Sørensen *et al. Energy Informatics* 2022, **5**(Suppl 4):39
https://doi.org/10.1186/s42162-022-00227-2

Energy Informatics

RESEARCH **Open Access**

Potentials of game engines for wind power digital twin development: an investigation of the Unreal Engine

Jonas Vedsted Sørensen, Zheng Ma* and Bo Nørregaard Jørgensen

From Energy Informatics.Academy Conference 2022 (EI.A 2022)
Vejle, Denmark. 24-25 August 2022

*Correspondence:
zma@mmmi.sdu.dk

SDU Center for Energy Informatics, Maersk Mc-Kinney Moller Institute, University of Southern Denmark, 5230 Odense, Denmark

Abstract

Digital twin technologies have become popular in wind energy for monitoring and what-if scenario investigation. However, developing a digital representation of the wind is challenging, especially due to the digital twin platform constraints. Game engines might be possible to solve this issue, especially since game engines have been used for product design, testing, prototyping, and also digital twins. Therefore, this study investigates the potential of developing a digital twin of wind power in the Unreal game engine. A case study of two types of wind turbines (Vestas V164-8 and Enercon E-126 7.580) and one location (Esbjerg, Denmark) is chosen for this study. The digital twin includes the environment with historical wind data and the visual representation of the wind turbine with a wind power production model and the estimated production in the given wind conditions of the area. The results show that game engines are viable for building entire digital twins where a realistic graphical user interface is required. Unreal Engine 5 provides the tools for modelling the landscape, surrounding water, and lighting. In addition, the Unreal Engine ecosystem provides vast amounts of content, such as 3D assets and game logic plugins, easing the digital twin development. The results prove that digital twins built in Unreal Engine 5 have great potential development of digital twins and user interfaces for communicating with a digital twin. The developed digital twin allows for further extension to benefit future digital twins utilizing wind turbines.

Keywords: Digital twin, Wind energy, Game engine, Unreal Engine, Simulation

Introduction

Increasing the implementation of renewable energy resources in the energy systems is critical for reaching the global and local climate goals (Christensen et al. 2019). For instance, the European Union has decided that the capacity of offshore wind energy is to be expanded to 300 GW before 2050 (Larsen 2022). To reach this goal, much more and better wind turbines are required. Wind turbine technologies have been developed

comprehensively in recent years, and there is a great implementation of wind turbines, especially in Denmark and China. However, there are still many challenges to the massive adoption of wind turbines. For instance, noise and natural life disturbance due to wind turbines, climate change-related natural disasters' impacts on wind power production (Ma et al. 2019a, b). Therefore, pre-investigation, analysis, and planning are essential for the success of the implementation, especially in the selection of wind turbine sites (Ma et al. 2017).

Usually, onsite inspections are conducted before the installation of wind turbines. However, onsite inspection only can collect static information. Even with historical data, onsite investigation is not good enough to ensure the success of the wind turbine implementation. Therefore, modelling and simulations of wind turbines and wind farms have been well developed to conduct what-if scenario investigation and testing. Lately, digital twin technologies have become popular in wind energy. Different types of simulations have been used popularly in the energy sector, e.g., discrete event simulations (Ma et al. 2020) and agent based simulations (Værbak et al. 2019). A digital twin supports the connection to the real-world environment and allows for real-time visualization and estimated energy production. However, simulations do not include this integration, but simulations are sually used for the prediction capabilities in digital twins.

The applications of digital twin technology in wind energy in the literature mainly focus on the wind turbine, e.g., the blade (Fahim et al. 2022), or wind farms (Ospina-Bohórquez et al. 2022), with the main purposes of monitoring (Moghadam and Nejad 2022) or maintenance (Moghadam et al. 2021). However, there are challenges in the development and implementation of most applications. For instance, the developed digital twins are difficult to be integrated with other applications. There is no graphical interface where the third parties, e.g., residents and wind turbine contractors, can understand or participate in developing the digital twins. Furthermore, the developed digital twins only focus on the turbine or multiple turbines (farm), but the wind (environment) is neglected. There is no doubt that wind is an important element in the digital twins of wind energy. However, it is difficult to develop a digital representation of the wind, especially due to the digital twin platform constraints.

In recent years, game engines have been used for product design, testing, prototyping, marketing, and digital twins. The two top game engines in the market are Unreal and Unity. Both Unreal and Unity have claimed their ambitions in digital twin applications. For instance, many digital twins of industrial facilities have been developed in the Unity engine, e.g., the digital twin of Hyundai's actual factory (Chang-won 2022). Comparatively, besides industrial facilities, several digital twins of cities have been developed in the Unreal Engine e.g., 51 World modelling Sydney harbour (51World: 51World https://www.51vr.com.au/). There are many similarities and differences between these two game engines, and one of the significant differences is that the Unreal Engine has a higher-quality physical environment imitation.

Therefore, this study aims to investigate the potential of developing a digital twin of wind power in the Unreal Engine. A digital twin prototype is developed in Unreal Engine 5 with a 3D visualization for an area where various wind turbine models can be placed. A case study of two types of wind turbines (Vestas V164-8 and Enercon E-126 7.580) and one location (Esbjerg, Denmark) is chosen for this study. The digital twin includes the

environment with historical wind data and the visual representation of the wind turbine with a wind power production model and the estimated production in the given wind conditions of the area.

This report firstly introduces the background of digital twins, followed by the methodology section. The main elements of the developed digital twin prototype for wind energy are introduced in the wind power digital twin architecture section, followed by the scenario design section. Furthermore, the results for each scenario are introduced in the result section, followed by the discussion and conclusion sections.

Literature review

The digital twin is a concept which has been growing rapidly in both industry and academia in the later years (Howard et al. 2021). The first publication presenting digital twins was by Glaessgen and Stargel from their work on NASA air force vehicles in 2012. Their vision was a high-fidelity digital version of a cyber-physical system to enhance the operation in the physical world. Their definition of a digital twin was: "A Digital Twin is an integrated multiphysics, multiscale, probabilistic simulation of a … system that uses the best available physical models, sensor updates, … history, etc., to mirror the life of its corresponding … twin" (Glaessgen and Stargel 2012). Note that the domain-specific elements are removed from the definition to improve the interpretation.

Although Glaessgen and Stargel defined a digital twin as an ultra-high-fidelity model for simulation, there is still no univocal definition of a digital twin commonly used in either industry or academia (Negri et al. 2020; Örs et al. 2020; Traore 2021; Worden et al. 2020). With definitions varying from a concept to technology, the term and the related research encapsulate district implementations with no limits to what it can solve. Furthermore, the lack of standardization of capabilities, terminology and structure is an increasing problem (Bradac et al. 2019; Mostafa et al. 2021; Redeker et al. 2021). Currently, the digital twin concept encapsulates the digitization of physical systems—the physical twin—to support goals such as real-time system insights, scenario and configuration testing, and optimal automation of physical systems. Systems with these capabilities are often complex structures, making the digital twin—which may combine multiple—a complex structure, often with multiple pieces of distributed software working towards the common goal.

Digital twins can be categorised into digital twin instances and digital twin prototypes. A digital twin instance (DTI) is a digital twin developed from an existing system in the real world—the physical twin. This DTI requires large amounts of data to construct models matching the behaviour of the physical twin. When constructed, the DTI can provide the capability of monitoring, fault detection, scenario testing, and control of the physical twin (Grieves et al. 2017).

The digital twin prototype (DTP) is a variation of the digital twin. A DTP is a digital twin built before or without a physical system. The use of a DTP is to simulate the system's behaviour in a setting without building it, saving on development and testing costs. When the DTP is of the desired quality, it can help construct the physical twin.

The disadvantage for digital twins is primarily the development costs and the required knowledge. The development of digital twins is complex and often time-consuming (Howard et al. 2021). The complexity also increases the cost of maintenance.

The required knowledge to develop a digital twin encapsulates the domain, modelling, software development, and development of graphical interfaces (Grieves et al. 2017).

The literature on digital twins in the energy sector has mainly focused on electricity grids in general, e.g., (Andryushkevich et al. 2019; Steindl et al. 2020; Zhang et al. 2021). So far, little literature on digital twins has focused on renewable energy resources, although there are some discussions regarding digital twins for solar farms (Zohdi 2021) and wind farms (Ospina-Bohórquez et al. 2022), the focuses are not on the development of digital twins, but on monitoring (Moghadam and Nejad 2022) or maintenance (Moghadam et al. 2021).

Recently, gaming technology has developed powerful abilities of sophisticated visualization capabilities that help enhance simulations, from product prototypes to digital twins. Multiple industries have used game engines for product design, testing, prototyping, marketing, and digital twins' functionality. According to Perforce Software's 2021 State of Automotive Software Report (Perforce 2022), 50% of respondents from the automotive industry expressed interest in using game engines in the coming months (Grieves et al. 2017).

Game engines have in recent years realized their potential for developing digital twins due to the fast development of interactive and immersive visual applications. Game engines such as Unity and Unreal Engine both allow for real-time connectivity to the physical twin across multiple devices and platforms and have begun to brand their engines on their capability of digital twin development. The connectivity of the game engines allows for real-time monitoring of the physical twin in 3D, virtual reality, and augmented reality while metrics can be calculated and monitored (Engine 2022; Unity: Digital Twin 2022). Unity and Unreal Engine are similar in many ways, however, the significant difference is that the Unreal Engine has a higher-quality physical environment imitation and focuses more on the environment and world-building. Not only industries but also academia has also applied game engines for the digital twin development in numerous applications, e.g., (Abdallah et al. 2022; Eyre et al. 2018; Kong et al. 2020; Lee et al. 2021; Leskovsk. 2020; Negrin et al. 2021; Pereira and Ellman 2020; Rolle et al. 2021; Shahsavari et al. 2021; Sharotry et al. 2020; Talkhestani et al. 2019).

