UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
CAMPUS DE BAURU
FACULDADE DE CIÊNCIAS
Bacharelado em Sistemas de Informação

FELIPE SANT'ANA CORRÊA
PAULO HENRIQUE COSTA MIRANDA

APRENDIZADO POR REFORÇO APLICADO EM JOGO DE FPS

Bauru
2023

FELIPE SANT'ANA CORRÊA
PAULO HENRIQUE COSTA MIRANDA

APRENDIZADO POR REFORÇO APLICADO EM JOGO DE FPS

Trabalho de conclusão de curso apresentado ao curso de Sistemas de Informação da Universidade Estadual Paulista "Júlio de Mesquita Filho", como requisito para obtenção do título de Bacharel em Sistemas de Informação.

Orientadora: Prof. Dra. Simone das Graças Domingues Prado

Bauru
2023

Corrêa, Felipe Sant’Ana.
Aprendizagem por reforço aplicado em jogo de fps/ Felipe Sant’Ana Corrêa, Paulo Henrique Costa Miranda, 2023
34 f. : il.

Orientador: Simone das Graças Domingues Prado

Monografia (Graduação)–Universidade Estadual Paulista (Unesp). Faculdade de Ciências, Bauru, 2023

1. Aprendizagem por Reforço. 2. FPS. 3. IA. I. Universidade Estadual Paulista. Faculdade de Ciências. II. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Faculdade de Ciências, Bauru. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

FELIPE SANT'ANA CORRÊA
PAULO HENRIQUE COSTA MIRANDA

APRENDIZADO POR REFORÇO APLICADO EM JOGO DE FPS

Trabalho de conclusão de curso apresentado ao curso de Sistemas de Informação da Universidade Estadual Paulista "Júlio de Mesquita Filho", como requisito para obtenção do título de Bacharel em Sistemas de Informação.

Banca Examinadora

Prof.ª Dr.ª Simone das Graças Domingues Prado
Orientadora
Departamento de Computação
Faculdade de Ciências
Universidade Estadual Paulista "Júlio de Mesquita Filho"

Prof.ª Dr.ª Márcia Aparecida Zanoli Meira e Silva
Departamento de Computação
Faculdade de Ciências
Universidade Estadual Paulista "Júlio de Mesquita Filho"

Prof.º Dr.º José Remo Ferreira Brega
Departamento de Computação
Faculdade de Ciências
Universidade Estadual Paulista "Júlio de Mesquita Filho"

Bauru, 09 de Fevereiro de 2023.

RESUMO

O já estabelecido mercado de jogos eletrônicos proporciona a entrada de desenvolvedores independentes com mais facilidade na indústria e impulsiona áreas de estudo muito importantes como é o caso do *Machine Learning*, desse modo, este trabalho, através da técnica conhecida como aprendizado por reforço, apresenta uma inteligência artificial para um jogo FPS construído na ferramenta Unity, bem como disponibiliza o ambiente para desenvolvedores indies e pesquisadores de inteligência artificial.

Palavras chaves: Inteligência artificial. jogos. FPS. Machine Learning. Aprendizado por Reforço.

ABSTRACT

The already established electronic games market provides the entry of independent developers more easily in the industry and boosted very important areas of study such as Machine Learning, thus, this work, through the technique known as reinforcement learning, presents a artificial intelligence for an FPS game built on the Unity tool, as well as providing the environment for indie developers and artificial intelligence researchers.

Keywords: Artificial intelligence. games. FPS. Machine Learning. Reinforcement Learning.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

FSM	Finite State Machine
FPS	First Person Shooter
IA	Inteligência Artificial
NPCs	Non-playable Characters
RL	Reinforcement Learning

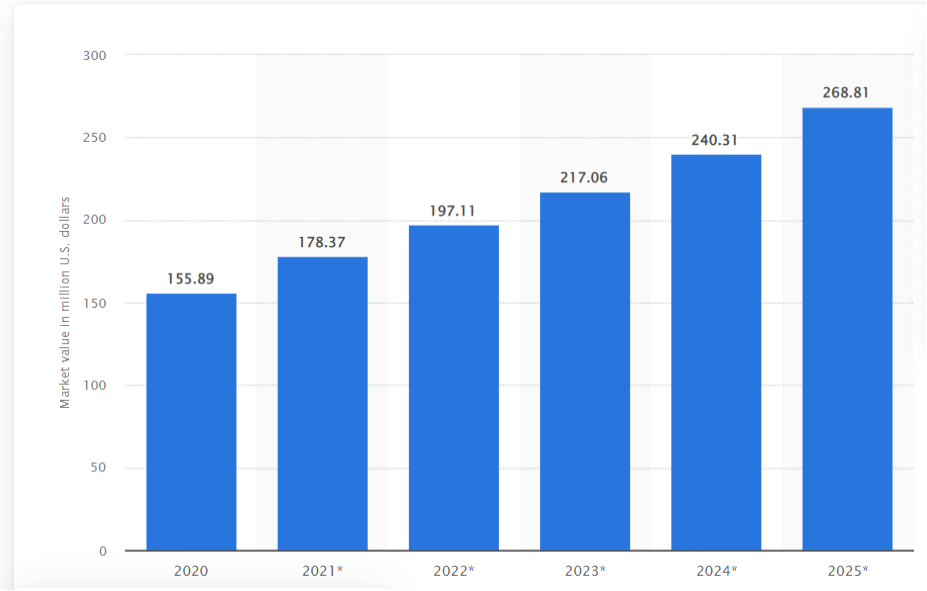
SUMÁRIO

1 INTRODUÇÃO

Com o constante crescimento da popularidade dos jogos eletrônicos e da indústria de videogames em geral, a visão que se tem sobre esse meio de entretenimento muda corriqueiramente. Os jogos hoje abrangem diversas áreas como educação, simuladores, saúde (fisioterapia, exercícios físicos) e áreas afins, e consequentemente desafiam barreiras antes postas sobre eles, como a questão da faixa etária, vício, danos à saúde entre outras.

“Estima-se que o mercado global de jogos chegue a 268,8 bilhões de dólares anualmente em 2025, acima dos 178 bilhões de dólares americanos em 2021” (CLEMENT, 2021, tradução nossa). Além disso, conforme as ferramentas de software de desenvolvimento de jogos se tornaram mais acessíveis, robustas, atrativas e populares, a entrada de novos desenvolvedores e estúdios, principalmente os independentes, cresce cada vez mais nessa indústria já consolidada. A Figura 1 demonstra a previsão de crescimento desse mercado em cinco anos com base no seu valor em 2020:

Figura 1 - Valor de mercado global de videogames de 2020 a 2025



Fonte: Statista (2021)

O crescimento e sofisticação dos jogos eletrônicos permitiram avanços

simultâneos no campo da inteligência artificial. Desde o princípio dos jogos os chamados NPCs (*Non-playable Characters*) são concebidos com linhas de código que buscam fazê-los executar um comportamento almejado, em muitos casos simular o comportamento humano dentro do jogo, sejam eles inimigos ou aliados, ou simular um ser humano jogando.

Dessa maneira, o conceito, a programação, os métodos de produção, o design, as funcionalidades, os objetivos e as recompensas dos jogadores, o fluxo de aprendizado e principalmente o tempo de desenvolvimento de jogos *indie* tendem a ser mais trabalhosos por esse motivo.

Jogos *indie (*independentes) são aqueles desenvolvidos por uma pequena equipe ou até mesmo por uma única pessoa, normalmente não possui apoio financeiro de uma empresa externa e sua distribuição também é de responsabilidade dos próprios desenvolvedores. Algumas empresas possuem programas para desenvolvedores independentes ajudando financeiramente e com a distribuição, mesmo assim, o orçamento desse tipo de jogo é muito inferior em relação às grandes produções da indústria.

Os jogos de tiro em primeira pessoa, ou *First-Person Shooter* (FPS), são um subgênero de videogames de tiro que focam em armas e outros combates baseados nelas em uma perspectiva em primeira pessoa, onde o jogador observa a ação através dos olhos do personagem e o controla em um espaço tridimensional. O gênero compartilha características com outros jogos de tiro, e pertence ao gênero de jogos de ação. Gráficos 3D e pseudo-3D avançados desafiaram o desenvolvimento de hardware desde que o gênero surgiu, e os jogos *multiplayer* se tornaram uma parte integral dele. A programação da Inteligência Artificial dos NPCs, seja o jogo *Single Player* ou *Multiplayer* (nesse caso popularmente conhecidos como *bots*)*,* é componente fundamental para o funcionamento do mesmo, demandando tempo e complexidade de código, ou seja, o grau de inteligência da I.A., as ações disponíveis para ela escolher e a medição da qualidade de sua decisão.

