CENTRO PAULA SOUZA

FACULDADE DE TECNOLOGIA DE FRANCA

“Dr. THOMAZ NOVELINO”

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

GABRIEL CESAR CINTRA BORROMEU

RUANN CARLOS MIRANDA GOMES

SMART PURSUER

Uso da Inteligência Artificial em Jogos Digitais

Trabalho de Graduação apresentado à Faculdade de Tecnologia de Franca - “Dr. Thomaz Novelino”, como parte dos requisitos obrigatórios para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientadora: Profa. Dra. Jaqueline Brigladori Pugliesi

FRANCA/SP

2022

SMART PURSUER: USO DA INTELIGÊNCIA ARTIFICIAL EM JOGOS DIGITAIS

Gabriel Cesar Cintra Borromeu[1]

Ruann Carlos Miranda Gomes[2]

Resumo

Este trabalho tem por objetivo apresentar a utilização da Inteligência Artificial dentro do cenário de jogos eletrônicos, criando assim o nosso próprio *software*, para mostrar a diferença para aqueles que não utilizam essa tecnologia e como isso pode trazer inovações e mudanças significativas na maneira como desenvolvemos novos jogos e como planejamos e construímos sua interação com os usuários**.** O contato com a máquina durante o consumo do *software* será diferenciado devido a adaptabilidade dos inimigos durante o jogo, definindo a dificuldade ideal para cada jogador, de acordo com suas habilidades, padrões e conhecimentos, proporcionando assim uma experiência única e emocionante em cada tentativa. O motor gráfico utilizado para este projeto foi a Unity 3D, que também utiliza o C# como linguagem de programação padrão.

Palavras-chave: Experiência. Inteligência Artificial. Jogos. Tecnologia.

Abstract

This paper aims at showing the use of Artificial Intelligence inside the game scenario, creating our own software to show the difference between those that do not use this technology and by what method it can bring innovations and significant changes in the way we develop our games, plan and build the interaction with the users. Their contact with the machine and the software will be unique because of the enemy's adaptability during the game, defining the ideal difficulty for each person according to their abilities, patterns and knowledge, providing a unique and exciting experience for each try. The engine used for this project was Unity 3D that also uses C# as standard programming language.

Keywords: *Experience. Artificial Intelligence. Games. Technology.*

1 Introdução

Os jogos digitais estão cada vez mais presentes em nosso cotidiano, seja em nossos *smartphones*, computadores pessoais ou consoles específicos para jogos, é possível encontrá-los próximos de nós.

[1] Graduando em Análise e Desenvolvimento de Sistemas pela Fatec Dr Thomaz Novelino – Franca/SP. Endereço eletrônico: gabriel.borromeu@fatec.sp.gov.br

[2] Graduando em Análise e Desenvolvimento de Sistemas pela Fatec Dr Thomaz Novelino – Franca/SP. Endereço eletrônico: ruann.gomes@fatec.sp.gov.br

A utilização desse tipo de entretenimento também tem demonstrado resultados positivos na capacidade de aprendizado de crianças, segundo estudos da universidade de Rochester. Nesses estudos, crianças que tem o hábito de jogar, conseguiram realizar tarefas para a detecção de padrões de maneira mais rápida do que aquelas que não possuem o hábito de jogar (ROCHESTER, 2014; INTERNET MATTERS, *sd*).

Outro estudo realizado pela Universidade da Califórnia, mostrou que os jogos também poderiam contribuir na habilidade de realizar várias tarefas ao mesmo tempo, visto a quantidade de escolhas que os jogadores devem fazer durante a *gameplay* (ABBOT, 2013).

Este projeto mostra o desenvolvimento de um jogo com foco no uso da Inteligência Artificial em um sistema de duelo de cavaleiros utilizando espadas, de forma que, cada vez que este é jogado, *Software* e Usuário aprendam mutuamente, tornando-se sempre uma experiência única. Essa diversidade na experiência possibilita um maior tempo de vida do produto, visto que a utilização do software não se tornará tão repetitiva como acontece com outros jogos, em que quando se completa todos os objetivos, existe pouquíssimo estímulo para repetições além do próprio gosto do jogador.

