Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

BRUNO DE ANDRADE LUCIANO

MACHINE LEARNING APLICADO AO CONTEXTO DE JOGOS DIGITAIS

Assis/SP
2021

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

BRUNO DE ANDRADE LUCIANO

MACHINE LEARNING APLICADO AO CONTEXTO DE JOGOS DIGITAIS

Projeto de pesquisa apresentado ao curso de Ciência da Computação do Instituto Municipal de Ensino Superior de Assis – IMESA e a Fundação Educacional do Município de Assis – FEMA, como requisito parcial à obtenção do Certificado de Conclusão.

Orientando(a): Bruno de Andrade Luciano
Orientador(a): Prof. Msc. Guilherme de Cleva Farto

Assis/SP
2021

FICHA CATALOGRÁFICA

L937m LUCIANO, Bruno de Andrade

Machine Learning aplicado ao contexto de jogos digitais / Bruno de Andrade Luciano. – Assis, 2021.

40p.

Trabalho de conclusão do curso (Ciência da Computação). – Fundação Educacional do Município de Assis-FEMA

Orientador: Ms. Guilherme de Cleva Farto

1. Inteligência artificial 2. Jogos digitais

CDD006.3

MACHINE LEARNING APLICADO AO CONTEXTO DE JOGOS DIGITAIS

BRUNO DE ANDRADE LUCIANO

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, avaliado pela seguinte comissão examinadora:

Orientador: _____
Prof. Msc. Guilherme de Cleva Farto

Examinador: _____
Prof. Dr. Luiz Carlos Begosso

Assis/SP
2021

RESUMO

Ao assumir o constante crescimento e relevância que a indústria de jogos digitais vem alcançado nos últimos anos, fica nítida a necessidade de criar novos métodos e abordagens no que diz respeito principalmente a interação entre jogador e máquina. O grande impedimento que pode ser encontrado nas aplicações do mercado de *games* atual é a ausência da dinâmica desempenhada pelos elementos que compõem o mesmo, no qual por muitas vezes são implementados com o objetivo de reagirem exclusivamente a situações previamente determinadas pelos desenvolvedores. Um procedimento solucionador capaz de resolver a problemática citada seria a implementação de um método de Inteligência Artificial que aplique conceitos de *Machine Learning*. Desta forma, a abordagem tem como objetivo produzir interações com o usuário convincentes e o mais próximo do real, melhorando significativamente a experiência e imersão do mesmo com o jogo.

A proposta desse projeto tem por objetivo a implementação de um jogo eletrônico tridimensional que elucide a aplicação do Aprendizado de Máquina, a fim de que um agente inserido em um determinado ambiente atinja de maneira autônoma e inteligente os objetivos estipulados para o mesmo. A abordagem é de um veículo autônomo que busque percorrer determinado trajeto sem que colida com qualquer obstáculo durante a tarefa estabelecida.

Palavras-chave: *Machine Learning*; Jogos Digitais, Carro Autônomo, Inteligência Artificial.

ABSTRACT

By assuming the constant growth and relevance that the digital games industry has achieved in recent years, the need to create new methods and approaches regarding mainly the interaction between player and machine becomes clear. The great impediment that can be found in the applications of the current game market is the absence of the dynamics played by the elements that make up the same, which are often implemented with the objective of reacting exclusively to situations previously determined by the developers. A solving procedure capable of solving the aforementioned problem would be the implementation of an Artificial Intelligence method that applies Machine Learning concepts. In this way, the approach aims to produce convincing user interactions and as close to reality as possible, significantly improving the user's experience and immersion with the game.
The purpose of this project is to implement a three-dimensional electronic game that elucidates the application of Machine Learning, so that an agent inserted in a given environment can achieve, in an autonomous and intelligent way, the objectives stipulated for it. The approach is that of an autonomous vehicle that seeks to follow a certain path without colliding with any obstacle during the task established.

Keywords: Machine Learning; Digital Games, Autonomous Car, Artificial Intelligence.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1. INTRODUÇÃO

Nos últimos anos, a indústria dos jogos digitais vem sendo uma das maiores do mundo. Segundo Stephenson (2019), a indústria de *games* gerou no mercado global um valor de 108.9 bilhões de libras em maio de 2018, sendo maior que a indústria da música e cinema juntos, destacando sua grande importância na economia mundial.

Com a evolução e crescimento da área, torna-se necessário promover novas tecnologias e abordagens no que se refere ao desenvolvimento de jogos, principalmente acerca da interação entre o jogador e elementos do *game* propriamente dito. É essencial que tais componentes controlados pelo computador reajam de maneira convincente e natural, visando criar uma experiência agradável ao usuário final. Conforme descrito por Lara-Cabrera et al. (2015), essa interação pode ser análoga ao teste de Turing, na qual esse agente controlado pela máquina possui um comportamento humano, e não pode ser distinguido pelo real jogador. Desta forma, temas como Inteligência Artificial e Aprendizado de Máquina são vigentes, na qual proporcionariam reações dinâmicas a cada ato do jogador.

De acordo com Jess (2004, p.71-72), a Inteligência Artificial ou IA procura apresentar estratégias para simular ou imitar comportamentos humanos, realizando tomadas de decisões a partir de um problema proposto. Outra definição é que a IA caracteriza-se por ser um campo de estudo que busca explicar e emular comportamento inteligente em termos de processo computacional (SCHALKOFF, 1990, apud RUSSELL; NORVIG, 1995, p. 5).

Com o passar dos anos os jogos digitais ficam cada vez mais complexos, desta forma é nítido que elementos controlados pelo computador como personagens, objetos e o próprio ambiente, devem seguir tal evolução. No entanto, assim como descrito por Galway, Charles e Black (2008, p.123), a maioria das abordagens atuais dos elementos que compõem os jogos são estáticos e não se adaptam dinamicamente conforme ao comportamento do jogador. Nesse contexto, o Aprendizado de Máquina aplicado aos componentes descritos pode melhorar significativamente a experiência do usuário entregando respostas e interações mais convincentes ao mesmo.

1.1. OBJETIVOS

1.1.1. Objetivos Gerais

O objetivo desse trabalho é aplicar o Aprendizado de Máquina em um jogo eletrônico tridimensional, bem como utilizar tecnologias de redes neurais e aprendizado por reforço para que o agente inserido no ambiente seja capaz de atingir determinado objetivo a partir de regras propostas.

Pretende-se que a IA aprenda de maneira autônoma a conduzir um automóvel por meio de um trajeto predefinido, com a condição de que não colida com qualquer objeto e obstáculo em cena. Desta forma, a máquina irá gradualmente definindo o melhor percurso possível para que o carro atinja seu objetivo principal.

1.1.2. Objetivos Específicos

Pretende-se com este trabalho, implementar elementos baseados em conceitos de Inteligência Artificial e Aprendizado de Máquina por meio de um jogo tridimensional, utilizando-se o *game engine* (em Português, motor de jogo) *Unity3D* como ambiente de desenvolvimento. De forma a tornar possível a elaboração e execução, tanto das etapas teóricas quanto das práticas, os seguintes objetivos específicos foram instituídos:

- Pesquisar e analisar métodos e algoritmos de aprendizado de máquina;
- Pesquisar e analisar conceitos de redes neurais;
- Pesquisar e analisar a plataforma *Unity3D*, tornando possível a inclusão de tecnologias de IA nesse ambiente;
- Especificar um estudo de caso:
    - Modelar o problema;
    - Desenvolver a arquitetura do ambiente;
    - Desenvolver os componentes do jogo necessários;
    - Aplicar a IA aos componentes;
    - Testar, analisar os dados e avaliar os resultados;

- Descrever os resultados obtidos.