Methodology

The development of a digital twin of wind power was chosen to be carried out in Unreal Engine 5 (UE5). UE5 was selected due to its physical environment imitation and world-building capabilities allowing for the digital twin to closely imitate the real-world area (the physical twin). For the development of the digital twin in UE5, both Blueprints and C++ were chosen for fast development and flexibility. Blueprints are the built-in graphical scripting system based on nodes that allow interaction with the game engine or extension of complex C++ classes. For more complex implementations, C++ was chosen as it is the default supported programming language in UE5. UE5 Plugins were chosen as part of the setup to allow for the reusability of non-specific area components. Plugins are a collection of code, data, and visual assets which can be added to multiple UE5 projects.

Five steps of wind power digital twin development in unreal engine 5

To develop the digital twin of wind power in UE5, this study applies the following 5 steps: environment setup, wind turbine design, display interface design, data integration design, and wind power production prediction design:

1. Environment setup:
   Environment setup is the creation of a specific area, which aims to create a detailed digital version of the area. The step utilises a range of the built-in world-building tools provided by UE5. The Landscape tool was chosen to create large-scale immersive terrains. Furthermore, the Water System was chosen to create rivers, lakes, and oceans ensuring that the digital twin closely resembles the physical environment. In addition, the Water System allows for physics interactions and fluid simulations. Quixel was used as a 3D content library providing highly detailed assets and scans of real-world objects to be placed in the world.
2. Wind turbine design:
   The design and implementation of specific wind turbines include the functionality and 3D assets. This step aimed to create a UE5 plugin containing 3D wind turbine assets and their associated logical models. Firstly the wind turbine brands and models were selected. The selection was based on the environment type—onshore or offshore—and the data availability.
   Assets that closely resemble the selected wind turbines were acquired from 3D model brokers. The logic for the wind turbine model—rotation speeds and energy production—was implemented in C++ based on a designed inheritance hierarchy. The wind turbine hierarchy design was created to ease the implementation of future wind turbines and expand the collection of wind turbines for future projects.
3. Display interface design:
   Interface design display was performed using Unreal Motion Graphics to display the current wind speed and the energy production information of the wind turbines. The observer design pattern was used to let the interface observe the data and receive notifications whenever a value changes (Gamma et al. 1994).
4. Data integration design:
   Data integration design aims to design the flow to and from the digital twin, including the connection to the physical twin. The data integration was split into two parts, import and export. The data import was designed as an event-driven systems architecture using the publish-subscribe pattern, allowing the digital twin to subscribe to data published from the physical twin and other applications. The data export was designed to periodically publish information from the digital twin for storage and further analysis.
   The event-driven architecture for both import and export ensures that the services are decoupled and allows for replacement and/or addition of services. An example of this is the use of a software application to act as the physical twin to conduct experiments using predefined data. The realisation of the event-driven architecture utilised the MQTT (Message Queue Telemetry Transport) protocol to transport JSON (JavaScript object notation) formatted messages to and from the digital twin. MQTT

was chosen due to its wide popularity within IoT and the digital twin domain as a lightweight publish-subscribe messaging protocol (Gppert et al. 2021; Malakuti et al. 2021; Yun et al. 2017). The messaging format's choice of JSON is because it is lightweight, human-readable, and easy to process (Introducing JSON www.json.org).

5. Wind power production prediction design:

Wind power production prediction design is the design of the prediction capability provided by the digital twin to assess a specific wind turbine in the area. The input parameters for the prediction are the specific wind turbine and wind data for the desired prediction period. Furthermore, the interface was designed to easily select the wind turbine model and input the wind data.

Data sources

To develop and verify the digital twin for wind power, the data used for wind speed in this study is part of the Typical Meteorological Year (2005–2020) from the Photovoltaic Geographical Information System provided by the European Commission (Commission E 2022). This data set contains wind data for a typical year for a given location. Furthermore, the data for the Danish city of Esbjerg (Lat 55.457, Long 8.526) is used in this study, because Esbjerg is well known as a city of onshore and offshore wind farms. The data on wind turbines are gathered through manufacturer specifications and websites such as https://www.thewindpower.net, and https://en.wind-turbine-models.com/, which provide power curves for specific models.

Wind power digital twin architecture

The wind power architecture consists of the wind power digital twin, the physical twin, and their interaction (as shown in Fig. 1). The digital twin contains the following main elements:

- Environment (e.g., wind, land) used for visualising the area
- Connection to Physical Twin
- Physical Twin
- Wind turbine plugin
- Wind
- Input interface
- Prediction

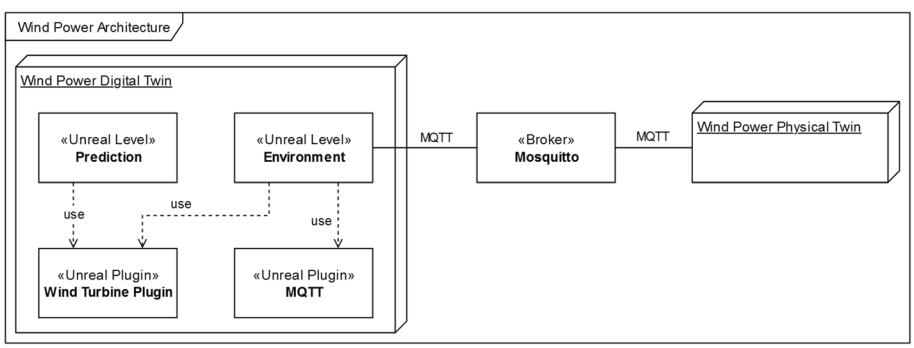

Fig. 1 The wind power architecture

Environment

The physical twin of wind power is a software application created as a stand-in for the real-world environment. The environment utilises the MQTT plugin for receiving changes of state in the physical twin and publishing information for storage. It should be noted that MQTT stands for MQ Telemetry Transport, and the protocol is a set of rules that defines how IoT devices can publish and subscribe to data over the Internet.

In this study, the environment is modelled to match the selected area by first creating an empty world partition level. This creates an empty black world without lighting or a landscape. Hereafter, the lighting and atmosphere were created using the Environment Light Mixer. The Environment Light Mixer eases the process of creating the atmosphere and lighting. The mixer (see Fig. 2) was used to create the sky light, atmospheric light, sky atmosphere, volumetric clouds, and height fog resulting in a naturally lighted environment—see Fig. 3.

Fig. 2 Environment Light Mixer

Fig. 3 Environment after using the Environment Light Mixer

The landscape was created using the landscaping tool provided by Unreal Engine. The texture for the landscape was selected using Quixel Bridge, which provided access to the Quixel MegaScan library. The library contains assets, surfaces, etc. categorized into collections such as broadleaf forest, American sidewalk, wetland, etc. For the landscape, the surface "Beach Sand" was chosen and imported into the project, resulting in the landscape seen in Fig. 4. Hereafter, The water was created using the Water plugin provided by Epic Games—see Fig. 5.

Fig. 4 Environment after creating landscape

Fig. 5 Environment after creating the water

After the environment was created, the wind turbines from the wind turbine plugin could be placed—see Figs. 6 and 7.

