

Carpilator: Jogo digital educativo para ensino de programação

Aiesa Moraes Rosa¹, João Vitor Xavier², Leonardo Bravo Estácio³

¹Instituto Federal de Santa Catarina - Câmpus Lages
R. Heitor Villa Lobos, 225 - São Francisco, Lages - SC, 88506-400
Curso de Ciência da Computação

aiesamoraesrosa@gmail.com, fast.jonh@gmail.com, leonardo.bravo@ifsc.edu.br

Resumo. A evasão de cursos superiores na área da tecnologia são muito comuns devido a dificuldade que de abstração dos discentes em matérias voltadas a programação, dessa forma jogos digitais têm sido usados como estratégia de gamificação voltado ao contexto educacional para o ensino-aprendizagem de programação. O objetivo deste artigo foi desenvolver um jogo digital para auxiliar o professor no ensino de programação em sala de aula. O trabalho teve início com uma pesquisa de campo, seguindo por uma apresentação dos passos necessários para o desenvolvimento do jogo digital. Os resultados encontrados foram satisfatórios e o jogo produzido poderá auxiliar um docente ao ministrar conteúdos introdutórios de programação.

Abstract. Dropping out of higher education courses in the area of technology is very common due to the difficulty that students have in abstracting subjects related to programming, thus digital games have been used as a gamification strategy aimed at the educational context for the teaching-learning of programming. The objective of this article was to develop a digital game to assist the teacher in teaching programming in the classroom. The work began with a field survey, followed by a presentation of the necessary steps for the development of the digital game. The results found were satisfactory and the game produced can help a teacher to teach introductory programming content.

1. Introdução

A disciplina de programação faz parte da grade curricular, principalmente, dos cursos de graduação de tecnologia, como por exemplo Sistemas de Informação, onde um em cada três alunos ingressantes finalizam o curso de Ciência da Computação, no qual um a cada quatro se forma. É possível pontuar que a falta de compreensão do raciocínio lógico da programação é uma das causas de reprovação e desistência (Silva et al., 2018a).

A evasão no ensino superior, é o ato de desistir e trancar o curso, dessa forma, para diminuir a evasão dos alunos, técnicas têm sido desenvolvidas, uma dessas estratégias é a gamificação, uma técnica que usa a competitividade para atingir um determinado objetivo (Silva et al., 2018a). Neste artigo, voltado ao contexto educacional, usando jogos para o ensino-aprendizagem de programação.

Segundo Grossmann (2019) a procura por profissionais na área de Tecnologia da Informação (TI) no país será de 420 mil pessoas até o ano de 2024, entretanto, o número se contrapõe à baixa quantidade na formação de profissionais por ano e desperta um alerta

para o risco de um apagão de pessoas qualificadas. Um dos fatores que podem causar isso, é porque o processo de aprendizagem dos conceitos iniciais de programação é complexo e precisa de um nível de abstração que não está presente nos discentes que estão tendo um primeiro contato com a área da programação e assim, acabam desanimando e evadindo (Zanetti e Oliveira, 2015).

Para tentar resolver o problema supracitado, o objetivo deste trabalho é desenvolver um jogo digital para auxiliar o professor no ensino de programação em sala de aula.

São objetivos específicos:

- Realizar uma pesquisa com os alunos de graduação que cursaram disciplinas de programação no Instituto Federal de Santa Catarina (IFSC) - Câmpus Lages a fim de verificar as dificuldades ao ter o primeiro contato com o conteúdo;
- Elaborar o design e desenvolver o novo interpretador do jogo digital;
- Testar com alunos de graduação do IFSC - Câmpus Lages o jogo digital desenvolvido e apresentar um relatório com os dados levantados.

Além desta seção introdutória, este documento é composto das seguintes seções: A seção 2 apresenta a metodologia; A seção 3 mostra o referencial teórico necessário para ao entendimento do trabalho; A seção 4 registra o desenvolvimento do trabalho; E, na seção 5, as conclusões sobre o trabalho são apresentadas.

2. Metodologia

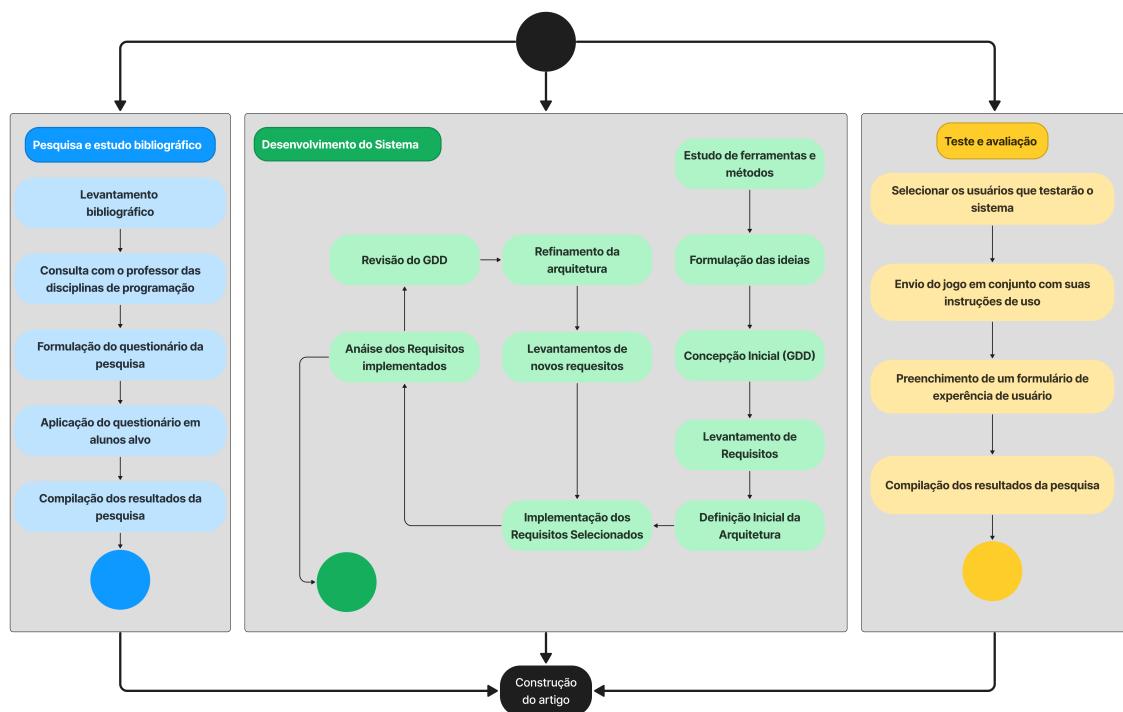


Figura 1. Metodologia geral

Conforme apresentado na Figura 1, o trabalho será composto por 3 etapas. A primeira etapa consiste em uma pesquisa de campo, a segunda etapa na apresentação dos passos necessários para o desenvolvimento do jogo digital e a terceira etapa no teste de qualidade do jogo desenvolvido nos critérios de usabilidade e funcionalidade e a confecção de um relatório apontando as experiências dos alunos envolvidos.

A primeira etapa consiste em realizar um questionário de múltipla escolha visando identificar os principais problemas apontados pelos alunos de graduação, e, a partir daí, formular estratégias para serem adicionadas ao jogo digital para suprir essas dificuldades. O questionário será formulado junto ao professor da disciplina de Introdução a Programação do IFSC Câmpus Lages.

A segunda etapa consiste no desenvolvimento do jogo digital na plataforma *Unity*, conhecida como motor de jogos que possui como linguagem de programação o C#. Por fim, na terceira etapa o jogo digital estará pronto para ser testado pelos alunos do IFSC - Câmpus Lages e a partir destes testes será elaborado um relatório com as considerações e experiência dos testantes.

Do ponto de vista da natureza, esse trabalho é uma pesquisa aplicada. Quanto à abordagem do problema a pesquisa é qualitativa. Referente aos objetivos, esse artigo classifica-se como sendo uma pesquisa exploratória. Considerando os procedimentos técnicos, a pesquisa é bibliográfica.

3. Referencial teórico

A seção apresenta os assuntos fundamentais para o entendimento do trabalho como um todo. Estando dividida em 2 subseções. A subseção 3.1 descreve a revisão sistemática, a 3.2 os trabalhos correlatos e a subseção 3.3 descreve as tecnologias utilizadas para o desenvolvimento do jogo digital.

3.1. Revisão sistemática

Para pesquisar trabalhos semelhantes a este projeto foi realizada uma revisão sistemática na literatura. A ferramenta utilizada foi o *Google Acadêmico* através da string de consulta “Jogos digitais voltados para o ensino da programação”. Foram selecionados os 20 primeiros artigos com período de tempo como parâmetro de 8 anos para serem analisados. Após a leitura dos resumos e objetivos foi atribuído um valor de relevância, de 1 a 5, para cada artigo, sendo 1 pouco relevante e 5 muito relevante. Dos 20 trabalhos selecionados, foi feita uma leitura completa nos 8 de maiores notas e desses foram escolhidos os 4 trabalhos de maior relevância e importância para serem resumidos nesse artigo.

3.2. Trabalhos correlatos

O trabalho de Andrade et al. (2016) descreve, uma abordagem que visa estimular a programação no ensino médio por meio do desenvolvimento de jogos digitais, dando suporte à multidisciplinaridade intrínseca ao processo, através da ferramenta *Scratch*. Foi estimulada a competitividade entre os grupos e os resultados apontaram que os alunos conseguiram documentar o processo de produção, desenvolveram habilidades cognitivas relacionadas à programação e empregaram heurísticas de jogabilidade nos jogos produzidos.

O projeto proposto por Silva et al. (2018b) tem como foco replicar uma revisão sistemática da literatura (RSL) referente aos anos de 2008 a 2012, e apresentar uma nova RSL sobre o uso de jogos digitais focado no ensino de programação para iniciantes em Computação no nível superior no Brasil abrangendo 2013-2017. Foram comparados os resultados obtidos em ambos os períodos da última década e identificadas as ferramentas e teorias pedagógicas que são aplicadas para o ensino-aprendizagem de programação.

