

# Aquário

**Simulação gráfica em Unreal Engine 4 com ligação a CUDA,  
Inteligência Artificial e Base de Dados Cloud**

Pedro Miguel Medinas Fresco

2020



**INSTITUTO POLITÉCNICO DE BEJA**  
**Escola Superior de Tecnologia e Gestão**  
**Licenciatura em Engenharia Informática**

# **Aquário**

**Simulação gráfica em Unreal Engine 4 com ligação a CUDA,  
Inteligência Artificial e Base de Dados Cloud**

Elaborado por:

Pedro Miguel Medinas Fresco

Orientado por:

Doutor João Carlos Martins, IPBeja

Relatório de projeto de fim de curso apresentado na  
Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Beja

2020



# Resumo

## *Aquário*

*Simulação gráfica em Unreal Engine 4 com ligação a CUDA, Inteligência Artificial e Base de Dados Cloud*

*Este relatório pretende descrever a realização do projeto Aquário.*

*Este projeto tem como objetivo o desenvolvimento de uma aplicação em Unreal Engine 4 com ligação a CUDA e Python. Sendo uma prova de conceito, implementou-se uma ideia, realizada com o propósito de verificar se os conceitos em questão são suscetíveis de serem compatíveis.*

*O projeto consiste na criação de vários aquários virtuais com peixes, possibilitando a movimentação dos peixes de forma realista e do utilizador entrar dentro do aquário. Para além da simulação dos aquários, foi criado um mapa, onde se pode percorrer uma floresta, onde pássaros voam.*

**Palavras-chave:** *Aquário, Aprendizagem, Unreal Engine 4, Inteligência Artificial, CUDA, Modelos 3D, Python, C++.*



# Abstract

## *Aquarium*

*Graphic simulation in Unreal Engine 4 with connection to CUDA, Artificial Intelligence and Cloud Database*

*This report aims to describe the realization of the Aquarium project.*

*This project aims to develop an application in Unreal Engine 4 with connection to CUDA and Python. Being a proof of concept, an idea was implemented, carried out with the purpose of verifying whether the concepts in question are likely to be compatible.*

*The project consists of creating several virtual aquariums with fish, allowing the movement of fish realistically and the user entering the aquarium.*

*In addition to the aquarium simulation, a map was created, where you can walk through a forest, where birds fly.*

**Keywords:** *Aquarium, Learning, Unreal Engine 4, Artificial Intelligence, CUDA, 3D Models, Python, C++.*



## *Agradecimentos*

Agradeço a toda a minha família, por serem o suporte para a realização dos meus objetivos de vida.

Agradeço ao meu Orientador de projeto Doutor João Carlos Martins, ESTIG/IPBeja que me acompanhou, por toda a sua colaboração, disponibilidade, paciência, compreensão e profissionalismo que me proporcionou a concluir este projeto de fim de curso.

Agradeço a todos os professores que me acompanharam na minha formação, por terem enriquecido os meus conhecimentos.

Agradeço a todos os colegas, com quem tive o prazer de trabalhar em equipa e que me proporcionaram grandes aprendizagens.



# Índice

<b>Resumo</b>	<b>i</b>
<b>Abstract</b>	<b>iii</b>
<b>Agradecimentos</b>	<b>v</b>
<b>Índice</b>	<b>vii</b>
<b>Índice de Figuras</b>	<b>ix</b>
<b>1 Introdução</b>	<b>1</b>
<b>2 Estado da Arte</b>	<b>5</b>
2.1 Motor De Jogo . . . . .	5
2.2 CUDA . . . . .	11
2.3 Outras Tecnologias Usadas . . . . .	11
<b>3 Análise e Implementação</b>	<b>13</b>
3.1 Unreal Engine 4 . . . . .	13
3.1.1 Inteligência artificial do UE4 . . . . .	13
3.1.2 Movimento 3D . . . . .	14
3.1.3 Objetos . . . . .	15
3.1.4 Modelos 3D . . . . .	19
3.1.5 Colisão dos Objetos . . . . .	21
3.2 Ligação UE4-CUDA . . . . .	23
3.3 Percurso dos Peixes . . . . .	24
3.3.1 Algoritmo . . . . .	24
3.3.2 Ligação C++/Python . . . . .	24
3.3.3 Firebase . . . . .	28
3.3.4 CUDA e Python . . . . .	29

## ÍNDICE

---

3.3.5    Implementação . . . . .	30
<b>4 Resultados</b>	<b>41</b>
4.1    Mapa . . . . .	41
4.1.1    Casa . . . . .	42
4.1.2    Aquários . . . . .	42
4.2    Movimento da câmara . . . . .	48
4.3    Textura . . . . .	50
4.4    Animação . . . . .	53
4.5    Controlos . . . . .	54
<b>5 Conclusão</b>	<b>57</b>
5.2    Trabalho futuro . . . . .	58
<b>Bibliografia</b>	<b>61</b>
<b>Anexos</b>	<b>65</b>
<b>I Código Desenvolvido</b>	<b>67</b>

# Índice de Figuras

1.1	Demonstração do Projeto . . . . .	2
1.2	Sistema do Projeto . . . . .	3
2.1	Exemplo de configuração dos canais de colisão onde este componente colide com todos os canais exceto <i>Pawns</i> (sobreposição) e <i>Aquarium</i> (ignora)	8
2.2	<i>Blueprint</i> para criar um Material modificado do Peixe Palhaço . . . . .	9
2.3	Objeto <b>ASFishTank</b> criado em C++ e os seus descendentes em <i>Blueprint</i>	10
2.4	Demonstração do mapa com a aplicação parada . . . . .	10
3.1	<b>FishTarget</b> . . . . .	15
3.2	<b>Fishtank</b> versão 0 . . . . .	16
3.3	Objeto <b>Fish</b> versão 0 . . . . .	16
3.4	Função <i>Tick</i> do objeto <b>FishTank</b> . . . . .	17
3.5	Função <i>Tick</i> do objeto <b>Fish</b> . . . . .	18
3.6	Função <b>HandleOverlapCom</b> do objeto <b>Fish</b> . . . . .	19
3.7	Exemplo de um modelo de física de um <i>Skeletal Mesh</i> com uma <i>hitbox</i> complexa . . . . .	21
3.8	Exemplo da uma colisão simples de uma <i>Static Mesh</i> . . . . .	22
3.9	Exemplo da uma colisão complexa de uma <i>Static Mesh</i> . . . . .	22
3.10	<i>Tipo de Configuração</i> no projeto da biblioteca estática . . . . .	23
3.11	Ficheiro <i>Build</i> do Projeto de UE4 . . . . .	23
3.12	Funções na biblioteca estática que inicializa o interpretador de Python . . . . .	25
3.13	Função na biblioteca estática que chama, recebe e prepara o resultado do <i>script</i> de Python responsável pelo percurso dos peixes . . . . .	26
3.14	Código no ficheiro <i>build</i> do projeto de UE4 para compilação de Python . . . . .	27
3.15	Localização dos <i>scripts</i> de Python para a simulação <b>packaged</b> . . . . .	28
3.16	Firebase Firestore do Aquário . . . . .	28
3.17	Firebase Firestore dentro da coleção <b>Rows</b> . . . . .	29
3.18	Firebase Storage . . . . .	29
3.19	Exemplo de CUDA <i>Kernel</i> em Python usando Numba . . . . .	30

## ÍNDICE DE FIGURAS

---

3.20 Exemplo de chamada de CUDA <i>Kernel</i> em Python . . . . .	30
3.21 Função de conexão com o Firebase . . . . .	30
3.22 Início do <i>script</i> <b>GetEntireFishPath</b> . . . . .	31
3.23 Fim do <i>script</i> <b>GetEntireFishPath</b> . . . . .	32
3.24 <i>script</i> <b>addnewaquariumtodatabase</b> . . . . .	32
3.25 Função Main do <i>script</i> <b>createModelCuda</b> . . . . .	33
3.26 Parte da Função Main do <i>script</i> <b>createModelCuda</b> relacionada com a preparação das matrizes de adjacência . . . . .	34
3.27 Função <b>prepareWeightedAdjMat</b> do <i>script</i> <b>createModelCuda</b> . .	35
3.28 <i>Kernel</i> <b>addWeightToadjMatrixCuda</b> do <i>script</i> <b>createModelCuda</b> .	35
3.29 Parte da Função Main do <i>script</i> <b>createModelCuda</b> relacionada com a preparação do <i>dataset</i> . . . . .	36
3.30 <i>Kernel</i> <b>dijkstracuda</b> do <i>script</i> <b>createModelCuda</b> . . . . .	37
3.31 <i>Kernel</i> <b>dijkstracuda</b> do <i>script</i> <b>createModelCuda</b> . . . . .	38
3.32 Função <b>trainModel</b> do <i>script</i> <b>createModelCuda</b> . . . . .	39
4.1 Modelo Floresta . . . . .	41
4.2 Modelo Casa . . . . .	42
4.3 Aquário ID=0 . . . . .	42
4.4 Aquário ID=0 com a aplicação a correr . . . . .	43
4.5 Aquário ID=3 . . . . .	43
4.6 Aquário ID=3 com a aplicação a correr . . . . .	44
4.7 Aquário ID=2 . . . . .	44
4.8 Aquário ID=2 com a aplicação a correr . . . . .	45
4.9 Aquário ID=1 . . . . .	46
4.10 Aquário ID=1 com a aplicação a correr . . . . .	46
4.11 Dois Aquários com ID=1 . . . . .	47
4.12 Aquário ID=4 . . . . .	47
4.13 Aquário ID=4 com a aplicação a correr . . . . .	48
4.14 Todos os <i>blueprints</i> dentro do projeto . . . . .	48
4.15 <b>BP_ProtoFish_C</b> . . . . .	49
4.16 Parte do <i>playercontroller</i> para definir a câmara a possuir . . . . .	50
4.17 <i>Blueprint</i> para a modificação da cor da textura original do <b>ProtoFish</b> .	51
4.18 Textura original à esquerda e a nova textura à direita . . . . .	52
4.19 <i>Blueprint</i> para a modificação da cor da textura original do <b>SmallFish</b> , à esquerda a nova textura e à direita a original . . . . .	53
4.20 Texturas criadas com este método . . . . .	53

## Índice de Figuras

4.21 Animação de movimento para o <b>BP_KoiFish</b> . . . . .	54
4.22 Inputs possíveis dentro do jogo . . . . .	55



# Capítulo 1

## Introdução

O projeto tem como tema Aquário: Simulação gráfica em Unreal Engine 4(UE4) com ligação a CUDA, Inteligência Artificial e Base de Dados Cloud, simulação gráfica que mostra vários peixes a moverem-se dentro de vários aquários, usando várias tecnologias, com o objetivo de juntar diversos recursos proporcionados por cada uma delas.

Sendo uma prova de conceito, isto é, um modelo prático para explorar os seguintes conceitos:

- Uso de um Motor de Jogo para criação de objetos 3D
- Utilização CUDA em funções que são paralelizáveis
- Fazer ligação entre várias tecnologias.
- Criar um ou vários aquários com vários peixes.
- Possibilitar a movimentação dos peixes dentro do aquário.
- Impedir os peixes de saírem do aquário.
- Inserir inteligência artificial(IA) nos peixes de modo a criar percursos realistas.

## 1. INTRODUÇÃO

---

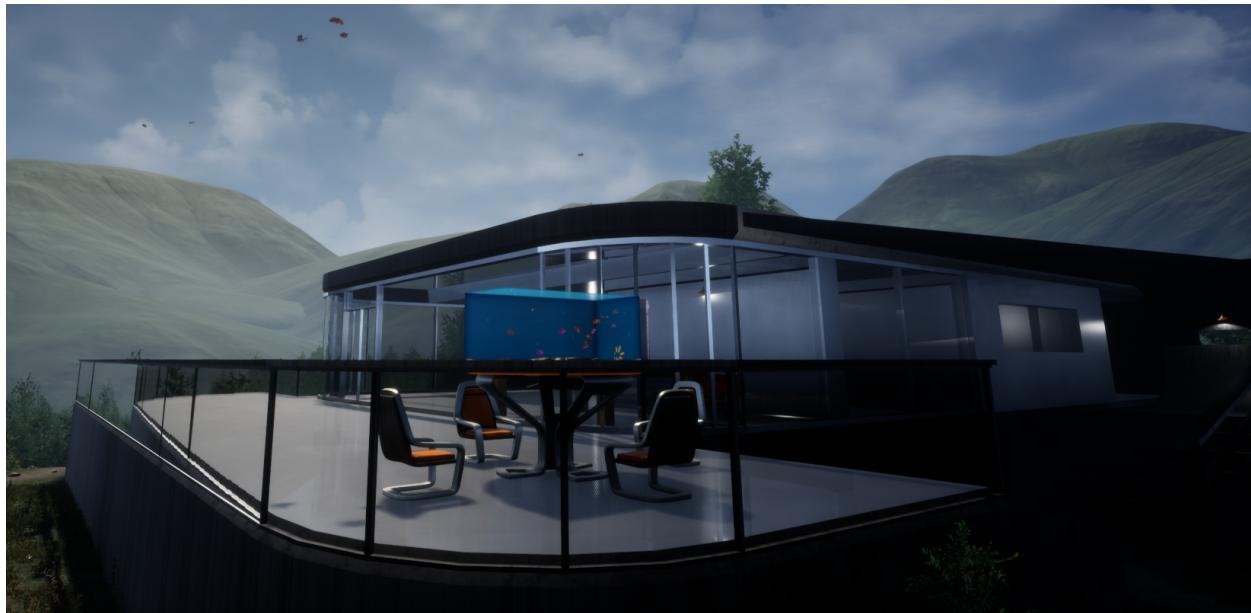


Figura 1.1: Demonstração do Projeto

Na figura 1.2 encontra-se o framework que este projeto segue, com o UE4 como ponto inicial, que para fazer os percursos dos objetos do tipo peixe, vai para uma biblioteca estática de C++ (*staticLib C++*) para chegar ao script de Python responsável pela sua criação, através do uso varias bibliotecas Python. E de seguida retorna esses valores pelo o mesmo percurso.

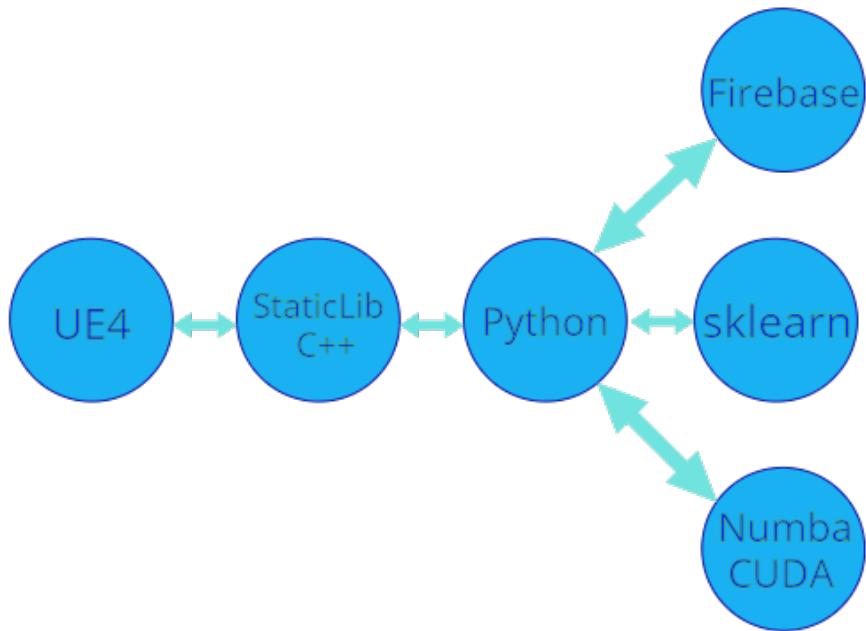


Figura 1.2: Sistema do Projeto

O relatório está organizado em mais quatro capítulos. O Capítulo 2 é uma visão geral do estado da arte que representa as várias tecnologias que se vai usar, nomeadamente:

- Motor De Jogo
- CUDA
- Outras Tecnologias Usadas

No Capítulo 3 é analisado e explicado o desenvolvimento no UE4, a ligação UE4-CUDA, o percurso dos peixes entre outros temas, salientando os seus problemas e

## 1. INTRODUÇÃO

a ligação entre os mesmos e o encontrar de soluções para cada problema que foi surgindo, de modo tornar o projeto mais realista. No Capítulo 4 é analisado e explicado o estado final do projeto. Por fim, no Capítulo 5, são apresentadas as conclusões, as várias dificuldades ao longo do projeto e trabalho futuro.

# Capítulo 2

## Estado da Arte

Neste capítulo são apresentadas várias tecnologias utilizadas durante o desenvolvimento do projeto e alternativas às mesmas.

### 2.1 Motor De Jogo

Para reduzir o tempo necessário para a criação da simulação do aquário usou-se um motor de jogo, designação de um *framework* que programadores utilizam para estruturar os vários mecanismos e elementos do jogo, por exemplo: gráficos, áudio, simulação da física.

Sendo as duas grandes escolhas:

1. Unity [1]
2. Unreal Engine [2]

Das principais diferenças entre estas tecnologias pode-se destacar:

Unity: [3, 4, 5]

1. Utiliza a linguagem de programação C#
2. Menor curva de dificuldade de aprendizagem para principiantes
3. Interface mais interativa e comprehensível
4. Mais utilizado para criação de jogos *indie* (Jogos eletrônicos independentes)
5. 50% dos jogos entre todas as plataformas são desenvolvidos em Unity
6. 60% dos jogos de realidade virtual usam Unity

## 2. ESTADO DA ARTE

---

7. Preferido no mercado de jogos para *smartphone*
8. Grande quantidade de tutoriais
9. Mais fácil de criar jogos 2D
10. Melhor para *low-end* hardware
11. Fornece apenas a fundação para a criação de jogos, mas tem poucas ferramentas especializadas.
12. Plataformas compatíveis: iOS, Android, Windows Phone 8, Tizen, Android TV, Samsung SMART TV, Xbox One, XBox 360, Windows PC, Mac OS X, Linux, Web Player, WebGL, HoloLens, SteamOS, PS4, Playstation Vita, and Wii U.

Unreal Engine: [3, 4, 5]

1. Utiliza a linguagem de programação C++ e Blueprint
2. Maior curva de dificuldade de aprendizagem para principiantes
3. Mais utilizado na criação de jogos triplo-A e filmes.
4. Melhores gráficos
5. Melhor para *high-end* hardware
6. Melhor para jogos de ação e *FPS*
7. Contém ferramentas mais especializadas, sendo que essas só podem ser utilizadas da maneira planeada pelo Unreal.
8. Existe mais procura de programadores com conhecimento em C++ e/ou Unreal
9. Plataformas compatíveis: iOS, Android, VR, Linux, Windows PC, Mac OS X, SteamOS, HTML5, Xbox One, and PS4

Sendo o Unreal Engine 4 (UE4) escolhido especialmente devido a utilizar C++, sendo essa a linguagem que o CUDA utiliza.

## Unreal Engine 4

O Unreal Engine é constituído por vários componentes que em conjunto conduz ao jogo, com uma grande biblioteca de ferramentas e editores que permite organizar e manipular os vários objetos do jogo.

Os principais componentes são os seguintes:

1. O motor da física, que mantém as leis da física, por exemplo colisões, gravidade, o mais parecido com a realidade.
2. O motor gráfico que organiza o que aparece no ecrã, verificando os recursos do sistema e reduzindo ou retirando gráficos, dependendo da necessidade, normalmente quando o objeto em questão se encontra distante do utilizador.
3. O motor de som, tem como único objetivo organizar os vários sons, como música, som ambiente e passos, e definir o que o utilizador é suposto ouvir na posição em que se encontra e o seu volume.
4. *framework* do jogo, prepara os *inputs* possíveis do utilizador, as regras do jogo e a inteligência artificial.

A base do UE4 são os seus objetos, que contêm muitas funções pré-definidas, sendo o tipo de objeto mais utilizado neste projeto os *actors*, que são objetos colocados estaticamente no mapa sem movimento. Os objetos *pawns* são capazes de realizarem movimento e os *characters* tem o objetivo de serem controlados pelo o utilizador. Todos os objetos criados pelo programador são descendentes destes objetos, em que se define o que se quer que ele faça modificando/acrescentando funções existentes ou criando novas.

Os objetos são compostos por vários componentes que têm vários objetivos pré-definidos, sendo os mais usados neste projeto *StaticMesh/SkeletalMesh*, onde se encontra o modelo 3D do objeto e os componentes baseados em formas geométricas, sendo estes usados normalmente para verificar sobreposição entre objetos de modo a chamar um *trigger*, função que é chamada em casos específicos, por exemplo quando objetos colidem ou se sobrepõem.

Os vários componentes também têm canais de colisão e de sobreposição, onde se define com que tipo de objetos eles devem colidir, sobrepor ou ignorar, com o objetivo de controlar os *triggers*.

## 2. ESTADO DA ARTE

---



Figura 2.1: Exemplo de configuração dos canais de colisão onde este componente colide com todos os canais exceto *Pawns*(sobreposição) e *Aquarium*(ignora)

## 2.1. Motor De Jogo

Também se deve notar a existência de blocos de código, chamados de *blueprint*, que neste projeto são utilizados para controlar e definir a forma dos objetos, controlar a visão do jogador e a modificação de texturas.

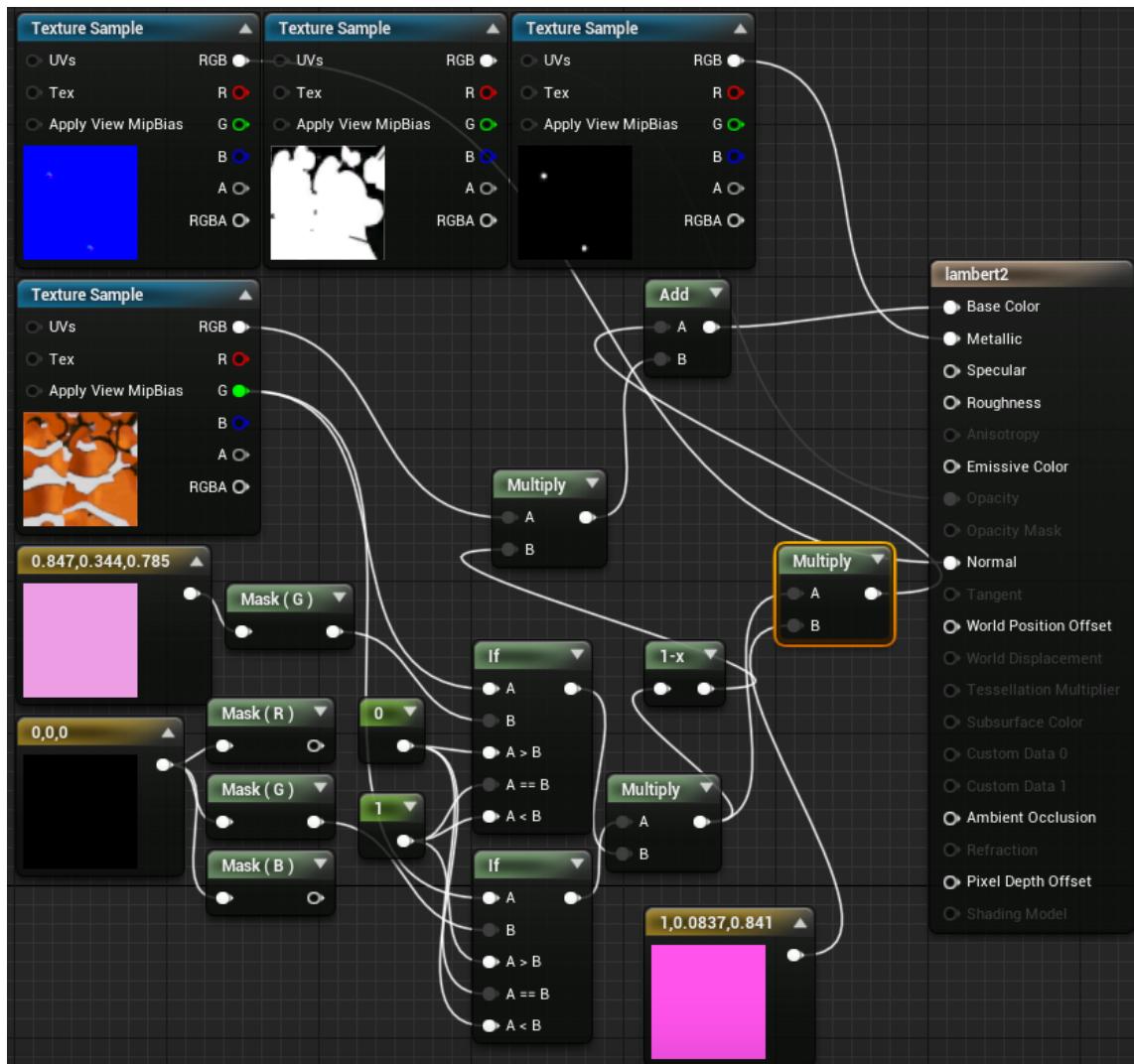


Figura 2.2: *Blueprint* para criar um Material modificado do Peixe Palhaço

Os objetos criados em C++ normalmente herdam em *blueprint* para facilitar a introdução de animações e *meshes* (formato 3D do objeto) entre outros parâmetros.