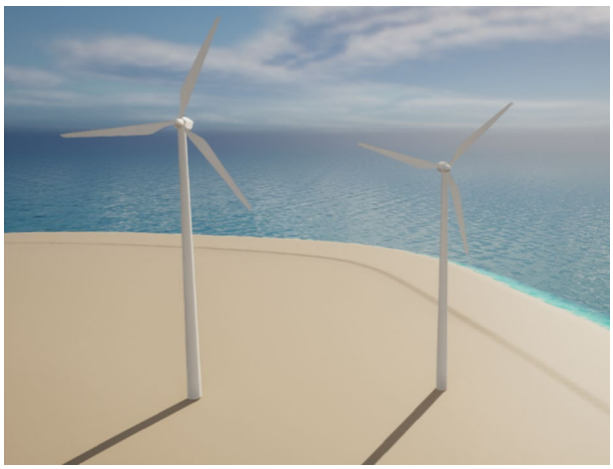

Fig. 6 Screenshot of the environment from the front

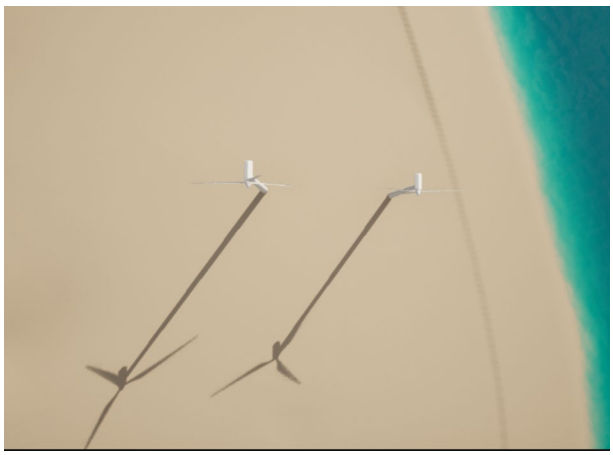

Fig. 7 Screenshot of the environment from the sky

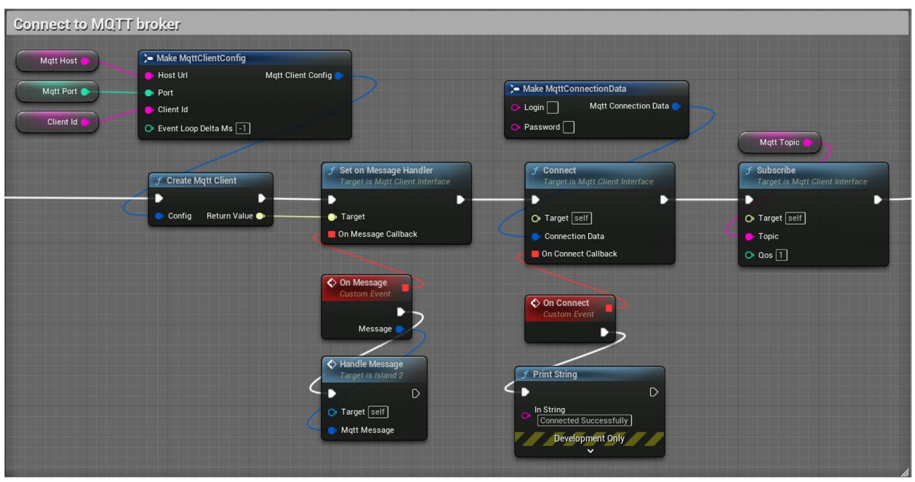

Fig. 8 Initialize connection to MQTT broker

Connection to physical twin

The MQTT connection to the physical twin was implemented in Blueprints using the MQTT Utilities Plugin developed by Nineva Studios (Studios N 2022). The implementation was placed in the Environment level blueprint. This allowed the environment to connect and receive the state of the physical twin. Figure 8 shows the blueprint implementation of the connection to the MQTT broker. First, an MQTT client is created based on an MQTT client configuration containing the host, port and the ID of the client application. Hereafter, the message handler is defined. The message handler is set to the function "HandleMessage" where the MQTT message is passed as an argument (See Fig. 9). After the message handler is configured the client connects to the broker. Lastly, the client is set to subscribe to the topic used by the physical twin to publish its state.

When a message is received from the broker, the function "Handle Message" is executed. The function breaks the message object to examine the topic of the message. Based on the topic, the message is processed accordingly. Figure 9 shows that the topic "wind" triggers the function "Wind Speed Received" followed by "Print Message", whereas messages from other topics only are processed by "Print Message".

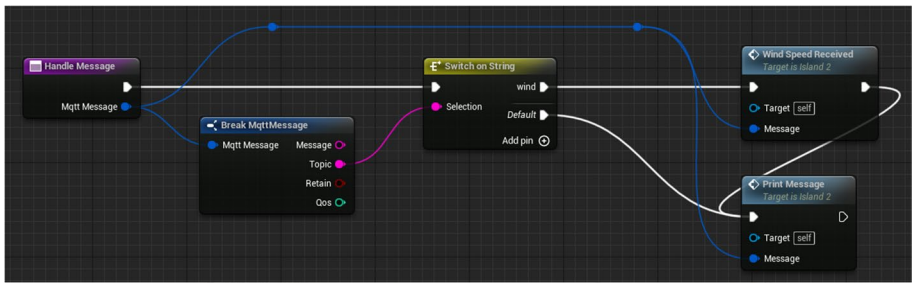

Fig. 9 HandleMessage implementation

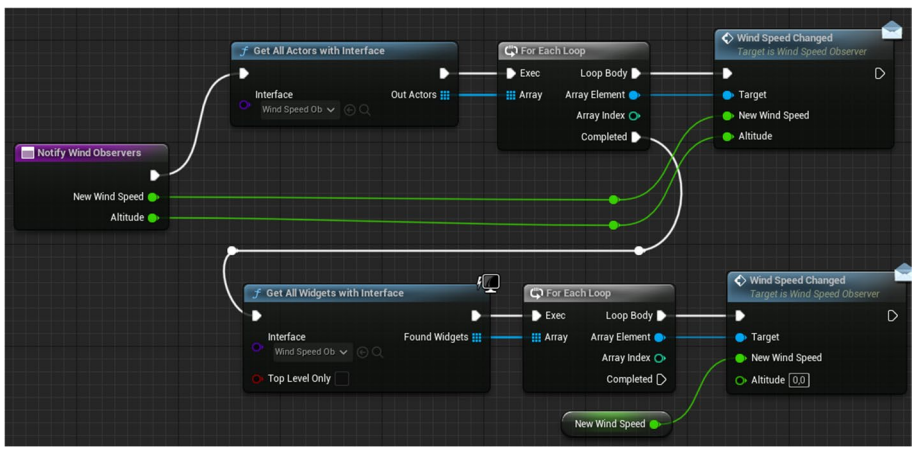

Fig. 10 Notify Wind Observers

The "Wind Speed Received" function extracts the wind speed from the MQTT message object and calls the "Notify Wind Observers" function seen in Fig. 10. The function utilizes the observer pattern by finding all Actors and Widgets in the world implementing the WindSpeedObserver interface and notifying them of the new wind speed and altitude.

Physical twin

To simplify the creation of the digital twin, the physical twin was mocked using a python application allowing the user to notify the digital twin with new wind speeds via MQTT. The python script connects to the MQTT broker and publishes the wind speed before disconnecting again.

Wind turbine plugin

The wind turbine plugin is a UE5 plugin which enables the use of the wind turbine implementations in multiple projects. The plugin consists of visual models and their associated wind turbine implementations.

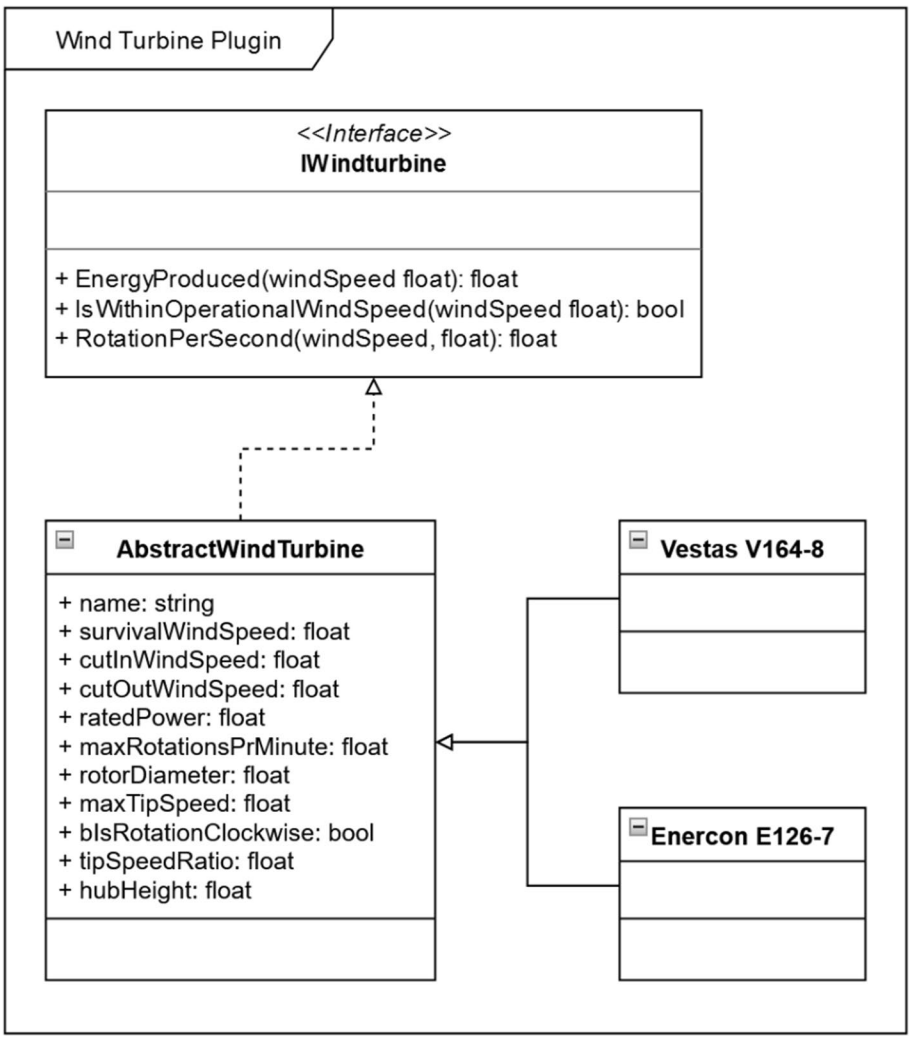

Fig. 11 The wind turbine class hierarchy

The implementations of the wind turbines were designed to allow for fast future additions of new wind turbines to the plugin. The class hierarchy in Fig. 11 shows the wind turbine interface. This interface allows the components in the Planning Application that only require the functionality of the wind turbine to depend on a generalised type, allowing for reusability in the application.

The abstract implementation expands this by having fields for the wind turbine specs. Like the interface, the abstract class allows for reusability when working with the wind turbine plugin by ensuring that each wind turbine implementation contains the required specs to function in the environment. The abstract class also implement the two methods IsWithinOperational-Speed and RotationPerSecond dictated by the interface. This allows the concrete implementations to use or override the implementations. If implementations in the abstract class are used, the concrete applications only need to define the specs via the fields and implement the EnergyProduced method.

The wind turbine hierarchy was implemented in C++. The implementation of IWindTubine- see Listing 1—defined the functions. Unreal Engine generated the interface template inheriting from UInterface. The template was extended with the three functions. Each function is annotated with UFUNCTION to let UE interpret the functions. Furthermore, the tag BlueprintCallable allows blueprints to call the function, while BlueprintNativeEvent allows blueprints to implement the function.