Facilitar o desenvolvimento do comportamento dos NPCs bem como proporcionar um ambiente para teste de seus algoritmos de forma gratuita é de grande ajuda na trajetória de produção de projetos independentes.

Desse modo, este trabalho apresenta um ambiente desenvolvido na

plataforma Unity, uma plataforma de desenvolvimento de jogos, para treinamento do comportamento NPCs em um jogo do gênero FPS, buscando acelerar o desenvolvimento de desenvolvedores e/ou servindo como um ambiente protótipo para projetos iniciantes.

Este trabalho tem como objetivo desenvolver uma inteligência artificial capaz de simular o desempenho de um jogador humano em jogos do gênero *First-Person Shooter*, especificamente para o modo *Team Deathmatch*, através do método de *Machine Learning* denominado Aprendizado por Reforço. Alguns objetivos específicos são:

- Criar uma inteligência artificial capaz de aprender com seus erros e acertos;
- Criar uma inteligência artificial que se adapta a situações similares à que foi treinada previamente, de acordo com a experiência adquirida; e
- Disponibilizar o ambiente criado para servir como base em projetos de desenvolvedores independentes.

No capítulo 2 serão abordadas as soluções disponíveis atualmente que se assemelham ao projeto proposto. O capítulo 3 descreve as tecnologias utilizadas para a elaboração do trabalho. Mais a frente, no capítulo 4, será descrito o desenvolvimento do projeto, desde as características do ambiente, a implantação da aprendizagem por reforço, o método de treinamento dos agentes até detalhes do ambiente no *Unity.* Já no capítulo 5 são apresentados os métodos de validação. Por fim, no capítulo 6 demonstra-se os resultados obtidos, sugestões de aprimoramento e possíveis usos futuros do trabalho.

2 SOLUÇÕES EXISTENTES

Como parte desse projeto, procurou-se soluções disponíveis na *Unity Asset Store*, que é um site da própria Unity para compra de *assets* (basicamente gráficos prontos para um jogo), para atender às necessidades de IA em um jogo de tiro em primeira pessoa. Após uma busca, três soluções se encaixam como soluções porém apresentam alguns problemas nos quais o projeto possa se sobressair ou possuem um preço alto para um desenvolvedor iniciante que são: *Enemy AI*, *Zombie AI System* e *Deathmatch AI Kit*.

2.1 Enemy AI

Essa solução atualmente se encontra disponível na Unity Asset Store por $39.99 dólares, desenvolvida por Vinicius Marques.

De acordo com sua descrição na Unity Asset Store feita pelo seu criador, conta com NPCs humanóides inimigos prontos para usar, com um sistema completo de IA FSM com ações de patrulha, busca e engajamento (com tiro, recarga, cobertura, entre outras ações). Também fornece uma configuração rápida para qualquer modelo personalizado de avatar humanoide.

2.2 Zombie AI System

Essa solução atualmente se encontra disponível na Unity Asset Store por $100 dólares, desenvolvida por Remesh Games. Permite a criação e customização de uma inteligência artificial que replicará o movimento de um Zumbi.

Segundo sua descrição, o Zombie AI System permite criar inteligência artificial para os zumbis em seus jogos, você pode arrastar e soltar rápida e facilmente em seu projeto e também permite alterar facilmente as propriedades do NPC como (estado de movimento, ativação de ataque , Danos, Saúde e muito mais), este *asset* é compatível com qualquer projeto *Single Player*. Foi destacada pois pode exercer a função de patrulha em busca de alvos.

2.3 Deathmatch AI Kit

Essa solução atualmente se encontra disponível na Unity Asset Store por $30

dólares, desenvolvida por Opsive. Permite a criação e customização de IA para criar jogos de *Deathmatch* usando *Behavior Designer* e *Ultimate Character Controller*, outros pacotes do Unity.

Segundo seu desenvolvedor na Unity Asset Store, o Deathmatch AI Kit inclui um conjunto de tarefas de árvore de comportamento com um objetivo: eliminar todos os alvos inimigos. Para atingir esse objetivo, os agentes se movem em formações, se protegem, determinam a melhor arma e atacam estrategicamente o alvo.

Mesmo sendo um solução muito boa e versátil, com alta possibilidade de customização e IAs extremamente inteligentes direto da caixa, há o problema de depender de três outros pacotes do unity, sendo eles *Behavior Designer, Ultimate Character Controller* e *Movement Pack*. Embora sejam pacotes muito bons e ajudam na praticidade e facilidade na criação do projeto eles elevam demasiadamente o preço para uso da *Deathmatch AI Kit*, sendo necessário pagar mais $330 apenas para o pacote poder ser utilizado, ou seja acaba sendo um grande investimento para desenvolvedores iniciantes que estão, por exemplo, criando seu primeiro jogo.

2.4 Comparação

Comparando as soluções apresentadas observa-se que:

- Enemy AI é uma solução com um preço alto mas possui uma variedade de ações de IA prontas para usar. Ele também fornece uma configuração rápida para qualquer modelo personalizado de avatar humanoide.
- Zombie AI System é uma solução ainda mais cara, mas permite a criação e customização de uma inteligência artificial que implicará o movimento de um zumbi em um jogo. Ele é compatível apenas com jogos *Single Player*.
- Deathmatch AI Kit é uma solução com um custo um pouco menor, mas depende de outros pacotes do Unity para funcionar. Ele inclui um conjunto de tarefas de árvore de comportamento e oferece uma alta possibilidade de customização e IA inteligente. No entanto, o custo adicional dos pacotes necessários pode ser proibitivo para alguns

desenvolvedores iniciantes.

Considerando as opções apresentadas, o preço e a necessidade de outros pacotes, a solução Enemy AI pode ser a melhor opção para desenvolvedores iniciantes, pois é a mais acessível e oferece recursos suficientes para atender às suas necessidades mas ainda possui um preço alto de entrada, principalmente para desenvolvedores brasileiros por seu preço ser em dólar.

3 TECNOLOGIAS UTILIZADAS

3.1 Unity

Unity é um motor de jogo amplamente utilizado que foi selecionado para o desenvolvimento desse projeto. A principal razão para sua seleção é sua flexibilidade e facilidade de uso. O editor visual fornecido pelo Unity permite que os desenvolvedores criem e manipulem facilmente objetos, níveis e outros elementos do jogo em uma interface visual. Isso torna mais fácil para os desenvolvedores ver como o jogo vai parecer e sentir enquanto trabalham e permitem fazer alterações e ajustes em tempo real. Além disso, o Unity permite que os desenvolvedores escrevam código em C#, UnityScript ou Boo para adicionar funcionalidade e comportamento aos objetos do jogo.

O Unity também fornece uma ampla variedade de ferramentas de depuração que podem ser usadas para identificar e corrigir problemas de desempenho no jogo. O Unity Profiler, por exemplo, fornece informações detalhadas sobre o desempenho do jogo, incluindo informações sobre uso de memória, uso de CPU e uso de GPU, que podem ser usadas para identificar e corrigir gargalos de desempenho.

O mecanismo de física integrado, iluminação e efeitos especiais e suporte para uma ampla variedade de plataformas, como Windows, Mac, Linux, iOS e Android, também são recursos importantes que foram considerados na seleção do Unity para este projeto. Esses recursos integrados podem ajudar os desenvolvedores a criar experiências mais envolventes. O mecanismo de física, por exemplo, permite que eles criem ambientes realistas e verossímeis, enquanto a iluminação e os efeitos especiais podem ser usados para aprimorar o apelo visual do jogo.

Além disso, a comunidade grande e ativa do Unity fornece um recurso valioso para desenvolvedores, oferecendo suporte, recursos e *plugins* de terceiros que podem ser usados para adicionar funcionalidades e recursos adicionais. Essa comunidade é uma grande fonte de conhecimento e experiência, pois fornece o suporte e os recursos necessários para criar jogos e experiências interativas de alta qualidade. Além disso, o Unity também fornece uma ampla variedade de *plugins* e ativos de terceiros que podem ser usados para adicionar funcionalidades e recursos adicionais ao jogo.

No geral, os recursos e capacidades do Unity o tornam a escolha ideal para o projeto. Seu editor visual, API de *script*, suporte multiplataforma, recursos integrados e comunidade ativa fornecem aos desenvolvedores as ferramentas e os recursos necessários para criar experiências envolventes e interativas enquanto implementam este trabalho. Isso torna o Unity um mecanismo de jogo popular e amplamente usado na indústria, sendo uma ótima opção para desenvolvedores de jogos experientes e novatos.