1.2. JUSTIFICATIVAS

A evolução no mercado de *games* deve crescer por volta de 5,3% até 2022, enquanto já no ano de 2018 proporcionou um faturamento de 1,5 bilhão de dólares (EXAME, 2019). Além disso, esse setor vem cada vez mais recebendo investimentos bilionários de empresas de grande representatividade tecnológica, como *Amazon*, *Google* e *Apple* (ESTADÃO, 2020).

Tendo em vista o crescimento e a grande relevância em que a área de jogos digitais possui atualmente, é interessante que seja investido em novas tecnologias no que se refere a esse contexto, principalmente naquelas que proporcionam uma melhor experiência ao jogador.

Desta forma, uma grande parcela de *games* que se encontram vigentes no cenário são constituídos por elementos que não respondem adequadamente a partir de certas interações que são realizadas pelo usuário. Um exemplo são os chamados NPCs (*non-player character*), na qual podem ser definidos como personagens não jogáveis presentes nos jogos que têm por finalidade interagir com o jogador, bem como apresentar o ambiente em questão, realizar diálogos, estabelecer tarefas, entre outros. Entretanto, esses elementos são implementados de maneira que reajam a interações já definidas, não se adequando dinamicamente de acordo com o comportamento do usuário.

Ao assumir esta problemática, uma solução seria empregar tecnologias de Inteligência Artificial em conjunto com Aprendizado de Máquina nos componentes presentes em *games*, assim tais elementos estariam preparados para qualquer ação e consequentemente entregaria uma resposta convincente e cativante.

1.3. MOTIVAÇÃO

O desenvolvimento deste projeto de pesquisa consiste no fato de que o *Machine Learning* (em Português, Aprendizado de Máquina) ainda é pouco aplicado ao contexto de jogos digitais, na qual pode ser um objeto de solução para problemas complexos que a programação convencional custosamente resolveria.

Outra motivação é o exponencial avanço e desenvolvimento no que se refere a Inteligência Artificial, principalmente na área de Aprendizado de Máquina, que constantemente demonstra-se promissor em suas aplicações. Líderes de negócio e investidores concordam que essas tecnologias transformarão seus negócios reduzindo custos, gerenciando riscos, otimizando operações, acelerando o crescimento e estimulando a inovação (FORBES, 2020).

1.4. PERSPECTIVAS DE CONTRIBUIÇÃO

Ao final do desenvolvimento deste trabalho, espera-se que o mesmo seja publicado no formato de artigos e divulgado em instituições de ensino para pessoas com interesse em jogos digitais e *Machine Learning*, com o objetivo de promover e compartilhar os conhecimentos e resultados alcançados. O jogo desenvolvido utilizando tecnologias de IA e Aprendizado de Máquina possibilitou o treinamento de um agente e apresentou sua evolução diante da interação com o ambiente, com o objetivo de contribuir com futuros projetos na área apresentada.

1.5. METODOLOGIA DE PESQUISA

A proposta e objetivos deste trabalho acadêmico são alcançados por meio de pesquisas pertinentes relacionadas a temática proposta, de forma a adquirir os conhecimentos necessários para que seja possível a realização do projeto. A metodologia empregada para o presente trabalho é composta pelo levantamento bibliográfico, caracterização do jogo a ser desenvolvido e resultados alcançados.

O levantamento bibliográfico é um estudo exploratório das principais referências relacionadas a Inteligência Artificial e suas vertentes. São abordados temas como *Machine Learning*, redes neurais e tipos de aprendizado de máquina.

No que se refere ao desenvolvimento do jogo, foram primeiramente estabelecidos a cena em que o mesmo ocorre e os elementos nele presentes, além de preparar todo ambiente necessário para que o Aprendizado de Máquina seja aplicado. Em seguida, foi de fato definido os componentes de Inteligência Artificial que interagiram com o *game*, na qual gradualmente realizou ações para que atinja da melhor forma o objetivo estipulado.

Por fim, a partir do processo da IA com o jogo, foi feita uma análise dos dados e uma avaliação qualitativa dos resultados obtidos, na qual definiu a evolução da máquina em função do tempo decorrido.

1.6. RECURSOS NECESSÁRIOS

Para desenvolver a pesquisa foram necessários os recursos de *hardware* e *software* citados a seguir:

- ***Hardware***
  - *Notebook Lenovo G40.*
  - Processador *Intel Core i3-5005U 2.00 GHz.*
  - Disco Rígido *Western Digital* 5400 RPM SATA de 1 TB.
  - Memória de 1600 MHz DDR3 8 GB

- ***Software***
  - ***Unity3D*** – Plataforma para desenvolvimento de jogos que reúne ferramentas e recursos necessários para a implementação do mesmo.

1.7. ESTRUTURA DO TRABALHO

A estrutura deste trabalho é composta das seguintes partes:

- **Capítulo 1 – Introdução:** Neste capítulo é contextualizada a área de estudo e apresentará os objetivos, justificativas, motivação, perspectivas de contribuição e metodologia de pesquisa para o desenvolvimento deste trabalho.
- **Capítulo 2 – Inteligência Artificial:** Neste capítulo, apresentam-se as vertentes de IA pertinentes ao trabalho, tais como conceitos de *Machine Learning*, Redes Neurais e tipos de Aprendizado de Máquina.
- **Capítulo 3 – Plataforma *Unity3D*:** Neste capítulo, é abordado a plataforma de desenvolvimento a ser utilizada no desenvolvimento do *game*, bem como suas respectivas ferramentas e recursos disponíveis.
- **Capítulo 4 – Proposta de Trabalho:** Neste capítulo, é apresentado uma solução de direção autônoma de um veículo aplicado em um jogo digital a partir do *Machine Learning*.

- **Capítulo 5 – Desenvolvimento do Trabalho:** Neste capítulo, é apresentado todo o processo de desenvolvimento do projeto, bem como a configuração do ambiente, evolução do treinamento de *Machine Learning* e obtenção de resultados.
- **Capítulo 6 – Conclusão:** Neste capítulo, são expostos as vantagens e desvantagens da implementação de uma Inteligência Artificial aplicada ao contexto de jogos digitais como solucionador de determinada problemática. Além disso, é citado pontos que podem ser evoluídos e ideias para projetos futuros.
- **Referências**

2. INTELIGÊNCIA ARTIFICIAL

Este capítulo possui como finalidade, abranger conceitos da Inteligência Artificial, ao expor definições de *Machine Learning* e seus respectivos tipos de Aprendizado de Máquina. Ademais, será abordado o tema de Redes Neurais, na qual é levantado ideias pertinentes ao contexto apresentado.

2.1. MACHINE LEARNING

Machine Learning pode ser definido como estudos de métodos utilizados no ambiente computacional que proporciona a implementação de comportamentos sem que essa seja programada diretamente. Desta forma, em situações que indivíduos são incapazes de fornecer informações precisamente para o sistema a partir de um comportamento desejado, o *Machine Learning* (ML) torna-se pertinente para o cenário em questão (DIETTERICH, 2003).

Outra definição do termo é que assim como nós, seres humanos, na qual somos capazes de reconhecer e classificar padrões dos mais diversos tipos, o *Machine Learning* busca apresentar esse mesmo comportamento aplicado em um determinado ambiente computacional. A partir de milhões de anos de evolução, o reconhecimento de padrões foi necessário para a sobrevivência da espécie, e que desta forma naturalmente atribuiríamos algumas dessas tarefas para máquinas como resultado do constante crescimento de tecnologias referentes. Ao responsabilizar máquinas na execução de funções de forma automatizada, considerando sua confiabilidade e precisão, deixaríamos de importar-se com tarefas simples e triviais, mas sim focando no entendimento mais profundo dos dados gerados pela mesma (DUDA et al., 2000).

Inconscientemente conseguimos por exemplo caracterizar e reconhecer a face de qualquer pessoa, identificando características e nuances através de diferentes poses, contrastes de iluminação, formatos, entre outros. Entretanto, considerando esse fato da incapacidade de compreender o que fazemos para tal tarefa, igualmente não somos capazes de desenvolver manualmente um sistema que realize o reconhecimento de faces da mesma forma que praticamos naturalmente. Por esse motivo, o Aprendizado de Máquina se adequa perfeitamente, na qual identificaria

padrões e estruturas comuns a partir de uma determinada imagem, que por fim resultaria no reconhecimento facial esperado (ALPAYDIN, 2009).