## 2. ESTADO DA ARTE

---

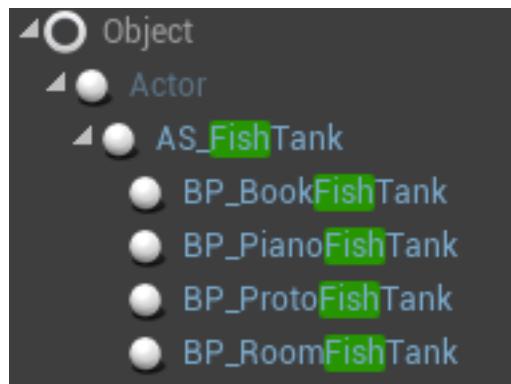


Figura 2.3: Objeto **ASFishTank** criado em C++ e os seus descendentes em *Blueprint*

Normalmente os objetos são colocados manualmente no mapa.



Figura 2.4: Demonstração do mapa com a aplicação parada

Devido à forma de como o UE4 compila o código, verifica-se várias limitações, como, por exemplo, não ser possível utilizar um ficheiro de C++ normal com funções, pois esse não chega a ser compilado, conduzindo a um problema com a utilização de CUDA e todos os objetos necessitam de ser criados no editor do *Inreal* indispensável para a criação dos vários ficheiros que definem aquilo que o UE4 deve compilar.

## 2.2 CUDA

CUDA é uma plataforma de computação paralela e modelo de programação da NVIDIA que permite a aceleração de programas utilizando os GPUs para paralelismo da computação. [6]

Sendo este utilizado em vários setores. Como por exemplo:

- Finanças
- Análise de dados
- Deep learning e machine learning
- Medicina

Sendo usado neste projeto com o objetivo de reduzir o tempo necessário para correr os vários algoritmos necessários para o funcionamento dele.

## 2.3 Outras Tecnologias Usadas

1. Linguagem de programação, C++[7]  
Linguagem de programação que o CUDA e UE4 utilizam.
2. Linguagem de programação, Python [8]  
Linguagem de programação que o scikit-learn e Firebase utilizam.
3. Controlador de versões, Github [9]  
O projeto e o relatório estão guardados num repositório de Github.
4. Website com Modelos 3D, sketchfab [10]  
Website onde se recolheu todos os modelos 3D utilizados no projeto.
5. Programa para modelagem, animação, texturização e composição, blender[11]  
Programa utilizado para converter os modelos 3D para um formato aceite no UE4 e correção de problemas nos modelos recolhidos.
6. Base de Dados, Firebase [12]  
Guarda a informação necessária para a execução da aplicação
7. IDE, Visual Studio 2019 [13]  
IDE necessário para trabalhar com UE4



# Capítulo 3

## Análise e Implementação

Neste capítulo são analisados os diferentes componentes da aplicação gráfica e é descrito o seu desenvolvimento nos seus diversos componentes, mais especificamente, UE4, CUDA, Python, C++ e Firebase.

### 3.1 Unreal Engine 4

#### 3.1.1 Inteligência Artificial do UE4

Verificou-se que para movimentar o *pawn*, é necessário recorrer à Inteligência Artificial (IA) do UE4, para esse se movimentar sem *input* do utilizador ou força externa. [14]

Existem três métodos principais com origem no UE4 para este objetivo:

1. Arrastar o *pawn* para uma coordenada definida, este método é o mais instável em termos de movimento, obriga o *pawn* a ter gravidade o que causa o objeto (peixe) a estar constantemente no chão do Aquário e ignora a rotação do peixe (peixe a nadar e a cair em 90º).
2. Definir vários pontos e utilizar *navmesh*. A *navmesh* é um objeto que cria uma carpeta verde no chão, onde se define todos os pontos em que é possível movimentar-se, por exemplo, existindo uma caixa no chão o *navmesh* deixa a área a volta dessa vazia. Assim quando os objetos se mexem esses conseguem desviar-se da caixa em questão, depois define-se os vários pontos entre os quais o objeto vai andar. No caso do peixe este método não funciona devido ao facto de o objeto no momento em que deixa de tocar no *navmesh* perde a IA e deixa de se poder movimentar, em outras palavras não é capaz do movimento 3D, necessário para movimento do peixe.

### **3. ANÁLISE E IMPLEMENTAÇÃO**

---

3. *Behavior tree*, IA mais avançada do UE4, devido a que neste método ser definidos vários comportamentos com prioridades e condições. No caso do movimento utiliza *queries*, onde são criados vários pontos temporários à volta do objeto em que o peso de cada ponto é determinado através das condições definidas na *query*. Este método tem o mesmo problema que o *navmesh*, devido à *query* ser automaticamente colocada no chão, não possibilitando movimento 3D.

Por outro lado existem estes três métodos com várias origens:

1. Adquirir *plugins* que corrigem e possibilitam usar a *navmesh* em 3D, sendo estes normalmente bastante caros.
2. Criar um descendente do *navmesh/query*, devido ás funções de movimento estarem quase todas dependentes do *navmesh/query*, uma solução seria modificar os ficheiros do UE4 para possibilitar movimento 3D, mas devido ao tempo necessário para seguir este método foi verificado não ser o mais prático.
3. Criar um método de movimento não dependente do *navmesh*, sendo este o método escolhido neste projeto.

#### **3.1.2 Movimento 3D**

Tendo em conta a decisão tomada na secção anterior, encontrou-se dois projetos usados como base para a solução do problema.

Sendo esses:

- Um dos projetos base do UE4 é o processo em que o jogador controla um objeto (nave espacial), o movimento é feito *frame* a *frame*, movimentando a nave espacial para a frente pela variável velocidade, multiplicado pelo tempo da frame e depois rodar a nave pelo vetor de rotação criado pelos vários inputs do utilizador. [2]
- O outro projeto é o **FlyAI**, criado por Peter L. Newton, em *blueprint*, que reutiliza o código de movimento encontrado no projeto base do UE4 e faz com que a rotação do objeto (nave espacial) seja definida pelo vetor de rotação entre o local onde a nave se encontra e o ponto de destino da nave, criando vários atores invisíveis ao longo do mapa que servem como alvo para a nave espacial. [15]

Foi utilizando estes projetos, limpando as partes desnecessárias, modificando as variáveis para se obter um movimento mais realista e acrescentou-se um limite à rotação de Y entre  $-30^{\circ}$  e  $30^{\circ}$ , para os peixes não nadarem de lado nem de cabeça para baixo.

### 3.1.3 Objetos

Criou-se 3 objetos, com objetivo de serem os aquários e os peixes, sendo estes:

1. **FishTarget**, *actor*, que vai servir de alvo, com um componente esférico para ocorrer sobreposição com os peixes com o objetivo de dar um volume de sucesso onde o peixe deve seguir para o alvo seguinte. Sem este componente o peixe só seguiria para o alvo seguinte quando chega-se a posição x,y,z exatas onde o alvo está colocado.

Devido a utilizar uma matriz de adjacência para definir os percursos possíveis entre vários alvos e a ordem de como o UE4 vai buscar os alvos ser inconsistente, acrescentou-se a variável Id.

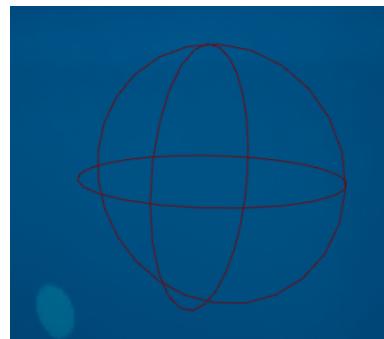


Figura 3.1: **FishTarget**

2. **FishTank**, *actor*, tem apenas um componente retangular com o simples objetivo de ir buscar todos os **FishTarget** que estão dentro do volume e organiza por Id. Dentro do editor de *unreal* quando é criado um herdeiro deste objeto é acrescentado um *staticmesh* para inserir o modelo do aquário. Também têm variáveis que definem a velocidade que o peixe poderá ter dentro do aquário (*CurrentForwardSpeed*, *Acceleration*, *TurnSpeed*, *MaxSpeed*, *MinSpeed*). Este objeto foi criado com o intuito de limitar a busca dos alvos na área do aquário, indispensável ao objetivo de ter mais do que um aquário no mapa.

### 3. ANÁLISE E IMPLEMENTAÇÃO

---

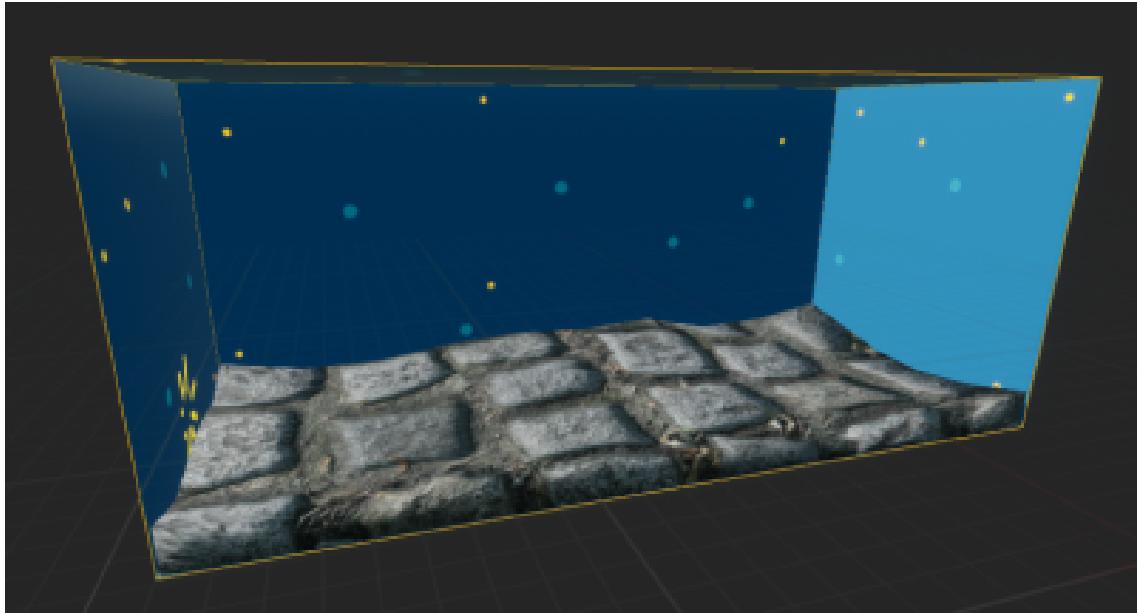


Figura 3.2: **Fishtank** versão 0

3. **Fish**, *pawn*, tem dois componentes, um *SkeletalMesh*, para colocar o modelo do peixe, e um componente esférico com o objetivo de fazer *trigger* à sobreposição com o **FishTarget** para seguir para o próximo **FishTarget**. Este objeto move-se usando o movimento 3D referido anteriormente. Este objeto está ligado a um aquário, onde foi buscar a velocidade e o *array* de **fishTargets** desse mesmo aquário. Notar que o modo de controle para que alvo se deve dirigir é feito através da criação de um *array* de inteiros com dimensão 5000 que contém o percurso, o identificador do **fishTarget**, e deve-se percorre-los por ordem do array, tendo um contador que vai buscar esse identificador para ser usado para indexar o array de **FishTarget** de modo a encontrar o **fishTarget** respetivo.



Figura 3.3: Objeto **Fish** versão 0

## Implementação

As funções nas figuras 3.4 3.5 3.6 são as mais relevantes em relação aos objetos criados em *unreal*.

A figura 3.4 é a função *Tick* (corre em todos os *frames*) do objeto **FishTank**, que vai buscar todos os **FishTarget** dentro da área do **FishTank**, e criando um *array* organizado pelos seus identificadores.

```
void AAS_FishTank::Tick(float DeltaTime)
{
    Super::Tick(DeltaTime);
    TArray<AActor*> tmp;
    rangeToSearchForTarget->GetOverlappingActors(tmp, AAS_FishTarget::StaticClass());
    //Vai Buscar Todos os Targets Dentro da Area
    //organiza os targets por Id
    targets.Init(NULL, tmp.Num());
    for (int32 i = 0; i < tmp.Num(); i++)
    {
        AAS_FishTarget* fishTarget = Cast<AAS_FishTarget>(tmp[i]);
        if (fishTarget)
        {
            int fishTargetId = fishTarget->Id;
            targets[fishTargetId] = fishTarget;
        }
    }
    SetActorTickEnabled(false); //Faz com que esta objeto não corra os ticks outra vez
}
```

Figura 3.4: Função *Tick* do objeto **FishTank**

A figura 3.5 é parte da função *Tick* do objeto **Fish**, onde vai esperar pelo objeto **FishTank** terminar de criar o seu *array* de **FishTarget**, e o vai utilizar para criar o seu percurso.

Sendo o motivo do objeto **FishTank** criar o *array* na função *Tick* em vez no seu construtor ou *BeginPlay*, porque a ordem de criação de objetos do UE4, baseado na ordem que o objeto foi criado, e não quando foi colocado no mapa, o que faz com que neste caso o objeto **FishTank** apenas seja capaz de procurar os **FishTarget** durante o *Tick*, devido a ordem de criação ter sido, **Fish** → **FishTank** → **FishTarget**.

### 3. ANÁLISE E IMPLEMENTAÇÃO

---

```
void AAS_Fish::Tick(float DeltaTime)
{
    Super::Tick(DeltaTime);
    //verifica se o peixe ja recebeu os fishtargets do fishtank
    if (gotTargets){ ... }
    else {
        if (Aquarium->targets.Num() > 0)
            //verifica se o array do fishTank(Aquarium) ja existe
        {
            //UE_LOG(FishLogCategory, Log, TEXT("inside2"));
            this->targets = Aquarium->targets;
            gotTargets = true;

            SetPath(); //Cria o Percurso
            //SetPath();

            TargetIndex = 0;
        }
    }
}
```

Figura 3.5: Função *Tick* do objeto **Fish**

A função na figura 3.6 é chamada sempre que o objeto **Fish** se sobreponem com outro objeto, onde se vai verificar se esse objeto é um **FishTarget**. No caso de ser um objeto **FishTarget** é verificar se é o alvo do percurso que se estava a procura.

```

void AAS_Fish::HandleOverlapCom(UPrimitiveComponent* OverlappedComponent,
                                AActor* OtherActor, UPrimitiveComponent* OtherComp,
                                int32 OtherBodyIndex, bool bFromSweep,
                                const FHitResult& SweepResult)
{
    //UE_LOG(FishLogCategory, Log, TEXT("overlap start"));
    AAS_FishTarget* targetHit = Cast<AAS_FishTarget>(OtherActor);
    //verifica se o objeto que deu overlap é um FishTarget
    if (!targetHit)
    {
        return;
    }
    //UE_LOG(FishLogCategory, Log, TEXT("overlap middle"));
    if (targetHit == targets[PathIndex[TargetIndex]])
        //verifica se é o fishTarget que se esta a procura
    {
        TargetIndex += 1;
        if (TargetIndex >= PathSize)
        {
            //UE_LOG(FishLogCategory, Log, TEXT("inside"));
            TargetIndex = 0;
        }
        if (PathIndex[TargetIndex] < 0 || PathIndex[TargetIndex] >= targets.Num())
            //no caso de error na memoria
        {
            PathIndex[TargetIndex] = FMath::RandRange(0, targets.Num()-1);
        }
        CurrentForwardSpeed = 0;
    }
}

```

Figura 3.6: Função HandleOverlapCom do objeto Fish

### 3.1.4 Modelos 3D

Tendo em conta o tempo necessário para criar modelos 3D, que são compostos por: esqueletos, animações e colisão. Houve a decisão de procurar modelos 3D online. Constatou-se que o UE4 apenas aceita ficheiros 3D com o formato .fbx, sendo possível converter outros formatos para .fbx usando o programa para modelagem 3D com o nome de blender [11], tendo em atenção que a conversão poderá não funcionar por vários motivos, sendo o mais comum o objeto ter dois *roots* de ossos, a base e o primeiro osso de um esqueleto que é usado como ponto de referência, pois o UE4 só aceita a existência de uma *root*.

Existe a possibilidade de a *mesh* de colisão ter ficado com buracos ao inserir o modelo no UE4, havendo a possibilidade de, neste caso, o peixe ficar preso na parede. Verifica-se raramente que o UE4 poderá correr o jogo antes de definir corretamente a *mesh* de colisão do aquário, fazendo com que os peixes sejam ”tele-

### **3. ANÁLISE E IMPLEMENTAÇÃO**

---

transportados” para fora desse.

Verificou-se também que entre as colisões dos peixes com o aquário e com outros peixes existe a possibilidade de eles se sobrepor em vez de colidirem. O problema é que depois de se perceber que eram para colidir eles ficam presos dentro da *mesh* de colisão de um e do outro, sendo que ao fim de segundos, ou minutos, eles serão libertados um do outro. Estes modelos muitas vezes não vêm centrados, o que poderá ser um problema no caso de este modelo ser o componente *root* do objeto, pois não se poderá movimentar dessa posição, sendo necessário reimportar o modelo alterando variáveis que modificam a posição onde esse objeto ficará quando importado.

Este são os modelos 3D utilizados neste projeto:

1. Redcoat Robin Download, criado por RunemarkStudio, licenciado sob CC BY 4.0 [16]
2. Fish Animated, criado por Pia Krensel, licenciado sob CC BY 4.0 [17]
3. Koi Fish, criado por lesliestowe , licenciado sob CC BY 4.0 [18]
4. Aquarium Piano, criado por Koppenhaver, licenciado sob CC BY 4.0 [19]
5. Aquarium Bookcase, criado por TRYFIELD, licenciado sob CC BY 4.0 [20]
6. Room aquarium (Now Animated), criado por NeoT1, licenciado sob CC BY 4.0 [21]
7. Futuristic house model WIP 2, criado por 3DHaupt, licenciado sob CC BY NC 4.0 [22]
8. Aquarium, criado por lennartburgold, licenciado sob CC BY 4.0 [23]
9. Clownfish, criado por misaooo, licenciado sob CC BY NC 4.0 [24]

### 3.1.5 Colisão dos Objetos

Os modelos de objetos são chamados de *mesh*, existindo dois tipos de *mesh*, a *Skeletal Mesh* e a *Static Mesh*.

A *Skeletal Mesh* para modelos com esqueleto que possibilita o uso de animações, contendo esta mais de três ficheiros, sendo esse o modelo do esqueleto, o modelo do objeto, o modelo de física (onde se define Hitbox do objeto) e as várias animações que o objeto poderá utilizar.

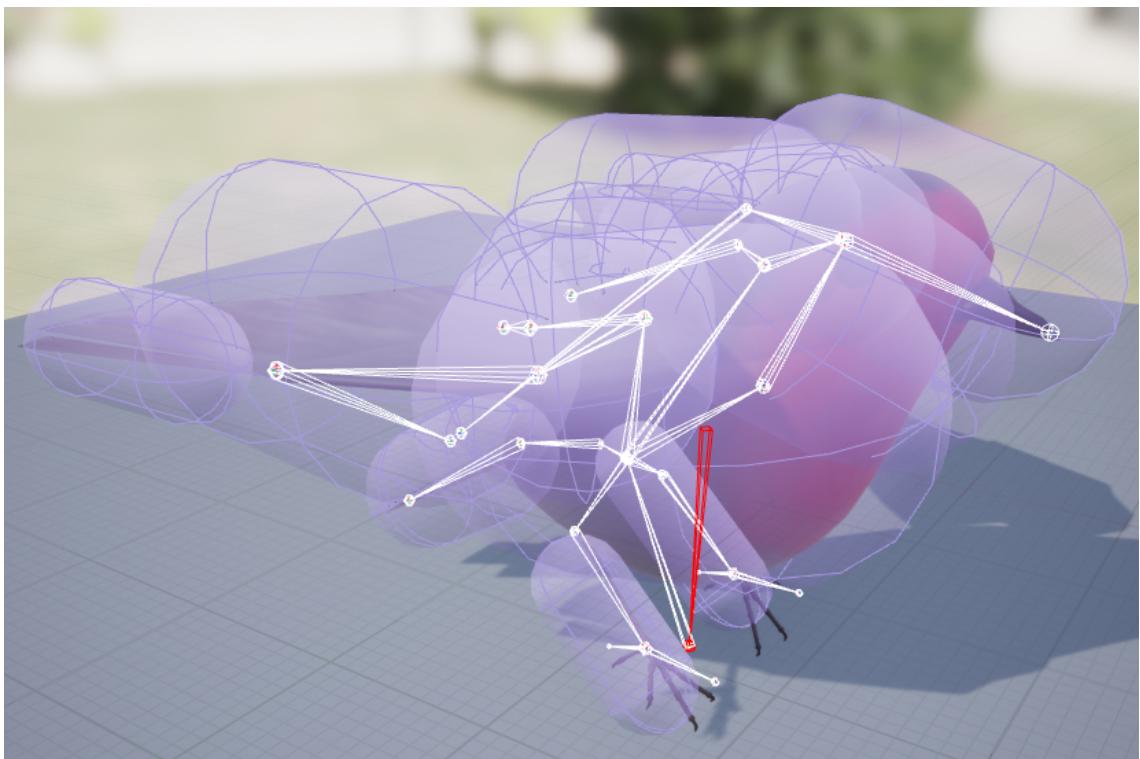


Figura 3.7: Exemplo de um modelo de física de um *Skeletal Mesh* com uma *hitbox* complexa

A *Static Mesh* para modelos sem esqueleto, isto é, sem a possibilidade de usar animações com este objeto, contendo uma colisão simples (a rede de colisão é composta por objetos simples à volta deste objeto) e uma colisão complexa (a rede de colisão complexa em que se segue todos os vértices do objeto). Sendo necessário, no caso de se pretender usar a colisão complexa, definir a *Collision Complexity* (complexidade da colisão) para ”*Use complex collision as simple*” (usar colisão complexa como simples), sempre que se utiliza uma colisão complexa aumenta o uso de hardware necessário.

### 3. ANÁLISE E IMPLEMENTAÇÃO

---

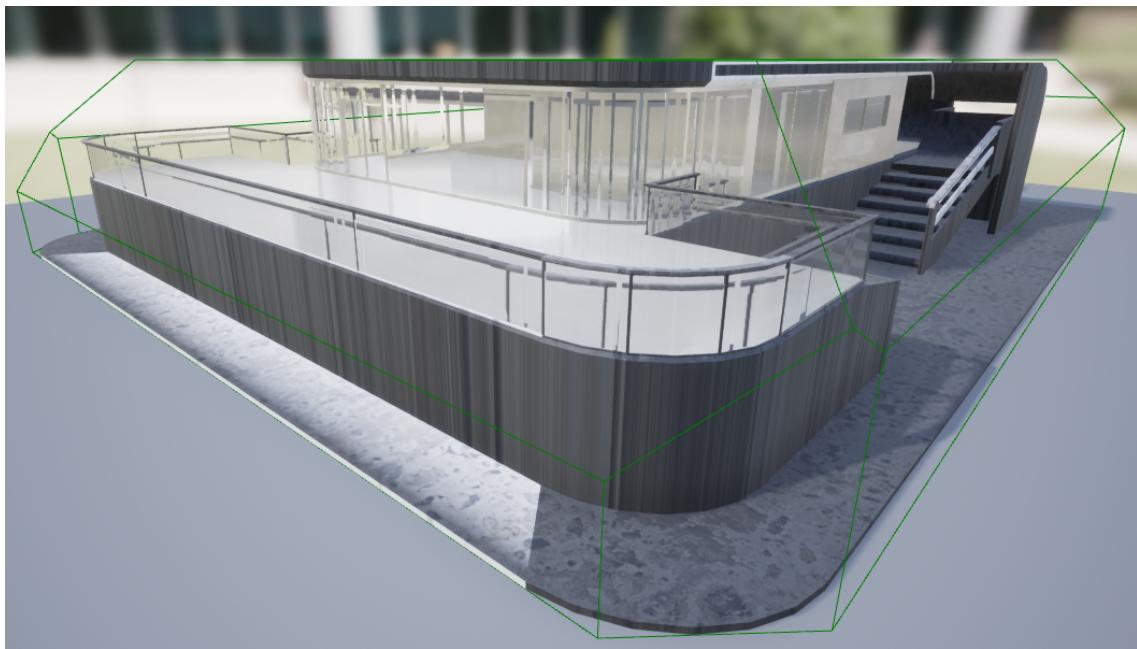


Figura 3.8: Exemplo da uma colisão simples de uma *Static Mesh*

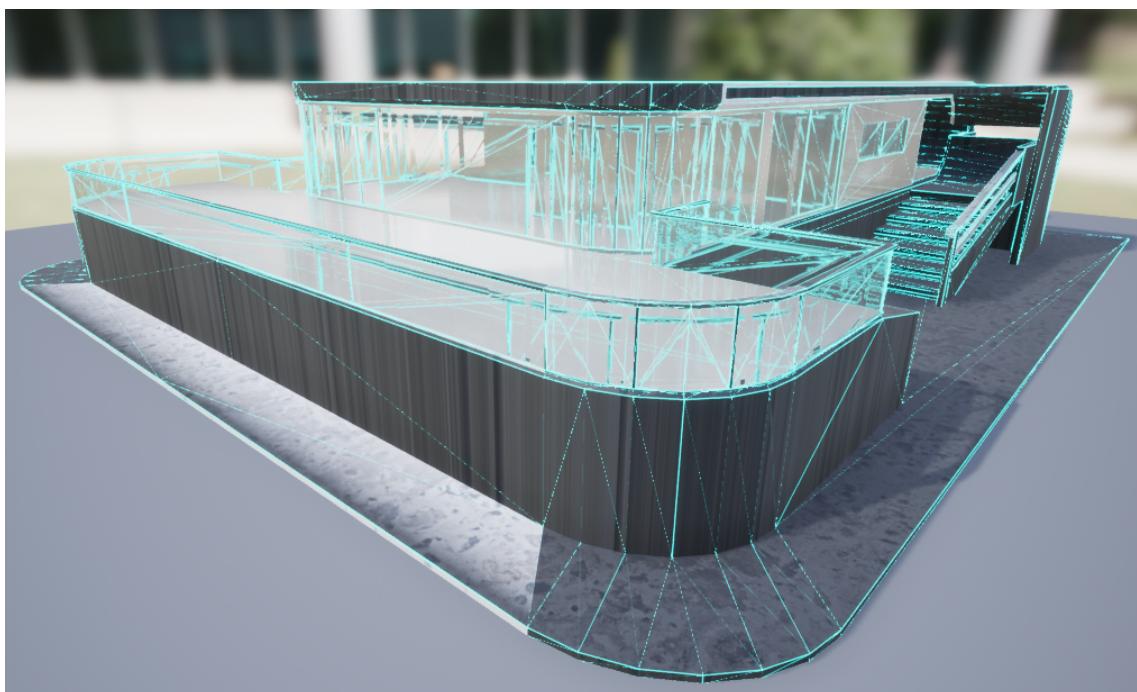


Figura 3.9: Exemplo da uma colisão complexa de uma *Static Mesh*

## 3.2 Ligação UE4-CUDA

Tendo em conta a inexistência de compatibilidade nativa entre UE4 e CUDA, descobriu-se a possibilidade de criar uma biblioteca estática de C++ com as funções de CUDA implementados nela, possibilitando chamar essas funções dentro do UE4.