3.2 Aprendizagem por Reforço

Reinforcement Learning (RL) é um tipo de aprendizado de máquina que se concentra no treinamento de agentes para tomar decisões em um ambiente, aprendendo com suas ações, recompensas ou penalidades. O objetivo do agente é aprender uma política que maximize a recompensa cumulativa esperada ao longo do tempo. A RL é particularmente útil em situações em que a solução ótima não é conhecida de antemão e deve ser aprendida por tentativa e erro.

Na RL, um agente interage com um ambiente realizando ações e recebendo recompensas ou penalidades. O objetivo do agente é aprender uma política, que é um mapeamento de estados para ações, que maximize a recompensa cumulativa esperada. Ele usa essa política para decidir quais ações realizar em diferentes estados. O desempenho é avaliado por meio de um valor escalar denominado retorno, que é a soma das recompensas obtidas em um determinado estado ou intervalo de tempo.

Para que um agente aprenda a política ótima, ele precisa ser capaz de atribuir um valor às suas ações com base nas recompensas que recebe. Na RL, as recompensas são usadas para sinalizá-lo se uma determinada ação é boa ou ruim. O agente aprende a associar um valor mais alto a ações que levam a recompensas mais altas e um valor mais baixo a ações que levam a recompensas mais baixas. Esse processo é conhecido como aprendizagem baseada em recompensas.

As recompensas em RL podem vir de diferentes formas, como um valor escalar fixo para cada ação ou uma função mais complexa do estado e da ação. A função de recompensa é normalmente projetada pelo desenvolvedor para guiar o

processo de aprendizado do agente em direção ao comportamento desejado. Por exemplo, em um jogo FPS, o agente pode ser recompensado por acertar o alvo e penalizado por errar o alvo.

No contexto do Unity, o projeto ajudará desenvolvedores independentes a treinar IA para seu jogo FPS. O recurso foi projetado para ser simples e fácil de usar, tornando-o acessível para desenvolvedores independentes com experiência limitada em programação. O recurso pode fornecer agentes e ambientes pré-construídos, permitindo que os desenvolvedores treinem personagens de IA de forma rápida e fácil para seus jogos. Isso pode ajudar a economizar tempo e recursos para desenvolvedores independentes, que podem ter recursos limitados para se dedicar ao desenvolvimento de IA.

Outra vantagem de usar RL no Unity é a capacidade de treinar os personagens de IA para tomar decisões com base no estado atual do jogo e nas ações do jogador. Isso permite que os personagens de IA respondam em tempo real às ações do jogador, tornando o jogo mais dinâmico e envolvente.

3.3 ML-Agents

The Unity Machine Learning Agents Toolkit (*ML-Agents*) é um *plugin Unity* desenvolvido pela *Unity Technologies* que permite aos desenvolvedores treinar agentes inteligentes usando aprendizado por reforço. Ele fornece uma ampla gama de agentes, ambientes e ferramentas pré-construídas que podem ser usados para treinar modelos para realização de diferentes tarefas.

Uma das principais vantagens de usar *ML-Agents* é a capacidade de treinar agentes em um ambiente simulado, sem a necessidade de hardware caro ou experimentação no mundo real. Outra vantagem é a possibilidade de utilização de diferentes algoritmos de RL. Isso permite que os desenvolvedores escolham o algoritmo mais apropriado para sua tarefa e comparem seus desempenhos.

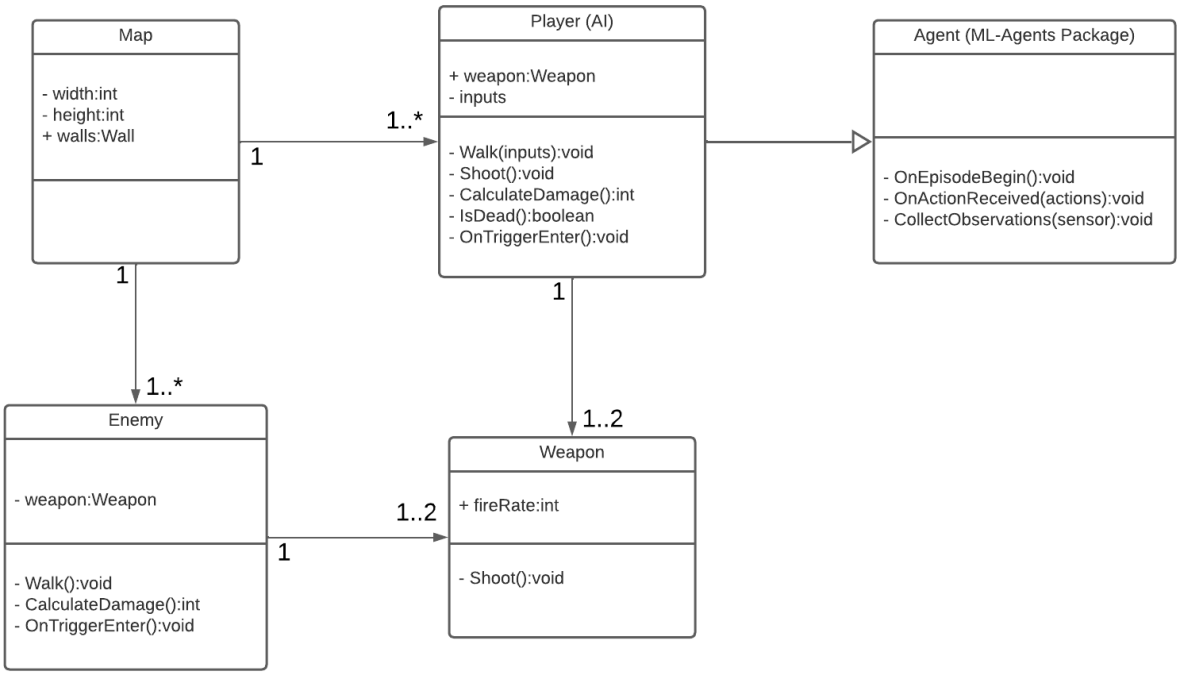
Além disso, o plugin fornece ferramentas e recursos que podem ser usados para visualizar e analisar o processo de treinamento, contribuindo com a análise do comportamento do agente e a identificação e correção de problemas.

4 DESCRIÇÃO DO PROJETO

O projeto foi desenvolvido considerando um ambiente jogável construído com base nos fundamentos dos jogos do gênero FPS, sendo composto inicialmente por uma área quadrangular, um personagem controlável com equipamento para realizar disparos e inimigos que servem como alvo. O mesmo pode ser encontrado neste repositório: github.com/Felipe-Correa06/FPS_Project.

Na sequência inicia-se a implementação do Agente inteligente, onde os *inputs* que seriam feitos pelo jogador para controlar o personagem são substituídos pelas decisões tomadas a partir das observações do estado do ambiente feitas pelo agente através de seu algoritmo de treinamento. A Figura 2 mostra um diagrama de classes com várias interações entre os atores do ambiente.

Figura 2 - Diagrama de Classes

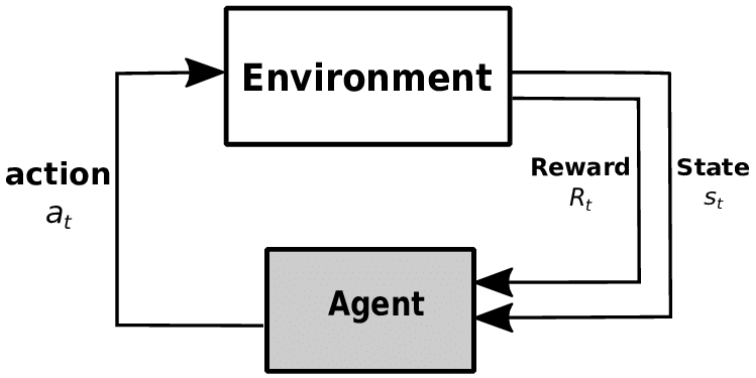


Fonte: Elaborado pelos autores (2022)

O treinamento da I.A. foi realizado a partir da aprendizagem por reforço, baseado na ideia de dar uma “Recompensa” quando a mesma realiza uma ação desejada para que ela a refaça de forma cada vez mais eficiente maximizando esse

valor. A Figura 3 contém uma exemplificação desse conceito.

