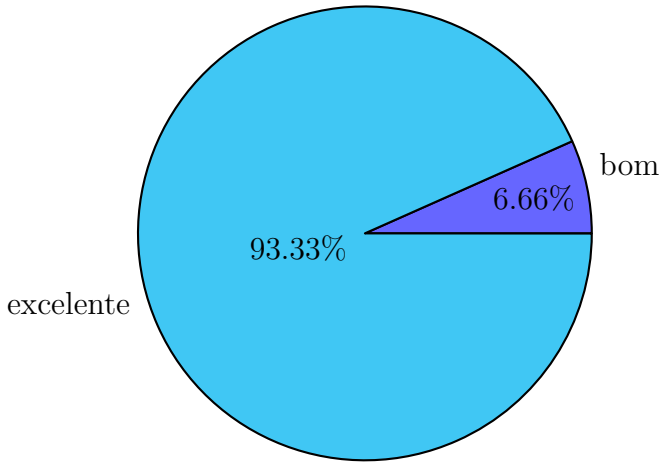


Figura 47 – Gráfico, segundo ano: Grau de satisfação

5.1.3 Terceira seção

A terceira e última seção da entrevista busca criar melhorias no jogo através do *feedback* dos alunos. A primeira pergunta analisa quais dificuldades os alunos tiveram em termos técnicos. No primeiro ano do ensino fundamental dois alunos afirmaram que tiveram dificuldades com a câmera e movimentação do personagem e no segundo ano um aluno informou essa mesma dificuldade.

A segunda pergunta verifica a opinião dos alunos em relação a duração do tempo do jogo.

O primeiro ano teve dois alunos que acharam que o jogo foi muito curto, dez alunos acharam que estava de bom tamanho e quatro alunos acharam que o jogo é longo.

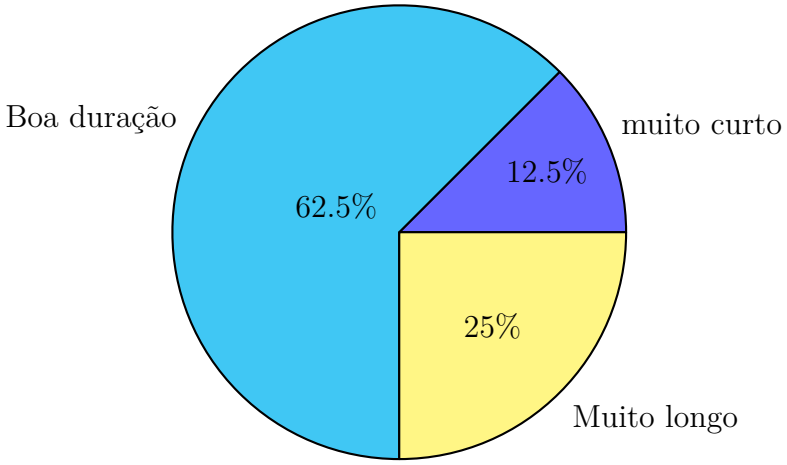


Figura 48 – Gráfico, primeiro ano: Sobre a duração do jogo:

Quatro alunos do segundo ano acharam o jogo curto, dez acharam que estava de bom tamanho e um aluno achou o jogo longo.

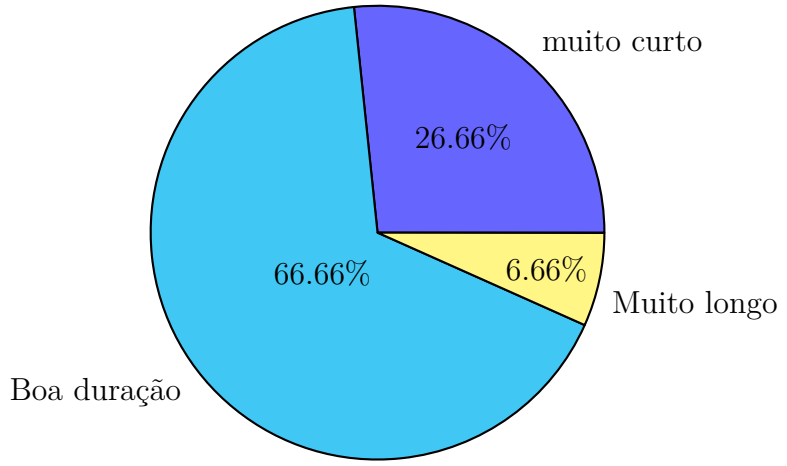


Figura 49 – Gráfico, segundo ano: Sobre a duração do jogo:

A terceira pergunta verifica sugestões dos alunos para melhorar o jogo.
No primeiro ano surgiram três sugestões:

- Níveis de dificuldade: Foi sugerido colocar níveis de dificuldade nas fases para aumentar o nível de desafio.
- Customização: Poder alterar o personagem, gênero, cabelo, cor da pele, etc.
- Personagens não jogáveis: Personagens que possam interagir com o jogador e façam parte das fases, ajudando ou atrapalhando.

No segundo houve duas sugestões:

- Fases mais desafiadoras: alguns alunos do segundo ano acharam algumas fases maçantes

- Melhorar a movimentação do personagem; Alguns alunos apresentaram dificuldades na movimentação do personagem.

A última pergunta da entrevista verifica se os alunos tem interesse em ter contato com mais jogos educativos.

No primeiro ano quinze alunos informaram que tem interesse em jogar mais jogos educativos e um aluno não chegou a uma conclusão sobre o assunto.

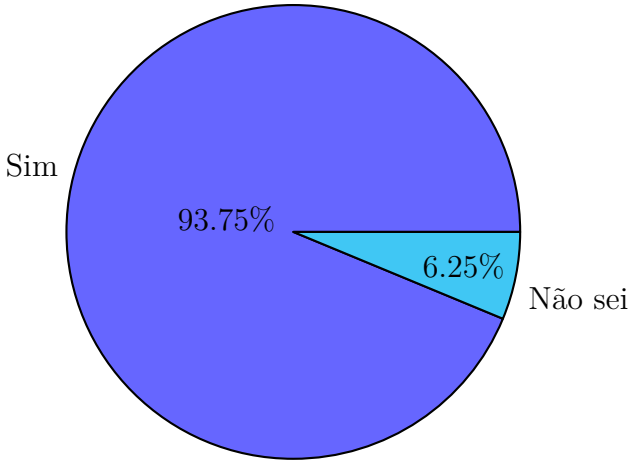


Figura 50 – Gráfico, primeiro ano: Você gostaria de ter contato com mais jogos educativos?

No segundo ano quatorze alunos têm interesse em jogos eletrônicos educacionais e um aluno não tem interesse.

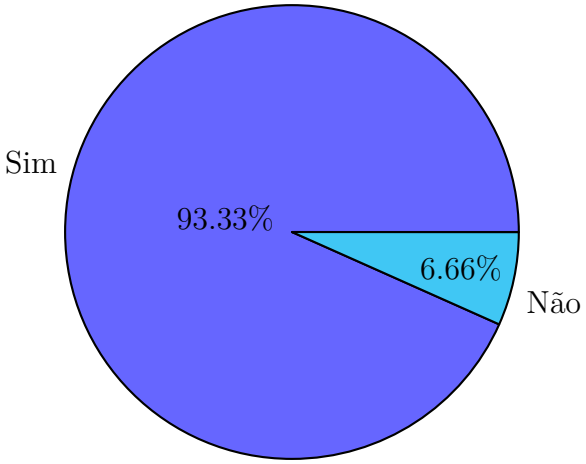


Figura 51 – Gráfico, segundo ano: Você gostaria de ter contato com mais jogos educativos?

Apesar de apenas dois alunos terem apontado a dificuldade em usar a câmera e mover o personagem essa foi uma dificuldade apresentada por todos os alunos em um certo grau.

Os alunos que estão acostumados a jogar (principalmente no computador) conseguiram entender a jogabilidade de maneira rápida, mas a grande maioria dos alunos ficou confusa e com dificuldades ao usar um teclado e *mouse* ao mesmo tempo.

5.2 Análise dos testes pelo desenvolvedor

O jogo foi desenvolvido pensando no uso de uma mão no teclado movendo o personagem e outra no *mouse* controlando a câmera, mas a pesquisa de campo mostrou que os alunos do primeiro e segundo ano tiveram uma dificuldade para usar as duas mãos simultaneamente: a maioria dos alunos usou apenas uma mão durante todo o teste. Também foi analisado que os alunos que já possuem contato com jogos eletrônicos, principalmente os que jogam em um console de *video-game* apresentaram uma fácil adaptação para controlar o personagem.

Entre os alunos que não costumam jogar regularmente foi notado que os alunos do segundo ano tiveram uma facilidade maior para se adaptar aos controles do jogo do que os alunos do primeiro ano.

Tanto os alunos do primeiro ano quanto do segundo apresentam dificuldades em realizar a primeira tarefa de cada fase. Após que realizam a primeira tarefa, eles compreendiam o objetivo da fase e não apresentavam grandes dificuldades nas próximas tarefas da fase. Dois alunos do primeiro ano erraram três vezes seguidas em uma tarefa e sofreram *Game-Over*, no segundo ano, um aluno errou uma tarefa três vezes seguida.

