Curso de Sistemas de Informação
Universidade Estadual de Mato Grosso do Sul

DESENVOLVIMENTO DE JOGO 3D PARA AUXÍLIO NO ENSINO DE OPERAÇÕES MATEMÁTICAS BÁSICAS

Jean Rodrigo Santos

Profa. Dra. Mercedes Rocío Gonzales Márquez (Orientadora)

DOURADOS/MS
2021

DESENVOLVIMENTO DE JOGO 3D PARA AUXÍLIO NO ENSINO DE OPERAÇÕES MATEMÁTICAS BÁSICAS

Jean Rodrigo Santos

Este exemplar corresponde à redação final da monografia da disciplina Projeto Final de Curso devidamente corrigida e defendida por Jean Rodrigo Santos e aprovada pela Banca Examinadora, como parte dos requisitos para a obtenção do título de Bacharel em Sistemas de Informação.

Dourados, 26 de Novembro de 2021.

Profa. Dra. Mercedes Rocío Gonzales Márquez (Orientadora)

S235d Santos, Jean Rodrigo
Desenvolvimento de jogo 3d para auxílio no ensino de operações matemáticas básicas / Jean Rodrigo Santos. – Dourados, MS: UEMS, 2021.
41 p.

Monografia (Graduação) – Sistema de Informação – Universidade Estadual de Mato Grosso do Sul, 2021.
Orientadora: Profa. Dra. Mercedes Rocío Gonzales Márquez

1. Gamificação 2. Matemática 3. Educação I. Márquez, Mercedes Rocío Gonzales II. Título

CDD 23. ed. - 005.4

Curso de Sistemas de Informação
Universidade Estadual de Mato Grosso do Sul

DESENVOLVIMENTO DE JOGO 3D PARA AUXÍLIO NO ENSINO DE OPERAÇÕES MATEMÁTICAS BÁSICAS

Jean Rodrigo Santos

Novembro de 2021

Banca Examinadora:

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

Profa. Dra. Raquel Marcia Muller
Área de Computação - UEMS

Prof. Dr. Diogo Fernando Trevisan
Área de Computação - UEMS

RESUMO

Nas duas últimas décadas houve uma grande evolução no ramo da tecnologia e sua consequente inserção em outras áreas como a educação, a qual se beneficiou de forma importante com esta expansão tecnológica.

O presente trabalho apresenta a proposta de jogo educativo que visa o aprendizado de matemática com enfoque em alunos em séries iniciais. Apresenta-se também conceitos de Gamificação, jogos educacionais, ferramentas de modelagem e motores de jogo, além de reflexões sobre como a tecnologia descrita pode ser utilizada no ramo da educação, por meio de jogos educacionais.

Palavras-Chave: Jogos eletrônicos. Jogos educacionais. Gamificação. Matemática.

ABSTRACT

In the last two decades, there has been a great evolution in the field of technology, such evolution has made it easier to integrate with other areas. Education is one of the areas that benefits from this expansion, the benefit that technology offers an area of education is evident.

The following document presents how technology can be used in education, through electronic games, using gamification techniques and educational games, if an improvement in the quality of students' teaching in mathematics, with a focus on students in early grades. Soon after, an educational game proposal is presented, demonstrating in detail its potential and characteristics.

Keywords: Electronic games. Serious games. Gamification. Mathematics.

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).

LISTA DE *SCRIPTS*

1 INTRODUÇÃO

A Matemática é uma das matérias em que os alunos demonstram dificuldade no aprendizado, principalmente nos primeiros anos de sua vida escolar (STEENBRUGGE MARTIN VALCKE, 2010). Os resultados do Programa Internacional de Avaliação de Estudantes 2018, realizado no Brasil, demonstram claramente essa deficiência em Matemática, onde apenas 33% dos alunos conseguiram atingir o nível de proficiência 2 em Matemática, sendo considerado o nível básico do teste, ou seja mais de 60% dos alunos brasileiros se encontram em um nível abaixo do básico na matéria de Matemática (INEP/MEC, 2018).

Indivíduos que apresentam dificuldades de aprendizagem nem sempre possuem deficiências intelectuais. Conjectura-se, portanto, que a dificuldade no aprendizado de Matemática ocorra, principalmente, pelo fato dos alunos, não terem acesso a um material didático adequado, que instigue o aluno a pensar por si próprio ou talvez a métodos mecânicos de ensino, os quais não permitem a interação dos alunos com o conhecimento (CARNINE ASHA K. JITENDRA, 1997; GRAVEMEIJER, 1994).

O Referencial Curricular Nacional da Educação Infantil (RCNEI, 1998) afirma que o aprendizado da Matemática ajuda na formação dos seres independentes e com facilidade para se expressarem, sendo capazes de solucionar seus problemas e obstáculos. Incentiva a não ensinar Matemática por meio da memorização e repetição, pois os alunos não entendem a lógica e apenas decoram.

Trabalhar com o que é orientado pelo referencial através do lúdico é a forma mais interessante e dinâmica para o desenvolvimento do raciocínio lógico, memorização, capacidade de estratégia e, desta forma, formar alunos expressivos e capazes de argumentar.

O uso da gamificação e jogos educacionais criam experiências ricas através de ambientes lúdicos de aprendizado envolventes com uma série de elementos benéficos, como pensamento inovador, promoção de um senso de controle e permissão para o aluno falhar. A adição de elementos de jogos em ambientes de aprendizagem tradicionais é uma forma de ajudar a retenção do aprendizado de uma maneira impactante (KAPP, 2012). Logo, jogos digitais, que utilizem técnicas de Gamificação ou Jogos Educacionais, se mostram uma abordagem interessante ao auxílio do ensino de Matemática no Ensino Fundamental, pois possuem elementos necessários ao aprendizado de Matemática.

Neste trabalho de conclusão de curso foi proposto um jogo com cenários infantis atraentes como camelos, pinguins, patos, cestas, frutas dentro de ambientes da natureza para introduzir aos pequenos usuários do primeiro ano do ensino fundamental no estudo da matemática. Acreditamos que introduzir ao pequeno aluno no aprendizado da Matemática em forma bastante lúdica e interativa, traga uma maior confiança e familiaridade dos usuários, prevenindo o impacto negativo de conceitos frios que devam ser simplesmente

decorados. O jogo OperaMat consiste de quatro fases, cada uma das quais treina o aluno no aprendizado das operações básicas da Matemática como soma, subtração, multiplicação e divisão. Acredita-se que a proposta destes jogos inéditos se une à iniciativa da comunidade de desenvolvedores de jogos, de introduzir os jogos como uma ferramenta de auxílio interessante e atraente para o ensino da Matemática. Deixa-se em aberto a incorporação de outras funcionalidades que visem aprimorar estas ferramentas. O diferencial do jogo proposto neste trabalho, em relação às múltiplas opções já disponíveis, é que ele será construído em 3D. Gerar um modelo 3D é vantajoso, porque uma experiência mais realista torna-se possível ao usuário girar este modelo a vontade, assimilando seus detalhes.

1.1 OBJETIVO

1.1.1 OBJETIVO GERAL

O objetivo geral desse trabalho é o desenvolvimento de um Jogo Educacional com intuito de contribuir no ensino das operações matemáticas básicas, as quais são ministradas no primeiro ano do ensino fundamental. O jogo aborda o ensino e treinamento das operações de adição, subtração, divisão, multiplicação.

1.1.2 OBJETIVOS ESPECÍFICOS

- Desenvolvimento de mecânicas para o jogo.
- Criação de cenários.
- Pesquisa e criação de modelos 3d para uso nos cenários do jogo.

1.1.3 METODOLOGIA

O trabalho foi desenvolvido em três etapas correspondendo diretamente aos objetivos específicos citados.

A primeira etapa consistiu na criação das mecânicas envolvidas no jogo. Nesta etapa foram analisados e construídos algoritmos para o desenvolvimento da lógica de cada fase do jogo utilizando a linguagem C-sharp.

Na segunda e terceira etapa foram desenvolvidos os modelos e cenários do jogo, contendo toda a lógica construída na primeira etapa. Relacionam-se os scripts com seus respectivos objetos, contendo colisões, movimentações, etc.

1.2 ORGANIZAÇÃO DO TEXTO

O documento está dividido em 4 capítulos, no Capítulo 1 é feita uma breve contextualização e apresentado o problema de pesquisa, assim como os objetivos gerais e objetivos específicos.

No Capítulo 2 são apresentados alguns conceitos importantes e ferramentas usadas no desenvolvimento do jogo. No Capítulo 3 são apresentados alguns trabalhos correlatos, dentre os quais alguns foram usados como inspiração da presente proposta. No Capítulo 4 é proposto o *GDD*, que contém a idealização do jogo.

No Capítulo 5 apresentam-se a descrição detalhada das mecânicas envolvidas no jogo. Mais especificamente, os algoritmos construídos para o desenvolvimento da lógica de cada fase do jogo utilizando a linguagem C#.

Finalmente no Capítulo 6 fala-se da implementação do jogo.

2 MOTOR DE JOGO UNITY E MODELOS USADOS

Neste capítulo é descrito com certo detalhe o motor de jogo Unity, o qual foi usado na implementação do jogo proposto. Também são descritos os modelos disponíveis que foram usados. O intuito deste capítulo é apresentar ao leitor uma introdução à ferramenta Unity e também um conhecimento preliminar das opções de modelos disponíveis para inserção em cenários de jogos.

