LABORATÓRIO DE PROJETO EM ENGENHARIA INFORMÁTICA

Augmented Reality Application for Monitoring Offshore Wind Farms with Meta Quest 3

2024/2025

João David Fraga da Silva al78400

Guilherme Videira Gonçalo al79954

1-Licenciatura em Engenharia Informática

utad UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO

Conteúdo

Link:
https://github.com/Guilhermegoncalo/WindFarm_Offshore_for_Meta_Quest_3

1. Resumo

Este projeto visa o desenvolvimento de uma "*digital twin*" de um parque eólico offshore, com enfoque na criação de um ambiente de Realidade Aumentada (AR). A proposta foi escolhida pelo interesse crescente pelas tecnologias imersivas aplicadas hoje dia, sendo que o foco neste projeto, está direcionado para o sector energético.

O dispositivo *Oculus Meta Quest 3* permitiu a visualização tridimensional do parque eólico da Aguçadoura, aliada ao acesso a dados operacionais em tempo real. O sistema integra modelos 3D, componentes de monitorização meteorológica, simulações de velocidade do vento e do funcionamento dos aerogeradores, recorrendo à *API* "*Open-Meteo*" para recolha de dados climáticos.

A arquitetura do sistema foi delineada com vista à sua escalabilidade, viabilizando assim futuras extensões da solução.

Este projeto representa um avanço inovador na integração de tecnologias AR, sendo aplicável não só ao domínio da energia e sustentabilidade, como também a outros setores que beneficiem da visualização e análise em tempo real de sistemas físicos complexos.

2. Introdução

Uma "*digital twin*" é uma representação digital de um objeto, sistema ou infraestrutura real que permite a monitorização, simulação e análise contínua do seu funcionamento em tempo real. Esta tecnologia tem ganho cada vez mais relevância em sectores como a indústria, construção e a energia, pela sua capacidade de melhorar a eficiência, antecipar falhas e otimizar operações com base em dados reais para verificação da viabilidade do processo.[9]

Assim, no contexto deste projeto, a ambição de desenvolver esta solução digital representou uma oportunidade única de aplicar este conceito numa área crítica da transição energética. A energia eólica offshore, ou seja, gerada por turbinas localizadas no mar, oferece vantagens significativas em relação à energia eólica terrestre, nomeadamente maiores velocidades de vento e maior estabilidade, o que se traduz numa maior rentabilidade e produção energética mais consistente. Além disso, o interesse estratégico por parte de países como Portugal, com extensas zonas costeiras, torna o investimento em offshore particularmente promissor no panorama energético nacional e europeu.[10]

A proposta inicial contemplava a utilização do dispositivo Apple Vision Pro, para visualizar e interagir com o modelo digital do parque eólico. No entanto, por motivos de acessibilidade a estes óculos e juntamente em concordância com os orientadores, optou-se pelo dispositivo *Oculus Meta Quest 3*, mantendo-se a aposta na imersividade e na exploração de ambientes de Realidade Aumentada. Para o desenvolvimento da aplicação recorreu-se ao motor de jogo *Unity*, dada a sua robustez no suporte a experiências imersivas e a sua compatibilidade com múltiplos dispositivos, o que inclui *Quest 3*. A nossa proposta foi realizada com base na localização da Aguçadoura, localizado na Póvoa de Varzim.

Desde o arranque do projeto, foi organizada uma reunião inicial para definição de objetivos concretos, sendo estes construir uma aplicação funcional que permita não só observar a infraestrutura em ambiente virtual, mas também interpretar visualmente os dados associados ao funcionamento dos aerogeradores.

Ao longo do semestre, o progresso foi acompanhado de forma através de reuniões periódicas com orientadores e colegas, permitindo validar decisões técnicas, integrar sugestões e garantir que a solução se mantinha coerente com os objetivos definidos. Foram abordadas questões como a modelação 3D do parque, a receção e visualização de dados, e a integração com os dispositivos de visualização imersiva.

2. Levantamento e análise de requisitos

O sucesso tecnológico assenta numa fase inicial estruturada de acordo com o levantamento e análise de requisitos, tendo esta etapa como objetivo compreender as necessidades dos utilizadores, as condições técnicas da infraestrutura envolvida, os recursos disponíveis e as limitações tecnológicas associadas. As reuniões iniciais permitiram traçar um conjunto de requisitos fundamentais para orientar o processo de desenvolvimento da aplicação.

2.2 Requisitos Funcionais

RF1 — A aplicação deve mostrar o ambiente imersivo do parque eólico.

RF2 — A aplicação deve recolher os dados da *API OpenMeteo.*

RF3 — A aplicação deve demonstrar os dados recolhidos da *API* através de um painel.

RF4 — A aplicação deve demonstrar turbinas eólicas.

RF5 — As eólicas devem ajustar a rotação das pás de acordo com os dados da *API*.

RF6 — A aplicação deve ter *sliders* para ajuste de valores como simulação.

RF7 — As turbinas eólicas devem ajustar a rotação de acordo com os *sliders*.

RF8 — As eólicas devem rodar o rotor horizontalmente de acordo com os valores da API.

RF9 — As eólicas devem rodar o rotor horizontalmente de acordo com os valores do *slider*.

RF10 — A aplicação deve conter um *prefab* de forma a simular o oceano.

RF11 — O oceano deve ter ondas ajustáveis a partir de valores da API.

RF12 — O oceano deve ter ondas ajustáveis de acordo com valores do *slider*.

RF13 —A aplicação deve conter um painel com as fórmulas utilizadas.

RF14 — A aplicação deve conter um painel com os *sliders* de interação.

RF15 —A aplicação deve conter um painel para o ajuste de turbinas na cena.

RF16 —A aplicação deve conter a opção de ajustar o número de eólicas em cena.

RF17 —A posição das turbinas deve ser ajustável.

RF18 — A aplicação deve conter um sistema de partículas para simular vento.

RF19 — A aplicação deve conter um sistema de partículas para simular chuva.

RF20 — A aplicação deve conter um sistema de partículas para simular neve.

RF21 —A aplicação deve conter um sistema de partículas para simular granizo.

RF22 —O vento deve ser ajustável através de valores da *API*.

RF24 —A chuva deve ser ajustável através de valores da *API*.

RF25 —A chuva deve ser ajustável através do respetivo *slider.*

RF26 —A neve deve ser ajustada através de valores da *API*.

RF27 —A neve deve ser ajustada através dos *sliders.*

RF28 —O granizo deve ser ajustado através da *API*.

RF29 —O granizo deve ser ajustado através do respetivo slider.

RF30 —A aplicação deve conter um *toggle* para trocar entre os valores da API e dos *sliders*.

RF31 —A aplicação deve ser capaz de ser visualizada através dos *Oculus Meta Quest 3*.

RF32 — A aplicação deve ser controlada através de comandos do Meta *Quest 3*.

RF33 —As turbinas devem possuir um painel que demonstra a potência, as rotações por minuto e a temperatura do motor.

2.2 Requisitos Não Funcionais

RNF1 — A aplicação deve ter atualizada em tempo real.

RNF2 — A aplicação deve ser compatível com o Meta Quest 3.

RNF3 — A aplicação deve ser acessível e intuitiva para o utilizador.

RNF4 — Os dados apresentados deverão ser consistentes, precisos e atualizados de forma periódica, tanto em modo de simulação como em modo de real time.

RNF5— Todos os elementos visuais (textos, gráficos, ícones e animações) deverão manter uma identidade visual uniforme, garantindo legibilidade e estética profissional em todo o ambiente.

RNF6 — A renderização de gráficos, animações, mar, turbinas e partículas meteorológicas deverá ser otimizada para garantir um bom desempenho gráfico sem comprometer a fluidez da experiência.

2.3 Contextualização e Localização

Com a implementação deste sistema na zona da Póvoa de Varzim (Aguçadoura), região costeira do norte de Portugal, esta escolha impacta diretamente os requisitos operacionais da aplicação, principalmente no que diz respeito às condições meteorológicas, acessibilidade ao parque eólico, infraestrutura de comunicação disponível e necessidades específicas dos operadores locais.

O contexto offshore implica desafios adicionais quando comparado com instalações *onshore*, nomeadamente:

- Exposição constante a ambientes marítimos adversos;
- Acesso físico limitado para manutenção;
- Maior dependência de tecnologias de monitorização remota e comunicação fiável;
- Elevada complexidade na recolha e interpretação de dados em tempo real.

Estes fatores exigem que a aplicação desenvolvida proporcione uma interface intuitiva, robusta e informativa, permitindo aos técnicos e operadores realizar uma supervisão eficaz do estado de cada aerogerador, mesmo em condições adversas.

3. Definição de modelos 3D e Arquitetura

3.1 Ambiente Marinho e Terrestre

Na fase inicial de desenvolvimento, procedeu-se à definição da estrutura base da aplicação, incluindo a organização dos elementos 3D, o sistema de visualização imersiva e os modelos a integrar. Esta etapa envolveu a seleção criteriosa de *assets* e componentes gráficos compatíveis com a *pipeline URP* no *Unity*.

Primeiramente para representar o ambiente marítimo envolvente dos parques eólicos offshore, foi necessário incluir uma superfície de água 3D que transmitisse realismo visual. Após análise de várias opções disponíveis na *Unity Asset Store*, foi selecionado o *asset "Simple Shader Water"*, um *shader* de água leve, com suporte para possível integração para animação de ondas, reflexos e movimento fluido.

A utilização deste *asset* permitiu escolher qual o material mais adequado uma vez que este veio com alguns materiais para escolha, o que acelerou o processo do primeiro protótipo. A integração do mesmo foi realizada numa fase inicial, permitindo validar rapidamente o aspeto do cenário marinho.

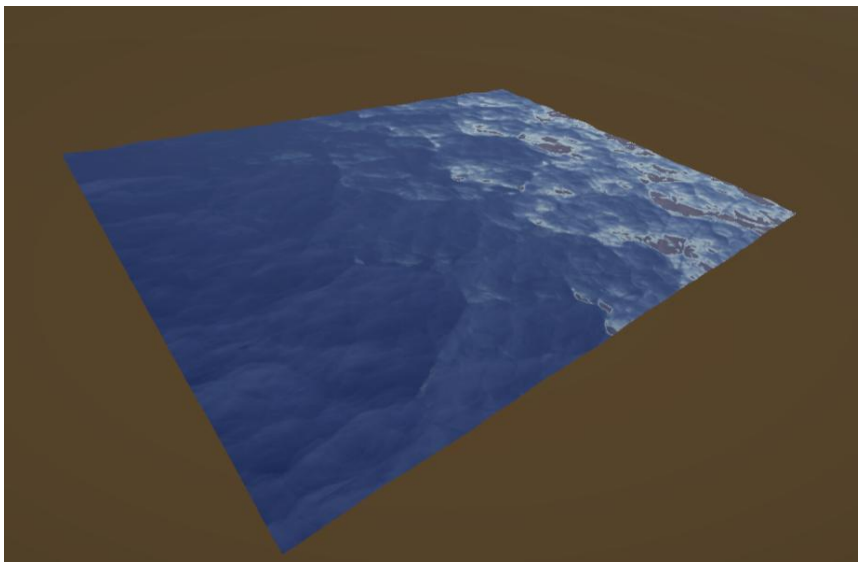

Figura 1: Simple water shader.