Monclar et al. (2018) realizaram uma revisão, iniciando no ano de 2001 até 2018, buscando identificar jogos que promovam a evolução dos discentes na disciplina de programação no ensino superior. Também foram considerados aqueles que fundamentavam os pilares do pensamento computacional, uma vez que são capazes de despertar o interesse em um público mais amplo e mais jovem. São apresentados neste trabalho uma lista de 26 jogos, onde se destacam características importantes destes.

O trabalho de Pantaleão et al. (2017) descreve uma metodologia com o uso da ferramenta *Robocode* no ensino de algoritmos e programação em *Java* para alunos do Ensino Médio. A metodologia inclui a participação de discentes da graduação que cursaram a disciplina de programação. Os resultados observados mostram o interesse dos alunos do Ensino Médio pela programação e o potencial do *Robocode* como ferramenta lúdica de apoio ao ensino, também foi possível observar o quanto a competitividade influenciou a motivação dos estudantes em aprender e superar desafios.

Os artigos citados foram importantes para este trabalho, contribuindo com o entendimento do que já foi aplicado em jogos digitais voltados ao ensino de programação, levando em conta em qual nível de escolaridade aconteceu essa aplicação. Através deles investigamos a eficácia e resultados que obtiveram, erros e acertos e também a metodologia que cada autor utilizou.

3.3. Tecnologias para Desenvolvimento de Jogos Digitais

Nesta subseção apresentará as ferramentas que serão utilizadas no desenvolvimento do jogo digital proposto pelo trabalho. Todas as tecnologias possuem licenças gratuitas ou versões para estudantes.

3.3.1. Unity

Criada pela *Unity Technologies*, a plataforma *Unity* funciona como um motor de jogos digitais que trabalha com a linguagem de programação *C#*. Além de oferecer um ambiente para programadores experientes, também oferece recursos prontos que auxiliam os desenvolvedores a focarem mais no funcionamento do seu jogo (da Silva et al., 2016).

A *Unity* é uma das mais populares *engines* para jogos, uma vez que conta com mais de 770 milhões de pessoas que jogam os títulos criados na plataforma, entre os 1000 dos maiores títulos de jogos para plataformas móveis grátis. A *Unity* é responsável por 34% de 1000 dos maiores títulos de jogos para plataformas móveis grátis, sendo escolhida principalmente pela sua interface simples e completa. No site da plataforma é possível encontrar sua documentação completa e vários tutoriais que auxiliam na aprendizagem do funcionamento da *Unity* (da Silva et al., 2016). É nesta plataforma que o jogo será desenvolvido.

3.3.2. C#

O C# foi desenvolvido pela *Microsoft* e é uma linguagem de programação orientada a objetos que permite que os desenvolvedores criem vários tipos de aplicativos robustos e seguros que são executados no .NET (Wagner et al., 2022). Segundo Overflow (2021) é a quinta linguagem de programação mais utilizada e alguns de seus recursos são:

- Coletor de lixo: recupera de forma automática a memória ocupada por objetos não utilizados e inacessíveis;
- Tipos anuláveis: é uma proteção contra variáveis que não se referem a objetos aloca-dos;
- Manipulação de exceção: oferece uma abordagem estruturada e extensível para detecção e recuperação de erros;
- Expressões lambda: fornecem suporte a técnicas de programação funcional;
- Linguagem de consulta integrada (LINQ): padrão criado para permitir trabalhar com dados de qualquer fonte.

Neste artigo será usado no *Unity* por ser a linguagem de programação que a plataforma utiliza.

3.3.3. Adobe Photoshop

É um *software* que cria ou aprimora fotografias, imagens, vídeos, animações e ilustrações 3D. Essas funcionalidades tornaram-o uma das plataformas mais usadas em computadores e de maior destaque no mundo da fotografia e do design gráfico (De Andrade, 2019).

Pra este artigo, o *Adobe Photoshop* será utilizado para criação da interface gráfica para interação com os usuários, através da criação das telas do jogo digital, menu e botões.

3.3.4. GitHub

É uma plataforma de desenvolvimento colaborativo para projetos que utilizam o sistema de controle de versão *Git* e todo código armazenado pode ser aberto ou restrito, permitindo que usuários contribuam em projetos privados ou *Open Source* de qualquer lugar do mundo. Tais funcionalidades, podem levar a uma melhor tomada de decisões, redução de conflitos e aumento da produtividade (Portugal e do Prado Leite, 2015).

O *GitHub* será usado como repositório para esse projeto, sendo utilizado, principalmente, pelos seus recursos de edição simultânea, versionamento e histórico de alteração.

3.4. Trello

O *Trello* é um sistema de gerenciamento de projeto *online* e funciona como um *Kanban*, organizando as atividades em cartões e filas. Suas funcionalidades intuitivas possibilitam que qualquer time configure e personalize rapidamente os fluxos de trabalho de praticamente qualquer atividade (Trello, 2022).

Neste artigo, será usado para controle das tarefas de desenvolvimento do jogo digital.

4. Desenvolvimento

A seguinte seção contempla os passos realizados para a construção do jogo digital, apresentando a elaboração do *Game Design Document* (GDD), mostrando o questionário que foi aplicado com a turma que fez a disciplina de programação no IFSC Câmpus Lages no semestre 2022/1 e que foi usado para complementar o processo de levantamento de requisitos do jogo digital. Também apresenta o desenvolvimento do novo interpretador do jogo digital, lista de requisitos, ordem dos requisitos implementados, controle de versões e, por fim a programação na plataforma *Unity* com as descrições da arquitetura, movimentações do personagem, desenvolvimento das estradas e o teste de usabilidade do jogo.

4.1. GDD

O Game Design Documet (GDD) apresenta toda a documentação relacionada ao design do jogo digital, dessa forma, esta subseção é composta pelo *Gameplay*, descrição dos personagens, controles e câmera do jogo digital.

4.1.1. *Gameplay*

A *gameplay* consiste em 2 modos de jogos diferentes, o primeiro é o *user-only*, onde o usuário deve apenas descobrir quais *inputs* precisa ser dado ao programa para que chegue ao objetivo final e o *code*, onde o usuário insere um código que gere uma cena idêntica à sugerida pelo nível.

4.1.2. Personagens

- Carro: representa o processador executando o código do usuário;
- Estruturas do ambiente (estruturas da programação):
 - Estrutura condicional (*If*): representado por uma bifurcação na estrada;
 - Estrutura de repetição (*While*): representado por uma rotatória;
 - Estrutura de leitura (*Read*): representado por um acostamento onde o carro deve parar para o usuário inserir dados;
 - Estrutura de início;
 - Estrutura de saída;
 - Estrutura de final;
 - *Print*: representado por uma placa com informações.

4.1.3. Controles

A *gameplay* é baseada em poucas *inputs* do jogador, sendo elas:

- Botão *Play/Pause*: Continua ou pausa a movimentação do carrinho (execução do código);
- Botão *Reset*: Para reiniciar o programa;
- Caixa de Texto para *Input* do código: Onde o usuário digitará o código para ser executado ou para selecionar um nível;

4.1.4. Câmera

A câmera escolhida para o jogo possui o formato *Top Down*, isto é, de cima para baixo. Assim, ao aproximá-la o jogador pode ter mais clareza dos textos escritos e da organização das estruturas.



Figura 2. Imagem da câmera do jogo

4.2. Questionário aplicado para identificar problemas de aprendizado em disciplinas introdutórias de programação

Foi aplicado um questionário com a turma do semestre 2022/1 do IFSC Câmpus Lages, com o objetivo de identificar as principais dificuldades nos conteúdos que a turma estudou durante o semestre e averiguar se os alunos consideram alguma outra forma de ensino-aprendizagem diferente da que é aplicada.

O resultados da figura 3 diz respeito sobre as dificuldades de cada aluno nos temas que são abordados na disciplina de programação, usando o seguinte critério de grau de dificuldade na votação: 1 - muito fácil; 2 - fácil; 3 - mais ou menos; 4 - difícil e 5 - muito difícil.

Marque o grau de dificuldade que teve em cada tópico da disciplina descrito abaixo. Sendo 1 - muito fácil; 2 - fácil; 3 - mais ou menos; 4 - difícil e 5 - muito difícil.

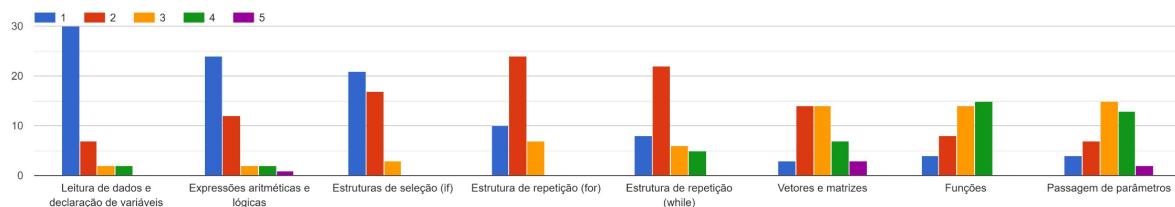


Figura 3. Gráfico 1

Pensando em trazer como sugestão a instituição e verificar se o nosso projeto despertaria o interesse dos discentes. Questionamos se existiria algum outro recurso de ensino-aprendizagem que eles considerariam eficaz na disciplina. A figura 4 nos mostra os resultados mais votados pelos alunos, podendo observar que a inclusão dos jogos digitais durante as aulas está em segundo lugar com 43,9% como uma ferramenta válida para melhorar a aprendizagem.

Qual estratégia facilitaria a aprendizagem da disciplina?