2.1 UNITY

A ferramenta Unity é um dos motores de jogos mais populares atualmente. Sua primeira versão foi lançada em 2005, como um motor de jogo proprietário, desenvolvido e mantido pela Unity. Possui duas versões, uma gratuita e outra paga. É possível comercializar os jogos utilizando a versão gratuita, desde que o faturamento não ultrapasse cem mil reais por ano. Desse modo, muitos desenvolvedores e estudantes podem realizar a confecção de jogos gratuitamente (TEOTÔNIO; ARAÚJO, 2017). Unity torna a produção de um jogo simples fornecendo uma série de facilidades que permitem construir um cenário de jogo. É utilizado o conceito de *GameObject*, que permite dividir as partes do jogo em vários componentes fáceis de serem gerenciados(GOLDSTONE, 2009).

2.1.1 SCRIPTS

Scripts são considerados uma parte essencial da produção de um jogo. São eles que determinam como os eventos irão ocorrer, ou como será a mecânica principal de um jogo. No Unity a linguagem principal utilizado na criação dos *scripts* ó o C# (GOLDSTONE, 2009).

O script é responsável pelo controle do comportamento dos *GameObject*, sendo caracterizado como uma parte funcional do mesmo. O Unity possuí vários componentes integrados. Embora esses componentes integrados possam ser muito versáteis, é necessário ir além do que eles podem fornecer para implementar recursos específicos de um jogo. Para isso o Unity permite a criação de componentes customizados com o uso de scripts. Permitindo a adição de eventos e modificação das propriedades de componentes, dentre outros (UNITY,).

O Unity oferece suporte nativo à linguagem de programação C#. Além disso, muitas outras linguagens .NET podem ser usadas com o Unity se puderem compilar uma DLL compatível.

2.1.2 GAME OBJECTS

GameObject é o conceito mais importante no Editor do Unity. Cada objeto presente no jogo é um *GameObject*, de personagens e itens colecionáveis a luzes, câmeras e efeitos especiais. No entanto, um *GameObject* não pode fazer nada sozinho, é necessário atribuir propriedades a ele antes que se torne um personagem, um ambiente ou um efeito especial.

2.1.3 C-SHARP

C# é uma Linguagem de Programação simples, moderna, orientada a objetos. *C-sharp* tem sua raiz na família de linguagens C. C# é padronizado por ECMA como ECMA-334 e pela ISO/IEC como ISO/IEC 23270 (HEJLSBERG et al., 2008).

C# Possuí um sistema de tipo unificado. Todos tipos em C#, incluindo tipos primitivos como *int* e *double*, são herdados de um único tipo de objeto. Assim, todos os tipos compartilham um conjunto de operações em comum (HEJLSBERG et al., 2008).

O *script* a seguir foi escrito em C# e demonstra a estrutura de um procedimento de movimentação de objeto (HEJLSBERG et al., 2008).

Listing 2.1 – Procedimento de movimentação de um objeto

```
1  public class NewBehaviourScript: MonoBehaviour
2  {
3      public float movementSpeed = 10;
4
5      void Update(){
6
7          transform.Translate(Vector3.right * movementSpeed * Time
               .deltaTime);
8
9      }
10 }
```

O Unity C# *Job System* permite aos usuários escrever código *multithread* que interage bem com o resto do Unity e torna mais fácil escrever o código correto.

Escrever código multithread pode fornecer benefícios de alto desempenho. Isso inclui ganhos significativos na taxa de quadros. Usar o compilador *Burst* com tarefas C# oferece qualidade de geração de código aprimorada, o que também resulta em redução substancial do consumo de bateria em dispositivos móveis.

2.1.4 UNITY UI

Unity UI é um kit de ferramentas para o desenvolvimento de interfaces de usuário para jogos e aplicativos. É um sistema de interface baseado em *GameObject* que usa componentes e

o *gameview* para organizar, posicionar e estilizar interfaces de usuário.(UNITY,).

O seu uso ocorre em todas as fases presentes no jogo. Sendo utilizado na construção de botões e textos informativos, Figura 1. O Unity possibilita a edição da interface na própria construção da cena, para ter acesso a tela de edição da interface, basta ativar o modo 2D.

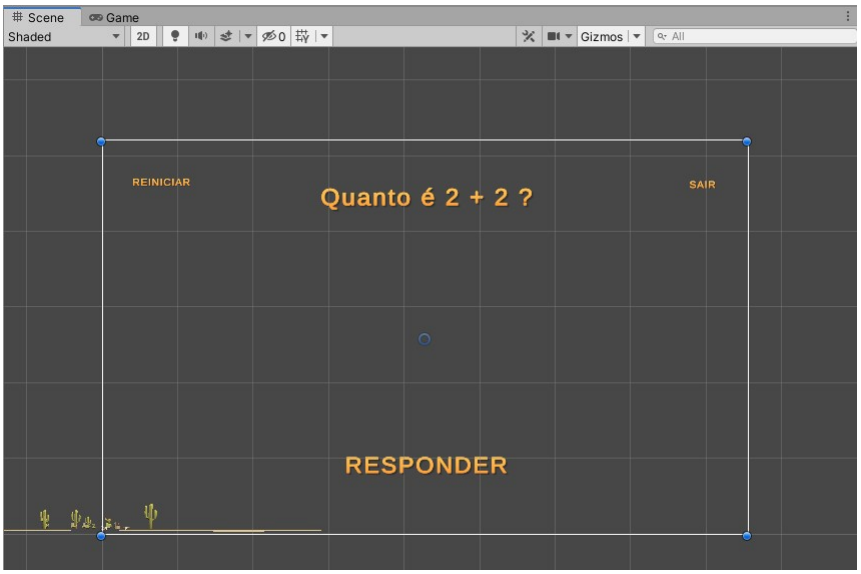


Figura 1 – Unity - Canvas

2.1.5 ASSET STORE

A Unity *Asset Store* contém uma biblioteca de *assets* gratuitos e comerciais que a Unity e também os membros da comunidade criam. Uma ampla variedade de recursos está disponível, incluindo texturas, modelos, animações, exemplos de projetos inteiros, tutoriais e extensões do editor (UNITY,).

Os pacotes da *Asset Store* contêm informações sobre como encontrar, adquirir e organizar seus pacotes diretamente no site do Unity (UNITY,).

Durante o desenvolvimento do jogo em questão, foram utilizados diversos modelos disponíveis pela na *Asset Store*. Tais como terrenos, luzes, árvores, entre outros.

2.1.6 *RIGIDBODY*

O *Rigidbody* é um componente responsável pelo controle da posição de um objeto por meio de simulação de física.

Adicionar um componente *Rigidbody* a um objeto colocará seu movimento sob o controle do motor de física do Unity. Mesmo sem adicionar nenhum código, um objeto *Rigidbody* será puxado para baixo pela gravidade e reagirá a colisões com objetos de entrada se o componente Collider correto também estiver presente (UNITY,).

O Rigidbody possui uma API de script que permite aplicar forças ao objeto e controlá-lo de forma fisicamente realista. Por exemplo, o comportamento de um carro pode ser especificado em termos das forças aplicadas pelas rodas. Com essas informações, o motor físico pode lidar com a maioria dos outros aspectos do movimento do carro, então ele irá acelerar de forma realista e responder corretamente a colisões (UNITY,).

Este componente é de suma importância para que a mecânica de arrastar e soltar objetos se torne possível. E com a combinação dele e o componente *Box collider* que a mecânica de arrastar e soltar objetos se tornou possível.

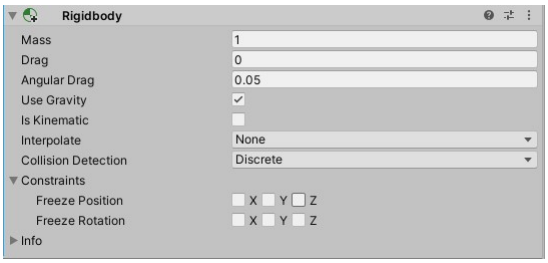


Figura 2 – Tela de abertura

2.1.7 *COLLIDERS*

Colliders definem a forma de um *GameObject* para efeitos de colisões físicas. Um colisor, que é invisível, não precisa ter exatamente a mesma forma que a malha do *GameObject*. Uma aproximação da malha é frequentemente mais eficiente e indistinguível no jogo.

2.1.8 *BOX COLLIDER*

Box Colliders, Figura 3 , são cuboides retangulares e são úteis para itens como caixas ou baús. No entanto, é possível usa-lo em objetos finos, tais como paredes ou rampas (UNITY,).

Esse tipo de colisor geralmente é utilizado um objetos que não necessitam de um envolvimento muito específico, como por exemplo, detectar a colisão em um riacho.

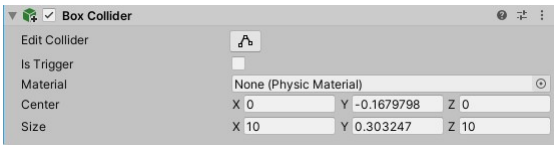


Figura 3 – *Box collider*

2.1.9 *MESH COLLIDER*

O *Mesh collider*, Figura 4 , constrói seu volume colisor com base em uma malha específica. É mais preciso para detecção de colisão do que usar primitivas para malhas complexas.

Mesh colliders marcados como convexos podem colidir com outros *Mesh colliders* (UNITY,).