Para a implementação de um método de Inteligência Artificial considerando o *Machine Learning*, usa-se inicialmente parâmetros para o desenvolvimento de um modelo definido, e que a partir destes dados coletados, a máquina realiza uma execução em busca de otimizar essas informações, bem como utilizar sua própria base de dados e conhecimento para o processamento inteligente da tarefa empregada. Em geral, o modelo a ser aplicado pode desempenhar o papel de predição ou descrição, na qual respectivamente é definido por realizar previsões no futuro, ou consolidar dados na produção de conhecimento (ALPAYDIN, 2009).

2.1.1. Tipos de Aprendizado de Máquina

As tarefas relacionadas ao Aprendizado de Máquina podem ser classificadas em três tipos, comumente chamadas de Aprendizado Supervisionado, Não Supervisionado e Aprendizado por Reforço.

Na Figura 1 são apresentadas as vertentes do *Machine Learning*, bem como a área de atuação de cada tipo de aprendizado. Além disso, são detalhados alguns algoritmos existentes de acordo com a característica dos mesmos.

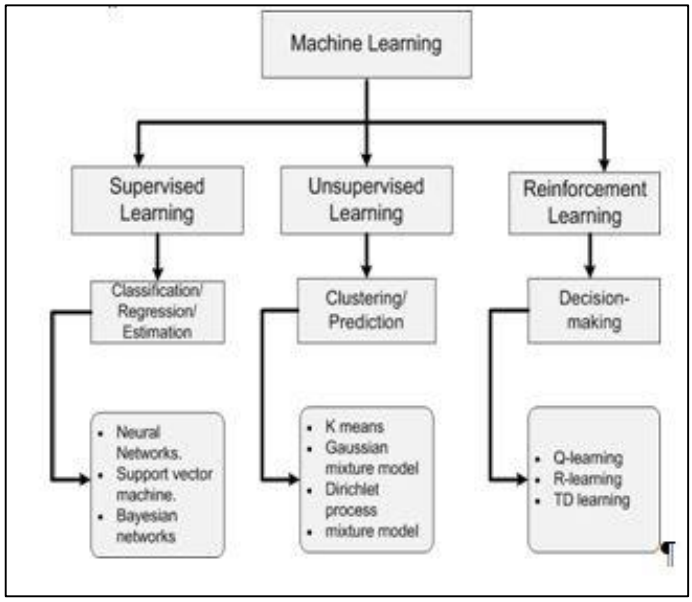

Figura 1: Classificação e Algoritmos relacionados ao *Machine Learning*
Fonte: GOEL, 2019

2.1.1.1. Aprendizado Supervisionado

De acordo com Alpaydin (2009), um modelo de *Machine Learning* é caracterizado como supervisionado ao encontrarmos uma regra de associação entre os dados de treinamento, ou seja, para cada informação de entrada é gerado respectivamente um resultado definido de saída.

A execução do algoritmo deste tipo de aprendizado busca-se por retornar os valores de saída ao receber os parâmetros de entrada do treinamento, levando em consideração sua base de conhecimento após diversas iterações em um ambiente de exemplo proposto (BREVE, 2010). Além disso, de forma geral, este aprendizado abrange grande parte dos problemas encontrados, na qual é identificado explicitamente todos os parâmetros de entrada e os possíveis dados de saída identificado pela máquina (HONDA, et al., 2017).

2.1.1.2. Aprendizado Não Supervisionado

No que se refere a Aprendizado Não Supervisionado, o processo de aprendizado não necessita de uma base de informações com classes rotuladas da forma como ocorre no Aprendizado Supervisionado, na qual utilizada de exemplos pré-classificados. O foco deste tipo de aprendizado é a busca de propriedades relevantes nos dados que estão disponíveis para o treinamento, e levando em consideração que estes não dispõem exemplos com valores de saídas, os algoritmos empregados procuram por grupos ou exemplos de valores semelhantes. A técnica citada é denominada de *Cluster*, que por sua vez é a tarefa fundamental que o Aprendizado Não Supervisionado executa, identificando de alguma forma grupos presentes nos dados informados, e gerando algum tipo de resultado para tal classificação (KUBAT, 2017).

2.1.1.3. Aprendizado por Reforço

Diferentemente dos Aprendizados Supervisionados e Não Supervisionados, a abordagem que o Aprendizado por Reforço segue é outra. Ao invés de realizar a indução de exemplos pré-classificados como dos modelos anteriores, nesta abordagem o aprendizado ocorre devido a interação dos agentes com o sistema, desempenhando literalmente um comportamento de tentativa e erro. Esses

parâmetros são definidos por um método de obtenção de recompensas e punições. Após várias iterações com o modelo, o agente buscará por priorizar as recompensas e evitar as punições, criando desta forma uma base de conhecimento a partir da experimentação (KUBAT, 2017).

Fundamentalmente uma ação isolada não importa para o aprendizado, mas sim uma sequência de ações que resultam no alcance da meta estipulada. Desta forma, o *Machine Learning* realiza uma análise das sequências que permitem a maximização de recompensas, e prioriza que esses caminhos adotados pelo agente sejam repetidos em próximas iterações (ALPAYDIN, 2009).

2.2. REDES NEURAIS ARTIFICIAIS

De acordo com Haykin (2008), Redes Neurais Artificiais ou comumente conhecida como "Redes Neurais", obtiveram atenção ao realizarem estudos a respeito do cérebro humano, que por sua vez é caracterizado por um sistema altamente complexo e veloz, distinto de qualquer computador convencional conhecido previamente a estes trabalhos. A partir de observações da anatomia cerebral, definiu-se que havia estruturas capazes de se organizar e armazenar conhecimento a um ponto que consigam fazer cálculos muito rapidamente.

As estruturas responsáveis por tal comportamento passaram a ser conhecidas como neurônios, e que ao se interconectarem com outros em uma complexa teia, constroem um grande sistema de aprendizado. Basicamente em uma Rede Neural Artificial (em Inglês, *Artificial Neural Network* - ANN), cada neurônio possui um valor único de entrada, e por consequência produzem um respectivo valor de saída. Conforme o tamanho da Rede Neural, o dado de entrada de certa unidade refere-se ao valor de saída de outros neurônios, enquanto o dado de saída de um neurônio relaciona-se a vários valores de entrada de outras unidades (MITCHELL, 1997).

A Figura 2 representa a arquitetura de uma Rede Neural Artificial, indicada por neurônios na camada de entrada, camada oculta e camada de saída. É possível analisar que para cada neurônio é associado o valor de saída para todas as unidades subsequentes da próxima camada.

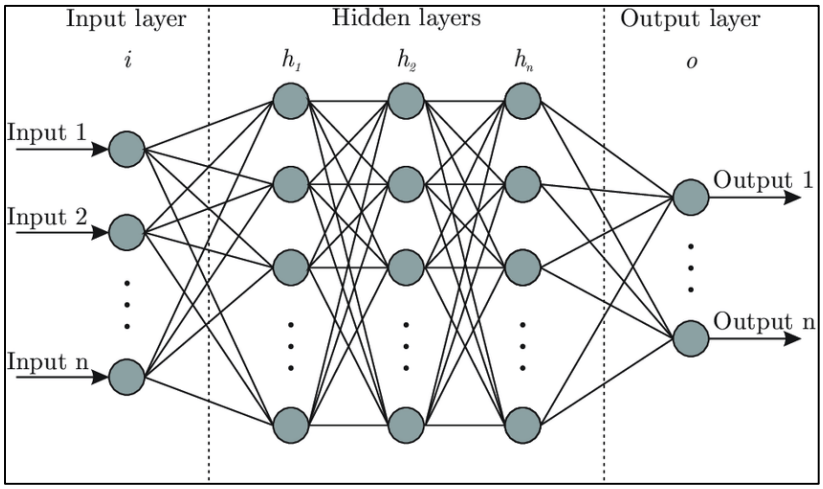

Figura 2: Arquitetura de uma Rede Neural Artificial
Fonte: BRE et al., 2018

3. PLATAFORMA UNITY3D

3.1. INTRODUÇÃO

Unity3D é uma plataforma de desenvolvimento em tempo real capaz de oferecer um ecossistema unificado de ferramentas que possibilitam a implementação de jogos digitais, renderização de ambientes tridimensionais para projetos de arquitetura, engenharia e construção. Além disso, a aplicação possibilita a criação de produções cinematográficas, como filmes e animações (UNITY, 2021).

