



UNIVERSIDADE NOVE DE JULHO - UNINOVE
PROJETO EM SISTEMAS INTERATIVOS

923110351 - DIEGO DO NASCIMENTO
922201364 - BRUNO BATISTA FEITOSA
421202804 - JHON FABRICIO TICONA MAMANI
923103670 - MATHEUS SOUZA GACHE
922201327 - NICOLAS ALMEIDA PRATA DA SILVA
923107268 - PEDRO GUILHERME DE CASTRO GONÇALVES

BYTE WEAVER ESTRUTURAS AVANÇADAS DE AUTÔMATOS
FINITOS DETERMINÍSTICOS

São Paulo
2024

923110351 - DIEGO DO NASCIMENTO
922201364 - BRUNO BATISTA FEITOSA
421202804 - JHON FABRICIO TICONA MAMANI
923103670 - MATHEUS SOUZA GACHE
922201327 - NICOLAS ALMEIDA PRATA DA SILVA
923107268 - PEDRO GUILHERME DE CASTRO GONÇALVES

**BYTE WEAVER ESTRUTURAS AVANÇADAS DE AUTÔMATOS
FINITOS DETERMINÍSTICOS**

Projeto apresentado a Universidade Nove de Julho - UNINOVE, como parte dos requisitos obrigatórios para obtenção do título de Bacharel em Ciência da Computação.

Prof. Orientador: Edson Melo de Souza, Dr.

São Paulo
2024

RESUMO

Contexto: Crianças e Jovens que tem curiosidades em programação e querem aprender,

começando por AFD **Objetivo:** Ensinar o sobre AFD através de um jogo independente que é umas das bases de programação

Método: Criar um jogo em linguagem Csharp e colocando em Unity sobre AFD e colocando tudo em prática, através de um carrinho

Resultados: O jogo funcionou corretamente obedecendo os comandos certamente

Conclusão: As crianças e os jovens conseguiram aprender sobre AFD e também concluímos que um jogo ilustrativo ajuda as pessoas a aprender.

Palavras-chave: Palavra 1, Palavra 2, Palavra 3, Palavra 4, Palavra 5, Palavra 6.

ABSTRACT

Contextualization: Children and Young People who are curious about programming and want to learn, starting with AFD **Objective:** Teach about AFD through an independent game that is one of the bases of programming **Method:** Creating a game in Csharp language and putting it in Unity over AFD and putting everything into practice **Results:** The game worked correctly, obeying the commands. **Conclusion:**Children and young people were able to learn about AFD and we also concluded that an illustrative game helps people learn.

Keywords: Keyword 1, Keyword 2, Keyword 3, Keyword 4, Keyword 5, Keyword 6.

SUMÁRIO

Lista de Ilustrações	6
Lista de Tabelas	7
Lista de Quadros	8
Lista de Abreviaturas	9
1 Introdução	10
1.1 Autômatos Finitos Determinísticos	10
1.1.1 Explicação de AFD(Automatos finitos Deterministicos)	10
1.1.2 Linguagem e Regras	10
1.2 Montagem de Tabela	11
1.3 Diagrama de Estado	11
1.4 Explicação do Jogo Educativo do Carrinho	11
2 Fundamentação Teórica	13
2.1 Links das Referencias e Fontes de Pesquisa	13
2.2 Referencias e Fontes de Pesquisa	13
3 Metodologia	14
3.1 Objetivo	14
3.2 Explicação de como vai funcionar	14
3.3 Programação	14
3.4 Site do Jogo	16
4 Análise dos Resultados	17
4.1 Código em C sharp	17
4.2 Colocando na Unity	17
4.3 Jogo em Funcionamento	18
5 Conclusões	20
Referências Bibliográficas	21
Apêndices	22
A : Automatos Finitos Determiniscos	22
Anexos	23
A : Título	23

LISTA DE ILUSTRAÇÕES

1.1	11
1.2	12
4.1	17
4.2	17
4.3	18
4.4	18
4.5	19
4.6	19

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS

MM	Morfologia matemática
CC	Componente conexo
EE	Elemento estruturante

1 INTRODUÇÃO

Resumo do capítulo

Este software promete abrir portas para uma ampla gama de aplicações, desde a análise de linguagens formais até a criação de jogos cativantes e educacionais. Ao conceber essa ferramenta, visamos não apenas aprofundar nossa compreensão teórica sobre expressões regulares e autômatos finitos determinísticos, mas também explorar os limites entre a teoria da computação e a prática do entretenimento digital.