2 Viabilidade do projeto

A viabilidade do projeto foi pensada a partir da criação do modelo Canvas no qual este possibilita a criação de modelos de negócios e bem como prototipações de projetos próprios, conforme apresentado na Figura 1. (CANVAS, *sd*)

As bases para a receita do produto consistiriam na venda de licenças do jogo, através dos principais parceiros, como as grandes lojas de jogos digitais, que realizam boa parte do marketing e a atração de clientes de forma orgânica. Esses canais também possuem formas de contato com o cliente, que poderiam ser usados como os principais canais de comunicação com os mesmos.

A grande diferenciação do produto é a aplicação da inteligência artificial de forma clara dentro do mundo do jogo, e que proporciona novas possibilidades para os jogadores, permitindo não só a exploração de segmentos de clientes que gostam apenas de aproveitar os jogos, mas também de interessados pela de *Machine*

Learning, que é o princípio de reconhecimento de padrões feitos por um algoritmo para que este tome decisões conforme for parametrizado.

Figura 1 - Modelo de Negócios Canvas

Fonte: Autores.

Os custos principais a princípio não são grandes, pois o jogo não possui a necessidade da manutenção de um servidor, já que não conta com componentes *online*, ou interação entre usuários. Os maiores gastos girariam em torno dos impostos sobre as vendas dos jogos e a energia elétrica utilizada durante o tempo de desenvolvimento.

3 Levantamento de Requisitos

O *Minimum Viable Product* (MVP) do jogo contará com algumas funções, ciclos e mecânicas, que serão descritos nesta seção, no formato de *Game Design Document*, utilizado por empresas de jogos no processo da criação e elaboração de seus jogos.

Aqui os controles de jogos serão chamados de *joystick ou com menção específica a de alguma marca conhecida.*

3.1 Mecânicas básicas de *gameplay*

O jogo construído conta com várias mecânicas disponíveis ao jogador, que pode realizá-las por meio dos controles implementados no jogo, sendo estas ações descritas, a seguir, nas subseções.

3.1.1 Personagens

O jogador terá disponível apenas um modelo 3D controlável para utilização durante a *gameplay*.

3.1.2 Movimentação

O jogador poderá mover o personagem em todas as direções disponíveis, utilizando as teclas A, S, D, F, ou o joystick de um controle conectado.

3.1.3 Combate

- Ataque

  O jogador poderá realizar ataques com o personagem, utilizando o botão esquerdo do mouse, ou o botão superior direito do *joystick* (R1 em controles *Playstation* ou RB em controles *Xbox*).

- Correr

  O jogador poderá correr enquanto se movimenta, utilizando a tecla Shift do teclado, ou apertando o botão “L3” no *joystick*.

- Defender

  O jogador poderá defender-se de ataques inimigos utilizando a função de defesa, apertando o botão direito do mouse, ou o botão superior esquerdo do *joystick* (L1 em controles *Playstation* ou LB em controles *Xbox*).

3.1.4 Interface

- Pausa

  O jogador poderá pausar o jogo a qualquer momento, pressionando o botão “ESC” no teclado ou o botão “*Start*” no *joystick*.

- Vida

  O jogador poderá verificar sua vida no canto superior esquerdo da tela, através de uma barra vermelha, que se reduzirá conforme sofre ataques e recebe danos do oponente.

Ações possíveis de serem realizadas pela máquina utilizando a Inteligência Artificial:

- Movimentação

  A máquina poderá mover o personagem em todas as direções disponíveis no eixo Z e X, mas não poderá se movimentar utilizando o eixo Y.

- Atacar

  A máquina atacará o jogador quando este se encontrar a uma determinada distância, que é definida manualmente dentro da *engine* utilizada.

- Defender

  A máquina poderá defender os ataques realizados pelo jogador, caso a colisão da espada do jogador tenha sido detectada na espada da máquina, e não em contato com seu corpo.

Regras de jogo que são seguidas obrigatoriamente tanto pela máquina quanto pelo jogador, sendo algumas delas independente de ambos, similar as regras de negócio presentes em outros modelos de *software*:

- Vida do jogador:

Quando a vida do personagem chegar ao final da barra, ele morrerá. Um menu será aberto para a escolha do reinício do jogo ou voltar ao menu inicial.

- Vida da máquina:
  Quando a vida da máquina chegar ao final, ela morrerá, e uma nova máquina será colocada no local de início das máquinas para a continuação do jogo, sem a necessidade de comandos do jogador.

- Conectividade:
  O jogo será totalmente *offline*, sem a necessidade de conexão com a internet para suas funcionalidades.