Esse tipo de colisor é usado em objetos que necessitam de um envolvimento de colisão mais específico, tais como baldes, cestas, entre outros.

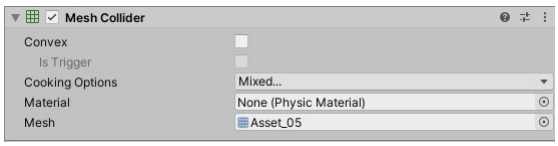


Figura 4 – *Mesh colider*

2.1.9.1 *IS TRIGGER*

O *is trigger* é uma propriedade do componente de colisão, que atua como um gatilho para detectar a colisão em um determinado volume de colisão. Ele envia as mensagens *OnTriggerEnter*, *OnTriggerExit* e *OnTriggerStay* quando um corpo rígido entra ou sai do volume de colisão.

Essa funcionalidade é fundamental para o reconhecimento de objetos em um determinado volume de colisão. Ao ativa-la, Figura 5, qualquer objeto, ao invés de colidir com o volume colisão, irá atravessa-lo, e desta maneira será possível reconhecer os eventos de entrada e saída do mesmo. Abaixo temos um trecho de *script* que reconhece a entrada de objetos em um determinado volume de colisão.

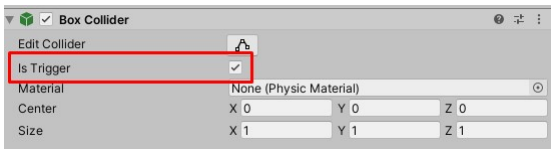


Figura 5 – Propriedade *is trigger*

Listing 2.2 – Contador de objetos

```
1 void OnTriggerEnter(Collider other) {
2     GameManager gameManagerScript = gameManager.GetComponent<
          GameManager>();
3
4     var sceneName = SceneManager.GetActiveScene().name;
5
6     var sceneNameToChar = sceneName.ToCharArray();
7
8     var numeroDaFase = sceneNameToChar[5];
9     var numeroDaFasaEmValorNumerico = Char.GetNumericValue(
          numeroDaFase);
```

```
10      var numeroDaEtapa = (Char.GetNumericValue(sceneNameToChar[
            sceneName.Length - 1]));
11
12      var resposta = 0;
13
14      if (numeroDaFasaEmValorNumerico == 1) {
15          if (numeroDaEtapa == 1) {
16              resposta = gameManagerScript.respostaEtapa1Fase1;
17          }
18          if (numeroDaEtapa == 2) {
19              resposta = gameManagerScript.respostaEtapa2Fase1;
20          }
21          if (numeroDaEtapa == 3) {
22              resposta = gameManagerScript.respostaEtapa3Fase1;
23          }
24      }
25
26      if (numeroDaFasaEmValorNumerico == 2) {
27          if (numeroDaEtapa == 1) {
28              resposta = gameManagerScript.respostaEtapa1Fase2;
29          }
30          if (numeroDaEtapa == 2) {
31              resposta = gameManagerScript.respostaEtapa2Fase2;
32          }
33          if (numeroDaEtapa == 3) {
34              resposta = gameManagerScript.respostaEtapa3Fase2;
35          }
36      }
37
38      if (numeroDaFasaEmValorNumerico == 3) {
39          if (numeroDaEtapa == 1) {
40              resposta = gameManagerScript.respostaEtapa1Fase3;
41          }
42          if (numeroDaEtapa == 2) {
43              resposta = gameManagerScript.respostaEtapa2Fase3;
44          }
45          if (numeroDaEtapa == 3) {
46              resposta = gameManagerScript.respostaEtapa3Fase3;
47          }
```

```
48      }
49
50      if (numeroDaFasaEmValorNumerico == 4) {
51          if (numeroDaEtapa == 1) {
52              resposta = gameManagerScript.respostaEtapa1Fase4;
53          }
54          if (numeroDaEtapa == 2) {
55              resposta = gameManagerScript.respostaEtapa2Fase4;
56          }
57          if (numeroDaEtapa == 3) {
58              resposta = gameManagerScript.respostaEtapa3Fase4;
59          }
60      }
61
62      if(other.CompareTag("Object")) {
63          qtdObjectsInBox += 1;
64      }
65  }
```

2.2 MODELAGEM

2.2.1 *TURBOSQUID*

Os modelos *TurboSquid* são usados por desenvolvedores de jogos, agências de notícias, arquitetos, estúdios de efeitos visuais, anunciantes e profissionais criativos em todo o mundo.

2.2.2 *CGTRADER*

CGTrader é a maior fonte mundial de ações licenciáveis e modelos 3D personalizados. O *CGTrader Marketplace* de autoatendimento apresenta 1.330.000 modelos 3D e inclui uma comunidade gerenciada de 4,68 milhões de usuários.

3 JOGOS CORRELATOS

Neste capítulo aborda-se a jogabilidade e mecânicas de uma série de jogos que foram utilizados como inspiração para o desenvolvimento do jogo proposto neste trabalho.

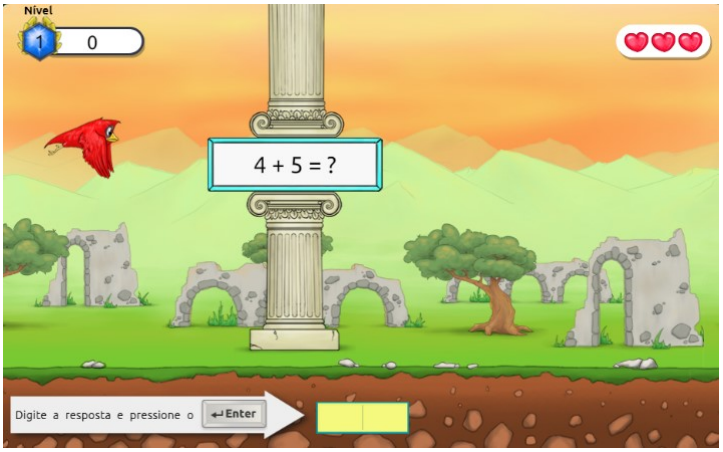
As mecânicas destes jogos envolvem uma importante interação do jogador com o mundo do jogo, estimulando o jogador a pensar de maneria crítica, prestar atenção a detalhes e a trabalhar sua criatividade.

3.1 PLATAFORMA *MATIFIC*

O plataforma *Matific* possui uma variedade de jogos educativos de alta qualidade. Os jogos presentes nessa plataforma buscam ensinar os alunos com foco em resolução de problemas, aprender fazendo, por curiosidade e descoberta. O *Matific* desafia os alunos a formular, modelar e examinar inúmeras abordagens para a resolução de problemas, resultando na aplicação de estratégias para buscar soluções e verificar respostas (MATIFIC, 2020). As fases a serem desenvolvidas na proposta do jogo, tomam como base alguns jogos presentes na plataforma *Matific*.

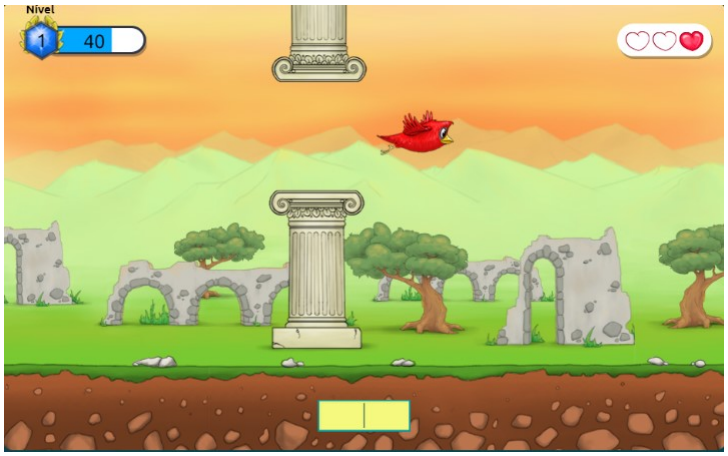
3.1.1 SOMANDO ATÉ 10

Estão presentes no cenário diversos obstáculos que devem serem removidos pelo jogador, Figura 7. É apresentado uma pergunta ao aluno e um campo de resposta, Figura 6, caso o aluno responda corretamente a pergunta um obstáculo é removido e ele está livre para para prosseguir para o próximo obstáculo, caso erre perderá uma de suas 3 vidas. A fase contínua até que o jogador atinja a pontuação desejada. O aspecto principal presente neste jogo é a sua dinâmica de perguntas, no qual o usuário deve acertar a questão apresentada para continuar seu progresso na fase.



Retirado de: <https://www.matific.com/>

Figura 6 – Inicio da fase com obstáculo fechado



Retirado de: <https://www.matific.com/>

Figura 7 – Obstáculo liberado

3.1.2 SUBTRAINDO DEZENA DE DEZENAS

Nesse módulo do *Matific* o jogador interage com uma bandeja, na qual insere a quantidade de *tickets*, que ficam em uma caixa ao lado. Ou seja, ao iniciar a fase é pedido para o jogador arrastar uma certa quantidade de *tickets* para dentro da bandeja, Figura 8. Após o jogador inserir os *tickets* é necessário clicar no botão de "pronto"para iniciar a operação de subtração Figura 9.