Segundo Brodkin (2013), por volta dos anos 2000, David Helgason, Joachim Ante e Nicholas Francis foram os três programadores responsáveis por iniciarem a criação de uma plataforma integrada para produção de jogos, no qual posteriormente se tornaria um dos *softwares* mais utilizados no mercado mundial. Após anos de consolidação e atualizações, o *Unity3D* trouxe compatibilidade para diversos dispositivos e plataformas presentes no mercado, como computadores de diferentes sistemas operacionais *(Windows, Mac e Linux)*, consoles, navegadores e dispositivos móveis.

O *Game Engine* ficou popularmente conhecido por sua fácil acessibilidade para qualquer um que deseja desenvolver, desde *games* mais simples até aqueles com megaproduções, envolvendo diversas equipes e setores diferentes. Desta forma, levando em consideração a disponibilidade e o baixo custo de uso da plataforma, esta torna-se de grande benefício principalmente para aqueles desenvolvedores *indie*, ou seja, trabalham de forma independente e com um número reduzido de pessoas aplicado em sua equipe de produção. Por consequência, o indivíduo que esteja no processo de criação de um *game*, não necessita atentar-se a tecnologias que farão de fato o mesmo ser executado, mas sim destinando todo o foco e empenho no que se refere ao processo artístico e criativo (BRODKIN, 2013).

3.2. FERRAMENTAS E RECURSOS

3.2.1. Editor

No ecossistema do *Unity3D*, é apresentado nativamente um editor que proporciona toda manipulação referente aos elementos existentes de um jogo, tais como modelos 3D, *scripts* (códigos quem podem ser associados a elementos com o propósito de realizar determinados comportamentos), áudios, efeitos de imagem, interface de usuário (em inglês, *User Interface* - UI), etc.

A plataforma é dividida em várias janelas, na qual individualmente possuem responsabilidades e funções diferentes (HAAS, 2014). Das visualizações do editor mais usadas encontram-se as seguintes janelas:

- *Project Browser*: semelhante ao explorador de arquivos do sistema operacional, na qual lista todos os elementos disponíveis e importados para dentro da plataforma.
- *Inspector*: tem por função listar todos os detalhes de determinado elemento (no *Unity3D*, cada objeto é denominado como *Game Object*, em português, Objeto de Jogo), como informações e valores associados ao mesmo.
- *Game View*: basicamente é uma visualização de como o *game* ficará ao ser executado.
- *Scene View*: é o local em que o jogo é construído, ou seja, é possível manipular e arrastar elementos para dentro da visualização.
- *Hierarchy*: lista todos os *Game Objects* que estão na cena atual.

Na Figura 3 é possível identificar as janelas citadas anteriormente, bem como visualizar alguns Objetos de Jogo já instanciados no projeto.

Figura 3: Editor do Unity3D
Fonte: UNITY, 2021

3.2.2. Scripts

Para que a mecânicas e funcionalidades funcionem adequadamente é preciso que sejam implementadas tais tarefas em uma determinada linguagem de programação, e para isso o *Unity3D* dispõem dos chamados *Scripts*, na qual são arquivos que podem ser associados aos Objetos de Jogo do projeto. A linguagem comumente utilizada para desenvolver esses *Scripts* é o C#, que por sua vez segue o paradigma orientado a objetos incorporado a estrutura nativa da plataforma.

Exemplo de implementação de um *Script*, que neste exemplo é responsável por realizar a movimentação de um elemento (UNITY, 2021):

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Example : MonoBehaviour
{
    private CharacterController controller;
    private Vector3 playerVelocity;
    private bool groundedPlayer;
    private float playerSpeed = 2.0f;
    private float jumpHeight = 1.0f;
    private float gravityValue = -9.81f;

    private void Start()
    {
        controller = gameObject.AddComponent<CharacterController>();
```

```
    }

    void Update()
    {
        groundedPlayer = controller.isGrounded;
        if (groundedPlayer && playerVelocity.y < 0)
        {
            playerVelocity.y = 0f;
        }

        Vector3 move = new Vector3(Input.GetAxis("Horizontal"), 0,
Input.GetAxis("Vertical"));
        controller.Move(move * Time.deltaTime * playerSpeed);

        if (move != Vector3.zero)
        {
            gameObject.transform.forward = move;
        }

        // Changes the height position of the player..
        if (Input.GetButtonDown("Jump") && groundedPlayer)
        {
            playerVelocity.y += Mathf.Sqrt(jumpHeight * -3.0f *
gravityValue);
        }

