Instituto Superior de Engenharia de Lisboa
Mestrado em Engenharia Informática e Computadores

Complementos de Sistemas de Informação

Trabalho Prático

PostGIS, Espaços virtuais e Realidade Aumentada

2º Semestre 2023/2024

Dezembro de 2023

Professor Orientador:

Professor Doutor Paulo Trigo

Alunos:

Rafael Carvalho – 47663

João Rocha - 47196

Índice

Índice de Figuras

Índice de Tabelas

Não foi encontrada nenhuma entrada do índice de ilustrações.

Lista de Acrónimos

Modelo EA-EPE: Modelo Entidade Associação Extendido com Pictogramas

SGBD: Sistema de Gestão de Bases de Dados

1 Introdução

As bases de dados relacionais são coleções de informação ou dados bem estruturados, tipicamente em linhas e colunas sob forma de entidades ou tabelas. A estes dados podem ser aplicadas relações de integridade referencial para garantir a consistência entre as tabelas, utilizando conceitos como chaves estrangeiras, garantido que determinados valores de um tuplo fazem referência a valores de outra tabela.

Com a evolução dos sistemas de informação, surgiu a motivação de incluir componente espacial em bases de dados, para armazenar dados referentes a localizações geográficas ou objetos num determinado espaço. Estes tipos de bases de dados são úteis na representação de mapas, regiões ou fronteiras de determinados locais. Desta forma, passou a ser possível fazer consultas espaciais, como por exemplo a geometria de um determinado terreno ou região, localização de um determinado objeto.

A extensão espacial PostGIS permite estender as funcionalidades do sistema de gestão de base de dados PostgreSQL, adicionando a componente espacial mencionada anteriormente.

A unidade curricular de Complementos de Sistemas de Informação visa explorar bases de dados com atributos espaciais. Desta forma, foi proposto um trabalho final onde se pretendia aprofundar os aspetos de modulação e utilização de extensões espaciais, utilizando PostgreSQL e PostGIS.

O trabalho, inspirado no conceito de realidade aumentada, pretende que se desenvolva uma solução para a representação de diferentes terrenos, rios, e objetos móveis. Os objetos móveis devem movimentar-se pelos terrenos, dentro da geometria de um "mundo"; para tal, possuem atributos como a posição, velocidade, aceleração e orientação. Para além disso, pretende-se que os objetos sejam capazes de perseguir um terceiro, tendo a sua velocidade diretamente afetada pelo tipo de terreno que atravessam.

Este documento encontra-se segmentado em cinco diferentes capítulos, seguindo-se a explicação do Modelo Entidade-Associação com Pictogramas (EA-EPE), e o Modelo Relacional (MR). No terceiro capítulo são mostradas as soluções propostas para resolver o problema proposto e a lógica por detrás das mesmas. Num quarto capítulo descrevem-se possíveis melhorias do projeto; terminando o documento com as conclusões, no quinto capítulo.

2 Modelo EA com Pictogramas e Modelo Relacional

Ao introduzir a extensão espacial à base de dados, surge a motivação de estender o modelo EA também, com conceitos espaciais. Isto deve-se devido aos dados espaciais pertencerem a conjuntos contínuos com relações implícitas. Os pictogramas adicionados aos símbolos gráficos do modelo EA permitem etiquetar entidades espaciais e o respetivo tipo de dados, bem como realizar a inferência sobre relações e restrições espaciais, aumentando a legibilidade.

2.1 Modelo EA-EPE

Tomando como referência as restrições impostas no enunciado do trabalho prático [1], o grupo propôs o modelo EA-EPE da Figura 1.