Retirado de: <https://www.matific.com/>

Figura 8 – Bandeja sem *tickets*

Retirado de: <https://www.matific.com/>

Figura 9 – Bandeja após o jogador inserir os *tickets*

Logo em seguida o jogador é desafiado a retirar uma certa quantidade de *tickets*, então os *tickets* remanescentes irão corresponder ao resultado da operação de subtração, que deverá ser inserido em um formulário de resposta Figura 10.

Retirado de: <https://www.matific.com/>

Figura 10 – Bandeja após o jogador remover os *tickets*

Caso o jogador acerte a resposta, o resultado é apresentado na tela, Figura 11, permitindo-o seguir para a próxima fase.

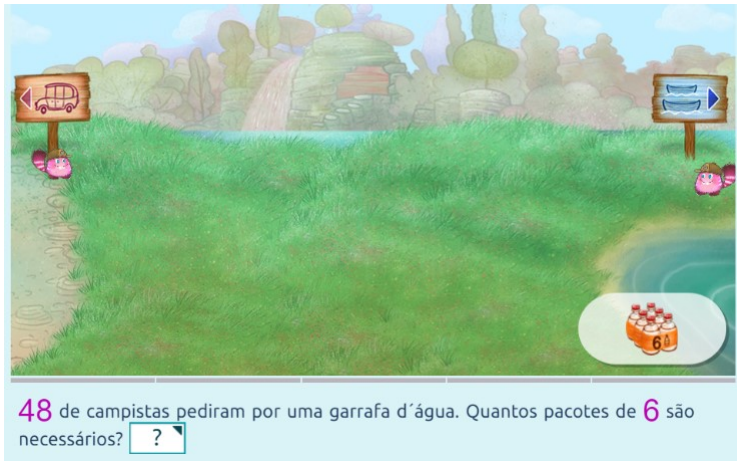
A operação de arrastar objetos até um recipiente de validação foi utilizada na idealização da fase 2, na qual será utiliza uma cesta que receberá uma quantidade variável de frutas.

Retirado de: <https://www.matific.com/>

Figura 11 – Tela final

3.1.3 DIVIDINDO ATÉ 100

Nesse módulo *Matific*, o jogador deve interagir com garrafas e cadeiras, posicionando a quantidade de objetos presentes na pergunta apresentada na parte inferior da tela, Figura 12. Após posicionar os objetos é necessário responder a questão corretamente no formulário através do mouse, Figura 15.



Retirado de: <https://www.matific.com/>

Figura 12 – Tela inicial da fase

Retirado de: <https://www.matific.com/>

Figura 13 – Posicionamento das garrafas

Retirado de: <https://www.matific.com/>

Figura 14 – Pergunta relacionada a quantidade de mesas

Retirado de: <https://www.matific.com/>

Figura 15 – Posicionamento das mesas

Após respondida corretamente as perguntas, o jogador pode prosseguir para a próxima etapa da fase.

4 GDD (*GAME DESIGN DOCUMENT*)

Este capítulo apresenta o *GDD* criado para o desenvolvimento do jogo desse trabalho, o *GDD* foi baseado em diversos modelos encontrados sendo o principal o GDD desenvolvido por alunos da Unicamp no jogo As Aventuras de Jackie e Tony (LATANSIO CARLA FLORENTINO, 2011).

Antes de entendermos qual é o objetivo de um *GDD*, é necessário entender o conceito de *Game Design*, pois seu desenvolvimento está diretamente relacionado a ele. *Game Desgin* detemina qual escolhas os jogadores serão capazes de fazer no mundo do jogo, e qual será o impacto dessas escolhas no progresso do jogador. *Game Design* também determina qual será o critério de vitória ou derrota, como o jogador irá interagir com o jogo, quais informações o jogo irá transmitir para o jogador, como o jogador irá controlar o jogo e qual será o nível de dificuldade do jogo(ROUSE, 2004).

O processo de *Game Desgin*, é uma fase onde as ideias não necessariamente precisam estar documentadas, podendo ser um processo que envolva apenas o pensamento. No entanto, na maioria dos casos é interessante que haja a documentação das ideias, tendo em vista a limitação da memória e o fácil esquecimento de detalhes importantes (SCHELL, 2008). Detalhar o design de um jogo também facilita a apresentação das ideias quando for necessário compartilha-las com terceiros. É nesse cenário que o *GDD* entra em ação, tendo como foco documentar as ideias construídas através de um processo de *Game Design*.

Game Design Document GDD são ferramentas comuns de documentação utilizadas por projetistas de jogos com o intuito de gerenciar o processo de desenvolvimento. Tipicamente, *GDD* descrevem, na maioria das vezes, a história de um jogo, suas mecânicas, direção de arte, projeto de nível, música e sons, obstáculos e recompensas. A necessidade de gerenciar o processo de desenvolvimento enquanto respeitando a criatividade e capacidade técnica da equipe de desenvolvimento pode ser desafiadora. Logo o *GDD* precisa estar de acordo com a maturidade técnica da equipe de desenvolvimento (SANSONE, 2014).

Escrever um *GDD* é considerado um prática comum na industria de jogos, no entanto nenhum padrão foi estabelecido em como, quando o para qual propósito um *GDD* é escrito. No entanto alguns semelhanças são encontradas em todo desenvolvimento de *GDD*, que geralmente deve determinar quais elementos irão criar um jogo interessante para o jogador e como esses elementos serão entregados a ele. Tipicamente o *GDD* deve conter a descrição da mecânica principal do jogo, o detalhamento dos níveis e suas artes, personagens e suas atribuições, entre outros.

O *Game Design Document GDD* apresentado nesse documento visa detalhar a proposta de jogo, demonstrando em detalhes as características que estarão presentes no jogo. O

jogo é descrito detalhadamente, incluindo os seguintes elementos:

- Fases e suas mecânicas
- Artes das fases
- Sons ambiente e músicas
- Desafios

4.1 VISÃO GERAL

4.1.1 TEMA

O jogo será educativo, com o intuito de ajudar no ensino de matemática básica. Irá possuir elementos interessantes que despertem o interesse do aluno.

4.1.2 MECÂNICAS BÁSICAS DE JOGABILIDADE

Segundo (CAZELLI, 2019) o uso de muitos comandos pode ser um problema para jogadores muito novos, pois dificulta a sua rápida adaptação ao jogo. Sendo assim a proposta em questão irá utilizar o *mouse* e seus botões como interação principal do jogador com o jogo, visando um inicio rápido e fácil.

- Uso do mouse para interagir com o todas as funcionalidades do jogo
- Dinâmica de perguntas para prosseguir para fases posteriores

4.1.3 PLATAFORMAS

- Windows.
- Linux.

4.1.4 MODELO DE MONETIZAÇÃO

O jogo será gratuito.

4.1.5 PÚBLICO ALVO

O jogo visa ensinar matemática para os alunos que estejam cursando entre o primeiro e segundo ano do ensino fundamental. O *design* das fases foram baseadas em ementas disponibilizadas pela instituição de ensino Abeu colégios (ABEU, 2020a; ABEU, 2020b).

4.1.6 GÊNERO E INSPIRAÇÕES

O gênero do jogo é educativo, especificamente utilizando técnicas de Jogos Educativos. O jogo será inspirado na plataforma Matific.

4.1.7 DESCRIÇÃO DO PROJETO

As tecnologias usadas serão:

- Blender.
- Unity.

4.1.8 DIFERENCIAL DO PROJETO

O projeto em questão visa criar um jogo educativo gratuito, tendo em vista que a maioria dos jogos educativos disponíveis atualmente necessitam de pagamento para o acesso.

4.1.9 *GAMEPLAY*

O jogador pode interagir com o cenário através do mouse. O jogador poderá interagir com alguns objetos, que serão utilizados como meio de resposta das perguntadas propostas.

A missão do jogador será resolver etapas para passar de fase. Cada fase terá em média 4 etapas a serem cumpridas.

4.2 CONTROLE

O jogo irá utilizar o *mouse* para receber a entrada do usuário.

4.2.1 *MOUSE*

O *mouse* é utilizado para movimentar a câmera, interagir com os objetos espalhados pelo cenário e responder as questões apresentadas.

- **Clique botão esquerdo**: Interage com certos objetos do jogo, movimenta o jogador e responde as questões.