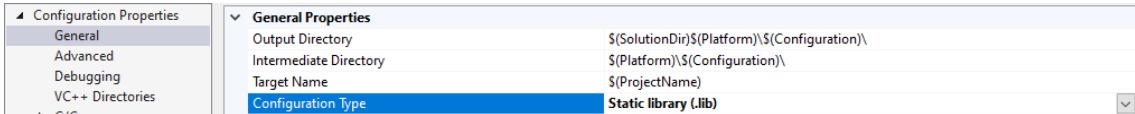


Figura 3.10: *Tipo de Configuração* no projeto da biblioteca estática

Para se conseguir fazer esta ligação é necessário guardar a biblioteca e as importações de outras bibliotecas na pasta do projeto UE4. Modificou-se o ficheiro **“Nome Do Projeto”.Build.cs** e acrescentou-se aos **PublicAdditionalLibraries** e **PublicIncludePaths** o percurso para biblioteca e as importações de outras bibliotecas respetivamente. Para possibilitar ao projeto compilar as funções de CUDA é também necessário acrescentar o **cudart.lib** e as suas respetivas bibliotecas ai importadas, sendo que estes se encontram no **NVIDIA GPU Computing Toolkit**, como se verifica na figura 3.11. [25]

Note-se que não é possível passar objetos nem funções do UE4 para uma função da biblioteca.

```
using UnrealBuildTool;
using System.IO;

public class UE4AquaSim : ModuleRules
{
    private string project_root_path
    {
        get { return Path.Combine(ModuleDirectory, "../../"); }
    }

    public UE4AquaSim(ReadOnlyTargetRules Target) : base(Target)
    {
        PCHUsage = PCHUsageMode.UseExplicitOrSharedPCHs;

        PublicDependencyModuleNames.AddRange(new string[] { "Core", "CoreUObject", "Engine", "InputCore" });

        PrivateDependencyModuleNames.AddRange(new string[] { });
        //biblioteca estatica
        string custom_fishpath_lib_include = "FishPathLib/include";
        string custom_fishpath_lib_lib = "FishPathLib/lib";

        PublicIncludePaths.Add(Path.Combine(project_root_path, custom_fishpath_lib_include));
        PublicAdditionalLibraries.Add(Path.Combine(project_root_path, custom_fishpath_lib_lib, "fishPathFindingLibrary.lib"));
        //cuda
        string cuda_path = "C:/Program Files/NVIDIA GPU Computing Toolkit/CUDA/v11.0";
        string cuda_include = "include";
        string cuda_lib = "lib/x64";

        PublicIncludePaths.Add(Path.Combine(cuda_path, cuda_include));
        PublicAdditionalLibraries.Add(Path.Combine(cuda_path, cuda_lib, "cudart_static.lib"));
    }
}
```

Figura 3.11: Ficheiro *Build* do Projeto de UE4

### 3.3 Percurso dos Peixes

Nesta secção é feita a descrição de como os peixes se movimentam segundo um dado percurso.

#### 3.3.1 Algoritmo

Devido aos vários alvos estarem localizados sempre na mesma posição no mapa converteu-se os vários grafos, um por cada aquário, tendo correspondência entre o numero do ponto e o ID do alvo, para várias matrizes de adjacência. Por fim, criou-se uma matriz de tipo, onde se verifica o tipo de alvo(exemplo: tipo A, tipo B, ou tipo C).

E por fim é criada uma matriz de pesos onde se mostram os pesos de andar entre pontos de vários tipos, sendo esta matriz assimétrica (exemplo: ir de um alvo do tipo A para um do tipo B têm peso 5, e voltar tem peso 10).

Usando-se as três matrizes criadas anteriormente, criou-se uma matriz de adjacência com pesos, sendo usado o algoritmo de Dijkstra para descobrir o percurso mais curto entro dois alvos.

Posteriormente correu-se o algoritmo de Dijkstra entre todos os alvos e criou-se um modelo de inteligência artificial com os resultados, tendo o objetivo de reduzir o tempo necessário para criar o percurso de 5000 alvos por peixe.

#### 3.3.2 Ligação C++/Python

Tendo em conta as várias limitações existentes de momento, houve a decisão de utilizar a biblioteca de Python scikit-learn [26] para criar o modelo e também utilizar a base de dados cloud, Firebase Firestore, da Google, para guardar as matrizes de adjacência, sendo que esta não têm compatibilidade com UE4 e a compatibilidade com C++ está limitada a Android, mas têm compatibilidade com Python.

Usando a Python/C API para chamar os *scripts* de python na biblioteca estática de C++. [27]

#### Implementação

A função na figura 3.12 foi criada para que, o mesmo interpretador de Python mantém-se carregado ao chamar outras funções de Python, através de verificar se esse existe ou não.

Devido que ao inicializar um novo interpretador, enquanto já existe outro interpretador, vai o fechar e libertar toda a memoria preparada para ele, e devido aos vários

peixes que vão chamar a mesma função que precisa de interpretador para receber o percurso, resultaria que apenas o último peixe fica-se com o percurso completo, enquanto os outros estão dependente se os endereços da memoria do seu percurso foram utilizados para outros serviços.

```
void FPF::InitializePython() {
    if (!Py_IsInitialized()) {
        //printf("Start");
        Py_Initialize();
    }
}
```

Figura 3.12: Funções na biblioteca estática que inicializa o interpretador de Python

Nesta figura está uma das várias funções que chama um dos vários *scripts* de Python.

### 3. ANÁLISE E IMPLEMENTAÇÃO

---

```
int* FPF::GetEntireFishPath(int aquariumId, int startPoint)
{
    int tmpArray[5001];

    PyObject* module, * func, * prm, * ret;

    FPF::InitializePython();

    PyObject* sys = PyImport_ImportModule("sys");
    PyObject* path = PyObject_GetAttrString(sys, "path");
    PyList_Insert(path, 0, PyUnicode_FromString("."));

    module = PyImport_ImportModule("getEntireFishPath");

    if (module != 0)
    {
        func = PyObject_GetAttrString(module, "main");
        prm = Py_BuildValue("(ii)", aquariumId, startPoint);
        ret = PyObject_CallObject(func, prm);
        if (PyList_Check(ret)) {
            int count = (int)PyList_Size(ret);
            printf("count %d\n", count);
            for (int i = 0; i < count; i++)
            {
                tmpArray[i] = (int)PyLong_AsLong(PyList_GetItem(ret, (Py_ssize_t)i));
            }
            tmpArray[5000] = count;
        }
        Py_DECREF(module);
        Py_DECREF(func);
        Py_DECREF(prm);
        //Py_DECREF(ret);
    }
    //FPF::FinalizePython();

    for (int i = 0; i < 10; i++)
    {
        printf("position %d value %d\n", i, tmpArray[i]);
    }
    printf("count %d\n", tmpArray[5000]);
    return tmpArray;
}
```

Figura 3.13: Função na biblioteca estática que chama, recebe e prepara o resultado do *script* de Python responsável pelo percurso dos peixes

Durante a preparação da variável **sys**, onde estão os parâmetros e funções específicas do sistema, verificou-se que se necessita de acrescentar a pasta onde os *scripts* de Python a ser chamados ao **sys.Path**, para se ter acesso aos *scripts* e aos *packages* instalados no Python.

Na função **ImportModule** inseriu-se o nome do *script* que se quer correr. No caso de este existir escolhe-se a função(**func**) dentro do *script* que se quer executar, preparando-se os parâmetros de entrada (**prm**, onde *input* de "(ii)", significa que precisa de dois inteiros). Estes parâmetros utilizam-se no **PyObjectCallObject** que vai correr as funções e retorna os valores em **PyObject**.

Posteriormente vai-se verificar se o **PyObject** do resultado é uma lista que contém de vários **PyObject** e quantos tem, indo uma a uma converter esses vários **PyObject** dentro da lista para inteiros e guarda-los num *array*.

### Ligaçāo UE4-Python

De modo a que o UE4 possa compilar os *scripts* de Python é necessário fazer duas modificações:

1. Acrescentar o caminho para a biblioteca **python\*versão\*.lib** e os seus *includes* ao ficheiro **"Nome Do Projeto".Build.cs** exatamente como se modificou para possibilitar a compilação de CUDA.

```
//python
string python_path = "C:/Users/fresc/AppData/Local/Programs/Python/Python38";
string python_include = "include";
string python_lib = "libs";

PublicIncludePaths.Add(Path.Combine(python_path, python_include));

PublicAdditionalLibraries.Add(Path.Combine(python_path, python_lib, "python38.lib"));
```

Figura 3.14: Código no ficheiro *build* do projeto de UE4 para compilação de Python

2. Acrescentar os *scripts* de Python na diretoria de trabalho do UE4, que se encontra no percurso **"/Epic Games/UE\_4.25/Engine/Binaries/Win64"**. Para se conseguir correr os *scripts* num projeto **Packaged** (compilado para uso comercial) deve de se colocar os *scripts* no percurso **"Percorso Escolhido Na Compilação/Nome do Projeto/Binaries/Win64"**.

### 3. ANÁLISE E IMPLEMENTAÇÃO

Nome	Data de modificação	Tipo	Tamanho
__pycache__	13/09/2020 21:26	Pasta de ficheiros	
addnewaquariumtodbatabase	17/08/2020 05:45	Python File	2 KB
createModel	16/08/2020 02:32	Python File	5 KB
createModelCuda	03/09/2020 02:46	Python File	7 KB
firebasekey	15/08/2020 01:44	JSON File	3 KB
getEntireFishPath	13/09/2020 21:13	Python File	2 KB
getShortestPathFromModel	17/08/2020 05:58	Python File	1 KB
modifyMatrix	13/09/2020 05:00	Python File	1 KB
testingifthisworkspython	07/09/2020 06:01	Python File	1 KB
UE4AquaSim	14/09/2020 23:49	Aplicação	134 949 KB
verifyIfAquariumExistsInDB	17/08/2020 06:40	Python File	1 KB

Figura 3.15: Localização dos *scripts* de Python para a simulação **packaged**

#### 3.3.3 Firebase

Esta base de dados do tipo NO-SQL, é organizada através de um documento para cada aquário em que o id do documento é o id do aquário dentro do UE4.

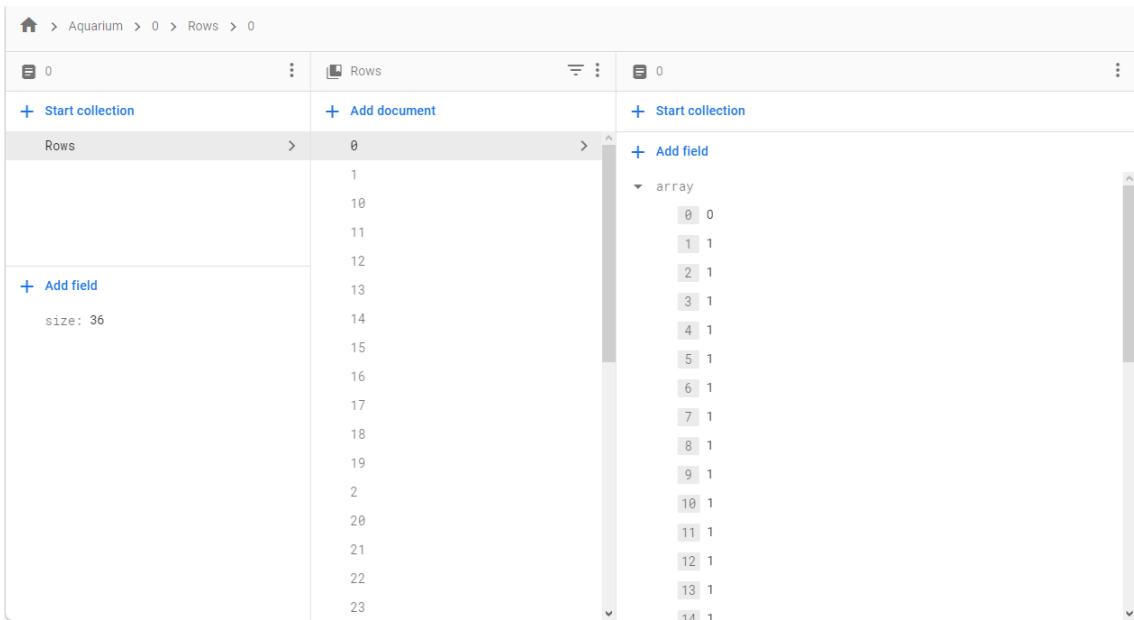
Dentro de cada documento encontrou-se a quantidade de alvos (size) que o aquário contém e uma coleção chamada **Rows**.

The screenshot shows the Firebase Firestore interface. At the top, there's a navigation bar with icons for home, back, and forward. Below it, the path 'Aquarium' is shown. The main area displays a table with three columns: 'finalprojectaquarium' (document), 'Aquarium' (subcollection), and 'Rows' (collection). The 'finalprojectaquarium' column shows a '+ Start collection' button. The 'Aquarium' column shows '+ Add document' and a list of four documents: 1, 2, 3, and 4. Each document has a 'size' field set to 36. The 'Rows' column shows '+ Start collection'. At the bottom right, there's a '+ Add field' button and a note 'size: 36'.

Figura 3.16: Firebase Firestore do Aquário

Dentro da coleção **Rows**, está a matriz de adjacência, em que cada linha é um documento com um array. Esta divisão foi necessária para reduzir o tempo que se demora a fazer o *download* e *upload* do documento, devido no caso de estes dados terem demorado 5 min para fazer a leitura no web-site do Firebase, tendo como desvantagem aumentar o custo da base de dados devido ao numero total de documentos a ler.

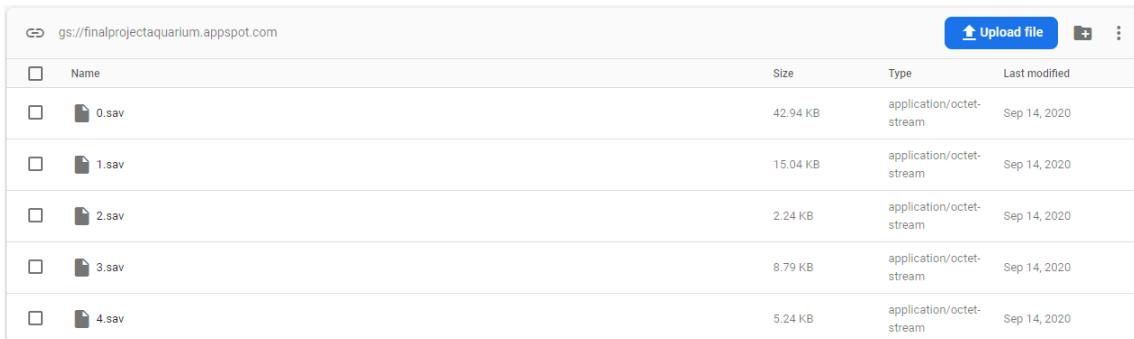
### 3.3. Percorso dos Peixes



The screenshot shows the Firebase Firestore interface. The left sidebar has a tree structure with 'Rows' selected. The main area shows a document named '0' under the 'Rows' collection. The document contains a field 'size: 36' and an array field. The array field's value is a list of 36 elements, each consisting of a number from 0 to 14 followed by a value of either 0 or 1. The interface includes standard Firestore navigation and creation buttons.

Figura 3.17: Firebase Firestore dentro da coleção **Rows**

Em relação aos modelos de IA, estes foram convertidos para ficheiros .sav através do Pickle e guardados no Firabase Storage, em que o seu nome é o identificador do aquário relevante para esse ficheiro.



The screenshot shows the Firebase Storage interface. It lists five files in the 'finalprojectaquarium.appspot.com' bucket: '0.sav', '1.sav', '2.sav', '3.sav', and '4.sav'. Each file is a binary file (application/octet-stream) with a size between 2.24 KB and 42.94 KB and was last modified on Sep 14, 2020. The interface includes an 'Upload file' button and a '+' icon for creating new buckets.

Figura 3.18: Firebase Storage

#### 3.3.4 CUDA e Python

Tendo em conta que as funções que podem fazer uso a de se encontram nos *scripts* de Python, utiliza-se Numba, um compilador para *arrays* e funções numéricas de Python, devido ao seu suporte ao uso de CUDA através da compilação de um subconjunto restrito de código de Python diretamente nos CUDA *kernels*. [28]

### 3. ANÁLISE E IMPLEMENTAÇÃO

---

```
@cuda.jit
def add_kernel(x, path, aquarium_id):
    start = cuda.grid(1)      # 1 = one dimensional thread grid, returns a single value
    stride = cuda.gridsize(1) # ditto
    # assuming x and y inputs are same length
    for i in range(start, x.shape[0], stride):
        x[i] = 0
```

Figura 3.19: Exemplo de CUDA *Kernel* em Python usando Numba

```
x=cuda.to_device(m)
threads_per_block = 128
blocks_per_grid = 30
add_kernel[blocks_per_grid, threads_per_block](x, path, aquarium_id)
x.copy_to_host(result)
```

Figura 3.20: Exemplo de chamada de CUDA *Kernel* em Python

#### 3.3.5 Implementação

A figura 3.21 mostra a função que é chamada em todos os *scripts* de Python que necessitam de se conectar à base de dados Firebase chamada no início. [29]

```
def createDBConnection():
    # Use a service account
    try:
        firebase_admin.get_app()
    except ValueError:
        cred = credentials.Certificate('firebasekey.json')
        firebase_admin.initialize_app(cred, {
            'storageBucket': 'finalprojectaquarium.appspot.com'
        })
    return firestore.client()
```

Figura 3.21: Função de conexão com o Firebase

A figura 3.22 mostra o início do *script* **GetEntireFishPath**, que cria o percurso do peixe que o chama. Neste *script* vai-se buscar o documento e o ficheiro .sav do aquário onde o peixe se encontra, sendo criado um ficheiro temporário com um nome único para anular as hipóteses de outro peixe fazer *overwrite* ao ficheiro, sendo este ficheiro apagado quando não for mais necessário.

```

def main(aquarium_id, startPoint):

    db = createDBConnection()
    doc_ref = db.collection(u'Aquarium').document(str(aquarium_id))
    doc = doc_ref.get()
    size = doc.get('size')

    bucket = storage.bucket()
    blob = bucket.blob(str(aquarium_id)+'.sav')
    millis = int(round(time.time() * 1000))
    tmpfilename="tmp_"+str(aquarium_id)+"_"+str(millis)+".sav"
    blob.download_to_filename(tmpfilename)

    with open(tmpfilename, 'rb') as f:
        loaded_model = pickle.load(f)

    fullpath = [int]*5000

```

Figura 3.22: Início do *script* GetEntireFishPath

Na segunda parte do *script* onde é criado o percurso, a partir do ponto inicial recebido do peixe, sendo este o identificador do alvo onde o peixe se encontra no início, para anular as possibilidades de o alvo do peixe ser a localização onde este se encontra no princípio, que resulta no peixe andar as voltas do mesmo ponto, até eventualmente sair da área do alvo e voltar a entrar.

O percurso é criado indo buscar o ponto alvo anterior e procurar um alvo aleatório, que não seja o mesmo que o ponto anterior, onde se vai procurar o caminho mais curto entre esses dois ponto através do modelo de IA, onde se acrescenta cada alvo no caminho mais curto ao *array* do percurso até esse ficar completo.

### 3. ANÁLISE E IMPLEMENTAÇÃO

---

```
k=0
while k < len(fullpath):
    if k==0:
        start = startPoint
    else:
        start=fullpath[k-1]
end=random.randint(0, size-1)
while start == end:
    end=random.randint(0, size-1)
path = loaded_model.predict([[start, end]])[0]
splitpath = path.split(',')
for point in splitpath:
    if k < len(fullpath) and k!=0 and fullpath[k-1] != int(point):
        fullpath[k]=int(point)
        k+=1
    elif k == 0 and start!=int(point):
        fullpath[k]=int(point)
        k+=1

if os.path.exists(tmpfilename):
    os.remove(tmpfilename)

return fullpath
```

Figura 3.23: Fim do *script* GetEntireFishPath

O segundo *script* que se deve tomar nota é utilizado para criar os documentos na base de dados, inserindo-se uma matriz 2D de tamanho X, composta completamente pelo valor 1 excepto no caso de coluna ser igual à linha, onde tem o valor 0.

```
def addMatToDb(id, mat, db, amount):
    doc_ref = db.collection(u'Aquarium').document(str(id))
    doc_ref.set({
        u'size': amount
    })
    for x in range(amount):
        doc_ref.collection(u'Rows').document(str(x)).set({u'array': mat[x]})

def main(id, amount):
    db = createDBConnection()
    mat = createMat(amount)
    mat = mat.tolist()
    addMatToDb(id, mat, db, amount)
```

Figura 3.24: *script* addnewaquariumtodatabase

O último *script*, chamado de **createModelCuda**, contém o algoritmo que se descreve anteriormente e corre para cada um dos Aquários na base de dados.

```
def main(maxWeight, typeAmount, path):

    db = createDBConnection()
    docs = [snapshot for snapshot in db.collection(u'Aquarium').stream()]
    for doc in docs:
        print(f'{doc.id}-----')
        print('Prepare Mat:')
        aquarium_id = int(doc.id)
        size = doc.get('size')
        list = [None] * size
        rows = [snapshot for snapshot in doc.reference.collection(u'Rows').stream()]
        for row in rows:
            list[int(row.id)] = row.get('array')
        matrix = np.asarray(list, dtype=np.int16)
        #print(matrix)
        adjMatrix = prepareWeightedAdjMat(matrix, size, maxWeight, typeAmount)

        print('Prepare Dataset:')

        dfc = pd.DataFrame(columns = ['start', 'end', 'path'])
        dfc['start']=dfc['start'].astype('int')
        dfc['end']=dfc['end'].astype('int')

        resultMat = np.empty((size,size,size), dtype=np.int32)
        resultMat[:]=-1

        drm=cuda.to_device(resultMat)

        threads_per_block = 32
        blocks_per_grid = 2

        dijkstracuda[blocks_per_grid, threads_per_block](adjMatrix, drm)

        drm.copy_to_host(resultMat)

        dfc = prepareDataframeFromResult(resultMat, dfc)

        print('dataset created')

        #print(dfc)
        print('Train Model:')
        trainModel(dfc, aquarium_id, path)

    print('-----')
```

Figura 3.25: Função Main do *script* **createModelCuda**

Este script está dividido em três grandes partes:

1. Preparar as matrizes

### 3. ANÁLISE E IMPLEMENTAÇÃO

---

2. Preparar o *dataset*(dijkstra)

3. Treino dos modelos

#### Preparação das Matrizes

No início da função vai-se buscar a coleção de Aquários a partir da base de dados. De seguida retira-se os valores do documento e prepara-se o *array* 2D com a matriz de adjacência, sendo modificado e resultando numa matriz de adjacência com pesos.

```
db = createDBConnection()
docs = [snapshot for snapshot in db.collection(u'Aquarium').stream()]
for doc in docs:
    print(f'{doc.id}-----')
    print('Prepare Mat:')
    aquarium_id = int(doc.id)
    size = doc.get('size')
    list = [None] * size
    rows = [snapshot for snapshot in doc.reference.collection(u'Rows').stream()]
    for row in rows:
        list[int(row.id)] = row.get('array')
    matrix = np.asarray(list, dtype=np.int16)
    #print(matrix)
    adjMatrix = prepareWeightedAdjMat(matrix, size, maxWeight, typeAmount)
```

Figura 3.26: Parte da Função Main do *script* **createModelCuda** relacionada com a preparação das matrizes de adjacência

Sendo esta a função que cria as matrizes de pesos e tipo, através de valores aleatórios, é depois chamado um *kernel* em CUDA (**addWeightToadjMatrixCuda**) que, através das duas matrizes criadas anteriormente vai modificar a matriz de adjacência.

```

def prepareWeightedAdjMat(matrix, size, maxWeight, typeAmount):
    targetAmount = size
    weightMat = createWeightMat(typeAmount, maxWeight)
    typeMatrix = createTypeMat(targetAmount, typeAmount)

    dwm=cuda.to_device(weightMat)
    dtm=cuda.to_device(typeMatrix)
    dm=cuda.to_device(matrix)

    threadsperblock = (32, 32)
    blockspergrid_x = math.ceil(len(dm) / threadsperblock[0])
    blockspergrid_y = math.ceil(len(dm[0])/ threadsperblock[1])
    blockspergrid = (blockspergrid_x, blockspergrid_y)

    addWeightToadjMatrixCuda[blockspergrid, threadsperblock](dm, dwm, dtm)

    return dm

```

Figura 3.27: Função `prepareWeightedAdjMat` do script `createModelCuda`

```

@cuda.jit
def addWeightToadjMatrixCuda(adjMatrix, weightMat, typeMatrix):
    xstart, ystart = cuda.grid(2)
    stridex, stridey = cuda.gridsize(2)
    for x in range(xstart, len(adjMatrix), stridex):
        xType = typeMatrix[x]
        for y in range(ystart, len(adjMatrix[x]), stridey):
            if(x == y):
                adjMatrix[x][y] = 0
            else:
                yType = typeMatrix[y]
                weightOnSpot = weightMat[xType-1][yType-1]
                adjMatrix[x][y] = adjMatrix[x][y] * weightOnSpot

```

Figura 3.28: Kernel `addWeightToadjMatrixCuda` do script `createModelCuda`

## Preparação Dataset

Nesta parte criou-se um Panda [30] *DataFrame* que guarda o identificador do ponto inicial do percurso (**start**), o identificador do ponto final do percurso (**end**) e o percurso entre os dois (**path**).