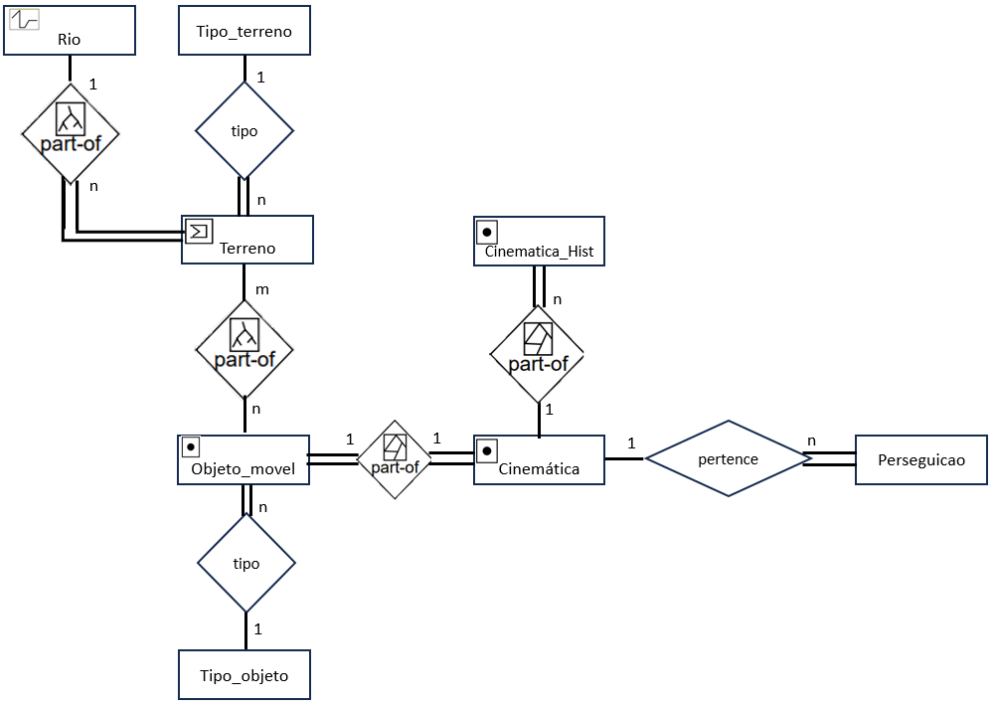

Figura 1 - Modelo EA-EPE proposto

Por questões de legibilidade e melhor leitura, os atributos das entidades foram omitidos, no entanto, podem e devem ser compreendidos pelo leitor no subcapítulo que se segue, referente ao modelo relacional.

2.2 Modelo Relacional

A transformação do Modelo EA-EPE para o modelo relacional permite compreender as restrições de integridade referencial entre as várias entidades, permitindo avançar para a elaboração do modelo físico no Sistema de Gestão de Bases de Dados (SGBD), em PostgreSQL.

Tipo_terreno:

Tipo_terreno (PK[nome])

Terreno:

Terreno (PK[id_terreno], FK[nome] ref tipo_terreno(nome), hierarquia, geo_terreno)

Rio:

Rio (PK[id_rio], nome, geo_linha)

Rota:

Rota (PK[id], geo_ponto)

GPS_Ponto:

GPS_Ponto (PK[FK[id_terreno ref terreno(id_terreno)], id_ordem], geo_ponto)

Tipo_objeto:

Tipo_objeto (PK[nome], velocidade_max, aceleracao_max)

Objeto_terreno:

objeto_terreno (FK[nome_objeto] ref tipo_objeto(nome), FK[nome_terreno] ref tipo_terreno(nome), efeito);

Cinematica:

cinematica (PK[id], nome, orientacao, velocidade, aceleracao, FK[nome] ref tipo_objeto(nome), g_posicao);

Cinematica_hist:

cinematica_hist (PK[id_hist], id, orientacao, velocidade, aceleracao, FK[id] ref cinematica(id), g_posicao);

Objeto_movel:

objeto_movel (PK[id], FK[id_cinematica] ref cinematica(id), FK[nome] ref tipo_objeto(nome), geo);

Figura 2 - Modelo Relacional resultante do Modelo EA-EPE

3 Soluções propostas

Elaborado o modelo físico da base de dados proposta, o grupo seguiu para a determinação de soluções para o problema proposto, descritas nos sub-capítulos que se seguem.

3.1 Hierarquia dos terrenos

Transcrevendo um dos requisitos do enunciado do trabalho prático: "*Um terreno, de determinado tipo, pode intersetar terrenos de outros tipos. Por exemplo, um terreno do tipo floresta pode conter um terreno do tipo pântano*"; a solução encontrada pelo grupo foi atribuir níveis de hierarquia distintos a cada tipo de terreno. Uma maior hierarquia indica que o terreno se encontra sobreposto ao anterior é o que prevalece, isto é, observando a Figura 3 e Figura 4, conclui-se que a "floresta" se sobrepõe a uma "montanha".