- Controles:
  Para o controle, serão aceitos o teclado e mouse e *Gamepads* (PS4, Xbox, Switch, entre outros).

As mecânicas descritas são as mínimas viáveis para os testes do produto, sendo as demais funcionalidades inseridas por meio de atualizações.

4 Ferramentas e Métodos ou Desenvolvimento

Diversas ferramentas e métodos são utilizados no desenvolvimento de um jogo. Os principais itens utilizados neste projeto estão descritos na Seção 4.1, em que se pode observar mais claramente para quais funções foram aplicados.

4.1 Ferramentas

O principal elemento utilizado para a construção do jogo é o motor gráfico, também chamado de *Engine*. O motor gráfico utilizado neste projeto foi a Unity, que consiste em um motor gráfico multiplataforma desenvolvida por ela mesma e teve sua primeira distribuição em 2005. Seu modelo de negócio se baseia em licenças, possuindo um plano gratuito para usuários ou empresas cuja renda não

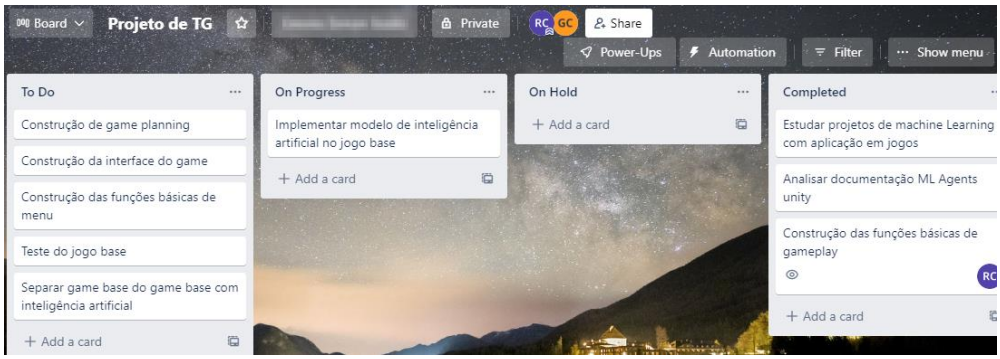
ultrapasse $100.000,00 (cem mil dólares) por ano. A versão utilizada neste projeto foi a v2021.3.8f1 (UNITY, sd).

Com ela é possível desenvolver jogos para vários dispositivos e sistemas, como *Mac, Linux e Windows*, além do Android, *Virtual Reality* (VR) e *eXtended Reality (XR)* e consoles das duas últimas gerações de vídeo games.

A ferramenta Trello, possibilita a organização de *checklists* e procedimentos em geral de forma intuitiva e colaborativa, até mesmo é possível realizar um PMBOK, que engloba todas as etapas da gestão de um projeto, desde o planejamento até ele entregue e finalizado, no caso deste projeto foi usada para gerenciamento das tarefas a serem realizadas pela equipe, de uma forma que o progresso do desenvolvimento pudesse ser mais facilmente visualizado, além da facilidade para criação de novas atividades e da análise do escopo do *software*, conforme apresentado na Figura 2. (TRELLO, sd)

Foi utilizado *Python* durante boa parte da programação das Redes Neurais que é elemento primordial na *Machine Learning* da Inteligência Artificial, mas a maior parte foi feita em C#