Os alunos que consideraram o jogo longo demais em geral foram os que apresentaram dificuldades na realização das tarefas. E os que acharam muito curto foram os que se adaptaram aos controles e as mecânicas de cada fase de maneira ligeira.

A primeira fase foi considerada entediante por demorar muito e ter pouca ação para os jogadores realizarem. A quinta fase foi a que os alunos menos compreenderam, isso pode ser devido a um *Design* de nível que não deixou claro o objetivo, pela falta de controle do personagem apresentado pelos jogadores, diminuindo o fator exploração do jogo ou por ambos os casos, muitos alunos apresentaram dificuldades em usar o *mouse* para arrastar um objeto e larga-lo no local correto. Além disso alguns alunos descobriram um *bug* na fase 5, onde é possível burlar o sistema de abrir portas para seguir o jogo o que pode ter diminuindo ainda mais a imersão e objetividade da fase.

6 CONCLUSÃO E TRABALHOS FUTUROS

6.1 Conclusão

O objetivo traçado para esse trabalho foi o desenvolvimento de um jogo eletrônico tridimensional com uma proposta educacional sobre os conceitos matemáticos vistos por alunos do primeiro ano do ensino fundamental. Também teve-se como objetivo receber o devido *feedback* acerca do interesse e desempenho dos alunos em jogos educacionais.

O jogo foi idealizado através da criação de um GDD e implementado através do motor de jogos Unity3D, o jogo apresenta tarefas matemáticas com conteúdo que são vistos no primeiro ano do ensino fundamental em um ambiente lúdico tridimensional, a escolha do tridimensional aconteceu pela falta de jogos educacionais escolares nesse gênero.

Após ser testado em duas salas de aula (primeiro e segundo ano do ensino fundamental) foi visto que os alunos possuem interesse em jogos eletrônicos com propostas educacionais. O ambiente do jogo permitia que eles errassem sem uma punição alta.

Segundo a pesquisa realizada a maioria dos alunos joga algum tipo de jogo eletrônico, o ato de jogar se tornou algo cotidiano para muitas pessoas, o uso de jogos eletrônicos no meio educacional é algo que deve ser mais estudado e explorado, pois estes possuem um potencial incrível para o ensino, mas ainda são pouco explorados e usados.

6.2 Trabalhos Futuros

Apesar de poucos alunos terem reclamado do controle do personagem foi visto que esse foi um pequeno problema para os alunos, o fluxo de jogo não fluiu tão bem quanto o esperado, apesar de alguns alunos terem se adaptado bem aos comandos. Uma solução para esse problema é desenvolver novos métodos de controle de personagem sem abandonar o atual, permitindo que o jogador escolha como jogar.

Também seria interessante ter trabalhos usando tecnologias como a realidade virtual a fim de criar uma imersão maior ainda ao jogo.

REFERÊNCIAS

ABT, C. C. ***Serious Games***. 1. ed. New York, USA: University Press of America, 1987. ISBN 0-8191-6147-0.

BLENDER. ***About***. 2019. Disponível em: <https://www.blender.org/about>. Acesso em: 22 set. 2019.

BNCC. ***Base Nacional Comum Curricular***. 2019. Disponível em: <http://basenacionalcomum.mec.gov.br/images/BNCC_EI_EF_110518_versaofinal_site.pdf>. Acesso em: 27 set. 2019.

BNDES. ***Panorama do setor de jogos digitais no Brasil***. 2018. Disponível em: <https://www.bndes.gov.br/wps/portal/site/home/conhecimento/noticias/noticia/jogos-digitais-brasil-infografico>. Acesso em: 12 ago. 2018.

CLEINA, N. ***O que é engine ou motor gráfico?*** 2011. Disponível em: <https://www.tecmundo.com.br/video-game-e-jogos/9263-o-que-e-engine-ou-motor-grafico-.htm>. Acesso em: 23 set. 2019.

DAMIEN ALVAREZ JULIAN, J. J.-P. R. O. D. ***origins of serious games***. *Springer London*, 2011.

GALLEGO, J. P.

A utilização dos jogos como recurso didático no ensino-aprendizagem da matemática — UNESP, 2007.

GCOMPRIS. ***GCompris introdução***. 2019. Disponível em: <https://gcompris.net/wiki/Manual_pt-BR#Introdu.C3.A7.C3.A3o>. Acesso em: 28 set. 2019.

GIFE. ***7ª edição do Relatório De Olho nas Metas apresenta avanços e desafios na educação***. 2017. Disponível em: <https://gife.org.br/7a-edicao-do-relatorio-de-olho-nas-metas-apresenta-avancos-e-desafios-no-campo-educacional/>. Acesso em: 14 out. 2019.

GIMP. ***About Gimp***. 2019. Disponível em: <https://www.gimp.org/about/>. Acesso em: 24 set. 2019.

HALL, T. ***Doom Bible***. 1992. Disponível em: <https://5years.doomworld.com/doombible/doombible.pdf>. Acesso em: 12 out. 2019.

HUIZINGA, J. ***Homo Ludens***. 4. ed. São Paulo, BR: Editora Perspectiva, 2000. ISBN 85-273-0075-3.

KAAP, K. ***Advantages of 3D for Learning***. 2009. Disponível em: <http://karlkapp.com/advantages-of-3d-for-learning/>. Acesso em: 14 out. 2019.

KAPP, K. M. ***The Gamification of Learning and instruction***. 1. ed. San Francisco, USA: Pfeiffer, 2012. ISBN 978-1-118-09634-5.

LATANSIO, F. ***Game Design Document: "As Aventuras de Jackie e Tony"***. 2011. Disponível em: <http://www.dca.fee.unicamp.br/~martino/disciplinas/ia3691s11/grupoA/gdd-ga-v1.pdf>. Acesso em: 13 out. 2019.

MEC. ***Educação é a base***. 2019. Disponível em: <http://basenacionalcomum.mec.gov.br/>. Acesso em: 27 set. 2019.

MELO, I. R. P.

O uso de jogos eletrônicos como ferramenta de ensino: um estudo da suíte de jogos GCompris — UNIFAP, 2012.

NEWZOO. ***2018 Global Games Market***. 2018. Disponível em: <https://newzoo.com/insights/articles/global-games-market-reaches-137-9-billion-in-2018-mobile-games-take-half/>. Acesso em: 13 ago. 2018.

NOVAK, J. ***Desenvolvimento de Games***. 2. ed. [S.l.]: Cengage Learning, 2010. ISBN 978-8522106325.

OLIVEIRA, F. N. ***Game Design Document: Realizando a documentação do seu projeto de Jogo Digital***. 2019. Disponível em: <https://www.fabricadejogos.net/posts/artigo-game-design-document-realizando-a-documentacao-do-seu-projeto-de-jogo-digital/>. Acesso em: 07 out. 2019.

PAULA, B. H. d. ***Jogos digitais como artefatos pedagógicos: o desenvolvimento de jogos digitais como estratégia educacional***. Dissertação (Mestrado) — UNICAMP, 2015.

PEDRO, L. Z. ***Uso de gamificação em ambientes virtuais de aprendizagem para reduzir o problema da externalização de comportamentos indesejáveis***. Dissertação (Mestrado) — USP, 2016.

SILVA, M. P.

Aplicação da inteligência artificial em jogos: Uma abordagem histórica — FACOMP, 2015.

SOLARI, G. ***Além do jogo: a história e o impacto do 'save'***. 2011. Disponível em: <https://www.uol.com.br/start/ultimas-noticias/2011/07/06/alem-do-jogo-a-historia-e-o-impacto-do-save.htm>. Acesso em: 19 set. 2019.

UNITY3D. ***Editor***. 2019. Disponível em: <https://unity3d.com/pt/unity>. Acesso em: 23 set. 2019.

VILARINHO, R. G. **Avaliação de Jogos Eletrônicos Para Uso na Prática Pedagógica: Ultrapassando A Escolha Baseada no Bom Senso**. *Revista Renote*, 2015.

APÊNDICE A – TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

Eu _____ RG: _____
Nome do Pai/Mãe ou Responsável

Abaixo qualificado(a), DECLARO para fins de participação em pesquisa, na condição de representante legal do(a) aluno(a) _____, que fui devidamente esclarecido quanto ao Projeto de Pesquisa intitulado: Desenvolvimento de Um Jogo 3D para o Ensino de Matemática Básica, desenvolvido pelo acadêmico Wesley de Faria Cazelli, do curso de Ciência da Computação e orientado pela prof[a]. Dra. Mercedes Rocío Gonzales Márquez da Universidade Estadual de Mato Grosso do Sul. O objetivo da pesquisa é verificar o interesse dos alunos do primeiro e segundo ano em obterem aprendizado através de jogos eletrônicos educativos, com enfoque em matemática. Após assistirem uma aula experimental, os alunos deverão responder um questionário por meio de entrevista composto por 12 questões estruturadas no referencial teórico. Salientamos que as respostas do questionário não influenciarão na nota do aluno na escola. Não serão divulgados nomes ou quaisquer dados que identifiquem os participantes da pesquisa. DECLARO, que após convenientemente esclarecido pelo pesquisador e ter entendido o que me foi explicado, CONSINTO voluntariamente (que meu dependente legal) participe desta pesquisa.