Posteriormente é executado o *Kernel* do dijkstra que utiliza a matriz preparada na

### 3. ANÁLISE E IMPLEMENTAÇÃO

---

fase anterior e guarda o percurso entre todos os pontos numa matriz 3D, sendo o x o ponto inicial, o y o ponto final e o z o percurso entre os dois, no fim esta matriz é inserida no *dataset*.

```
dfc = pd.DataFrame(columns = ['start', 'end', 'path'])
dfc['start']=dfc['start'].astype('int')
dfc['end']=dfc['end'].astype('int')

resultMat = np.empty((size,size,size), dtype=np.int32)
resultMat[:]=-1

drm=cuda.to_device(resultMat)

threads_per_block = 32
blocks_per_grid = 2

dijkstracuda[blocks_per_grid, threads_per_block](adjMatrix, drm)

drm.copy_to_host(resultMat)

dfc = prepareDataframeFromResult(resultMat, dfc)
```

Figura 3.29: Parte da Função Main do *script* **createModelCuda** relacionada com a preparação do *dataset*

O *kernel* da figura 3.30 executa o algoritmo de Dijkstra para todos os pontos. Denotar que devido ao facto do Dijkstra necessitar de vários *arrays* para guardar os nós, as distâncias e a sequência de nós necessários para cada *thread*, e todas terem valores individuais, criou-se *arrays* locais (**cuda.local.array**), sendo estes *arrays* privados para cada *thread*.

```

@cuda.jit
def dijkstracuda(graph, dest):

    start = cuda.grid(1)
    stride = cuda.gridsize(1)

    for src in range(start, graph.shape[0], stride):

        row = graph.shape[0]
        col = row

        dist = cuda.local.array(150, dtype=numba.float64)
        dist[:] = np.inf

        parent = cuda.local.array(150, dtype=numba.int64)
        parent[:] = -1

        dist[src] = 0

        queue = cuda.local.array(150, dtype=numba.int64)
        for i in range(row):
            queue[i] = i

        while verifyQueue(queue):

            u = minDistancecuda(dist,queue, row)

            if(u == -1):
                queue[:]=-1
            else:
                queue[u]=-1
                for i in range(col):
                    if graph[u][i] and queue[i] != -1:
                        if dist[u] + graph[u][i] < dist[i]:
                            dist[i] = dist[u] + graph[u][i]
                            parent[i] = u

            for i in range(0, row):
                s = addPathcuda(parent,i, dest, src)

```

Figura 3.30: Kernel dijkstracuda do script createModelCuda

### 3. ANÁLISE E IMPLEMENTAÇÃO

---

Salienta-se que todas as funções que o *kernel* chama devem ser funções do dispositivo, sendo que estas só podem ser chamadas por *kernels* ou outras funções do dispositivo.

```
@cuda.jit(device=True)
def verifyQueue(queue):
    for element in queue:
        if element != -1:
            return True
    return False

@cuda.jit(device=True)
def minDistancecuda(dist,queue, row):
    minimum = np.inf
    min_index = -1

    for i in range(row):
        if dist[i] < minimum and queue[i] != -1:
            minimum = dist[i]
            min_index = i
    return min_index

@cuda.jit(device=True)
def addPathcuda(parent,i,dest, src):
    j=i
    dest[src][i][0]=j
    k=1
    while parent[j]!=-1:
        j=parent[j]
        dest[src][i][k]=j
        k=k+1
```

Figura 3.31: *Kernel* dijkstracuda do script createModelCuda

### Treino dos Modelos

Depois de criado os *datasets* treina-se os modelos de IA e de seguida cria-se um ficheiro temporário do tipo .sav, onde se encontra o modelo, e envia-se esse ficheiro para a base de dados, com o nome de ”identificador”.sav, apagando-se o ficheiro temporário no fim do processo.

```
def trainModel(train, aquarium_id, path):
    train.info()
    y=train.pop('path')

    print('Start Fit:')
    p = LogisticRegression(max_iter=5000)
    print('Model Fit:')
    p.fit(train, y)
    print('Test Model:')
    p.predict([[0,1]])
    print('Save Model:')

    with open(path+'tmp.sav', 'wb') as f:
        pickle.dump(p, f)

    bucket = storage.bucket()
    blob = bucket.blob(str(aquarium_id)+'.sav')
    blob.upload_from_filename(path+"tmp.sav")

    if os.path.exists(path+"tmp.sav"):
        os.remove(path+"tmp.sav")
```

Figura 3.32: Função `trainModel` do script `createModelCuda`



# Capítulo 4

## Resultados

A aplicação gráfica acabou por conter 5 tipos de aquários, uma casa e uma floresta, com várias funcionalidades para facilitar a sua visualização.

### 4.1 Mapa

A floresta foi criada através do uso dos modelos e texturas da biblioteca para UE4, *Meadow - Environment Set*, onde se modificou o mapa exemplo utilizando o modo *landscape* e *foliage* do UE4.



Figura 4.1: Modelo Floresta

## 4. RESULTADOS

---

### 4.1.1 Casa

Dentro da floresta encontra-se uma casa onde estão expostos os vários aquários.



Figura 4.2: Modelo Casa

### 4.1.2 Aquários

Todos os tipos de Aquários foram criados para verificar várias possibilidades de movimentação dos peixes.

O aquário com ID=0 é o protótipo dos aquários, usado para fazer experiências com o movimento dos peixes e testar a lógica por detrás do projeto, contendo 36 alvos e 12 peixes.

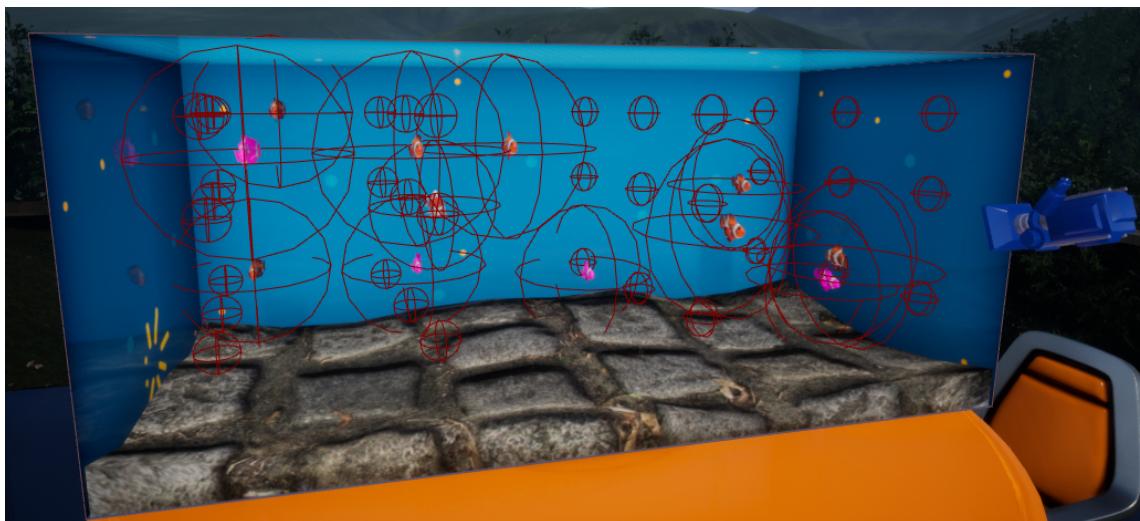


Figura 4.3: Aquário ID=0



Figura 4.4: Aquário ID=0 com a aplicação a correr

O aquário com ID=3 tem como objetivo verificar o padrão de movimento com menos alvos e as posições deles mais irregulares, contendo 16 alvos e 5 peixes.



Figura 4.5: Aquário ID=3

#### 4. RESULTADOS

---



Figura 4.6: Aquário ID=3 com a aplicação a correr

O aquário com ID=2 foi o protótipo usado para controlo do percurso dos peixes através de modificações na matriz de adjacência do aquário na base de dados, limitando o movimento dos peixes a zig zag, contendo 7 alvos e 2 peixes.

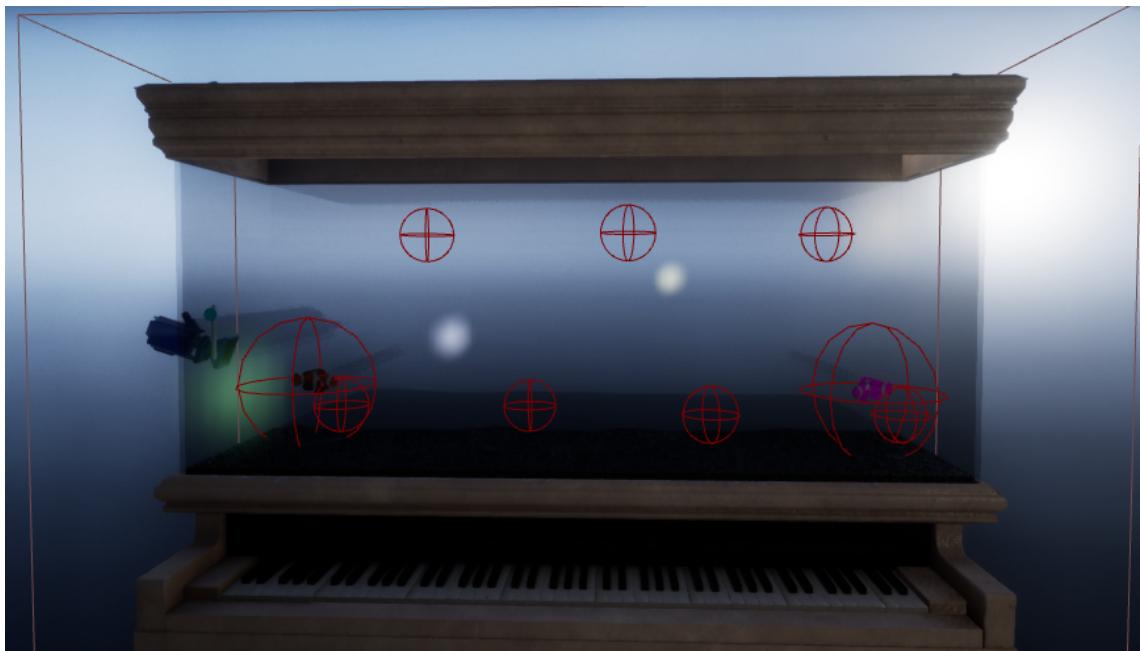


Figura 4.7: Aquário ID=2



Figura 4.8: Aquário ID=2 com a aplicação a correr

O aquário com ID=1 tem como objetivo verificar a possibilidade de movimento com obstáculos dentro do aquário através de verificação e modificação da matriz de adjacência para conter apenas movimentos possíveis, contém 20 alvos e 10 peixes.

#### 4. RESULTADOS

---



Figura 4.9: Aquário ID=1



Figura 4.10: Aquário ID=1 com a aplicação a correr

#### 4.1. Mapa

---

Este aquário também serviu para verificar se não existe problemas em repetir o mesmo aquário, através de cópia do aquário com o mesmo identificador, desta vez apenas com 1 peixe com maior tamanho.

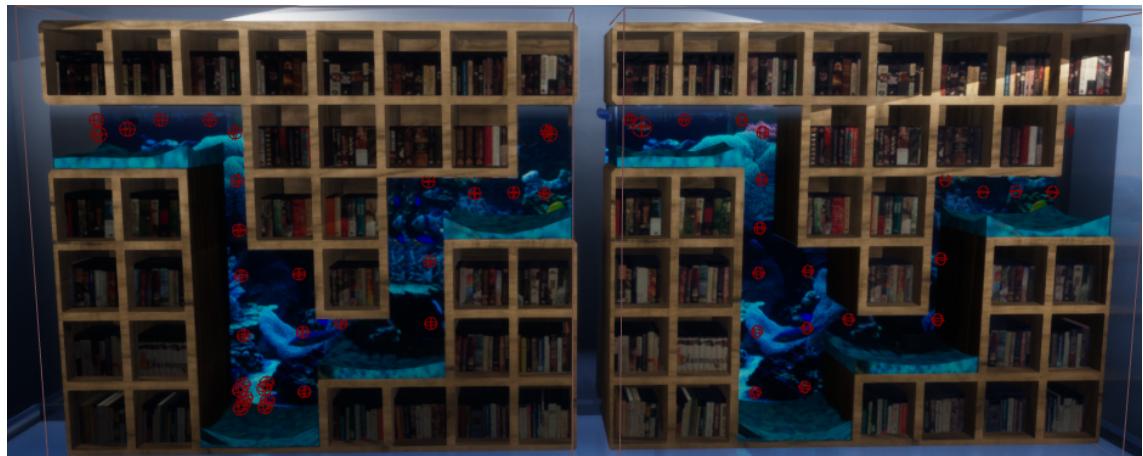


Figura 4.11: Dois Aquários com ID=1

O aquário com ID=4 têm o objetivo de mostrar que as funções criadas para os aquários podem ser reutilizadas com outros objetivos, como pássaros a voar. Contendo 12 alvos e 7 pássaros.

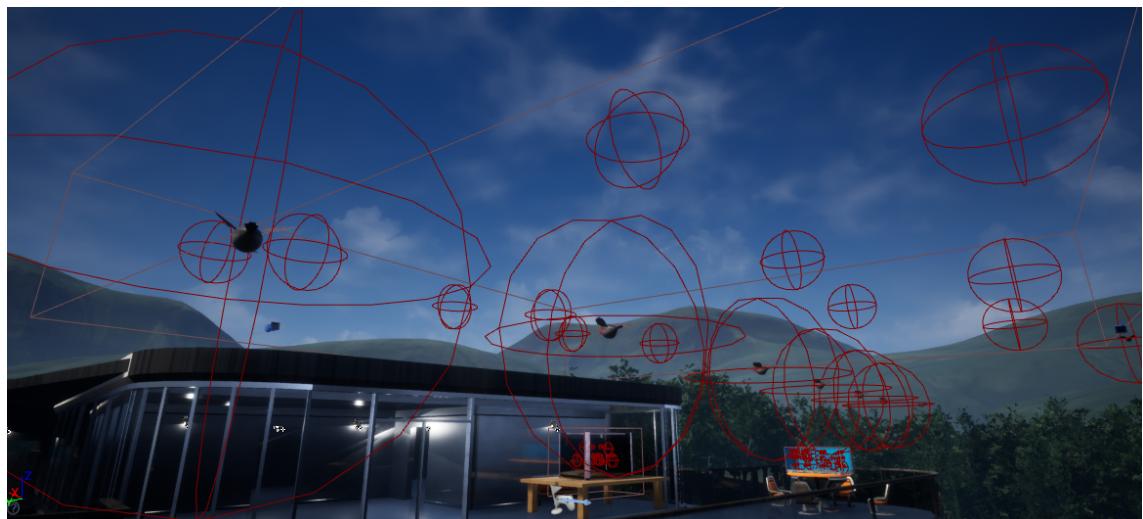


Figura 4.12: Aquário ID=4

#### 4. RESULTADOS

---



Figura 4.13: Aquário ID=4 com a aplicação a correr

## 4.2 Movimento da câmara

Com o objetivo de facilitar a visualização dentro dos vários aquários, criou-se descendentes dos vários *blueprints* de peixes e acrescentou-se câmaras a estes.

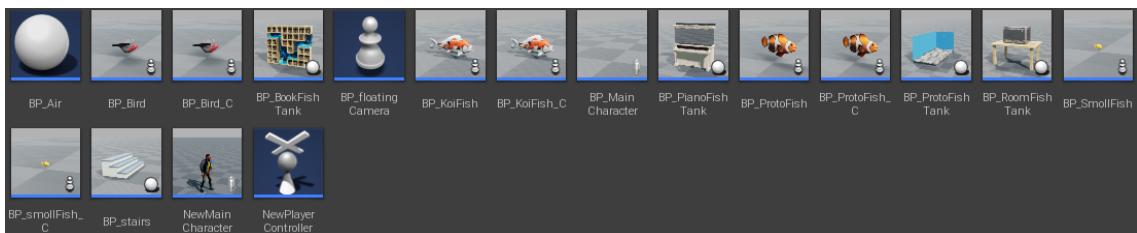


Figura 4.14: Todos os *blueprints* dentro do projeto

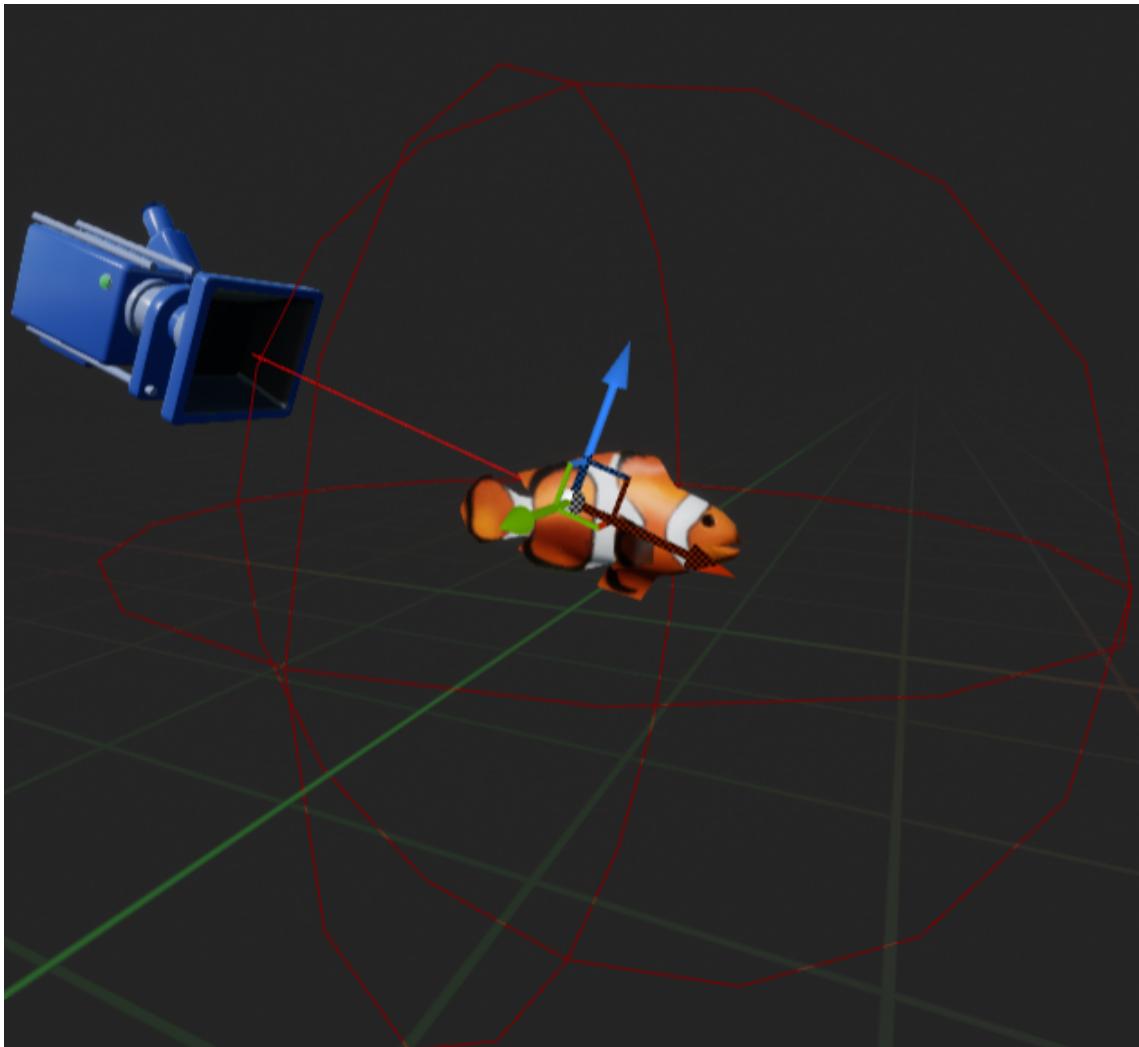


Figura 4.15: **BP\_ProtoFish\_C**

Possibilitando ao utilizador entrar dentro da câmara, que simboliza o ecrã do utilizador.

Através de pressionar as teclas entre 1 e 7, servindo estas como *trigger* para o utilizador possuir a câmara do objeto definido, sendo o 1 para a personagem e as outras teclas para um peixe de cada aquário. Tendo sido utilizado os *blueprint* dentro do *playercontroller*(controlador de jogador) como se mostra na figura 4.16.

#### 4. RESULTADOS

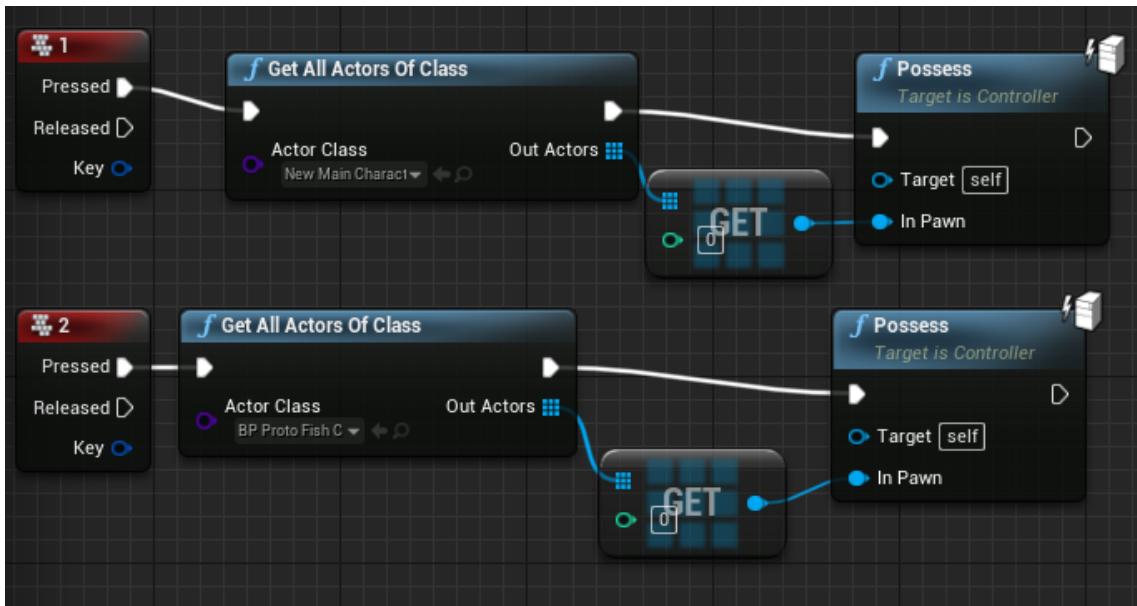


Figura 4.16: Parte do *playercontroller* para definir a câmera a possuir

### 4.3 Textura

Para aumentar a beleza e diversidade dos peixes sem ter de criar novos modelos, decidiu-se criar novas texturas, através da alteração da textura do modelo original. Na figura 4.17, a modificação é feita através da verificação se o componente da cor verde de cada pixel se encontra com um valor maior que 0,344 (com escala de 0 até 1), transformando esses para a nova cor decidida.