Figura 3 - Lógica do Aprendizado por Reforço



Fonte: ResearchGate[1] (2018).

A Figura 3 representa um *Step,* ou seja, um ciclo e seus elementos, sendo:

- *Action*: a ação escolhida pelo Agente
- *Reward*: a recompensa fornecida pelo ambiente ao agente por sua última ação realizada, podendo ser positiva ou negativa.
- *State*: o estado em que o ambiente se encontra após ser influenciado/modificado pela ação do agente.

Esse treinamento continua dessa forma até um desempenho satisfatório nos dados de saída, ou seja, uma pontuação alta e um tempo relativamente pequeno de eliminação dos inimigos. A aprendizagem por reforço possui mais alguns fundamentos importantes, tais como:

- *Observation Space*: Todos os elementos que compõem o ambiente e são definidos como “observáveis”, ou seja, elementos que serão transformados em dados para o Agente. Neste trabalho os elementos observáveis são: a posição em que o Agente se encontra e tudo o que ele “enxerga” a partir de um sensor implementado nele, ou seja,

[1]A Machine Learning Approach for Power Allocation in HetNets Considering QoS. ResearchGate, 2018. Disponível em: https://www.researchgate.net/publication/323867253_A_Machine_Learning_Approach_for_Power_Allocation_in_HetNets_Considering_QoS. Acesso em: 06, jun. 2022.

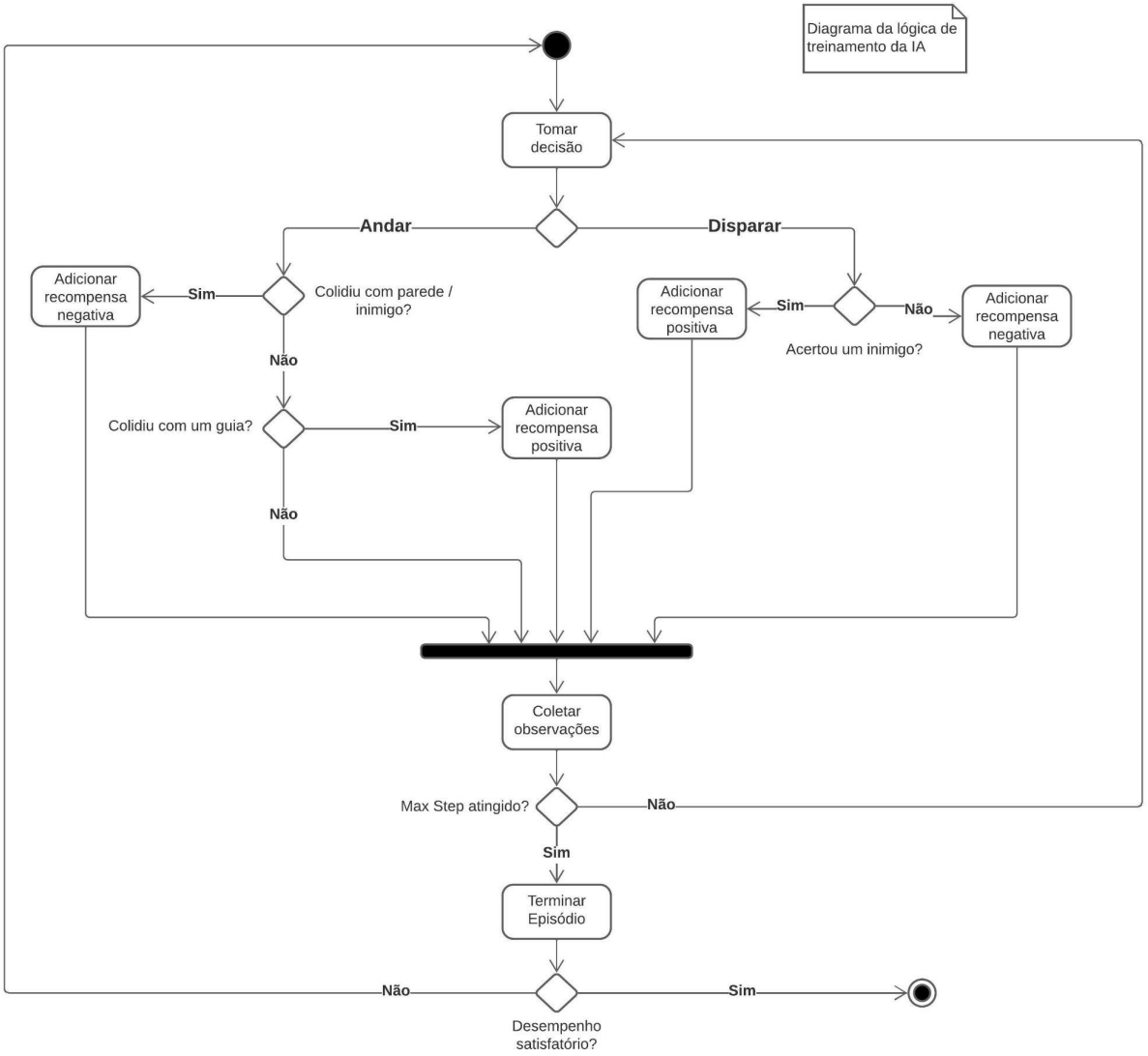
posição de paredes, inimigos e guias.

- *Action Space*: Todas as ações disponíveis para serem executadas pelo Agente a cada *Step*. As ações disponibilizadas ao Agente neste trabalho são: três vetores de valores contínuos sendo um para a movimentação do personagem no eixo x, outro para a movimentação no eixo z e um terceiro para a movimentação da câmera no eixo x e um vetor de valores discretos para controle do disparo (0 ou 1).
- *Max Step*: A definição de um número *n* representando a quantidade de *Steps* necessários para completar um *Episode*.
- *Episode*: Um ciclo completo de treinamento. Normalmente são necessários vários *Episodes* para que o Agente atinja um desempenho satisfatório.

O projeto possui um sistema de *tags* específicas que são usadas pelo sensor para identificar os elementos do ambiente, sendo elas: a *tag* "*enemy"* identifica inimigos indicando que o Agente deve atirar, a *tag "wall"* indica uma parede que deve ser evitada e bloqueia a visão da I.A. e, por fim, a *tag "guide"* aponta um objeto que ela deve colidir.

O ciclo de treinamento para o agente utilizado neste projeto está especificado na Figura 4. Nela é possível ver todas as ações disponíveis ao Agente e os resultados dessas ações. Por exemplo, seguindo o fluxo do diagrama em que o Agente decide realizar a ação de andar, é verificado se ele colidiu com uma parede, se sim, é atribuída uma recompensa negativa, caso contrário, é verificado se ele colidiu com um guia, se sim é atribuída uma recompensa positiva, se não significa que ele não colidiu em nada e não recebe nenhuma recompensa, pois se ele recebesse uma recompensa negativa por não colidir com nada em determinado momento ele ficaria apenas parado, do mesmo modo se a recompensa fosse positiva ele andaria em círculos.

Figura 4 - Diagrama de lógica de treinamento da I.A.



Fonte: Elaborado pelos autores (2023).

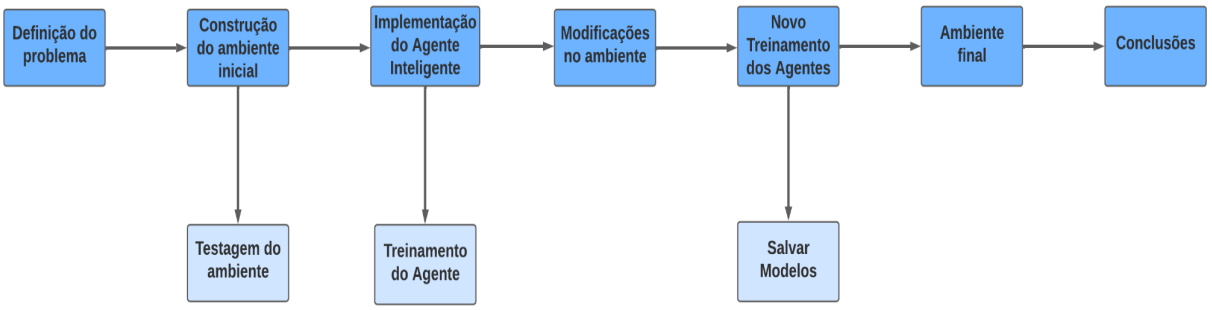
Dando continuidade ao trabalho foram feitas alterações no ambiente, como adição de guias para auxiliar na movimentação do Agente, paredes formando “caminhos” distintos e alteração no número de inimigos (alvos), como realizado anteriormente, submeteu-se o Agente ao treinamento e sua performance foi analisada fazendo ajustes necessários às funções de recompensa.