41 respostas

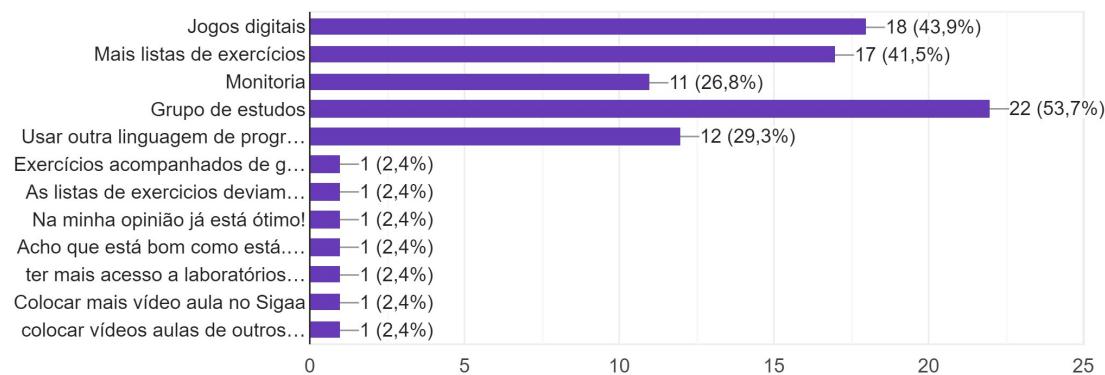


Figura 4. Gráfico 2

Portanto, foi possível verificar que os alunos temem principalmente, a parte lógica e de passagem de parâmetros e observar que os alunos consideraram interessante a ideia de um complemento com ferramentas lúdicas na hora do ensino-aprendizagem da disciplina. Apesar da detecção que passagem de parâmetros é um dos assuntos que os discentes mais tem dificuldades, ela não foi implementada pois o foco do jogo digital nesse primeiro momento é nas funções básicas.

4.3. Lista de Requisitos

Lista de requisitos para o Interpretador.

- Analisador Léxico: Responsável por ler o código digitado pelo usuário e reconhecer os *tokens* da linguagem;
- Analisador Sintático: Responsável por transformar esses *tokens* em uma árvore sintática abstrata (AST) que possa ser lida pelo Executor;
- Gerador de Cena: programa que trabalhará com o *Unity* para gerar a cena condizente ao código digitado pelo usuário a partir da AST;
- Executor: programa que recebe a AST e é responsável por traduzir o código para C# em tempo de execução do jogo e retornar os devidos comandos ao jogo *Unity*.

4.4. Desenvolvimento do interpretador que será utilizado no jogo digital

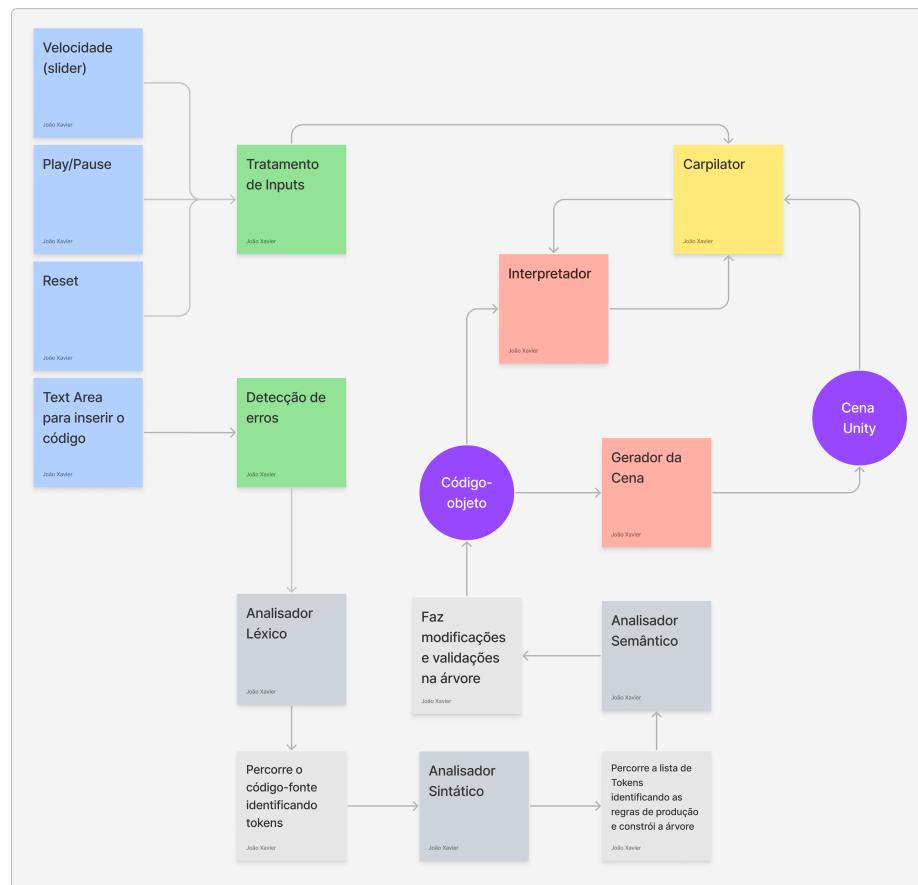


Figura 5. Fluxograma do Sistema

A Figura 5 mostra um fluxograma geral do sistema, onde em azul temos as inputs do usuário, em verde os módulos que fazem o tratamento das inputs para alterar o comportamento do jogo, em cinza as etapas do "Compilador" e em roxo os objetos de saída do programa.

A implementação iniciou-se pelo interpretador da linguagem de programação do jogo, a C#. Dividido em partes:

- Analisador Léxico;

Essa etapa recebe como *input* o próprio código-fonte da linguagem em forma de texto, e vai transformar esse código em uma lista de *tokens* através de um Autômato finito determinístico, que vai gerar tais *tokens* buscando dentro da *string* os padrões definidos da linguagem.

- Analisador Sintático;

Essa etapa recebe como *input* a lista de *tokens* da etapa anterior, e é responsável por agrupar os *tokens* em uma árvore sintática abstrata através das regras gramaticais definidas para formar os construtos da linguagem

- Gerador de Cena;

Essa etapa recebe a árvore sintática abstrata e é responsável por gerar a Cena 3D no *Unity* com todas as sub-cenas e componentes do game.

- Executor.

Por fim o executor é o responsável por dar vida à cena, receber as inputs do usuário passar para o interpretador, controlar o código de acordo com elas e mostrar toda esse comportamento para o usuário graficamente.

A Figura 6 mostra o método responsável por reconhecer as gramáticas da linguagem com base na lista de *tokens* que foi obtida pela etapa da análise léxica.

```
private Statement Statement()
{
    return Current.TokenType switch
    {
        TokenType.Identifier => Identifier(),
        TokenType.ReservedWord => ReservedWord(),
        _ => throw new UnexpectedToken(Current, TokenType.Identifier, TokenType.ReservedWord),
    };
}
```

Figura 6. Função para reconhecimento sintático dos comandos do programa

A Figura 7 mostra o método responsável por reconhecer as possíveis gramáticas ao analisar um identificador.

```
private Statement Identifier()
{
    var identifier = (Identifier)Assert(TokenType.Identifier);

    return Current.TokenType switch
    {
        TokenType.ParenthesisOpen => FunctionCall(identifier),
        TokenType.Attribution => AssignmentExpression(identifier),
        TokenType.Identifier => VariableDeclaration(identifier),
        _ => throw new UnexpectedToken(identifier, TokenType.Identifier, TokenType.ParenthesisOpen, TokenType.Attribution),
    };
}
```

Figura 7. Função para reconhecimento sintático de identificadores

A Figura 8 mostra o método responsável por reconhecer a gramática de uma chamada de função.

```

private FunctionCall FunctionCall(Identifier identifier)
{
    Assert(TokenType.ParenthesisOpen);

    var parameters = new List<IValuable>();

    DoUntil(TokenType.ParenthesisClose, () =>
    {
        parameters.Add(GetExpression());
        AssertOptional(TokenType.Comma);
    });

    var functionCall = new FunctionCall(identifier, parameters);

    Assert(TokenType.ParenthesisClose);
    AssertOptional(TokenType.Semicolon);

    return functionCall;
}

```

Figura 8. Função para reconhecimento sintático de uma chamada de função

A Figura 9 mostra o método responsável por reconhecer a gramática de uma expressão.

```

private IValuable GetExpression()
{
    var leftValue = Assert(
        TokenType.StringValue,
        TokenType.IntValue,
        TokenType.FloatValue,
        TokenType.BoolValue,
        TokenType.Identifier
    );

    if (leftValue.TokenType == TokenType.Identifier && Current.TokenType == TokenType.ParenthesisOpen)
    {
        return FunctionCall((Identifier)leftValue);
    }

    if (IsEOL())
    {
        return (IValuable)leftValue;
    }

    return new BinaryExpression((IValuable)leftValue, GetOperator(), GetExpression());
}

```

Figura 9. Função para reconhecimento sintático de expressões

4.5. Desenvolvimento do jogo digital no *Unity*

Nas seguintes seções serão apresentadas as classes desenvolvidas para implementar o jogo na *Unity*, essa implementação tem uma arquitetura baseada em componentes, ou seja, baseia-se em componentes independentes, substituíveis e modulares, as quais gerenciam a complexidade e motivam a reutilização.