1.1 AUTÔMATOS FINITOS DETERMINÍSTICOS

1.1.1 Explicação de AFD(Automatos finitos Deterministicos)

Um autômato finito determinístico (AFD) é um modelo matemático utilizado em ciência da computação e teoria da computação para representar sistemas que seguem um conjunto específico de regras. Resumidamente, um AFD consiste em:

Estados: Representam as condições do sistema em um determinado momento. Símbolos de Entrada: São os elementos que o sistema pode receber como entrada. Transições: Indicam como o sistema muda de um estado para outro quando recebe determinados símbolos de entrada. Estado Inicial: É o estado onde o sistema começa sua execução. Estados de Aceitação: São os estados que indicam que o sistema alcançou um estado desejado ou concluiu uma tarefa específica. Um AFD determina uma sequência específica de estados e transições, tornando previsível o comportamento do sistema em resposta a diferentes entradas. Ele é "determinístico" porque, para cada estado e símbolo de entrada, há apenas uma transição possível, o que elimina ambiguidades na execução do sistema.

1.1.2 Linguagem e Regras

No coração deste sistema estão as regras claras e precisas dos autômatos finitos determinísticos, ou AFD. Imagine um carrinho percorrendo uma pista, onde cada ação é representada por símbolos de entrada específicos: 0, 1 e 2. Esses símbolos guiam o comportamento do carrinho ao longo do percurso.

Os estados internos do nosso sistema são fundamentais para compreender o fluxo do jogo. Temos os estados L0, L1, L2, L3, L4, L5, L6 e LF, cada um com sua função no processo de movimentação do carrinho.

Além disso, destacamos os estados de aceitação, L2 e L5, que indicam situações onde o carrinho atinge objetivos ou realiza ações desejadas. Por outro lado, temos os estados de não aceitação, como L0, L1, L3, L4, L6 e LF, que representam condições onde o carrinho não alcança seus objetivos ou comete erros durante a movimentação.

O estado inicial, L0, marca o ponto de partida do jogo, enquanto os estados de não aceitação e os estados finais, como LF, delimitam os limites e as condições de término da experiência de movimentação do carrinho.

1.2 MONTAGEM DE TABELA

A seguir a tabela mostra o processo de transição de um estado para outro utilizando a linguagem indicada nas regras.

Figura 1.1

F	0	1	2
L0	L3	L1	L2
L1	L0	L4	L2
L2	L3	L1	L5
L3	L6	L0	L2
L4	L1	LF	L5
L5	L6	L4	LF
L6	LF	L3	L5
Lf	LF	LF	LF

Fonte: Matheus, Diego, Nicolas, Pedro.

1.3 DIAGRAMA DE ESTADO

Um diagrama de estado (Figura 1.2), também conhecido como diagrama de transição de estado, é uma representação visual de um sistema baseado em autômatos finitos, como o autômato finito determinístico (AFD). Ele mostra os estados possíveis em que o sistema pode estar e as transições entre esses estados em resposta a diferentes eventos ou entradas.