Figura 2 - Interface do Trello com cards de projeto

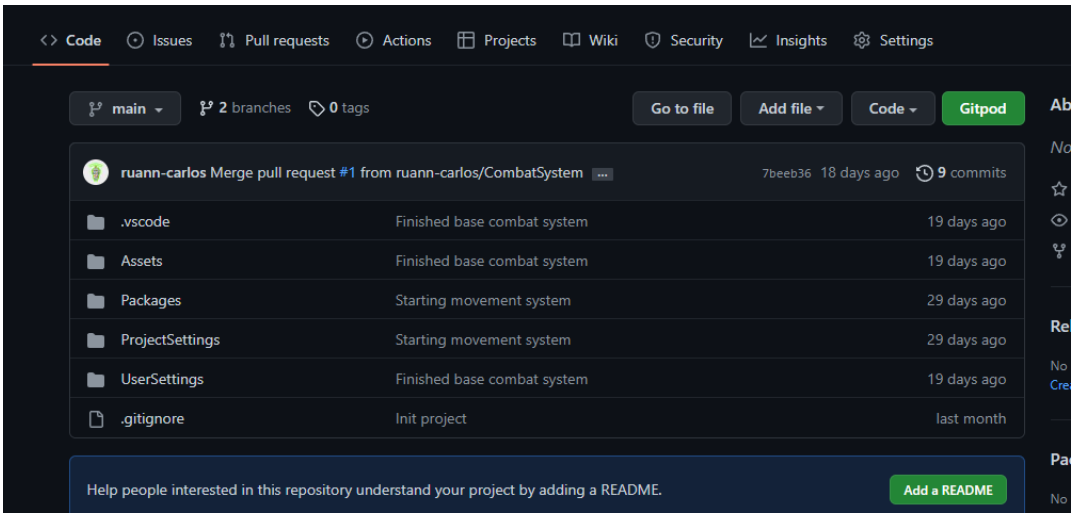


Fonte: Autores

Para o versionamento e histórico das alterações realizadas durante o projeto, utilizou-se a ferramenta GIT, em conjunto com o Github, que pode ser visualizado na Figura 3. Ele se trata um serviço de hospedagem para código na internet. Utiliza como base o Git, para versionamento do código, que é uma ferramenta de código aberto, criada originalmente por Linus Torvalds, durante a criação do Kernel do Linux. O

Github é o maior serviço de hospedagem de código utilizado atualmente (GITHUB, *sd*).

Figura 3 - Interface GitHub em um projeto



Fonte: Autores

A *Unity Machine Learning Agents Toolkit (ML-Agents*) é um projeto *open source* que possibilita o desenvolvimento e treinamento de Inteligência Artificial para jogos na *engine Unity*, utilizando diferentes abordagens, como a aprendizagem por reforço, imitação, entre outros. A versão utilizada no projeto é a **v0.15** (MARYHONARI, 2020).

4.2 Métodos ou Desenvolvimento

O princípio do processo de desenvolvimento deste projeto seguiu o fluxo presente em diversas empresas de jogos, e foi realizado principalmente em cinco etapas.

O primeiro material a ser construído foi o *Game Design Document* (GDD). Nele estão descritos o básico das mecânicas e *assets* (imagens, áudios etc.) utilizados no projeto. Esta etapa é de suma importância para todo o desenvolvimento do projeto, pois ela guia como será o estilo artístico utilizado, o que o jogador deverá conseguir realizar em um primeiro teste de produto, bem como a definição da história (quando esta existe) por trás do jogo.

Com esses fatores básicos estabelecidos, foi necessário buscar referências artísticas para o projeto, tomando por base alguns jogos com mecânicas parecidas. Uma das maiores inspirações utilizada neste projeto foi o jogo “*Dark Souls*”, mostrado

na Figura 4 e criado pelo estúdio japonês *FromSoftware*, distribuído pela Bandai Namco. Esse jogo possui uma ambientação medieval sombria, com visão em terceira pessoa, em que o jogador assume o papel de um cavaleiro em sua jornada.

Figura 4 - Jogo *Dark Souls* durante uma gameplay

Fonte: PERES, 2018, online.

Com as referências em mente, foi necessário a obtenção dos *assets* a serem utilizados no jogo, como o modelo 3D dos personagens, áudios e texturas. Todo esse material foi obtido na *Unity Asset Store*, uma plataforma disponibilizada pela própria *Unity*, em que é possível comprar e vender *assets*, para que os mesmos possam ser utilizados nos jogos. A loja possui também alguns *assets* gratuitos, os quais foram usados para este projeto.

Foi pensado em inicialmente buscar alternativas gratuitas de desenvolvimento para medir a possibilidade de retorno num possível investimento, de modo que desperdícios sejam evitados

A temática dos personagens é medieval, porém não tão sombria como as fontes de inspiração, possuindo um visual quase caricaturado, visto na Figura 5.

Figura 5 - Modelo 3D do jogador

Fonte: autores

Por se tratar de um MVP, não foi construído o cenário todo, e sim um quadrado de tamanho apropriado para a movimentação do jogador e da máquina, necessário para futuro treinamento da Inteligência Artificial, com um *asset* para a grama no chão, sem nenhum tipo de modelo 3D.

Construído o cenário, deu-se início ao processo de construção das mecânicas de *gameplay* através do código. A linguagem de programação utilizada foi o C#, que é a linguagem padrão da *Unity*.