Listing 1. IWindturbine implementation

```
UINTERFACE(MinimalAPI, Blueprintable)
class UWindTurbineInterface : public UInterface
{
    GENERATED_BODY()
};

class TURBINE_2_API IWindTurbineInterface
{
    GENERATED_BODY()

public:

    UFUNCTION(BlueprintCallable, BlueprintNativeEvent)
        float EnergyProduced(float WindSpeed);

    UFUNCTION(BlueprintCallable, BlueprintNativeEvent)
        bool IsWithinOperationalSpeed(float WindSpeed);

    UFUNCTION(BlueprintCallable, BlueprintNativeEvent)
        float RotationPerSecond(float WindSpeed);
};
```

The implementation of the AbstractWindTurbine class was annotated with the tags Abstract and Blueprintable—see Listing 2. Abstract defines that the class can not be initialized during runtime. Blueprintable allows for blueprint implementations of the class. The AbstractWindTurbine implements the IWindturbine interface. Furthermore, the class inherits from the UActorComponent class provided by UE. By inheriting from UActorComponent, the wind turbine implementations can be used as logic components by actors in the environment.

Listing 2. AbstractWindTurbine class declaration

```
UCLASS(Abstract, Blueprintable, ClassGroup=(Custom),
meta=(BlueprintSpawnableComponent) )
class TURBINE_2_API UAbstractWindTurbine : public UActorComponent, public
IWindTurbineInterface
{
…
```

The AbstractWindTurbine class defined the properties of a wind turbine. This was implemented as floating point variables on the class with the UPROPERTY annotation—see Listing 3.

Listing 3. AbstractWindTurbine property

```
...
    UPROPERTY(blueprintreadOnly, EditAnywhere)
        float rotorDiameter;
...
```

In addition to the properties, the abstract class implemented the functions IsWithinOperationalSpeed and RotationPerSecond. The implementation of the functions in the abstract class allowed for the reuse of the implementations for all wind turbines inheriting the abstract class. The functions were implemented firstly in the header file—see Listing 4—and thereafter in the C++ file. The header file defined that the functions had to be implemented by the abstract class. In addition, the EnergyProduced function had to be implemented to throw an error if not extended due to how inheritance is implemented in UE.

Listing 4. AbstractWindTurbine header file

```
UFUNCTION(BlueprintCallable, BlueprintNativeEvent)
bool IsWithinOperationalSpeed(float WindSpeed);
virtual bool IsWithinOperationalSpeed_Implementation(float WindSpeed) override;

UFUNCTION(BlueprintCallable, BlueprintNativeEvent)
float RotationPerSecond(float WindSpeed);
virtual float RotationPerSecond_Implementation(float WindSpeed) override;

UFUNCTION(BlueprintCallable, BlueprintNativeEvent)
float EnergyProduced (float WindSpeed);
virtual float EnergyProduced_Implementation(float WindSpeed) override {
    check(0 && "Must be overridden");
    return 0;
};
```

The C++file contained the implementations for the two functions – see Listing 5. The function IsWithinOperationalSpeed checks if the wind speed is within the operational speed and returns a Boolean based on the check. The function RotationPerSecond calculates the number of rotations of the wind turbine rotor based on the wind speed. First, the tip speed in m/s is calculated based on the wind speed and the tip speed ratio of the wind turbine. The speed is ensured that it can not be greater than the max tip speed. Hereafter the rotor circumference is calculated in meters. The two values are thereafter used to calculate the number of rotations per second.

Listing 5. AbstractWindTurbine function implementations

```
bool UAbstractWindTurbine::IsWithinOperationalSpeed_Implementation(float WindSpeed)
{
    return (cutInWindSpeed <= WindSpeed && cutOutWindSpeed >= WindSpeed) ? true : false;
}

float UAbstractWindTurbine::RotationPerSecond_Implementation(float WindSpeed)
{
    float tipSpeed = tipSpeedRatio * WindSpeed;
    tipSpeed = std::min(tipSpeed, maxTipSpeed);

    float rotorCircumference = rotorDiameter * PI;

    return tipSpeed / rotorCircumference;
                                        }
```

The AbstractWindTurbine is implemented by the classes VestasV164 and EnerconE126. The implementation of a wind turbine shown here is based on the Vestas V164. Listing 6 shows that the properties from the abstract class are initialized with the wind turbine specifications. In addition to the properties, the EnergyProduced function is implemented accordingly to the power curve found for the specific model.

Listing 6. Vestas V164 implementation

```
UVestasV164_8::UVestasV164_8()
{
    survivalWindSpeed = 50.0f;
    cutInWindSpeed = 4.0;
    cutOutWindSpeed = 25.0f;
    ratedPower = 8000.0f;
    maxRotationsPrMinute = 12.1f;
    rotorDiameter = 164.0f;
    maxTipSpeed = 104.0f;
    bIsRotationClockwise = true;
    tipSpeedRatio = 5.0f;
}

float UVestasV164_8::EnergyProduced_Implementation(float windSpeed)
{
    float powerProduction = 0.0f;

    if (windSpeed < cutInWindSpeed || cutOutWindSpeed < windSpeed) {
        powerProduction = 0.0f;
    }
    else if (cutInWindSpeed <= windSpeed && windSpeed < 5) {
        powerProduction = 549.9999999999999 * windSpeed - 2099.9999999999995;
    }
    else if (5 <= windSpeed && windSpeed < 6) {
        powerProduction = 499.99999999999983 * windSpeed + -1849.999999999999;
    }
    else if (6 <= windSpeed && windSpeed < 7) {
        powerProduction = 699.9999999999998 * windSpeed + -3049.999999999998;
    }
    else if (7 <= windSpeed && windSpeed < 8) {
        powerProduction = 1049.9999999999998 * windSpeed + -5499.999999999998;
    }
    else if (8 <= windSpeed && windSpeed < 9) {
        powerProduction = 1249.9999999999998 * windSpeed + -7099.999999999998;
    }
    else if (9 <= windSpeed && windSpeed < 10) {
        powerProduction = 1449.9999999999998 * windSpeed + -8899.999999999998;
    }
    else if (10 <= windSpeed && windSpeed < 11) {
        powerProduction = 1499.999999999999 * windSpeed + -9399.99999999999;
    }
    else if (11 <= windSpeed && windSpeed < 12) {
        powerProduction = 699.9999999999998 * windSpeed + -599.9999999999973;
    }
    else if (12 <= windSpeed && windSpeed < 13) {
        powerProduction = 199.99999999999994 * windSpeed + 5400.000000000001;
    }
    else if (13 <= windSpeed) {
        powerProduction = ratedPower;
    }
    else {
        throw std::invalid_argument("Power production could not be calculated for provided
wind speed.");
    }

    return powerProduction;
}
```

Wind

The built-in wind system in Unreal Engine only affects foliage objects. Therefore the wind turbine was not affected. To solve the problem, the wind was stored in an invisible object in the environment and the wind turbines could receive the current wind speed from this object. Due to the difference in the height of the wind turbines, the wind speed is adjusted to the height of the wind turbine hub using the wind profile power law. Equation 1 shows the rearranged wind profile power law, where u is the wind speed at height z and u_r is the known wind speed at height z_r. Lastly, the exponent α is the coefficient of the stability in the atmosphere (Touma 1977).

$$u = u_r \left(\frac{z}{z_r} \right)^{\alpha} \tag{1}$$

Equation 1. Wind profile power law.

The adjustment of the wind speed was performed by each of the observers implementing the WindSpeedObserver interface. This allows each wind turbine model to adjust the new wind speed to the height of the hub. To allow for code reuse, a blueprint function library was created. The library contained an implementation of the rearranged wind power law—see Fig. 12.

Input interface

The interface for receiving and sending messages was based on MQTT topics and the messages were formatted as JSON strings. Listing 7 is an example of a message sent from the physical twin containing the latest wind speed. The JSON object contains the following:

- *timestamp*—int64—Unix timestamp of when the message is sent.
- *wind_speed*—float—Wind speed in meters per second.
- *altitude*—int32—The altitude of the wind speed reading in meters.

Listing 7. Example of JSON object sent from the physical twin

```
{
    "timestamp": 1657175353000,
    "wind_speed": 5.2,
    "altitude": 10
}
```

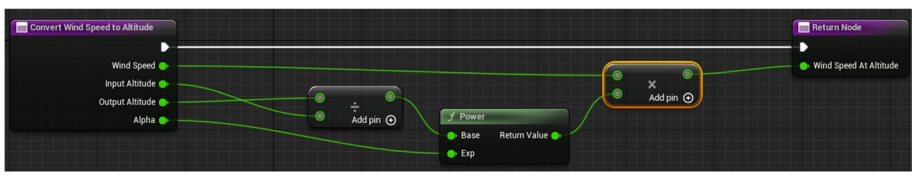

Fig. 12 Implementation of wind profile power law

Prediction

The prediction mechanism in the digital twin was implemented as a series of functions in a blueprint function library able to calculate:

- Number of hours where the wind turbine produces energy
- Max wind speed the wind turbine experiences
- Number of hours at rated energy production
- Hourly production
- The total amount of energy produced for a period

All functions take an object of the abstract wind turbine, a list of hourly wind speeds, and the altitude of the measured wind speeds as input.

The number of hours with production was implemented as seen in Fig. 13. When the function is called the list of wind speed values is converted to the altitude of the wind turbine hub using the wind profile power law. The new wind speeds are iterated, checking if the value is within the cut-in and cut-out values of the wind turbine. If the check is true, an integer variable is incremented. When the list has been iterated, the variable is returned as the number of hours with production.