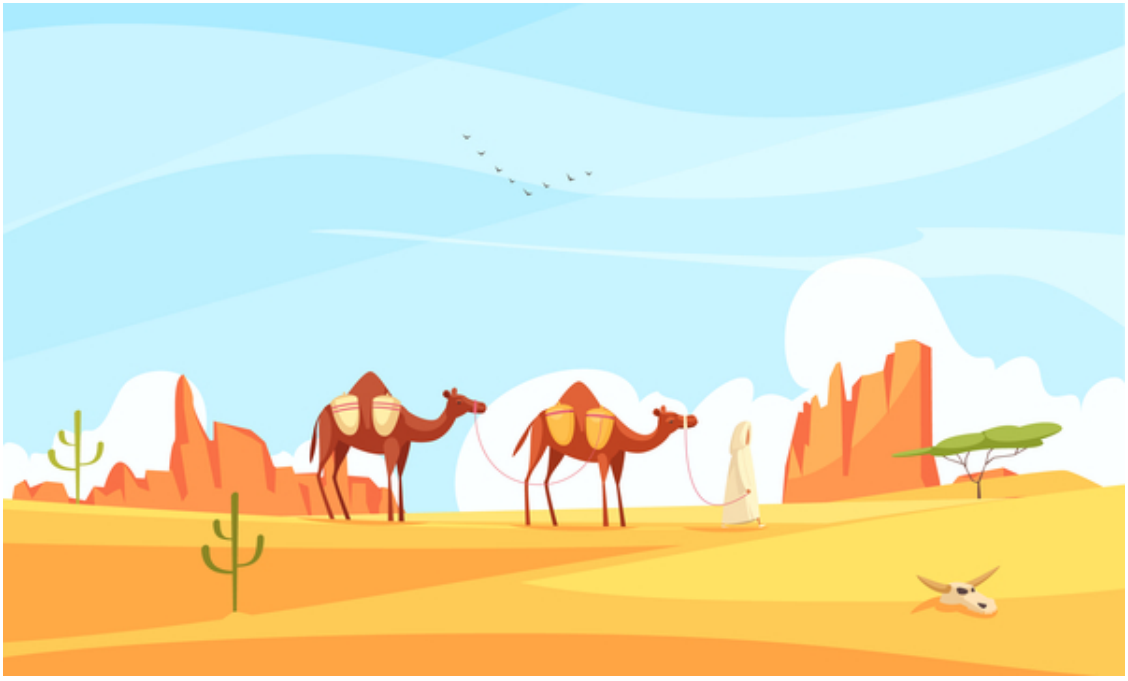
4.3 FASES E SEUS CONCEITOS

4.3.1 FASE 1

A fase 1 irá abordar o conceito de soma de números de 0 a 9. Será feito ao jogador perguntas com dificuldade progressiva, cada pergunta representa uma etapa da fase, caso

o jogador acerte a pergunta ele passa para a próxima etapa, caso erre, perde uma de suas três vidas e avança para a próxima etapa. Se suas vidas acabarem a fase é reiniciada.

O cenário será representado por um deserto, Figura 16, no ambiente estarão presentes objetos correlatos, que deverão ser utilizados como recurso para responder as questões apresentadas.



Retirado de: <https://www.freepik.com/>

Figura 16 – Cenário da fase 1

Retirado de: <https://www.freepik.com/>

Figura 17 – Oasis

O objetivo central da fase é conduzir dois camelos até um oásis, para que eles possam beber a água, Figura 17. Cada etapa respondida corretamente diminuirá a distância entre os camelos e o oásis. Os camelos serão utilizados como recipientes dos objetos presentes no cenário, Figura 18. Por exemplo, caso a pergunta direcionada ao jogador seja "Quanto é 5 + 4, coloque a quantidade de pedras e cactos correspondentes em cada camelo", o jogador deverá colocar 5 pedras no camelo da esquerda e 4 cactos no camelo da direita, a quantidade total será 9 e os camelos terão a distância diminuída até o oásis.

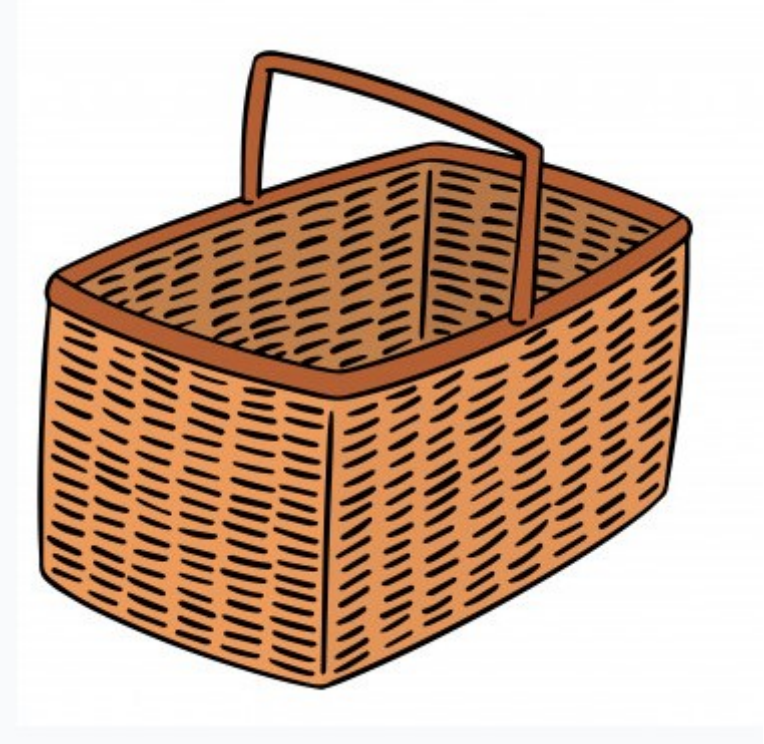
Retirado de: <https://www.freepik.com/>

Figura 18 – Objetos da fase 1

4.3.2 FASE 2

A fase 2 irá abordar o conceito de subtração de números de 0 a 9. O aspecto central presente nessa fase é uma cesta de frutas, Figura 19, que será responsável por validar as respostas informadas pelo jogador.

Estilo de cesta, onde irá ocorrer os eventos da fase:



Retirado de: <https://www.freepik.com/>

Figura 19 – Cesta

O cenário será representado por um ambiente de piquenique:

O objetivo principal dessa fase e ensinar o aluno o conceito de subtração de maneira interativa. De modo que ele, primeiramente irá preencher uma cesta, Figura 19, com uma determinada quantia de frutas, Figura 21, por exemplo 9, e logo após terá que remover uma outra quantia informada na tela, por exemplo 5, o foco dessa parte é demonstrar o conceito de diminuição entre dois valores. Após ter realizado a subtração o jogador deve informar o resultado, que será: 9 - 5 = 5.

Retirado de: <https://www.freepik.com/>

Figura 20 – Cenário da fase 2

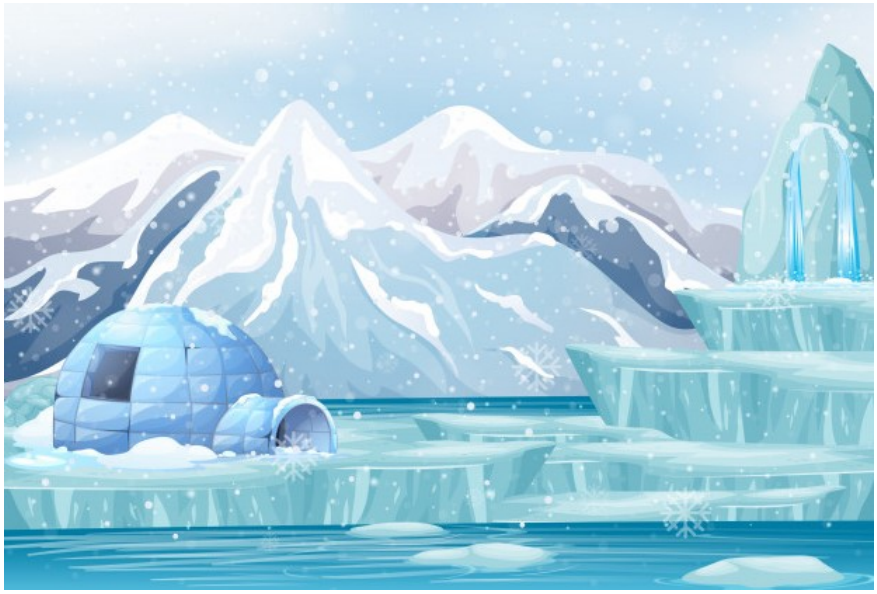
Retirado de: <https://www.freepik.com/>

Figura 21 – Objetos da fase 2

4.3.3 FASE 3

A fase 3 irá abordar o conceito de multiplicação dos números 2 ao 5. O cenário é representado por um clima gelado, Figura 22, onde irá haver uma construção iglu e diversos grupos de pinguins, Figura 23, ao lado da iglu.

Exemplo de cenário a ser utilizado:



Retirado de: <https://www.freepik.com/>

Figura 22 – Cenário da fase 3

O objetivo da fase é popular a iglu com a quantidade de grupos de pinguins correspondentes ao número presente na questão que ficará na parte inferior da tela. Por exemplo, se o número passado na pergunta for 10 e o múltiplo atual for 5, então o jogador deverá inserir dois grupos de pinguins na iglu, caso realizado corretamente, o jogador avança para a próxima etapa.