A movimentação do jogador foi a primeira a ser desenvolvida, sendo feita de maneira a aceitar tanto comandos do teclado e mouse, quanto de *joysticks*, que são amplamente utilizados pelos jogadores. O modo como a movimentação é construída em um jogo dita muito a maneira como ele será jogado pelos usuários, tendendo a ser mais fluída e rápida em jogos de ação, e mais lenta e travada em jogos de estratégia, por exemplo.

Outra função extremamente importante é o controle da câmera. Como este projeto utiliza um modelo de jogo em terceira pessoa, conforme inspirações citadas anteriormente, a câmera fica posicionada atrás do jogador, de forma a estar sempre observando o eixo Z, o qual o modelo 3D fica direcionado e utiliza por base em sua movimentação.

O terceiro componente implementado foi o sistema de combate. Esse sistema deve atender tanto aos comandos dos jogadores quanto às funções executadas pela máquina, possibilitando um melhor aproveitamento de código. Cada personagem presente no mapa, seja ele jogável ou não, possui uma espada em sua mão direita, que por sua vez utiliza um colisor próprio, responsável por averiguar quaisquer colisões com outros corpos, enviando a mensagem para o *script* responsável pela verificação do objeto colisor e suas características, permitindo a diferenciação entre uma parede ou um corpo que possa receber dano da espada, como mostra a Figura 6.

Figura 6 - Modelo de espada 3D e seu colisor

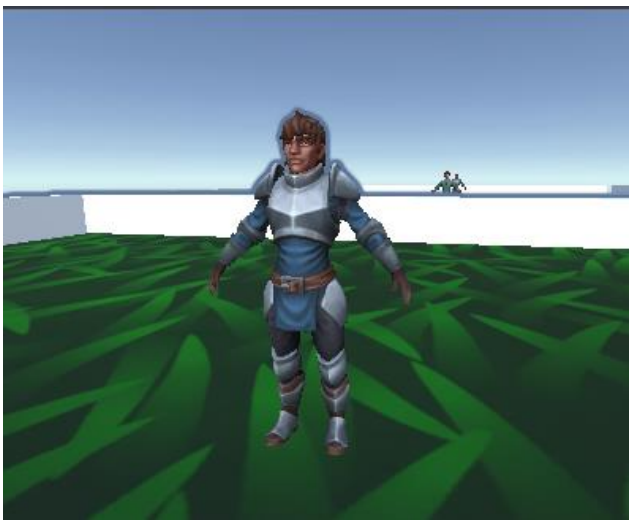
Fonte: Autores

É necessário que os sistemas de colisões utilizados em jogos sejam precisos e muito bem calculados pela *engine*, para que nenhum comportamento inesperado ocorra, como o jogador receber danos sem que a colisão tenha de fato ocorrido ou que objetos que não devam receber nenhum tipo de influência do sistema de combate possam chamar os métodos responsáveis pelo mesmo, ocasionando assim em exceções, já que esses componentes não possuem os itens necessários para funcionamento em conjunto com o sistema de combate.

Por último, foi implementado o agente treinador da Inteligência Artificial, para que esta possa começar a aprender e executar movimentações por conta própria, e assim alcançar o nível esperado de proficiência no jogo.

Para a construção desse agente, foi colocado o modelo 3D de uma personagem parecida com a do jogador, apresentada na Figura 7, porém com características que o permitam ser diferenciado como inimigo. Nele foram colocados todos os *scripts* necessários para a movimentação e o combate, possibilitando que ele realize todas as ações que um jogador faz, com exceção da câmera, do qual ele não possui necessidade.

Figura 7 - Inimigos presentes no jogo



Fonte: Autores

Após isso, deu-se início ao processo de construção e treinamento do modelo de Inteligência Artificial. A abordagem utilizada neste caso foi a aprendizagem por reforço, que consiste em premiar o modelo lógico quando ele tomar ações desejadas, e puni-lo quando ações incorretas forem realizadas. A aplicação das recompensas pode ser vista no trecho de código apresentado na Figura 8, que atribui um ponto ao modelo sempre que a distância para o jogador for menor do que a distância desejada, ou seja, quando a máquina se aproxima o suficiente do jogador para poder atacá-lo.