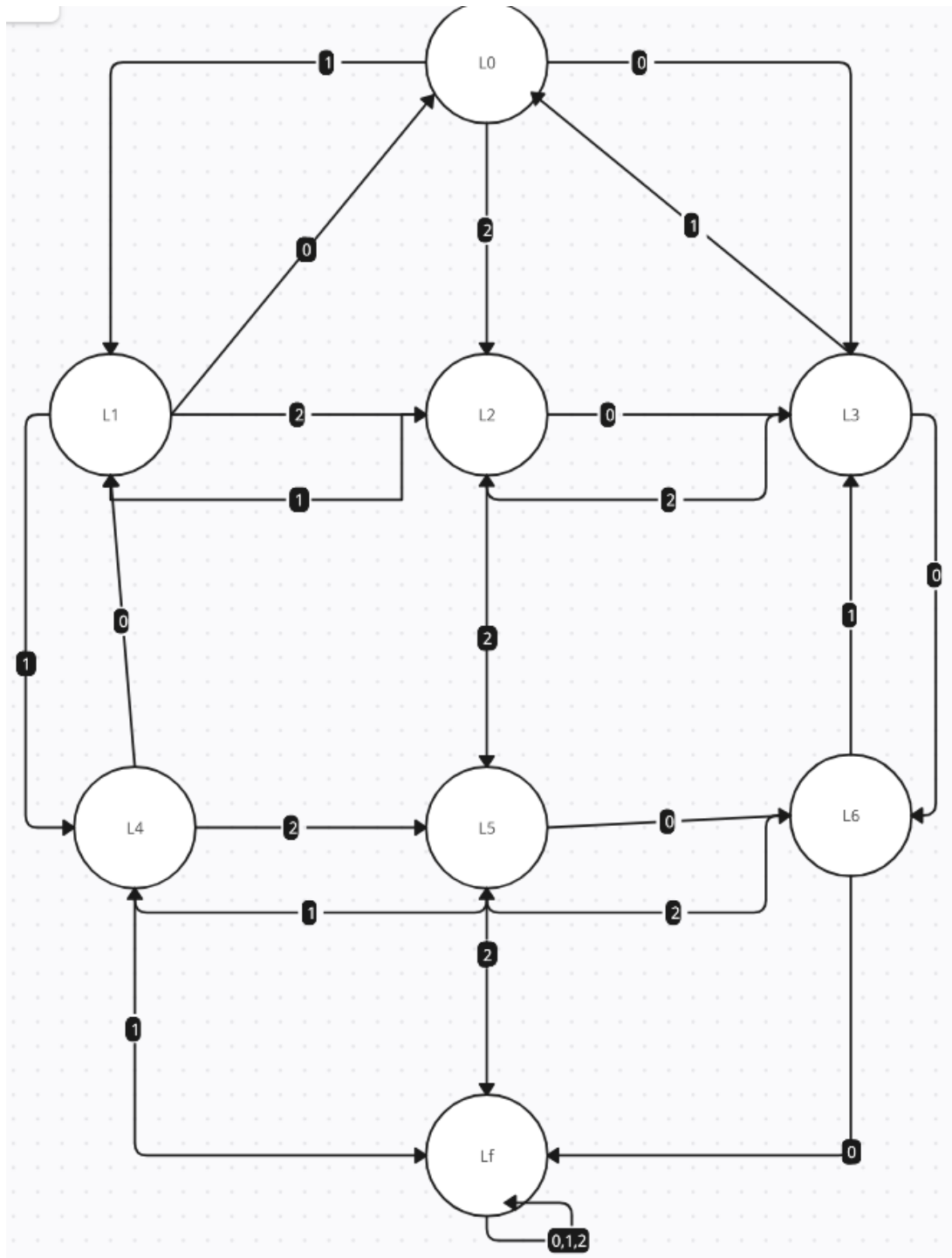
Estados: L0, L1, L2, L3, L4, L5, L6, LF Símbolos de Entrada: 0, 1, 2 Estados de Aceitação: L2, L5 Estado Inicial: L0 Estados de Não Aceitação: L0, L1, L3, L4, L6, LF

1.4 EXPLICAÇÃO DO JOGO EDUCATIVO DO CARRINHO

Com base na explicação de AFD, o grupo decidiu criar um jogo em Csharp utilizando as regras de linguagem criadas com a Tabela de transição de estados e o Diagrama de estado, o jogo sera feito na plataforma Unity sobre os conhecimentos do nosso colega Matheus.

O jogo consiste em dar a direção do nosso carro de acordo com as nossas regras,

Figura 1.2



Fonte: Matheus, Diego, Nicolas, Pedro.

sendo que para vencer voce precisa entrar no estado de aceite que são "L2 e L5" assim você vence o jogo porem se você for no estado LF que é um lugar sem saída ou você parar em qualquer estado sem ser de aceite como mostrado acima, assim você perde o jogo.

2 FUNDAMENTAÇÃO TEÓRICA

Resumo do capítulo

Para desenvolver este trabalho, utilizamos referências e fontes provenientes de pesquisa acadêmica em Teoria dos Autômatos e Ciência da Computação. Além disso, consultamos trabalhos de pesquisa relevantes disponíveis em repositórios digitais de universidades e instituições de pesquisa. Essas fontes foram fundamentais para compreender os conceitos teóricos dos Autômatos Finitos Determinísticos (AFD), assim como para explorar métodos de ensino e ferramentas educacionais utilizadas na área.

2.1 LINKS DAS REFERENCIAS E FONTES DE PESQUISA

https://pt.wikipedia.org/wiki/Aut%C3%B4mato_finito_determin%C3%ADstico#:~:text=Na%20Teoria%20dos%20aut%C3%B4matos%2C%20um,para%20cada%20cadeia%20de%20entrada