Adicionou-se também um cubo com a textura de terra, formando um bloco terrestre no ambiente.

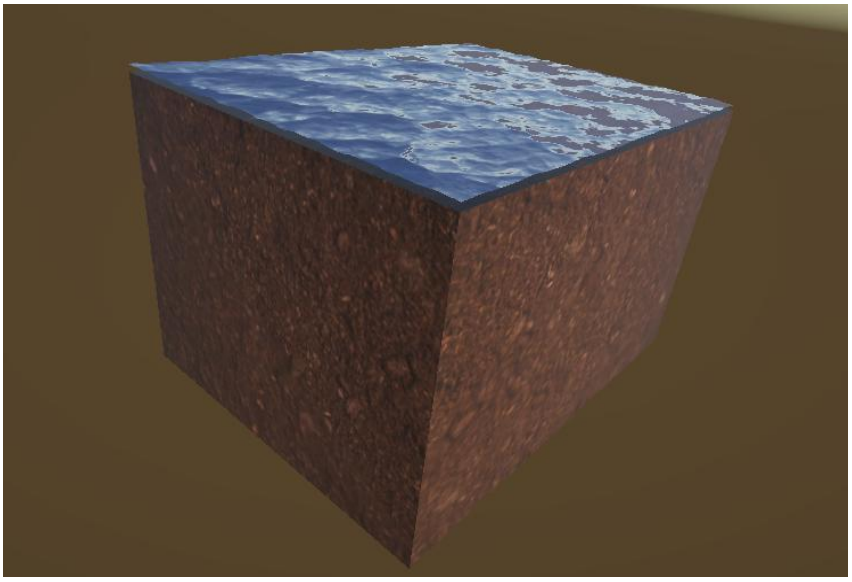

Figura 2: Bloco terrrestre.

3.2 Protótipo Inicial da Turbina Eólica

Além da definição do ambiente marítimo e terrestre, procedeu-se à criação de um protótipo funcional de turbina eólica, com o objetivo de validar a representação 3D e a lógica de rotação dos componentes.

Nesta fase, não se pretendeu ainda alcançar fidelidade visual, mas sim desenvolver um modelo simplificado para testar animações, interações, através de *scripts C#* e o posicionamento no espaço tridimensional.

Para esse fim, recorreu-se a formas geométricas básicas disponíveis no *Unity*:

- As pás da turbina foram representadas por cubos alongados, organizados em torno de um eixo central;
- O rotor foi representado por uma esfera, que funcionava como base rotativa das pás;
- O corpo da turbina foi moldado através de um cubo fino, apenas com função estrutural.

Esta abordagem modular simples permitiu testar a lógica de rotação, a interação do utilizador com elementos individuais, bem como a visualização dos primeiros testes de posicionamento geoespacial no cenário 3D.

Figura 3: Protótipo eólica.

3.3 Integração do Controlo da Velocidade do Vento

De seguida, foram realizados testes de interação com elementos de interface gráfica UI, nomeadamente *sliders,* com o objetivo de controlar dinamicamente a velocidade do vento que influencia a rotação das pás da turbina. Para tal, foi criado um *canvas* na janela de visualização, dentro do qual foi adicionado um *panel* denominado *"Eolic Estimation Panel"*, que serviu como espaço dedicado à experimentação de componentes da interface.

Para melhorar a acessibilidade da interface, foi também inserido um botão junto ao painel, com a funcionalidade de abrir e fechar o *"Eolic Estimation Panel"*. Este botão foi ligado a um *script* em *C#*, que permite ativar ou desativar a visibilidade do painel com um simples clique, contribuindo para uma experiência de utilizador mais limpa e adaptável ao contexto visual.

Esta funcionalidade revelou-se útil durante os testes, permitindo ao utilizador ocultar o painel quando não era necessário, mantendo o foco no ambiente tridimensional e nas eólicas.

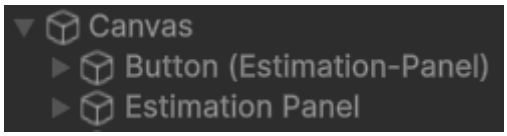

Figura 4: Criação do canvas, do "Estimation Panel" e butão na hierarquia.

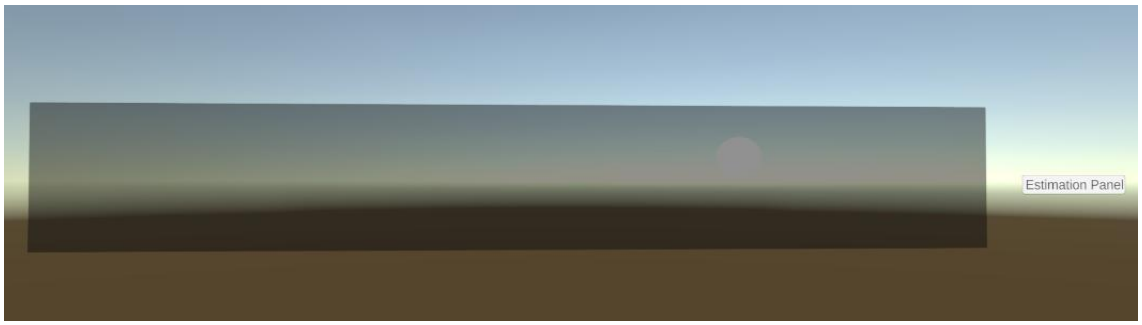

Figura 5: "Estimation Panel".

Neste painel foi introduzido um *slider* simples de controlo da velocidade do vento, com o objetivo de testar a relação direta entre a intensidade do vento e o comportamento mecânico da turbina. Este elemento de interface apresentava a velocidade atual do vento em km/h e permitia ao utilizador ajustá-la manualmente.

Para complementar, foi também criado um indicador visual da velocidade do vento sendo um campo de texto (*TMP_Text*) posicionado acima do *slider*, que exibe em tempo real o valor selecionado pelo utilizador, expresso em km/h. Este valor varia entre 0 e 150 km/h, intervalo previamente definido no código para representar de forma realista as possíveis intensidades do vento. Esta funcionalidade permite ao utilizador ter uma perceção imediata do valor exato aplicado à simulação, contribuindo para uma interação mais intuitiva com o sistema.

Figura 6: "Slider velocidade do vento".

Para simular a influência da velocidade do vento no comportamento da turbina, foi desenvolvido um script no qual a velocidade do vento, expressa em km/h, é convertida utilizando a função Mathf.Lerp(), que realiza uma interpolação linear entre dois valores.

```
float speedInKmH = Mathf.Lerp(0f, 150f, Slider1.value);
windspeed = speedInKmH;
```

Figura 7: "speedInKmH".

O intervalo definido, entre 0 e 150 km/h, foi escolhido com base em dados reais, considerando que este valor corresponde aproximadamente à maior rajada de vento já registada em Portugal, segundo fontes noticiosas [7]. Essa escolha permite abranger uma gama realista de velocidades de vento, possibilitando a observação de variações significativas no desempenho da turbina durante os testes de simulação.

Após a conversão, a velocidade é transformada para metros por segundo (m/s), sendo esta a unidade universal apropriada para os cálculos físicos subsequentes:

```
float velocityMetric = windspeed / 3.6f;
```

Figura 8: "velocityMetric".

Com a velocidade do vento em m/s, procede-se ao cálculo da velocidade angular do rotor (Ω), com base na seguinte fórmula:

```
float Omega = (TSR * velocityMetric) / raiopa;
```

Figura 9: "Omega".

No contexto deste projeto:

- **TSR** (Tip Speed Ratio) foi definido como 7, valor típico para turbinas de três pás;
- **velocityMetric** representa a velocidade do vento em m/s;
- **raiopa** corresponde ao raio das pás da turbina, estabelecido em 107 metros.

Assim o cálculo do RPM irá ser:

```
rpm = (Omega * 60) / (2 * Mathf.PI);
```

Figura 10: "rpm".

Para dar resposta ao comportamento visual da turbina, foi necessário aplicar a rotação calculada (em rotações por minuto) aos elementos tridimensionais que compõem o rotor e as pás. Para isso o rotor da turbina passou a funcionar como pivot (definido no centro da esfera) que foi inserido para ser o centro de massa na rotação das pás aerodinâmicas. Este objeto-pivot foi associado a um *array* de *GameObjects*, através da variável pública fans, permitindo que todos os pivots fossem manipulados programaticamente. Desta forma, tornou-se possível controlar mais do que um centro de rotação, ou seja, mais do que uma eólica.

Para centralizar e organizar melhor este processo, foi criado um *GameObject* denominado *WindScaleManager*. A este objeto foi atribuído o script C# criado (WindScaleManager.cs) responsável pela lógica de rotação e cálculo, permitindo assim associar dinamicamente diferentes turbinas através da interface do *Unity,* sem necessidade de inserir o script diretamente em cada pivot.

No método *Update()* do script (WindScaleManager.cs), foi adicionado um ciclo *foreach* que percorre todos os objetos incluídos nas fans e aplica-lhes uma rotação contínua em torno do eixo Z (neste caso, negativo), simulando o movimento das pás da turbina no sentido horário:

```
foreach (GameObject fan in fans)
{
    if (fan != null)
    {
        float rotationSpeed = (rpm / 60) * 360 * Time.deltaTime;
        fan.transform.Rotate(0, 0, -rotationSpeed);
    }
}
```

Figura 11: "Ciclo foreach".

Assim, o valor de rpm calculado anteriormente é convertido em graus por segundo, multiplicando por 360 (graus por rotação) e ajustado pelo *Time.deltaTime* para garantir uma rotação suave e consistente com a taxa de atualização do motor gráfico.

3.4 Controlo da Ondulação e outras Dinâmicas

Após a segunda reunião, foi-nos fornecido o modelo da turbina eólica, desta vez com um aspeto realista.

Para além disso à semelhança do *slider* para controlo da velocidade do vento, foi implementado adicionalmente outro *slider,* mas desta vez dedicado ao controlo da altura das ondas. Este elemento de interface tem como objetivo permitir ao utilizador testar diferentes intensidades de ondulação no mar, afetando visualmente a superfície da água no ambiente virtual.