Retirado de: <https://www.freepik.com/>

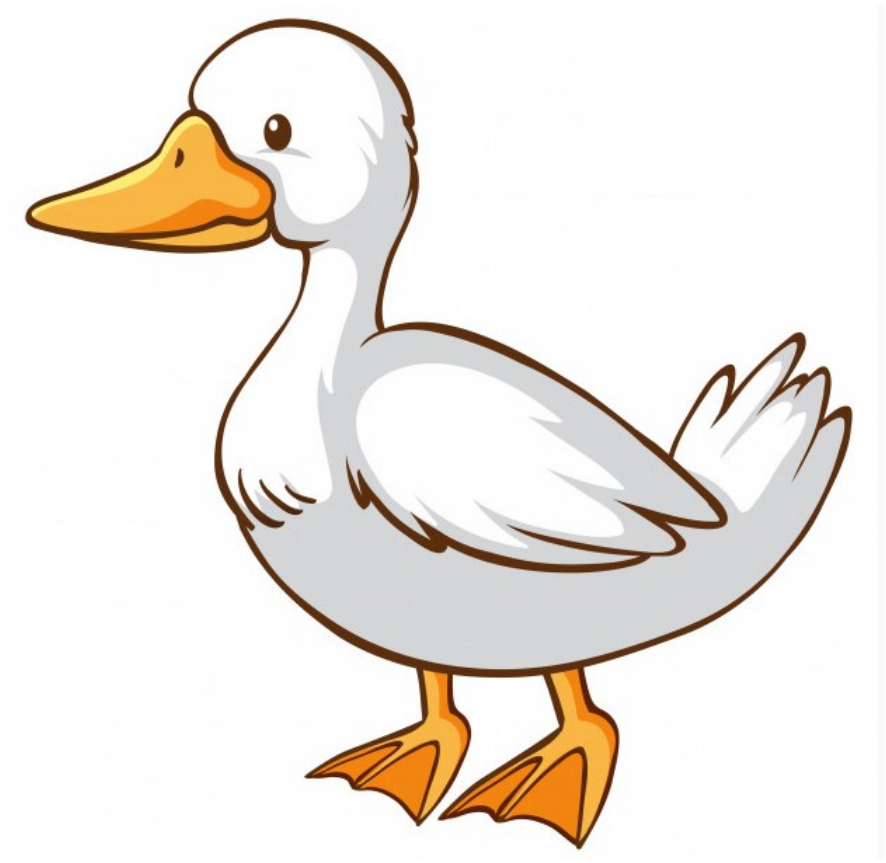
Figura 23 – Objetos da fase 3

4.3.4 FASE 4

A fase 4 irá abordar o conceito de divisão, envolvendo números de 2 a 5. O cenário irá ser composto por um rio envolto de uma gramado, Figura 24. O objetivo da fase é responder as questões apresentadas de maneira interativa. Estará presente no cenário uma quantidade variável de patos, Figura 25, deve ser colocado no rio a quantidade de patos que corresponde a resposta do problema de divisão. Por exemplo, caso o número seja 10 e o divisor 5, deve ser colocado 2 patos no rio.

Retirado de: <https://www.freepik.com/>

Figura 24 – Cenário da fase 4



Retirado de: <https://www.freepik.com/>

Figura 25 – Objetos da fase 4

4.4 SOM

Esse capitulo aborda os aspectos sonoros do game.

4.4.1 MÚSICAS

A musica será instrumental e diferente para cada fase.

4.4.2 ARTE VISUAL

4.4.2.1 HUD *HEAD-UP-DISPLAY*

O HUD é a interface de usuário, responsável por emitir informações de jogo, o jogo deverá conter os seguintes elementos em seu HUD:

- Botão de reiniciar a fase atual
- Botão de sair da fase atual
- Perguntas relacionadas as atividades desempenhadas durante a fase

4.4.2.2 TELAS DE MENU

Fluxo de telas:

O jogo irá possuir telas de menu:

- Menu inicial: irá conter as informações essenciais do jogo, por exemplo: iniciar o jogo e configurações do jogo
- Fases: Essa tela irá conter as fases presentes no jogo

5 MECÂNICAS

5.1 COLISÃO

A detecção de colisão possuí várias soluções. A solução mais simples apresentada para o problema, é geralmente declarado da seguinte forma: Dado um conjunto de objetos e uma descrição de seus movimentos ao longo de um certo período de tempo, determine se algum par de objetos entrará em contato. Mais versões complexas exigem, como por exemplo, "encontrar o tempo e os recursos envolvidos na colisão (JIMÉNEZ; THOMAS; TORRAS, 2001). (JIMÉNEZ; THOMAS; TORRAS, 2001) Apresenta quatro diferentes soluções para o problema de detecção de colisões, sendo elas: *Spatio-temporal intersection*, *Swept volume interference*, *Multiple interference detection* e *Trajectory parameterization*. Neste documento foi decidio explorar a primeira: *Spatio-temporal intersection*. (JIMÉNEZ; THOMAS; TORRAS, 2001) define essa solução da seguinte maneira: A representação mais geral da detecção de colisão, é baseado na operação de extrusão. O volume extrudado de um objeto é o espaço-temporal conjunto de pontos que representam a ocupação espacial do objeto ao longo de sua trajetória. Uma colisão entre dois objetos ocorre se, e somente se, seus volumes extrudados se cruzam.

Segundo (TU; YU, 2009) volume limitante é reconhecido como um bom método de detecção de colisão e é amplamente utilizado, estando presente no *Unity* (motor de jogo utilizado neste projeto). A ideia de um volume limitante é usar um corpo geométrico simples, um cubo por exemplo, para substituir um corpo geométrico complexo, por exemplo, um personagem. Primeiro, detectando a caixa delimitadora do objeto aproximadamente, apenas quando dois volumes limitantes se cruzam. Este método possue algumas limitações, pois irá causar um conflito de interseção em alguns objetos, haja vista que quando aplicado em objetos de volume muito variável, irá deixar alguns espaços vazios, onde a colisão poderá ser detectado, caso outro objeto entre em contato.

Na Figura 26 temos um exemplo de volume limitante aplicado ao motor de jogos Unity, o volume limitante é representado pelas linhas verdes em volta do quadrado no centro da tela.

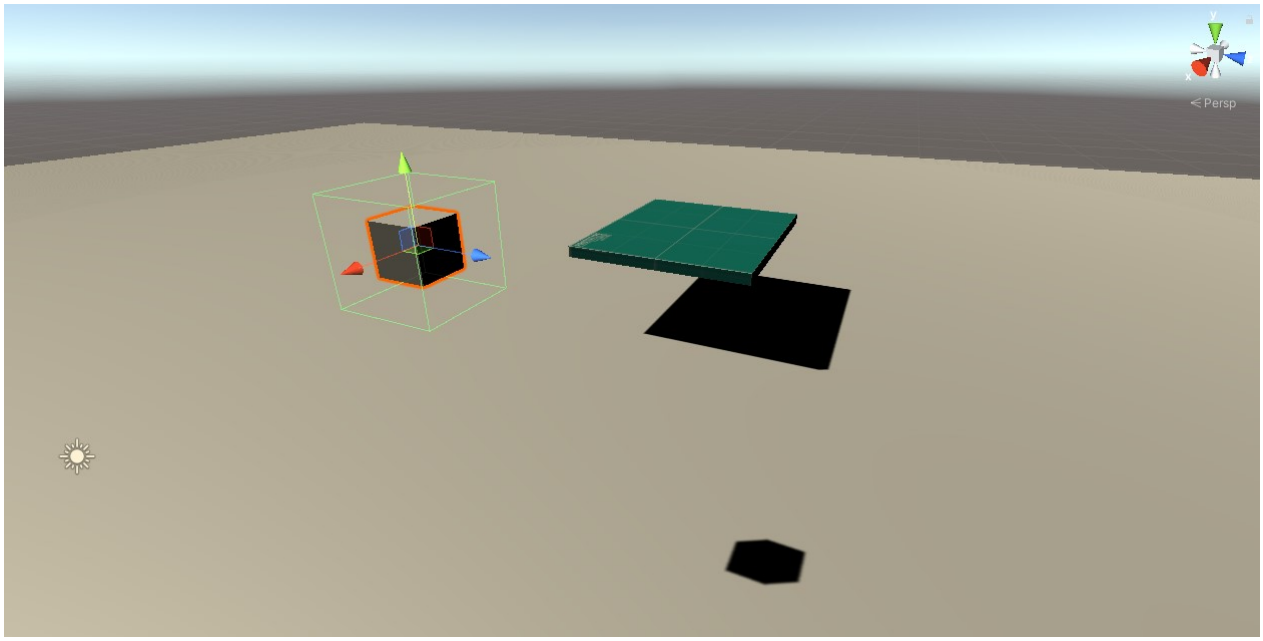


Figura 26 – Volume limitante

5.2 SELEÇÃO DE OBJETOS EM MUNDOS 3D

A seleção de objetos 3D frequentemente ocorre através do reconhecimento do clique do mouse em um objeto em um mundo 3D. (JIMÉNEZ; THOMAS; TORRAS, 2001) em sua abordagem, através do conceito *trackball interface* demonstra a interação do mouse com um objeto 3D, onde um clique no objeto torna possível movimenta-lo livremente pelo cenário. Um clique inicial, com o botão esquerdo do mouse, inicia a ação de rotação e a movimentação para um novo ponto do mundo define a rotação desse objeto, que pode é aplicada para ele, dessa maneira uma sensação de movimentação é transmitida para o observador. Em seu exemplo, soltar o mouse faz com que a rotação atual, advinda da movimentação, se torne permanente.

Dois aspectos podem ser observados nessa abordagem. Um evento de clique deve ser recordado para que a rotação ou transformação atual do objeto e o ponto inicial do clique sejam gravados. Outro ponto é o tratamento dos eventos de movimentação, que ocorre através da transformação da posição atual do mouse adicionado com a posição atual em que ocorreu o clique no objeto.

A realização da rotação de um objeto em um espaço tridimensional possuí uma limitação em relação ao posicionamento da câmera. Suponhamos que a câmera esteja direcionado em relação ao eixo Y, dessa maneira o processo de movimentação de um objeto estará limitado aos eixos X e Y, ou seja não será possível realizar a movimentação do objeto, no eixo Z, livremente. A movimentação até pode ocorrer, no entanto irá ocorrer devido a inclinação da câmera, seja ela para baixo ou para cima, e desta maneira não será possível retornar o objeto, na posição Z, em uma inclinação que seja contrário ao o que a câmera

se encontra.