Figura 8 - Código responsável pela recompensa da IA

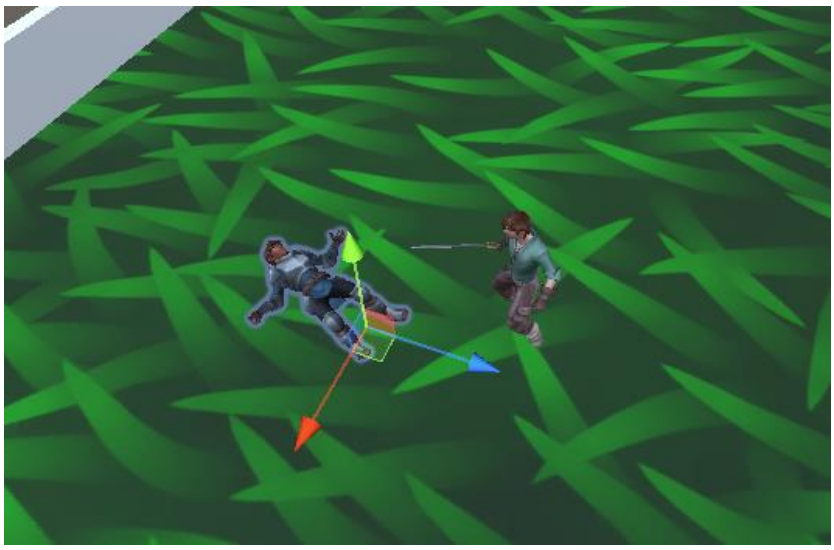
```
public override void OnActionReceived(ActionBuffers actions)
{
    Vector3 vectorForce = new Vector3();
    vectorForce.x = actions.ContinuousActions[0];
    vectorForce.z = actions.ContinuousActions[1];
    agentRigidBody.AddForce(vectorForce * speed * Time.deltaTime);
    float distanceToTarget = Vector3.Distance(transform.localPosition, target.transform.localPosition);
    if(distanceToTarget < distanceRequired){
        AddReward(1f);
        EndEpisode();
    }

}
```

Fonte: Autores

Estando o código para o treinamento da Inteligência Artificial completo, deu-se início a etapa de treinamento propriamente dita. Após entrar em modo de treinamento, a máquina irá realizar movimentações em direções aleatórias, aprendendo que a colisão com objetos como a parede são negativas e lhe retiram pontos, enquanto a aproximação com o jogador lhe trará aos poucos mais pontos. Também foram permitidos *inputs* do jogador durante o treinamento, conforme mostra a Figura 9, onde o jogador abateu o inimigo durante seu treinamento.

Figura 9 - Inimigo derrotado pelo jogador

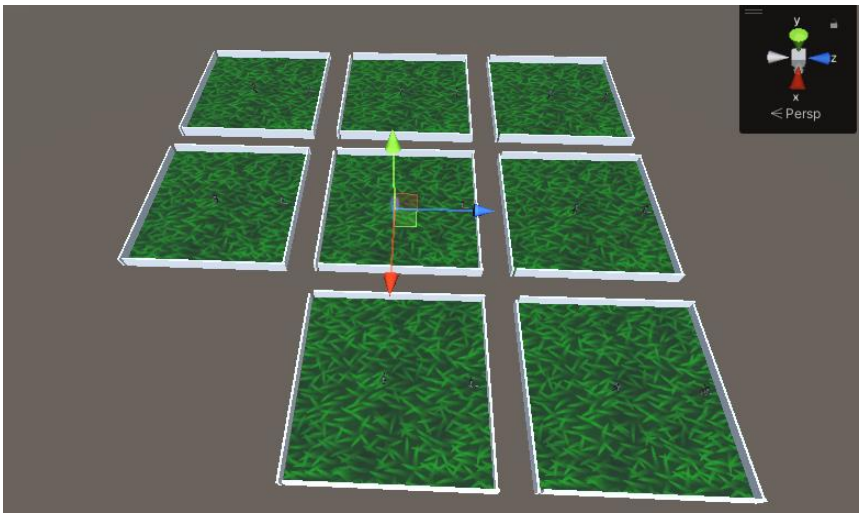


Fonte: Autores

Essa abordagem consome um longo período para que o modelo lógico aprenda com precisão as suas tarefas, porém, o processo de treinamento pode ser acelerado ao multiplicar o número de sessões de treinamento, conforme a Figura 10, o que quer dizer que quanto mais ambientes com o cenário, jogador e inimigo, forem replicados,

maior será a velocidade de aprendizado. O número de réplicas de ambiente deve ser de acordo com a capacidade da máquina utilizada para treinamento, para que ela não seja sobrecarregada durante o processo. Nos casos de teste realizados, utilizou-se oito réplicas de ambiente, cada uma com um jogador e um inimigo.