        playerVelocity.y += gravityValue * Time.deltaTime;
        controller.Move(playerVelocity * Time.deltaTime);
    }
}
```

3.2.3. Biblioteca para *Machine Learning*

O *Unity3D* disponibiliza ferramentas que possibilitam o desenvolvimento e treinamento prático de agentes utilizando uma combinação de Aprendizado Profundo por Reforço e Aprendizado por Imitação. O *Unity Machine Learning Agents (ML-Agents)* é uma biblioteca de código aberto que dispõem de diversos algoritmos prontos para utilização, não exigindo de um grande conhecimento específico para realizar a manipulação da ferramenta (UNITY, 2021).

4. PROPOSTA DE TRABALHO

Este trabalho tem por objetivo desenvolver um jogo tridimensional por meio da plataforma *Unity3D* que desempenhe a representação de um carro autônomo. Foi especificado uma área que este veículo possa locomover-se, além de ser adicionado obstáculos em locais estratégicos para que em seguida afete propositalmente os testes e evolução do método de Inteligência Artificial empregado. Basicamente, foi definido o objetivo que o automóvel deva deslocar-se pelo circuito e que obrigatoriamente consiga completar toda a trajetória estabelecida sem que ele atinja qualquer obstáculo.

O responsável em realizar estas rotinas mencionadas é o agente aplicado ao elemento do veículo, na qual aplicou o método de Aprendizado por Reforço para que atinja os objetivos estipulados.

Após a atuação de uma sequência de iterações, foram avaliados os resultados obtidos e verificado a evolução que o Aprendizado de Máquina teve aplicado ao contexto do jogo, e por consequência, foi possível realizar uma conclusão a partir dos dados coletados. As métricas referentes ao aprendizado foram coletadas por meio da ferramenta *Tensorboard*, na qual trabalha em conjunto com o *ML-Agents* e converte todos os parâmetros de treinamento em gráficos e estatísticas, criando desta forma uma visualização geral do modelo empregado.

4.1. ELEMENTOS E ESPECIFICAÇÃO DO JOGO

A princípio foram definidos todos os elementos a serem empregados dentro do ambiente do *game*, bem como o veículo, o circuito e os objetos que fizeram o papel de obstáculos a fim de dificultar a movimentação do carro pelo trajeto. Nas Figuras 4, 5 e 6 seguem respectivamente exemplos de modelos tridimensionais que foram aplicados no contexto do projeto.

Figura 4: Modelo 3D do veículo

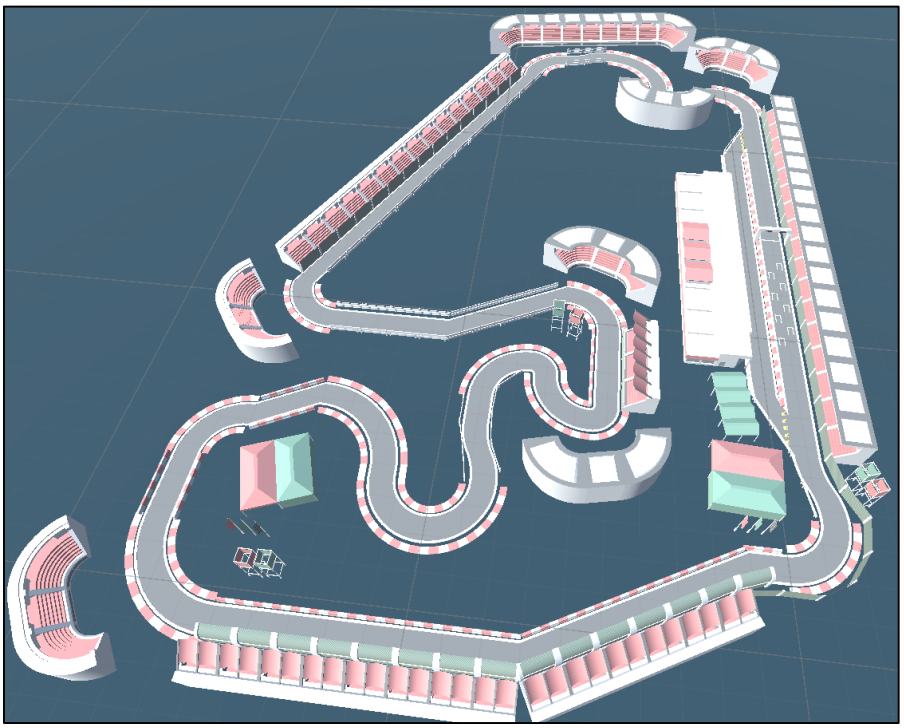

Figura 5: Modelo 3D do circuito

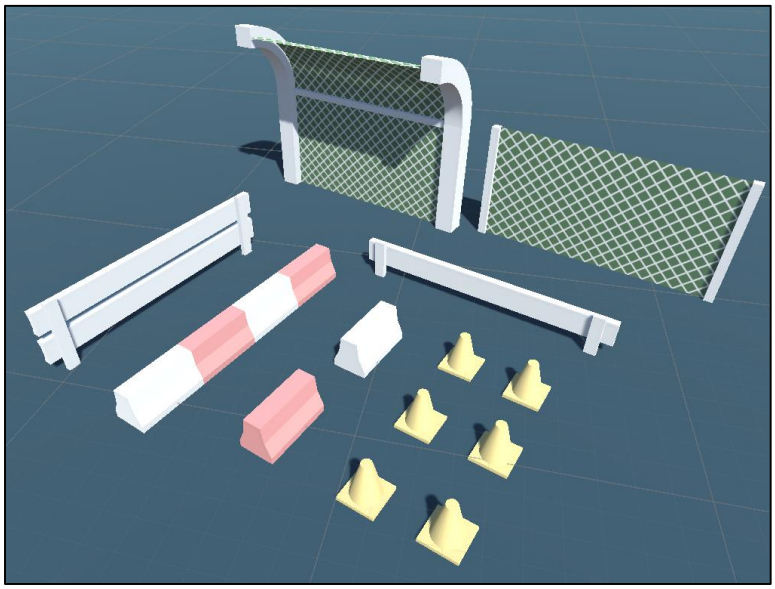

Figura 6: Modelo 3D de obstáculos

4.2. APLICAÇÃO DO APRENDIZADO DE MÁQUINA

O *Machine Learning* aplicado ao contexto do projeto teve como abordagem o Aprendizado por Reforço, que por sua vez é a característica aplicada pela biblioteca *ML-Agents* incorporado na plataforma *Unity3D*. Assumindo este tipo de Aprendizado de Máquina, a cada ação correta realizada pelo veículo, ele receberá uma recompensa, que por sua vez pode ser interpretado como um comportamento correto e que deve ser priorizado e mantido nas próximas iterações. De outro modo, caso o agente execute um ato que seja considerado falho a partir dos objetivos estipulados, ele será punido e consequentemente tentará evitar estas mesmas ações para iterações futuras.

Nos circuitos em que o carro atuou, tiveram pontos estratégicos referentes as recompensas, que caso o agente consiga atingir essas áreas foram indicados que esses comportamentos devem ser priorizados. Analogamente, todos os obstáculos do ambiente tiveram como função indicar uma punição no momento em que estes forem colididos pelo veículo.

De forma prática, utilizou-se o *ML-Agents*, em que possui todas as ferramentas necessárias para a execução do contexto em questão. Esta biblioteca possui algoritmos e implementações baseadas na linguagem *Python* (biblioteca *PyTorch*) a

fim de facilitar todo o processo de criação e treinamento do Aprendizado de Máquina de um agente inserido no ambiente determinado.

4.3. AVALIAÇÃO E OBTENÇÃO DE RESULTADOS