https://sol.sbc.org.br/index.php/sbgames_estendido/article/view/19680

2.2 REFERENCIAS E FONTES DE PESQUISA

Aqui estão nossas fontes de pesquisas sobre o nosso projeto

1. Na Teoria dos autômatos, um sub-tópico da Ciência da computação teórica, um autômato finito determinístico — também chamado máquina de estados finita determinística (AFD) — é uma Máquina de estados finita que aceita ou rejeita cadeias de símbolos gerando um único ramo de computação para cada cadeia de entrada.[1] "Determinística" refere-se à unicidade do processamento. O primeiro conceito similar ao de autômatos finitos foi apresentado por McCulloch e Pitts em 1943.[2][3] Modelo esse que foi produzido na busca por estruturas mais simples para a reprodução de máquinas de estado finitas.
2. O uso de jogos no ensino pode favorecer o aprendizado, uma vez que o caráter lúdico tende a estimular e melhorar a concentração e percepção dos alunos. Na área da Ciência da Computação o uso de jogos é frequente em diferentes tópicos, no entanto, em Linguagens Formais e Autômatos isso não é comum, dada a natureza teórica e abstrata de seus conceitos. Este trabalho apresenta o projeto e desenvolvimento de um jogo voltado ao ensino de conceitos de Autômato Finito Determinístico, ao mesmo tempo em que entretém o usuário. O jogo segue a ideia de um escape room no qual o jogador tem que achar os recursos necessários para escapar das salas. Como forma de avaliar a qualidade do jogo foi utilizado o Instrumento de Avaliação da Qualidade de Jogos Educativos (IAQJEd) com 16 participantes. O resultado obtido foi satisfatório, indicando que o jogo tem boa qualidade para finalidade de ensino.

3 METODOLOGIA

Resumo do capítulo

O capítulo aborda a teoria e aplicação dos autômatos finitos determinísticos (AFD) no contexto do projeto, com destaque para o desenvolvimento do jogo educacional utilizando a Unity e Csharp. O AFD e Componentes no Jogo, Explicação dos estados, símbolos de entrada, transições, estado inicial, estados de aceitação e não aceitação, integrados ao jogo educacional desenvolvido em Csharp na Unity. Funcionalidades do Jogo: Descrição das regras e mecânicas do jogo, que permitem aos jogadores interagir com autômatos finitos determinísticos de forma educativa. Visualização no Unity: Apresentação visual do jogo e do AFD por meio da Unity, demonstrando como os conceitos teóricos são implementados e visualizados em um ambiente de entretenimento para os jovens e crianças.

3.1 OBJETIVO

Nosso objetivo é que, por meio de um jogo divertido e educativo, as crianças e adolescentes possam aprender sobre autômatos finitos determinísticos (AFD). Desta forma pretendemos não só introduzir os principais conceitos de AFD, bem como proporcionar uma experiência prática de programação utilizando a linguagem C sharp. Acreditamos que a combinação de aprendizado divertido e prático é essencial para despertar o interesse e o entendimento em áreas complexas como a programação.

3.2 EXPLICAÇÃO DE COMO VAI FUNCIONAR

Nos decidimos programar na linguagem Csharp, usando a Unity para o nosso jogo, procuramos o modelo do carro já que vai ser em 3D. montamos as estradas e o mapa, e então decidimos fazer a programação, testando cada metodo diferente para o carro andar e chegar na rua correta de acordo com o diagrama de estado.

1. Montamos as estradas e o mapa, e então decidimos fazer a programação, testando cada metodo diferente para o carro andar e chegar na rua correta de acordo com o diagrama de estado.
2. Usamos o nosso diagrama de estado para saber onde o carro devera ir e aonde é a rua que o carro vai ganhar que é o estado de aceite que são "L2 e L5" e para perder é quando o carro chegar no estado "LF".

3.3 PROGRAMAÇÃO

Aqui mostraremos a programação e de cada detalhe que foi feito.