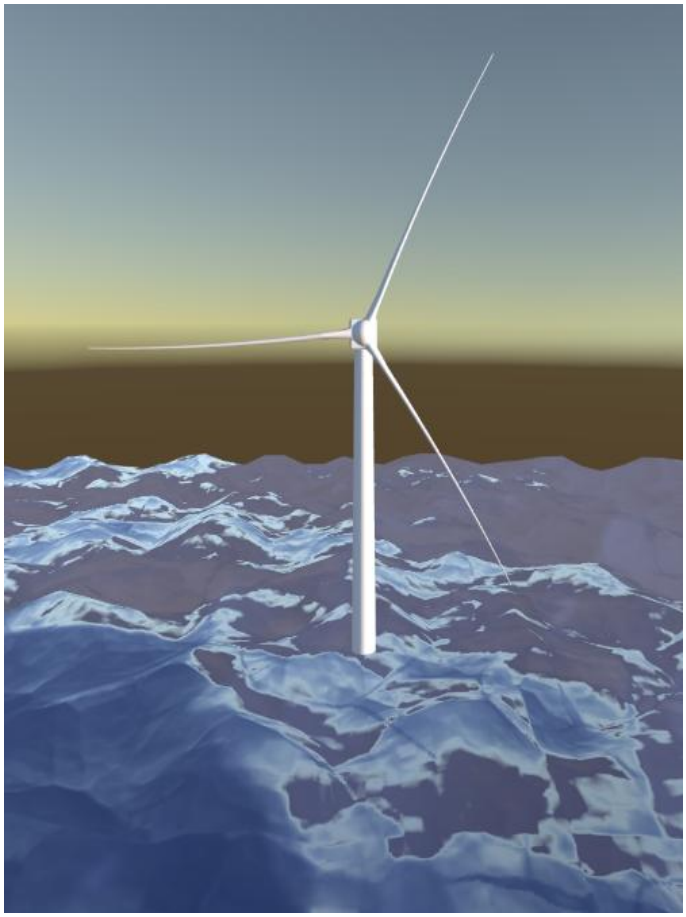

Figura 12: "Prefab Eólica".

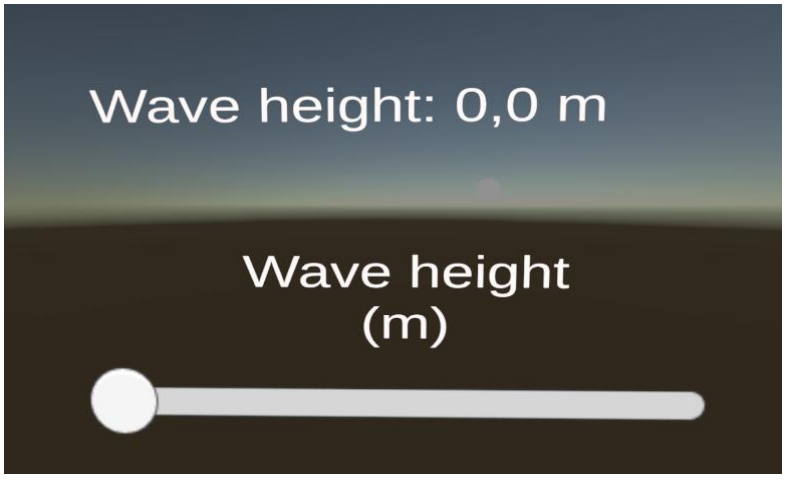

Figura 13: "Slider altura das ondas (m)".

Relativamente ao *slider* da altura das ondas, este apresenta valores compreendidos entre 0 e 3 metros. Este intervalo foi escolhido de forma a garantir uma representação visual realista do mar, permitindo simular desde condições de mar calmo até situações de forte agitação. Optou-se por não aumentar o valor máximo, uma vez que valores superiores causavam artefactos visuais no *shader*, resultando em picos pouco naturais na superfície da água. Ao ajustar este valor, o utilizador modifica, em tempo real, o parâmetro responsável pela deslocação vertical das ondas no material da água, o que se traduz numa simulação visual da ondulação mais ou menos intensa.

```
displacementSlider.minValue = 0;
displacementSlider.maxValue = 3;
```

Figura 14: "Valor min e max estipulado".

Para complementar este controlo, foi também introduzido um campo de texto *(TMP_Text)*, posicionado acima do *slider*, que exibe em tempo real o valor da altura das ondas selecionado pelo utilizador, expresso em metros (m).

O comportamento do *slider* foi implementado num *script C#* chamado *WaterDisplacementController.cs,* que contém a lógica de aplicação do valor do *slider* ao material da água, bem como a atualização do texto de feedback visual. Para isso, foram utilizadas instruções simples que atualizam o *shader* do mar com o novo valor, como exemplificado abaixo:

```
void ApplyWaveHeight(float value)
{
    if (waterMaterialInstance != null)
    {
        waterMaterialInstance.SetFloat(displacementProperty, value);
    }
}
```

Figura 15: "Função ApplyWaveHeight".

Durante a constante evolução a partir da terceira reunião com os orientadores foi escolhido o segundo *prefab* da turbina eólica realista como modelo principal, tendo sido descartadas as versões anteriores. Foi selecionado como final, uma vez que inclui a funcionalidade de rotação horizontal do rotor, a qual foi aproveitada e complementada com a implementação de um *slider* num script associado, permitindo que a rotação "girassol" seja controlada manualmente pelo utilizador.

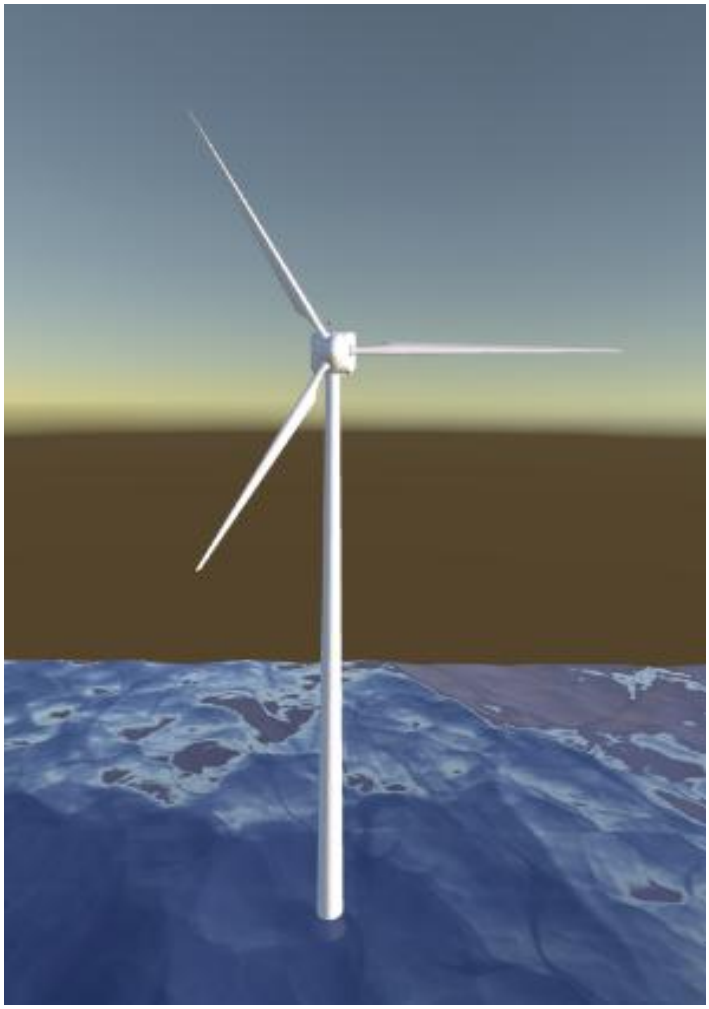

Figura 16: "Eólica Final".

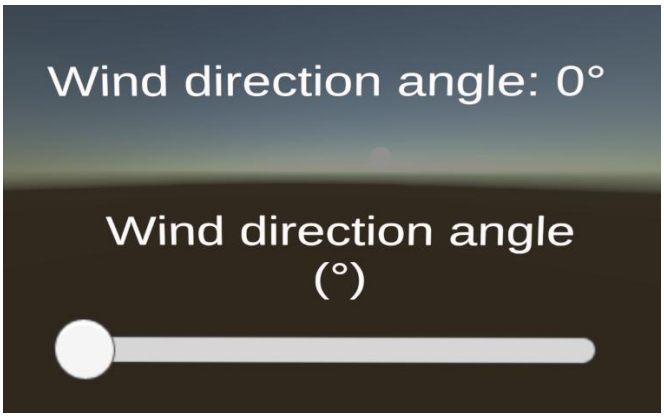

Figura 17: "Slider direção do vento (º)".

Este *slider* permite ao utilizador ajustar o ângulo de orientação da turbina entre 0 e 360 graus, simulando a rotação da estrutura conforme desejado. A posição do rotor influencia diretamente a direção em que as pás aerodinâmicas apontam, o que proporciona maior interatividade e controlo sobre o comportamento do sistema.

```
windDirection = Mathf.Lerp(0f, 360f, directionSlider.value);
UpdateDirectionText(windDirection);
```

Figura 18: Definição do ângulo de 0 a 360 graus no slider.

```
foreach (GameObject fan in fans)
{
    if (fan != null)
    {
        fan.transform.rotation = Quaternion.Euler(0f, -windDirection, 0f);
    }
}
```

Figura19: Aplicação da rotação aos objetos "fans" com base na direção do vento.

A lógica implementada verifica constantemente o valor do *slider* dentro da função *Update()*, e a orientação de cada um dos objetos definidos como "fans" é atualizada dinamicamente. A rotação é aplicada no eixo Y com base no valor selecionado. Além disso, um campo de texto foi incluído para exibir o ângulo atual, como foi anteriormente realizado para a velocidade do vento e ondulação das ondas.

Através de um novo pivot definido para a rotação horizontal, garantimos que o movimento ocorra corretamente em torno do eixo vertical da base do rotor, mantendo a coerência visual e estrutural do modelo 3D da turbina. Esta configuração permite que a rotação aconteça de forma precisa e centrada, evitando deslocamentos indesejados durante a interação com o *slider*. Assim, a simulação mantém-se fiel ao comportamento esperado de uma turbina eólica, reforçando o realismo da interface desenvolvida.

3.5 Sistema Meteorológico

Para a meteorologia, desenvolvemos um sistema de partículas dinâmico, concebido para simular de forma visualmente apelativa e realista três fenómenos climáticos distintos: a chuva, a neve e o granizo. O objetivo primordial foi criar um ambiente interativo, em que a intensidade destes efeitos meteorológicos fosse ajustável tal como para a velocidade do vento, ondulação e angulo de direção do vento já mencionadas, através de *sliders*.

Cada efeito climático foi implementado como um sistema de partículas individual, aproveitando as robustas capacidades de *Visual Effects (VFX)* do *Unity*. A configuração de cada *VFX* foi meticulosamente moldada para replicar as características visuais específicas de cada fenómeno natural.