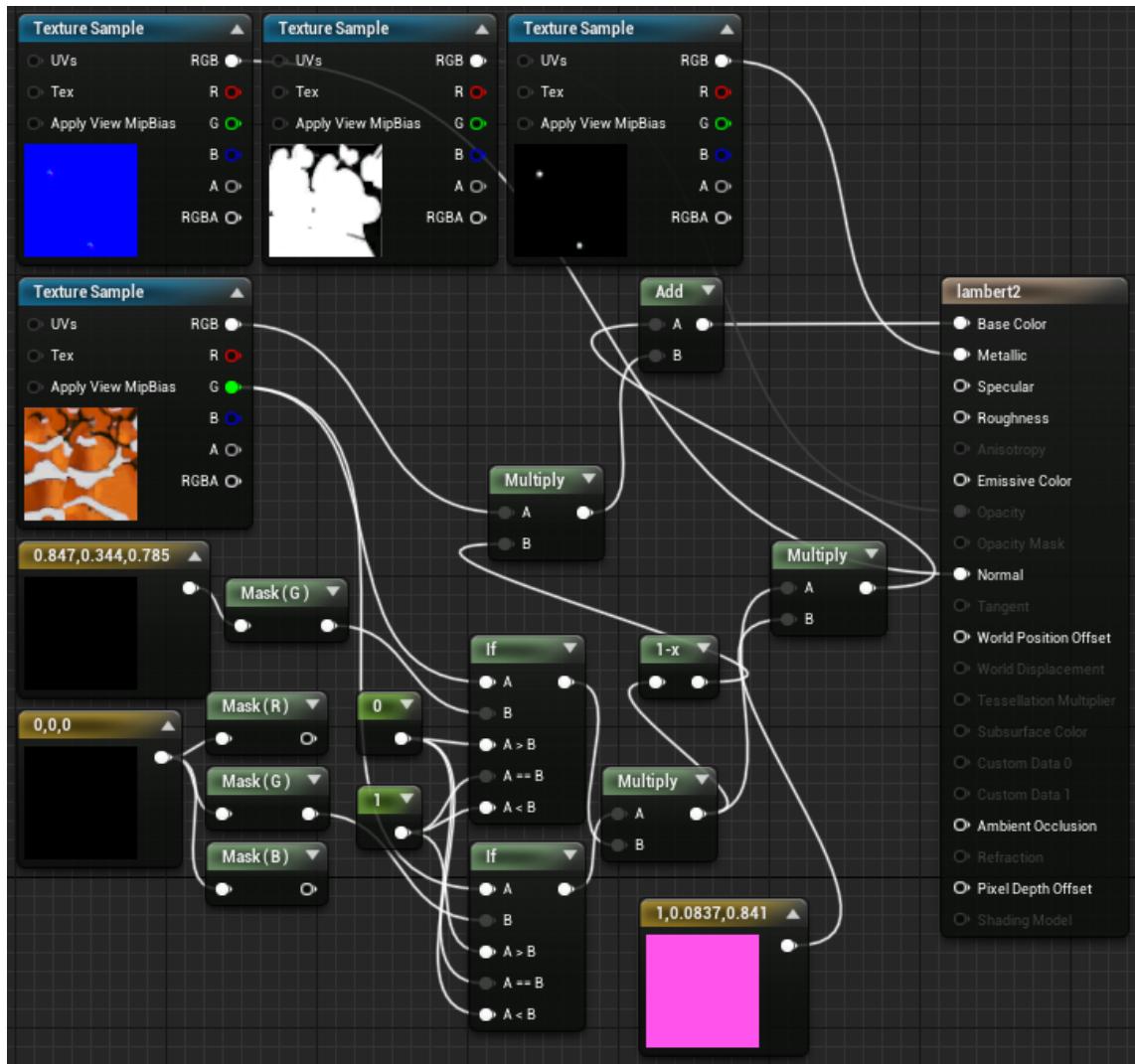


Figura 4.17: *Blueprint* para a modificação da cor da textura original do **ProtoFish**

#### 4. RESULTADOS

---



Figura 4.18: Textura original à esquerda e a nova textura à direita

Também foram criadas 8 novas texturas para o **BP\_SmallFish** em que simplesmente se multiplicou o valor da cor original de cada pixel por um valor, resultando em novas texturas com um filtro por cima.

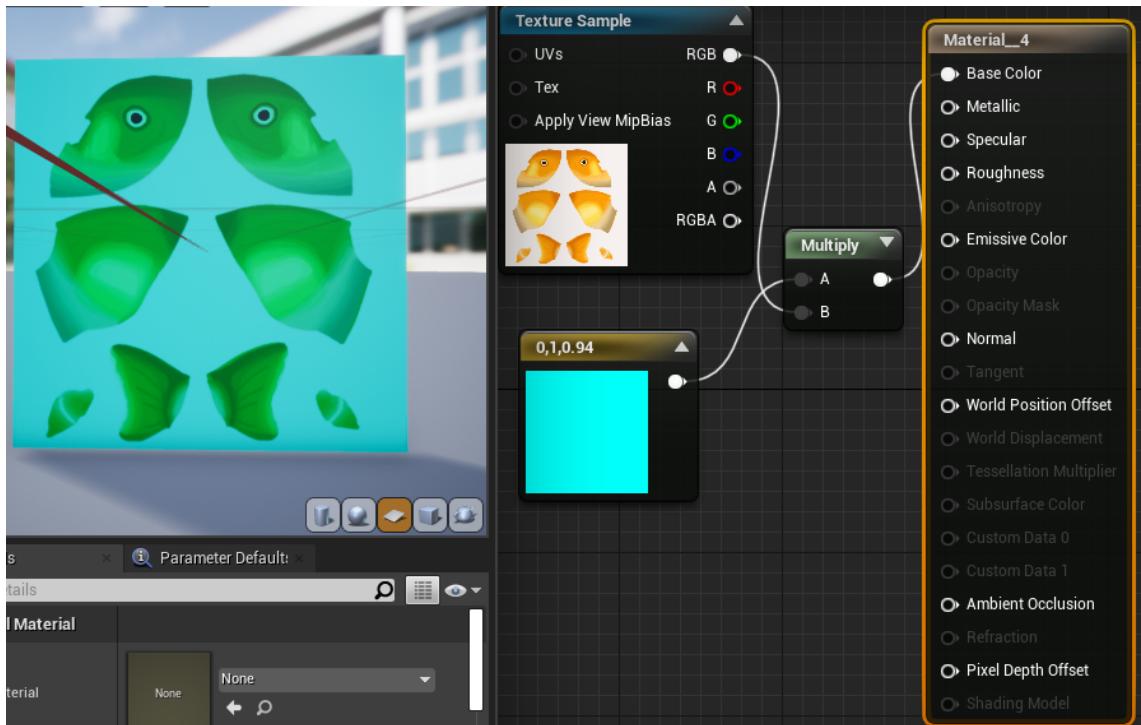


Figura 4.19: *Blueprint* para a modificação da cor da textura original do **SmollFish**, à esquerda a nova textura e à direita a original



Figura 4.20: Texturas criadas com este método

## 4.4 Animação

Devido ao modelo **BP\_KoiFish** conter problemas na conversão para formato .fbx, houve a necessidade de criar uma nova animação para o movimento deste peixe através da definição da posição dos vários ossos do esqueleto durante uma certa duração da animação.

#### 4. RESULTADOS

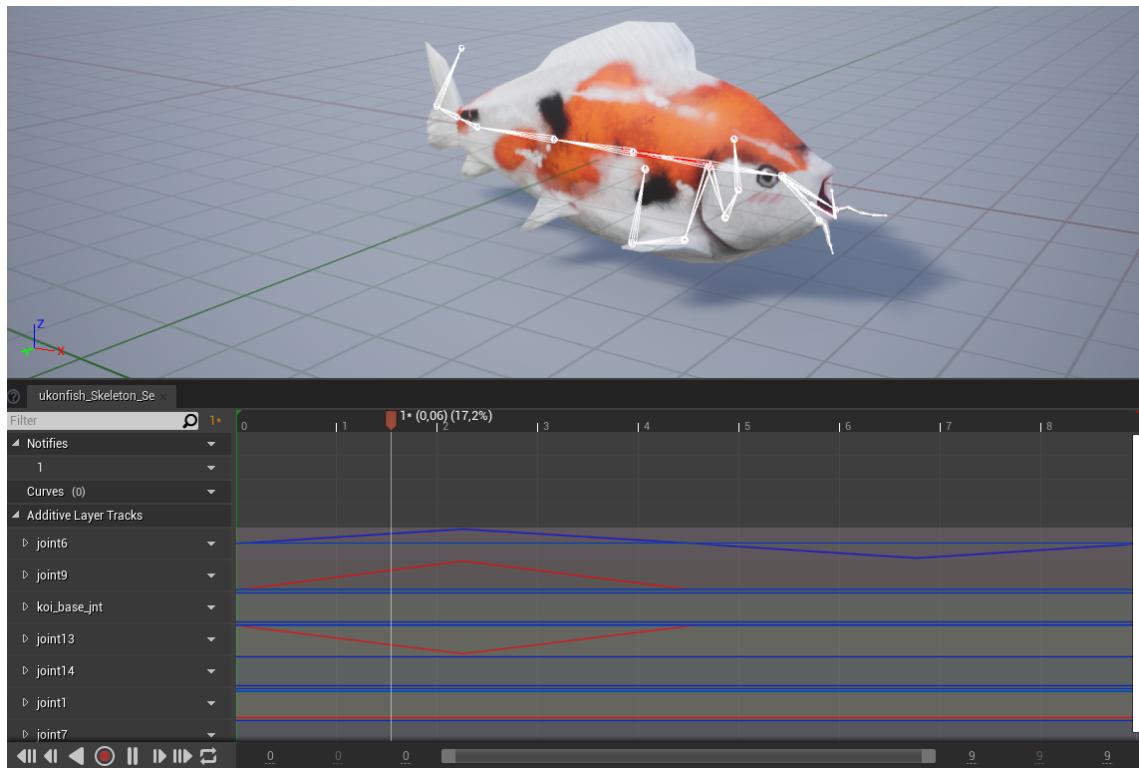


Figura 4.21: Animação de movimento para o **BP\_KoiFish**

### 4.5 Controlos

Para o utilizador conseguir ter controlo da aplicação, é necessário definir todos os *inputs* aceitáveis e o seu valor, sendo estes chamados dentro de funções que definem o que estes fazem, através de ir ao *Project Settings* —> *Engine* —> *Input* no editor do UE4 e acrescentar uma ação, as teclas que ativam essa ação e no caso de ser para movimento o seu valor, sendo o valor multiplicativo nas funções de movimento, onde se o valor for negativo faz o movimento oposto.

Os *inputs* existentes nesta aplicação encontram-se na figura 4.22

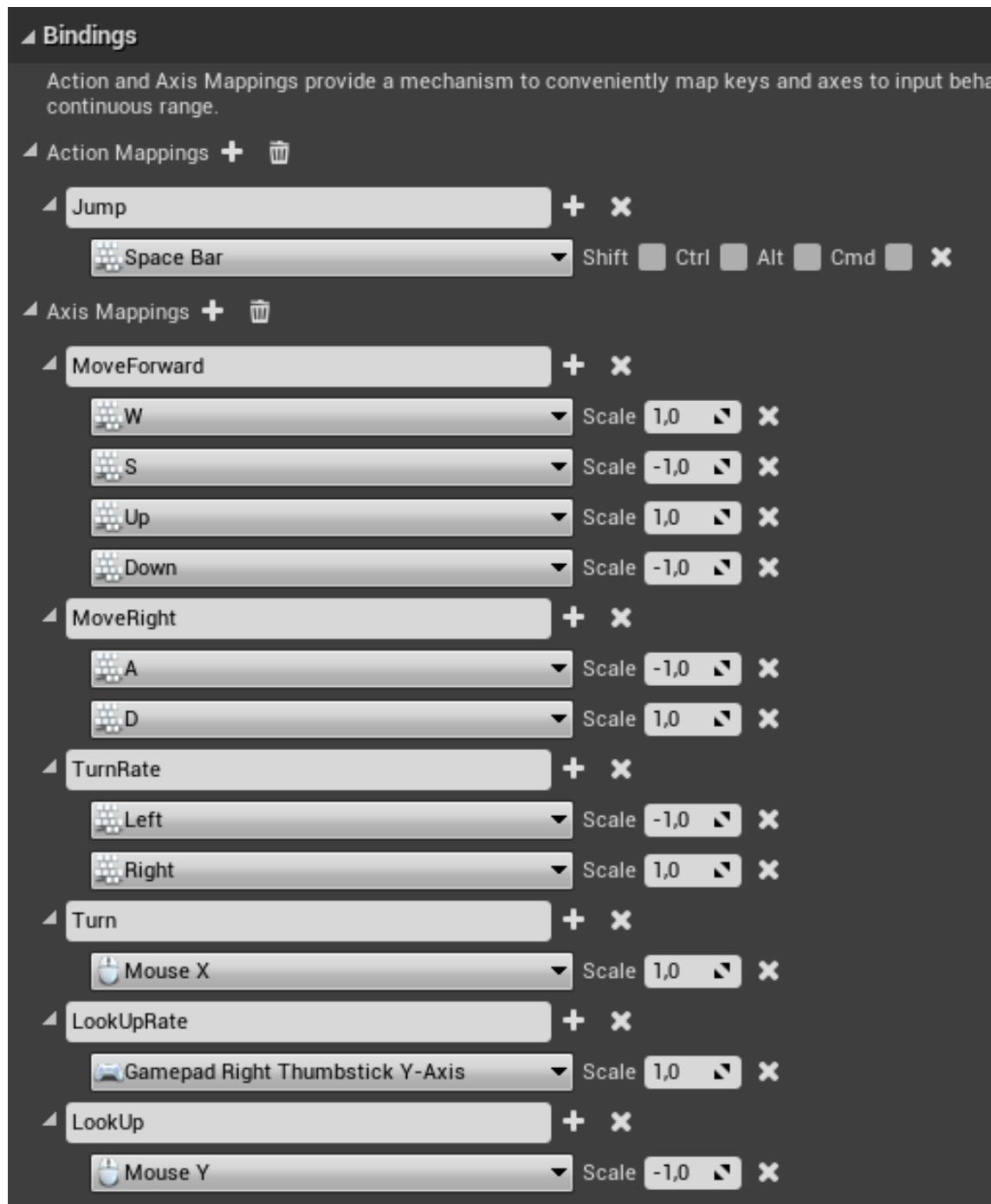


Figura 4.22: Inputs possíveis dentro do jogo



# Capítulo 5

## Conclusão

Neste projeto, os objetivos principais foram o uso de UE4, o uso CUDA, o uso de IA, a ligação das tecnologias usadas, existência de objetos do tipo peixes a movimentarem-se de uma forma realista, fazer com que os objetos do tipo peixes não saiam de dentro objetos do tipo aquário nem fiquem presos nas paredes desse mesmo. Todos estes objetivos foram cumpridos, mesmo certos desses terem tido menor importância na totalidade do projeto do que esperado, implementados de uma forma completamente diferente do que o previsto, mais especificamente a IA, o CUDA.

Em relação à IA a expectativa era o uso de *behavior tree*, devido a não esperar este necessitar que os objetos tenham de estar em cima de uma superfície resultando em ter de utilizar um modelo de IA do scikit-learn.

Em relação ao CUDA a expectativa inicial era de estar em quase todas as funções, mas devido a falta de incompatibilidade nativa com o UE4 e da maior parte das funções fora do UE4 acabarem por perderem performance ou não aceitarem o uso de CUDA.

Também acabou-se por utilizar várias tecnologias que não estavam planeadas, como o Firebase e o Python e todas as bibliotecas relacionadas com o Python, para tapar os buracos causados por as varias incompatibilidades e falhas na lógica inicial.

As principais dificuldades que existiram no desenvolvimento de este projeto estão relacionadas com o desconhecimento e a inexperiência sobre as tecnologias usadas. Para resolver esta questão foi pesquisado informação, tais como tutoriais, documentos e cursos online, especialmente sobre UE4, ligação entre C++ e Python e CUDA. Destacar a obtenção de diploma de formação em CUDA da NVIDIA. Em relação UE4, surgiram dificuldades devido a não existir muita informação relevante para o projeto realizado, embora existam muitos exemplos e vídeos a ilustrar programas

## 5. CONCLUSÃO

---

logicamente parecidos com este projeto, mas não explicando como funcionam e poucos explicam a lógica em si. Outra complicação foi a utilização de CUDA neste projeto, em virtude de este ser incompatível com o UE4 e mesmo depois de se saber como ligar o CUDA e UE4 continuou a existir muitas limitações devido à quantidade limitada de informação que se consegue passar para o CUDA do UE4 e os tipos de dados aceites dentro de Kernel do CUDA, implicando que o CUDA passou a ter menor peso no projeto do que foi inicialmente pretendido.

Outro dos obstáculos foi fazer a ligação entre Python e C++, o que se encontra é como correr funções de C++ em Python e não scripts de Python em C++, tendo sido necessário uma grande quantidade de tentativas e erro só para conseguir colocar os scripts de Python, que chamam funções de uma biblioteca e retorna valores, a correr em C++. Seguindo-se o problema de encontrar a pasta de trabalho de um projeto de UE4 para colocar os scripts de Python e descobrir como inserir as dependências necessárias para chamar o Python (python.lib), sendo esta última facilitada devido a ter conseguido fazer a ligação da lógica necessária neste problema com a solução do CUDA.

Em suma, a criação de novos projetos deste género, com a experiência adquirida, estaria bastante simplificada pois a pesquisa de informação e desenvolvimento demoraria menos tempo, pois aprendi muito neste projeto especialmente de como conseguir procurar informação, a resolução e descoberta de *bugs* e como fazer ligação entre várias tecnologias que à partida são incompatíveis, contornar problemas "impossíveis" de resolver e como verificar os prós e os contras de uma função ou correção de *bug*(tempo necessário vs recompensa).

## 5.2 Trabalho futuro

O projeto obtido ainda têm várias possibilidades de ser melhorada nestes seguintes aspetos:

- Remover a limitação do movimento do peixes que esta dependente dos vários alvos, possibilitando a melhoraria do modelo de IA para criar percursos ainda mais parecidos com a realidade.
- Aumentar a performance da aplicação, através de ir mais a fundo nas varias ferramentas oferecidas pelo UE4.
- Corrigir os bugs de colisão que podem acontecer, através de utilização de *watchdogs* que verificam se os vários objetos do tipo peixes estão presos ou

fora do aquário.

- Adquirir/criar novos modelos 3D.
- Melhorar a performance dos *Kernels* CUDA



# Bibliografia

- [1] U. Technologies. Unity. [Online]. Disponível: <https://unity.com> (citado na pág. 5)
- [2] I. Epic Games. Unreal engine. [Online]. Disponível: <https://www.unrealengine.com/en-US/> (citado nas págs. 5 e 14)
- [3] PONTYPANTS. (2020, mar) Unity vs unreal – which engine should you choose? [Online]. Disponível: <https://sundaysundae.co/unity-vs-unreal/> (citado nas págs. 5 e 6)
- [4] D. Buckley. (2020, oct) Unity vs. unreal – choosing a game engine. [Online]. Disponível: <https://gamedevacademy.org/unity-vs-unreal/> (citado nas págs. 5 e 6)
- [5] B. Tristem. Unity vs unreal – which game engine is best for you? [Online]. Disponível: <https://blog.udemy.com/unity-vs-unreal-which-game-engine-is-best-for-you/> (citado nas págs. 5 e 6)
- [6] N. Corporation. Cuda toolkit. [Online]. Disponível: <https://developer.nvidia.com/cuda-toolkit> (citado na pág. 11)
- [7] S. C. Foundation. C++. [Online]. Disponível: <https://isocpp.org> (citado na pág. 11)
- [8] P. S. Foundation. Python. [Online]. Disponível: <https://www.python.org> (citado na pág. 11)
- [9] I. GitHub. Github. [Online]. Disponível: <https://github.com> (citado na pág. 11)
- [10] Sketchfab. sketchfab. [Online]. Disponível: <https://sketchfab.com/> (citado na pág. 11)
- [11] B. Foundation. Blender. [Online]. Disponível: <https://www.blender.org> (citado nas págs. 11 e 19)

## BIBLIOGRAFIA

---

- [12] Google. Firebase. [Online]. Disponível: <https://firebase.google.com> (citado na pág. 11)
- [13] Microsoft. Visual studio 2019. [Online]. Disponível: <https://visualstudio.microsoft.com/vs/> (citado na pág. 11)
- [14] E. G. Tom Looman. (2020, may) Unreal engine 4 mastery: Create multiplayer games with c++. [Online]. Disponível: <https://www.udemy.com/course/unrealengine-cpp/> (citado na pág. 13)
- [15] P. L. Newton. (2016, apr) Flying a.i. project. [Online]. Disponível: <https://forums.unrealengine.com/community/community-content-tools-and-tutorials/11886-updated-5-16-a-i-templates-bot-car-flying-ai?25073-DOWNLOAD-A-I-Templates-Community-Project-Bot-Car-amp-Flying-AI=> (citado na pág. 14)
- [16] RunemarkStudio. (2016) Redcoat robin download. [Online]. Disponível: <https://sketchfab.com/3d-models/redcoat-robin-download-6f81ad29c150407c84670c84d4f216c3> (citado na pág. 20)
- [17] P. Krensel. (2016) Fish animated. [Online]. Disponível: <https://sketchfab.com/3d-models/fish-animated-665ab73155a44d91908519d84e5fd002> (citado na pág. 20)
- [18] lesliestowe. (2019) Koi fish. [Online]. Disponível: <https://sketchfab.com/3d-models/koi-fish-f7e2e4858f2f438aa2832566220199f4> (citado na pág. 20)
- [19] Koppenhaver. (2017) Aquarium piano. [Online]. Disponível: <https://sketchfab.com/3d-models/aquarium-piano-c3d8ee6391af42f68df9501c0bf84d75> (citado na pág. 20)
- [20] TRYFIELD. (2018) Aquarium bookcase. [Online]. Disponível: <https://sketchfab.com/3d-models/aquarium-bookcase-c8e90f34d7854ad189aa6d50a8dd36d1> (citado na pág. 20)
- [21] NeoT1. (2018) Room aquarium (now animated). [Online]. Disponível: <https://sketchfab.com/3d-models/room-aquarium-now-animated-3d2177c3e90a4379b3484d811c013284> (citado na pág. 20)

- [22] 3DHaupt. (2013) Futuristic house model wip 2. [Online]. Disponível: <https://sketchfab.com/3d-models/futuristic-house-model-wip-2-b55c5898b9ed4b6081b68c73b655d62e> (citado na pág. 20)
- [23] lennartburgold. (2018) Aquarium. [Online]. Disponível: <https://sketchfab.com/3d-models/aquarium-e954de85f1f64c7eb5657c8f644fc066> (citado na pág. 20)
- [24] misaooo. (2017) Clownfish. [Online]. Disponível: <https://sketchfab.com/3d-models/lownfish-68e9afcb79534dc7a2316d780994e619> (citado na pág. 20)
- [25] SCIEMENT. (2017, oct) [c++][cuda][ue4]cuda unreal engine 4. [Online]. Disponível: [http://www.sciement.com/tech-blog/c/cuda\\_in\\_ue4/](http://www.sciement.com/tech-blog/c/cuda_in_ue4/) (citado na pág. 23)
- [26] scikit-learn developers. scikit-learn. [Online]. Disponível: <https://scikit-learn.org/stable/> (citado na pág. 24)
- [27] P. S. Foundation. (2020, oct) Python/c api reference manual. [Online]. Disponível: <https://docs.python.org/3/c-api/index.html> (citado na pág. 24)
- [28] Anaconda. Numba. [Online]. Disponível: <http://numba.pydata.org> (citado na pág. 29)
- [29] Google. (2019) google-cloud-firebase. [Online]. Disponível: <https://googleapis.dev/python/firebase/latest/client.html> (citado na pág. 30)
- [30] NumFOCUS. pandas. [Online]. Disponível: <https://pandas.pydata.org> (citado na pág. 35)



## **Anexos**



# Anexo I

## Código Desenvolvido

UE4:

---

```
1 // Copyright Epic Games, Inc. All Rights Reserved.
2
3 using UnrealBuildTool;
4 using System.IO;
5
6 public class UE4AquaSim : ModuleRules
7 {
8     private string project_root_path
9     {
10         get { return Path.Combine(ModuleDirectory, "../.."); }
11     }
12
13     public UE4AquaSim(ReadOnlyTargetRules Target) : base(Target)
14     {
15         PCHUsage = PCHUsageMode.UseExplicitOrSharedPCHs;
16
17         PublicDependencyModuleNames.AddRange(new string[] { "Core", "CoreUObject", "Eng
18
19         PrivateDependencyModuleNames.AddRange(new string[] { });
20         //biblioteca estatica
21         string custom_fishpath_lib_include = "FishPathLib/include";
22         string custom_fishpath_lib_lib = "FishPathLib/lib";
23
24         PublicIncludePaths.Add(Path.Combine(project_root_path, custom_fishpath_lib_incl
25         PublicAdditionalLibraries.Add(Path.Combine(project_root_path, custom_fishpath_l
26         //cuda
27         string cuda_path = "C:/Program Files/NVIDIA GPU Computing Toolkit/CUDA/v11.0";
28         string cuda_include = "include";
```

## I. CÓDIGO DESENVOLVIDO

---

```
29         string cuda_lib = "lib/x64";
30
31         PublicIncludePaths.Add(Path.Combine(cuda_path, cuda_include));
32
33         PublicAdditionalLibraries.Add(Path.Combine(cuda_path, cuda_lib, "cudart_static.lib"));
34         //python
35         string python_path = "C:/Users/fresc/AppData/Local/Programs/Python/Python38";
36         string python_include = "include";
37         string python_lib = "libs";
38
39         PublicIncludePaths.Add(Path.Combine(python_path, python_include));
40
41         PublicAdditionalLibraries.Add(Path.Combine(python_path, python_lib, "python38.lib"));
42         // Uncomment if you are using Slate UI
43         // PrivateDependencyModuleNames.AddRange(new string[] { "Slate", "SlateCore" });
44
45         // Uncomment if you are using online features
46         // PrivateDependencyModuleNames.Add("OnlineSubsystem");
47
48         // To include OnlineSubsystemSteam, add it to the plugins section in your uproject f
49     }
50 }
```

---

Listing 1: UE4AquaSim.Build.cs

```
1 // Copyright Epic Games, Inc. All Rights Reserved.
2
3 #pragma once
4
5 #include "CoreMinimal.h"
```

---

Listing 2: UE4AquaSim.h

```
1 // Copyright Epic Games, Inc. All Rights Reserved.
2
3 #include "UE4AquaSim.h"
4 #include "Modules/ModuleManager.h"
5
6 IMPLEMENT_PRIMARY_GAME_MODULE( FDefaultGameModuleImpl, UE4AquaSim, "UE4AquaSim" );
```

---

Listing 3: UE4AquaSim.cpp

---

---

```
1 // Fill out your copyright notice in the Description page of Project Settings.
2
3 #pragma once
4
5 #include "CoreMinimal.h"
6 #include "GameFramework/Character.h"
7 #include "AS_MainCharacter.generated.h"
8
9 class UInputComponent;
10
11 UCLASS()
12 class UE4AQUASIM_API AAS_MainCharacter : public ACharacter
13 {
14     GENERATED_BODY()
15
16 public:
17     // Sets default values for this character's properties
18     AAS_MainCharacter();
19
20 protected:
21
22     UPROPERTY(VisibleDefaultsOnly, Category = Mesh)
23     class USkeletalMeshComponent* Mesh1P;
24
25     UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = Camera, meta = (AllowPrivateAccess))
26     class UCameraComponent* FirstPersonCameraComponent;
27     // Called when the game starts or when spawned
28     virtual void BeginPlay() override;
29
30     /** Handles moving forward/backward */
31     void MoveForward(float Val);
32
33     /** Handles strafing movement, left and right */
34     void MoveRight(float Val);
35
36     /**
37      * Called via input to turn at a given rate.
38      * @param Rate      This is a normalized rate, i.e. 1.0 means 100% of desired turn rate
39      */
40     void TurnAtRate(float Rate);
41
42     /**
43      * Called via input to turn look up/down at a given rate.
44      * @param Rate      This is a normalized rate, i.e. 1.0 means 100% of desired turn rate
45      */
```

## I. CÓDIGO DESENVOLVIDO

---

```
45     */
46     void LookUpAtRate(float Rate);
47
48     // Called to bind functionality to input
49     virtual void SetupPlayerInputComponent(class UInputComponent* PlayerInputComponent) override
50
51 public:
52
53
54
55     /** Base turn rate, in deg/sec. Other scaling may affect final turn rate. */
56     UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category=Camera)
57     float BaseTurnRate;
58
59     /** Base look up/down rate, in deg/sec. Other scaling may affect final rate. */
60     UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category=Camera)
61     float BaseLookUpRate;
62
63     /** Returns Mesh1P subobject */
64     FORCEINLINE class USkeletalMeshComponent* GetMesh1P() const { return Mesh1P; }
65     /** Returns FirstPersonCameraComponent subobject */
66     FORCEINLINE class UCameraComponent* GetFirstPersonCameraComponent() const { return FirstPersonCameraComponent; }
67 }
```

---

Listing 4: AS\_MainCharacter.h

```
1 // Fill out your copyright notice in the Description page of Project Settings.
2
3
4 #include "AS_MainCharacter.h"
5 #include "Animation/AnimInstance.h"
6 #include "Camera/CameraComponent.h"
7 #include "Components/CapsuleComponent.h"
8 #include "Components/InputComponent.h"
9 #include "GameFramework/InputSettings.h"
10
11 // Sets default values
12 AAS_MainCharacter::AAS_MainCharacter()
13 {
14
15     GetCapsuleComponent()->InitCapsuleSize(55.f, 96.0f);
16
17     // set our turn rates for input
```

---

```

18     BaseTurnRate = 45.f;
19     BaseLookUpRate = 45.f;
20
21     // Create a CameraComponent
22     FirstPersonCameraComponent = CreateDefaultSubobject<UCameraComponent>(TEXT("FirstPerson"));
23     FirstPersonCameraComponent->SetupAttachment(GetCapsuleComponent());
24     FirstPersonCameraComponent->SetRelativeLocation(FVector(-39.56f, 1.75f, 64.f)); // Position
25     FirstPersonCameraComponent->bUsePawnControlRotation = true;
26
27     // Create a mesh component that will be used when being viewed from a '1st person' view
28     Mesh1P = CreateDefaultSubobject<USkeletalMeshComponent>(TEXT("CharacterMesh1P"));
29     Mesh1P->SetOnlyOwnerSee(true);
30     Mesh1P->SetupAttachment(FirstPersonCameraComponent);
31     Mesh1P->bCastDynamicShadow = false;
32     Mesh1P->CastShadow = false;
33     Mesh1P->SetRelativeRotation(FRotator(1.9f, -19.19f, 5.2f));
34     Mesh1P->SetRelativeLocation(FVector(-0.5f, -4.4f, -155.7f));
35 }
36
37 // Called when the game starts or when spawned
38 void AAS_MainCharacter::BeginPlay()
39 {
40     Super::BeginPlay();
41
42 }
43
44
45 // Called to bind functionality to input
46 void AAS_MainCharacter::SetupPlayerInputComponent(UInputComponent* PlayerInputComponent)
47 {
48     // set up gameplay key bindings
49     check(PlayerInputComponent);
50
51     // Bind jump events
52     PlayerInputComponent->BindAction("Jump", IE_Pressed, this, &ACharacter::Jump);
53     PlayerInputComponent->BindAction("Jump", IE_Released, this, &ACharacter::StopJumping);
54
55     // Bind movement events
56     PlayerInputComponent->BindAxis("MoveForward", this, &AAS_MainCharacter::MoveForward);
57     PlayerInputComponent->BindAxis("MoveRight", this, &AAS_MainCharacter::MoveRight);
58
59     // We have 2 versions of the rotation bindings to handle different kinds of devices differently
60     // "turn" handles devices that provide an absolute delta, such as a mouse.
61     // "turnrate" is for devices that we choose to treat as a rate of change, such as an analog stick
62     PlayerInputComponent->BindAxis("Turn", this, &APawn::AddControllerYawInput);

```

## I. CÓDIGO DESENVOLVIDO

---

```
63     PlayerInputComponent->BindAxis("TurnRate", this, &AAS_MainCharacter::TurnAtRate);
64     PlayerInputComponent->BindAxis("LookUp", this, &APawn::AddControllerPitchInput);
65     PlayerInputComponent->BindAxis("LookUpRate", this, &AAS_MainCharacter::LookUpAtRate);
66
67 }
68 }
69
70
71 void AAS_MainCharacter::MoveForward(float Value)
72 {
73     if (Value != 0.0f)
74     {
75         // add movement in that direction
76         AddMovementInput(GetActorForwardVector(), Value);
77     }
78 }
79
80 void AAS_MainCharacter::MoveRight(float Value)
81 {
82     if (Value != 0.0f)
83     {
84         // add movement in that direction
85         AddMovementInput(GetActorRightVector(), Value);
86     }
87 }
88
89 void AAS_MainCharacter::TurnAtRate(float Rate)
90 {
91     // calculate delta for this frame from the rate information
92     AddControllerYawInput(Rate * BaseTurnRate * GetWorld()->GetDeltaSeconds());
93 }
94
95 void AAS_MainCharacter::LookUpAtRate(float Rate)
96 {
97     // calculate delta for this frame from the rate information
98     AddControllerPitchInput(Rate * BaseLookUpRate * GetWorld()->GetDeltaSeconds());
99 }
```

---

Listing 5: AS\_MainCharacter.cpp

---

```
1 // Copyright Epic Games, Inc. All Rights Reserved.
2
3 #pragma once
```

---

```
4
5 #include "CoreMinimal.h"
6 #include "GameFramework/GameModeBase.h"
7 #include "UE4AquaSimGameModeBase.generated.h"
8
9 /**
10 *
11 */
12 UCLASS()
13 class UE4AQUASIM_API AUE4AquaSimGameModeBase : public AGameModeBase
14 {
15     GENERATED_BODY()
16 public:
17     AUE4AquaSimGameModeBase();
18 };
```

---

Listing 6: UE4AquaSimGameModeBase.h

---

```
1 // Copyright Epic Games, Inc. All Rights Reserved.
2
3
4 #include "UE4AquaSimGameModeBase.h"
5 #include "UObject/ConstructorHelpers.h"
6 #include "Public/AS_MainCharacter.h"
7
8 AUE4AquaSimGameModeBase::AUE4AquaSimGameModeBase()
9     : Super()
10 {
11     // set default pawn class to our Blueprinted character
12     static ConstructorHelpers::FClassFinder<APawn> PlayerPawnClassFinder(TEXT("/Game/AquaSi
13     DefaultPawnClass = PlayerPawnClassFinder.Class;
14
15 }
```

---

Listing 7: UE4AquaSimGameModeBase.cpp

---

```
1 // Fill out your copyright notice in the Description page of Project Settings.
2
3 #pragma once
4
5 #include "CoreMinimal.h"
```

## I. CÓDIGO DESENVOLVIDO

---

```
6 #include "GameFramework/Actor.h"
7 #include "AS_FishTarget.generated.h"
8
9 class USphereComponent;
10
11 UCLASS()
12 class UE4AQUASIM_API AAS_FishTarget : public AActor
13 {
14     GENERATED_BODY()
15
16 public:
17     // Sets default values for this actor's properties
18     AAS_FishTarget();
19     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
20     int Id;
21
22 protected:
23     UPROPERTY(VisibleAnywhere, Category = "Components")
24     USphereComponent* Spherecomponent;
25
26 };
```

---

Listing 8: AS\_FishTarget.h

```
1 // Fill out your copyright notice in the Description page of Project Settings.
2
3
4 #include "AS_FishTarget.h"
5 #include <Components/SphereComponent.h>
6
7 // Sets default values
8 AAS_FishTarget::AAS_FishTarget()
9 {
10     Spherecomponent = CreateDefaultSubobject<USphereComponent>(TEXT("SphereComp"));
11     Spherecomponent->SetCollisionEnabled(ECollisionEnabled::QueryOnly);
12     Spherecomponent->SetGenerateOverlapEvents(true);
13     //Spherecomponent->SetHiddenInGame(false);
14     Spherecomponent->SetSphereRadius(100.f);
15     RootComponent = Spherecomponent;
16
17 }
```

---

---

Listing 9: AS\_FishTarget.cpp

---

```
1 // Fill out your copyright notice in the Description page of Project Settings.
2
3 #pragma once
4
5 #include "CoreMinimal.h"
6 #include "GameFramework/Actor.h"
7 #include "AS_FishTank.generated.h"
8
9 class UBoxComponent;
10
11 UCLASS()
12 class UE4AQUASIM_API AAS_FishTank : public AActor
13 {
14     GENERATED_BODY()
15
16 public:
17     // Sets default values for this actor's properties
18     AAS_FishTank();
19
20 protected:
21
22     UPROPERTY(VisibleAnywhere, Category = "Components")
23     UBoxComponent* rangeToSearchForTarget;
24
25     virtual void BeginPlay() override;
26
27 public:
28
29     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
30     TArray<AActor*> targets;
31     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
32     float CurrentForwardSpeed;
33     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
34     float Acceleration;
35     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
36     float TurnSpeed;
37     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
38     float MaxSpeed;
39     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
40     float MinSpeed;
41     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
42     int Id;
```

## I. CÓDIGO DESENVOLVIDO

---

```
43     // Called every frame
44     virtual void Tick(float DeltaTime) override;
45 }
```

---

Listing 10: AS\_FishTank.h

```
1 // Fill out your copyright notice in the Description page of Project Settings.
2
3 #pragma once
4
5 #include "CoreMinimal.h"
6 #include "GameFramework/Actor.h"
7 #include "AS_FishTank.generated.h"
8
9 class UBoxComponent;
10
11 UCLASS()
12 class UE4AQUASIM_API AAS_FishTank : public AActor
13 {
14     GENERATED_BODY()
15
16 public:
17     // Sets default values for this actor's properties
18     AAS_FishTank();
19
20 protected:
21
22     UPROPERTY(VisibleAnywhere, Category = "Components")
23     UBoxComponent* rangeToSearchForTarget;
24
25     virtual void BeginPlay() override;
26
27 public:
28
29     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
30     TArray<AActor*> targets;
31     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
32     float CurrentForwardSpeed;
33     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
34     float Acceleration;
35     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
36     float TurnSpeed;
37     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
```

---

```
38     float MaxSpeed;
39     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
40     float MinSpeed;
41     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
42     int Id;
43     // Called every frame
44     virtual void Tick(float DeltaTime) override;
45 }
```

---

Listing 11: AS\_FishTank.cpp

---

```
1 // Fill out your copyright notice in the Description page of Project Settings.
2
3 #pragma once
4
5 #include "CoreMinimal.h"
6 #include "GameFramework/Pawn.h"
7 #include "Logging/LogMacros.h"
8 #include "AS_Fish.generated.h"
9
10 DECLARE_LOG_CATEGORY_EXTERN(FishLogCategory, Log, All);
11
12 class USphereComponent;
13 class USkeletalMeshComponent;
14 class AAS_FishTank;
15
16 UCCLASS()
17 class UE4AQUASIM_API AAS_Fish : public APawn
18 {
19     GENERATED_BODY()
20
21 public:
22     // Sets default values for this pawn's properties
23     AAS_Fish();
24
25 protected:
26     UPROPERTY(VisibleAnywhere, Category = "Components")
27     USphereComponent* Collisioncomponent;
28
29     UPROPERTY(VisibleAnywhere, Category = "Components")
30     USphereComponent* Spherecomponent;
31
32     UPROPERTY(VisibleDefaultsOnly, Category = "Components")
```

## I. CÓDIGO DESENVOLVIDO

---

```
33     USkeletalMeshComponent* Mesh1P;
34     // Called when the game starts or when spawned
35     virtual void BeginPlay() override;
36
37     void SetPath();
38     UFUNCTION()
39     void OnHit(UPrimitiveComponent* HitComp, AActor* OtherActor, UPrimitiveComponent* OtherComp,
40
41
42     void ApplyForwardSpeedAndSteeringRotation(float DeltaTime);
43
44     void GetSteering(float* X, float* Y, float* Z, float* VelocityWeight);
45
46     void SetForwardSpeedValueUsingThrustInput(float DeltaTime, float X, float VelocityWeight);
47
48     void SetPitchRotationRateUsingDirectionalInput(float DeltaTime, float Z, float VelocityWeight);
49
50     void SetYawSpeedBasedOnTurnInput(float DeltaTime, float Y);
51
52     void AlsoSetRollSpeedBasedOnTurnInput(float DeltaTime, float Y, float Roll);
53
54     UFUNCTION()
55     void HandleOverlapCom(UPrimitiveComponent* OverlappedComponent, AActor* OtherActor, UPrimiti
56
57
58 private:
59
60     int targetArraySize;
61     int PathIndex[500];
62     int PathSize;
63     /** Current yaw speed */
64     float CurrentYawSpeed;
65
66     /** Current pitch speed */
67     float CurrentPitchSpeed;
68
69     /** Current roll speed */
70     float CurrentRollSpeed;
71
72     int32 TargetIndex;
73
74     FVector AvoidVector;
75
76 public:
77     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
```

---

```

78     AAS_FishTank* Aquarium;
79     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
80     TArray<AActor*> targets;
81     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
82     float CurrentForwardSpeed;
83     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
84     float Acceleration;
85     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
86     float TurnSpeed;
87     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
88     float MaxSpeed;
89     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
90     float MinSpeed;
91     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
92     bool blog;
93     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
94     bool updatable;
95     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
96     bool trainMode;
97     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
98     int startpoint;
99     bool gotTargets;
100    // Called every frame
101    virtual void Tick(float DeltaTime) override;
102
103 };

```

---

Listing 12: AS\_Fish.h

---

```

1 // Fill out your copyright notice in the Description page of Project Settings.
2
3 #pragma once
4
5 #include "CoreMinimal.h"
6 #include "GameFramework/Pawn.h"
7 #include "Logging/LogMacros.h"
8 #include "AS_Fish.generated.h"
9
10 DECLARE_LOG_CATEGORY_EXTERN(FishLogCategory, Log, All);
11
12 class USphereComponent;
13 class USkeletalMeshComponent;
14 class AAS_FishTank;

```

---

## I. CÓDIGO DESENVOLVIDO

---

```
15
16 UClass()
17 class UE4AQUASIM_API AAS_Fish : public APawn
18 {
19     GENERATED_BODY()
20
21 public:
22     // Sets default values for this pawn's properties
23     AAS_Fish();
24
25 protected:
26     UPROPERTY(VisibleAnywhere, Category = "Components")
27     USphereComponent* Collisioncomponent;
28
29     UPROPERTY(VisibleAnywhere, Category = "Components")
30     USphereComponent* Spherecomponent;
31
32     UPROPERTY(VisibleDefaultsOnly, Category = "Components")
33     USkeletalMeshComponent* Mesh1P;
34     // Called when the game starts or when spawned
35     virtual void BeginPlay() override;
36
37     void SetPath();
38     UFUNCTION()
39     void OnHit(UPrimitiveComponent* HitComp, AActor* OtherActor, UPrimitiveComponent* OtherComp,
40
41
42     void ApplyForwardSpeedAndSteeringRotation(float DeltaTime);
43
44     void GetSteering(float* X, float* Y, float* Z, float* VelocityWeight);
45
46     void SetForwardSpeedValueUsingThrustInput(float DeltaTime, float X, float VelocityWeight);
47
48     void SetPitchRotationRateUsingDirectionalInput(float DeltaTime, float Z, float VelocityWeight);
49
50     void SetYawSpeedBasedOnTurnInput(float DeltaTime, float Y);
51
52     void AlsoSetRollSpeedBasedOnTurnInput(float DeltaTime, float Y, float Roll);
53
54     UFUNCTION()
55     void HandleOverlapCom(UPrimitiveComponent* OverlappedComponent, AActor* OtherActor, UPrimiti
56
57
58 private:
59
```

---

```
60     int targetArraySize;
61     int PathIndex[5001];
62     int PathSize;
63     /** Current yaw speed */
64     float CurrentYawSpeed;
65
66     /** Current pitch speed */
67     float CurrentPitchSpeed;
68
69     /** Current roll speed */
70     float CurrentRollSpeed;
71
72     int32 TargetIndex;
73
74     FVector AvoidVector;
75
76 public:
77     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
78     AAS_FishTank* Aquarium;
79     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
80     TArray<AActor*> targets;
81     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
82     float CurrentForwardSpeed;
83     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
84     float Acceleration;
85     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
86     float TurnSpeed;
87     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
88     float MaxSpeed;
89     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
90     float MinSpeed;
91     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
92     bool blog;
93     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
94     bool updatable;
95     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
96     bool trainMode;
97     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = "Variable")
98     int startpoint;
99     bool gotTargets;
100    // Called every frame
101    virtual void Tick(float DeltaTime) override;
102
103 };
```

---

## I. CÓDIGO DESENVOLVIDO

---

Listing 13: AS\_Fish.cpp

Biblioteca Estática:

---

```
1 #pragma once
2 #include <Python.h>
3 #include <string>
4
5 namespace FPF {
6     void AddAquariumToDB(int id, int amount);
7     bool verifyAquariumExistence(int id);
8     void RemovePathFromAquarium(int id, int start, int end);
9     char* getShortestFishPath(char* pathToModels, int aquarium_id, int start, int end);
10    void createFishPathModels(int maxWeight, int typeAmount, char* pathToModels);
11    void createFishPathModelsCuda(int maxWeight, int typeAmount, char* pathToModels);
12    void InitializePython();
13    void FinalizePython();
14    int* GetEntireFishPath(int aquariumId, int startPoint);
15    int TestingIFThisWorks();
16    int TestingIFThisWorksPython();
17    std::string get_current_dir();
18 }
```

---

Listing 14: FPFcallPythonScripts.h

---

```
1 #include "FPFcallPythonScripts.h"
2
3
4 void FPF::InitializePython() {
5     if (!Py_IsInitialized()) {
6         //printf("Start");
7         Py_Initialize();
8     }
9 }
10
11 void FPF::FinalizePython() {
12     if (Py_IsInitialized()) {
13         //printf("End");
14         Py_FinalizeEx();
15     }
16 }
```

---

---

```
18 void FPF::AddAquariumToDB(int id, int amount)
19 {
20
21     PyObject* module, * func, * prm;
22
23     FPF::InitializePython();
24
25     PyObject* sys = PyImport_ImportModule("sys");
26     PyObject* path = PyObject_GetAttrString(sys, "path");
27     PyList_Insert(path, 0, PyUnicode_FromString("."));
28
29     module = PyImport_ImportModule("addnewaquariumtodb");
30
31     if (module != 0)
32     {
33         func = PyObject_GetAttrString(module, "main");
34         prm = Py_BuildValue("(ii)", id, amount);
35         PyObject_CallObject(func, prm);
36
37         Py_DECREF(module);
38         Py_DECREF(func);
39         Py_DECREF(prm);
40
41     }
42
43 }
44
45 bool FPF::verifyAquariumExistence(int id)
46 {
47
48     int rel;
49     PyObject* module, * func, * prm, * ret;
50
51     FPF::InitializePython();
52
53     PyObject* sys = PyImport_ImportModule("sys");
54     PyObject* path = PyObject_GetAttrString(sys, "path");
55     PyList_Insert(path, 0, PyUnicode_FromString("."));
56
57     module = PyImport_ImportModule("verifyIfAquariumExistsInDB");
58
59     if (module != 0)
60     {
61         func = PyObject_GetAttrString(module, "main");
62         prm = Py_BuildValue("(i)", id);
```

## I. CÓDIGO DESENVOLVIDO

---

```
63     ret = PyObject_CallObject(func, prm);
64     rel = PyLong_AsLong(ret);
65
66     Py_DECREF(module);
67     Py_DECREF(func);
68     Py_DECREF(prm);
69     Py_DECREF(ret);
70
71 }
72
73 if (rel == 0)
74 {
75     return true;
76 }
77 else
78 {
79     return false;
80 }
81 }
82 }
83
84 void FPF::RemovePathFromAquarium(int id, int start, int end)
85 {
86     PyObject* module, * func, * prm;
87
88     FPF::InitializePython();
89
90     PyObject* sys = PyImport_ImportModule("sys");
91     PyObject* path = PyObject_GetAttrString(sys, "path");
92     PyList_Insert(path, 0, PyUnicode_FromString("."));
93
94     module = PyImport_ImportModule("modifyMatrix");
95
96     if (module != 0)
97     {
98         func = PyObject_GetAttrString(module, "main");
99         prm = Py_BuildValue("(iii)", id, start, end);
100        PyObject_CallObject(func, prm);
101
102        Py_DECREF(module);
103        Py_DECREF(func);
104        Py_DECREF(prm);
105
106    }
107 }
```