4.5.1. Arquitetura do jogo digital

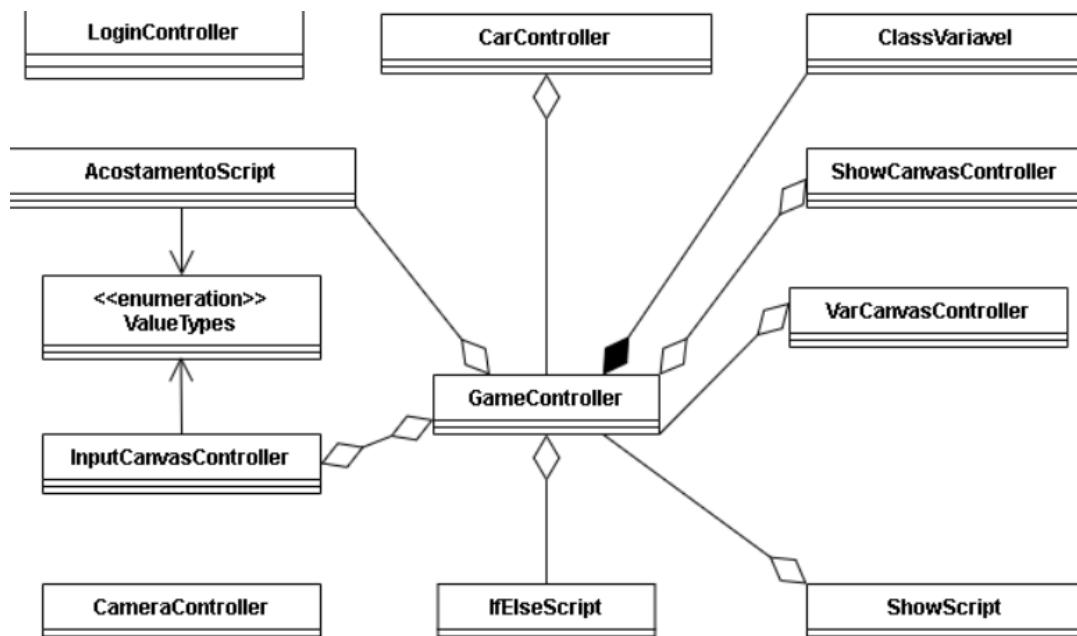


Figura 10. Arquitetura do jogo digital

4.5.2. *GameObject* Acostamento

Este *GameObject* tem como objetivo manipular o carro e fazer que quando ele passar por cima do *Collider* apareça na tela a caixa de *input*. Dentro do *GameObject* Acostamento temos os seguintes componentes: *Transform*, *BoxCollider* e o *script* Acostamento, como podemos observar na figura 11 e na 12.

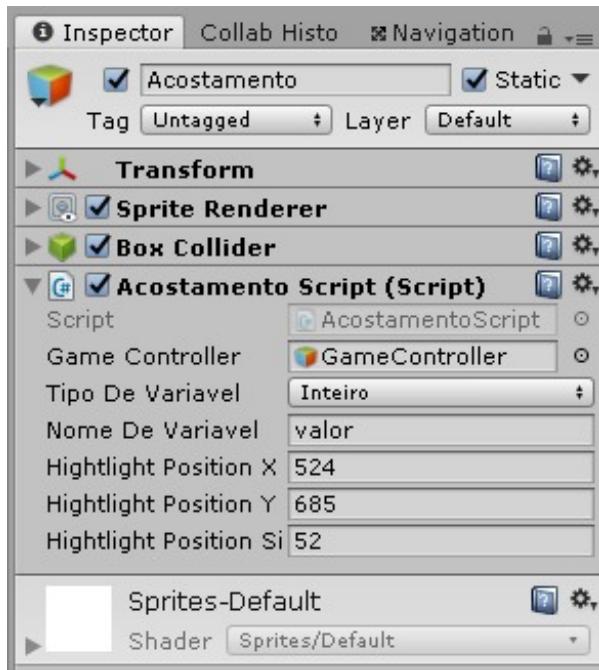


Figura 11. GameObject Acostamento

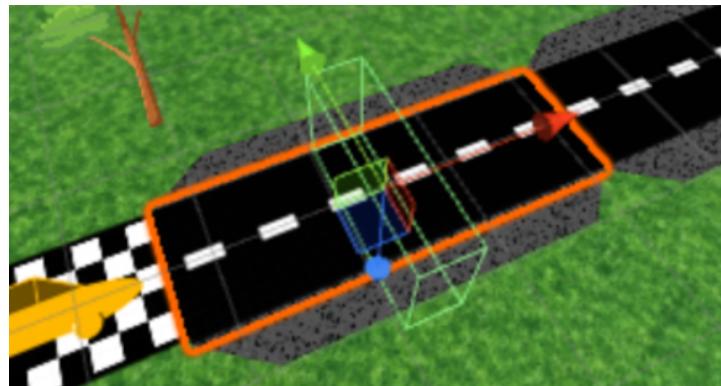


Figura 12. Parte do jogo que representa o GameObject Acostamento

Conforme, é possível ver na figura 13, o atributo da linha 10 é responsável por fazer a comunicação com o *GameObject Controller*, o da linha 11 e 12 definem qual é o nome e o tipo da variável que será lida pelo *GameObject InputCanvas*. Os três atributos das linhas 14,15,16 definem o X, Y e Z do quadrado amarelo que faz o *Highlight* do código. Na linha 19 o atributo segura a referência ao componente *Collider* do *GameObject* e na linha 21 o atributo faz referência ao componente *GameController*, que faz a comunicação entre todos os objetos da cena.

```
1  using UnityEngine;
2
3  namespace Assets.Scripts
4  {
5
6
7      public class AcostamentoScript : MonoBehaviour {
8
9
10     public GameObject GameController;
11     public ValueTypes TipoDeVariavel;
12     public string NomeDeVariavel;
13
14     public float HightlightPositionX;
15     public float HightlightPositionY;
16     public float HightlightPositionSize;
17
18
19     private Collider _col;
20
21     private GameController _gameControllerScript;
22 }
```

Figura 13. GameObject Acostamento - Parte 1 do código

No método *Start*, como podemos observar na figura 14 ao dar o *play* no *Unity*, o *script* pega a referência do componente *Collider* definido dentro do *GameObject* Acostamento e coloca no atributo. Além disso, pega a referência do componente do *script GameController* de dentro do Objeto *GameController*.

```
22
23
24      public void Start () {
25
26          _col = GetComponent<Collider>();
27
28          _gameControllerScript = GameController.GetComponent<GameController>();
29
30
31 }
```

Figura 14. GameObject Acostamento - Parte 2 do código

O método *OnTriggerEnter* é chamado sempre que ocorre uma colisão entre *Colliders* de diferentes objetos, sendo chamado toda vez que um outro objeto invade um *Collider* definido como *Trigger*. A diferença entre *Collider* e o *Trigger* é uma área de ativação e o *Collider* impede que outros objetos invadam essa região. Nas linhas 35 e 36 da figura 15, o *_gameControllerScript* configura o *GameObject TextHighlighter* que mostra em qual ponto do código o carrinho está e o *GameController* é um objeto centralizador que fará a comunicação entre os objetos, todo este processo é possível observar na figura 15.

```

32     public void OnTriggerEnter(Collider other)
33     {
34         _gameControllerScript.SetHighlightTextColor(255f,255f,0f);
35         _gameControllerScript.MoveHighlightToPosition(HightlightPositionX, HightlightPositionY, HightlightPositionSize);
36
37         _gameControllerScript.StopCar();
38
39         _gameControllerScript.ActivateInputCanvas(NomeDeVariavel, TipoDeVariavel);
40
41         _col.enabled = false;
42     }
43 }
44
45
46
47

```

Figura 15. GameObject Acostamento - Parte 3 do código

4.5.3. *GameObject* câmera

A câmera fica posicionada em cima da cena e apontando para baixo. Dentro da câmera, o script *CâmeraController* controla tanto o posicionamento do *GameObject Camera*, como o zoom na tela. Na figura 16 o eixo X é representado pelo vetor vermelho e o eixo Z é representado pelo vetor azul.

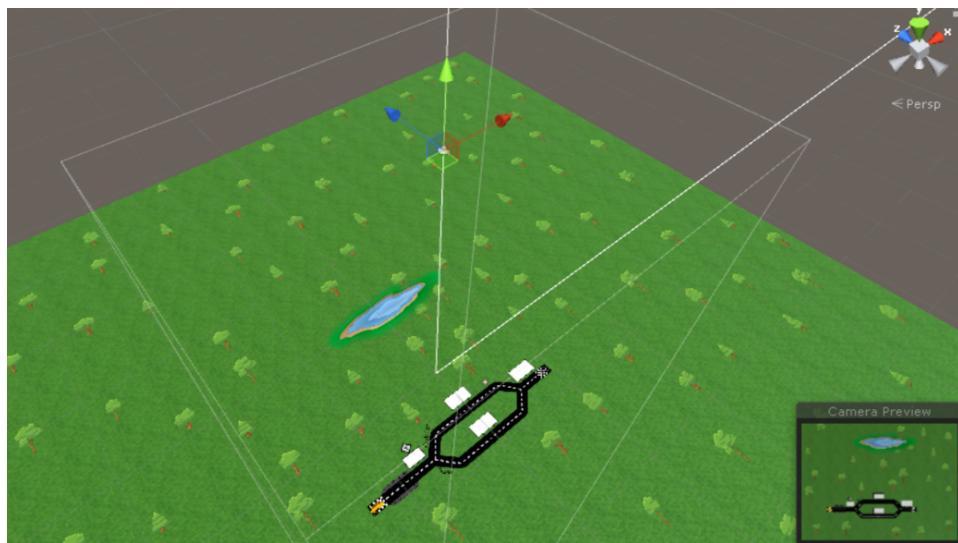
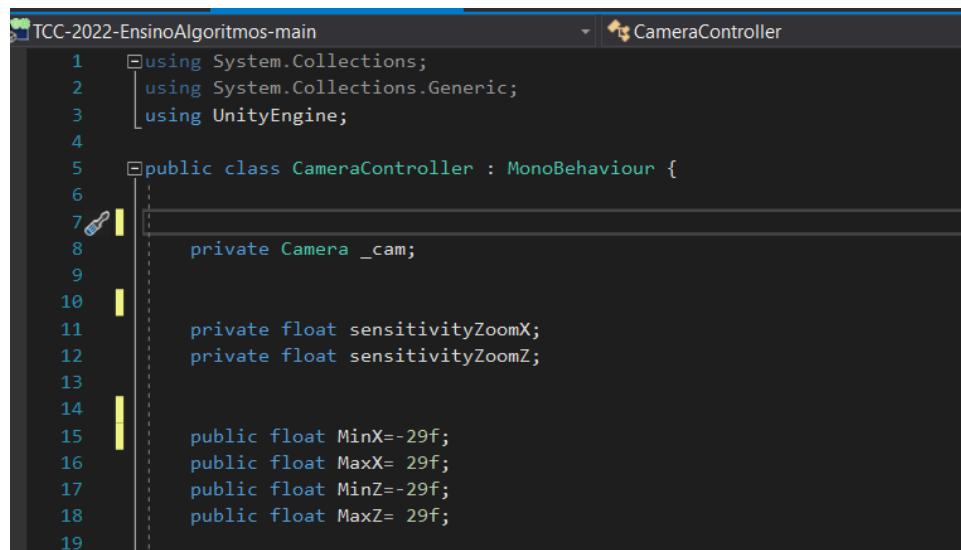