Esta solução de hierarquia de terrenos permite ao grupo determinar o quociente que deve ser aplicado à velocidade dos objetos que os intersetam, definido na tabela "*objeto_terreno*", no atributo *"efeito"*.

```
INSERT INTO terreno( id_terreno, nome, hierarquia)
VALUES
    (0, 'Mundo', 0),

    (1, 'Montanha', 1),
    (2, 'Floresta', 2),
```

Figura 3 - Hierarquia do Mundo, Montanha e Floresta

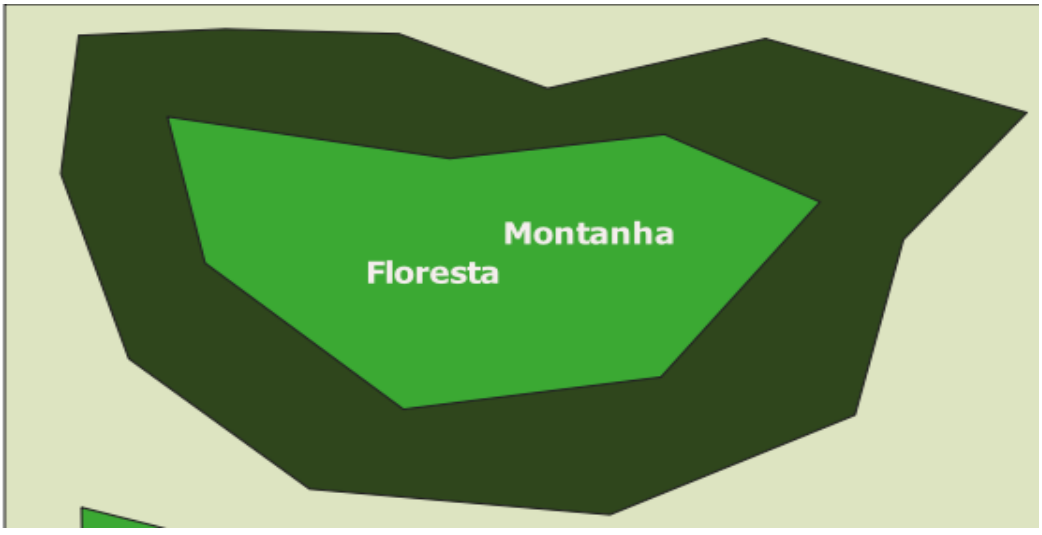

Figura 4 - Exemplo de hierarquia de terrenos

3.2 Efeito dos terrenos na velocidade dos objetos

Tal como mencionado anteriormente, para cada tipo de terreno a velocidade do objeto deverá ser condicionada ao efeito definido na tabela "*objeto_terreno"*, representada na Figura 5.

```sql
INSERT INTO objeto_terreno (nome_objeto, nome_terreno, efeito)
VALUES
    ('Carro', 'Mundo', 1),
    ('Carro', 'Montanha', 0.4),
    ('Carro', 'Deserto', 0.3),
    ('Carro', 'Floresta', 0.6),
    ('Carro', 'Vale', 0.5),
    ('Carro', 'Planicie', 0.5),
    ('Mota', 'Mundo', 1),
```

Figura 5 - Tabela objeto_terreno

Por cada iteração de simulação de movimento do objeto, a velocidade é atualizada tendo em conta a sua velocidade máxima. A função *comparar_velocidade(t_velocidade, real)*, retorna a nova velocidade calculada em *novo_velocidade(t_velocidade, t_aceleracao, integer)* multiplicada pelo quociente entre a norma do vetor da nova velocidade, e a aceleração máxima, caso a nova velocidade ultrapasse a máxima, definida na tabela *tipo_objeto*.
Após validação da nova velocidade, é aplicado o efeito do terreno na cinemática, com a função *get_efeito_terreno(integer)*.

```sql
-- ======================================================================
-- APPLIES THE EFFECT OF THE TERRAIN IN THE CINEMATIC AND CHECKS MAX-VELOCITY
-- ======================================================================
CREATE OR REPLACE FUNCTION calcular_velocidade_atualizada(id_cinematica integer)
RETURNS t_velocidade AS $$
DECLARE
    alvo_max_velocidade real;
    alvo_velocidade t_velocidade;
BEGIN
    SELECT velocidade_max FROM tipo_objeto t INNER JOIN cinematica c ON t.nome = c.nome WHERE c.id = id_cinematica INTO alvo_max_velocidade;
    alvo_velocidade := comparar_velocidade((SELECT novo_velocidade(velocidade, aceleracao, 1) FROM cinematica WHERE id = id_cinematica), alvo_max_ve
    alvo_velocidade := alvo_velocidade * get_efeito_terreno(id_cinematica);
    RETURN alvo_velocidade;
END;
$$ LANGUAGE plpgsql;
```