---

```
108
109 char* FPF::getShortestFishPath(char* pathToModels, int aquarium_id, int start, int end)
110 {
111     char* result;
112     char* value;
113     PyObject* module, * func, * prm, * ret;
114
115     FPF::InitializePython();
116
117     PyObject* sys = PyImport_ImportModule("sys");
118     PyObject* path = PyObject_GetAttrString(sys, "path");
119     PyList_Insert(path, 0, PyUnicode_FromString("."));
120
121     module = PyImport_ImportModule("getShortestPathFromModel");
122
123     if (module != 0)
124     {
125         func = PyObject_GetAttrString(module, "main");
126         prm = Py_BuildValue("(siii)", pathToModels, aquarium_id, start, end);
127         ret = PyObject_CallObject(func, prm);
128         PyObject* str = PyUnicode_AsEncodedString(ret, "utf-8", "~E~");
129         result = PyBytes_AsString(str);
130
131         Py_DECREF(module);
132         Py_DECREF(func);
133         Py_DECREF(prm);
134         Py_DECREF(ret);
135         Py_XDECREF(str);
136
137         std::string valueString(result);
138         value = &valueString[0];
139     }
140
141     return value;
142 }
143
144
145 void FPF::createFishPathModels(int maxWeight, int typeAmount, char* pathToModels)
146 {
147     PyObject* module, * func, * prm;
148
149     FPF::InitializePython();
150
151     PyObject* sys = PyImport_ImportModule("sys");
152     PyObject* path = PyObject_GetAttrString(sys, "path");
```

## I. CÓDIGO DESENVOLVIDO

---

```
153     PyList_Insert(path, 0, PyUnicode_FromString("."));  
154  
155     module = PyImport_ImportModule("createModel");  
156  
157     if (module != 0)  
158     {  
159         func = PyObject_GetAttrString(module, "main");  
160         prm = Py_BuildValue("(iis)", maxWeight, typeAmount, pathToModels);  
161         PyObject_CallObject(func, prm);  
162  
163         Py_DECREF(module);  
164         Py_DECREF(func);  
165         Py_DECREF(prm);  
166     }  
167  
168     Py_Finalize();  
169 }  
170  
171 void FPF::createFishPathModelsCuda(int maxWeight, int typeAmount, char* pathToModels)  
172 {  
173     PyObject* module, * func, * prm;  
174  
175     FPF::InitializePython();  
176  
177     PyObject* sys = PyImport_ImportModule("sys");  
178     PyObject* path = PyObject_GetAttrString(sys, "path");  
179     PyList_Insert(path, 0, PyUnicode_FromString("."));  
180  
181     module = PyImport_ImportModule("createModelCuda");  
182  
183     if (module != 0)  
184     {  
185         func = PyObject_GetAttrString(module, "main");  
186         prm = Py_BuildValue("(iis)", maxWeight, typeAmount, pathToModels);  
187         PyObject_CallObject(func, prm);  
188  
189         Py_DECREF(module);  
190         Py_DECREF(func);  
191         Py_DECREF(prm);  
192     }  
193  
194     Py_Finalize();  
195 }  
196  
197 }
```

---

```
198
199 int* FPF::GetEntireFishPath(int aquariumId, int startPoint)
200 {
201     int tmpArray[5001];
202
203     PyObject* module, * func, * prm, * ret;
204
205     FPF::InitializePython();
206
207     PyObject* sys = PyImport_ImportModule("sys");
208     PyObject* path = PyObject_GetAttrString(sys, "path");
209     PyList_Insert(path, 0, PyUnicode_FromString("."));
210
211     module = PyImport_ImportModule("getEntireFishPath");
212
213     if (module != 0)
214     {
215         func = PyObject_GetAttrString(module, "main");
216         prm = Py_BuildValue("(ii)", aquariumId, startPoint);
217         ret = PyObject_CallObject(func, prm);
218         if (PyList_Check(ret)) {
219             int count = (int)PyList_Size(ret);
220             printf("count %d\n", count);
221             for (int i = 0; i < count; i++)
222             {
223                 tmpArray[i] = (int)PyLong_AsLong(PyList_GetItem(ret, (Py_ssize_t)i));
224             }
225             tmpArray[5000] = count;
226         }
227         Py_DECREF(module);
228         Py_DECREF(func);
229         Py_DECREF(prm);
230         //Py_DECREF(ret);
231     }
232 }
233
234
235 //FPF::FinalizePython();
236
237 for (int i = 0; i < 10; i++)
238 {
239     printf("position %d value %d\n", i, tmpArray[i]);
240 }
241 printf("count %d\n", tmpArray[5000]);
242 return tmpArray;
```

## I. CÓDIGO DESENVOLVIDO

---

```
243 }
244
245 int FPF::TestingIFThisWorks() {
246     return 13;
247 }
248
249 int FPF::TestingIFThisWorksPython()
250 {
251     int rel;
252     PyObject* module, * func, * prm, * ret;
253
254     FPF::InitializePython();
255
256     PyObject* sys = PyImport_ImportModule("sys");
257     PyObject* path = PyObject_GetAttrString(sys, "path");
258     PyList_Insert(path, 0, PyUnicode_FromString("."));
259
260     module = PyImport_ImportModule("testingifthisworkspython");
261
262     if (module != 0)
263     {
264         func = PyObject_GetAttrString(module, "main");
265         prm = Py_BuildValue("(O)");
266         ret = PyObject_CallObject(func, prm);
267         rel = PyLong_AsLong(ret);
268
269         Py_DECREF(module);
270         Py_DECREF(func);
271         Py_DECREF(prm);
272
273         printf("inside %d", rel);
274     }
275     return rel;
276 }
277
278 #include <direct.h>
279 #define GetCurrentDir _getcwd
280
281 std::string FPF::get_current_dir() {
282     char buff[FILENAME_MAX]; //create string buffer to hold path
283     GetCurrentDir(buff, FILENAME_MAX);
284     std::string current_working_dir(buff);
285
286     return current_working_dir;
287 }
```

---

---

Listing 15: FPFcallPythonScripts.cpp

Python Scripts:

---

```
1 import numpy as np
2 import firebase_admin
3 from firebase_admin import credentials
4 from firebase_admin import firestore
5
6 def createDBConnection():
7     # Use a service account
8     try:
9         firebase_admin.get_app()
10    except ValueError:
11        cred = credentials.Certificate('firebasekey.json')
12        firebase_admin.initialize_app(cred)
13    return firestore.client()
14
15 def createMat(targetAmount):
16     adjMatrix = np.ones((targetAmount, targetAmount), dtype=int)
17     rowSize = len(adjMatrix)
18     colSize = len(adjMatrix[0])
19     for x in range(rowSize):
20         for y in range(colSize):
21             if(x == y):
22                 adjMatrix[x][y] = 0
23     return adjMatrix
24
25 def addMatToDb(id, mat, db, amount):
26     doc_ref = db.collection(u'Aquarium').document(str(id))
27     doc_ref.set({
28         u'size': amount
29     })
30     for x in range(amount):
31         doc_ref.collection(u'Rows').document(str(x)).set({u'array': mat[x]})
32
33 def main(id, amount):
34     db = createDBConnection()
35     mat = createMat(amount)
36     mat = mat.tolist()
37     addMatToDb(id, mat, db, amount)
```

---

## I. CÓDIGO DESENVOLVIDO

---

Listing 16: addnewaquariumtodatabase.py

```
1 import numpy as np
2 import random
3 import pandas as pd
4 from sklearn.svm import LinearSVC
5 from sklearn.pipeline import make_pipeline
6 from sklearn.preprocessing import StandardScaler
7 import firebase_admin
8 from firebase_admin import credentials
9 from firebase_admin import firestore
10 import pickle
11
12 def createDBConnection():
13     # Use a service account
14     try:
15         firebase_admin.get_app()
16     except ValueError:
17         cred = credentials.Certificate('firebasekey.json')
18         firebase_admin.initialize_app(cred)
19     return firestore.client()
20
21 def createWeightMat(typeAmount, maxWeight):
22     weightMat = np.ones((typeAmount,typeAmount), dtype=int)
23     rowSize = len(weightMat)
24     colSize = len(weightMat[0])
25     for x in range(rowSize):
26         for y in range(colSize):
27             weightMat[x][y] = random.randint(1, maxWeight)
28     #print(weightMat)
29     return weightMat
30
31 def createTypeMat(targetAmount, typeAmount):
32     typeMatrix = np.ones((targetAmount), dtype=int)
33     rowSize = len(typeMatrix)
34     for x in range(rowSize):
35         typeMatrix[x] = random.randint(1, typeAmount)
36     #print(typeMatrix)
37     return typeMatrix
38
39 def addWeightToadjMatrix(adjMatrix, weightMat, typeMatrix):
40     rowSize = len(adjMatrix)
41     colSize = len(adjMatrix[0])
42     for x in range(rowSize):
```

---

```

43         xType = typeMatrix[x]
44         for y in range(colSize):
45             if(x == y):
46                 adjMatrix[x][y] = 0
47             else:
48                 yType = typeMatrix[y]
49                 weightOnSpot = weightMat[xType-1][yType-1]
50                 adjMatrix[x][y] = adjMatrix[x][y] * weightOnSpot
51
52     return adjMatrix
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87

```

## I. CÓDIGO DESENVOLVIDO

---

```
88 def dijkstra(graph, src, df):
89
90     row = len(graph)
91     col = len(graph[0])
92
93     dist = [float("Inf")] * row
94
95     parent = [-1] * row
96
97     dist[src] = 0
98
99     queue = []
100    for i in range(row):
101        queue.append(i)
102
103    while queue:
104        u = minDistance(dist, queue)
105
106        queue.remove(u)
107
108        for i in range(col):
109            if graph[u][i] and i in queue:
110                if dist[u] + graph[u][i] < dist[i]:
111                    dist[i] = dist[u] + graph[u][i]
112                    parent[i] = u
113
114    df = printSolution(dist, parent, src, df)
115    return df
116
117 def trainModel(train, path, aquarium_id):
118     train.info()
119     y=train.pop('path')
120
121     p = make_pipeline(StandardScaler(),
122                         LinearSVC(random_state=0, tol=1e-5, max_iter=1000))
123     print('Start Fit:')
124     p.fit(train, y)
125     print('Model Fit:')
126     print(p.predict([[0, 1]]))
127     print('Save Model:')
128     with open(path+str(aquarium_id)+'.sav', 'wb') as f:
129         pickle.dump(p, f)
130
131 def main(maxWeight, typeAmount, path):
```

---

```
133     db = createDBConnection()
134     docs = [snapshot for snapshot in db.collection(u'Aquarium').stream()]
135     for doc in docs:
136         print(f'{doc.id}-----')
137         print('Prepare Mat:')
138         aquarium_id = int(doc.id)
139         size = doc.get('size')
140         list = [None] * size
141         rows = [snapshot for snapshot in doc.reference.collection(u'Rows').stream()]
142         for row in rows:
143             list[int(row.id)] = row.get('array')
144         matrix = np.asarray(list, dtype=np.int16)
145         print(matrix)
146         adjMatrix = prepareWeightedAdjMat(matrix, size, maxWeight, typeAmount)
147         print(adjMatrix)
148         print('Prepare Dataset:')
149         df = pd.DataFrame(columns = ['start', 'end', 'path'])
150         df['start']=df['start'].astype('int')
151         df['end']=df['end'].astype('int')
152         for x in range(len(adjMatrix)):
153             print(f'{x} out of {size}')
154             df = dijkstra(adjMatrix, x, df)
155         print('dataset created')
156         print(df)
157         print('Train Model:')
158         trainModel(df, path, aquarium_id)
159     print('-----')
```

---

Listing 17: createModel.py

---

```
1 import numpy as np
2 import random
3 import pandas as pd
4 from sklearn.linear_model import LogisticRegression
5 import firebase_admin
6 from firebase_admin import credentials
7 from firebase_admin import firestore
8 from firebase_admin import storage
9 import pickle
10 from numba import cuda
11 import numba
12 from numba import types
13 import math
```

## I. CÓDIGO DESENVOLVIDO

---

```
14 import os
15
16 def createDBConnection():
17     # Use a service account
18     try:
19         firebase_admin.get_app()
20     except ValueError:
21         cred = credentials.Certificate('firebasekey.json')
22         firebase_admin.initialize_app(cred, {
23             'storageBucket': 'finalprojectaquarium.appspot.com'
24         })
25     return firestore.client()
26
27 def createWeightMat(typeAmount, maxWeight):
28     weightMat = np.ones((typeAmount,typeAmount), dtype=int)
29     rowSize = len(weightMat)
30     colSize = len(weightMat[0])
31     for x in range(rowSize):
32         for y in range(colSize):
33             weightMat[x][y] = random.randint(1, maxWeight)
34     #print(weightMat)
35     return weightMat
36
37 def createTypeMat(targetAmount, typeAmount):
38     typeMatrix = np.ones((targetAmount), dtype=int)
39     rowSize = len(typeMatrix)
40     for x in range(rowSize):
41         typeMatrix[x] = random.randint(1, typeAmount)
42     #print(typeMatrix)
43     return typeMatrix
44
45 @cuda.jit
46 def addWeightToadjMatrixCuda(adjMatrix, weightMat, typeMatrix):
47     xstart, ystart = cuda.grid(2)
48     stridex, stridey = cuda.gridsize(2)
49     for x in range(xstart, len(adjMatrix), stridex):
50         xType = typeMatrix[x]
51         for y in range(ystart, len(adjMatrix[x]), stridey):
52             if(x == y):
53                 adjMatrix[x][y] = 0
54             else:
55                 yType = typeMatrix[y]
56                 weightOnSpot = weightMat[xType-1][yType-1]
57                 adjMatrix[x][y] = adjMatrix[x][y] * weightOnSpot
58
```

---

```
59
60 def prepareWeightedAdjMat(matrix, size, maxWeight, typeAmount):
61     targetAmount = size
62     weightMat = createWeightMat(typeAmount,maxWeight)
63     typeMatrix = createTypeMat(targetAmount, typeAmount)
64
65     dwm=cuda.to_device(weightMat)
66     dtm=cuda.to_device(typeMatrix)
67     dm=cuda.to_device(matrix)
68
69     threadsperblock = (32, 32)
70     blockspergrid_x = math.ceil(len(dm) / threadsperblock[0])
71     blockspergrid_y = math.ceil(len(dm[0])/ threadsperblock[1])
72     blockspergrid = (blockspergrid_x, blockspergrid_y)
73
74     addWeightToadjMatrixCuda[blockspergrid, threadsperblock](dm, dwm, dtm)
75
76     return dm
77
78
79 @cuda.jit(device=True)
80 def verifyQueue(queue):
81     for element in queue:
82         if element != -1:
83             return True
84     return False
85
86 @cuda.jit(device=True)
87 def minDistanceCuda(dist,queue, row):
88     minimum = np.inf
89     min_index = -1
90
91     for i in range(row):
92         if dist[i] < minimum and queue[i] != -1:
93             minimum = dist[i]
94             min_index = i
95     return min_index
96
97 @cuda.jit(device=True)
98 def addPathCuda(parent,i,dest, src):
99     j=i
100    dest[src][i][0]=j
101    k=1
102    while parent[j]!=-1:
103        j=parent[j]
```

## I. CÓDIGO DESENVOLVIDO

---

```
104         dest[src][i][k]=j
105         k=k+1
106
107     @cuda.jit
108     def dijkstracuda(graph, dest):
109
110         start = cuda.grid(1)
111         stride = cuda.gridsize(1)
112
113     for src in range(start, graph.shape[0], stride):
114
115         row = graph.shape[0]
116         col = row
117
118         dist = cuda.local.array(150, dtype=numba.float64)
119         dist[:] = np.inf
120
121
122         parent = cuda.local.array(150, dtype=numba.int64)
123         parent[:] = -1
124
125         dist[src] = 0
126
127         queue = cuda.local.array(150, dtype=numba.int64)
128         for i in range(row):
129             queue[i] = i
130
131
132         while verifyQueue(queue):
133
134             u = minDistancecuda(dist,queue, row)
135
136             if(u == -1):
137                 queue[:]=-1
138             else:
139                 queue[u]=-1
140                 for i in range(col):
141                     if graph[u][i] and queue[i] != -1:
142                         if dist[u] + graph[u][i] < dist[i]:
143                             dist[i] = dist[u] + graph[u][i]
144                             parent[i] = u
145
146             for i in range(0, row):
147                 s = addPathcuda(parent,i, dest, src)
148
```

---

```
149 def preparePathString(path, pathString, i):
150     if i >= len(path):
151         return pathString
152     v = path[i]
153
154     if v == -1:
155         return pathString
156     pathString = preparePathString(path, pathString, i+1)
157     pathString += str(v) + ","
158
159     return pathString
160
160 def prepareDataframeFromResult(resultMat, df):
161     for x in range(len(resultMat)):
162         for y in range(len(resultMat[0])):
163             pathString=""
164             pathString = preparePathString(resultMat[x][y], pathString, 0)
165             df = df.append({'start': x, 'end': y, 'path': pathString[:-1]}, ignore_index=True)
166
167     return df
168
168 def trainModel(train, aquarium_id, path):
169     train.info()
170     y=train.pop('path')
171
172     print('Start Fit:')
173     p = LogisticRegression(max_iter=5000)
174     print('Model Fit:')
175     p.fit(train, y)
176     print('Test Model:')
177     p.predict([[0,1]])
178     print('Save Model:')
179
180     with open(path+'tmp.sav', 'wb') as f:
181         pickle.dump(p, f)
182
183     bucket = storage.bucket()
184     blob = bucket.blob(str(aquarium_id)+'.sav')
185     blob.upload_from_filename(path+"tmp.sav")
186
187     if os.path.exists(path+"tmp.sav"):
188         os.remove(path+"tmp.sav")
189
190
191 def main(maxWeight, typeAmount, path):
192
193     db = createDBConnection()
```

## I. CÓDIGO DESENVOLVIDO

---

```
194     docs = [snapshot for snapshot in db.collection(u'Aquarium').stream()]
195     for doc in docs:
196         print(f'{doc.id}-----')
197         print('Prepare Mat:')
198         aquarium_id = int(doc.id)
199         size = doc.get('size')
200         list = [None] * size
201         rows = [snapshot for snapshot in doc.reference.collection(u'Rows').stream()]
202         for row in rows:
203             list[int(row.id)] = row.get('array')
204         matrix = np.asarray(list, dtype=np.int16)
205         #print(matrix)
206         adjMatrix = prepareWeightedAdjMat(matrix, size, maxWeight, typeAmount)
207
208         print('Prepare Dataset:')
209
210         dfc = pd.DataFrame(columns = ['start', 'end', 'path'])
211         dfc['start']=dfc['start'].astype('int')
212         dfc['end']=dfc['end'].astype('int')
213
214         resultMat = np.empty((size,size,size), dtype=np.int32)
215         resultMat[:]=-1
216
217         drm=cuda.to_device(resultMat)
218
219         threads_per_block = 32
220         blocks_per_grid = 2
221
222         dijkstracuda[blocks_per_grid, threads_per_block](adjMatrix, drm)
223
224         drm.copy_to_host(resultMat)
225
226         dfc = prepareDataframeFromResult(resultMat, dfc)
227
228         print('dataset created')
229
230         #print(dfc)
231         print('Train Model:')
232         trainModel(dfc, aquarium_id, path)
233
234         print('-----')
```

---

Listing 18: createModelCuda.py

---

---

```
1 import pickle
2 import firebase_admin
3 from firebase_admin import credentials
4 from firebase_admin import firestore
5 from firebase_admin import storage
6 import os
7 import random
8 import time
9
10 def createDBConnection():
11     # Use a service account
12     try:
13         firebase_admin.get_app()
14     except ValueError:
15         cred = credentials.Certificate('firebasekey.json')
16         firebase_admin.initialize_app(cred, {
17             'storageBucket': 'finalprojectaquarium.appspot.com'
18         })
19     return firestore.client()
20
21
22 def main(aquarium_id, startPoint):
23
24     db = createDBConnection()
25     doc_ref = db.collection(u'Aquarium').document(str(aquarium_id))
26     doc = doc_ref.get()
27     size = doc.get('size')
28
29     bucket = storage.bucket()
30     blob = bucket.blob(str(aquarium_id)+'.sav')
31     millis = int(round(time.time() * 1000))
32     tmpfilename="tmp_"+str(aquarium_id)+"_"+str(millis)+".sav"
33     blob.download_to_filename(tmpfilename)
34
35     with open(tmpfilename, 'rb') as f:
36         loaded_model = pickle.load(f)
37
38     fullpath = [int]*5000
39
40     k=0
41     while k < len(fullpath):
42         if k==0:
43             start = startPoint
44         else:
```

## I. CÓDIGO DESENVOLVIDO

---

```
45         start=fullpath[k-1]
46         end=random.randint(0, size-1)
47         while start == end:
48             end=random.randint(0, size-1)
49         path = loaded_model.predict([[start, end]])[0]
50         splitpath = path.split(',')
51         for point in splitpath:
52             if k < len(fullpath) and k!=0 and fullpath[k-1] != int(point):
53                 fullpath[k]=int(point)
54                 k+=1
55             elif k == 0 and start!=int(point):
56                 fullpath[k]=int(point)
57                 k+=1
58
59     if os.path.exists(tmpfilename):
60         os.remove(tmpfilename)
61
62     return fullpath
```

---

Listing 19: getEntireFishPath.py

```
1 import pickle
2
3 def main(path, aquarium_id, start, end):
4     with open(path+str(aquarium_id)+'.sav', 'rb') as f:
5         loaded_model = pickle.load(f)
6     path = loaded_model.predict([[start, end]])
7     return path[0]
```

---

Listing 20: getShortestPathFromModel.py

```
1 import firebase_admin
2 from firebase_admin import credentials
3 from firebase_admin import firestore
4 from firebase_admin import storage
5
6 def createDBConnection():
7     # Use a service account
8     try:
9         firebase_admin.get_app()
10    except ValueError:
```

---

```
11     cred = credentials.Certificate('firebasekey.json')
12     firebase_admin.initialize_app(cred, {
13         'storageBucket': 'finalprojectaquarium.appspot.com'
14     })
15     return firestore.client()
16
17 def main():
18
19     db = createDBConnection()
20     rows = [snapshot for snapshot in db.collection(u'Aquarium').document(str(1)).collection(u'R'
21     for row in rows:
22         id = int(row.id)
23         array = row.get('array')
24         size = len(array)
25         for i in range(size):
26             array[i]=0
27         if id == 0:
28             array[1]=1
29             array[2]=1
30         elif id == 1:
31             array[0]=1
32             array[2]=1
33         elif id == 2:
34             array[0]=1
35             array[1]=1
36             array[3]=1
37         elif id == 3:
38             array[2]=1
39             array[4]=1
40             array[5]=1
41             array[6]=1
42             array[7]=1
43             array[10]=1
44         elif id == 4:
45             array[3]=1
46             array[5]=1
47             array[6]=1
48             array[7]=1
49             array[10]=1
50         elif id == 5:
51             array[3]=1
52             array[4]=1
53             array[6]=1
54             array[7]=1
55             array[10]=1
```

## I. CÓDIGO DESENVOLVIDO

---

```
56     elif id == 6:
57         array[3]=1
58         array[4]=1
59         array[5]=1
60         array[10]=1
61         array[7]=1
62         array[11]=1
63         array[9]=1
64         array[8]=1
65     elif id == 7:
66         array[3]=1
67         array[4]=1
68         array[5]=1
69         array[6]=1
70         array[10]=1
71         array[11]=1
72         array[9]=1
73         array[8]=1
74     elif id == 8:
75         array[6]=1
76         array[7]=1
77         array[10]=1
78         array[9]=1
79         array[11]=1
80         array[12]=1
81         array[13]=1
82     elif id == 9:
83         array[6]=1
84         array[7]=1
85         array[10]=1
86         array[8]=1
87         array[11]=1
88     elif id == 10:
89         array[3]=1
90         array[4]=1
91         array[5]=1
92         array[6]=1
93         array[7]=1
94         array[11]=1
95         array[9]=1
96         array[8]=1
97     elif id == 11:
98         array[6]=1
99         array[9]=1
100        array[8]=1
```

---

```
101         array[7]=1
102         array[10]=1
103     elif id == 12:
104         array[8]=1
105         array[13]=1
106     elif id == 13:
107         array[8]=1
108         array[12]=1
109         array[14]=1
110         array[15]=1
111     elif id == 14:
112         array[13]=1
113         array[15]=1
114     elif id == 15:
115         array[13]=1
116         array[14]=1
117         array[16]=1
118     elif id == 16:
119         array[15]=1
120         array[17]=1
121         array[18]=1
122     elif id == 17:
123         array[16]=1
124         array[18]=1
125     elif id == 18:
126         array[16]=1
127         array[17]=1
128         array[19]=1
129     elif id == 19:
130         array[18]=1
131     print(id,array)
132     print(type(row.reference))
133     row.reference.update({u'array': array})
```

---

Listing 21: modifyingaquarium1scripts.py

---

```
1 import firebase_admin
2 from firebase_admin import credentials
3 from firebase_admin import firestore
4
5 def createDBConnection():
6     # Use a service account
7     try:
```