Levando em consideração a avaliação e obtenção de resultados, foram definidas métricas na aplicação que possibilitem identificar as ações do agente em tempo real, apresentando as ações corretas ou falhas na obtenção de recompensas ou punições de acordo com o comportamento do veículo autônomo. Os parâmetros de aprendizado analisados foram aqueles coletados pelo *ML-Agents*, como a relação entre as recompensas acumuladas pelo agente e a duração dos episódios.

5. DESENVOLVIMENTO DO TRABALHO

Para o desenvolvimento do trabalho foi iniciado um novo projeto pela plataforma do *Unity3D*, na qual configura um ambiente padrão com câmera e iluminação, permitindo o início ágil do desenvolvimento.

5.1. CONFIGURAÇÃO E PREPARAÇÃO DE AMBIENTE

Primeiramente, foram instaladas todas as dependências referentes ao funcionamento do projeto, como o *Python* na versão 3.7, pacotes referentes ao *PyTorch* (biblioteca para *Machine Learning* de código livre) e *ML-Agents* para linha de comando e plataforma *Unity3D*.

Posteriormente, foram importados modelos tridimensionais do veículo, componentes da pista e obstáculos. Em seguida, implementou-se um *script* responsável pelos controles básicos do veículo, como a ação de acelerar, frear e virar para direita ou esquerda.

Considerando o ambiente de treinamento do agente utilizando *Machine Learning*, foram elaborados três circuitos com obstáculos para que o veículo realize os aprendizados necessários. Duas pistas são caracterizadas por um formato oval, diferenciados somente pela direção de suas respectivas curvas, na qual uma possui curvas somente para esquerda e outra para direita. A estratégia estabelecida teve por objetivo focar o aprendizado no que se refere a realização de curvas e evitar que o veículo autônomo colida com os obstáculos, que em geral são as paredes que circundam o trajeto estabelecido. Por fim, o terceiro circuito foi definido por variadas curvas e retas, apresentando-se de um modo mais complexo comparado com as outras pistas, propondo expor todo aprendizado obtido do agente através de seu comportamento ao ser incluído neste ambiente.

Nas Figuras 7, 8 e 9 são apresentados os circuitos para treinamento da máquina, na qual são envolvidos por paredes que delimitam a trajetória em que o agente deve realizar.

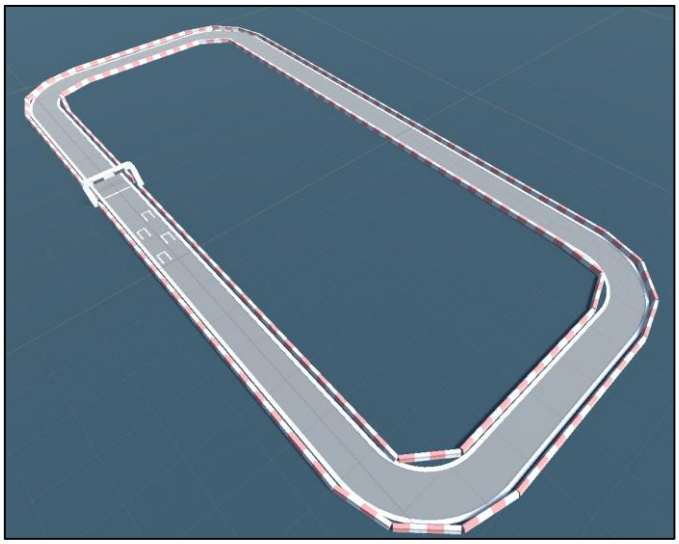

Figura 7: Circuito oval no sentido horário

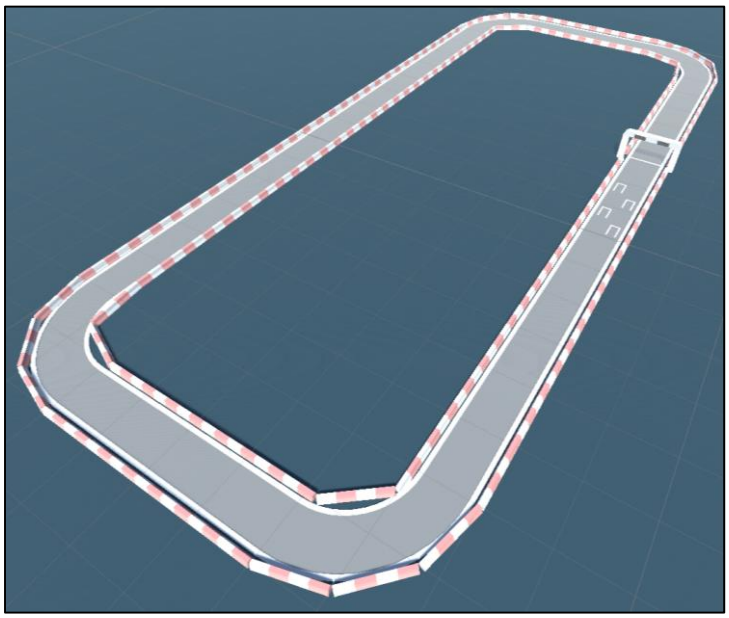

Figura 8: Circuito oval no sentido anti-horário

Figura 9: Circuito definitivo para treinamento

De forma que o veículo consiga aprender de forma eficaz a realizar a tarefa de completar uma volta pelo circuito sem que colida com qualquer obstáculo foi preciso definir certos pontos de controle, também nomeados de *checkpoints*. Estes estiveram distribuídos uniformemente pela pista, porém em maior quantidade nas curvas.

Na Figura 10 e 11 é evidenciada a alocação destes pontos de controle em todo trajeto, buscando elucidar o treinamento do *Machine Learning* para o caminho correto que devia ser percorrido.

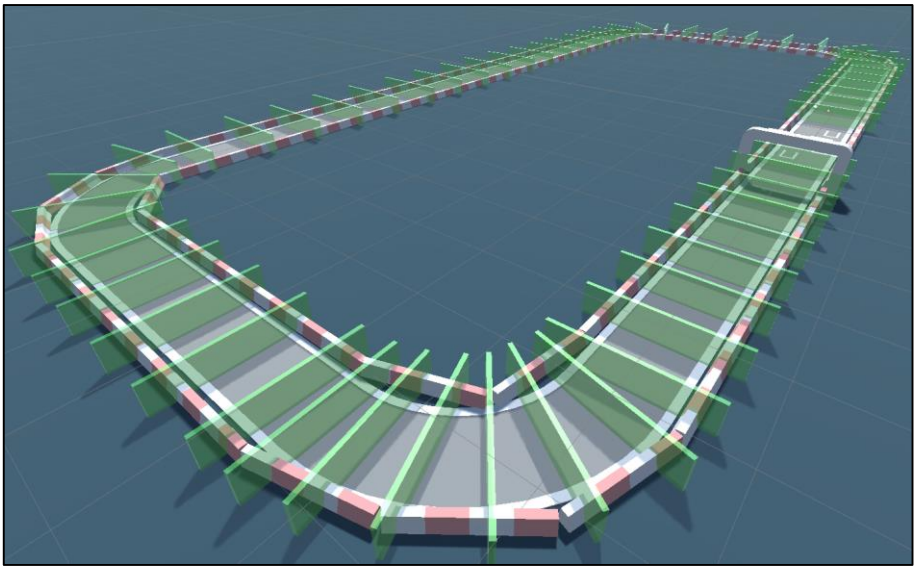

Figura 10: *Checkpoints* distribuídos pelo circuito oval

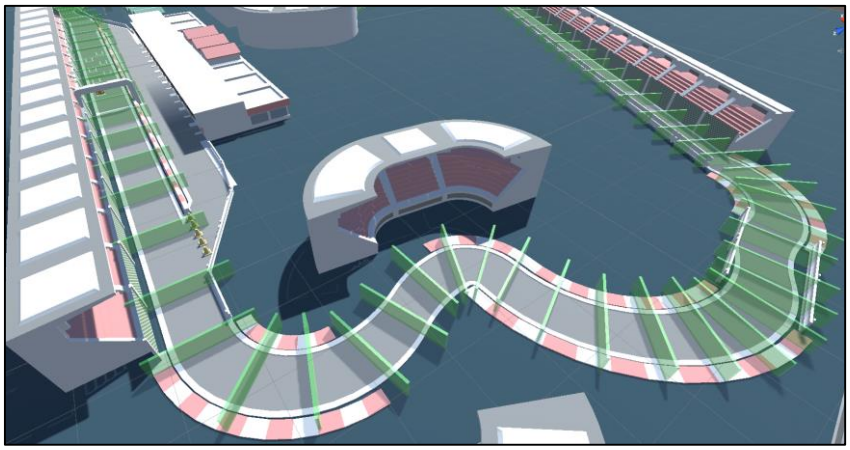

Figura 11: *Checkpoints* distribuídos pelo circuito definitivo

Para que o agente seja capaz de localizar-se no ambiente do jogo, foi utilizado raios sensores de percepção tridimensional, disponibilizados na plataforma *Unity3D* juntamente com a biblioteca *ML-Agents*. Em geral, ao inserir tal componente em um objeto, são definidas linhas em diferentes ângulos a partir do centro dele, e desta forma se torna capaz de enviar informações de objetos próximos como parâmetros de aprendizado para a máquina. Basicamente, durante o treinamento o agente irá entender que deve priorizar a proximidade de objetos definidos como *checkpoints*, e por consequência evitar objetos especificados como parede.

A Figura 12 ilustra esse componente empregado ao veículo, definindo pontos de colisão com objetos próximos ao mesmo.

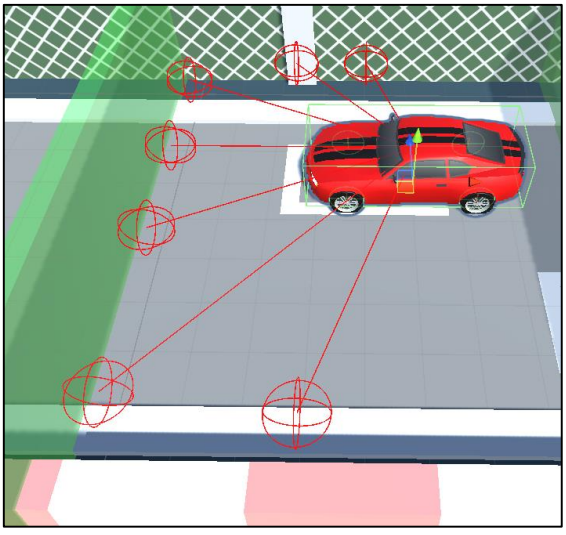

Figura 12: Sensor de percepção 3D no veículo

O sensor de percepção tridimensional não é capaz unicamente de atingir o objetivo principal do projeto, pois não oferece dados suficientes para que a IA entenda a tarefa e faça com o que o agente percorra perfeitamente o trajeto definido. Assim sendo, foi preciso enviar informações relacionadas a direção do veículo em relação ao *checkpoint* mais próximo*,* desta forma o carro sempre estará alinhado com o ponto de controle, evitando que em curvas ou retas mais acentuadas ele saia fora do trajeto estabelecido, e por consequência colida com os obstáculos.