1. Aqui está o código que fizemos para o jogo: using UnityEngine, se quiser acessar o código ele tá disponível no nosso GitHub:

```
public class MoveToTarget : MonoBehaviour
{
    public Transform L0; public Transform L1; public Transform L2; public Transform L3; public Transform L4; public Transform L5; public Transform L6; public Transform Lf; private Transform currentTarget;
```

```
void Start()
{
    currentTarget = L0;
}

private void Update()
```

```
{
    // Verifica se um dos botões foi pressionado
    switch (Input.inputString)
    {
        case "1":
```

```
            if (currentTarget == L0) currentTarget = L1; break;
            if (currentTarget == L1) currentTarget = L4; break;
            if (currentTarget == L2) currentTarget = L1; break;
            if (currentTarget == L3) currentTarget = L0; break;
            if (currentTarget == L4) currentTarget = Lf; break;
            if (currentTarget == L5) currentTarget = L4; break;
            if (currentTarget == L6) currentTarget = L3; break;
            if (currentTarget == Lf) currentTarget = Lf; break;
            break;
```

```
        case "2":
            if (currentTarget == L0) currentTarget = L2; break;
            if (currentTarget == L1) currentTarget = L2; break;
            if (currentTarget == L2) currentTarget = L5; break;
            if (currentTarget == L3) currentTarget = L2; break;
            if (currentTarget == L4) currentTarget = L5; break;
            if (currentTarget == L5) currentTarget = Lf; break;
            if (currentTarget == L6) currentTarget = L5; break;
            if (currentTarget == Lf) currentTarget = Lf; break;
            break;
        case "0":
            if (currentTarget == L0) currentTarget = L3; break;
            if (currentTarget == L1) currentTarget = L0; break;
            if (currentTarget == L2) currentTarget = L3; break;
            if (currentTarget == L3) currentTarget = L6; break;
            if (currentTarget == L4) currentTarget = L1; break;
            if (currentTarget == L5) currentTarget = L6; break;
            if (currentTarget == L6) currentTarget = Lf; break;
            if (currentTarget == Lf) currentTarget = Lf; break;
            break;
```

```
    // Move o objeto em direção ao alvo atual
    float speed = 10f; // Velocidade de movimento
    transform.position = Vector3.MoveTowards(transform.position, currentTarget.position, speed * Time.deltaTime);
}
```

2. **Variáveis e Inicialização:** As variáveis L0, L1, L2, ..., Lf são do tipo Transform e representam os alvos para os quais o objeto se moverá. A variável currentTarget armazena o alvo atual, inicializada como L0 no método Start().
3. **Método Update():** O método Update() está vazio, o que significa que não há lógica sendo executada nele. Se você planeja adicionar alguma lógica posteriormente, pode fazê-lo aqui.

4. **Método FixedUpdate():** Este método é chamado em intervalos fixos (geralmente a cada quadro) e é onde a lógica principal acontece. O switch verifica qual botão foi pressionado com base na entrada do usuário (Input.inputString). Dependendo do botão pressionado (casos “1”, “2” ou “0”), o código atualiza o currentTarget para o próximo alvo correspondente. O objeto é movido em direção ao alvo atual usando Vector3.MoveTowards() com uma velocidade definida.
5. **Velocidade de Movimento:** A variável speed controla a velocidade de movimento do objeto em direção ao alvo. Atualmente, está definida como 10f.
6. **Quebra de Linha no switch:** Note que adicionei break; após cada atualização do currentTarget dentro dos casos. Isso garante que apenas um caso seja executado por vez.

3.4 SITE DO JOGO

O jogo está disponível no site chamado itch.io, um site feito para jogos independentes, aqui está o link do site do jogo <<https://mtssg.itch.io/projetouni9>>, o site possui a senha, a senha dele é uni94637

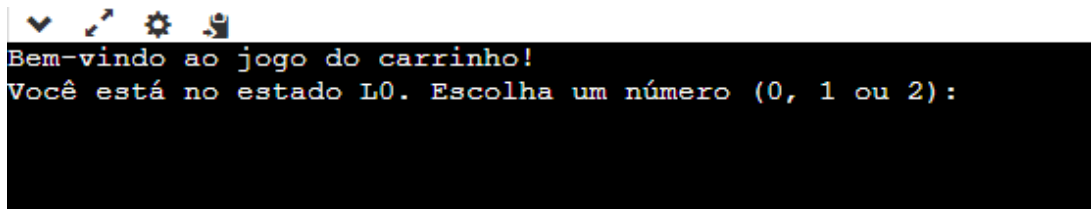
4 ANÁLISE DOS RESULTADOS

Aqui mostraremos passo a passo de nossos resultados

4.1 CÓDIGO EM C SHARP

De acordo com o que foi mostrado no capítulo anterior fizemos o código em C sharp colocamos de acordo com a nossa tabela e o diagrama de estados, com tudo já montado, o código esta funcionando e agiu de maneira correta e funcionou de acordo com as nossas regras que definimos a maquina, como mostrado nas imagens (Figura 4.1), (Figura 4.2), (Figura 4.3).