Figura 10 - Ambientes de treinamento da IA



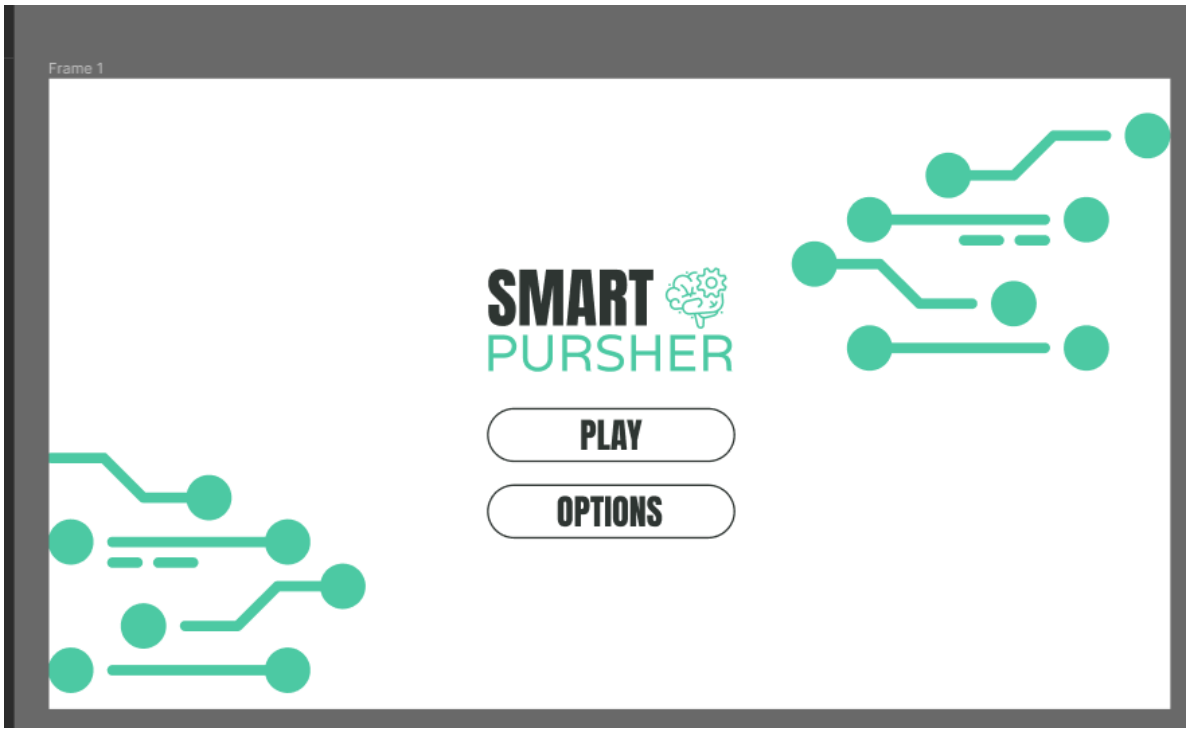
Fonte: Autores

5 Resultados e Discussão

O resultado do projeto construído foi um jogo com objetivo simples, mas divertido, com as ações do jogador restritas aos itens destacados no levantamento de requisitos deste trabalho, além também da Inteligência Artificial aplicada em situações de jogo, mais precisamente na perseguição do jogador, realizada não mais de uma maneira repetitiva, mas com a máquina sempre analisando seus arredores e possíveis obstáculos.

A interface do jogo foi construída conforme a Figura 11, com os botões necessários para o início do jogo e para sua saída.

Figura 11 - Menu inicial



Fonte: Autores

Já a movimentação dos jogadores pode ser verificada na Figura 12. O personagem pode se mover para todos os lados e correr, porém, enquanto corre ele fica impossibilitado de atacar, podendo apenas defender-se de ataques.

Figura 12: Jogador correndo

Fonte: Autores

O sistema de combate construído também não possui funcionalidades extravagantes ou movimentos extremamente complexos, visando um jogo mais rápido e que requeira menos habilidade dos jogadores, conforme pode ser visto nas Figuras 13, 14 e 15. O jogador pode atacar e defender-se dos ataques, sem combos de ataque ou ataques especiais.