Figura 16. Parte do jogo que mostra a Script Câmera

Na figura 17, na linha 8 vemos o atributo que fará referência a câmera principal na cena, já as linhas 11 e 12 são atributos que controlam a velocidade que a tela movimenta-se quando modificamos o X e Z da câmera, tais atributos são necessários pois quando a câmera está com muito zoom, ela deve movimentar-se mais lentamente. Nas linhas 15,16,17 e 18 são atributos que controlam o máximo X e Z que a câmera pode andar, para que o jogador não vá com a câmera em locais onde não existe mais cenário, sendo atributos públicos para que o programador possa alterar esses valores pelo editor do *Unity* e os valores definidos no código são os valores que são atribuídos inicialmente quando o *script* é vinculado, conforme é possível verificar na figura 18.



```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class CameraController : MonoBehaviour {
6
7      private Camera _cam;
8
9
10     private float sensitivityZoomX;
11     private float sensitivityZoomZ;
12
13
14     public float MinX=-29f;
15     public float MaxX= 29f;
16     public float MinZ=-29f;
17     public float MaxZ= 29f;
18
19 }
```

Figura 17. Script Câmera - Parte 1 do código



Figura 18. Script Câmera

Na linha 26 na figura 19, quando apertamos o *play* no *Unity*, a primeira coisa que esse componente *script* faz é ligar a câmera na cena e esse atributo no código, desta forma, podemos manipular seu comportamento via código.

```

23     public void Start()
24     {
25
26         _cam = Camera.main;
27     }

```

Figura 19. Script Câmera - Parte 2 do código

No método *Update*, como vemos na figura 20 na linha 33 e 34 a instrução *_cam.orthographicSize* controla o zoom atual da câmera, quanto maior, mais longe. O código das linhas 33 e 34 através do zoom da câmera determinam a velocidade em que a câmera poderá mover-se, pois quanto mais perto, a câmera deve mover-se mais lentamente conforme fazer o movimento com o mouse. Na linha 37, se o usuário mexer no *scroll* do mouse para cima, deve aumentar o zoom na tela, na linha 41, se o usuário mexer no *scroll* do mouse para baixo, deve diminuir o zoom na tela e a linha 45 é acionada caso o usuário aperte o botão esquerdo (0) ou pressionar o *scroll* do mouse (2), ao pressionar esses botões as linhas 48 e 52 fazem o movimento de *padding* na tela.

```

30     public void Update()
31     {
32
33         sensitivityZoomX = 0.05f * _cam.orthographicSize;
34         sensitivityZoomZ = 0.05f * _cam.orthographicSize;
35
36
37         if (Input.GetAxis("Mouse ScrollWheel") > 0 && _cam.orthographicSize > 5.0f)
38             _cam.orthographicSize -= .5f;
39
40
41         if (Input.GetAxis("Mouse ScrollWheel") < 0 && _cam.orthographicSize < 25.0f)
42             _cam.orthographicSize += .5f;
43
44
45         if (Input.GetMouseButton(0) || Input.GetMouseButton(2))
46         {
47
48             if ((_cam.transform.position - _cam.transform.right * Input.GetAxis("Mouse X")) * sensitivityZoomX.x >
49                 MinX && (_cam.transform.position - _cam.transform.right * Input.GetAxis("Mouse X")) * sensitivityZoomX.x < MaxX)
50                 _cam.transform.position -= _cam.transform.right * Input.GetAxis("Mouse X") * sensitivityZoomX;
51             if ((_cam.transform.localPosition - _cam.transform.up * Input.GetAxis("Mouse Y")) * sensitivityZoomZ.z >
52                 MinZ && (_cam.transform.localPosition - _cam.transform.up * Input.GetAxis("Mouse Y")) * sensitivityZoomZ.z < MaxZ)
53                 _cam.transform.localPosition -= _cam.transform.up * Input.GetAxis("Mouse Y") * sensitivityZoomZ;
54
55         }
56     }

```

Figura 20. Script Câmera - Parte 3 do código

4.5.4. *GameObject CarController*

Nesta parte, conforme conseguimos visualizar na imagem 21, dentro do *GameObject Car* estão presentes os componentes *script CarController*, *Collider* e o *NavMeshAgent*, o qual trabalha com a inteligencia artificial dos *GameObjects* para que possam andar em um terreno, ele só consegue andar por cima de objetos que se movimentam, esse atributo fica visível na imagem 22.

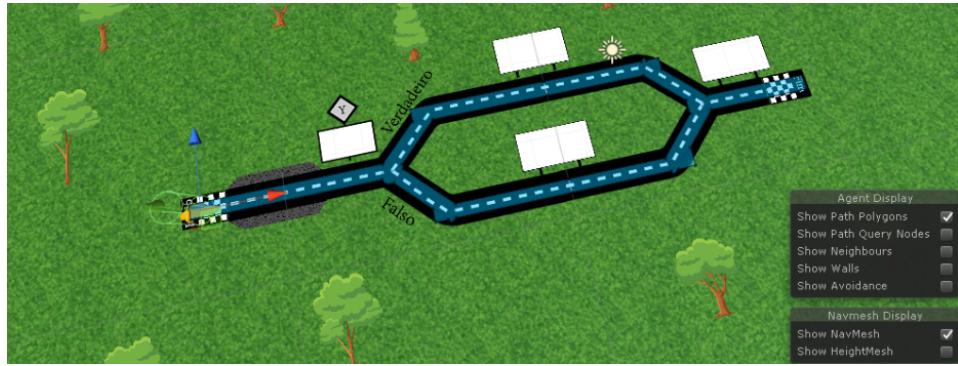


Figura 21. Parte do jogo que mostra o Script CarController

No método *Start*, tem seu início na linha 14 conforme a figura 23, o *GameObject CarController* é localizado dentro do *GameObject Car* e é ligado com o *script* e tenta direcioná-lo para o destino definido. Já o método *public void SetNavMeshAgentSpeed (float speed)* que começa na linha 22, define a velocidade atual do carro, sendo possível modificar um atributo dentro do componente *NavMeshAgent* do carro.

```

12
13
14  public void Start () {
15
16      _navMeshAgent = GetComponent<NavMeshAgent>();
17
18      _navMeshAgent.SetDestination(Destination);
19
20
21  public void SetNavMeshAgentSpeed (float speed)
22  {
23
24      _navMeshAgent.speed = speed;
25
26  }
27
28
29

```

Figura 23. Script CarController - parte 2 do código

4.5.5. ClassVariavel

Conforme a figura 24, a classe é usada para criar a lista de variáveis usadas no *GameObject e no GameController*.

```
1  TCC-2022-EnsinoAlgoritmos-main
2  Assets.Scripts.ClassVariavel
3
4  9     private string nomeVariavel;
5  10    private string valorVariavel;
6  11    private ValueTypes tipoVariavel;
7
8  12
9  13    public ClassVariavel(string nomeVariavel, ValueTypes tipoVariavel, string valorVariavel)
10   {
11      NomeVariavel = nomeVariavel;
12      TipoVariavel = tipoVariavel;
13      ValorVariavel = valorVariavel;
14   }
15
16
17
18
19
20
21    public string NomeVariavel
22   {
23      get
24      {
25          return nomeVariavel;
26      }
27
28      set
29      {
30          nomeVariavel = value;
31      }
32   }
33
34    public string ValorVariavel
35   {
36      get
37      {
38          return valorVariavel;
39      }
40
41      set
42      {
43          valorVariavel = value;
44      }
45   }
46
47    public ValueTypes TipoVariavel
48   {
49      get
50      {
51          return tipoVariavel;
52      }
53
54      set
55      {
56          tipoVariavel = value;
57      }
58   }
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
```

Figura 24. Class Variável

4.5.6. *EnumTypes*

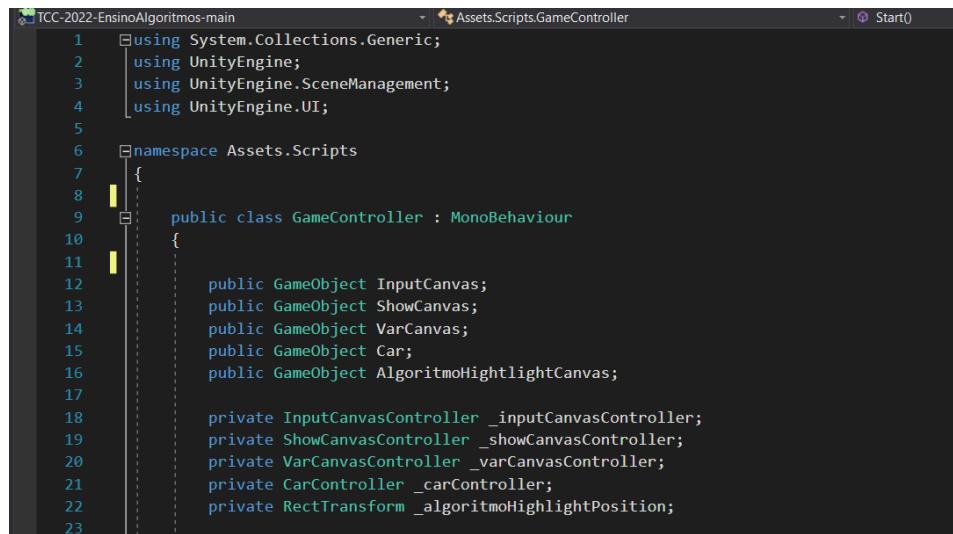
Conforme observamos na figura 25, o *EnumTypes* é um tipo de variável onde definimos algum valores para ele, no caso inteiro, real e texto

```
1  TCC-2022-EnsinoAlgoritmos-main
2  Assets.Scripts.ValueTypes
3
4  1
5  2  namespace Assets.Scripts
6  3  {
7  4      public enum ValueTypes // your custom enumeration
8  5      {
9  6          Inteiro,
10         Real,
11         Texto
12     };
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
```