_____
Assinatura do Pai/Mãe ou Responsável.

Agradecemos a participação na pesquisa. Wesley de Faria Cazelli - Acadêmico

APÊNDICE B – ENTREVISTA: QUESTIONÁRIO

As seguintes perguntas serviram de guia para a entrevista:

1. Você costuma jogar em qual aparelho?
    a) Console ()
    b) Computador ()
    c) Celular ()
    d) Console portátil ()
    e) Outro () _____
    f) Não costumo jogar ()
2. Qual gênero de jogo eletrônico você mais joga?
    a) Ação/Aventura/RPG ()
    b) Estratégia ()
    c) Educativo ()
    d) Quebra-cabeça ()
    e) Luta ()
    f) Tiro em primeira pessoa ()
    g) Construção/Gerenciamento ()
    h) Musical/Ritmo ()
    i) Esportes ()
    j) Corrida ()
    k) Outro () _____
3. Como você costuma jogar?
    a) *Offline* sozinho ()
    b) *Offline* com família/amigos ()
    c) *Online* sozinho ()
    d) *Online* com família/amigos ()

4. Você já jogou algum jogo educativo antes?

   a) Sim ()

   b) Não ()

5. As atividades apresentadas no jogo, como contagem de objetos, soma e subtração de cubos e comparação entre valores, estão em qual grau de dificuldade?

   a) Fácil ()

   b) Médio ()

   c) Difícil ()

6. O jogo apresenta os objetivos de forma clara? qual o grau de dificuldade você teve para entender os objetivos?

   a) Fácil ()

   b) Médio ()

   c) Difícil ()

7. O jogo foi desafiador? você considera que o jogo possui qual grau de dificuldade?

   a) Fácil ()

   b) Médio ()

   c) Difícil ()

8. O jogo motiva o jogador? Te mantém entretido do início ao fim? Informe o grau de satisfação que você teve ao jogar:

   a) Péssimo ()

   b) Fraco ()

   c) Regular ()

   d) Bom ()

   e) Excelente ()

9. Você teve alguma dificuldade acentuada durante o jogo?

   a) Controlar o personagem e/ou a câmera ()

   b) Compreender o objetivo de uma ou mais fases ()

   c) Leitura ()

   d) Outro () _____

   e) Não tive dificuldades ()

10. Sobre a duração do jogo, você achou:

    a) Curto ()

    b) Satisfatório ()

    c) Longo ()

11. Tem algo que você gostaria de mudar, retirar ou acrescentar ao jogo?

    a) Adicionar *Multiplayer* ()

    b) Personagem customizável ()

    c) Novos métodos de controlar o personagem e/ou câmera ()

    d) Outro () _____

12. Você gostaria de ter contato com mais jogos educativos?

    a) Sim ()

    b) Não ()

    c) Não sei/talvez ()

APÊNDICE C – CLASSE SOM

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.Audio;
5
6  [System.Serializable]
7  public class Sound
8  {
9      public string name;
10     public AudioClip clip;
11     [Range(0f, 1f)]
12     public float volume;
13     [Range(.1f, 3f)]
14     public float pitch;
15     public bool loop;
16     [HideInInspector]
17     public AudioSource source;
18 }
```

APÊNDICE D – *SCRIPT* GERENCIADOR DE SOM

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.Audio;
5  using System;
6  using UnityEngine.SceneManagement;
7
8  public class AudioManager : MonoBehaviour
9  {
10     public Sound[] sounds;
11
12     private void Start()
13     {
14         if (SceneManager.GetActiveScene().name == "Level5")
15         {
16             Play("Sci-fi Main Theme");
17         }
18         else
19         {
20             Play("Main Theme");
21         }
22     }
23
24     void Awake()
25     {
26         foreach (Sound s in sounds)
27         {
28             s.source = gameObject.AddComponent<AudioSource>();
29             s.source.clip = s.clip;
30             s.source.volume = s.volume;
31             s.source.pitch = s.pitch;
32             s.source.loop = s.loop;
33         }
```

```
34      }
35
36      public void Play(string name)
37      {
38          Sound s = Array.Find(sounds, sound => sound.name == name
                );
39          if (s == null)
40          {
41              return;
42          }
43          s.source.Play();
44      }
45
46      public void PlayOneShot(string name)
47      {
48          Sound s = Array.Find(sounds, sound => sound.name == name
                );
49          if (s == null)
50          {
51              return;
52          }
53          s.source.PlayOneShot(s.clip);
54      }
55
56      public void Stop(string name)
57      {
58          Sound s = Array.Find(sounds, sound => sound.name == name
                );
59          if (s == null)
60          {
61              return;
62          }
63          s.source.Stop();
64      }
65  }
```

APÊNDICE E – *SCRIPT* MENU PRINCIPAL

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.SceneManagement;
5
6  public class MainMenuScript : MonoBehaviour {
7      [SerializeField]
8      private GameObject _mainPanel;
9      [SerializeField]
10     private GameObject _chooseLevelPanel;
11     private AudioManager _audioManager;
12
13     private void Start()
14     {
15         Cursor.visible = true;
16         Cursor.lockState = CursorLockMode.None;
17         _audioManager = GameObject.Find("AudioManager").
               GetComponent<AudioManager>();
18     }
19
20     public void Play()
21     {
22         SceneManager.LoadScene("Level1");
23     }
24
25     public void Exit()
26     {
27         Application.Quit();
28     }
29
30     public void CloseMainPanel()
31     {
32         _mainPanel.SetActive(false);
```

```
33          _chooseLevelPanel.SetActive(true);
34      }
35
36      public void CloseChooseLevelPanel()
37      {
38          _chooseLevelPanel.SetActive(false);
39          _mainPanel.SetActive(true);
40      }
41
42      public void PlayHoverMouseSound()
43      {
44          _audioManager.PlayOneShot("UI␣Mouse␣Hover");
45      }
46
47      public void PlayOnClick()
48      {
49          _audioManager.Play("UI␣Mouse␣Click");
50      }
51  }
```

APÊNDICE F – *SCRIPT* MENU ESCOLHA DE FASE

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.SceneManagement;
5
6  public class LevelsChoose : MonoBehaviour
7  {
8      public void GoToLevel1()
9      {
10         Time.timeScale = 1;
11         SceneManager.LoadScene("Level1");
12     }
13
14     public void GoToLevel2()
15     {
16         Time.timeScale = 1;
17         SceneManager.LoadScene("Level2");
18     }
19
20     public void GoToLevel3()
21     {
22         Time.timeScale = 1;
23         SceneManager.LoadScene("Level3");
24     }
25
26     public void GoToLevel4()
27     {
28         Time.timeScale = 1;
29         SceneManager.LoadScene("Level4");
30     }
31
32     public void GoToLevel5()
33     {
```

```
34            Time.timeScale = 1;
35            SceneManager.LoadScene("Level5");
36        }
37    }
```

APÊNDICE G – *SCRIPT* GERENCIADOR DE UI

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5  using UnityEngine.SceneManagement;
6
7  public class UIManager : MonoBehaviour {
8      [SerializeField]
9      private Image _menu;
10     [SerializeField]
11     private GameObject[] _buttons;
12     private bool _isPaused = false;
13     [SerializeField]
14     private RawImage[] _hearts;
15     [SerializeField]
16     private GameObject _gameOverPanel;
17     private AudioManager _audioManager;
18     [SerializeField]
19     private GameObject _chooseLevelCanvas;
20
21     private void Start()
22     {
23         _audioManager = GameObject.Find("AudioManager").
               GetComponent<AudioManager>();
24     }
25
26     void Update () {
27         if (Input.GetKeyDown(KeyCode.P) || Input.GetKeyDown(
               KeyCode.Escape))
28         {
29             if (_isPaused == false)
30             {
31                 PauseGame();
```

```
32              }
33              else
34              {
35                  ResumeGame();
36              }
37          }
38          }
39
40      private void PauseGame()
41      {
42          Time.timeScale = 0;
43          _menu.gameObject.SetActive(true);
44          _isPaused = true;
45          Cursor.visible = true;
46          Cursor.lockState = CursorLockMode.None;
47          _menu.enabled = true;
48          for (int i = 0; i < 4; i++)
49          {
50              _buttons[i].SetActive(true);
51          }
52      }
53
54      public void ResumeGame()
55      {
56          _isPaused = false;
57          _menu.gameObject.SetActive(false);
58          Time.timeScale = 1;
59          Cursor.visible = false;
60          Cursor.lockState = CursorLockMode.Locked;
61          _menu.enabled = false;
62          for (int i = 0; i < 4; i++)
63          {
64              _buttons[i].SetActive(false);
65          }
66      }
67
68      public void ReturnMainMenu()
69      {
70          Time.timeScale = 1;
```