The function for finding the highest wind speed the wind turbine experiences was implemented as seen in Fig. 14. When the function is called, the wind speeds are converted to match the height of the wind turbine hub. Hereafter, the largest value is found and returned.

The function for calculating the hours where the wind turbine is at its rated production was implemented as seen in Fig. 15. When the function is called, the wind speeds are converted to match the height of the wind turbine hub. Hereafter, the wind speeds are iterated and used to calculate the power output from the wind turbine. If the produced power is equal to the rated power of the wind turbine, an integer variable is incremented. When the array has been iterated, the integer variable is returned.

The function for calculating the hourly production for a wind turbine was implemented as seen in Fig. 16. First, the wind speeds are converted to the height of the wind turbine hub. The wind speeds are thereafter looped and used to calculate the production using the power curve of the wind turbine. The production value is added to an array which is returned when the loop is completed.

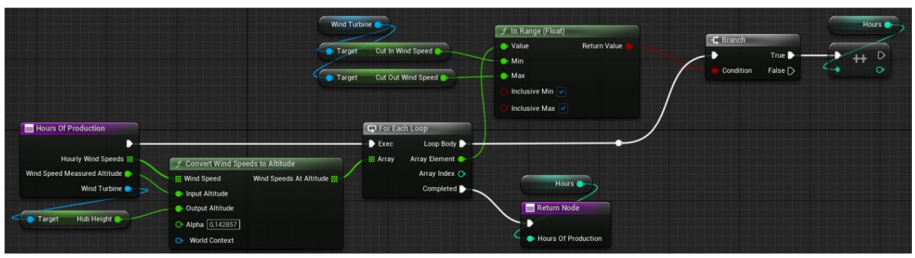

Fig. 13 Calculate the number of hours with production

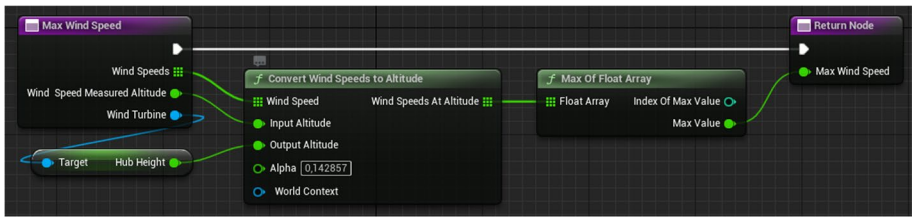

Fig. 14 Max wind speed implementation

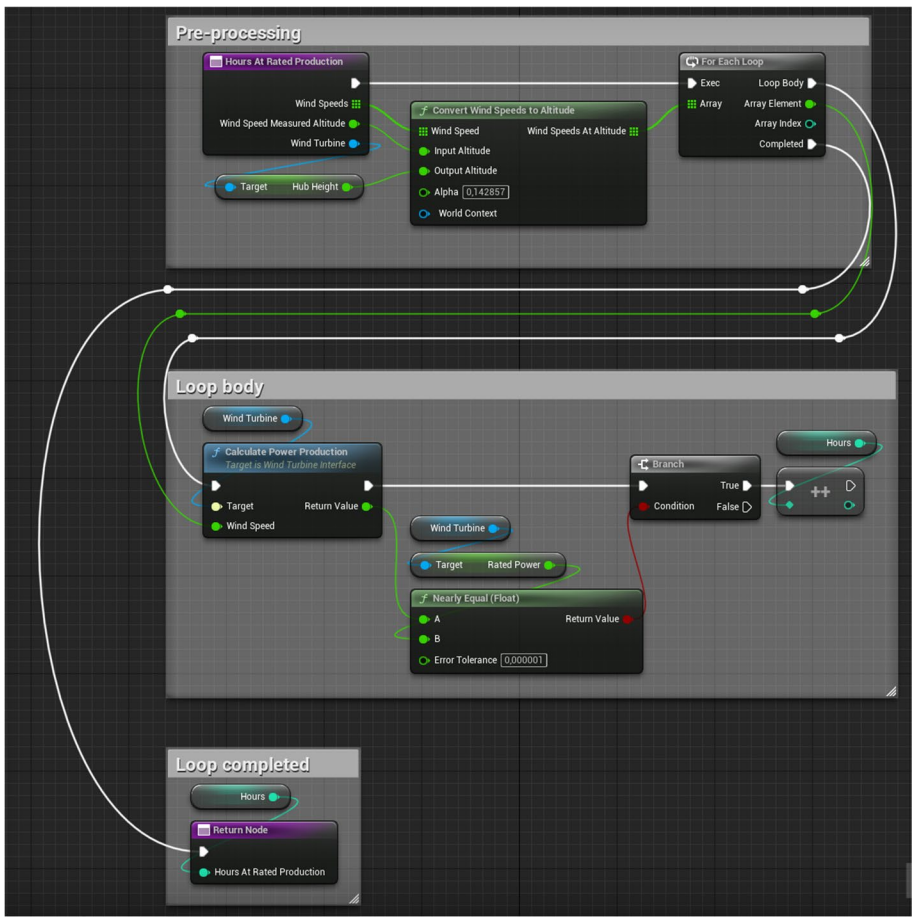

Fig. 15 Hours at rated production implementation

The function for calculating the total production for a wind turbine was implemented as seen in Fig. 17. The function uses the Hourly Production function described above. The sum of the array is thereafter found and returned.

Digital twin scenario design

To verify the designed digital twin of wind power, six scenarios are designed in this study. The five scenarios are tested under various conditions to showcase the response of the digital twin. Furthermore, a scenario with real-world data is designed to validate the developed digital twin of wind power.

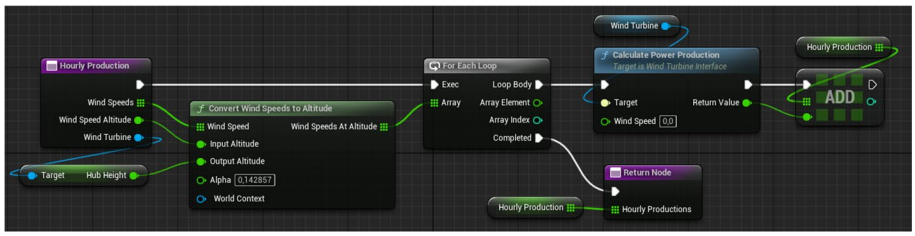

Fig. 16 Hourly production implementation

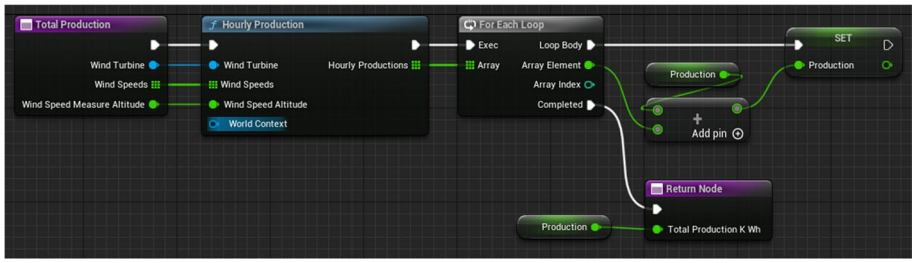

Fig. 17 Total production implementation

Scenario 1—if there is no wind

- Description: If there is no wind, how do the selected brands and selected models of wind turbines react, and how much wind power produced
- Purpose: To examine how the different wind turbines react to an environment containing no wind.
- Expected result: It is expected that all models produce no energy. Furthermore, it is expected that the wind turbine will not rotate.

Scenario 2—if the wind is in the normal range

- Description: If the wind is in the normal range, how do the selected brands and selected models of wind turbines react, and how much wind power produced
- Purpose: To examine how the different wind turbines react to an environment with a wind speed within normal operational wind speeds.
- Expected result: It is expected that the wind speed affects the wind turbines based on the height of the wind turbine hub. Wind turbines with a higher hub are expected to experience a higher wind speed. Furthermore, wind turbines are expected to have a fixed energy production according to their power curves. The wind turbines are expected to be rotating.

Scenario 3—if the wind is out of the normal range

- Description: If the wind is out of the normal range, how do the selected brands and selected models of wind turbines react, and how much wind power produced
- Purpose: To examine how wind turbines react to an environment with a wind speed that is out of normal range.
- Expected result: It is expected that all the models produce no energy. Furthermore, it is expected that the wind turbine will not rotate.

Scenario 4—if the wind is increasing

- Description: If the wind is increasing, whether and how the selected brands and selected models of wind turbines change the reaction, whether and how the wind power production changes
- Purpose: To examine how the wind turbines react to increasing wind speeds.
- Expected result: The wind turbines will begin energy production, and the rotor will rotate when the wind speed meets the cut-in speed of the specific model. The energy production will thereafter increase together with the wind speed. When the wind speed becomes greater than the cut-out wind speed of each wind turbine, the energy production and rotation are expected to stop.

Scenario 5—if the wind is decreasing

- Description: If the wind is decreasing, whether and how the selected brands and selected models of wind turbines change the reaction, whether and how the wind power production changes
- Purpose: To examine how the wind turbines react to decreasing wind speeds.
- Expected result: The wind turbines will begin energy production, and the rotor will rotate when the wind speed decreases to the cut-out speed of the specific model. The energy production will thereafter decrease together with the wind speed. When the wind speed becomes less than the cut-in wind speed of each wind turbine, the energy production and rotation are expected to stop.