Figura 25. EnumTypes

4.5.7. GameController

O *GameController* é a classe centralizadora do jogo digital, nela possui todas as interações entre os objetos, ou seja, tudo que tem ação ou reação passa por esta classe. Na figura 26, notamos que todos os *GameObjects* que o *GameController* pode manipular estão públicos para que possa ser feito a vinculação manual dos *GameObjects* pelo *Unity*.



```
1  using System.Collections.Generic;
2  using UnityEngine;
3  using UnityEngine.SceneManagement;
4  using UnityEngine.UI;
5
6  namespace Assets.Scripts
7  {
8
9      public class GameController : MonoBehaviour
10  {
11
12      public GameObject InputCanvas;
13      public GameObject ShowCanvas;
14      public GameObject VarCanvas;
15      public GameObject Car;
16      public GameObject AlgoritmoHighlightCanvas;
17
18      private InputCanvasController _inputCanvasController;
19      private ShowCanvasController _showCanvasController;
20      private VarCanvasController _varCanvasController;
21      private CarController _carController;
22      private RectTransform _algoritmoHighlightPosition;
23  }
```

Figura 26. Game Controller - Parte 1 do código

Já na figura 27, na linha 26 temos a lista que armazena todas as variáveis adicionadas pelo código. Da linha 28 a 36 temos o método *Start*, o qual pega a referência do *InputCanvasController* que está dentro do *GameObject InputCanvas* e através dessas *scripts* o *GameController* pode enviar mensagens para esses objetos. Na linha 35 deste método é pego a referência do *RectTransform* para futuramente manipular as coordenadas do *TextHighlighter*.



```
24
25
26  public List<ClassVariavel> ListaDeVariaveis;
27
28  public void Start()
29  {
30
31      _inputCanvasController = InputCanvas.GetComponentInChildren<InputCanvasController>();
32      _showCanvasController = ShowCanvas.GetComponentInChildren<ShowCanvasController>();
33      _varCanvasController = VarCanvas.GetComponentInChildren<VarCanvasController>();
34      _carController = Car.GetComponent<CarController>();
35      _algoritmoHighlightPosition = AlgoritmoHighlightCanvas.GetComponentInChildren<RectTransform>();
36  }
```

Figura 27. Game Controller - Parte 2 do código

Na linha 48 da figura 28, temos o método que modifica a cor do *TextHighlighter*, já na linha 54 a 62 o método pega o valor de uma variável que existe dentro da *List* *ListaDeVariaveis* e retorna o conteúdo encontrado.

```

47     public void SetHighlightTextColor(float r, float g, float b)
48     {
49         _algoritmoHighlightPosition.GetComponent<Image>().color = new Color(r, g, b);
50     }
51
52
53     public string GetValorDeVariavel(string nomeVariavel)
54     {
55         foreach (var variavel in ListaDeVariaveis)
56         {
57             if (nomeVariavel == variavel.NomeVariavel)
58                 return variavel.ValorVariavel;
59         }
60     }
61
62 }
```

Figura 28. Game Controller - Parte 3 do código

Na figura 29, a partir da linha 66 observamos o método que é chamado toda vez que o usuário faz uma entrada válida no *InputCanvas*, sendo adicionada uma nova variável e impresso no *canvas* de variáveis de texto.

```

64
65     public void AddVariavel(string nomeDeVariavel, ValueTypes tipoDeVariavel, string valorDeVariavel)
66     {
67
68         ListaDeVariaveis.Add(new ClassVariavel(nomeDeVariavel, tipoDeVariavel, valorDeVariavel));
69
70         _varCanvasController.AddVarText(nomeDeVariavel + ":" + valorDeVariavel + "\n");
71     }
72
73 }
```

Figura 29. Game Controller - Parte 4 do código

O método que faz aparecer o *InputCanvas* na tela pode ser observado na figura 30 e o método que faz aparecer o *ShowCanvas* na tela está representado na imagem 31.

```

74
75     public void ActivateInputCanvas(string nomeDeVariavel, ValueTypes tipoDeVariavel)
76     {
77         _inputCanvasController.ActivateInputCanvas(nomeDeVariavel, tipoDeVariavel);
78     }
79 }
```

Figura 30. Game Controller - Parte 5 do código

```

79
80     public void ActivateShowCanvas(string text)
81     {
82         _showCanvasController.ActivateShowCanvas(text);
83     }
84 }
```

Figura 31. Game Controller - Parte 6 do código

Na figura 32 temos o método chamado toda vez que precisamos parar o carro e na imagem 33 observamos o método chamado toda vez que precisamos fazer com que o carro volte a andar.

```
86
87     public void StopCar()
88     {
89         _carController.SetNavMeshAgentSpeed(0);
90     }
91 }
```

Figura 32. Game Controller - Parte 7 do código

```
92
93     public void ResumeCar()
94     {
95         _carController.SetNavMeshAgentSpeed(3f);
96     }
97 }
```

Figura 33. Game Controller - Parte 8 do código

Na figura 34 na linha 99 a 102 temos representado o método chamado toda vez que precisamos recarregar a cena para seu estado inicial e na linha 105 a 108 temos o método chamado toda vez que precisamos recarregar uma cena desejada.

```
97
98
99     public void ReloadScene()
100    {
101        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
102    }
103
104     public void LoadScene(string nome)
105    {
106        SceneManager.LoadScene(nome);
107    }
108
109    }
110
111 }
```

Figura 34. Game Controller - Parte 9 do código

4.5.8. *InputCanvasController*

Na figura 35, a linha 10 faz referência ao *GameObject GameController*, a linha 12 referencia para o script *GameController* dentro do *GameObject GameController*, a linha 14 faz referência para o texto dentro do *GameObject InputCanvas* e na linha 16 faz referência para o *InputText* dentro do *GameObject InputCanvas*.

```

1  using System;
2  using UnityEngine;
3  using UnityEngine.UI;
4
5  namespace Assets.Scripts
6  {
7      public class InputCanvasController : MonoBehaviour
8      {
9
10         public GameObject GameController;
11
12         private GameController _gameControllerScript;
13
14         private Text _text;
15
16         private InputField _inputField;
17

```

Figura 35. InputCanvas Controller - Parte 1 do código

Na imagem 36 são atributos que controlam a variável que será adicionada pelo *InputCanvas*.

```

17
18
19         private string _nomeDeVariavelParaSerAdicionada;
20         private ValueTypes _tipoDeVariavelParaSerAdicionada;
21

```

Figura 36. InputCanvas Controller - Parte 2 do código

Na figura 37 temos o método que procura dentro do *GameObject* que está vinculado esse *script*, que no caso é o *GameObject InputCanvas*, ele busca um componente chamado *InputField* e guarda a referência no atributo *_inputField*.

```

21
22
23     public void Awake ()
24     {
25
26         _inputField = GetComponentInChildren<InputField>();
27         _text = GetComponentInChildren<Text>();
28         _gameControllerScript = GameController.GetComponent<GameController>();
29
30

```

Figura 37. InputCanvas Controller - Parte 3 do código

Na figura 38 o método é chamado quando é feito o *submit* do *InputCanvas*. Na linha 38 Caso o tipo da variável não seja texto, será necessário tentar converter para ver se é um *float* ou *int* válido. Na linha 51 acontece se a entrada é válida e então a linha 57 desativa o *InputCanvas* e a 59 define se o carro pode continuar o movimento.

```

71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1186
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1296
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1396
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1490
1491
1492
1493
1494
1495
1496
1496
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1590
1591
1592
1593
1594
1595
1596
1596
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1690
1691
1692
1693
1694
1695
1696
1696
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1790
1791
1792
1793
1794
1795
1796
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1890
1891
1892
1893
1894
1895
1896
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1990
1991
1992
1993
1994
1995
1996
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2090
2091
2092
2093
2094
2095
2096
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2188
2189
2190
2191
2192
2193
2194
2195
2196
2196
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2288
2289
2290
2291
2292
2293
2294
2295
2296
2296
2297
2298
2299
2299
2300
2301
2302
230
```

```
1  using UnityEngine;
2  using UnityEngine.EventSystems;
3  using UnityEngine.UI;
4
5  namespace Assets.Scripts
6  {
7      public class ShowCanvasController : MonoBehaviour
8      {
9
10         public GameObject GameController;
11
12         private GameObject _playerShowCanvas_Button;
13
14         private Text _text;
15
16         private GameController _gameControllerScript;
17     }
```

Figura 40. ShowCanvasController - Parte 1 do código

Na linha 26 da figura 41, temos o método que é chamado toda vez que precisamos mostrar o *ShowCanvas*. Já na figura 42, na linha 35 temos o método chamado para definir o texto do *ShowCanvas* e na linha 41 temos o método chamado quando o botão confirmar do *ShowCanvas* é apertado, no qual na linha 45 manda o carro parar quando ocorrer uma colisão.

```
33
34
35     private void SetShowCanvasText(string text)
36     {
37         _text.text = text;
38     }
39
40     public void ConfirmShowCanvas()
41     {
42         gameObject.SetActive(false);
43
44         _gameControllerScript.ResumeCar();
45     }
46
47 }
48
49 }
```

Figura 42. ShowCanvasController - Parte 3 do código

4.5.10. *ShowScript*

O *ShowScript* controla o *Collider* que aciona o *ShowCanvasController*. Na figura 43, a linha 9 controla o texto que será exibido no *GameObject Show*, a linha 11 aponta para o *RectTransform* de um *Canvas* que controla o tamanho da fonte do *GameObject Show* e a linha 13 aponta para o *GameObject GameController*. Já na figura ?? os códigos determinam a posição do *TextHighligher*.

```

1  using UnityEngine;
2  using UnityEngine.UI;
3
4  namespace Assets.Scripts
5  {
6      public class ShowScript : MonoBehaviour
7      {
8          public string Text;
9
10         public RectTransform TextRectTransform;
11
12         public GameObject GameController;
13
14     }

```

Figura 43. ShowScript - Parte 1 do código

```

15
16         public float HightlightPositionX;
17         public float HightlightPositionY;
18         public float HightlightPositionSize;
19

```

Figura 44. ShowScript - Parte 2 do código

Na figura 45 os atributos das linhas 21 e 22 controlam o tamanho da fonte com base no *zoom* e no posicionamento da câmera, a linha 25 é responsável por apontar para o *collider* dentro do *GameObject Show*, o qual o carrinho irá colidir. Já a linha 27, aponta para o componente *script* dentro do *GameObject GameController* e a linha 29 aponta para o componente *Text* dentro do *Canvas* do *GameObject Show*.

```

20
21         private Vector3 _screenPos;
22         private float _larguraStart, _alturaStart, _cameraScaleStart;
23
24
25         private Collider _col;
26
27         private GameController _gameControllerScript;
28
29         private Text _textComponent;

```

Figura 45. ShowScript - - Parte 3 do código

O método *start* está representado na figura 46, o qual pega as configurações iniciais do *GameObject Text* dentro do *Canvas* do *GameObject Show*, dessa forma na linha 34 o valor referenciado é o *Width*, na linha 35 *Height* e na linha 36 temos o zoom da câmera referenciado.