```
71          SceneManager.LoadScene("MainMenu");
72      }
73
74      public void HeartDown(int heart)
75      {
76          _hearts[heart].color = new Color(1f, 0, 0, 0.2f);
77      }
78
79      public void HeartUp(int heart)
80      {
81          _hearts[heart].color = new Color(1f, 1f, 1f, 1f);
82      }
83
84      public void GameOverPanelOn()
85      {
86          _gameOverPanel.gameObject.SetActive(true);
87          Cursor.visible = true;
88          Cursor.lockState = CursorLockMode.None;
89      }
90
91      public void GameOverPanelOff()
92      {
93          _gameOverPanel.gameObject.SetActive(false);
94          Cursor.visible = false;
95          Cursor.lockState = CursorLockMode.Locked;
96      }
97
98      public void GameOverContinueButton()
99      {
100         SceneManager.LoadScene(SceneManager.GetActiveScene().
                buildIndex);
101     }
102
103     public void RestartLevel() //Reinicia o level
104     {
105         Time.timeScale = 1;
106         Scene scene = SceneManager.GetActiveScene();
107         SceneManager.LoadScene(scene.buildIndex);
108     }
```

```
109
110     public void PlayHoverMouseSound()
111     {
112         _audioManager.PlayOneShot("UI Mouse Hover");
113     }
114
115     public void PlayOnClick()
116     {
117         _audioManager.Play("UI Mouse Click");
118     }
119
120     public void ChooseLevelOn()
121     {
122         _chooseLevelCanvas.SetActive(true);
123     }
124
125     public void ChooseLevelOff()
126     {
127         _chooseLevelCanvas.SetActive(false);
128     }
129 }
```

APÊNDICE H – *SCRIPT* MENU PRÓXIMA FASE

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.SceneManagement;
5
6  public class CanvasNextLevelScript : MonoBehaviour
7  {
8      [SerializeField]
9      private GameObject _panelChooseLevel;
10     [SerializeField]
11     private GameObject[] _mainButtons;
12
13     public void NextLevel()
14     {
15         if (SceneManager.GetActiveScene() == SceneManager.
               GetSceneByName("Level1"))
16         {
17             SceneManager.LoadScene("Level2");
18         }
19         else if (SceneManager.GetActiveScene() == SceneManager.
               GetSceneByName("Level2"))
20         {
21             SceneManager.LoadScene("Level3");
22         }
23         else if (SceneManager.GetActiveScene() == SceneManager.
               GetSceneByName("Level3"))
24         {
25             SceneManager.LoadScene("Level4");
26         }
27         else if (SceneManager.GetActiveScene() == SceneManager.
               GetSceneByName("Level4"))
28         {
29             SceneManager.LoadScene("Level5");
```

```
30          }
31      }
32
33      public void MainMenu()
34      {
35          SceneManager.LoadScene("MainMenu");
36      }
37
38      public void ChooseLevel()
39      {
40          for (int i = 0; i < 3; i++)
41          {
42              _mainButtons[i].SetActive(false);
43          }
44          _panelChooseLevel.SetActive(true);
45      }
46
47      public void ReturnEndPanel()
48      {
49          for (int i = 0; i < 3; i++)
50          {
51              _mainButtons[i].SetActive(true);
52          }
53          _panelChooseLevel.SetActive(false);
54      }
55  }
```

APÊNDICE I – *SCRIPT* CONTROLADOR DE ANIMAÇÃO DOS CORAÇÕES (UI)

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class HeartAnimatorController : MonoBehaviour {
6      private PlayerScript _player;
7      [SerializeField]
8      private GameObject[] _hearts;
9      private Animator _anim;
10     private Animator _anim2;
11     private Animator _anim3;
12
13         void Start () {
14         _player = GameObject.Find("Player").GetComponent<
               PlayerScript>();
15         _anim = GameObject.Find("Heart1").GetComponent<Animator
               >();
16         _anim2 = GameObject.Find("Heart2").GetComponent<Animator
               >();
17         _anim3 = GameObject.Find("Heart3").GetComponent<Animator
               >();
18     }
19
20         void Update () {
21         if (_player.life == 3)
22         {
23             _anim3.SetBool("Pulse", true);
24             _anim2.SetBool("Pulse", false);
25             _anim.SetBool("Pulse", false);
26         }else if (_player.life == 2)
27         {
```

```
28              _anim3.SetBool("Pulse", false);
29              _anim2.SetBool("Pulse", true);
30              _anim.SetBool("Pulse", false);
31          }else if (_player.life == 1)
32          {
33              _anim3.SetBool("Pulse", false);
34              _anim2.SetBool("Pulse", false);
35              _anim.SetBool("Pulse", true);
36          }
37          else
38          {
39              _anim3.SetBool("Pulse", false);
40              _anim2.SetBool("Pulse", false);
41              _anim.SetBool("Pulse", false);
42          }
43          }
44  }
```

APÊNDICE J – *SCRIPTS* DA PRIMEIRA FASE

J.1 *Script* das placas

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  public class SignScript : MonoBehaviour {
7      [SerializeField]
8      private Image _eKeyboard;
9      [SerializeField]
10     private GameObject _signPanel;
11     private bool _panelActive = false;
12     private PlayerScript _playerScript;
13     private CameraController _cameraController;
14     [SerializeField]
15     private RawImage[] _images;
16     [SerializeField]
17     private float _time;
18     private float _timer = 0;
19     private bool _timerBool = false;
20     private bool _timeBool = false;
21     private float _timeRate = 1.5f;
22     private int _timeCount = 0;
23     [SerializeField]
24     private Text _numbers;
25     [SerializeField]
26     private int _signID;
27     [SerializeField]
28     private float _R; //Red
29     [SerializeField]
30     private float _G; //Green
31     [SerializeField]
```

```
32      private float _B; //Blue
33      private float _rRate;
34      private float _gRate;
35      private float _bRate;
36      private Level1Manager _level1Manager;
37      private bool _signRepead = false;
38      private AudioManager _signAudioManager;
39      private AudioManager _audioManager;
40      [SerializeField]
41      private GameObject _3DNumber;
42      private bool _alreadyVisited = false;
43      private bool _alreadyPlayedSound = false;
44
45      void Start()
46      {
47          _playerScript = GameObject.Find("Player").GetComponent<
                PlayerScript>();
48          _cameraController = GameObject.Find("Main Camera").
                GetComponent<CameraController>();
49          _time *= _signID;
50          _time += 1.5f;
51          _level1Manager = GameObject.Find("LevelManager").
                GetComponent<Level1Manager>();
52          if (_R != 0)
53          {
54              _R = _R / 255;
55          }
56          if (_G != 0)
57          {
58              _G = _G / 255;
59          }
60          if (_B != 0)
61          {
62              _B = _B / 255;
63          }
64          if (_R != 0)
65          {
66              _R = _R / _signID;
67          }
```

```
68          if (_G != 0)
69          {
70              _G = _G / _signID;
71          }
72          if (_B != 0)
73          {
74              _B = _B / _signID;
75          }
76          _rRate = _R;
77          _gRate = _G;
78          _bRate = _B;
79          _signAudioManager = GameObject.Find("SignAudioManager").
                GetComponent<AudioManager>();
80          _audioManager = GameObject.Find("AudioManager").
                GetComponent<AudioManager>();
81      }
82
83      void Update()
84      {
85          if (_timerBool == true )
86          {
87              _timer += Time.deltaTime;
88              if ((_timer >= _timeRate) && (_timeCount < _signID))
89              {
90                  _images[_timeCount].color = new Color(1f, 1f, 1f
                        , 1f);
91                  int soundNumber = _timeCount + 1;
92                  string soundString;
93                  soundString = soundNumber.ToString();
94                  _numbers.color = new Color(_R, _G, _B, 1f);
95                  _R += _rRate;
96                  _G += _gRate;
97                  _B += _bRate;
98                  _numbers.text = _timeCount+1 + "";
99                  _signAudioManager.PlayOneShot(soundString);
100                 _timeCount++;
101                 _timeRate += 1.5f;
102             }
103         }
```

```
104     }
105
106     void OnTriggerEnter(Collider other)
107     {
108         if (other.tag == "Player")
109         {
110             ShowEKeyboard();
111         }
112     }
113
114     void OnTriggerExit(Collider other)
115     {
116         if (other.tag == "Player")
117         {
118             HideEKeyboard();
119             HidePanel();
120         }
121     }
122
123     private void OnTriggerStay(Collider other)
124     {
125         if ((other.tag == "Player") && Input.GetMouseButton(0))
126         {
127             if (_panelActive == false)
128             {
129                 _audioManager.PlayOneShot("Page Sound");
130                 ShowPanel();
131                 HideEKeyboard();
132                 _playerScript.canMove = false;
133                 _cameraController.active = false;
134                 _timer = 0f;
135                 _timerBool = true;
136                 StartCoroutine(WaitTime());
137                 if (_signRepead == false)
138                 {
139                     _signRepead = true;
140                 }
141             }
142         }
```

```
143         else if (other.tag == "Player")
144         {
145             if (_timeBool == true)
146             {
147                 _3DNumber.SetActive(true);
148                 _playerScript.canMove = true;
149                 _cameraController.active = true;
150                 HidePanel();
151                 ShowEKeyboard();
152                 _timeBool = false;
153                 _timeRate = 1.5f;
154                 if (_alreadyVisited == false)
155                 {
156                     _audioManager.PlayOneShot("Magical Number");
157                     _level1Manager.signsRead++;
158                     _level1Manager.UpdateLevel1UI();
159                     _level1Manager.EndLevelCheck();
160                     _alreadyVisited = true;
161                 }
162             }
163         }
164     }
165 
166     private void ShowPanel()
167     {
168         _signPanel.gameObject.SetActive(true);
169         _panelActive = true;
170     }
171 
172     private void HidePanel()
173     {
174         _signPanel.gameObject.SetActive(false);
175         _panelActive = false;
176         for (int i = 0; i < _signID; i++)
177         {
178             _images[i].color = new Color(0f, 0f, 0f, 1f);
179         }
180         _timerBool = false;
181         _timeCount = 0;
```

```
182     }
183
184     private void ShowEKeyboard()
185     {
186         _eKeyboard.gameObject.SetActive(true);
187     }
188
189     private void HideEKeyboard()
190     {
191         _eKeyboard.gameObject.SetActive(false);
192     }
193
194     private IEnumerator WaitTime()
195     {
196         yield return new WaitForSeconds(_time);
197         _timeBool = true;
198     }
199 }
```