Dando continuidade ao trabalho foram feitas alterações no ambiente, como adição de guias para auxiliar na movimentação do Agente, paredes formando “caminhos” distintos e alteração no número de inimigos (alvos), como realizado anteriormente, submeteu-se o Agente ao treinamento e sua performance foi

analisada fazendo ajustes necessários às funções de recompensa.

Por fim, testou-se o desempenho final da inteligência artificial colocando um número *n* de agentes se enfrentando. Todos os processos descritos anteriormente foram executados em software em desktop. O diagrama de blocos da Figura 5 representa a estrutura do projeto:

Figura 5 - Diagrama de Blocos



Fonte: Elaborado pelos autores (2023).

Na imagem acima está representado em cada bloco:

- Definição do problema: É definido o problema que o projeto buscou resolver;
- Construção do ambiente Inicial: Desenvolvimento do primeiro mapa para a movimentação dos personagens controláveis;
- Testagem do ambiente: Testagem manual do ambiente, como movimentação, lógica de colisões e disparos;
- Implementação do Agente Inteligente: Inserção do Agente inteligente na arena criada e testes de suas ações.
- Treinamento do agente: Início do treinamento do Agente com aprendizagem do reforço;
- Modificações do ambiente: Modificações realizadas a partir de observações para acelerar o aprendizado e adicionar complexidade ao treinamento;
- Novo treinamento dos agentes: Continuação do treinamento com as modificações definidas;

- Salvar Modelos: Após o treinamento e a verificação de que o agente chegou a um nível de inteligência satisfatório seu modelo é salvo para poder ser usado fora do treinamento;
- Ambiente Final: Mapa final para observação do modelo em funcionamento.
- Conclusões: Definir as conclusões obtidas no final do projeto.

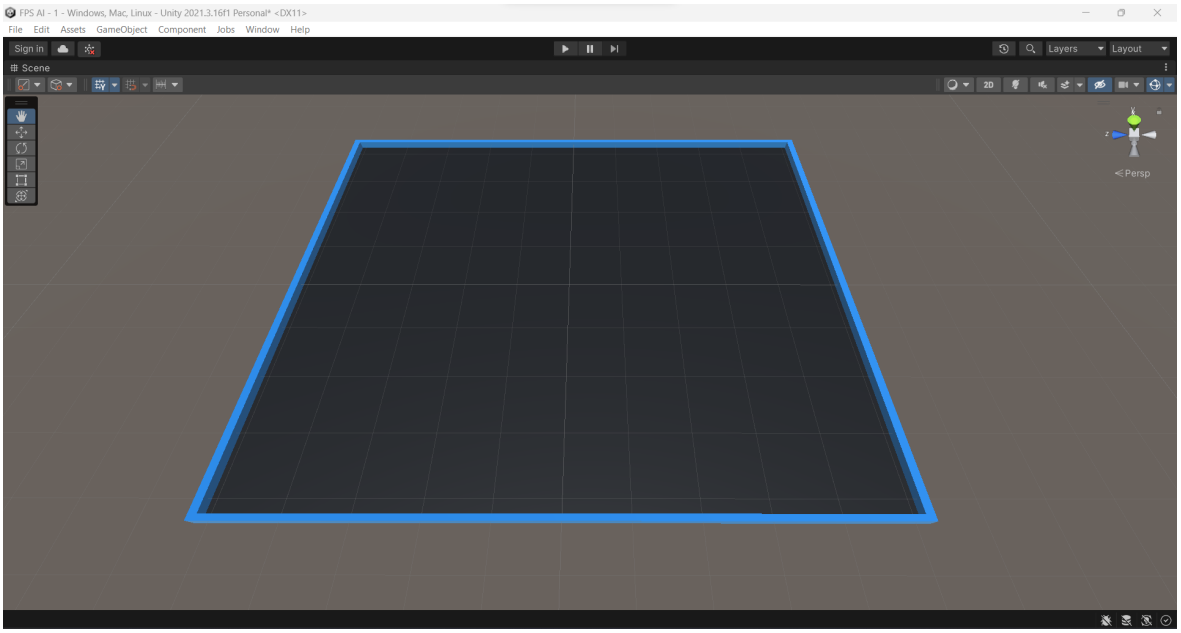
4.1 Detalhamento do projeto na Unity Engine

Devido ao objetivo do trabalho ser o desenvolvimento de uma inteligência artificial foram utilizados apenas *assets* padrões da Unity Engine, desse modo, o projeto pode ser facilmente compreendido e replicado além de ser customizável para um projeto independente.

4.1.1 Mapa/Arena

A Figura 6 mostra o mapa inicial utilizado para os primeiros *episodes* de treinamento em que os *Players* podem se movimentar livremente por toda a extensão da área que se encontra na cor preta, sendo apenas limitados pelas paredes em azul.

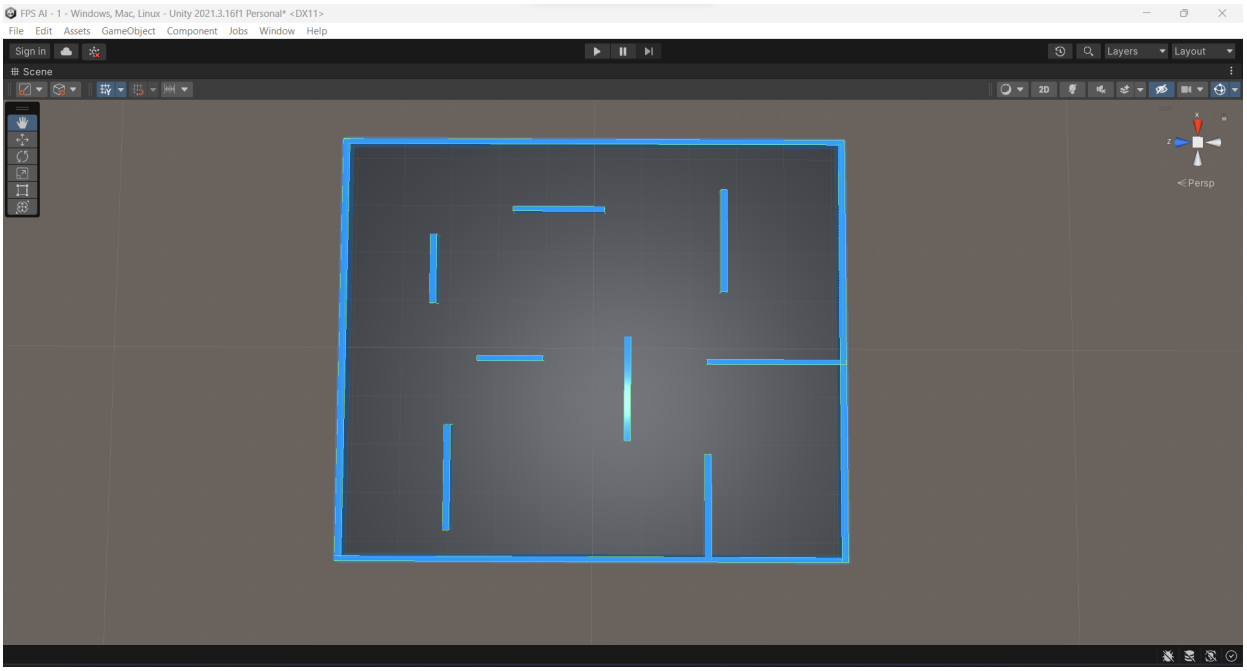
Figura 6 - Mapa/Arena inicial



Fonte: Elaborado pelos autores (2023).

Após certo tempo de treinamento adicionam-se paredes de forma aleatória distribuídas de modo a se parecer como um “labirinto” aumentando a complexidade do ambiente, conforme representado na Figura 7. Diferentes mapas foram criados alterando as posições das paredes para atestar o desempenho da I.A. de modo a evitar que ela adquirisse algum vício de movimentação.