Por exemplo, as partículas de chuva foram ajustadas para simular a queda de gotas com especial atenção à velocidade, dimensão e densidade das partículas para um efeito de precipitação fidedigno. Para a neve, as partículas foram configuradas para um movimento mais lento e uma forma que evoca a leveza dos flocos de neve a cair suavemente. Já no caso do granizo, as partículas foram desenhadas para serem mais densas e com um movimento mais abrupto, simulando a queda caraterística de pequenas pedras de gelo. Foram usados diferentes png para a textura destas partículas.

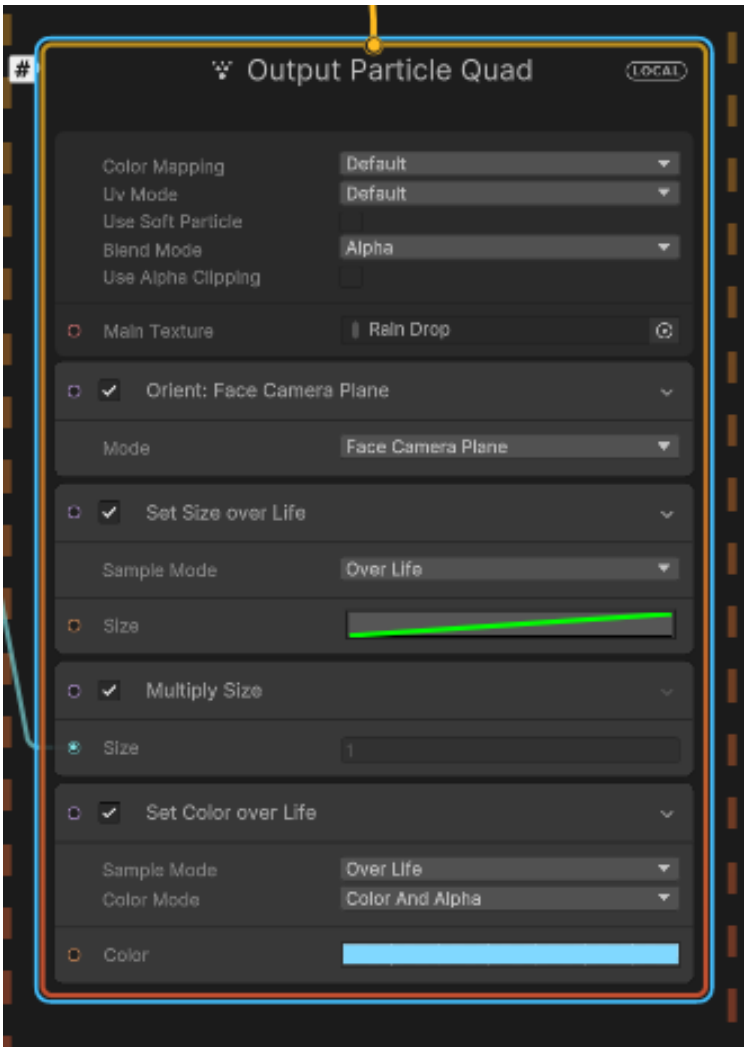

Figura 20: Exemplo da definição de cor da partícula chuva no VFX.

A gestão da intensidade de cada efeito climático é realizada através de um *GameObject* dedicado, que nomeámos *WeatherManager*. Este *GameObject* aloja o script *WeatherManager.cs,* que atua como o centralizador de toda a lógica de controlo e interação. O script *WeatherManager.cs* permite ao utilizador ajustar a intensidade da chuva, neve e granizo através de três sliders específicos, integrados no panel do canvas UI. À medida que o valor de cada slider é modificado pelo utilizador, o script ajusta de imediato o número de partículas geradas pelo respetivo sistema *VFX*, o que proporciona um controlo granular e visualmente percetível sobre a densidade e o impacto de cada fenómeno atmosférico.

```
private void Start()
{

    RainVFX.SetFloat("Intensity", RainSlider.value);
    SnowVFX.SetFloat("Intensity", SnowSlider.value);
    HailVFX.SetFloat("Intensity", HailSlider.value);


    RainSlider.onValueChanged.AddListener(UpdateRain);
    SnowSlider.onValueChanged.AddListener(UpdateSnow);
    HailSlider.onValueChanged.AddListener(UpdateHail);
}

1 referência
private void UpdateRain(float value)
{
    RainVFX.SetFloat("Intensity", value);
}

1 referência
private void UpdateSnow(float value)
{
    SnowVFX.SetFloat("Intensity", value);
}

1 referência
private void UpdateHail(float value)
{
    HailVFX.SetFloat("Intensity", value);
}
```

Figura 21: WeatherManager.cs.

Assim no script, *WeatherManager.cs* são estabelecidos "*listeners*" para cada um destes *sliders;* esta configuração garante que, sempre que o valor de um *slider* é modificado pelo utilizador, a função de atualização correspondente, *UpdateRain()*, *UpdateSnow()* ou *UpdateHail()* seja automaticamente invocada. Desta forma as funções são responsáveis por receber o novo valor transmitido pelo *slider* e utilizá-lo para ajustar a propriedade *"Intensity"* do respetivo Visual Effect.

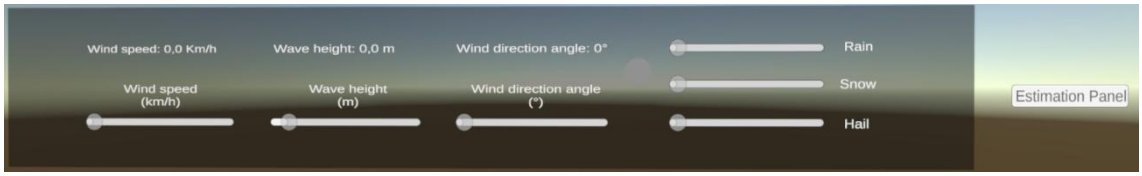

Figura 22: Sliders da meteorologia (chuva,neve,vento) no panel.

Figura 23: Hierarquia do canvas.

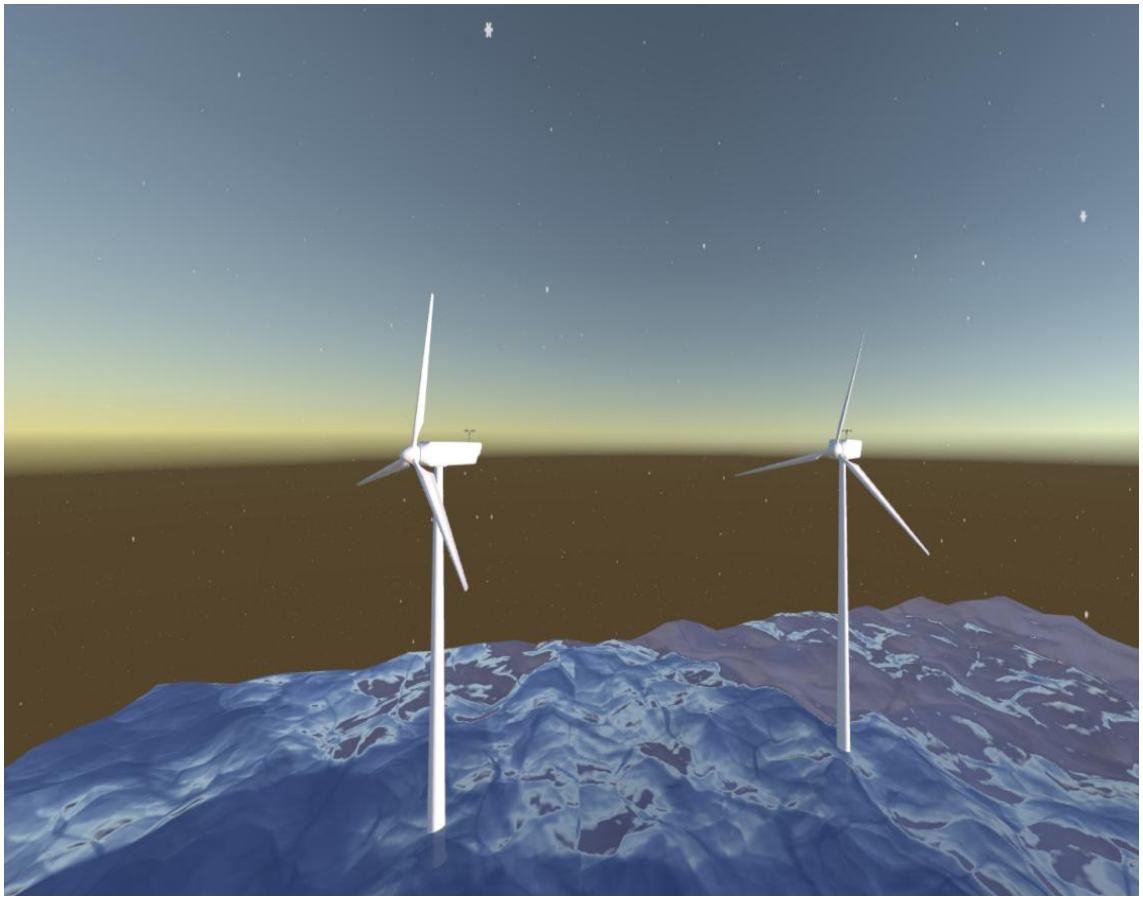

Figura 24: Neve.

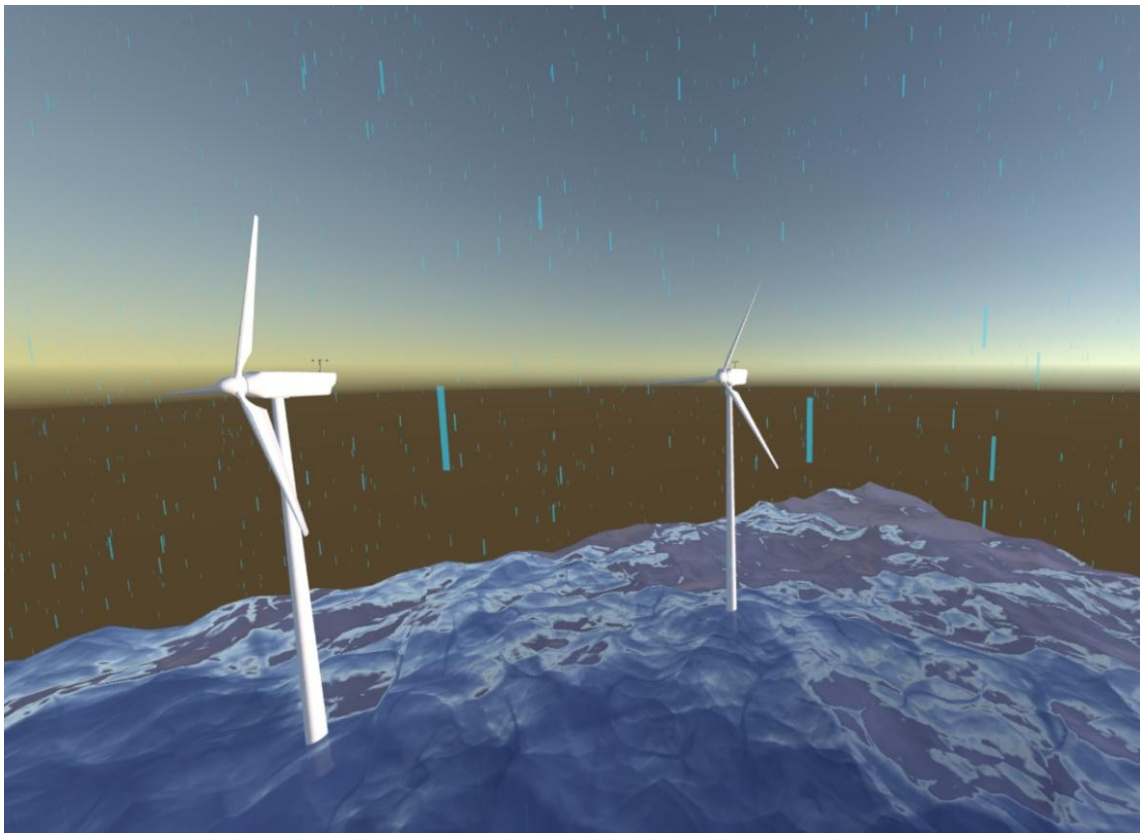

Figura 25: Chuva.