J.2 *Script* Gerenciador da fase 1

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Level1Manager : MonoBehaviour {
6      private int _goal = 9;
7      public int signsRead = 0;
8      private Level1UI _level1UI;
9      [SerializeField]
10     private GameObject _canvasNextLevel;
11     private PlayerScript _playerScript;
12     private CameraController _cameraController;
13     private AudioManager _audioManager;
14
15     void Start()
16     {
17         _level1UI = GameObject.Find("CanvasLevel").GetComponent<
               Level1UI>();
```

```
18          _playerScript = GameObject.Find("Player").GetComponent<
                PlayerScript>();
19          _cameraController = GameObject.Find("Main Camera").
                GetComponent<CameraController>();
20          _audioManager = GameObject.Find("AudioManager").
                GetComponent<AudioManager>();
21      }
22
23      public void UpdateLevel1UI()
24      {
25          _level1UI.UpdateTextCount();
26      }
27
28      public void EndLevelCheck()
29      {
30          if (signsRead == _goal)
31          {
32              _audioManager.Stop("Main Theme");
33              _audioManager.PlayOneShot("Victory");
34              _playerScript.canMove = false;
35              _cameraController.active = false;
36              Time.timeScale = 1;
37              Cursor.visible = true;
38              Cursor.lockState = CursorLockMode.None;
39              _canvasNextLevel.SetActive(true);
40          }
41      }
42  }
```

APÊNDICE K – *SCRIPTS* DA SEGUNDA FASE

K.1 *Script* pedestal

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  public class RockBaseScript : MonoBehaviour {
7      [SerializeField]
8      private Canvas _rockCanvas;
9      [SerializeField]
10     private bool _rightOrWrong;
11     [SerializeField]
12     private GameObject _wrongSphere;
13     [SerializeField]
14     private GameObject _rightSphere;
15     [SerializeField]
16     private GameObject _defaultSphere;
17     private Level2Manager _level2Manager;
18     private bool _wasPressedBefore = false;
19     private PlayerScript _playerScript;
20     private AudioManager _audioManager;
21     [SerializeField]
22     private GameObject _selfArrow;
23     [SerializeField]
24     private GameObject _arrow1;
25     [SerializeField]
26     private GameObject _arrow2;
27
28     private void Start()
29     {
30         _level2Manager = GameObject.Find("LevelManager").
               GetComponent<Level2Manager>();
```

```
31          _playerScript = GameObject.Find("Player").GetComponent<
                PlayerScript>();
32          _audioManager = GameObject.Find("AudioManager").
                GetComponent<AudioManager>();
33      }
34
35      private void Update()
36      {
37          if (_playerScript.life < 1)
38          {
39              HideEKey();
40          }
41      }
42
43      private void OnTriggerStay(Collider other)
44      {
45          if ((Input.GetKey(KeyCode.E) || Input.GetMouseButton(0))
                && _rightOrWrong == true)
46          {
47              _defaultSphere.gameObject.SetActive(false);
48              _rightSphere.gameObject.SetActive(true);
49              _selfArrow.SetActive(false);
50              _arrow1.SetActive(false);
51              _arrow2.SetActive(false);
52              if (_wasPressedBefore == false)
53              {
54                  if (_playerScript.life < 3)
55                  {
56                      _playerScript.LifeUp();
57                  }
58                  _audioManager.Play("Right");
59                  _level2Manager.rightChoice += 1;
60                  _wasPressedBefore = true;
61                  _level2Manager.EndLevelCheck();
62              }
63
64          } else if ((Input.GetKey(KeyCode.E) || Input.
                GetMouseButton(0)) && _rightOrWrong == false)
65          {
```

```
66                  _defaultSphere.gameObject.SetActive(false);
67                  _wrongSphere.gameObject.SetActive(true);
68                  _selfArrow.SetActive(false);
69                  if (_wasPressedBefore == false)
70                  {
71                      _audioManager.Play("Wrong");
72                      _playerScript.Damage();
73                      _wasPressedBefore = true;
74                  }
75              }
76          }
77
78          private void OnTriggerEnter(Collider other)
79          {
80              if (other.tag == "Player")
81              {
82                  ShowEKey();
83              }
84          }
85
86          private void OnTriggerExit(Collider other)
87          {
88              if (other.tag == "Player")
89              {
90                  HideEKey();
91              }
92          }
93
94          private void ShowEKey()
95          {
96              _rockCanvas.gameObject.SetActive(true);
97          }
98
99          private void HideEKey()
100         {
101             _rockCanvas.gameObject.SetActive(false);
102         }
103 }
```

K.2 *Script* gerenciador da fase 2

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Level2Manager : MonoBehaviour {
6      private int _goal = 3;
7      public int rightChoice = 0;
8      [SerializeField]
9      private GameObject _canvasNextLevel;
10     private PlayerScript _playerScript;
11     private CameraController _cameraController;
12     [SerializeField]
13     private GameObject[] _canvasToBeClosed;
14     private AudioManager _audioManager;
15
16     private void Start()
17     {
18         _playerScript = GameObject.Find("Player").GetComponent<
               PlayerScript>();
19         _cameraController = GameObject.Find("Main Camera").
               GetComponent<CameraController>();
20         _audioManager = GameObject.Find("AudioManager").
               GetComponent<AudioManager>();
21     }
22
23     public void EndLevelCheck()
24     {
25         if (rightChoice == _goal)
26         {
27             for (int i = 0; i < 12; i++)
28             {
29                 _canvasToBeClosed[i].SetActive(false);
30             }
31             _playerScript.canMove = false;
32             _cameraController.active = false;
33             Time.timeScale = 1;
34             Cursor.visible = true;
```

```
35                  Cursor.lockState = CursorLockMode.None;
36                  _canvasNextLevel.SetActive(true);
37                  _audioManager.Stop("Main Theme");
38                  _audioManager.Play("Victory");
39              }
40          }
41      }
```

APÊNDICE L – *SCRIPTS* DA TERCEIRA FASE

L.1 *Script* dos botões

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class GroundButton : MonoBehaviour {
6      [SerializeField]
7      private int _buttonID;
8      [SerializeField]
9      private bool _pressed = false;
10     private float _speed = 0.1f;
11     private float _downSpeed = 0.5f;
12     private Level3Manager _level3Manager;
13     private PlayerScript _playerScript;
14     private bool _buttonAlreadyPressed = false;
15     [SerializeField]
16     private int _buttonArea;
17     private AudioManager _audioManager;
18     [SerializeField]
19     private GameObject _yellowNumber;
20     [SerializeField]
21     private GameObject _secondCollorNumber;
22
23         void Start () {
24         _level3Manager = GameObject.Find("LevelManager").
               GetComponent<Level3Manager>();
25         _playerScript = GameObject.Find("Player").GetComponent<
               PlayerScript>();
26         _audioManager = GameObject.Find("AudioManager").
               GetComponent<AudioManager>();
27     }
28
```

```
29          void Update () {
30
31          if ( (_pressed == true) && (transform.position.y >
                -0.25))
32          {
33
34              transform.Translate(-Vector3.forward * _downSpeed *
                    Time.deltaTime);
35          }
36
37          if ((_pressed == false) && (transform.position.y <
                -0.06))
38          {
39              transform.Translate(Vector3.forward * _speed * Time.
                    deltaTime);
40          }
41          }
42
43      private void OnTriggerEnter (Collider other)
44      {
45          if (other.tag == "Player")
46          {
47              _pressed = true;
48              if (_buttonID == 1 && _level3Manager.stage == 1 &&
                    _buttonArea == 1 &&_buttonAlreadyPressed == false
                    )
49              {
50                  _playerScript.LifeUp();
51                  _audioManager.PlayOneShot("Right");
52                  _yellowNumber.SetActive(false);
53                  _secondCollorNumber.SetActive(true);
54              }else if (_buttonID != 1 && _level3Manager.stage ==
                    1 && _buttonAlreadyPressed == false &&
                    _buttonArea == 1)
55              {
56                  _playerScript.Damage();
57                  _audioManager.PlayOneShot("Wrong");
58                  _yellowNumber.SetActive(false);
59                  _secondCollorNumber.SetActive(true);
```

```
60              }
61              if (_buttonID == 8 && _level3Manager.stage == 2 &&
                    _buttonArea == 2 && _buttonAlreadyPressed ==
                    false )
62              {
63                  _playerScript.LifeUp();
64                  _audioManager.PlayOneShot("Right");
65                  _yellowNumber.SetActive(false);
66                  _secondCollorNumber.SetActive(true);
67              }
68              else if (_buttonID != 8 && _level3Manager.stage == 2
                    && _buttonAlreadyPressed == false && _buttonArea
                    == 2)
69              {
70                  _playerScript.Damage();
71                  _audioManager.PlayOneShot("Wrong");
72                  _yellowNumber.SetActive(false);
73                  _secondCollorNumber.SetActive(true);
74              }
75              if (_buttonID != 10 && _level3Manager.stage == 3 &&
                    _buttonAlreadyPressed == false && _buttonArea ==
                    3)
76              {
77                  _playerScript.Damage();
78                  _audioManager.PlayOneShot("Wrong");
79                  _yellowNumber.SetActive(false);
80                  _secondCollorNumber.SetActive(true);
81              }
82              else if (_buttonID != 10 && _level3Manager.stage ==
                    3 && _buttonAlreadyPressed == false &&
                    _buttonArea == 3)
83              {
84                  _yellowNumber.SetActive(false);
85                  _secondCollorNumber.SetActive(true);
86              }
87              _buttonAlreadyPressed = true;
88              _level3Manager.ButtonID(_buttonID);
89          }
90      }
```

```
91
92      private void OnTriggerExit(Collider other)
93      {
94          if (other.tag == "Player")
95          {
96              _pressed = false;
97          }
98      }
99  }
```