Figura 6 - Atualização da velocidade dos objetos

3.3 Lógica Perseguição

Para suportar a perseguição entre dois objetos, utiliza-se a tabela *perseguição* que mantém uma relação de um para um entre as cinemáticas do alvo e dos perseguidores, sendo que um objeto alvo pode ser perseguido por um ou mais objetos.

```
INSERT INTO cinematica( id, nome, orientacao, velocidade, aceleracao, g_posicao )
INSERT INTO perseguicao( id_perseguidor, id_alvo )
```

Os objetos perseguidores procuram aproximarem-se do alvo a cada iteração da simulação, no entanto, caso a distância euclidiana entre os dois pontos (alvo e perseguidor) seja inferior a 10 unidades, os perseguidores "hibernam" a perseguição, permanecendo na mesma posição.

O cálculo da nova posição do objeto é determinado através do seu vetor de velocidade e do tempo que este se desloca. É importante denotar que o cálculo da nova velocidade é diretamente afetado pela aceleração imposta no objeto, seguindo os conceitos empíricos do movimento de um corpo. As funções que calculam a nova posição, velocidade e aceleração são mostradas abaixo, na Figura 7, Figura 8 e Figura 9.

```
CREATE OR REPLACE FUNCTION novo_posicao( g_posicao geometry, velocidade t_velocidade, tempo real )
RETURNS geometry
AS $$
SELECT
ST_Translate( $1,
              (($2).linear * $3 ).x,
              (($2).linear * $3 ).y )
$$ LANGUAGE 'sql';
```

Figura 9 - Função novo_posicao de um objeto

```
CREATE OR REPLACE FUNCTION novo_velocidade( velocidade t_velocidade, aceleracao t_aceleracao, tempo real )
RETURNS t_velocidade
AS $$
SELECT
    ($1).linear + ($2).linear * ($3),
    ($1).angular + ($2).angular * ($3)
$$ LANGUAGE 'sql';
```

Figura 8 - Função nova_velocidade de um objeto

```
CREATE OR REPLACE FUNCTION novo_aceleracao_linear(
    g_posicao_perseguidor geometry,
    g_posicao_alvo geometry,
    velocidade_a_perseguir real
)
RETURNS t_vector
AS $$
DECLARE
    vector t_vector;
    aceleracao t_vector;
BEGIN
    vector := cast((ST_X($2) - ST_X($1), ST_Y($2) - ST_Y($1)) as t_vector) ;
    aceleracao := normalizar_PLPGSQL(vector) * velocidade_a_perseguir;

    RETURN aceleracao;
END
$$ LANGUAGE plpgsql;
```

Figura 7 - Função nova_aceleracao de um objeto

3.4 Simulação com Rota definida

Definidas as funções necessárias para a simulação do movimento pelos terrenos, o grupo optou por duas estratégias diferentes para demonstrar as funcionalidades do projeto: movimento com e sem rota definida.

A tabela *"rota"* mostrada em capítulos anteriores é composta por um conjunto de pontos ordenados que definem um caminho padrão que um objeto pode seguir.

Inserindo três objetos distintos na *cinemática* e em *perseguicao*: "Mota" como alvo, "Carro" e "Camião" como perseguidores, a solução para demonstrar a perseguição envolve transladar a "Mota", iterativamente, pelos pontos da rota, enquanto os outros dois objetos a seguem.

A rota definida encontra-se representada na Figura 10, e define o caminho do objeto "Mota", ao contrário dos restantes que a perseguem, atualizando a aceleração com base na distância euclidiana ao ponto alvo.