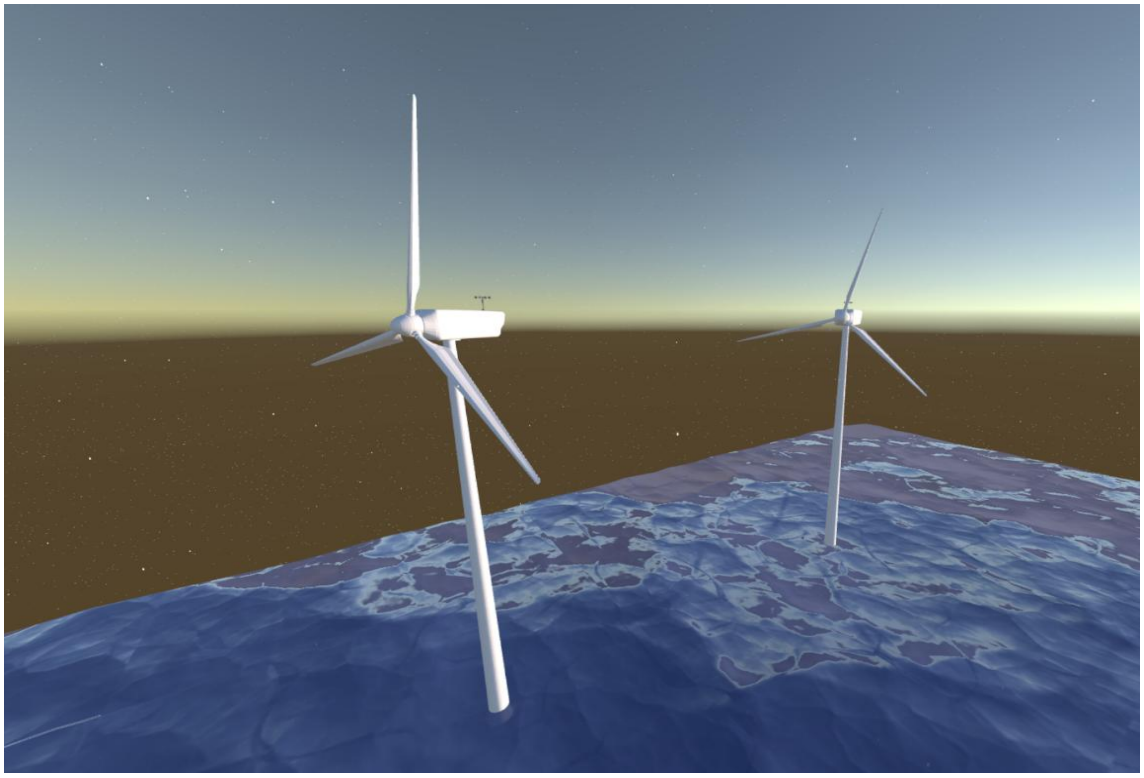

Figura 26: Granizo.

3.6 Prefab Vento

Para complementar os efeitos atmosféricos existentes e aumentar ainda mais o realismo da simulação, foi integrado um *prefab* de vento no projeto. A inclusão deste *asset* foi sugerida e fornecida pelos nossos orientadores. Este *prefab* permite simular a influência do vento nos elementos circundantes do ambiente, aprimorando a imersão visual e a credibilidade das condições climáticas apresentadas.

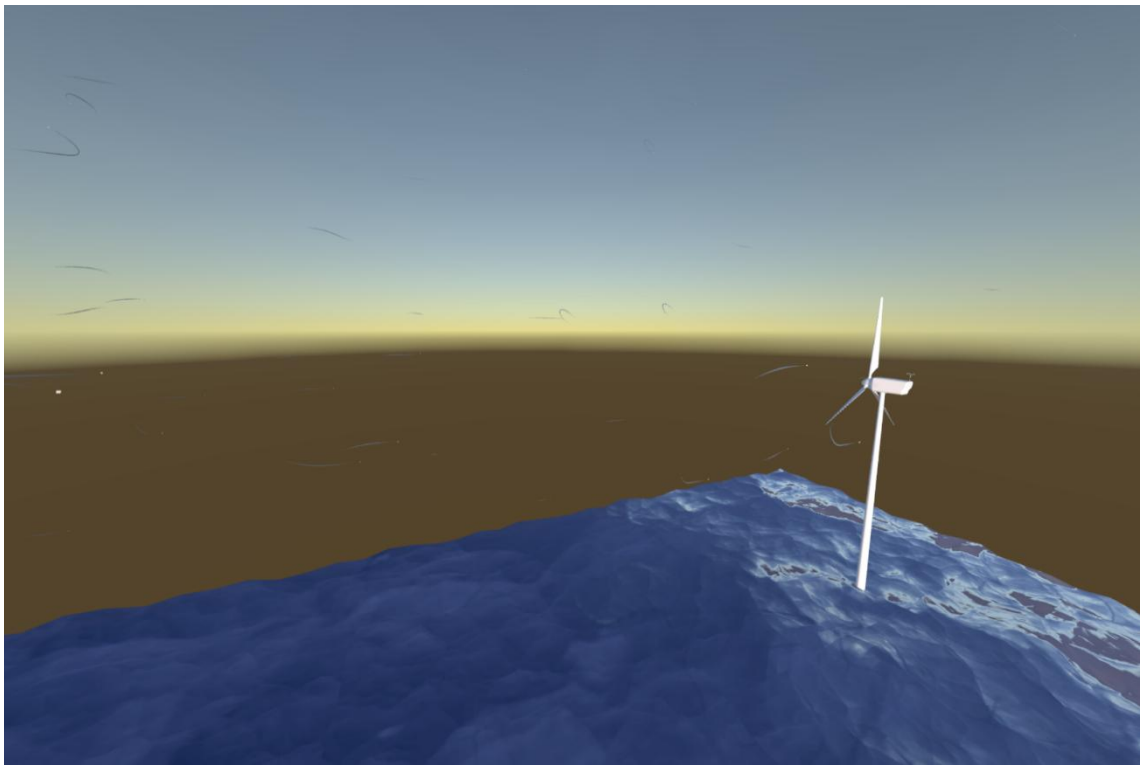

Figura 27: Asset vento.

4. Integração de *API Open-Meteo*

No decorrer do desenvolvimento do nosso projeto, fizemos a integração da *API Open-Meteo* que desempenhou um papel essencial na obtenção de dados meteorológicos cruciais. Esta *API* foi nos fornecida como referência para utilização pelos orientadores e, desde o início, permitiu-nos aceder a informações relevantes sobre a Póvoa de Varzim, local previamente estipulado para análise. Através dela, conseguimos recolher dados climáticos precisos, garantindo a fiabilidade das simulações e estudos realizados ao longo do projeto.

4.1 Compreensão do funcionamento da *API*

Antes de proceder à implementação, começámos por explorar o funcionamento da *API Open-Meteo*, analisando a sua documentação e as diferentes variáveis meteorológicas disponíveis.

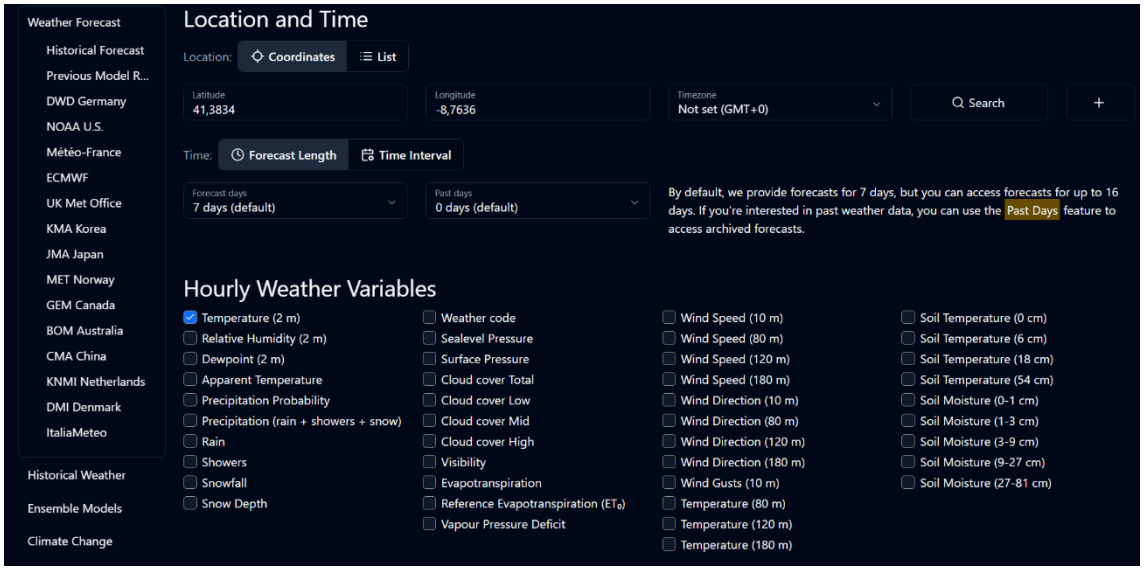

Figura 28: Interface Open-Meteo.

Exploramos que as informações meteorológicas pretendidas podem ser acedidas através de um link gerado pela resposta da *API*, que contém os dados estruturados com base na variável específica solicitada. Ao realizar as requisições adequadas, conseguimos visualizar a meteorologia diretamente através do link.