L.2 *Script* gerenciador da fase 3

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class Level3Manager : MonoBehaviour
6   {
7       public int stage;
8       [SerializeField]
9       private GameObject[] _bridge1;
10      [SerializeField]
11      private GameObject[] _bridge2;
12      private float _timer = 0;
13      [SerializeField]
14      private GameObject _canvasNextLevel;
15      private PlayerScript _playerScript;
16      private CameraController _cameraController;
17      [SerializeField]
18      private GameObject _bridge1InvisibleWall;
19      [SerializeField]
20      private GameObject _bridge2InvisibleWall;
21      private bool _pressed = false;
22      private AudioManager _audioManager;
23      private bool _playVictory = false;
24
25      void Start()
26      {
27          stage = 1;
```

```
28          _playerScript = GameObject.Find("Player").GetComponent<
                PlayerScript>();
29          _cameraController = GameObject.Find("Main Camera").
                GetComponent<CameraController>();
30          _audioManager = GameObject.Find("AudioManager").
                GetComponent<AudioManager>();
31      }
32
33      public void ButtonID(int id)
34      {
35          TestAnswer(id);
36      }
37
38      void TestAnswer(int answer)
39      {
40          if ((answer == 1) && (stage == 1))
41          {
42              stage = 2;
43              _bridge1InvisibleWall.SetActive(false);
44              StartCoroutine(WaitTime());
45          }
46          if ((answer == 8) && (stage == 2))
47          {
48              stage = 3;
49              _bridge2InvisibleWall.SetActive(false);
50              StartCoroutine(WaitTime());
51          }
52          if ((answer == 10) && (stage == 3))
53          {
54              _playerScript.canMove = false;
55              _cameraController.active = false;
56              Cursor.visible = true;
57              Cursor.lockState = CursorLockMode.None;
58              if (_playVictory == false)
59              {
60                  _audioManager.PlayOneShot("Victory");
61                  _playVictory = true;
62              }
63              _canvasNextLevel.SetActive(true);
```

```
64                 stage = 3;
65             }
66         }
67
68         private IEnumerator WaitTime()
69         {
70             for (int i = 0; i < 5; i++)
71             {
72                 yield return new WaitForSeconds(0.75f);
73                 if (stage == 2)
74                 {
75                     _audioManager.PlayOneShot("Metal Bridge");
76                     _bridge1[i].SetActive(true);
77                 }else if (stage == 3)
78                 {
79                     _audioManager.PlayOneShot("Metal Bridge");
80                     _bridge2[i].SetActive(true);
81                 }
82             }
83         }
84 }
```

APÊNDICE M – *SCRIPTS* DA QUARTA FASE

M.1 *Script* painel

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  public class PanelScript : MonoBehaviour {
7      [SerializeField]
8      private Image _whiteMouseButton;
9      [SerializeField]
10     private Camera _mainCamera;
11     [SerializeField]
12     private Camera _secondaryCamera;
13     [SerializeField]
14     private Text _textButton;
15     [SerializeField]
16     private GameObject[] _cubes;
17     private PlayerScript _playerScript;
18     private bool _panel = false;
19     [SerializeField]
20     private GameObject _panelPanel;
21     [SerializeField]
22     private int _count;
23     private AudioManager _audioManager;
24
25     void Start () {
26         _playerScript = GameObject.Find("Player").GetComponent<
               PlayerScript>();
27         _audioManager = GameObject.Find("AudioManager").
               GetComponent<AudioManager>();
28     }
29
```

```
30      private void OnTriggerEnter(Collider other)
31      {
32          if (other.tag == "Player")
33          {
34              _whiteMouseButton.gameObject.SetActive(true);
35          }
36      }
37
38      private void OnTriggerExit(Collider other)
39      {
40          if (other.tag == "Player")
41          {
42              _whiteMouseButton.gameObject.SetActive(false);
43          }
44      }
45
46      private void OnTriggerStay(Collider other)
47      {
48          if ((other.tag == "Player") && (Input.GetKeyDown(KeyCode
                .E) || Input.GetMouseButtonDown(0)))
49          {
50              if (_panel == false) {
51                  _panel = true;
52                  _playerScript.canMove = false;
53                  _whiteMouseButton.gameObject.SetActive(false);
54                  _secondaryCamera.enabled = true;
55                  _mainCamera.enabled = false;
56                  ShowPanel();
57                  _audioManager.PlayOneShot("Computer␣On");
58                  Cursor.visible = true;
59                  Cursor.lockState = CursorLockMode.None;
60              }
61              else if(other.tag == "Player" && (Input.GetKeyDown(
                    KeyCode.E)))
62              {
63                  ClosePanel();
64                  _audioManager.PlayOneShot("Computer␣Off");
65              }
66          }
```

```
67      }
68
69      private void ShowPanel()
70      {
71          _panelPanel.gameObject.SetActive(true);
72      }
73
74      private void HidePanel()
75      {
76          _panelPanel.gameObject.SetActive(false);
77      }
78
79      public void ClosePanel()
80      {
81          _panel = false;
82          _playerScript.canMove = true;
83          _whiteMouseButton.gameObject.SetActive(false);
84          _secondaryCamera.enabled = false;
85          _mainCamera.enabled = true;
86          HidePanel();
87          _audioManager.PlayOneShot("Computer Off");
88          Cursor.visible = false;
89          Cursor.lockState = CursorLockMode.Locked;
90      }
91  }
```

M.1.1 Funções do primeiro e segundo painel

```
1       public void PlusCount()
2       {
3           if (_count < 4) {
4               _count++;
5               UpdateCountText();
6               UpdateCubes();
7               _audioManager.PlayOneShot("Cube Plus");
8           }
9       }
10
11      public void MinusCount()
```

```
12      {
13          if (_count > 0) {
14              _count--;
15              UpdateCountText();
16              UpdateCubes();
17              _audioManager.PlayOneShot("Cube Minus");
18          }
19      }
20
21      private void UpdateCountText()
22      {
23          _textButton.text = "" + _count;
24      }
25
26      private void UpdateCubes()
27      {
28          for (int i = 0; i < 4; i++)
29          {
30              if (i < _count)
31              {
32                  _cubes[i].gameObject.SetActive(true);
33              }
34              else
35              {
36                  _cubes[i].gameObject.SetActive(false);
37              }
38          }
39      }
```

M.1.2 Funções do terceiro e quarto painel

```
1       public void RightButton()
2       {
3           _count = 6;
4           UpdateCountText();
5           UpdateCubes();
6           _button.image.color = Color.green;
7           if (_alreadyPressed == false)
8           {
```

```
9               _playerScript.LifeUp();
10              _alreadyPressed4 = true;
11              _audioManager.PlayOneShot("Right");
12          }
13      }
14
15      public void WrongButton()
16      {
17          _button.image.color = Color.red;
18          if (_alreadyPressed2 == false)
19          {
20              _playerScript.Damage();
21              _alreadyPressed = true;
22              _audioManager.PlayOneShot("Wrong");
23          }
24      }
25
26      private void UpdateCountText()
27      {
28          _textButton.text = "" + _count;
29      }
30
31      private void UpdateCubes()
32      {
33          for (int i = 0; i < _maxCubes; i++)
34          {
35              if (i < _count)
36              {
37                  _cubes[i].gameObject.SetActive(true);
38              }
39              else
40              {
41                  _cubes[i].gameObject.SetActive(false);
42              }
43          }
44      }
```