A seguir temos um trecho do algoritimo usado no jogo em questão, para controlar os eventos de movimetação pelo mundo 3D.

Listing 5.1 – Procedimento de movimentação de um objeto

```
1  void Start ()
2  {
3          gameManager = GameObject.Find("GameManager");
4  }
5
6  void Update ()
7  {
8          if (Input.GetMouseButtonDown (0)) {
9                  RaycastHit hitInfo;
10
11                 if (Target == GetClickedObject (out hitInfo)) {
12                         _mouseState = true;
13                         screenSpace = Camera.main.
                               WorldToScreenPoint (Target.transform.
                               position);
14                         offset = Target.transform.position -
                               Camera.main.ScreenToWorldPoint (new
                               Vector3 (Input.mousePosition.x, Input
                               .mousePosition.y, screenSpace.z));
15                 }
16         }
17
18         if (Input.GetMouseButtonUp (0)) {
19                 _mouseState = false;
20         }
21
22         if (_mouseState) {
23                 GameManager gameManagerScript = gameManager.
                       GetComponent<GameManager>();
24
25                 sceneName = SceneManager.GetActiveScene().name;
26                 sceneNameToChar = sceneName.ToCharArray();
27         buildedSceneName = "Level" + sceneNameToChar[5] + "Etapa
               " + (Char.GetNumericValue(sceneNameToChar[sceneName.
               Length - 1]) + 1);
```

```
28
29                  int numeroDaFaseAtual = sceneNameToChar[5];
30
31                  if (numeroDaFaseAtual == '1') {
32                          distanciaZEmRelacaoACameraEObjeto = 38;
33                  } else if (numeroDaFaseAtual == '2') {
34                          distanciaZEmRelacaoACameraEObjeto = 13;
35                  } else if (numeroDaFaseAtual == '3') {
36                          distanciaZEmRelacaoACameraEObjeto = 330;

37                  } else if (numeroDaFaseAtual == '4') {
38                          distanciaZEmRelacaoACameraEObjeto = 6;
39                  }
40
41                  //keep track of the mouse position
42                  var curScreenSpace = new Vector3 (Input.
                        mousePosition.x, Input.mousePosition.y,
                        distanciaZEmRelacaoACameraEObjeto);
43
44                  //convert the screen mouse position to world
                        point and adjust with offset
45                  var curPosition = Camera.main.ScreenToWorldPoint
                        (curScreenSpace) + offset;
46
47                  //update the position of the object in the world
48                  Target.transform.position = curPosition;
49          }
50  }
```

5.3 DETECÇÃO DE OBJETOS EM RECIPIENTES

Utilizando a lógica de arrastar e soltar os objetos pela fase, foi possível desenvolver a lógica de preencher recipientes com objetos. Para tal foi necessário utilizar o sistema de colisão do Unity, onde, para que os objetos sejam reconhecidos em um determinado espaço é inserido um *GameObject* vazio que atua apenas como auxiliar no processo da colisão. Na Figura 27, por exemplo, temos um objeto vazio, onde nele é adicionado é adicionado o componente *Box collider* e configurado o atributo *is trigger* como ativado. Desta maneira, com o auxílio do script *BoxObjectCounter* é possível observar a entrada e saída de objetos no espaço delimitado.

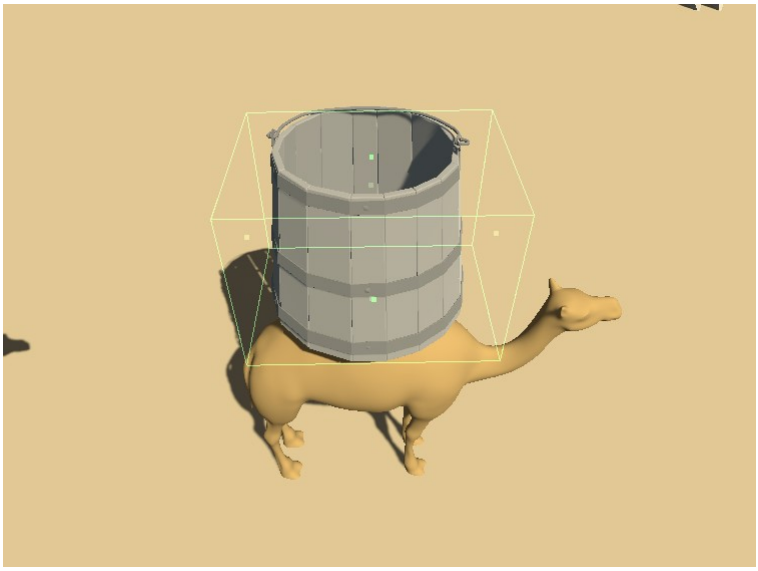


Figura 27 – Espaço de colisão

A seguir temos um trecho do algoritimo usado no jogo em questão, para detectar objetos em recipientes.

Listing 5.2 – Procedimento de detecção de colisão

```
1  void OnTriggerEnter(Collider other) {
2      GameManager gameManagerScript = gameManager.GetComponent<
           GameManager>();
3
4      var sceneName = SceneManager.GetActiveScene().name;
5
6      var sceneNameToChar = sceneName.ToCharArray();
7
8      var numeroDaFase = sceneNameToChar[5];
9      var numeroDaFasaEmValorNumerico = Char.GetNumericValue(
           numeroDaFase);
10     var numeroDaEtapa = (Char.GetNumericValue(sceneNameToChar[
           sceneName.Length - 1]));
11
12     var resposta = 0;
13
14     if(other.CompareTag("Object")) {
15         qtdObjectsInBox += 1;
16     }
17 }
```

5.4 DINÂMICA DOS ÁUDIOS

O jogo possuí quatro músicas, cada fase possuí um tipo de música, que toca durante toda sua extensão. Além disso está presente em cada inicio de etapa uma breve introdução da questão apresentada, onde foi gerado diversos áudios referentes as questões propostas. Os áudios são configurados em cada uma das câmeras, presentes nas diversas cenas do jogo, tendo em vista que cada cena possuí uma câmera, desta maneira é possível distribuir os áudios de acordo com cada fase. Em relação a música tocada nas fases, os seus áudios foram configurados para tocarem em *loop*.

6 DESENVOLVIMENTO DO JOGO

Após a escolha do tema do jogo que é o ensino de 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.

6.1 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.

6.2 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. 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.

6.3 DESIGN DE NÍVEIS

Design de níveis é definido pela criação de ambientes, cenários ou missões em um jogo eletrônico. (NOVAK, 2010) 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.

Visando isso, cada fase foi projetada para abordar um tema específico, esses temas são: soma, subtração, divisão e multiplicação.

6.3.1 FASE 1

A fase 1 é iniciada com a leitura em voz alta da pergunta proposta, é composta por um ambiente arenoso. A interação do jogador acontece entre as pedras, a direita do cenário, e dois camelos a esquerda do cenário. O objetivo é que o jogador insira a quantidade desejada de pedras nos baldes, acima dos camelos, e então busquem responder a questão corretamente. Por exemplo, na etapa 1, onde a pergunta é: quanto é 2 + 2, será necessário que o jogador insira duas pedras no balde da esquerda e duas pedras no balde da direita, só assim será capaz de avançar para a próxima etapa.

Figura 28 – Fase 1

É composta por três etapas, onde cada etapa apresenta a seguinte pergunta:

- Quanto é 2 + 2 ?
- Quanto é 2 + 6 ?
- Quanto é 3 + 3 ?

6.3.2 FASE 2

A fase 2 é iniciada com a leitura em voz alta da pergunta proposta, é composta por um floresta, onde está havendo um pinique. A interação do jogador acontece entre as frutas, a direita do cenário, e duas mesas a esquerda do cenário. O objetivo é que o jogador insira a quantidade desejada de frutas na cesta, acima da mesa, e então busquem responder a questão corretamente. Por exemplo, na etapa 1, onde a pergunta é: quanto é 4 - 2, será necessário que o jogador insira duas frutas na cesta, só assim será capaz de avançar para a próxima etapa.

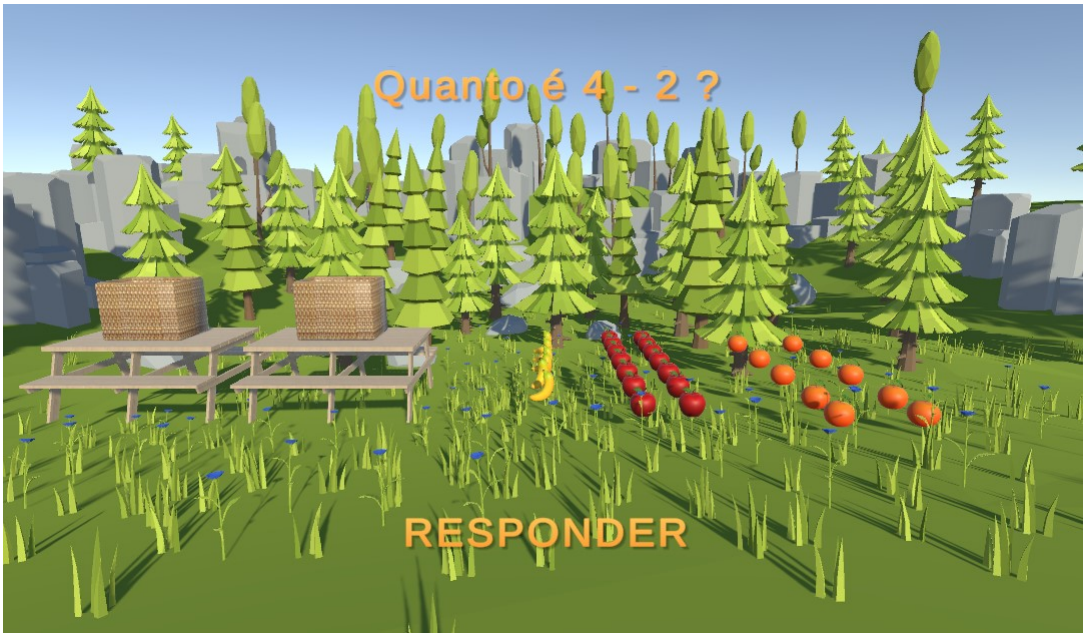


Figura 29 – Fase 2

É composta por três etapas, onde cada etapa apresenta a seguinte pergunta:

- Quanto é 4 - 2 ?
- Quanto é 7 - 4 ?
- Quanto é 9 - 5 ?