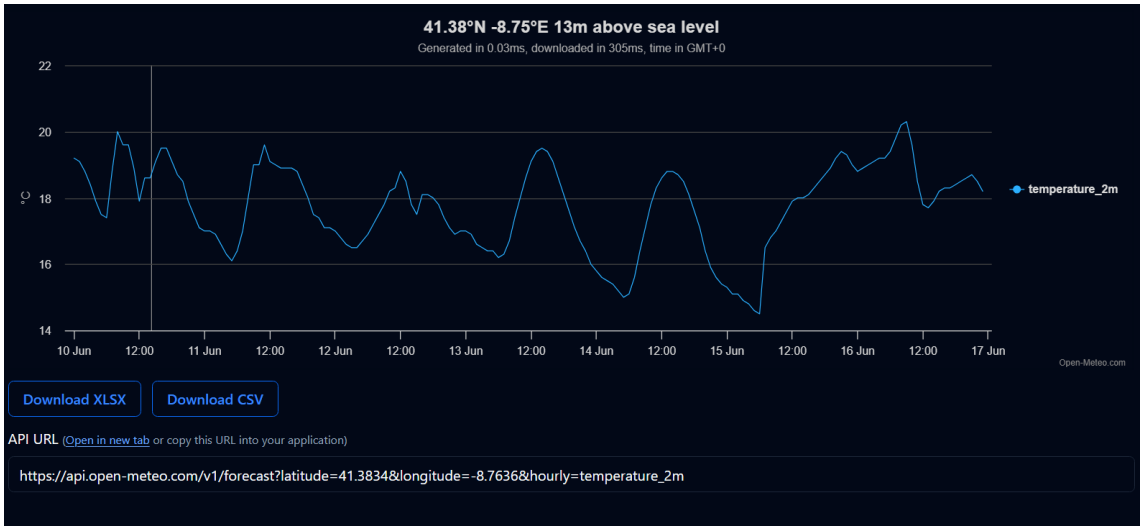

Figura 29: API URL (response) da variável temperatura.

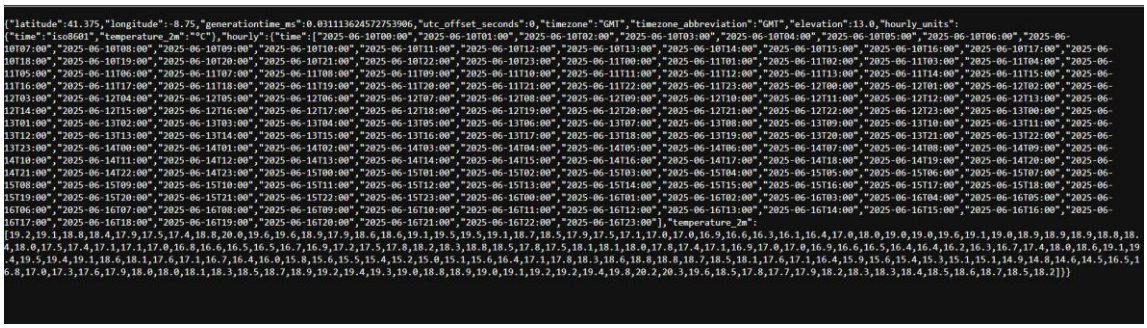

Figura 30: Temperaturas definidas por hora da API URL.

4.2 Criação do painel "*Real-Time*"

Para facilitar a visualização dos dados em tempo real, construímos um outro painel no *canvas (“Real-Time panel”)* do ambiente, onde integramos as informações meteorológicas obtidas através da *API Open-Meteo*. Este painel segue a mesma abordagem utilizada do “*Estimation Panel*” o que permite a interação intuitiva através de um botão. Com esta implementação, conseguimos então reunir as condições necessárias para começar a colocar as informações da *API*.

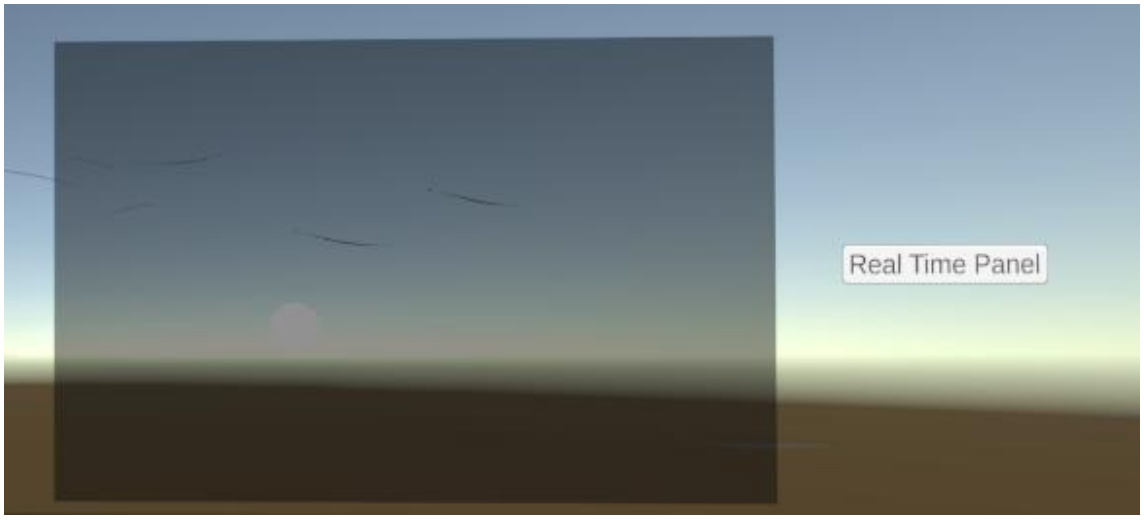

Figura 31: Real-Time Panel.

Para apresentar os dados meteorológicos em tempo real de forma clara e acessível, adicionámos um *TMP Text* ao painel *“Real-Time Panel”* no *Unity*, onde é exibida a temperatura atual obtida através da API Open-Meteo.

Para garantir a atualização automática dos valores, desenvolvemos um script em C#, denominado *CurrentTemperature*, que realiza requisições periódicas à API e processa a resposta JSON para extrair e exibir a informação relevante. Este *script* foi integrado diretamente no painel, garantindo que os valores meteorológicos aparecem corretamente no ambiente gráfico do *Unity*. A implementação segue um intervalo de atualização de 60 segundos, permitindo que os dados meteorológicos sejam atualizados de forma dinâmica e em tempo real.

```
public class CurrentTemperature : MonoBehaviour
{
    private string apiUrl = "https://api.open-meteo.com/v1/forecast?latitude=41.3834&longitude=-8.7636&current_weather=true";
    private float updateInterval = 60f;


    public TMP_Text temperatureText;

    Mensagem do Unity | 0 referências
    void Start()
    {
        StartCoroutine(GetCurrentTemperature());
    }

    Mensagem do Unity | 0 referências
    void Update()
    {

    }

    1 referência
    IEnumerator GetCurrentTemperature()
    {
        while (true)
        {

            using (UnityWebRequest request = UnityWebRequest.Get(apiUrl))
            {
                yield return request.SendWebRequest();

                if (request.result == UnityWebRequest.Result.Success)
                {

                    string jsonResponse = request.downloadHandler.text;
                    WeatherResponse weatherResponse = JsonUtility.FromJson<WeatherResponse>(jsonResponse);

                    if (weatherResponse != null)
                    {
                        float currentTemperature = weatherResponse.current_weather.temperature;
                        Debug.Log("Temperature: " + currentTemperature + "°C");


                        if (temperatureText != null)
                        {
                            temperatureText.text = "Temperature: " + currentTemperature.ToString("F1") + "°C";
                        }
                    }
                }
                else
                {
                    Debug.LogWarning("Error finding temperature: " + request.error);
                }
            }


            yield return new WaitForSeconds(updateInterval);
        }
    }
```

Figura 32: Script "CurrentTemperature.cs".

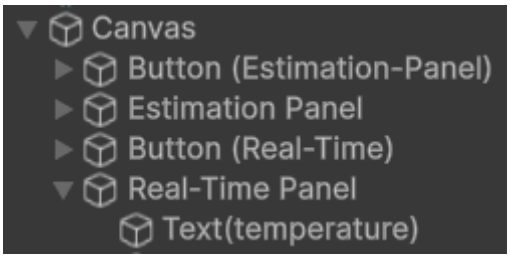

Figura 33: "Text(temperature) adicionado à hierarquia do canvas".

Após isto foi também adicionado um *script* com a hora atual.

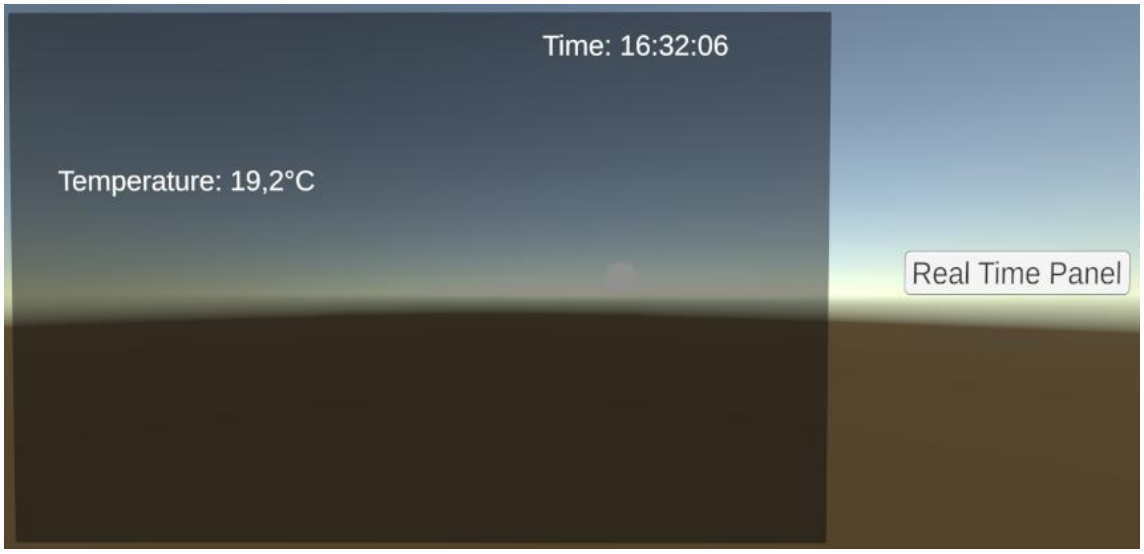

Figura 34: "Real-Time panel com a temperatura fornecida pela API Open-Meteo".

Assim seguindo a lógica do script específico para a variável temperatura implementamos mais informações para o *panel* com as diferentes *URLs* de reposta que a *Open-Meteo* disponibiliza. Adicionamos para além da temperatura a velocidade do vento, ângulo de direção do vento, altura das ondas, humidade, nível da pressão do mar, precipitação (Chuva/Granizo), cobertura das nuvens baixa, total, alta, media.