Figura 7 - Mapa/Arena, vista de cima com paredes adicionadas

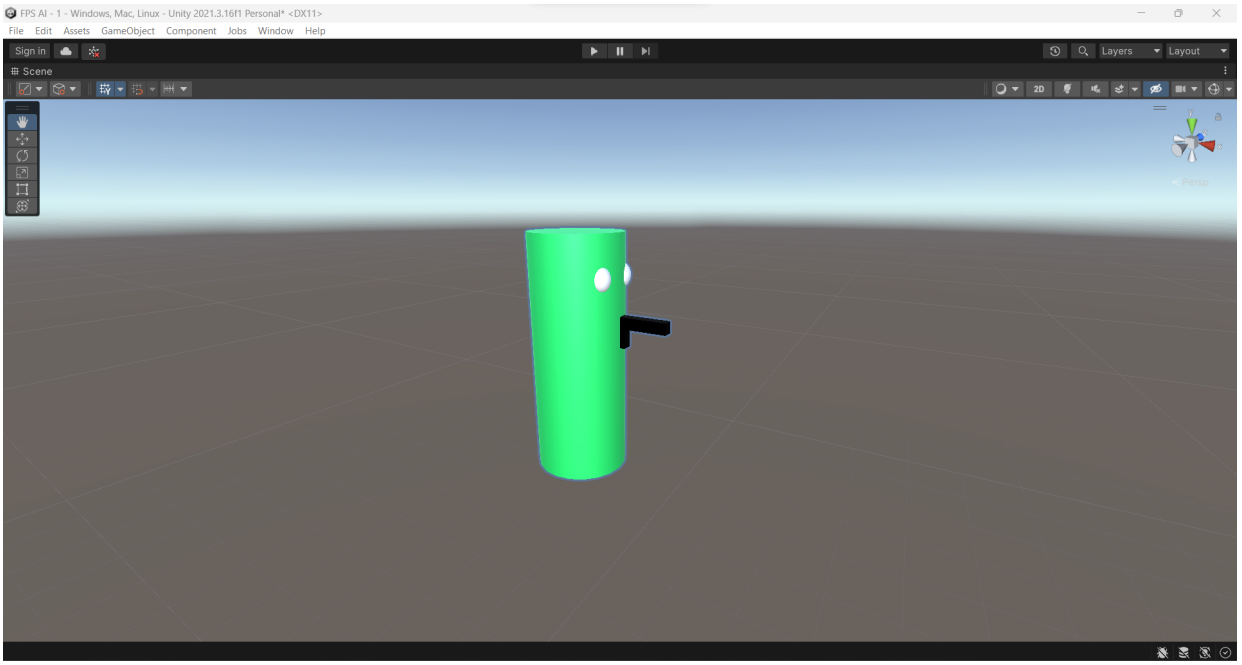


Fonte: Elaborado pelos autores (2023).

4.1.2 *First Person Player*

Na Figura 8 observa-se a representação de um personagem controlável (cilindro verde) por um jogador ou pela I.A., acompanhado de um objeto capaz de realizar disparos (de cor preta). É capaz de movimentar-se livremente por toda a extensão do mapa e pode colidir com paredes, *guides* e *enemies*.

Figura 8 - *First Person Player*

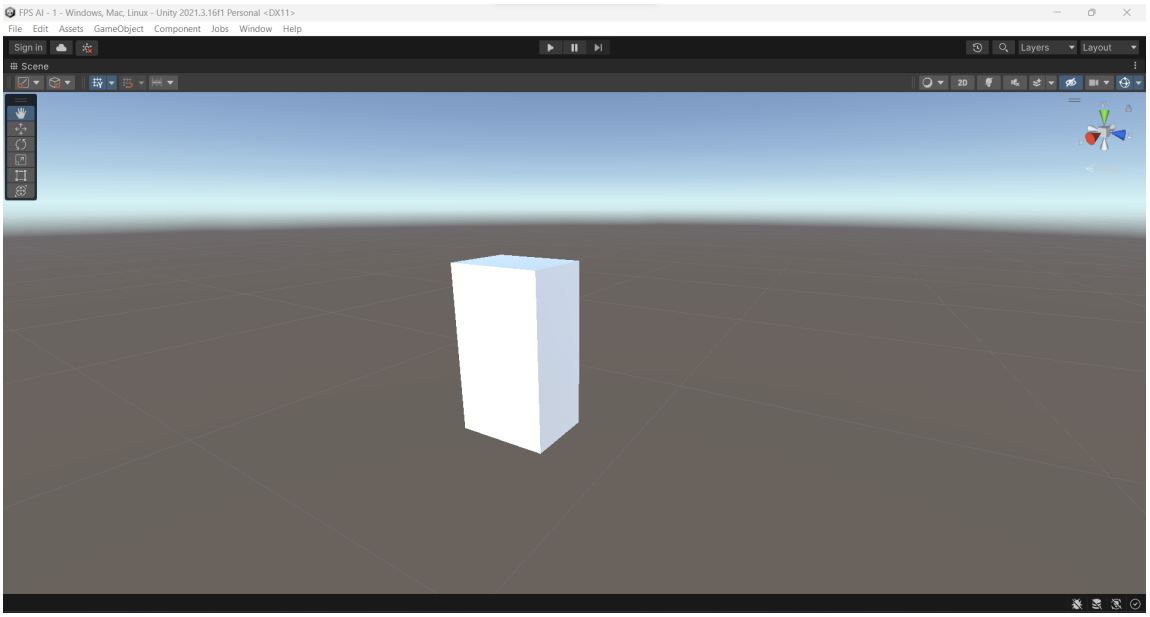


Fonte: Elaborado pelos autores (2023).

4.1.3 *Enemy/Target*

Na Figura 9 observa-se a representação de um inimigo que faz a função de alvo para o treinamento da ação de disparo do Agente, incapaz de realizar disparos e de se movimentar. O agente é recompensado positivamente ao acertar esse objeto através de sua função de disparo.

Figura 9 - *Enemy*

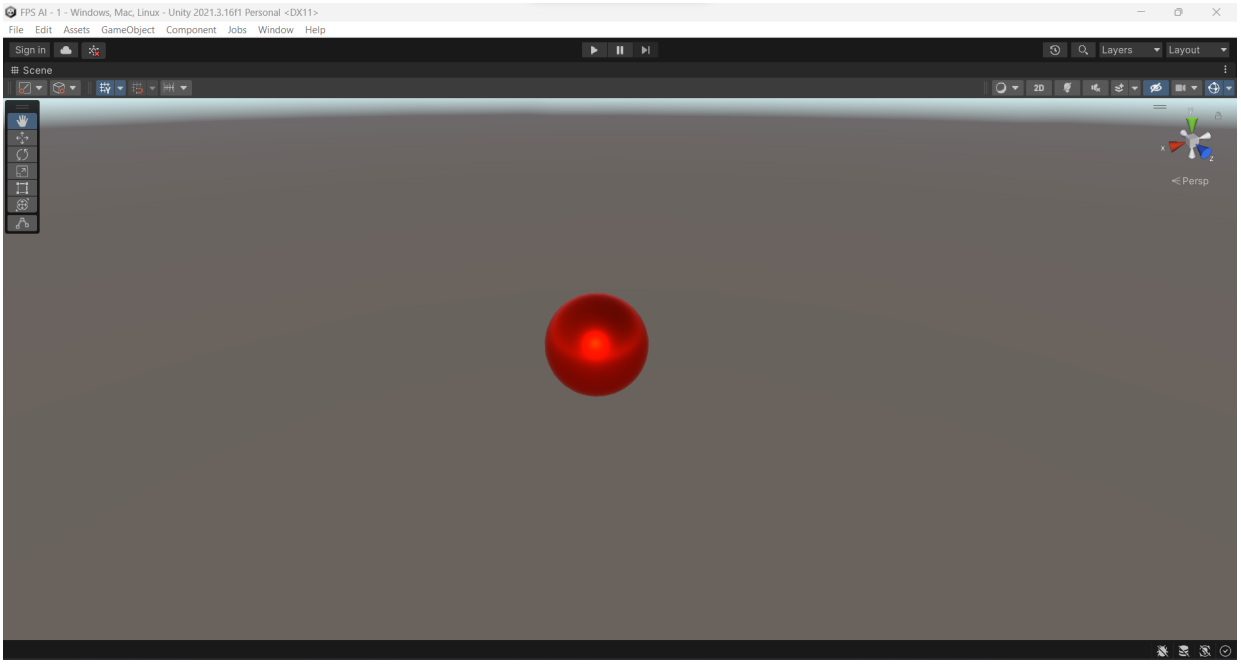


Fonte: Elaborado pelos autores (2023).

4.1.4 *Bullet*

Na Figura 10 observa-se o projétil que é disparado pelo *Player*, o formato e cor foram escolhidos para facilitar a visualização. É capaz de colidir com todos os outros componentes e é destruído quando isso acontece.