M.2 *Script* gerenciador da fase 4

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Level4Manager : MonoBehaviour
6  {
7      [SerializeField]
8      private GameObject _canvasNextLevel;
9      private PlayerScript _playerScript;
10     private CameraController _cameraController;
11     private AudioManager _audioManager;
12
13     private void Start()
14     {
15         _playerScript = GameObject.Find("Player").GetComponent<
               PlayerScript>();
16         _cameraController = GameObject.Find("Main Camera").
               GetComponent<CameraController>();
17         _audioManager = GameObject.Find("AudioManager").
               GetComponent<AudioManager>();
18     }
19
20     private void OnTriggerEnter(Collider other)
21     {
22         if (other.tag == "Player")
23         {
24             _playerScript.canMove = false;
25             _cameraController.active = false;
26             Cursor.visible = true;
27             Cursor.lockState = CursorLockMode.None;
28             _canvasNextLevel.SetActive(true);
29             _audioManager.PlayOneShot("Victory");
30         }
31     }
32 }
```

APÊNDICE N – *SCRIPTS* DA QUINTA FASE

N.1 *Script* objetos tridimensionais

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class KeyObject : MonoBehaviour
6  {
7      private ItemCanvasManager _itemCanvasManager;
8      private DropZone _dropZone;
9      private KeyManagers _keymanagers;
10     [SerializeField]
11     private GameObject _slotKey;
12     private AudioManager _audioManager;
13
14     void Start()
15     {
16         _itemCanvasManager = GameObject.Find("Item Canvas").
               GetComponent<ItemCanvasManager>();
17         _keymanagers = GameObject.Find("KeysManager").
               GetComponent<KeyManagers>();
18         _audioManager = GameObject.Find("AudioManager").
               GetComponent<AudioManager>();
19     }
20
21     private void OnTriggerEnter(Collider other)
22     {
23         if (other.tag=="Player")
24         {
25             if (this.gameObject.tag == "CubeKey")
26             {
27                 _slotKey.SetActive(true);
28                 _itemCanvasManager.ShowCubeItem();
```

```
29                  _keymanagers.cubeImage = true;
30              } else if (this.gameObject.tag == "Sphere Key")
31              {
32                  _slotKey.SetActive(true);
33                  _itemCanvasManager.ShowSphereItem();
34                  _keymanagers.sphereImage = true;
35              } else if (this.gameObject.tag == "PyramidKey")
36              {
37                  _slotKey.SetActive(true);
38                  _itemCanvasManager.ShowPyramidItem();
39                  _keymanagers.pyramidImage = true;
40              }
41              _audioManager.PlayOneShot("Sci-fi Pickup");
42              Destroy(this.gameObject);
43          }
44      }
45  }
```

N.2 *Script* correlação entre objetos

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  public class ProjectorScript : MonoBehaviour
7  {
8      [SerializeField]
9      private GameObject _whiteMouseCanvas;
10     [SerializeField]
11     private GameObject _projectorCanvas;
12     [SerializeField]
13     private GameObject _cubeImage;
14     [SerializeField]
15     private GameObject _sphereImage;
16     [SerializeField]
17     private GameObject _pyramidImage;
18     private KeyManagers _keyManagers;
19     public bool projectorDone = false;
```

```
20      private GreenLight _greenlight;
21      private GreenLight _greenlightSecond;
22      private GreenLight _greenLightThird;
23      private CameraController _cameraController;
24      private PlayerScript _playerScript;
25      private AudioManager _audioManager;
26      private bool _playSoundOnce = false;
27      [SerializeField]
28      private Image _arrowImage;
29
30      private void Start()
31      {
32          _keyManagers = GameObject.Find("KeysManager").
                GetComponent<KeyManagers>();
33          _greenlight = GameObject.Find("Light Wall").GetComponent
                <GreenLight>();
34          _greenlightSecond = GameObject.Find("Light Wall 2").
                GetComponent<GreenLight>();
35          _greenLightThird = GameObject.Find("Light Wall 3").
                GetComponent<GreenLight>();
36          _cameraController = GameObject.Find("Main Camera").
                GetComponent<CameraController>();
37          _playerScript = GameObject.Find("Player").GetComponent<
                PlayerScript>();
38          _audioManager = GameObject.Find("AudioManager").
                GetComponent<AudioManager>();
39      }
40
41      private void OnTriggerEnter(Collider other)
42      {
43          if (other.tag == "Player")
44          {
45              _whiteMouseCanvas.SetActive(true);
46          }
47      }
48
49      private void OnTriggerStay(Collider other)
50      {
```

```
51          if ((other.tag == "Player") && (Input.GetMouseButtonDown
                (0)))
52          {
53              _projectorCanvas.SetActive(true);
54              _whiteMouseCanvas.SetActive(false);
55              if (_keyManagers.cubeImage == true)
56              {
57                  _cubeImage.SetActive(true);
58                  if (_arrowImage != null)
59                  {
60                      _arrowImage.gameObject.SetActive(true);
61                  }
62              }
63              if (_keyManagers.sphereImage == true)
64              {
65                  _sphereImage.SetActive(true);
66              }
67              if (_keyManagers.pyramidImage == true)
68              {
69                  _pyramidImage.SetActive(true);
70              }
71              Cursor.visible = true;
72              Cursor.lockState = CursorLockMode.None;
73              _cameraController.active = false;
74              _playerScript.canMove = false;
75              if (_playSoundOnce == false)
76              {
77                  _audioManager.PlayOneShot("Sci-fi Projector");
78                  _playSoundOnce = true;
79              }
80
81          }
82          if ((other.tag == "Player") && (Input.GetKey(KeyCode.
                Escape)))
83          {
84              CloseCanvas();
85          }
86      }
87
```

```
88      private void OnTriggerExit(Collider other)
89      {
90          if (other.tag == "Player")
91          {
92              _whiteMouseCanvas.SetActive(false);
93              this.gameObject.transform.Find("Projector Canvas").
                    gameObject.SetActive(false);
94              Cursor.visible = false;
95              Cursor.lockState = CursorLockMode.Locked;
96              _cameraController.active = true;
97              _playerScript.canMove = true;
98              _playSoundOnce = false;
99          }
100     }
101
102     public void CloseCanvas()
103     {
104         this.gameObject.transform.Find("Projector Canvas").
                gameObject.SetActive(false);
105         _cameraController.active = true;
106         _playerScript.canMove = true;
107         _whiteMouseCanvas.SetActive(true);
108         Cursor.visible = false;
109         Cursor.lockState = CursorLockMode.Locked;
110         _playSoundOnce = false;
111     }
112
113     public void ProjectorComunicationWithDoorLight()
114     {
115         if (_keyManagers.keyCount == 1)
116         {
117             _greenlight.SetGreen();
118         }
119         if (_keyManagers.keyCount == 2)
120         {
121             _greenlightSecond.SetGreen();
122         }
123         if (_keyManagers.keyCount == 3)
124         {
```

```
125                 _greenLightThird.SetGreen();
126             }
127         }
128 }
```

N.3 *Script* luzes indicadoras

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class GreenLight : MonoBehaviour
6  {
7      private Material _material;
8      private Light _light;
9      [SerializeField]
10     private int _lightId;
11
12     void Start()
13     {
14         _material = GetComponent<Renderer>().materials[1];
15         _light = GetComponentInChildren<Light>();
16     }
17
18     public void SetGreen()
19     {
20         _material.SetColor("_EmissionColor", Color.green);
21         _light.color = Color.green;
22     }
23
24     public void SetRed()
25     {
26         _material.SetColor("_EmissionColor", Color.red);
27         _light.color = Color.red;
28     }
29 }
```