Além disso, foi implementado um *toggle* que ativa o modo de atualização em tempo real. Quando este modo está ativo, os dados de velocidade do vento, altura das ondas e ângulo da direção do vento são calculados automaticamente com base nos valores recebidos da API, e as eólicas no mar e a respetiva rotação horizontal passam a comportar-se como se estivessem no local real. Esta funcionalidade foi integrada nos scripts WindScaleManager.cs, WaterDisplacementManager.cs e WaterDirectionManager.cs, garantindo que as atualizações sejam refletidas corretamente no sistema.

Para além disso, os *sliders* do *"Estimation Panel"* são automaticamente desativados aquando da ativação do *toggle*, através da associação a um novo script assegura a consistência e fidelidade dos dados face às condições meteorológicas reais.

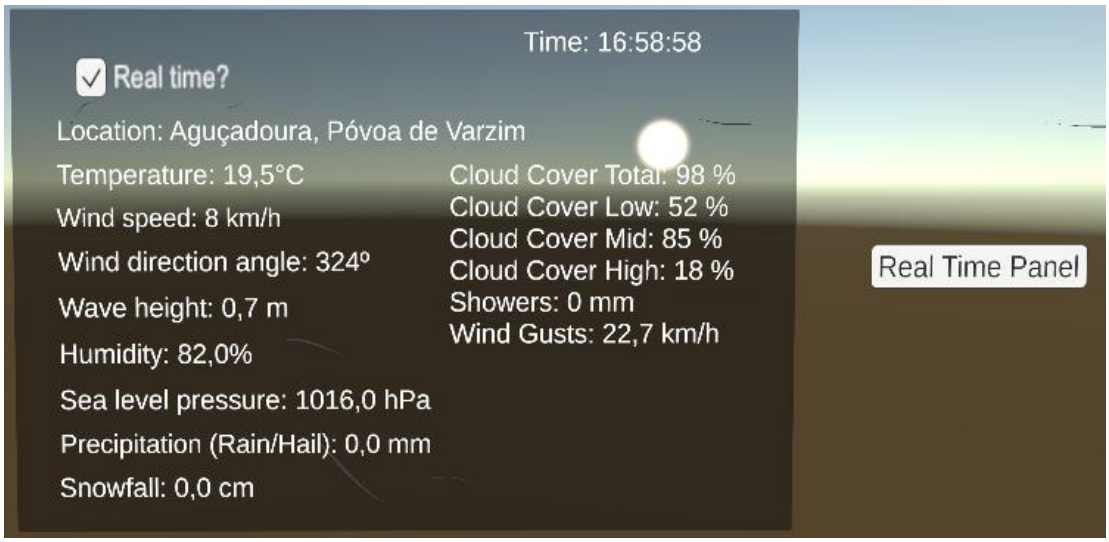

Figura 35: "Real-Time panel" com as variáveis implementadas escolhidas na API".

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

Script do Unity (2 referências de ativo) | 0 referências
public class CheckboxSliderControl : MonoBehaviour
{
    [SerializeField]
    private List<Slider> slidersToBlock;
    [SerializeField]
    private Toggle checkbox;

    Mensagem do Unity | 0 referências
    private void Start()
    {

        UpdateSliderState(checkbox.isOn);


        checkbox.onValueChanged.AddListener(UpdateSliderState);
    }


    2 referências
    private void UpdateSliderState(bool isChecked)
    {
        foreach (Slider slider in slidersToBlock)
        {
            if (slider != null)
                slider.interactable = !isChecked;
        }
    }
}
```

Figura 36: "Script CheckBoxControl".

4.3 Fórmulas

Este painel foi concebido com o objetivo de disponibilizar aos especialistas em energia eólica uma visão clara e imediata dos principais cálculos integrados na aplicação, garantindo que os fundamentos técnicos subjacentes à simulação do parque eólico sejam facilmente compreensíveis. Ao apresentar as fórmulas essenciais e os respetivos parâmetros, o painel funciona como uma referência rápida, permitindo a validação dos dados e a verificação da coerência dos resultados gerados pela plataforma.

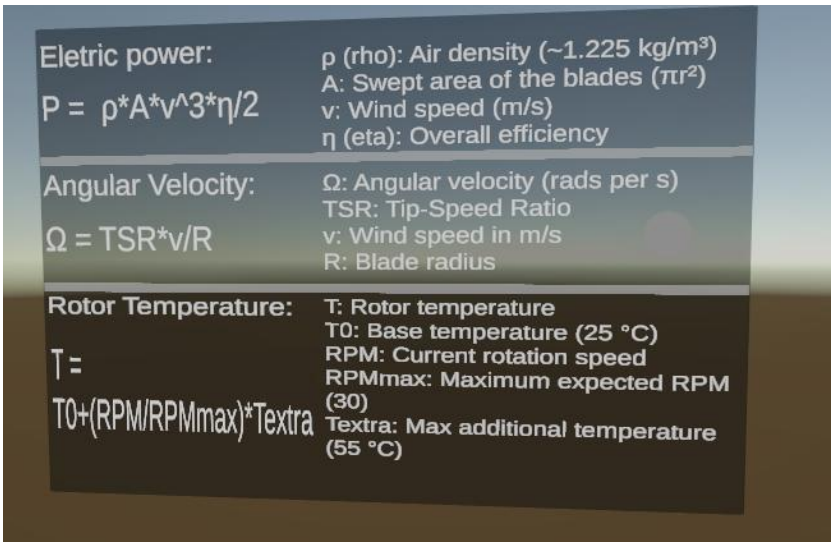

Figura 37: Painel das fórmulas.

A primeira fórmula apresentada refere-se ao cálculo da potência elétrica (P), expressa pela equação P=ρ·A·(v^3)/2. Aqui, *ρ* representa a densidade do ar, um valor que, em condições padrão, se aproxima de 1.225 kg/m^3 e que condiciona a quantidade de energia cinética disponível no vento. O parâmetro A corresponde à área varrida pelas pás do aerogerador, calculada com base no raio das mesmas (πr^2), enquanto *v* simboliza a velocidade do vento em metros por segundo. Esta equação traduz, de forma simplificada, a relação cúbica entre a velocidade do vento e a energia produzida, sublinhando a importância de locais com ventos consistentes para a eficiência dos parques eólicos.[6]

O segundo bloco de equações dedica-se à velocidade angular (Ω), determinada pela expressão Ω=TSR·v/R. Neste contexto, *TSR* (Tip-Speed Ratio) é um adimensional crítico que define a eficiência aerodinâmica das pás, estabelecendo a relação entre a velocidade na ponta da pá e a velocidade do vento incidente. *R* indica o raio da pá,

e *v* mantém-se como a velocidade do vento. Este cálculo é fundamental para ajustar a rotação do rotor às condições atmosféricas, assegurando que o sistema opera dentro dos limites ideais de desempenho.[4]

Por fim, o painel inclui uma fórmula para estimar a temperatura do rotor (T), dada por T=T0·(RPM/RPMmax)·Textra. T0 corresponde à temperatura base, fixada em 25 °C, enquanto RPM e RPMmax representam, respetivamente, a velocidade de rotação atual e a máxima esperada (neste caso, 30 RPM). O termo Textra reflete o acréscimo máximo de temperatura (55 °C), permitindo modelar o aquecimento do rotor em função da sua atividade. Este aspeto é particularmente relevante para monitorizar o desgaste mecânico e prevenir sobrecargas, contribuindo para a gestão proativa da manutenção.[5]

4.4 Painel Eólica

O painel apresentado é um componente fundamental do sistema de monitorização de turbinas eólicas, permitindo acompanhar em tempo real o desempenho de cada unidade individual do parque. O painel exibe três parâmetros essenciais para a avaliação do funcionamento da turbina: as rotações por minuto, a temperatura do rotor e a energia produzida. Estes valores são obtidos através de sensores instalados na própria estrutura da turbina e processados segundo as formas mencionadas, sendo estas documentadas no painel de fórmulas acessível a partir do sistema.

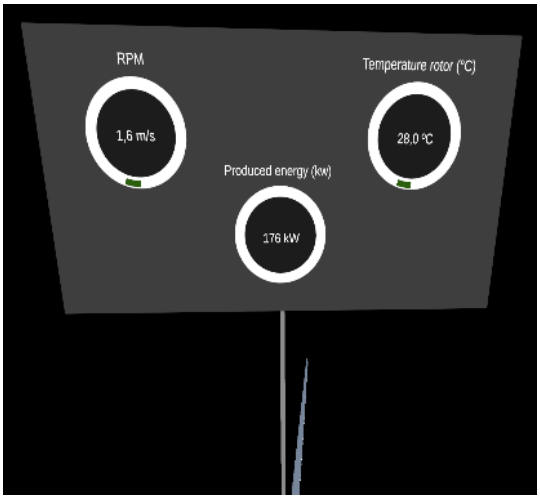

Figura 38: Painel de Monitorização da eólica.

4.5 Painel Adicionar e Remover Eólicas

O painel apresentado na figura tem como função principal permitir a interação dinâmica com o parque eólico simulado, oferecendo aos utilizadores a capacidade de adicionar ou remover aerogeradores em tempo real. Esta funcionalidade foi concebida para proporcionar uma experiência interativa, onde é possível observar imediatamente o impacto das alterações na configuração do parque na produção total de energia.

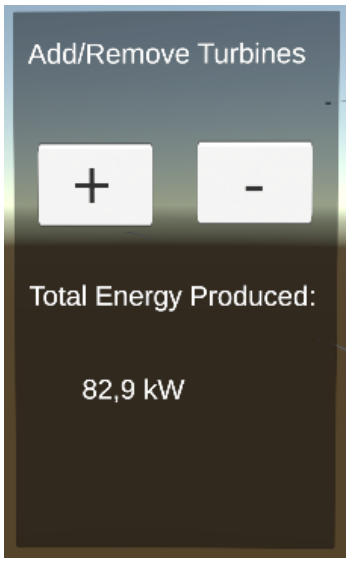

Figura 39: Painel adicionar/remover eólicas.

Na parte inferior do painel, é exibido de forma clara e destacada o valor atual da energia total produzida, que neste caso específico se situa nos 82,9 kW. Este indicador numérico atualiza-se automaticamente sempre que ocorre uma modificação no número de aerogeradores ativos, servindo como um feedback instantâneo sobre o desempenho global do sistema.

O painel utiliza um *script* chamado *WindTurbineAlocation*, que gera uma grelha virtual onde as turbinas são posicionadas. Inicia com duas turbinas pré-colocadas e oferece métodos para adicionar novas unidades *(AddFan)* ou remover as existentes *(RemoveFan)*. Cada turbina adicionada é automaticamente registada nos sistemas de controlo de velocidade e direção do vento, garantindo que o seu movimento reflete as condições ambientais simuladas.