Figura 13 - Inimigo se defendendo

Fonte: Autores

Figura 14 - Jogador atacando inimigo

Fonte: Autores

Figura 15 - Inimigo morrendo pelos golpes do jogador

Fonte: Autores

Essa simplicidade vem devido ao momento do projeto, onde seriam necessários a realização de testes beta, para a validação das mecânicas e do comportamento da inteligência artificial quando aplicada a uma grande quantidade de jogadores, o que não justifica grande polimento gráfico e uma alta aplicação de partículas. Isto pode ser implementado a longo prazo, através de atualizações, melhorando os aspectos visuais e mecânicos para torná-lo mais agradável e prazeroso visualmente.

Um dos pontos que também podem ser explorados futuramente são a interação dos jogadores e personagens não jogáveis (NPC's) com a ambientação presente no jogo. Atualmente, o mapa consiste em uma pequena arena, sem muitos obstáculos e sem a influência de eventos como vento, tempestades ou sol forte.

Através da construção de um mundo mais vívido e recheado de eventos naturais e não naturais, podemos tornar as batalhas ainda mais variáveis e intensas, trazendo um diferencial a mais para o projeto.

Considerações finais

A proposta inicial que foi mostrar o quanto que o uso de Inteligência Artificial traz benefícios para o que está sendo desenvolvido foi concluída. Assim, o jogo ganha complexidade e a plena capacidade de proporcionar uma experiência única a cada pessoa que for utilizar o *software* de maneira assertiva e eficaz.

Claro, há inúmeras possibilidades de criar soluções sem essa tecnologia, mas com o passar do tempo, vai se tornando repetitivo e muitas das vezes, não consegue proporcionar um atendimento personalizado e pode não resolver o problema do cliente.

Um dos principais desafios foi prever cada ação do usuário, para verificar se há algum problema com o código que poderia atrapalhar ou interromper o uso dele, pois existem diversas maneiras de interação com o jogo. Por conta disso, a parte de implementação e desenvolver a capacidade de *machine learning* dessa tecnologia foi uma incógnita.

Futuramente, planeja-se aumentar a complexidade do combate desenvolvido para ampliar o leque de possibilidades e personalização do teu personagem e um cenário mais fidedigno à realidade para ampliar ainda mais aquilo que já foi implementado.

Um dos grandes desafios deste projeto foi delimitar este projeto até que este fosse um Mínimo Produto Viável (MPV) dentro do prazo estipulado e disponibilidade de horário tendo que conciliar com outras obrigações, mas algo que foi muito positivo, foi a questão do ambiente de desenvolvimento totalmente colaborativo e aberto, como é esperado dentro do mercado de trabalho.

Referências

ABBOTT, Alisson. ***Gaming improves multitasking skills.*** 04/09/2013. Disponível em: <https://www.nature.com/articles/501018a>. Acesso em: 03.out.2022.

CANVAS. **Entenda de forma rápida o que é a ferramenta Business Model Canvas.** Atitude & Negócios. 2014. Disponível em: <https://atitudeenegocios.com/business-model-canvas/>. Acesso em: 09.out.2022.

GITHUB. *sd*. Disponível em: <https://github.com/>. Acesso em: 03.out.2022.

INTERNET MATTERS. **Benefícios de jogos de vídeo online.** *sd*. Disponível em: <https://www.internetmatters.org/pt/resources/online-gaming-advice/online-gaming-the-benefits/>. Acesso em: 03.out.2022.

MARYAHONARI ***Ml-agents.*** 2020. Disponível em: <https://github.com/Unity-Technologies/ml-agents>. Acesso em: 03.out.2022.

PERES, Gilson. **Análise: Dark Souls Remastered (Multi) é bom para novatos, mas nem tanto para veteranos**. 2018. Disponível em: <https://www.gameblast.com.br/2018/06/analise-dark-souls-remastered.html>. Acesso em: 13.out.2022.

ROCHESTER. ***Playing action video games can boost learning*** 10/11/2014. Disponível em: <https://www.rochester.edu/newscenter/playing-action-video-games-can-boost-learning-78452/>. Acesso em: 03.out.2022.

TRELLO, sd. Disponível em: <https://trello.com/> . Acesso em 11.out.2022

UNITY, sd. Disponível em: <https://unity.com/>. Acesso em 10.out.2022