Scenario 6—Wind turbine performance with real-world data

- Description: If the wind speed is based on real-world data, how the selected wind turbine models react, and how much electricity is produced by the selected wind turbine models
- Purpose: To examine how the wind turbines perform with the given inputs measured at a physical site.
- Expected result: It is expected that the selected wind turbines will produce electricity according to Eq. 1 and the read-world wind data input. It is expected that the differ-

ent wind turbine models will perform differently based on the hub height, cut-in and -out speeds, and their power curves according to the specification.

Case study

This section presents the case study performed in this study, containing the description of the environment, and the wind turbine models used.

Represented wind environment

The represented wind environment is an open space outside Esbjerg's Danish coastal city. The city of Esbjerg is chosen because the wind speeds in the surrounding area are suitable for wind turbines. Figure 18 shows the wind speeds for the location in October 2020. The data is from the Typical Meteorological Year, with data collected from 2005 until 2020. Note that the atmospheric stability (alpha) is defined as neutral for all scenarios, having a value of 1/7.

Wind turbine models

The wind turbines used for the case study were chosen based on the availability of specifications and power curves. The selected wind turbines are the Vestas V164-8.0 and the Enercon E126-7.58 which provides similar energy outputs. The specification for both wind turbines can be seen in Table 1. The V164-8 made by Vestas is a taller wind turbine with a rated power of 420 watts higher than the E126. However, the E126 has a broader range of operable wind speeds. It should be noted that the survival wind speed for the Enercon E126 could not be found. Therefore, the value is estimated to be 40 m/s for the remainder of the study.

Results

This research conducts six scenarios, the first scenario is for the verification testing and the last is for the validation testing of the developed digital twin of wind power. The verification testing results prove that the developed digital twin of wind power can perform

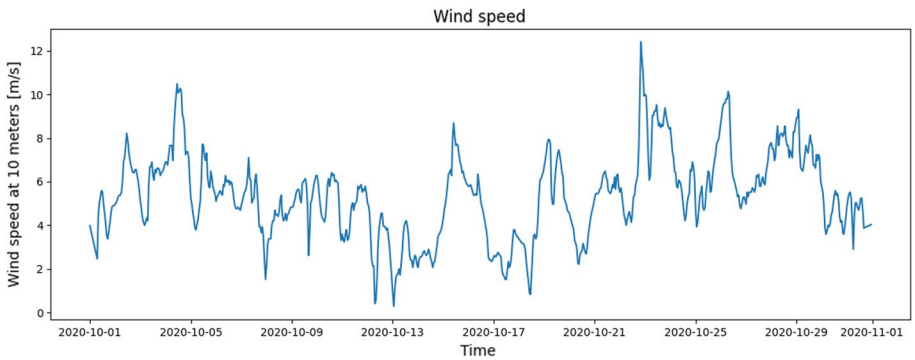

Fig. 18 The wind speeds in October 2020, Esbjerg, Denmark

Table 1 Two wind turbine models used in the case study (wind-turbine-models: Vestas V164-8.0 2022; wind-turbine-models: Enercon 2022)

Model	Unit	Vestas V164-8.0	Enercon E126-7.58
Survival wind speed	[m/s]	50	(~40)
Cut-in wind speed	[m/s]	4	3
Cut-out wind speed	[m/s]	25	34
Rated power	[33]	8000	7580
Max rotations per minute	[rpm]	12.1	12.1
Rotor diameter	[m]	164	127
Max tip speed	[m/s]	104	80
Rotation clockwise	[bool]	True	True
Tip speed ratio	[–]	5	5
Hub height	[m]	150	135

according to the design and the validation testing results show that the developed digital twin can facilitate the investigation of wind turbine planning and deployment.

Scenario 1—if there is no wind

Scenario 1 investigates how the selected brands and selected models of wind turbines react, and how much wind power is produced if there is no wind. The input data for scenario 1 is sent from a mocked version of the physical twin. The input message can be seen in Listing 9. The message contains the wind speed of 0 m per second measured at 10 m. The resulting output from the wind turbines can be seen in Table 2 and Fig. 19. The results show that both turbines have an output of 0 kW, matching the expected result.

Table 2 Scenario 1 wind turbine results

Wind turbine model	Vestas V164-8.0	Enercon E126-7.58
Electricity production	0 kW	0 kW
Experienced wind speed	0 m/s	0 m/s

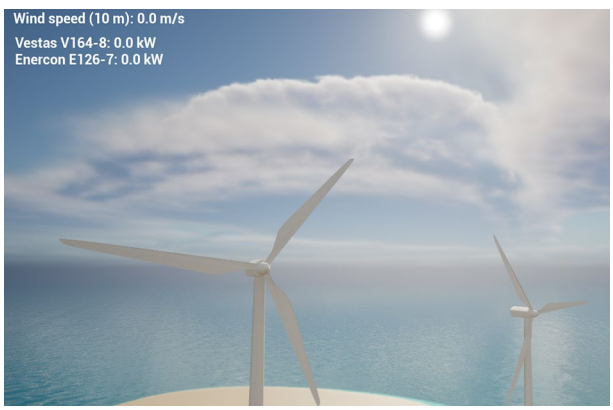

Fig. 19 Screenshot of the output at Scenario 1

Listing 9. Scenario 1 input

```
{
    "timestamp": 1657611557000,
    "wind_speed": 0.0,
    "altitude": 10
}
```

Scenario 2—if the wind is in the normal range

Scenario 2 investigates how the selected brands and selected models of wind turbines react, and how much wind power is produced if the wind is in the normal range. The input data for scenario 2 is sent from a mocked version of the physical twin. The input message can be seen in Listing 10. The message contains the wind speed of 5 m per second measured at 10 m.

The resulting output from the wind turbines can be seen in Table 3 and Fig. 20. The results show that both turbines are producing energy. The Vestas wind turbine experiences a higher wind speed (7.3617 m/s) and produces more electricity (2.229.87 kW) compared to the Enercon wind turbine (which experiences a wind speed of 7.2518 m/s and produces 1413.67 kW electricity) due to its hub being at a higher altitude.

Table 3 Scenario 2 wind turbine results

Wind turbine model	Vestas V164-8.0	Enercon E126-7.58
Electricity production	2229.87 kW	1413.67 kW
Wind speed at hub	7.3617 m/s	7.2518 m/s

Fig. 20 Screenshot of the output at Scenario 2

Listing 10. Scenario 2 input

```
{
    "timestamp": 1657611557000,
    "wind_speed": 5.0,
    "altitude": 10
}
```

Scenario 3—if the wind is out of the normal range

Scenario 3 investigates how the selected brands and selected models of wind turbines react, and how much wind power is produced if the wind is out of the normal range. The input data for scenario 2 is sent from a mocked version of the physical twin. The input message can be seen in Listing 11. The message contains the wind speed of 5 m per second measured at 10 m. The resulting output from the wind turbines can be seen in Table 4 and Fig. 21. The results show that both turbines are not producing energy due to the wind speed not being operational

Table 4 Scenario 3 wind turbine results

Wind turbine model	Vestas V164-8.0	Enercon E126-7.58
Electricity production	0.0 kW	0.0 kW
Wind speed at hub	2.9447 m/s	2.9007 m/s

Fig. 21 Screenshot of the output at Scenario 3

Listing 11. Scenario 3 input

```
{
    "timestamp": 1657611557000,
    "wind_speed": 2.0,
    "altitude": 10
}
```

Scenario 4—if the wind is increasing

Scenario 4 investigates whether and how the selected brands and selected models of wind turbines change the reaction, and whether and how the wind power production changes if the wind is increasing. The input for scenario 4 was a series of incremental changes to the wind speed at 10 m from 0 to 30 m/s.

The results of Scenario 4 can be seen in Fig. 22. The figure shows that both wind turbines start their energy production when the wind speed matches their cut-in wind speed. Likewise, the production stops when their cut-out speeds are met.

Scenario 5—if the wind is decreasing

Scenario 5 investigates whether and how the selected brands and selected models of wind turbines change the reaction, and whether and how the wind power production changes if the wind is decreasing. The input for scenario 5 was a series of changes to the wind speed at 10 m from 30 to 0 m/s.

The results from scenario 5 can be seen in Fig. 23. The figure shows that both wind turbines start their energy production when the wind speed is below their cut-out wind speed. The results show that the production from the wind turbines decreases as the wind speed decrease.

Scenario 6—the performance overview of the wind turbines

The electricity production by the two wind turbine models (Vestas V164 and Enercon E126). During the whole testing period (from 1 to 31st October 2020) is shown in Fig. 24 and summarized in Table 5.