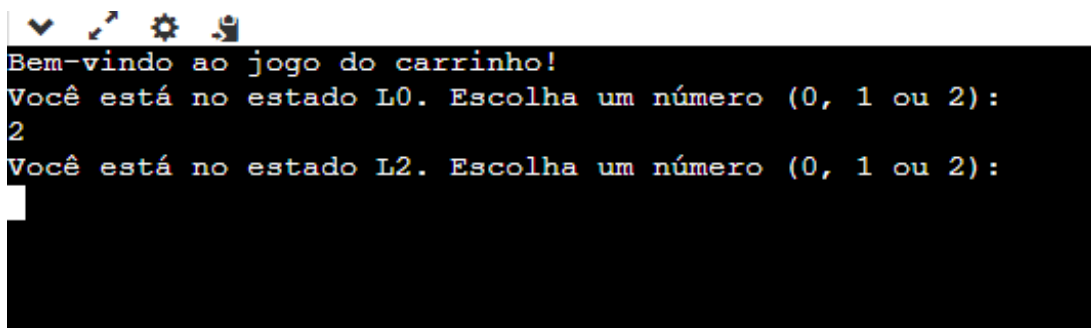
Figura 4.1



```
Bem-vindo ao jogo do carrinho!  
Você está no estado L0. Escolha um número (0, 1 ou 2):
```

Fonte: Matheus, Diego, Nicolas, Pedro.

Figura 4.2



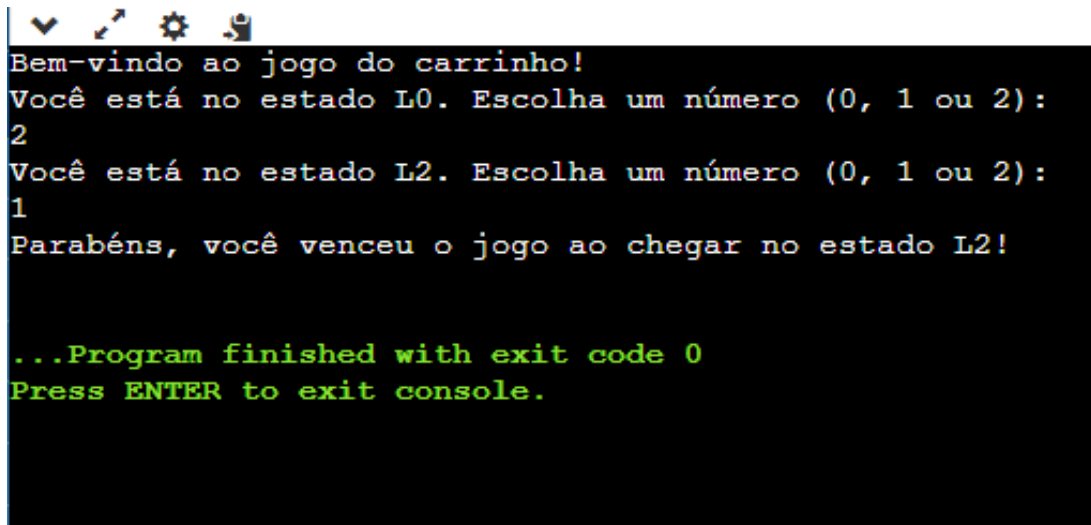
```
Bem-vindo ao jogo do carrinho!  
Você está no estado L0. Escolha um número (0, 1 ou 2):  
2  
Você está no estado L2. Escolha um número (0, 1 ou 2):
```

Fonte: Matheus, Diego, Nicolas, Pedro.

4.2 COLOCANDO NA UNITY

Conseguimos colocar o código em Unity mas sim alterando algumas coisas do código para que a Unity consiga entender melhor e usamos o "PUBLIC TRANSFORM" para declarar os estados e os "currentTarget" para que o carro se mova de estado para o outro, sendo assim, funcionou e a Unity conseguiu executar os comandos certamente, como mostrado nas imagens (Figura 4.4), (Figura 4.5), (Figura 4.6).