Figura 10 - *Bullet*

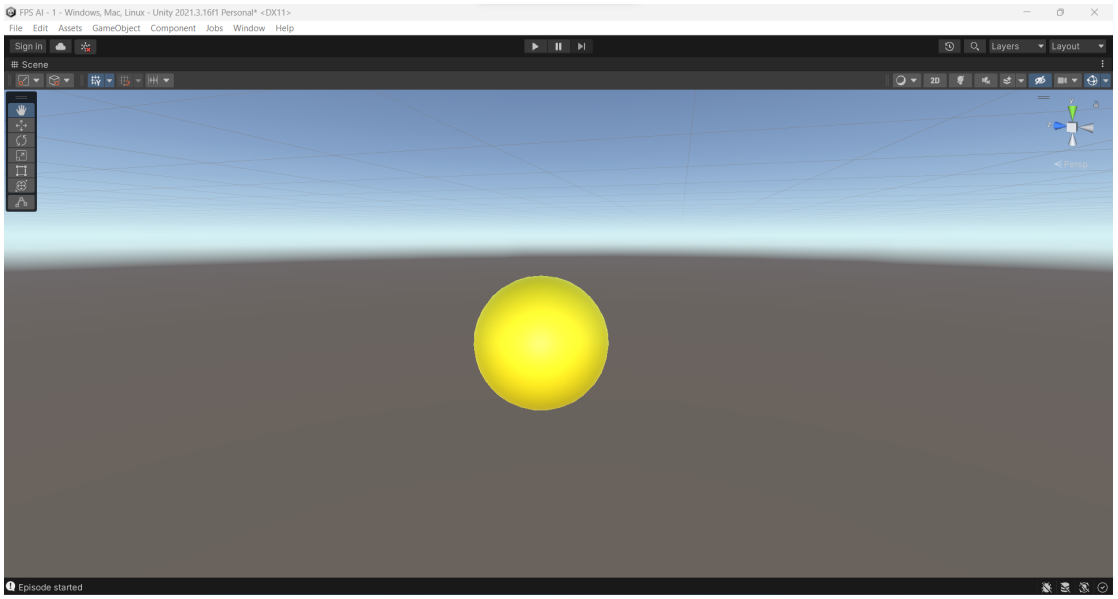


Fonte: Elaborado pelos autores (2023).

4.1.5 *Guide*

Na Figura 11 observa-se o guia utilizado para incentivar a movimentação do Agente pelo mapa. Quando o jogo entra em modo de execução ele se torna visível apenas para o sensor anexado ao *Player* e invisível para a câmera e visão dos jogadores. Quando ocorre uma colisão com o *Player* o guia é destruído e reaparece em um local aleatório do mapa, são necessários vários guias para que o Agente se movimente de uma maneira satisfatória e semelhante a um jogador humano.

Figura 11 - *Guide*

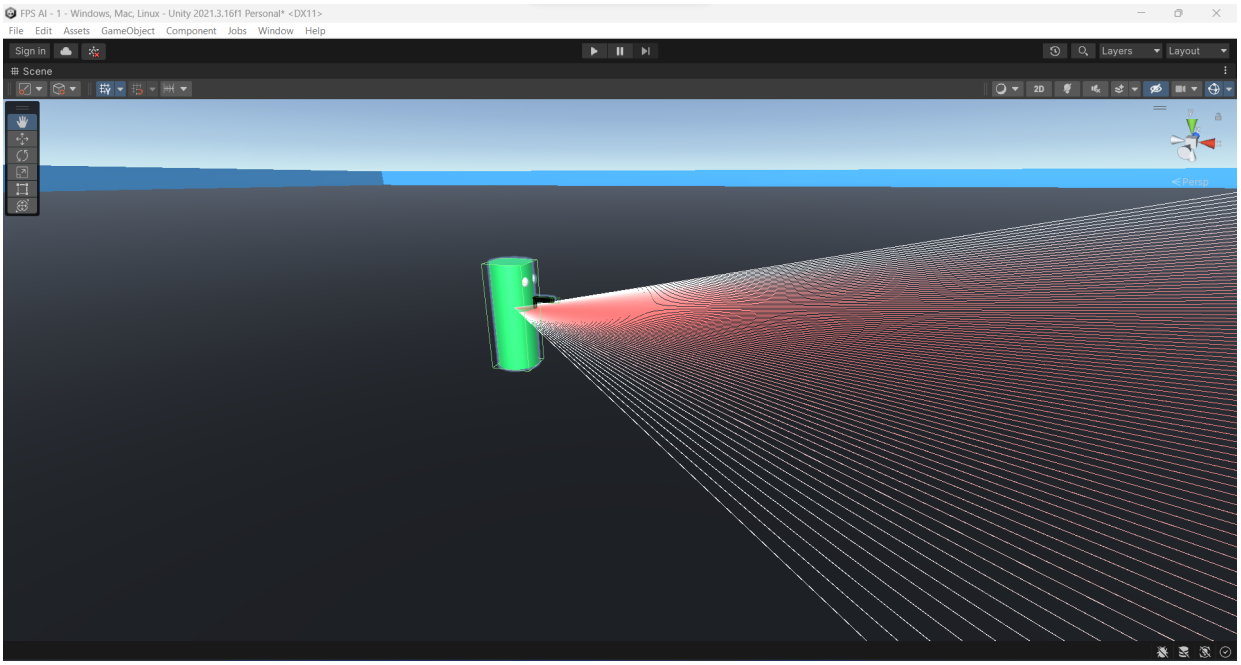


Fonte: Elaborado pelos autores (2023).

4.1.6 *Sensor*

O sensor responsável por coletar observações do ambiente está representado na Figura 12. Ele é composto por diversos *rays* que adquirem a cor vermelha ao colidirem com outro objeto e permanecem na cor branca quando não colidem. Não exerce nenhuma influência na física do jogo.

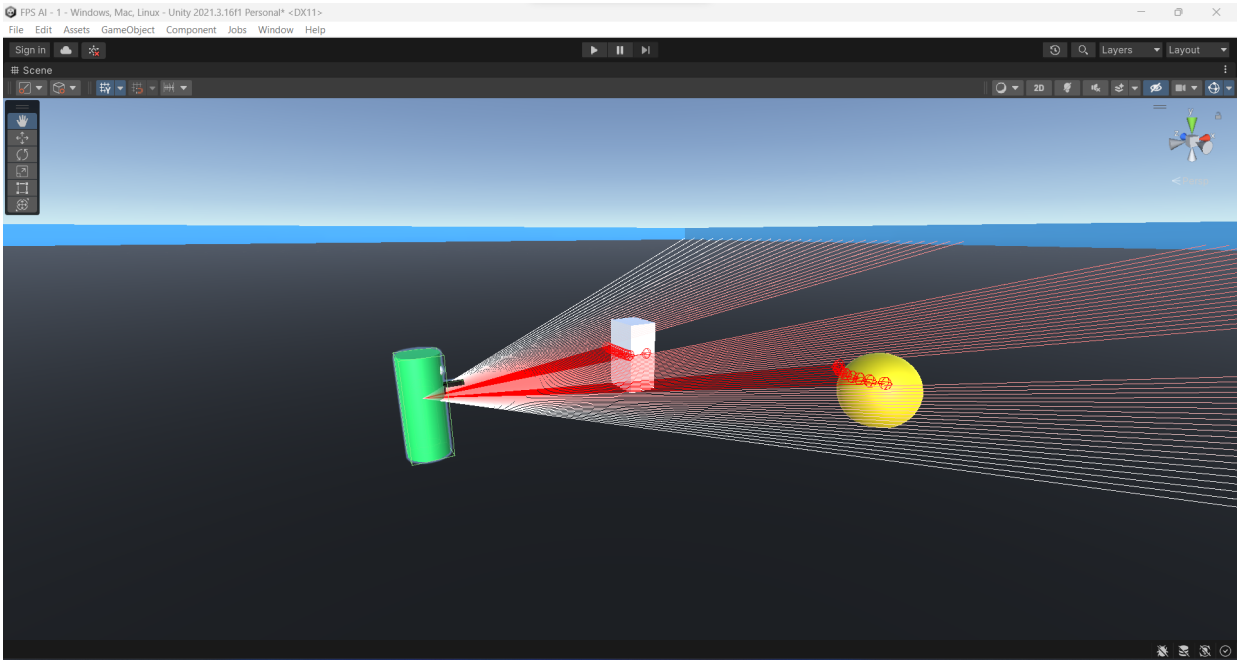
Figura 12 - *Sensor*



Fonte: Elaborado pelos autores (2023).