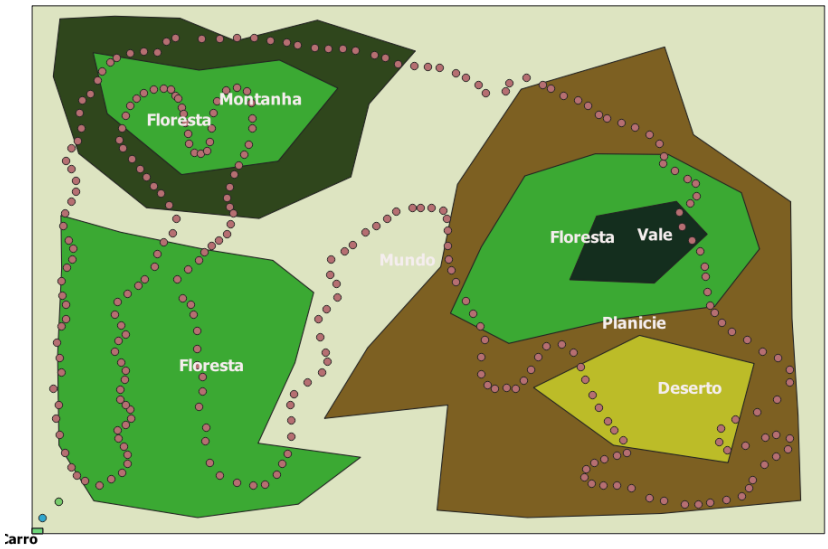

Figura 10 - Rota definida para travessia dos terrenos

O registo dos caminhos percorridos pelos objetos é representado numa *LINESTRING* que segue os objetos, como representado na Figura 11 abaixo:

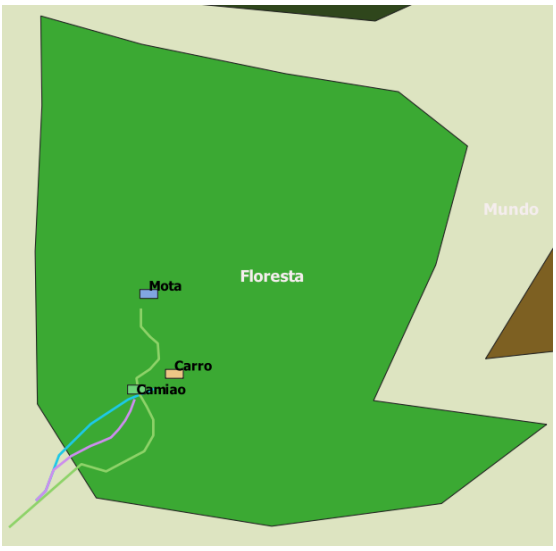

Figura 11 - Trajetória dos objetos seguidores

3.5 Simulação Livre com Orientação

Uma vez que a simulação anterior não permite ao objeto alvo seguir livremente pelo mapa, a segunda abordagem ao problema foi alterar a orientação do objeto alvo. Atendendo às restrições de data de entrega do trabalho final, esta segunda abordagem ficou por melhorar, uma vez que o objeto alvo segue sempre na mesma direção até que seja atualizada a sua orientação.

Nesta simulação, contrariamente à simulação com rota, o objeto alvo é transladado para a nova posição determinada pela função *novo_posicao_com_orientacao(geometry, t_velocidade, real, real)*.

```
CREATE OR REPLACE FUNCTION novo_posicao( g_posicao geometry, velocidade t_velocidade, tempo real )
RETURNS geometry
AS $$
SELECT
ST_Translate( $1,
              (($2).linear * $3 ).x,
              (($2).linear * $3 ).y )
$$ LANGUAGE 'sql';

CREATE OR REPLACE FUNCTION novo_posicao_com_orientacao(
    g_posicao geometry,
    velocidade t_velocidade,
    orientacao real,
    tempo real
)
RETURNS geometry AS $$
DECLARE
    novo_posicao_x real;
    novo_posicao_y real;
BEGIN
    -- Calcula as componentes x e y da velocidade linear com base na orientação
    novo_posicao_x := (velocidade.linear * TRUNC(cos(orientacao)::numeric, 2) * tempo).x;
    novo_posicao_y := (velocidade.linear * TRUNC(sin(orientacao)::numeric, 2) * tempo).y;

    -- Realiza a translação com as componentes calculadas da velocidade linear
    RETURN ST_Translate(
        $1,
        novo_posicao_x,
        novo_posicao_y
    );
END;
$$ LANGUAGE plpgsql;
```