A linha 40 é a referência do *Collider* e passo para o atributo no script e na linha 42 é referenciado o componente *script* dentro do *GameObject GameController* e por fim nas linhas 45 e 46 ocorre a definição do componente *Text*.

```

30
31     public void Start()
32     {
33
34         _larguraStart = TextRectTransform.sizeDelta.x;
35         _alturaStart = TextRectTransform.sizeDelta.y;
36         _cameraScaleStart = Camera.main.orthographicSize;
37
38
39
40         _col = GetComponent<Collider>();
41
42         _gameControllerScript = GameController.GetComponent<GameController>();
43
44
45         _textComponent = GetComponentInChildren<Text>();
46         _textComponent.text = Text;
47     }

```

Figura 46. ShowScript

Na figura 47 é exibido o método *LateUpdate*, nas linhas 52 e 53 ocorre o ajuste na posição dos *canvas* de texto na tela e a linha 56 ajusta a escala dos campos *canvas* de texto. Já na figura 48 está representado o método *OnTriggerEnter*, nas linhas 64,65 e 66 ocorre o uso do *GameController* para mudar a posição do *text highlighter*, a linha 68 possui o comando de parada ao carro colidir e a linha 70 desativa o *Collider*.

```

49
50
51     public void LateUpdate()
52     {
53
54         screenPos = Camera.main.WorldToScreenPoint(transform.position);
55         TextRectTransform.position = new Vector3(_screenPos.x, _screenPos.y, 0f);
56
57         TextRectTransform.sizeDelta = new Vector2(_larguraStart *
58             (2 * _cameraScaleStart - Camera.main.orthographicSize) / _cameraScaleStart, _alturaStart *
59             (2 * _cameraScaleStart - Camera.main.orthographicSize) / _cameraScaleStart);
59

```

Figura 47. ShowScript

```

61
62     public void OnTriggerEnter(Collider other)
63     {
64
65         _gameControllerScript.SetHighlightTextColor(255f, 255f, 0f);
66         _gameControllerScript.MoveHighlightToPosition(HightlightPositionX, HightlightPositionY,
67         HightlightPositionSize);
68         _gameControllerScript.ActivateShowCanvas(Text);
69
70
71         _gameControllerScript.StopCar();
72
73         _col.enabled = false;
74     }

```

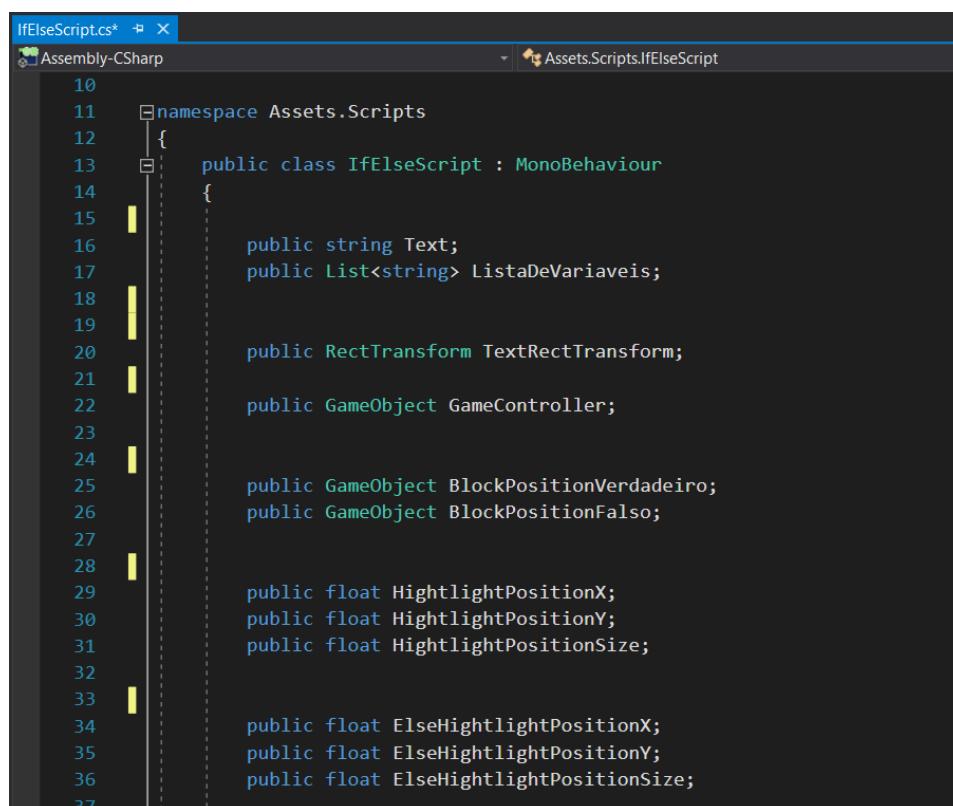
Figura 48. ShowScript

4.5.11. VarCanvasController

Esse *script* controla o painel de variáveis, seu código está representado na 49. Na linha 12 começa o método *start*, o qual na linha 15 encontra-se a primeira referência de um componente *Text* em seus filhos.

4.5.12. IfElseScript

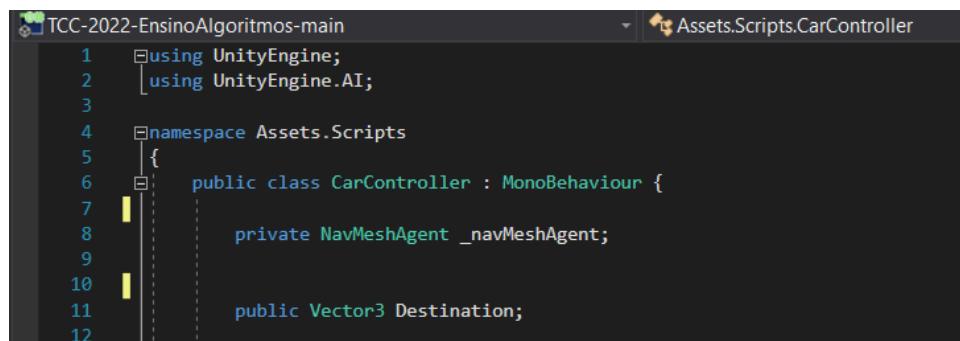
Conforme a figura 50, observamos que na linha 17 o texto dentro do objeto *If/Else* será a condição, o atributo da linha 20 aponta para o *canvas* que exibe o texto do *GameObject If/Else*, usado para ajustar o texto conforme o zoom da tela acontece, na linha 22 o atributo aponta para o *GameObject GameController*.Já nas linhas 25 e 26 são os *Gameobjects* que apontam para o local onde o fluxo da rodovia será cortado. As linhas 29,30,31,34,35 e 36 determinam a posição do *TextHiglither*



```
10
11  namespace Assets.Scripts
12  {
13      public class IfElseScript : MonoBehaviour
14      {
15
16          public string Text;
17          public List<string> ListaDeVariaveis;
18
19
20          public RectTransform TextRectTransform;
21
22          public GameObject GameController;
23
24
25          public GameObject BlockPositionVerdadeiro;
26          public GameObject BlockPositionFalso;
27
28
29          public float HightlightPositionX;
30          public float HightlightPositionY;
31          public float HightlightPositionSize;
32
33
34          public float ElseHightlightPositionX;
35          public float ElseHightlightPositionY;
36          public float ElseHightlightPositionSize;
37
```

Figura 50. IfElseScript - Parte 1 do código

Na figura 51, na linha 39 o atributo guardar a referência do componente *script GameController*, nas linhas 43,44 e 45 temos os atributos que verificam o posicionamento da câmera para ajustar o tamanho do texto exibido dentro do *GameObject If/Else*.