## I. CÓDIGO DESENVOLVIDO

---

```
8     firebase_admin.get_app()
9 except ValueError:
10     cred = credentials.Certificate('firebasekey.json')
11     firebase_admin.initialize_app(cred)
12     return firestore.client()
13
14 def main(id, start, end):
15     db = createDBConnection()
16     start_ref = db.collection(u'Aquarium').document(str(id)).collection(u'Rows').document(str(start))
17     startRow = start_ref.get().get('array')
18     startRow[end]=0
19     start_ref.update({u'array': startRow})
```

---

Listing 22: modifyMatrix.py

```
1 from numba import cuda
2 import numba
3 import numpy as np
4 import random
5 import pickle
6 import createModel
7 import time
8 import math
9 import pandas as pd
10 from numba import types
11 DjiRowSize = -1
12
13 @cuda.jit
14 def add_kernel(x, path, aquarium_id):
15     start = cuda.grid(1)      # 1 = one dimensional thread grid, returns a single value
16     stride = cuda.gridsize(1) # ditto
17     # assuming x and y inputs are same length
18     for i in range(start, x.shape[0], stride):
19         x[i] = 0
20
21 @cuda.jit
22 def addWeightToadjMatrixCuda(adjMatrix, weightMat, typeMatrix):
23     xstart, ystart = cuda.grid(2)
24     stridex, stridey = cuda.gridsize(2)
25     for x in range(xstart, len(adjMatrix), stridex):
26         xType = typeMatrix[x]
27         for y in range(ystart, len(adjMatrix[x]), stridey):
28             if(x == y):
```

---

```

29         adjMatrix[x][y] = 0
30     else:
31         yType = typeMatrix[y]
32         weightOnSpot = weightMat[xType-1][yType-1]
33         adjMatrix[x][y] = adjMatrix[x][y] * weightOnSpot
34
35
36
37 def print1d(m):
38     for i in range(len(m)):
39         print(m[i])
40
41 @cuda.jit
42 def print1dcudasingle(m):
43     pos = cuda.grid(1)
44     if pos < len(m):
45         print("value ", m[pos], "thread" , pos, "\n" )
46
47 @cuda.jit
48 def print1dcudafor(m):
49     start = cuda.grid(1)      # 1 = one dimensional thread grid, returns a single value
50     stride = cuda.gridsize(1) # ditto
51     for i in range(start, m.shape[0], stride):
52         print("value ", m[i], "start" , start, "stride",stride,"\\n" )
53
54
55 def print2d(m):
56     for x in range(len(m)):
57         for y in range(len(m[0])):
58             print("value",m[x][y],"x",x,"y",y)
59
60 @cuda.jit
61 def print2dcudasingle(m):
62     x, y = cuda.grid(2)
63     if x < m.shape[0] and y < m.shape[1]:
64         print("value",m[x][y],"x",x,"y",y)
65
66 @cuda.jit
67 def print2dcudafor(m):
68     xstart, ystart = cuda.grid(2)
69     stridex, stridey = cuda.gridsize(2)
70     for x in range(xstart, len(m), stridex):
71         for y in range(ystart, len(m[0]), stridey):
72             print("value",m[x][y],"xstart",xstart,"ystart",ystart,"xstride",stridex,"ystride",stridey)
73

```

## I. CÓDIGO DESENVOLVIDO

---

```
74
75 def minDistance(dist,queue):
76     minimum = float("Inf")
77     min_index = -1
78
79     for i in range(len(dist)):
80         if dist[i] < minimum and i in queue:
81             minimum = dist[i]
82             min_index = i
83
84     return min_index
85
86
87 def printPath(parent, j, s):
88
89     if parent[j] == -1 :
90         s += str(j) + ","
91         return s
92
93     s = printPath(parent , parent[j], s)
94     s += str(j) + ","
95
96     return s
97
98
99 def printSolution(dist, parent, src, df):
100
101     for i in range(1, len(dist)):
102         s=""
103         s = printPath(parent,i,s)
104         #print("start: %d end: %d path %s id %d" % (src, i, s, aquarium_id))
105         df = df.append({'start': src , 'end': i, 'path': s[:-1]}, ignore_index=True)
106
107     return df
108
109
110 def dijkstra(graph, src, df):
111
112     row = len(graph)
113     col = len(graph[0])
114
115     dist = [float("Inf")] * row
116
117     parent = [-1] * row
118
119     dist[src] = 0
120
121     queue = []
122     for i in range(row):
123         queue.append(i)
124
125     while queue:
```

---

```

119         u = minDistance(dist,queue)
120
121         queue.remove(u)
122
123         for i in range(col):
124             if graph[u][i] and i in queue:
125                 if dist[u] + graph[u][i] < dist[i]:
126                     dist[i] = dist[u] + graph[u][i]
127                     parent[i] = u
128
129         df = printSolution(dist, parent, src, df)
130
131     return df
132
133 @cuda.jit(device=True)
134 def verifyQueue(queue):
135     for element in queue:
136         if element != -1:
137             return True
138     return False
139
140 @cuda.jit(device=True)
141 def minDistanc(cuda,dist,queue):
142     minimum = np.inf
143     min_index = -1
144
145     for i in range(len(dist)):
146         if dist[i] < minimum and queue[i] != -1:
147             minimum = dist[i]
148             min_index = i
149
150     return min_index
151
152 @cuda.jit(device=True)
153 def printPathcuda(parent, j, s):
154     if parent[j] == -1 :
155         s += str(j) + ","
156     return s
157     s = printPathcuda(parent , parent[j], s)
158     s += str(j) + ","
159
160     return s
161
162 @cuda.jit(device=True)
163 def testcuda(parent,i):
164
165     j=i
166     s="hello"

```

## I. CÓDIGO DESENVOLVIDO

---

```
164     #s=j
165     while parent[j] !=-1:
166         j=parent[j]
167         #s=s+", "+j
168
169
170     return s
171
172 @cuda.jit
173 def dijkstracuda(graph, dest):
174
175     start = cuda.grid(1)      # 1 = one dimensional thread grid, returns a single value
176     stride = cuda.gridsize(1) # ditto
177
178     for src in range(start, 1, stride):
179         row = len(graph)
180         col = len(graph[0])
181
182         dist = cuda.local.array(DjiRowSize, dtype=numba.float64)
183         dist[:] = np.inf
184
185
186         parent = cuda.local.array(DjiRowSize, dtype=numba.int64)
187         parent[:] = -1
188
189         dist[src] = 0
190
191         queue = cuda.local.array(DjiRowSize, dtype=numba.int64)
192         for i in range(row):
193             queue[i] = i
194
195         #u = testcudaifik(dist, queue)
196
197
198         while verifyQueue(queue):
199
200             u = minDistancecuda(dist,queue)
201
202             queue[u]=-1
203
204             for i in range(col):
205                 if graph[u][i] and queue[i] != -1:
206                     if dist[u] + graph[u][i] < dist[i]:
207                         dist[i] = dist[u] + graph[u][i]
208                         parent[i] = u
```

---

```
209
210     print("time for while test")
211     #dest[0][1]=t
212     for i in range(0, len(dist)):
213         s = testcudafik(parent,i)
214         #s = printPathcuda(parent,i,s)
215
216         #print(dest[src][i])
217
218
219 @cuda.jit
220 def modiftvar(m):
221     x, y= cuda.grid(2)
222     if x < m.shape[0] and y < m.shape[1]:
223         zsize=len(m[0][0])
224         for z in range(zsize):
225             m[x][y][z]=2
226
227 def main():
228
229     x=cuda.to_device(m)
230     threads_per_block = 128
231     blocks_per_grid = 30
232     add_kernel[blocks_per_grid, threads_per_block](x, path, aquarium_id)
233     x.copy_to_host(result)
234
235     size = 10
236     typeAmount = 10
237     matrix = np.ones((size,size), dtype=np.int16)
238     wm = createModel.createWeightMat(typeAmount, 1000)
239     tm = createModel.createTypeMat(size, typeAmount)
240
241     start = time.time()
242
243     #am = createModel.addWeightToadjMatrix(matrix, wm, tm)
244
245     end = time.time()
246
247     #print(am)
248     print("Took %f ms" % ((end - start) * 1000.0))
249
250     m = np.ones((size,size), dtype=np.int16)
251     dwm=cuda.to_device(wm)
252     dtm=cuda.to_device(tm)
253     dm=cuda.to_device(m)
```

## I. CÓDIGO DESENVOLVIDO

---

```
254     threadsperblock = (32, 32)
255     blockspergrid_x = math.ceil(len(dm) / threadsperblock[0])
256     blockspergrid_y = math.ceil(len(dm[0])/ threadsperblock[1])
257     blockspergrid = (blockspergrid_x, blockspergrid_y)
258
259     start = time.time()
260
261     addWeightToadjMatrixCuda[blockspergrid, threadsperblock](dm, dwm, dtm)
262
263     end = time.time()
264
265     dm.copy_to_host(m)
266     print(m)
267     print("Took %f ms" % ((end - start) * 1000.0))
268
269     print("-----")
270
271     df = pd.DataFrame(columns = ['start', 'end', 'path'])
272     df['start']=df['start'].astype('int')
273     df['end']=df['end'].astype('int')
274
275     start = time.time()
276     #for x in range(len(m)):
277     #    df = dijkstra(m, x, df)
278
279
280     end = time.time()
281     print(df)
282     print("Took %f ms" % ((end - start) * 1000.0))
283
284     print("-----")
285
286     dp = pd.DataFrame(columns = ['start', 'end', 'path'])
287     dp['start']=dp['start'].astype('int')
288     dp['end']=dp['end'].astype('int')
289
290     resultMat = np.empty((size, size), dtype='<U300')
291     resultMat[:, :] = "hello"
292     resultMat[0][0] = "world"
293     global DjiRowSize
294     DjiRowSize=len(resultMat)
295
296     dm=cuda.to_device(m)
297     drm=cuda.to_device(resultMat)
298
```

---

```
299     threads_per_block = 128
300     blocks_per_grid = 30
301
302
303     print( numba.typeof(resultMat) )
304     print( numba.typeof(resultMat[0]) )
305     print( numba.typeof(resultMat[0][0]) )
306     print( numba.typeof("hello") )
307     start = time.time()
308     dijkstracuda[blocks_per_grid, threads_per_block](dm, drm)
309     end = time.time()
310     drm.copy_to_host(resultMat)
311
312     print("ended cuda")
313     for x in range(len(resultMat)):
314         for y in range(len(resultMat[0])):
315             dp = dp.append({ 'start': x , 'end': y, 'path': resultMat[x][y] }, ignore_index=True)
316     print(dp)
317     print("Took %f ms" % ((end - start) * 1000.0))
318     #-----
319     #n = 100000
320     #x = np.arange(n).astype(np.int32)
321     #print(x[:10])
322     #f_device = cuda.to_device(f)
323     #x_device = cuda.to_device(x)
324     #y_device = cuda.to_device(y)
325     #out_device = cuda.device_array_like(x)
326
327     #threads_per_block = 128
328     #blocks_per_grid = 30
329
330     #add_kernel[blocks_per_grid, threads_per_block](x_device, path, aquarium_id);
331     #x_device.copy_to_host()
332
333     #print(x_device[0])
334     #-----
335     #m = np.arange(10)
336     #print1d(m)
337     #print("-----\n")
338
339     #d_m = cuda.to_device(m)
340
341     #threads_per_block = 128
342     #blocks_per_grid = 30
343     #print1dcudasingle[blocks_per_grid, threads_per_block](d_m)
```

## I. CÓDIGO DESENVOLVIDO

---

```
344
345     #h_m = d_m.copy_to_host()
346
347     #print("-----\n")
348
349     #d_m2 = cuda.to_device(m)
350
351     #threads_per_block = 5
352     #blocks_per_grid = 1
353     #print1dcudafor[blocks_per_grid, threads_per_block](d_m2)
354
355     #h_m2 = d_m2.copy_to_host()
356     #print("-----\n")
357
358     #m = np.ones((10,10), dtype=np.int16)
359
360     #print2d(m)
361     #print("-----\n")
362     #d_m = cuda.to_device(m)
363     #threadsperblock = (16, 16)
364     #blockspergrid_x = math.ceil(len(d_m) / threadsperblock[0])
365     #blockspergrid_y = math.ceil(len(d_m[0])/ threadsperblock[1])
366     #blockspergrid = (blockspergrid_x, blockspergrid_y)
367     #print2dcudasingle[blockspergrid, threadsperblock](m)
368     #h_m = d_m.copy_to_host()
369     print("-----\n")
370
371     mas = np.empty((3,3,3), dtype=np.int32)
372     mas[:]=-1
373     print(mas)
374     d_m2 = cuda.to_device(mas)
375     threadsperblock = (16, 16)
376     blockspergrid_x = math.ceil(len(d_m2) / threadsperblock[0])
377     blockspergrid_y = math.ceil(len(d_m2[0])/ threadsperblock[1])
378     blockspergrid = (blockspergrid_x, blockspergrid_y)
379     modiftvar[blockspergrid, threadsperblock](d_m2)
380     h_m2 = d_m2.copy_to_host()
381     print(h_m2)
```

---

Listing 23: pythonCudaTest.py

---

```
1 from numba import cuda
2 import numba
```

---

```
3 import numpy as np
4 import createModel
5 import pickle
6 import time
7 import math
8 import pandas as pd
9 from numba import types
10
11 from sklearn.svm import LinearSVC
12 from sklearn.pipeline import make_pipeline
13 from sklearn.preprocessing import StandardScaler
14
15
16 DjiRowSize = -1
17
18 @cuda.jit
19 def addWeightToadjMatrixCuda(adjMatrix, weightMat, typeMatrix):
20     xstart, ystart = cuda.grid(2)
21     stridex, stridey = cuda.gridsize(2)
22     for x in range(xstart, len(adjMatrix), stridex):
23         xType = typeMatrix[x]
24         for y in range(ystart, len(adjMatrix[x]), stridey):
25             if(x == y):
26                 adjMatrix[x][y] = 0
27             else:
28                 yType = typeMatrix[y]
29                 weightOnSpot = weightMat[xType-1][yType-1]
30                 adjMatrix[x][y] = adjMatrix[x][y] * weightOnSpot
31
32
33
34
35 @cuda.jit(device=True)
36 def verifyQueue(queue):
37     for element in queue:
38         if element != -1:
39             return True
40     return False
41
42 @cuda.jit(device=True)
43 def minDistanceCuda(dist, queue):
44     minimum = np.inf
45     min_index = -1
46
47     for i in range(len(dist)):
```

## I. CÓDIGO DESENVOLVIDO

---

```
48         if dist[i] < minimum and queue[i] != -1:
49             minimum = dist[i]
50             min_index = i
51     return min_index
52
53 @cuda.jit(device=True)
54 def printPathcuda(parent,i,dest, src):
55     j=i
56     dest[src][i][0]=j
57     k=1
58     while parent[j]!=-1:
59         j=parent[j]
60         dest[src][i][k]=j
61         k=k+1
62
63 @cuda.jit
64 def dijkstracuda(graph, dest):
65
66     start = cuda.grid(1)
67     stride = cuda.gridsize(1)
68
69     for src in range(start, DjiRowSize, stride):
70         row = DjiRowSize
71         col = row
72
73         dist = cuda.local.array(row, dtype=numba.float64)
74         dist[:] = np.inf
75
76
77         parent = cuda.local.array(row, dtype=numba.int64)
78         parent[:] = -1
79
80         dist[src] = 0
81
82         queue = cuda.local.array(row, dtype=numba.int64)
83         for i in range(row):
84             queue[i] = i
85
86
87         while verifyQueue(queue):
88
89             u = minDistancecuda(dist,queue)
90
91             if(u == -1):
92                 queue[:]=-1
```

---

```
93         else:
94             queue[u]=-1
95             for i in range(col):
96                 if graph[u][i] and queue[i] != -1:
97                     if dist[u] + graph[u][i] < dist[i]:
98                         dist[i] = dist[u] + graph[u][i]
99                         parent[i] = u
100
101            for i in range(0, len(dist)):
102                s = printPathcuda(parent,i, dest, src)
103
104 def preparePathString(path, pathString, i):
105     if i >= len(path):
106         return pathString
107     v = path[i]
108     if v == -1:
109         return pathString
110     pathString = preparePathString(path, pathString, i+1)
111     pathString += str(v) + ","
112     return pathString
113
114 def prepareDataframeFromResult(resultMat, df):
115     for x in range(len(resultMat)):
116         for y in range(len(resultMat[0])):
117             pathString=""
118             pathString = preparePathString(resultMat[x][y], pathString, 0)
119             df = df.append({'start': x , 'end': y, 'path': pathString[:-1]}, ignore_index=True)
120     return df
121
122
123 def main(wm,tm, size, typeAmount):
124     start = time.time()
125
126     m = np.ones((size,size), dtype=np.int32)
127     dwm=cuda.to_device(wm)
128     dtm=cuda.to_device(tm)
129     dm=cuda.to_device(m)
130     threadsperblock = (32, 32)
131     blockspergrid_x = math.ceil(len(dm) / threadsperblock[0])
132     blockspergrid_y = math.ceil(len(dm[0])/ threadsperblock[1])
133     blockspergrid = (blockspergrid_x, blockspergrid_y)
134
135     addWeightToadjMatrixCuda[blockspergrid, threadsperblock](dm, dwm, dtm)
136
137     df = pd.DataFrame(columns = ['start', 'end', 'path'])
```

## I. CÓDIGO DESENVOLVIDO

---

```
138     df['start']=df['start'].astype('int')
139     df['end']=df['end'].astype('int')
140
141     resultMat = np.empty((size,size,size), dtype=np.int32)
142     resultMat[:]=-1
143
144     global DjiRowSize
145     DjiRowSize=size
146
147     drm=cuda.to_device(resultMat)
148
149     threads_per_block = 32
150     blocks_per_grid = 2
151
152     dijkstracuda[blocks_per_grid, threads_per_block](dm, drm)
153
154     drm.copy_to_host(resultMat)
155
156     df = prepareDataframeFromResult(resultMat, df)
157
158     end = time.time()
159     print("Took %f ms" % ((end - start) * 1000.0))
160     print(df)
```

---

Listing 24: pythoncudav2.py

```
1 import createModel
2 import addnewaquariumtodatabase
3 import getShortestPathFromModel
4 import modifyMatrix
5 import verifyIfAquariumExistsInDB
6 import pythonCudaTest
7 import pythoncudav2
8 import numpy as np
9 import pandas as pd
10 import time
11 import createModelCuda
12
13 #addnewaquariumtodatabase.main(0, 36) #aquarium 0
14
15 #addnewaquariumtodatabase.main(1, 20) #aquarium 1
16
17 #addnewaquariumtodatabase.main(2, 7) #aquarium 2
```

---

```
18
19 #addnewaquariumtodatabase.main(3, 16) #aquarium 3
20
21 #addnewaquariumtodatabase.main(4, 12) #aquarium 4
22
23 createModelCuda.main(1000, 10, 'C:/Users/fresc/OneDrive/Ambiente de Trabalho/Nova pasta (2)/')
24
25
26
27
28
29
30
31
32
33 #addnewaquariumtodatabase.main(1, 10)
34 #addnewaquariumtodatabase.main(2, 20)
35 #createModelCuda.main(1000, 10, 'C:/Users/fresc/OneDrive/Ambiente de Trabalho/Nova pasta (2)/')
36 #modifyMatrix.main(0,0,1)
37 #getShortestPathFromModel.main('C:/Users/fresc/OneDrive/Ambiente de Trabalho/Nova pasta (2)/',
38 #getShortestPathFromModel.main('C:/Users/fresc/OneDrive/Ambiente de Trabalho/Nova pasta (2)/',
39 #getShortestPathFromModel.main('C:/Users/fresc/OneDrive/Ambiente de Trabalho/Nova pasta (2)/',
40 #print	verifyIfAquariumExistsInDB.main(0))
41 #print	verifyIfAquariumExistsInDB.main(5))
42 #
43 ##pythonCudaTest.main('C:/Users/fresc/OneDrive/Ambiente de Trabalho/Nova pasta (2)/', 0, 5)
44 #pythonCudaTest.main()
45
46 #size = 30
47 #typeAmount = 10
48 #wm = createModel.createWeightMat(typeAmount, 1000)
49 #tm = createModel.createTypeMat(size, typeAmount)
50
51 #print("GPU-----")
52 #pythoncudaV2.main(wm, tm, size, typeAmount)
53
54 #print("CPU-----")
55
56 #def minDistance(dist, queue):
57 #    minimum = float("Inf")
58 #    min_index = -1
59
60 #    for i in range(len(dist)):
61 #        if dist[i] < minimum and i in queue:
62 #            minimum = dist[i]
```

## I. CÓDIGO DESENVOLVIDO

---

```
63 #             min_index = i
64 #     return min_index
65
66 #def printPath(parent, j, s):
67
68 #     if parent[j] == -1 :
69 #         s += str(j) + ","
70 #     return s
71 #     s = printPath(parent , parent[j], s)
72 #     s += str(j) + ","
73 #     return s
74
75 #def printSolution(dist, parent, src, df):
76
77 #     for i in range(0, len(dist)):
78 #         s=""
79 #         s = printPath(parent,i,s)
80 #         #print("start: %d end: %d path %s id %d" % (src, i, s, aquarium_id))
81 #         df = df.append({'start': src , 'end': i, 'path': s[:-1]}, ignore_index=True)
82 #     return df
83
84 #def dijkstra(graph, src, df):
85
86 #     row = len(graph)
87 #     col = len(graph[0])
88
89 #     dist = [float("Inf")] * row
90
91 #     parent = [-1] * row
92
93 #     dist[src] = 0
94
95 #     queue = []
96 #     for i in range(row):
97 #         queue.append(i)
98
99 #     while queue:
100 #         u = minDistance(dist,queue)
101
102 #         if(u == -1):
103 #             queue = []
104 #         else:
105 #             queue.remove(u)
106 #             for i in range(col):
107 #                 if graph[u][i] and i in queue:
```

---

```
108     #             if dist[u] + graph[u][i] < dist[i]:  
109     #                     dist[i] = dist[u] + graph[u][i]  
110     #                     parent[i] = u  
111  
112     #         df = printSolution(dist, parent, src, df)  
113     #         return df  
114  
115  
116     #start = time.time()  
117  
118     #m = np.ones((size,size), dtype=np.int32)  
119  
120     #am = createModel.addWeightToadjMatrix(m, wm, tm)  
121  
122     #df = pd.DataFrame(columns = ['start', 'end', 'path'])  
123     #df['start']=df['start'].astype('int')  
124     #df['end']=df['end'].astype('int')  
125  
126     #for x in range(len(am)):  
127     #     df = dijkstra(am, x, df)  
128  
129     #end = time.time()  
130     #print("Took %f ms" % ((end - start) * 1000.0))  
131  
132     #print(df)  
133  
134     #c=[0,1,2]  
135     #if 0 and 1 in c:  
136     #     print("true")  
137     #else:  
138     #     print("False")  
139  
140     #createModelCuda.main(1000, 10)  
141  
142     #import firebase_admin  
143     #from firebase_admin import credentials  
144     #from firebase_admin import storage  
145     #from firebase_admin import firestore  
146     #import os  
147  
148     #def createDBConnection():  
149     #     # Use a service account  
150     #     try:  
151     #         firebase_admin.get_app()  
152     #     except ValueError:
```

## I. CÓDIGO DESENVOLVIDO

---

```
153 #     cred = credentials.Certificate('firebasekey.json')
154 #     firebase_admin.initialize_app(cred, {
155 #         'storageBucket': 'finalprojectaquarium.appspot.com'
156 #     })
157 #     return firestore.client()
158
159 #db = createDBConnection()
160 #docs = [snapshot for snapshot in db.collection(u'Aquarium').stream()]
161 #for doc in docs:
162 #    print(doc.id)
163
164 #f = open("demofile2.txt", "a")
165 #f.write("Now the file has more content!")
166 #f.close()
167
168 #bucket = storage.bucket()
169 #blob = bucket.blob("demofile.txt")
170 #blob.upload_from_filename("demofile2.txt")
171
172 #if os.path.exists("demofile2.txt"):
173 #    os.remove("demofile2.txt")
174 #else:
175 #    print("The file does not exist")
176
177 #import getEntireFishPath
178
179 #f = getEntireFishPath.main(0,12)
180
181 #print(f)
182
183 #import modifyingaquarium1scripts
184
185 #modifyingaquarium1scripts.main()
```

---

Listing 25: sklearnfishpathfinder.py

```
1 import firebase_admin
2 from firebase_admin import credentials
3 from firebase_admin import firestore
4
5
6 def createDBConnection():
7     # Use a service account
```

---

```
8     try:
9         firebase_admin.get_app()
10    except ValueError:
11        cred = credentials.Certificate('firebasekey.json')
12        firebase_admin.initialize_app(cred)
13    return firestore.client()
14
15 def main(id):
16     db = createDBConnection()
17     doc_ref = db.collection(u'Aquarium').document(str(id))
18     doc = doc_ref.get()
19     if doc.exists:
20         return True
21     else:
22         return False
```

---

Listing 26: verifyIfAquariumExistsInDB.py