```
2 referências
public void AddFan()
{

    int freeIndex = GetFirstFreeGridIndex();
    if (freeIndex == -1)
    {
        Debug.Log("Grid cheia!");
        return;
    }

    Vector3 spawnPosition = GetPositionFromIndex(freeIndex);
    GameObject newFan = Instantiate(fanPrefab, spawnPosition, Quaternion.identity);
    newFan.SetActive(true);
    newFan.transform.SetParent(null);

    FanDragg dragScript = newFan.GetComponent<FanDragg>();
    if (dragScript != null)
        dragScript.gridManager = this;


    fans[freeIndex] = newFan;

    Transform rotatingPart = newFan.transform.Find("Dragger/pivotsiderotation/bladespivot");
    if (windScaleManager != null && rotatingPart != null)
        windScaleManager.fans.Add(rotatingPart.gameObject);

    Transform rotatingWindPart = newFan.transform.Find("Dragger/pivotsiderotation");
    if (windDirectionManager != null && rotatingWindPart != null)
        windDirectionManager.fans.Add(rotatingWindPart.gameObject);

    Debug.Log("Ventoinha adicionada no slot " + freeIndex);
}
```

Figura 40: função AddFan().

O cálculo da energia total é atualizado constantemente no método *Update*, multiplicando a energia base (fornecida pelo gestor de vento) pelo número de turbinas ativas. Este valor é apresentado num elemento de interface gráfica, formatado com uma casa decimal para maior precisão.

```
Mensagem do Unity | 0 referências
void OnMouseDown()
{
    isDragging = true;
    offset = transform.position - GetMouseWorldPosition();
    lastValidPosition = transform.root.position;

    if (gridManager != null)
    {
        // Usa o GameObject raiz da turbina
        gridManager.ClearFanFromGrid(transform.root.gameObject);
    }
}
```

Figura 41: função OnMouseDown().

5. Implementação *AR* e Controladores *Meta Quest 3*

Após diversas tentativas de integração, deparámo-nos com um desafio significativo que foi a adaptação e incorporação dos controladores em toda a arquitetura previamente desenvolvida. Embora tenhamos envidado vários esforços para integrar estes elementos de forma harmoniosa no sistema existente, verificámos inúmeras dificuldades, sobretudo na implementação do *raycasting* associado aos controladores, cuja deteção, tanto no *canvas* como nas estruturas eólicas, revelou-se repetidamente ineficaz.

Apesar de procurarmos ajustar e otimizar a lógica implementada, os resultados não corresponderam às expectativas, e a integração continuou a apresentar falhas que comprometiam a estabilidade e a funcionalidade da aplicação. Perante este cenário, e com o objetivo de garantir uma base sólida para continuar o desenvolvimento, decidimos recorrer a uma abordagem alternativa, utilizámos um sample de um projeto base disponibilizado pelo *Meta Interaction SDK*, reconhecido pela sua fiabilidade e estrutura bem definida. A partir deste ponto de partida mais robusto, procedemos à importação do nosso projeto existente, com a expectativa de facilitar a integração dos controladores e resolver os problemas que vínhamos enfrentando de forma mais eficaz e sustentável.

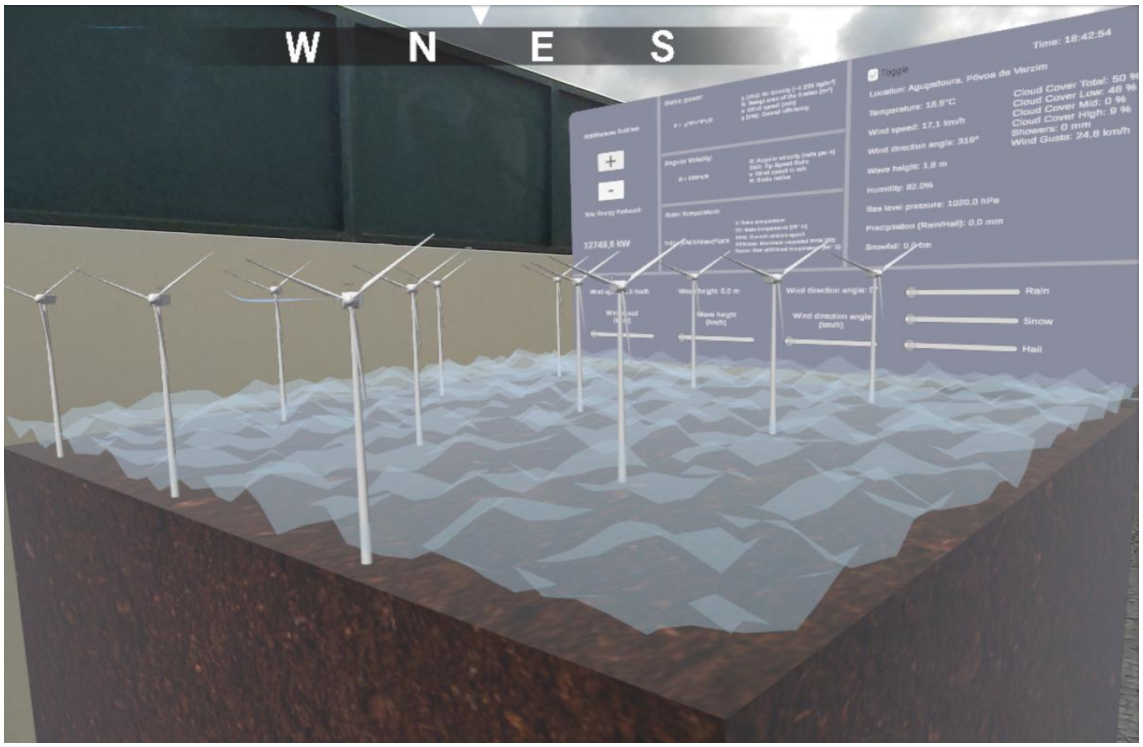

Figura 42: Aplicação final e atualização do painel.

Assim, atualizámos também o nosso *canvas*, agora com suporte a AR e interações por clique, garantindo maior estabilidade e funcionalidade na experiência desenvolvida, para além disso também foi implementada uma bussola, podendo fazer a orientação através dos pontos cardeais.

Adicionalmente, implementamos uma funcionalidade que permite a abertura do painel eólico sempre que o botão "A" do controlador direito é pressionado. Este painel surge associado à última estrutura eólica (*clone*) adicionada ao cenário, garantindo que a interação ocorre sempre com o elemento mais recentemente instanciado. Esta funcionalidade contribui para uma experiência mais interativa e contextualizada, permitindo ao utilizador aceder a informações específicas sobre os elementos do ambiente em realidade aumentada de forma intuitiva.

Figura 43: Ativação do panel da eólica através do controlador direito.

6. Escalabilidade do projeto

Apesar dos resultados alcançados e da funcionalidade sólida da aplicação desenvolvida, é importante reconhecer aspetos que poderiam ser aperfeiçoados numa iteração futura deste projeto, seja em termos técnicos, visuais ou de usabilidade.

Um dos aspetos identificados prende-se com o comportamento das turbinas eólicas em condições adversas. Atualmente, as eólicas continuam a operar mesmo em cenários simulados de tempestade, vento extremo ou granizo intenso. Na realidade, turbinas offshore são frequentemente desligadas automaticamente quando são atingidos certos limites de segurança, quer por excesso de velocidade do vento, quer por risco estrutural. A implementação de lógica de paragem automática das turbinas nestas situações, aumentaria o realismo do simulador e permitiria explorar também o impacto da indisponibilidade temporária na produção energética.

Por fim, a aplicação está atualmente limitada à simulação do parque eólico da Aguçadoura, na Póvoa de Varzim. Embora esta localização tenha sido estrategicamente escolhida, uma futura versão poderia permitir ao utilizador selecionar diferentes zonas geográficas, com base num mapa interativo ou coordenadas personalizadas, integrando automaticamente dados da *API Open-Meteo* relativos à nova localização. Esta funcionalidade aumentaria significativamente a flexibilidade da aplicação, permitindo comparar condições operacionais entre diferentes regiões e adaptando-se a múltiplos cenários de estudo ou demonstração.

Para além disso, a simulação considera que todas as turbinas eólicas produzem exatamente a mesma quantidade de energia, o que não corresponde à realidade. No mundo real, há sempre variações na produção de cada turbina, devido a fatores como diferenças de vento, localização e margens de erro nos sistemas.

Estas melhorias não comprometem o valor atual da solução desenvolvida, mas abrem caminho para uma evolução contínua do projeto, tornando-o mais robusto, realista e adaptável a contextos diversos.

7. Conclusão

O presente projeto materializou-se numa aplicação inovadora que alia a realidade aumentada à monitorização de parques eólicos offshore, constituindo uma verdadeira prova de conceito para o futuro da gestão energética inteligente. Durante o desenvolvimento, foram ultrapassados vários desafios técnicos, especialmente na integração dos controladores, na criação de ambientes 3D realistas e na adaptação para os óculos Meta Quest 3, sempre com atenção à interação com as interfaces gráficas, garantindo uma experiência imersiva, intuitiva e centrada no utilizador.

Este projeto representa, acima de tudo, um marco no aprofundamento das competências em engenharia informática, combinando conhecimentos de programação, modelação tridimensional, realidade aumentada e análise de dados em tempo real. A sua realização evidencia o valor do trabalho colaborativo, da iteração constante com os orientadores e da capacidade de transformar conceitos avançados em soluções concretas, funcionais e tecnologicamente pertinentes.

Num mundo em crescente transição para fontes de energia limpa e inteligente, esta aplicação posiciona-se como um contributo relevante para a inovação no setor, abrindo caminho a futuras investigações e desenvolvimentos no domínio da visualização imersiva de infraestruturas críticas. A conclusão deste projeto não assinala um fim, mas sim o início de novas possibilidades na interseção entre tecnologia, sustentabilidade e realidade aumentada.

8. Bibliografia

[1] Direção-Geral de Recursos Naturais, Segurança e Serviços Marítimos (DGRM). (s.d.). *WebGIS – Portal de Informação Geográfica*

[2] Open-Meteo. *for latitude 41.3834, longitude –8.7636* [Documentação da API]. Open-Meteo.

[3] INESC TEC. *Blue-X*.

[4] Energy Sustainability Directory. *Tip speed ratio*.

[5] Energy Systems Integration Group. *Modeling of Type 1 Wind Turbine Generators*. Retrieved June 10, 2025

[6] LinkedIn. *Como calcular a potência de uma turbina eólica?*

[7] Diário de Notícias. (2018, 14 de dezembro). Mau tempo: Rajada de vento na Figueira da Foz foi a maior registada em Portugal.

[8] Unity Asset Store.

[9] Boschert, S., & Rosen, R. (2016). Next generation digital twin. In A. Kusiak (Ed.), Proceedings of the 2016 TMCE Conference (pp. 1–11). ResearchGate.

[10] Bilgili, M., Yasar, A., & Simsek, E. (2011). Offshore wind power development in Europe and its comparison with onshore counterpart. Renewable and Sustainable Energy Reviews, 15(2), 905–915.