Figura 4.3

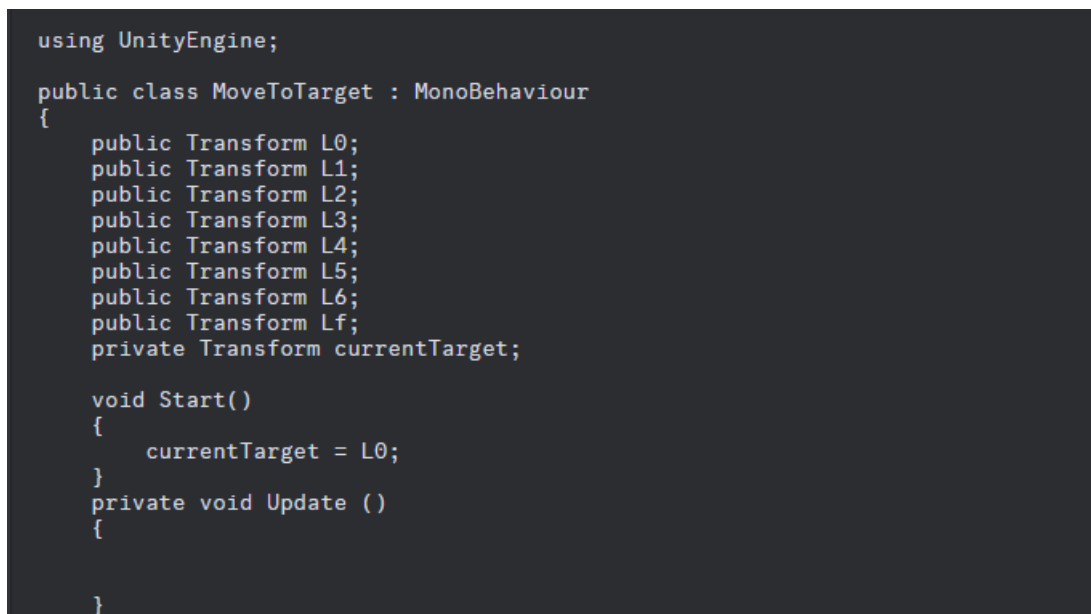


```
Bem-vindo ao jogo do carrinho!
Você está no estado L0. Escolha um número (0, 1 ou 2):
2
Você está no estado L2. Escolha um número (0, 1 ou 2):
1
Parabéns, você venceu o jogo ao chegar no estado L2!

...Program finished with exit code 0
Press ENTER to exit console.
```

Fonte: Matheus, Diego, Nicolas, Pedro.

Figura 4.4



```
using UnityEngine;

public class MoveToTarget : MonoBehaviour
{
    public Transform L0;
    public Transform L1;
    public Transform L2;
    public Transform L3;
    public Transform L4;
    public Transform L5;
    public Transform L6;
    public Transform Lf;
    private Transform currentTarget;

    void Start()
    {
        currentTarget = L0;
    }
    private void Update ()
    {
    }
}
```

Fonte: Matheus, Diego, Nicolas, Pedro.

4.3 JOGO EM FUNCIONAMENTO

O jogo começou a funcionar corretamente, com isso podemos confirmar que na tela o jogador pode digitar esses numeros 0,1 e 2, para que o carro alcance um estado feito no mapa, de acordo com o diagrama de estado (Figura 1.2), e o modelo do carro funcionou corretamente no jogo, tendo mudanças de direções por conta dos ponteiros usados na Unitty.

Figura 4.5

```

private void Update ()
{

}

void FixedUpdate ()
{
    // Verifica se um dos botões foi pressionado
    switch (Input.inputString)
    {
        case "1":

            if (currentTarget == L0) { currentTarget = L1; break; }
            if (currentTarget == L1) { currentTarget = L4; break; }
            if (currentTarget == L2){ currentTarget = L1; break; }
            if (currentTarget == L3) { currentTarget = L0; break; }
            if (currentTarget == L4) { currentTarget = Lf; break; }
            if (currentTarget == L5) { currentTarget = L4; break; }
            if (currentTarget == L6) { currentTarget = L3; break; }
            if (currentTarget == Lf) { currentTarget = Lf; break; }
            break;

        case "2":
            if (currentTarget == L0) { currentTarget = L2; break; }
            if (currentTarget == L1) { currentTarget = L2; break; }
            if (currentTarget == L2) { currentTarget = L5; break; }
            if (currentTarget == L3) { currentTarget = L2; break; }
            if (currentTarget == L4) { currentTarget = L5; break; }
            if (currentTarget == L5) { currentTarget = Lf; break; }
            if (currentTarget == L6) { currentTarget = L5; break; }
            if (currentTarget == Lf) { currentTarget = Lf; break; }
            break;
    }
}

```

Fonte: Matheus, Diego, Nicolas, Pedro.

Figura 4.6

```

        case "0":
            if (currentTarget == L0) { currentTarget = L3; break; }
            if (currentTarget == L1) { currentTarget = L0; break; }
            if (currentTarget == L2) { currentTarget = L3; break; }
            if (currentTarget == L3) { currentTarget = L6; break; }
            if (currentTarget == L4) { currentTarget = L1; break; }
            if (currentTarget == L5) { currentTarget = L6; break; }
            if (currentTarget == L6) { currentTarget = Lf; break; }
            if (currentTarget == Lf) { currentTarget = Lf; break; }
            break;
    }

    // Move o objeto em direção ao alvo atual
    float speed = 10f; // Velocidade de movimento
    transform.position = Vector3.MoveTowards(transform.position, currentTarget.position, speed * Time.deltaTime);
}
}

```

Fonte: Matheus, Diego, Nicolas, Pedro.