N.4 *Script* abrir e fechar portas

```
1  using UnityEngine;
2
3  public class DoorScript : MonoBehaviour
4  {
5      [SerializeField]
6      private GameObject _doorR;
7      [SerializeField]
8      private GameObject _doorL;
9      private float _speed = 1f;
10     [SerializeField]
11     private bool _open = false;
12     private Vector3 _initialPositionDoorR;
13     private Vector3 _initialPositionDoorL;
14     private Vector3 _endPositionDoorR;
15     private Vector3 _endPositionDoorL;
16     private ProjectorScript _projectorScript;
17     [SerializeField]
18     private int _doorId;
19     private OpenDoorManager _openDoorManager;
20     private AudioManager _audioManager;
21     private bool _playAudioOnceOpen = false;
22     private bool _playAudioOnceClose = false;
23
24     void Start()
25     {
26         _initialPositionDoorR = _doorR.transform.position;
27         _initialPositionDoorL = _doorL.transform.position;
28         if (_doorId == 3)
29         {
30             _endPositionDoorR.y = _initialPositionDoorR.y + 1.36
                   f;
31             _endPositionDoorL.y = _initialPositionDoorL.y - 2.0f
                   ;
32         }
33         else
34         {
```

```
35              _endPositionDoorR.x = _initialPositionDoorR.x - 1.36
                    f;
36              _endPositionDoorL.x = _initialPositionDoorL.x + 1.36
                    f;
37          }
38
39          _projectorScript = GameObject.Find("projector").
                GetComponent<ProjectorScript>();
40          _openDoorManager = GameObject.Find("Door Manager").
                GetComponent<OpenDoorManager>();
41          _audioManager = GameObject.Find("AudioManager").
                GetComponent<AudioManager>();
42      }
43
44      void Update()
45      {
46          if (_open == true)
47          {
48              if (_openDoorManager.door[_doorId-1] == 1)
49              {
50                  OpenRightDoor();
51                  if (_playAudioOnceOpen == false)
52                  {
53                      _audioManager.PlayOneShot("Door Open");
54                      _playAudioOnceOpen = true;
55                  }
56              }
57          }
58          else
59          {
60              CloseDoor();
61          }
62      }
63
64      private void OnTriggerEnter(Collider other)
65      {
66          if (other.tag == "Player")
67          {
68              _open = true;
```

```
69              _playAudioOnceClose = false;
70          }
71      }
72
73      private void OnTriggerExit(Collider other)
74      {
75          if (other.tag == "Player")
76          {
77              _open = false;
78              _playAudioOnceOpen = false;
79              if (_playAudioOnceClose == false && _openDoorManager
                    .door[_doorId - 1] == 1)
80              {
81                  _audioManager.PlayOneShot("Door Close");
82                  _playAudioOnceClose = true;
83              }
84          }
85      }
86
87      private void OpenRightDoor()
88      {
89          if ((_doorR.transform.position.x > _endPositionDoorR.x)
                && _doorId != 3)
90          {
91              _doorR.transform.Translate(Vector3.left * _speed * 2
                    * Time.deltaTime);
92              _doorL.transform.Translate(Vector3.right * _speed *
                    2 * Time.deltaTime);
93          }
94          else
95          {
96              if (_doorL.transform.position.y > _endPositionDoorL.
                    y)
97              {
98                  _doorR.transform.Translate(Vector3.up * _speed *
                        2 * Time.deltaTime);
99                  _doorL.transform.Translate(Vector3.down * _speed
                        * 2 * Time.deltaTime);
100             }
```

```
101         }
102     }
103
104     private void CloseDoor()
105     {
106         if ((_doorR.transform.position.x < _initialPositionDoorR
                .x) && _doorId != 3)
107         {
108             _doorR.transform.Translate(Vector3.right * _speed *
                    2 * Time.deltaTime);
109             _doorL.transform.Translate(Vector3.left * _speed * 2
                     * Time.deltaTime);
110         }
111         else
112         {
113             if (_doorL.transform.position.y <
                    _initialPositionDoorL.y)
114             {
115                 _doorR.transform.Translate(Vector3.down * _speed
                         * 2 * Time.deltaTime);
116                 _doorL.transform.Translate(Vector3.up * _speed *
                         2 * Time.deltaTime);
117             }
118         }
119     }
120 }
```

N.5 *Script* gerenciador da fase 5

```
1  using System.Collections;
2  using UnityEngine;
3  using UnityEngine.SceneManagement;
4  using UnityEngine.UI;
5
6  public class Level5Manager : MonoBehaviour
7  {
8      [SerializeField]
9      private GameObject _endGameCanvas;
10     private bool _endGame = false;
```

```
11      private Image _endingPanel;
12      private Text _endingText;
13      private Image _whiteMouseEndingImage;
14      private float _alpha = 0;
15
16      private void Start()
17      {
18          _endingPanel = GameObject.Find("Ending Panel").
                GetComponent<Image>();
19          _endingText = GameObject.Find("Ending Text").
                GetComponent<Text>();
20          _whiteMouseEndingImage = GameObject.Find("Ending White
                Mouse Image").GetComponent<Image>();
21      }
22
23      private void OnTriggerEnter(Collider other)
24      {
25          if (other.tag == "Player")
26          {
27              _endGameCanvas.SetActive(true);
28              _endGame = true;
29              StartCoroutine(WaitTime());
30          }
31      }
32
33      private void Update()
34      {
35          if (_endGame == true && Input.GetMouseButtonDown(0))
36          {
37              SceneManager.LoadScene("MainMenu");
38          }
39      }
40
41      private IEnumerator WaitTime()
42      {
43          for (int i = 0; i < 100; i++)
44          {
45              _alpha += 0.01f;
46              _endingPanel.color = new Color(0, 0, 0, _alpha);
```

```
47                _endingText.color = new Color(1, 1, 1, _alpha);
48                _whiteMouseEndingImage.color = new Color(1, 1, 1,
                      _alpha);
49                yield return new WaitForSeconds(0.04f);
50            }
51        }
52    }
```

APÊNDICE O – LISTA DE *ASSETS* DE TERCEIROS UTILIZADO

O.1 Objetos 3D

Personagem: *Character Pack: Free Sample.*

Por: SUPERCYAN

Disponível em: <https://assetstore.unity.com/packages/3d/characters/humanoids/character-pack-free-sample-79870>

Árvores: *Fox - Trees Pack*

Por: mehrasaur

Disponível em: <https://opengameart.org/content/fox-trees-pack>

Licença: CC0

Peras: *Green Pears*

Por: Artystarty

Disponível em: <https://www.turbosquid.com/3d-models/free-pears-normal-3d-model/729927>

Licença: *Royalty Free License*

Bananas: Banana

Por: dajver

Disponível em: <https://www.turbosquid.com/FullPreview/Index.cfm/ID/780590>

Licença: *Royalty Free License*

Diversos objetos da fase 5: Sci-Fi Styled Modular Pack

Por: KARBOOSX

Disponível em: <https://assetstore.unity.com/packages/3d/environments/sci-fi/sci-fi-styled-

O.2 Arte 2D

***Sprite* coração:** *Heart*

Por: cdgramos

Disponível em: <https://opengameart.org/content/heart-1>

Licença: CC0

Banana: Banana

Por: cdgramos

Disponível em: <https://opengameart.org/content/banana>

Licença: CC0

Pera: *Pear*

Por: cdgramos

Disponível em: <https://opengameart.org/content/pear>

Licença: CC0

Maçã: Fruits

Por: vandrake

Disponível em: <https://opengameart.org/content/fruits>

Licença: CC-BY 3.0

Seta indicadora: *glow arrow*

Por: oglsdl

Disponível em: <https://opengameart.org/content/glow-arrow>

Licença: CC-BY 4.0

Fundo espacial: Seamless Space / Stars

Por: n4

Disponível em: <https://opengameart.org/content/seamless-space-stars>

Licença: CC0

O.3 Som

O.3.1 Músicas

Menu principal *Electric Boy*

Por: FoxSynergy

Disponível em: <https://opengameart.org/content/electric-boy>

Licença: CC-BY 3.0

Tema 1º, 2º, 3º e 4º fase *Childrens Jungle Music*

Por: LauDic

Disponível em: <https://opengameart.org/content/childrens-jungle-music>

Licenças: CC-BY 3.0 e CC-BY-SA 4.0

Tema 5º fase *Lines of Code*

Por: Trevor Lentz

Disponível em: <https://opengameart.org/content/lines-of-code>

Licença: CC-BY-SA 3.0

Tema *Game Over* *Game Over Theme*

Por: CleytonKauffman

Disponível em: <https://opengameart.org/content/game-over-theme>

Licença: CC-BY-SA 3.0 e OGA-BY 3.0

O.3.2 Efeitos sonoros

Fim de fase *Victory!*

Por: jkfite01

Disponível em: <https://opengameart.org/content/victory-2>

Licença: CC-BY 3.0

Som ao passar sob um botão *Click*

Por: qubodup

Disponível em: <https://opengameart.org/content/click>

Licença: CC0

Som ao interagir com a placa *Page Turning Sfx Sound Effect*

Por: Nicole Marie T

Disponível em: <https://opengameart.org/content/page-turning-sfx-sound-effect>

Licença: CC-BY 4.0

Contador CONTAGEM REGRESSIVA EM PORTUGUÊS

Por: ANIMAÇÕES MATEMÁTICAS / PROFESSOR RICARDO

Disponível em: <https://www.youtube.com/watch?v=2CwMPIle_L8>

Som ao clicar em um botão *Point bell*

Por: Brandon75689

Disponível em: <https://opengameart.org/content/point-bell>

Licença: CC0 e OGA-BY 3.0

Ponte sendo construída 3º fase, Adicionar e remover blocos 4º fase *Wood and Metal Sound Effects: Volume 2*

Por: Ogrebane

Disponível em: <https://opengameart.org/content/wood-and-metal-sound-effects-volume-2>

Licença: CC0

Certo *UI Decline or Back*

Por: ViRiX

Disponível em: <https://opengameart.org/content/ui-decline-or-back>

Licença: CC-BY 3.0

Errado *Energy Drain*

Por: qubodup

Disponível em: <https://opengameart.org/content/energy-drain>

Licença: CC-BY-SA 3.0, GPL 3.0 e GPL 2.0

Interação com painel 4º fase e portas 5º fase *9 sci-fi computer sounds and beeps*

Por: Michel Baradari

Disponível em: <https://opengameart.org/content/9-sci-fi-computer-sounds-and-beeps>

Licença: CC-BY 3.0

Holograma final *Sci-Fi Drone Loop*

Por: jdagenet

Disponível em: <https://opengameart.org/content/sci-fi-drone-loop>

Licença: CC-BY 3.0