Figura 13 - Função novo_posicao_com_orientacao

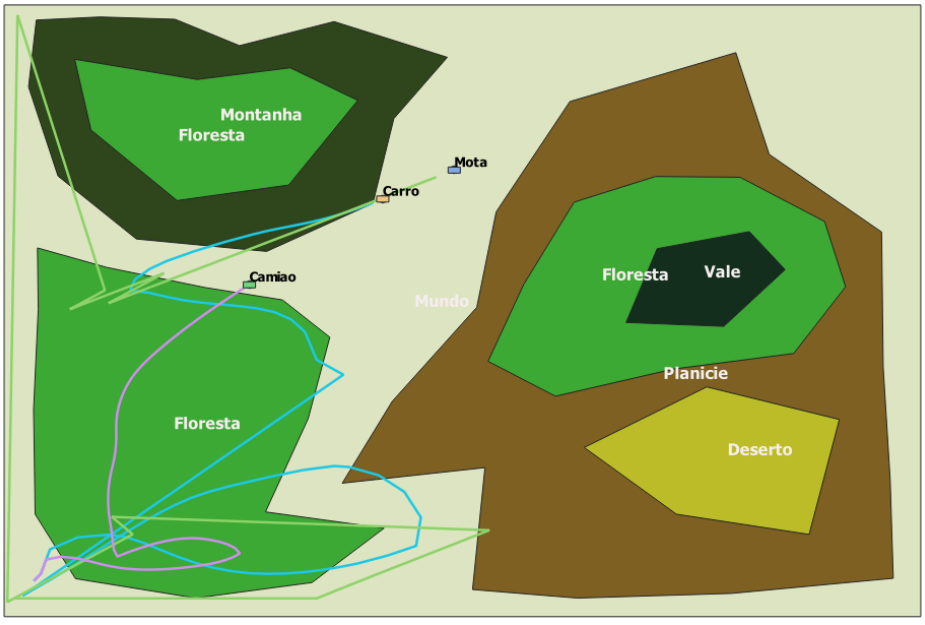

Figura 12 - Simulação com orientação

Observando a Figura 12, podemos comprovar a perseguição à "Mota" por parte do "Carro" e "Camião". A alteração do sentido da deslocação foi feito chamando manualmente a função *atualizar_orientacao(integer, integer)*. Uma melhoria a ser feita seria chamar esta função de forma dinâmica, por forma a tornar o movimento do objeto alvo mais aleatório.

3.6 *Batch files* e *Script* Python

Para facilitar os testes à solução proposta, o grupo desenvolveu um pequeno *script* Python que cria uma janela gráfica com um botão que permite simular iterações sobre a rota definida. É importante mencionar que este script deve ser alterado no módulo "simulation.py" para incluir as credenciais do SGBD PostgreSQL, caso contrário não funciona.

Em simultâneo, foram criadas *Batch files* numeradas com o prefixo '_go', que devem ser corridas de forma crescente, para criar e popular a base de dados.

4 Melhorias e trabalho futuro

4.1 Trabalho Futuro

De um ponto de vista crítico, o trabalho podia ainda sofrer alterações consideráveis para poder integrar objetivos mais ambiciosos, como por exemplo considerar um cenário de perseguição e fuga, em que o objeto alvo pudesse ter "noção de que está a ser perseguido", sendo aplicada uma aceleração simétrica à do perseguidor. Com isto, podia aplicar-se a ideia de diversos objetos perseguirem simultaneamente um outro objeto alvo, que foge, em cada instante, do perseguidor que lhe estiver mais próximo.

Podia também ser adicionada a funcionalidade de utilizar o *script* desenvolvido em Python para poder alterar de forma deliberada a aceleração de um determinado objeto, e indiretamente registar visualmente a alteração da sua velocidade aquando da simulação das trajetórias (movimento); bem como criar objetos dinamicamente e associar um alvo.

Pretende-se até à data de discussão incorporar algumas destas melhorias no trabalho desenvolvido.

5 Conclusão

O trabalho final da unidade curricular de Complementos de Sistemas de Informação tinha por objetivo aprofundar aspetos de modulação e utilização de extensões espaciais através da extensão PostGIS para o SGBD PostgreSQL.

Desenvolveu-se uma solução para construir uma base de dados espacial, com diversos terrenos, rios e objetos móveis, representada no QGIS, uma ferramenta de representação de informação espacial.

Os objetos podiam ser categorizados como alvo, andando pelo mundo segundo uma rota, ou numa segunda abordagem, livremente; ou como perseguidor, que segue um determinado alvo. Os vários movimentos são registados na base de dados por forma a construir uma representação gráfica resultante das várias iterações.

Esta solução permitiu ao grupo reforçar o conhecimento adquirido ao longo do semestre, reforçando também elementos base de SQL, como a definição de novos tipos de operadores, funções e vistas.

À data de entrega do documento, o grupo considera ter cumprido os objetivos base propostos no enunciado, no entanto, ambiciona implementar as melhorias mencionadas anteriormente.

6 Referências

[1] Enunciado do trabalho prático, disponível na plataforma moodle do ano letivo 23/24, elaborado pelo Professor Doutor Paulo Trigo