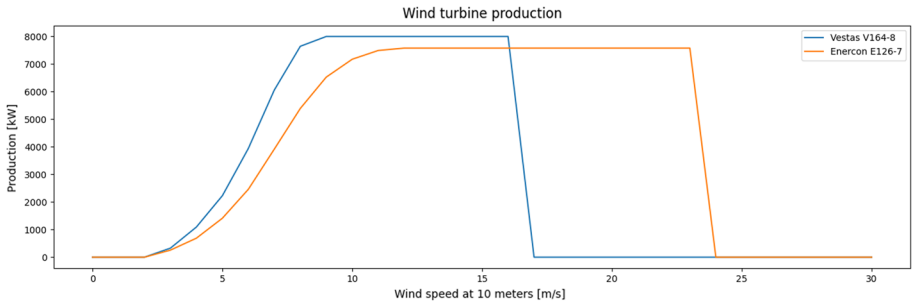

Fig. 22 Scenario 4 results

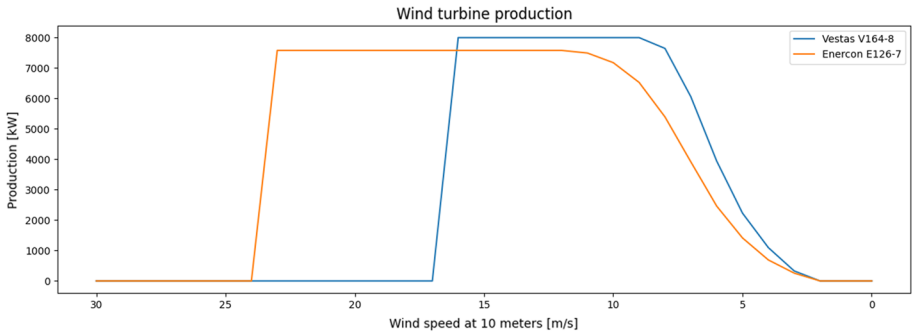

Fig. 23 Scenario 5 results

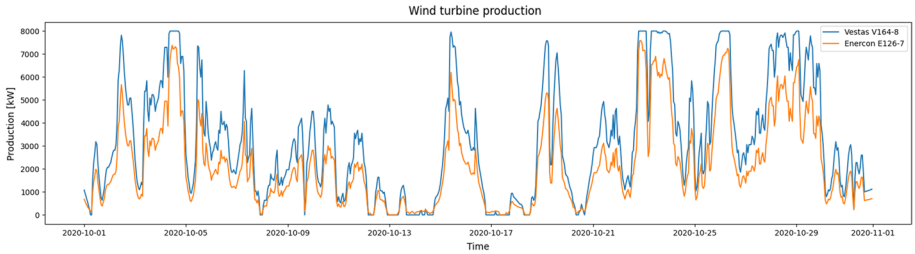

Fig. 24 The performance overview of the wind turbines in October 2020, Esbjerg, Denmark

Table 5 Summary of the performance overview of the wind turbines in October 2020, Esbjerg, Denmark

Wind turbine model	Vestas V164	Enercon E126
Total electricity production	2,410,838.142 kWh	1,641,128.283 kWh
Hours for production	89.7%	96.8%
Hours at rated production	5.0%	0.3%
The max wind speed experienced at hub height	18.272 m/s	17.999 m/s

The results show that, although Enercon E126 has produced more hours (96.8% of the time) in the period compared to Vestas V164 (89.7% of the time), Vestas V164 has produced more electricity (around 2.4 GWh) than Enercon E126 (around 1.64 GWh). Furthermore, the results for hours at rated production show that the hours that Vestas V164 (around 5.0%) has its peak production is more than Enercon E126 (around 0.3%).

Moreover, the results of the max wind speed at hub height that the two wind turbines were exposed to do not exceed either of their survival speeds. The Vestas V164 was exposed to a maximum wind speed of 18.272 m/s, while the Enercon E126 experienced a maximum wind speed of 17.999 m/s. Based on the results in scenario 6, it can be concluded that the wind turbine of Vestas V164 has a better overall performance than Enercon E126.

Discussion

The developed digital twin of wind power in this study is created with two purposes: monitoring the environment and predicting energy production from placed wind turbines either in real-time or for a given period. The monitoring provided by the digital twin for wind power enabled easy assessment of the physical environment via the user interface of the digital twin, allowing the user to place wind turbines in the environment and assess the visual impact and the potential production of the wind turbine. The prediction interface allowed the user to estimate total production from a wind turbine and test if the conditions in the area matched its specification. The results show that the developed digital twin for wind power can perform the assigned purposes.

Unreal Engine 5 is selected in this study for the digital twin development due to its promised functions of high-quality physical environment imitation and world-building capabilities and tools for realistic lighting, water, physics, etc. The results prove that Unreal Engine 5 is useable for developing entire digital twins, e.g., wind power.

The user interface development in the Unreal Engine for the digital twin development is beneficial, due to built-in world-building, lighting, atmosphere, and water tools. In addition, the vast amounts of content available via 3D model brokers and the Unreal Marketplace allowed for world-building without having to build these artefacts.

The Unreal Engine provides the options of Blueprint Scripting and C++ for the logic implementation. This allows developers to implement the logic closely to the visual logic and components. It should be noted that Blueprint Scripting is highly capable for the development of digital twins, but can be tedious and cumbersome in branching logic.

The benefit of developing a digital twin in the Unreal Engine is the plugin system, allowing for the distribution and integration of logic, assets, and tools. These plugins are reusable and speed up the development of future similar projects. With game engines becoming a more popular tool, it is estimated that more plugins will appear in the future to support more complex projects for digital twins.

Conclusion

This study examines the benefits of utilizing game engines for digital twin development. A digital twin prototype for wind power in Unreal Engine 5 is developed to investigate wind power planning and deployment potentials. A wind environment outside the city of Esbjerg, Denmark, is chosen to showcase the visual impact of the wind turbine, estimate the electricity production based on the wind conditions in the area, and assist in the selection of the wind turbine best fitting to the area.

The results show that the visualization function of the Unreal Engine can showcase various wind turbines in the environment and provide estimated energy production of the wind turbines based on the wind in the area. Furthermore, the realistic visualization of the area using the built-in weather systems in Unreal Engine 5 allows a third party, e.g., a wind turbine producer, to place specific models of wind turbines to assess the utilization of an area, optimize their position, estimate production, and assess whether the wind turbines disturb the surroundings in the developed digital twin.

Moreover, digital twins built in Unreal Engine 5 have great potential for communicating and showcasing different scenarios of wind turbines in an area with its visual

capabilities. The digital twin also allows for further extension in the future to include monitoring of wind turbine energy production, connection to the electricity grid, and more. Furthermore, the developed digital twin can facilitate users to investigate and plan wind power in chosen areas by estimating the electricity production and analysing the utilisation using historical data.

Overall, the use of a game engine can benefit the digital twin development by providing tools for environment modelling with a high degree of realistic visuals. However, it can be complicated to integrate models and third-party code into unreal applications, if not already existing as an Unreal plugin. This can, however, be mitigated by utilizing Unreal Engine as the user interface for distributed digital twins.

This study mainly focuses on investigating the potential of developing a digital twin of wind power in the Unreal Engine. Two types of wind turbines (Vestas V164-8 and Enercon E-126 7.580) and one location (Esbjerg, Denmark) are chosen for the verification and validation testing due to the data availability. Therefore, future work should consider investigating more types of advanced wind turbine models to more accurately predict their behaviour in the environment. Furthermore, this study mainly focuses on the modelling of the wind turbines and environment. The environment might be different between on-shore and off-shore wind energy. Therefore, an invesigtation of the off-shore environment is recommended in the future work.

Abbreviations

DTI	Digital twin instance
DTP	Digital twin prototype
KW	Kilowatt
GW	Gigawatt
MQTT	Message Queue Telemetry Transport
JSON	JavaScript object notation
IoT	Internet of things

Acknowledgements
Not applicable.

About this supplement
This article has been published as part of *Energy Informatics Volume 5 Supplement 4, 2022: Proceedings of the Energy Informatics. Academy Conference 2022 (EI.A 2022)*. The full contents of the supplement are available online at https://energyinformatics.springeropen.com/articles/supplements/volume-5-supplement-4.

Author contributions
JVS was the main contributor to the first draft writing of the manuscript. ZM was the main contributor to editing and finalising the manuscript. BNJ contribute to the discussion, comments, and inputs. All authors read and approved the final manuscript.

Funding
This paper is part of the ANNEX 81 project (Project IEA EBC ANNEX 81 Data-Driven Smart Buildings, funded by EUDP Denmark, Case no.64019-0539) by the Danish funding agency (the Danish Energy Technology Development and Demonstration (EUPD) program, Denmark) and part of the ClusterSoutH2 project (project title ClusterSoutH2- Designing a PtX Ecosystem in Southern Denmark) by the European Regional Development Fund.

Availability of data and materials
Not applicable.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Accepted: 11 October 2022
Published: 21 December 2022

References

Abdallah A, Primas M, Turcin I, Traussnigg U (2020) The Potential of Game Development Platforms for Digital Twins and Virtual Labs: Case Study of an Energy Analytics and Solution Lab. CAMPUS 02, University of Applied Sciences, Graz, 8010, Austria, annote = Export Date 3 January 2022 Methods & Technologies Unity3D Purpose Integration of DT and lab environment Application domain? Contributions? Other abstract to vague to be further analysed. p. 117–21

Andryushkevich SK, Kovalyov SP, Nefedov E (2019) Composition and application of power system digital twins based on ontological modeling. 2019 IEEE 17th International Conference on Industrial Informatics (INDIN). p. 1536–42

Bradac Z, Marcon P, Zezulka F, Arm J, Benesl T (2019) Digital Twin and AAS in the Industry 4.0 Framework. IOP Conference Series: Materials Science and Engineering

Chang-won L: Hyundai Motor works with Unity to build digital-twin of factory supported by metaverse platform. https://www.ajudaily.com/view/20220107083928529 (2022). Accessed 13/07 2022

Christensen K, Ma Z, Værbak M, Demazeau Y, Jørgensen BN (2019) Agent-based decision making for adoption of smart energy solutions. 2019 IEEE Sciences and Humanities International Research Conference (SHIRCON). p. 1–4

Commission E: photovoltaic geographical information system. https://re.jrc.ec.europa.eu/pvg_tools/en/tools.html Accessed 05/07 2022

Engine U: Digital twins—For architecture, real estate, and the built environment. https://www.unrealengine.com/en-US/digital-twins Accessed 05/07 2022