Assumindo o aprendizado por reforço e suas características, foram definidos valores positivos como recompensa para as ações realizadas no momento em que o agente percorre corretamente cada *checkpoint*, e valores negativos de mesma intensidade quando o veículo colide com algum obstáculo do ambiente.

Durante a evolução do treinamento do algoritmo de Aprendizado de Máquina, foram constatados que em certos momentos o agente parava completamente pelo circuito e não se movimentava até o próximo ponto de controle estipulado. Desta forma, foi necessário criar validações em que verificam se o veículo atingiu o *checkpoint* em determinado tempo, que por sua vez foi considerado um valor de 30 segundos. Ao ultrapassar o tempo estipulado, o agente sofre uma pequena punição com um valor negativo de recompensa a cada 2 segundos. Caso o veículo ainda não consiga atingir o próximo ponto de controle, é estipulado que a cena seja reiniciada e o episódio finalizado ao ultrapassar o dobro do tempo inicial, ou seja, o valor de 1 minuto. A alternativa implementada funcionou perfeitamente como o esperado, pois a máquina envolvida passou a compreender que não era vantajoso manter-se parado, e consequentemente passou a realizar ações que evitavam tais situações.

5.2. TREINAMENTO DO MODELO

Ao preparar todo ambiente para treinamento, por meio da linha de comando foi possível iniciar o treinamento do aprendizado de máquina. Na Figura 13 é evidenciado o comando principal para dar o início no *Machine Learning*, na qual é caracterizado pelo comando **mlagents-learn**, seguido respectivamente por um arquivo de configuração de parâmetros de aprendizado e nome do modelo. Após aguardar o

processamento, é necessário somente iniciar a execução do programa dentro da plataforma *Unity3D*.

Figura 13: Iniciar treinamento do *ML-Agents*

Inicialmente, aplicou-se o *Machine Learning* ao agente nas pistas ovais com curvas para direita e esquerda, priorizando o aprendizado na mecânica referente a condução e evitando a colisão do mesmo com os obstáculos.

Com o objetivo de melhorar a velocidade de aprendizado, foi utilizado para as duas pistas citadas um recurso disponibilizado pelo *ML-Agents* denominado aprendizado por imitação, na qual a máquina aprendeu de acordo com as ações gravadas pelo usuário. Ao adicionar o componente no agente, foi possível definir uma gravação de movimentos em tempo de execução do jogo. Após repetir diversas vezes o objetivo esperado, podemos encerrar a gravação da demonstração. Em seguida gerou-se um arquivo contendo todos dados das operações realizadas, e ao configurar o arquivo de configuração de aprendizado pelo *ML-Agents*, é disponível definir tal arquivo de demonstração para que o *Machine Learning* leve em consideração em sua evolução. Desta forma, a obtenção de resultados torna-se mais ágil do que comparado ao iniciar diretamente pelo aprendizado por reforço, considerando o nível de complexidade de tal tarefa.

A Figura 14 apresenta o componente citado empregado ao agente, contendo respectivamente a opção de gravar demonstração, número de passos que serão gravados, nome da demonstração e o diretório que o arquivo será gravado.

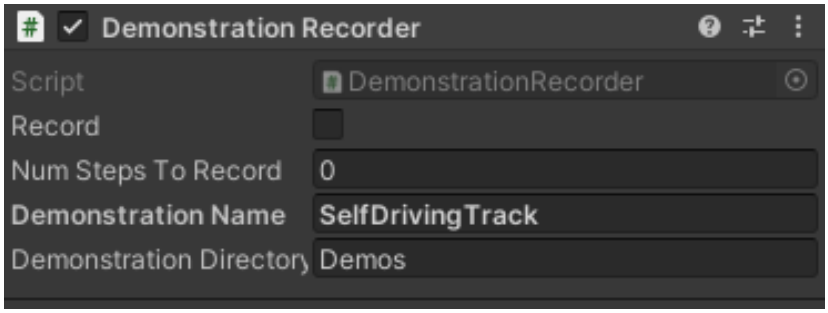

Figura 14: Componente do *ML-Agents* para aprendizado por imitação

Após realizar treinamento do veículo nas duas pistas ovais, foi redirecionado o mesmo para a pista definitiva, na qual possui uma complexidade maior do que as demais. Para este ambiente, utilizou-se somente do aprendizado por reforço, considerando a evolução realizada anteriormente nos outros circuitos.

5.3. RESULTADOS

No momento da instalação das dependências do *ML-Agents* foi incluída a biblioteca *Tensorboard*, na qual disponibiliza facilmente os resultados obtidos durante o aprendizado da máquina por meio de gráficos e estatísticas. A Figura 15 ilustra a execução do comando responsável em abrir a biblioteca *Tensorboard*.