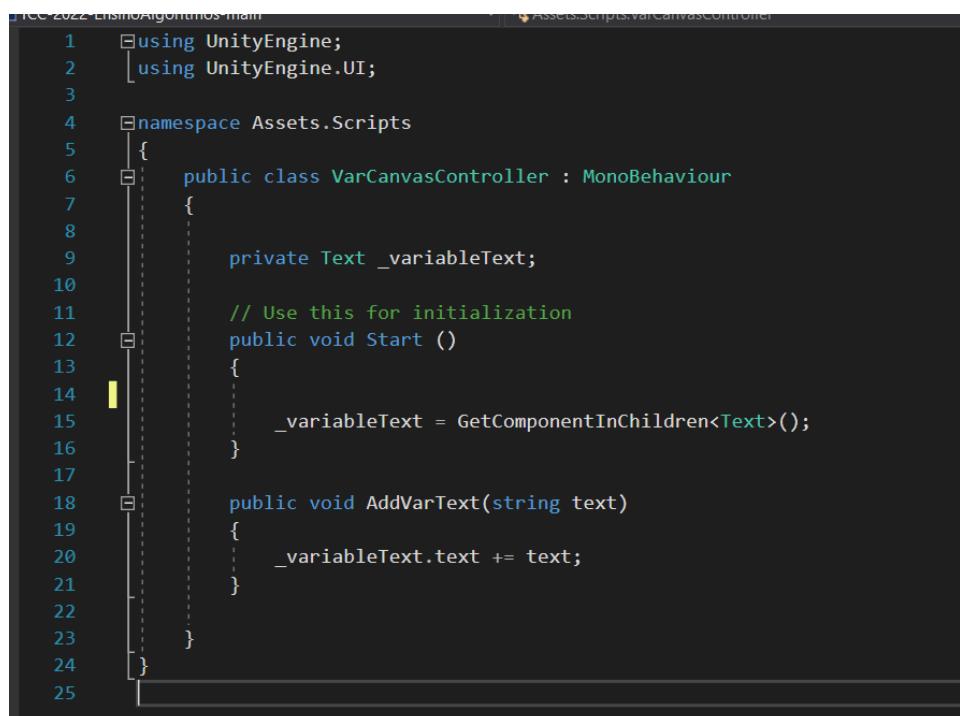
```
1  using UnityEngine;
2  using UnityEngine.AI;
3
4  namespace Assets.Scripts
5  {
6      public class CarController : MonoBehaviour {
7
8          private NavMeshAgent _navMeshAgent;
9
10         public Vector3 Destination;
11
12     }
```

Figura 22. Script CarController - Parte 1 do código



```
19     public void Awake ()
20     {
21         _playerShowCanvas_Button = GetComponentInChildren<Button>().gameObject;
22         _text = GetComponentInChildren<Text>();
23         _gameControllerScript = GameController.GetComponent<GameController>();
24     }
25
26     public void ActivateShowCanvas(string text)
27     {
28         gameObject.SetActive(true);
29         SetShowCanvasText(text);
30         EventSystem.current.SetSelectedGameObject(_playerShowCanvas_Button); // Request focus
31     }
```

Figura 41. ShowCanvasController - Parte 2 do código



```
1  using UnityEngine;
2  using UnityEngine.UI;
3
4  namespace Assets.Scripts
5  {
6      public class VarCanvasController : MonoBehaviour
7      {
8
9          private Text _variableText;
10
11         // Use this for initialization
12         public void Start ()
13         {
14
15             _variableText = GetComponentInChildren<Text>();
16         }
17
18         public void AddVarText(string text)
19         {
20             _variableText.text += text;
21         }
22     }
23
24
25 }
```

Figura 49. VarCanvasController

```

39     private GameController _gameControllerScript;
40
41     private Vector3 _screenPos;
42     private float _larguraStart, _alturaStart, _cameraScaleStart;
43     private Text _textComponent;
44
45
46     private Collider _col;
47
48

```

Figura 51. IfElseScript - Parte 2 do código

A representação do método *Start* ocorre na figura 52, o qual nas linhas 57 e 58 definem no componente *Text* o seu conteúdo, na linha 61 pega a referência do script *GameController* e na linha 64 pega a referencia do *collider*.

```

50     public void Start()
51     {
52         _larguraStart = TextRectTransform.sizeDelta.x;
53         _alturaStart = TextRectTransform.sizeDelta.y;
54         _cameraScaleStart = Camera.main.orthographicSize;
55
56
57         _textComponent = GetComponentInChildren<Text>();
58         _textComponent.text = Text;
59
60
61         _gameControllerScript = GameController.GetComponent<GameController>();
62
63
64         _col = GetComponent<Collider>();
65     }

```

Figura 52. IfElseScript - Parte 3 do código

O método representando na imagem 53, ajusta a posição dos *canvas* de texto na tela na linha 70 e na linha 74 ajusta a escala dos campos *canvas* de texto.

```

67     public void LateUpdate()
68     {
69
70         _screenPos = Camera.main.WorldToScreenPoint(transform.position);
71         TextRectTransform.position = new Vector3(_screenPos.x, _screenPos.y, 0f);
72
73
74         TextRectTransform.sizeDelta = new Vector2(_larguraStart * (2 * _cameraScaleStart -
75             Camera.main.orthographicSize) / _cameraScaleStart, _alturaStart * (2 * _cameraScaleStart -
76             Camera.main.orthographicSize) / _cameraScaleStart);
77
78

```

Figura 53. IfElseScript - Parte 4 do código

Na linha 78 da figura 54 temos o método que irá comunicar-se com o interpretador, o qual une o código da inicialização com um retorno da expressão sendo avaliada e envia para o *Carpilator* interpretar, obtendo o resultado da expressão (verdadeiro ou falso). Já na linha 85, o método é chamado toda vez que passa em um *IF/Else*. Esse método é responsável por gerar um código-fonte de inicialização das variáveis a partir do nome e seus valores.

```

78     public bool Evaluate()
79     {
80         var code = string.Join(Environment.NewLine, InitializeVariables(), $"return {Text};");
81         return new Carpilator(code, new CCsharp()).Run<bool>();
82     }
83
84     private string InitializeVariables()
85     {
86         var symbolTable = ListaDeVariaveis.Select(x => new { Variavel = x, Valor =
87             _gameControllerScript.GetValorDeVariavel(x) });
88         var varInitializations = symbolTable.Select(x => $"int {x.Variavel} = {x.Valor};");
89         return string.Join(Environment.NewLine, varInitializations);
90     }
91
92 }

```

Figura 54. IfElseScript - Parte 5 do código

Por fim, na figura 55 temos o método que é chamado, toda vez que um *collider* invade o *collider* desse objeto, se o resultado for verdadeiro lê-se o código *If*, caso ao contrário vai para o *else* e a linha 108 é responsável por desativar o *collider*.

```

92     public void OnTriggerEnter(Collider other)
93     {
94
95         if ( Evaluate() )
96         {
97             _gameControllerScript.SetHighlightTextColor(255f, 255f, 0f);
98             _gameControllerScript.MoveHighlightToPosition(HightlightPositionX, HightlightPositionY,
99                 HightlightPositionSize);
100            BlockPositionFalso.GetComponent<NavMeshObstacle>().enabled = true;
101        }
102        else
103        {
104            _gameControllerScript.SetHighlightTextColor(0f, 0f, 255f); // Cor azul
105            _gameControllerScript.MoveHighlightToPosition(ElseHightlightPositionX, ElseHightlightPositionY,
106                ElseHightlightPositionSize);
107            BlockPositionVerdadeiro.GetComponent<NavMeshObstacle>().enabled = true;
108        }
109        _col.enabled = false;
110    }
111 }

```

Figura 55. IfElseScript - Parte 6 do código

5. Conclusão

O objetivo deste artigo foi alcançado com êxito, entretanto a etapa três da metodologia que abordava o teste de qualidade do jogo desenvolvido nos critérios de usabilidade e funcionalidade e a confecção de um relatório apontando as experiências dos alunos envolvidos não foi realizado devido a falta de tempo hábil para a conclusão deste trabalho.

Com o intuito de identificar as principais dificuldades da turma do semestre 2022/1 do IFSC Câmpus Lages da disciplina de programação no curso de Ciência da Computação e também averiguar se os discentes considerariam uma outra forma de ensino-aprendizagem diferente da que se tem hoje, foi realizada a aplicação de um questionário.

O objetivo de desenvolver um interpretador independente foi alcançado, tendo sido seu desenvolvimento e sua integração com o jogo *Unity* finalizados durante o trabalho, possibilitando assim mais flexibilidade para a criação de outros níveis com estruturas de programação diferentes, estruturas de repetição ou funções por exemplo, e até mesmo a criação de níveis dinâmicos, onde o usuário poderia digitar o código que seria interpretado, gerando assim uma

estrada para aquele código, o que traria uma dinâmica completamente nova e muito útil para o aprendizado.

Em possíveis trabalhos futuros, pode ser feito a melhoria do jogo digital, implementando melhorias e ampliações como novas estruturas de código: funções, laços de repetição e entre outros, e aplicá-lo a testes para que seja realizado mais melhorias se for necessário.

Referências

- Andrade, R. et al. (2016). Uma proposta de oficina de desenvolvimento de jogos digitais para ensino de programação. *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, 5(1):1127.
- da Silva, F. R. et al. (2016). Desenvolvimento de jogos na plataforma unity. *RE3C-Revista Eletrônica Científica de Ciência da Computação*, 11(1).
- De Andrade, M. S. (2019). *Adobe Photoshop CC*. Editora Senac São Paulo.
- Grossmann, L. O. (2019). Ti precisa de 420 mil novos profissionais até 2024. Disponível em: <https://brasscom.org.br/ti-precisa-de-420-mil-novos-profissionais-ate-2024/>. Acesso em: 14 abr 2022.
- Monclar, R. S., Silva, M. A., e Xexéo, G. (2018). Jogos com propósito para o ensino de programação. *Anais do XVII Simpósio Brasileiro de Jogos e Entretenimento Digital-SBGames*, pages 1132–1140.
- Overflow, S. (2021). Developer survey 2021. Disponível em: <https://insights.stackoverflow.com/survey/2021>. Acesso em: 26 jun. de 2022.
- Pantaleão, E. et al. (2017). Uma abordagem baseada no ambiente robocode para ensino de programação no ensino médio. *Revista Brasileira de Informática na Educação*, 25(03):95.
- Portugal, R. L. Q. e do Prado Leite, J. C. S. (2015). A transparência do github para uso de artefatos como fontes de informação na engenharia de requisitos. *WTRANS*, 15:1–6.
- Silva, R. et al. (2018a). Panorama da utilização de jogos digitais no ensino de programação no nível superior na Última década: Uma revisão sistemática da literatura. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)*, 29(1):535.
- Silva, R. et al. (2018b). Panorama da utilização de jogos digitais no ensino de programação no nível superior na Última década: Uma revisão sistemática da literatura. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)*, 29(1):535.
- Trello (2022). O trello ajuda os times a agilizarem o trabalho. Disponível em: <https://trello.com/home/>. Acesso em: 20 jun. de 2022.
- Wagner, B. et al. (2022). Um tour pela linguagem c#. Disponível em: <https://docs.microsoft.com/pt-br/dotnet/csharp/tour-of-csharp/>. Acesso em: 20 jun. de 2022.
- Zanetti, H. e Oliveira, C. (2015). Práticas de ensino de programação de computadores com robótica pedagógica e aplicação de pensamento computacional. *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, 4(1):1236.