6.3.3 FASE 3

A fase 3 é iniciada com a leitura em voz alta da pergunta proposta, é composta por um ambiente gelado. A interação do jogador acontece entre os pinguins, na parte inicial do cenário, e a casa iglu, na parte inferior do cenário. O objetivo é que o jogador insira a quantidade desejada de pinguins ao lado da casa iglu, e então busquem responder a questão corretamente. Por exemplo, na etapa 1, onde a pergunta é: quanto é 2 * 2, será necessário que o jogador insira dois grupos de pinguins no ao lado da casa iglu, só assim será capaz de avançar para a próxima etapa.

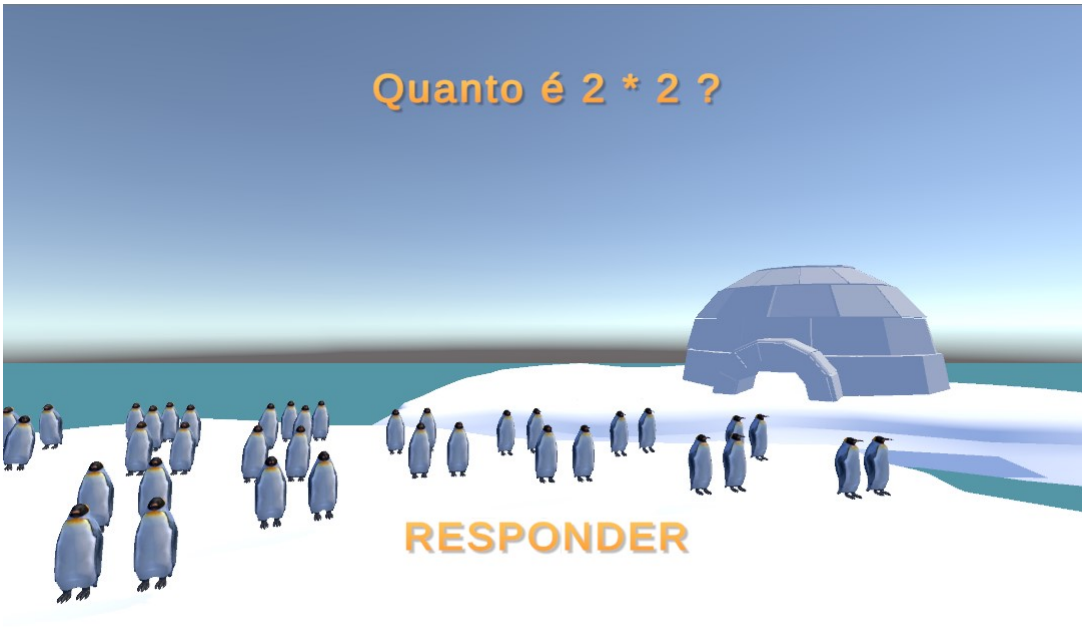


Figura 30 – Fase 1

É composta por três etapas, onde cada etapa apresenta a seguinte pergunta:

- Quanto é 2 * 2 ?
- Quanto é 3 * 8 ?
- Quanto é 4 * 2 ?

6.3.4 FASE 4

A fase 4 é iniciada com a leitura em voz alta da pergunta proposta, é composta por um floresta e um riacho. A interação do jogador acontece entre os patos, a beira do riacho, e o riacho. O objetivo é que o jogador insira a quantidade desejada de patos no riacho, e então busquem responder a questão corretamente. Por exemplo, na etapa 1, onde a pergunta é: quanto é 8 / 2, será necessário que o jogador insira dois patos no riacho, só assim será capaz de avançar para a próxima etapa.

Figura 31 – Fase 4

É composta por três etapas, onde cada etapa apresenta a seguinte pergunta:

- Quanto é 8 / 2 ?
- Quanto é 9 / 3 ?
- Quanto é 8 + 4 ?

6.3.5 ACESSO AO JOGO

O jogo está disponível no seguinte link: https://drive.google.com/drive/folders/1ROz3kVHfiP$_w$*kAan*96*pCH*
sharing.

Para executar o jogo entre no diretório "matematica-basica"e execute o arquivo "matematica-basica-demo.exe".

7 CONCLUSÕES

O processo tradicional de ensino/aprendizagem da matemática não tem contemplado a interação do aluno com os conceitos matemáticos, focando mais em um aprendizado passivo, levando a uma dificuldade na assimilação dos conteúdos e fomentando apenas a memorização. Acredita-se que a introdução do lúdico é uma forma mais interessante e dinâmica para engajar o aluno e introduzi-lo no universo da matemática.

O projeto atingiu seu principal objetivo, o de introduzir uma dose de ludicidade e interatividade no aprendizado da matemática básica. Embora o objetivo tenha sido concluído, ainda há muitas melhorias que podem ser aplicadas ao jogo desenvolvido, e também há muito a ser estudado a respeito do tema. O jogo será hospedado em um domínio e ficará disponível gratuitamente para a comunidade.

A idealização do jogo ocorreu através da construção de um *GDD* e implementado através do motor de jogos Unity.

REFERÊNCIAS

ABEU colégios. ***Relação de conteúdos programaticos ensino fundamental 1º ano***. 2020. Disponível em: <https://www.abeucolegios.com.br/docs/conteudos/conteudo_1_ano_EF.pdf>. Acesso em: 28 nov. 2020.

ABEU colégios. ***Relação de conteúdos programaticos ensino fundamental 2º ano***. 2020. Disponível em: <https://www.abeucolegios.com.br/docs/conteudos/conteudo_2_ano_EF.pdf>. Acesso em: 28 nov. 2020.

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.

CARNINE ASHA K. JITENDRA, J. S. D. **A Descriptive Analysis of Mathematics Curricular Materials from a Pedagogical Perspective**. 1997.

CAZELLI, W. de F. ***Desenvolvimento de Um Jogo 3D Para o Ensino de Matemática Básica***. 2019.

GOLDSTONE, W. ***Unity game development essentials***. [S.l.]: Packt Publishing Ltd, 2009.

GRAVEMEIJER, K. **Educational Development and Developmental Research in Mathematics Education**. 1994.

HEJLSBERG, A. et al. ***The C# programming language***. [S.l.]: Pearson Education, 2008.

INEP/MEC. ***Programa Internacional de Avaliação de Alunos (PISA) 2018. Relatório nacional***. Brasília-DF, 2018.

JIMÉNEZ, P.; THOMAS, F.; TORRAS, C. **3D collision detection: a survey**. *Computers & Graphics*, Elsevier, v. 25, n. 2, p. 269–285, 2001.

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

LATANSIO CARLA FLORENTINO, J. A. d. S. J. R. S. B. M. R. R. M. T. M. T. A. P. V. B. d. S. A. O. ***Game Design Document: "As Aventuras de Jackie e Tony"***. 2011.

MATIFIC. ***Matific***. 2020. Disponível em: <https://www.matific.com/>. Acesso em: 28 nov. 2020.

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

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

RCNEI. ***REFERENCIAL CURRICULAR NACIONAL PARA A EDUCAÇÃO INFANTIL***. 1998. Disponível em: <http://portal.mec.gov.br/seb/arquivos/pdf/rcnei_vol1.pdf>.

ROUSE, R. ***Game Design: Theory and Practice: Theory and Practice***. [S.l.]: Jones & Bartlett Learning, 2004.

SANSONE, A. T. **Game design documents: Changing production models, changing demands**. *Computer games and technical communication: Critical methods and applications at the intersection*, Ashgate, New York, NY, p. 109–124, 2014.

SCHELL, J. ***The Art of Game Design: A book of lenses***. [S.l.]: CRC press, 2008.

STEENBRUGGE MARTIN VALCKE, A. D. H. V. **Mathematics learning difficulties in primary education: Teachers' professional knowledge and the use of commercially available learning packages**. 2010.

TEOTÔNIO, W.; ARAÚJO, A. ***Desenvolvimento de Jogos para Dispositivos Móveis utilizando o Motor de Jogo Unity 3D: um Estudo de Caso***. 2017.

TU, C.; YU, L. **Research on Collision Detection Algorithm Based on AABB-OBB Bounding Volume**. In: *2009 First International Workshop on Education Technology and Computer Science*. [S.l.: s.n.], 2009. v. 1, p. 331–333.

UNITY. Disponível em: <https://docs.unity3d.com/Manual/index.html>. Acesso em: 12 jul. 2021.