```
(venv) C:\Users\Bruno\Desktop\TCC\MachineLearning_SelfDrivingCar\MachineLearning_SelfDrivingCar>tensorboard --logdir results
TensorFlow installation not found - running with reduced feature set.
Serving TensorBoard on localhost; to expose to the network, use a proxy or pass --bind_all
TensorBoard 2.5.0 at http://localhost:6006/ (Press CTRL+C to quit)
```

Figura 15: Iniciar visualização de resultados obtidos pelo *Tensorboard*

No que se refere ao aprendizado do agente no circuito oval com curvas para direita foi possível analisar uma evolução constante e progressiva dele. Na Figura 16, é caracterizado o baixo acúmulo de recompensas no início do aprendizado no gráfico referente a *Cumulative Reward*, porém foi aumentado progressivamente até certo ponto em que o agente dominou totalmente as ações empregadas em seu ambiente. No gráfico de *Episode Length*, refere-se à duração dos episódios de acordo com ambiente, na qual inicialmente levou-se um grande tempo até atingir alguma ação pelo veículo. Após algum tempo a máquina colidiu diversas vezes com os obstáculos e por

consequência os episódios foram encerrados e a cena reiniciou prematuramente. Após algumas horas de treinamento, o agente aprendeu todas as ações que deviam ser realizadas e por consequência a duração dos episódios passaram a ser maiores novamente.

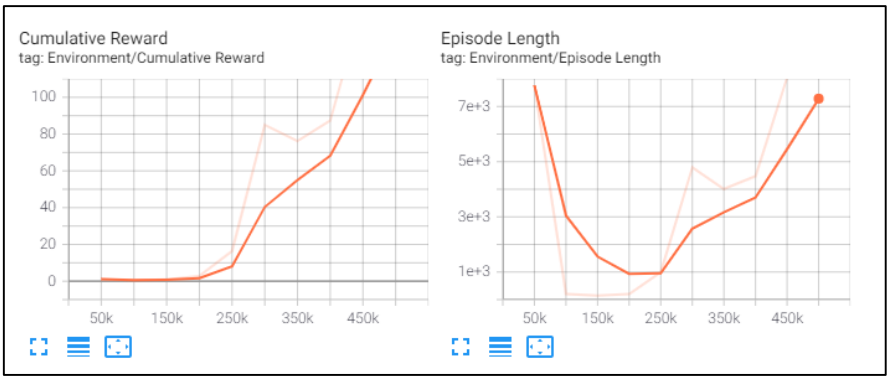

Figura 16: Aprendizado do agente no circuito oval com curvas para direita

Analogamente ao aprendizado anterior, na Figura 17 é apresentado a evolução do *Machine Learning* referente ao agente imposto no circuito definitivo, contando com diversas curvas e retas de maior complexidade. No gráfico de *Cumulative Reward* mantém-se semelhante com o aprendizado anterior, com pequenas variações de evolução. Já no *Episode Length*, iniciou-se com episódios de pouca duração, devido ao fato da condução do veículo nas curvas ser de maior complexidade. Com o tempo, a duração dos episódios se prolongou, porém o agente ainda não conseguia dar uma volta completa no circuito. Por fim, após diversas horas de aprendizado o índice de duração dos episódios foi prolongado e o *Machine Learning* conseguiu percorrer a pista perfeitamente e sem erros.

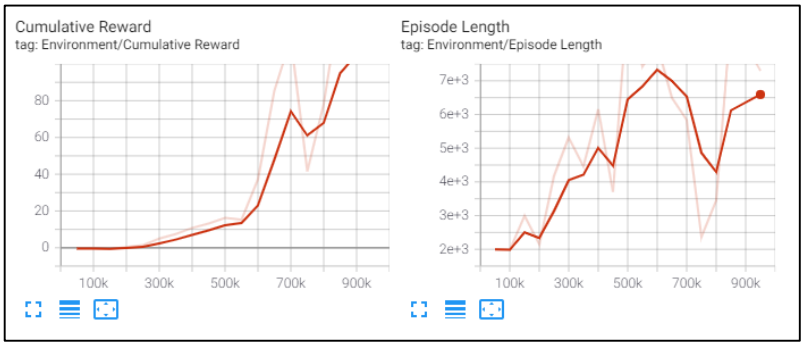

Figura 17: Aprendizado do agente no circuito definitivo

6. CONCLUSÃO

Após a implementação do projeto definido como a aplicação de um *Machine Learning* no contexto de um jogo digital tridimensional, foi possível realizar todo o aprendizado empregado, na qual o agente desenvolveu técnicas de forma autônoma para percorrer o circuito sem erros.

A plataforma *Unity3D*, juntamente com a biblioteca de *Machine Learning,* mostrou-se de fácil entendimento e manipulação, oferecendo diversos componentes prontos para o início ágil do aprendizado pelos agentes.

Foram possíveis as análises referentes ao aprendizado de máquina através do *Tensorboard,* uma biblioteca incluída em conjunto com o *ML-Agents* capaz de criar gráficos e estatísticas referentes aos parâmetros de aprendizado empregado ao agente incluído em determinado ambiente.

Desta forma, pode-se assumir que a abordagem realizada é válida para o contexto em questão, na qual o *Machine Learning* exerceu um grande benefício como solucionador do problema apresentado, embora existiu a necessidade de implementações para que o treinamento fosse realizado da melhor forma possível.

6.1. TRABALHOS FUTUROS

Após esta pesquisa pretende-se buscar um maior conhecimento sobre o tema, na qual todo ambiente empregado pelo trabalho pode ser evoluído em diversas formas mais complexas e especialistas, pois a plataforma *Unity3D* em conjunto com *ML-Agents* mostrou-se de grande capacidade para futuras expansões.

Uma alternativa de trabalho futuro seria permitir criar múltiplos agentes no mesmo circuito, e que desta forma aumentasse significativamente o tempo de aprendizado dos mesmos.

REFERÊNCIAS

ALPAYDIN, Ethem. **Introduction to Machine Learning**, 2. ed. Londres: The MIT Press, 2009.

BRE, Facundo; GIMENEZ, Juan M.; FACHINOTTI, Víctor D. Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks. **Energy and Buildings**, v.158, janeiro, 2018, p. 1429-1441.

BREVE, Fabrício A. **Aprendizado de máquina em redes complexas**. 2010. 184p. Tese (doutorado) – Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo, São Paulo, São Carlos, 2010.

BRODKIN, Jon. **How Unity3D Became a Game-Development Beast**. Disponível em <https://insights.dice.com/2013/06/03/how-unity3d-become-a-game-development-beast/>. Acesso em: 15 mar. 2021.

DIETTERICH, Thomas G. Machine Learning. **Encyclopedia Of Computer Science**, janeiro, 2003, p. 1056-1059.

DUDA, Richard O.; HART, Peter E.; STORK, David G. **Pattern Classification**, 2. ed. Nova Jersey: Wiley-Interscience, 2000.

ESTADÃO. **Mercado de games dispara: Saiba como aproveitar boa fase nos seus investimentos**. Disponível em: <https://einvestidor.estadao.com.br/mercado/crescimento-mercado-games-investimentos>. Acesso em: 17 nov. 2020.

EXAME. **Mercado de games no Brasil deve crescer 5,3% até 2022, diz estudo**. Disponível em: <https://exame.com/negocios/mercado-de-games-no-brasil-deve-crescer-53-ate-2022-diz-estudo/>. Acesso em: 17 nov. 2020.

FORBES. **Realizing the Growth Potential of AI**. Disponível em: <https://www.forbes.com/sites/forbesinsights/2020/05/08/realizing-the-growth-potential-of-ai/?sh=2679944b33f3>. Acesso em: 17 nov. 2020.

GALWAY, Leo; CHARLES, Darryl; BLACK, Michaela. Machine learning in digital games: a survey. In: DIGITAL BIBLIOGRAPHY & LIBRARY PROJECT (DBLP), 2008. **Artificial Intelligence Review**, v.29, abril, 2008, 123 - 161p.

GOEL, Kapil. Machine Learning Explained. In: TECHBYTE 2K19, 16, 2019, Nova Delhi, Índia. **16th Annual It Symposium "Technologies For Intelligent World"**, outubro, 2019.

HAAS, John K. **A History of the Unity Game Engine**. 2014. 44p. Trabalho de Conclusão de Curso – Social Studies of Science and Technology - Worcester Polytechnic Institute, Massachusetts, Worcester, 2014.

HAYKIN, Simon. **Neural Networks and Learning Machines**, 3. ed. Nova Iorque: Prentice Hall, 2008.

HONDA, Hugo; FACURE, Matheus; YAOHAO, Peng. **The three types of machine learning**. Disponível em: <https://lamfo-unb.github.io/2017/07/27/tres-tipos-am-english/>. Acesso em: 06 mar. 2021.

JESS, Gil Marcos. **Inteligência Artificial e Tecnologias da Inteligência - Um Repensar Segundo os Processos de Elaboração Matemática**. 2004. 139p. Dissertação (mestrado) – Universidade Federal do Paraná, Paraná, Curitiba, 2004.

KUBAT, Miroslav. **An Introduction to Machine Learning**, 2. ed. Springer, 2017.

LARA-CABRERA, Raúl; NOGUEIRA-COLLAZO, Mariela; COTTA, Carlos; FERNÁNDEZ-LEIVA, Antonio J. Game Artificial Intelligence: Challenges for the Scientific Community. In: PROCEEDINGS OF THE 2ND CONGRESO DE LA SOCIEDAD ESPAÑOLA PARA LAS CIENCIAS DEL VIDEOJUEGO, 2015, Barcelona, Espanha. **CEUR Workshop Proceeding**, junho, 2015, p.1-10.

MITCHELL, Tom M. **Machine Learning**, 1. ed. Nova Iorque: McGraw-Hill Science/Engineering/Math, 1997.

RUSSELL, Stuart J.; NORVIG, Peter. **Artificial Intelligence: A Modern Approach**. Englewood Cliffs: Editora Prentice Hall, 1995.

STEPHENSON, John. **6 Ways Machine Learning will be used in Game Development**. Disponível em: <https://www.logikk.com/articles/machine-learning-in-game-development/>. Acesso em: 12 out. 2020.

UNITY. **Plataforma de desenvolvimento em tempo real do Unity3D**. Disponível em <https://unity.com/>. Acesso em: 15 mar. 2021.