5 CONCLUSÕES

O desenvolvimento deste software marca um avanço significativo na interseção entre a teoria da computação e a prática aplicada, especialmente no contexto da educação e do entretenimento digital. Ao mergulharmos nos autômatos finitos determinísticos (AFD) e suas ramificações em linguagens formais, transcendemos a mera compreensão teórica para criar uma ferramenta multifacetada com aplicações que abrangem desde análises de linguagens até a construção de experiências interativas e educacionais.

Nossa imersão na teoria das expressões regulares e dos AFDs não apenas consolidou nosso entendimento conceitual, mas também serviu como fundação sólida para a concepção de um jogo educativo pioneiro. Ao integrar conceitos intrincados como estados, símbolos de entrada, transições e estados de aceitação em uma interface dinâmica e imersiva, facilitamos a assimilação desses conceitos, tornando-os mais acessíveis e atraentes para estudantes e profissionais da computação.

A implementação do jogo educativo do carrinho, desenvolvido em C na plataforma Unity, ilustra de forma concreta a aplicação prática dos princípios teóricos explorados neste projeto. Ao guiar os usuários através de estados e transições em um ambiente interativo, demonstramos não apenas a viabilidade, mas a eficácia de traduzir a teoria da computação em uma experiência lúdica e educativa.

Além disso, destacamos o papel essencial dos jogos educacionais como catalisadores do aprendizado e da exploração de conceitos complexos. Ao adotarmos uma abordagem inovadora e estimulante para o estudo desses temas, estamos pavimentando o caminho para futuras investigações e iniciativas que combinem a profundidade conceitual da teoria da computação com a atratividade e a interatividade dos jogos digitais. [12:44] Em suma, este trabalho não só contribui para a disseminação do conhecimento em teoria da computação, mas também evidencia o potencial transformador dos jogos educacionais como ferramentas de aprendizado envolventes e eficazes. Esperamos que essa iniciativa inspire e motive novas pesquisas e projetos que explorem a intersecção entre teoria e prática de maneira impactante e inovadora.

REFERÊNCIAS BIBLIOGRÁFICAS

APÊNDICES

A : AUTOMATOS FINITOS DETERMINISCOS

Descrição

1. Autômatos Finitos (AFs) Os autômatos finitos são modelos computacionais que descrevem sistemas com estados discretos e transições entre esses estados. Eles são amplamente utilizados em ciência da computação e linguagens formais. Vamos explorar os principais pontos:

Definição: Um autômato finito é uma máquina de estados finita que aceita ou rejeita cadeias de símbolos. Ele gera um único ramo de computação para cada cadeia de entrada. Componentes de um AF: Conjunto de Estados (Q): Um conjunto finito de estados. Alfabeto (Σ): Um conjunto finito e não vazio de símbolos de entrada. Função de Transição (δ): Uma função que mapeia pares (estado, símbolo) para um novo estado. Estado Inicial (q_0): O estado inicial a partir do qual a computação começa. Conjunto de Estados Finais (F): Um conjunto de estados que indicam aceitação. Representação Gráfica: Os autômatos finitos podem ser representados por diagramas de transição (grafos direcionados e rotulados). Os vértices representam os estados (fisicamente, são círculos). O estado inicial possui uma seta com rótulo “início”. Os estados finais são representados por círculos duplos. Autômatos Finitos Determinísticos (AFDs) Os autômatos finitos determinísticos são uma classe específica de AFs:

Definição: Um AFD é um autômato finito em que, para cada par (estado, símbolo), há no máximo uma transição possível. Isso significa que o próximo estado é determinado unicamente pelo estado atual e pelo símbolo de entrada. Reconhecimento de Sentenças: Um AFD reconhece ou rejeita sentenças com base em suas transições. Por exemplo, para reconhecer a sentença “abbaba”, o AFD realiza as seguintes transições: “ $q_0, q_1, q_1, q_1, q_0, q_0, q_1$ ”. Se terminar em um estado final, a sentença faz parte da linguagem definida pelo AFD. Exemplo de AFD: Dado o AFD abaixo e a sentença “abba”: !AFD Example A sentença “abba” pertence à linguagem representada por esse AFD. Em resumo, os autômatos finitos e determinísticos são ferramentas poderosas para modelar linguagens e sistemas computacionais. Eles são usados em áreas como compiladores, verificação de sistemas e detecção de padrões.

ANEXOS

A : TÍTULO

Publicação

1. : Prof. Amaury André - Autômatos Finitos 2: Wikiwand - Autômato finito determinístico 3: Prof. Amaury André - Autômatos Finitos 4: UFPE - Autômato finito não-determinístico