Eyre JM, Dodd TJ, Freeman C, Lanyon-Hogg R, Lockwood AJ, Scott RW, et al (2018) Demonstration of an Industrial Framework for an Implementation of a Process Digital Twin. ASMEInternational Mechanical Engineering Congress and Exposition (IMECE2018): Amer Soc Mechanical Engineers

Fahim M, Sharma V, Cao TV, Canberk B, Duong TQ (2022) Machine learning-based digital twin for predictive modeling in wind turbines. IEEE Access 10:14184–14194. https://doi.org/10.1109/ACCESS.2022.3147602

Gamma E, Helm R, Johnson R, Vlissides J (1994) Design patterns: elements of reusable object-oriented software

Glaessgen EH, Stargel DS (2012) The digital twin paradigm for future NASA and U.S. Air force vehicles. Collection of Technical Papers—AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference. https://doi.org/10.2514/6.2012-1818

Gppert A, Grahn L, Rachner J, Grunert D, Hort S, Schmitt RH (2021) Pipeline for ontology-based modeling and automated deployment of digital twins for planning and control of manufacturing systems. J Intell Manuf. https://doi.org/10.1007/s10845-021-01860-6

Grieves M, Vickers J (2017) Digital twin: mitigating unpredictable, undesirable emergent behavior in complex systems. In: Kahlen F-J, Flumerfelt S, Alves A (eds) Transdisciplinary perspectives on complex systems: new findings and approaches. Springer International Publishing, Cham, pp 85–113

Howard DA, Ma Z, Jørgensen BN (2021) Digital twin framework for energy efficient greenhouse industry 4.0. Springer International Publishing, Cham, pp 293–297

Howard DA, Ma Z, Veje C, Clausen A, Aaslyng JM, Jørgensen BN (2021) Greenhouse industry 4.0—digital twin technology for commercial greenhouses. Energy Inf 4(2):37. https://doi.org/10.1186/s42162-021-00161-9

Introducing JSON. www.json.org Accessed 05/07 2022

Kong LCW, Harper S, Mitchell D, Blanche J, Lim T, Flynn D (2020) Interactive digital twins framework for asset management through internet. 2020 IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT). p. 1–7

Larsen BL (2022) EU-Parlamentet vedtager strategi for havvind. https://www.danskenergi.dk/nyheder/eu-parlamentet-vedtager-strategi-havvind (2022). Accessed 02-07-2022

Lee D, Lee SH, Masoud N, Krishnan MS, Li VC (2021) Integrated digital twin and blockchain framework to support accountable information sharing in construction projects. Autom Constr 127:9. https://doi.org/10.1016/j.autcon.2021.103688

Leskovsk (2020) Proposal of digital twin platform based on 3D rendering and IIoT principles using virtual augmented reality. 2020 Cybernetics & Informatics (K&I). p. 1–8

Ma Z, Billanes JD, Jørgensen BN (2017) A business ecosystem driven market analysis: the bright green building market potential. 2017 IEEE Technology & Engineering Management Conference (TEMSCON). p. 79–85

Ma Z, Værbak M, Rasmussen RK, Jøergensen BN (2019a) Distributed energy resource adoption for campus microgrid. 2019a IEEE 17th International Conference on Industrial Informatics (INDIN). p. 1065–70

Ma Z, Broe M, Fischer A, Sφrensen TB, Frederiksen MV, Jφergensen BN (2019b) Ecosystem thinking: creating microgrid solutions for reliable power supply in India's power system. 2019b 1st Global Power, Energy and Communication Conference (GPECOM). p. 392–7

Ma Z, Korsgaard J, Jørgensen BN (2020) Optimization of greenhouse production process: an investigation of energy efficiency potentials. 2019 6th International Conference on Dependable Systems and Their Applications (DSA). p. 365–70

Malakuti S, Juhlin P, Doppelhamer J, Schmitt J, Goldschmidt T, Ciepal A (2021) An architecture and information metamodel for back-end data access via digital twins. 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). p. 1–8

Moghadam FK, Nejad AR (2022) Online condition monitoring of floating wind turbines drivetrain by means of digital twin. Mech Syst Signal Process 162:108087. https://doi.org/10.1016/j.ymssp.2021.108087

Moghadam FK, Rebouças GFdS, Nejad AR (2021) Digital twin modeling for predictive maintenance of gearboxes in floating offshore wind turbine drivetrains. Forsch Ingenieurwes 85(2):273–286. https://doi.org/10.1007/s10010-021-00468-9
Mostafa F, Tao LQ, Yu WJ (2021) An effective architecture of digital twin system to support human decision making and AI-driven autonomy. Concurr Comput-Pract Exp 33(19):15. https://doi.org/10.1002/cpe.6111
Negri E, Berardi S, Fumagalli L, Macchi M (2020) MES-integrated digital twin frameworks. J Manuf Syst 56:58–71. https://doi.org/10.1016/j.jmsy.2020.05.007
Negrin DAM, Cleophas L, Brand MVD (2021) Using Ptolemy II as a Framework for Virtual Entity Integration and Orchestration in Digital Twins. 2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C). p. 233–6
Örs E, Schmidt R, Mighani M, Shalaby M (2020) A conceptual framework for AI-based operational digital twin in chemical process engineering. 2020 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC). p. 1–8
Ospina-Bohórquez A, López-Rebollo J, Muñoz-Sánchez P, González-Aguilera D (2022) A digital twin for monitoring the construction of a wind farm. Eng Proc. https://doi.org/10.3390/engproc2022017003
Pereira JG, Ellman A (2020) From CAD to physics-based digital twin: framework for real-time simulation of virtual prototypes. Proceedings of the Design Society: DESIGN Conference. p. 335–44
Perforce: What Is Keeping Automotive Software Developers Up at Night? https://www.perforce.com/resources/sca/2022-state-automotive-software-development-report#what-is-keeping-automotive-software-developers-up-at-night Accessed 12/07 2022
Redeker M, Weskamp JN, Rssl B, Pethig F (2021) Towards a digital twin platform for Industrie 4.0. 2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS). p. 39–46
Rolle RP, Martucci VO, Godoy EP (2021) Modular framework for digital twins: development and performance analysis. J Control Autom Elect Syst 32(6):1485–1497. https://doi.org/10.1007/s40313-021-00830-w
Shahsavari S, Immonen E, Rabah M, Haghbayan MH, Plosila J (2021) MCX—an open-source framework for digital twins. Proceedings—European Council for Modelling and Simulation, ECMS. p. 119–24
Sharotry A, Jimenez JA, Wierschem D, Mediavilla FAM, Koldenhoven RM, Valles D, et al (2020) A digital twin framework for real-time analysis and feedback of repetitive work in the manual material handling industry. Winter Simulation Conference: IEEE; p. 2637–48
Steindl G, Stagl M, Kasper L, Kastner W, Hofmann R (2020) Generic digital twin architecture for industrial energy systems. Appl Sci-Basel 10:20. https://doi.org/10.3390/app10248903
Studios N: MQTT Utilities plugin for Unreal Engine. https://github.com/NinevaStudios/mqtt-utilities-unreal (2022). Accessed 15/07 2022
Talkhestani BA, Jung T, Lindemann B, Sahlab N, Jazdi N, Schloegl W et al (2019) An architecture of an intelligent digital twin in a cyber-physical production system. At-Automatisierungstechnik 67(9):762–782. https://doi.org/10.1515/auto-2019-0039
Touma JS (1977) Dependence of the wind profile power law on stability for various locations. J Air Pollut Control Assoc 27(9):863–866. https://doi.org/10.1080/00022470.1977.10470503
Traore MK (2021) Unifying Digital Twin Framework: Simulation-Based Proof-of-Concept. In: Control M, Logist IFACTCIa, editors. 17th IFAC Symposium on Information Control Problems in Manufacturing (INCOM): Elsevier; p. 886–93
Unity: Digital Twin. https://unity.com/solutions/digital-twin Accessed 05/07 2022
Værbak M, Ma Z, Christensen K, Demazeau Y, Jørgensen BN (2019) Agent-based modelling of demand-side flexibility adoption in reservoir pumping. 2019 IEEE Sciences and Humanities International Research Conference (SHIRCON). p. 1–4
wind-turbine-models: Enercon E-126 7.580. https://en.wind-turbine-models.com/turbines/14-enercon-e-126-7.580 Accessed 12/07 2022
wind-turbine-models: Vestas V164-8.0. https://en.wind-turbine-models.com/turbines/318-vestas-v164-8.0 Accessed 12/07 2022
Worden K, Cross EJ, Barthorpe RJ, Wagg DJ, Gardner P (2020) On digital twins, mirrors, and virtualizations: frameworks for model verification and validation. ASCE-ASME J Risk Uncert Eng Syst Part B Mech Eng. https://doi.org/10.1115/1.4046740
51World: 51World. https://www.51vr.com.au/ Accessed 12/707 2022
Yun S, Park JH, Kim WT (2017) IEEE. Data-centric Middleware based Digital Twin Platform for Dependable Cyber-Physical Systems. 9th International Conference on Ubiquitous and Future Networks (ICUFN): Ieee; p. 922–6
Zhang H, Xi Y, Wang S, Wang K, Liu Q, Wei L, et al (2021) Hybrid data-physics based digital twin modeling framework for the power system of bobsleigh and tobogganing venue for Beijing winter Olympics. 2021 6th International Conference on Power and Renewable Energy (ICPRE). p. 865–71
Zohdi TI (2021) A digital-twin and machine-learning framework for the design of multiobjective agrophotovoltaic solar farms. Comput Mech 68:357–370. https://doi.org/10.1007/s00466-021-02035-z

Publisher's Note