Observa-se na Figura 13 o sensor colidindo com um *Enemy* e um *Guide.*

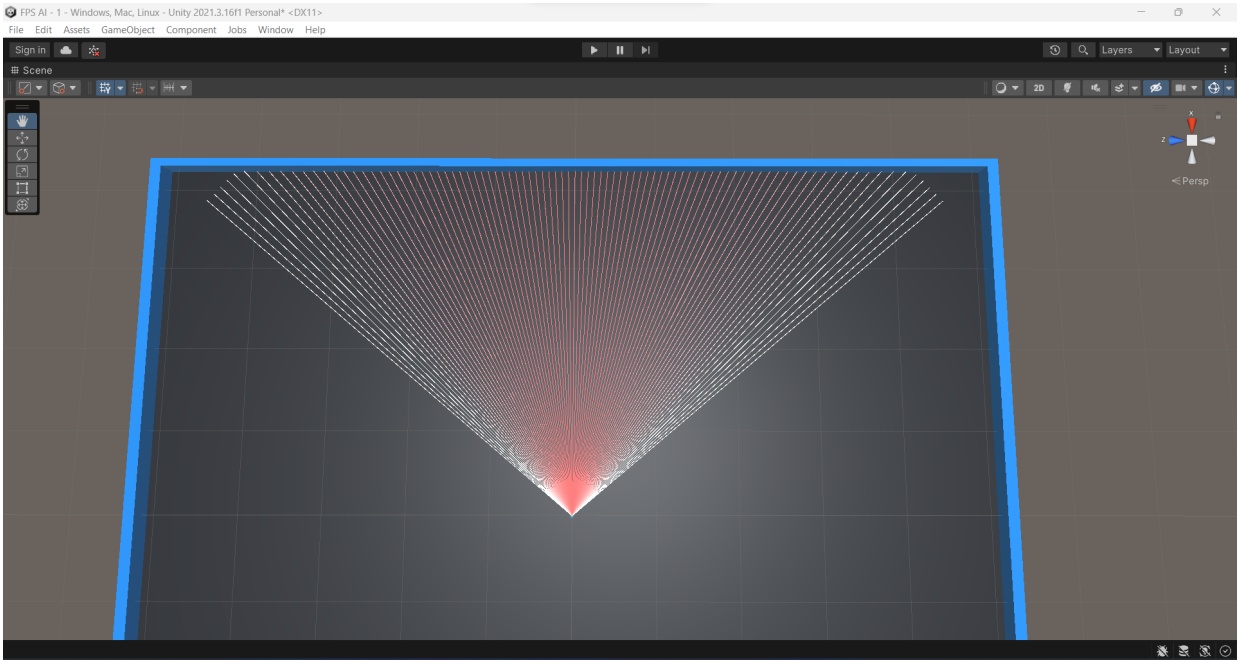
Figura 13 - *Sensor* colidindo com elementos do ambiente



Fonte: Elaborado pelos autores (2023).

O sensor possui 50° e está posicionado no meio do *Player* conforme mostra a Figura 14.

Figura 14 - Visão superior do Sensor



Fonte: Elaborado pelos autores (2023).

5 VALIDAÇÃO DO PROJETO

A validação inicial foi realizada verificando a movimentação do personagem jogável, colisões (entre personagens, de projéteis e paredes), lógica de disparo de projéteis, dentre outros.

Para validar a implementação do agente inteligente foi analisado se durante seu treinamento ele desempenhava um papel satisfatório, ou seja, não ficava “preso” em uma ação específica, o que definiu se as condições de recompensa estavam programadas corretamente.

Após a modificação do ambiente repete-se a validação inicial e da mesma forma acontece para o treinamento dos agentes no ambiente novo.

6 CONCLUSÃO

Visto que a área de jogos eletrônicos possui espaço para desenvolvimento e aprimoramento de muitas tecnologias sendo uma delas o *Machine Learning,* o projeto se mostrou promissor em ajudar a desenvolvedores *indie* tanto como pesquisadores de Inteligência Artificial, proporcionando um ambiente adaptável e simples e ganho de tempo de desenvolvimento.

Além disso, a inteligência artificial pode ser readaptada para qualquer projeto uma vez que todos os componentes principais, que são responsáveis pelo treinamento do Agente para um jogo FPS, já estão implementados, disponibilizados através do GitHub e identificados com clareza no código do trabalho.

A aprendizagem por reforço se mostra uma boa alternativa para a programação de I.A. em jogos mais simples, independentemente do gênero, considerando que, com um ambiente bem montado e regras de treinamento bem definidas, um agente pode ter um desempenho melhor que os métodos tradicionais utilizados atualmente.

Por fim, diante dos resultados obtidos, é possível afirmar que existem diversas possibilidades de trabalhos futuros a serem desenvolvidos com base neste projeto. Alguns exemplos incluem:

- Aplicação do projeto em outros gêneros de jogos: O projeto foi desenvolvido para jogos FPS, mas pode ser adaptado para outros gêneros de jogos, como jogos de estratégia, jogos de aventura, jogos de corrida, entre outros. Isso pode ser realizado através da modificação das regras de treinamento e da configuração do ambiente de jogo.
- Adição de novos componentes ao projeto: O projeto atual inclui componentes básicos para o treinamento de agentes, mas pode ser expandido para incluir outros componentes, como movimentação da mira no eixo vertical, função de recarregamento do dispositivo de disparo, reconhecimento de voz, processamento de linguagem natural, entre outros. Isso pode melhorar ainda mais o desempenho dos agentes e ampliar as possibilidades de interação com os jogadores.
- Teste do Agente em um jogo de FPS simples já existente: replicar um jogo existente dentro do ambiente ou implementar o projeto nele de

modo a analisar o desempenho da inteligência artificial.

- Teste do projeto em diferentes plataformas: O projeto foi desenvolvido e testado em um ambiente de computador, mas pode ser adaptado para outras plataformas, como dispositivos móveis, consoles de jogos, realidade virtual, entre outros.
- Estudo de outros algoritmos de aprendizagem: Este projeto utiliza o algoritmo de aprendizagem por reforço, mas outros algoritmos, como aprendizagem supervisionada, aprendizagem não-supervisionada, aprendizagem por transferência, entre outros, podem ser estudados e testados.

Em resumo, a área de jogos eletrônicos possui muitas possibilidades para desenvolvimentos futuros a serem explorados e o projeto oferece uma série de possibilidades para trabalhos futuros mesmo sendo uma pequena amostra do que pode ser alcançado com essas tecnologias.

REFERÊNCIAS

AMIRI, Roohollah ; MEHRPOUYAN, Hani ; FRIDMAN, Lex ; MALLIK, Ranjan ; NALLANATHAN, Arumugam ; MATOLAK, David . A Machine Learning Approach for Power Allocation in HetNets Considering QoS. ResearchGate, 2018. Disponível em: https://www.researchgate.net/publication/323867253_A_Machine_Learning_Approach_for_Power_Allocation_in_HetNets_Considering_QoS. Acesso em: 06, jun. 2022.

CLEMENT, J. Global video game market value from 2020 to 2025. Statista, 2021. Disponível em: https://www.statista.com/statistics/292056/video-game-market-value-worldwide/. Acesso em: 28, mai. 2022.

CAPÍTULO 62 – O Que é Aprendizagem Por Reforço?. Deep Learning Book, 2022. Disponível em: https://www.deeplearningbook.com.br/o-que-e-aprendizagem-por-reforco/. Acesso em: 30, mai. 2022.

DEATHMATCH AI Kit. Unity Asset Store, 2022. Disponível em: https://assetstore.unity.com/packages/tools/ai/deathmatch-ai-kit-184000. Acesso em: 30, mai. 2022

ENEMY AI. Unity Asset Store, 2022. Disponível em: https://assetstore.unity.com/packages/tools/ai/enemy-ai-163967. Acesso em: 30, mai. 2022.

SINICKI, Adam. What is Unity? Everything you need to know. Android Authority, 2021. Disponível em: https://www.androidauthority.com/what-is-unity-1131558/. Acesso em: 31, mai. 2022

UNITY ML-Agents Toolkit. GitHub, 2022. Disponível em: https://github.com/Unity-Technologies/ml-agents/tree/release_19_branch. Acesso em: 31 mai. 2022

ZOMBIE AI System. Unity Asset Store, 2022. Disponível em: https://assetstore.unity.com/packages/tools/ai/zombie-ai-system-200542. Acesso em: 30 mai. 2022

.