SIMULWIND: maintenance **SIMUL**ator for the sustainability of European **WIND** farms

Project nº 2017-1-DE02-KA202-004261

Technical Manual

By using the SIMULWIND simulator you accept the license terms and terms of use

The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

MOHSTOH

CREATIVOS

DT-00M REV 0

SIMULWIND MANUAL

QUICK START

Step 1: Download SIMULWIND.

Step 2: Unzip SIMULWIND (.ZIP file)

Step 3: Go to ANNEX 1

SILMULWIND: PROGRAMS AND MORE

1. PROGRAMS AND VERSIONS

a) UNITY VERSION: 2018.3.7.1F.

b) BLENDER VERSION: 2.8.

On the one hand, we have Unity Version. Is has great importance since its work is related to the compatibility between both previous and later versions, which is not usually a good one. The plugins and libraries used will be included in the project master itself without the need of incorporating anything new.

On the other hand, there is the blender 2.8 version. It is the one used due to his option of being able to prepare the embedded textures of the models. Previous versions did not allow you to easily include this option.

2. AEROGENERATOR MODEL IN BLENDER.

In the blender file, the complete wind turbine that will serve as the basis to incorporate new elements in their correct position is attached. (See export explanation in Blender so that the position and textures of the object are appropriate).

Important: NOT THE ENTIRE MODEL HAS TO BE EXPORTED, just what is new; the OBJECT that you want to include, however having beforehand the reference of the model for the scale and position.

3. PREPARATION OF NEW 3D OBJECTS (BLENDER).

We could differentiate two kind of objects basically:

· Those who are on stage or in the wind turbine. For these, the model of the wind turbine in blender mentioned above must be taken into account. This is duet to reasons related to scale and position. From the administrator mode the user will be able to change the scale, rotation and size of objects but it is always better to have everything prepared beforehand, to avoid a waste of time in the edition or preparation of the practice.

· Objects such as Tools, Epis, Materials: In these, it is not necessary to take the scale into account in the model since the program allows scaling. But it is always advisable to work thinking about your real size and it is the reason why it comes real-sized from the beginning. The centre of the object should be placed where we want it to simulate its position in the hand when holding it. Anyway, our program has an option to modify the position and to place the object correctly.

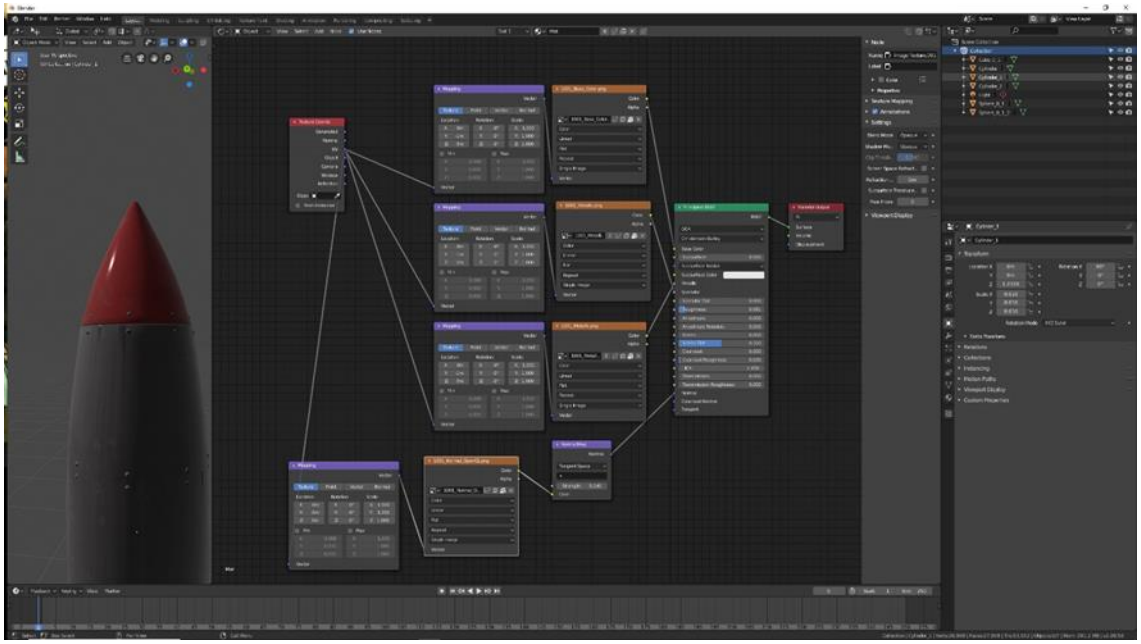
The rest of considerations is common to these two types of objects:

Supported formats: FBX or GLB. An example blender and FBX model will be delivered so that the embedding of textures and export parameters can be seen, thus ensuring that the scale and position are the appropriates.

A) Single Low Poly Mesh. Reaching the lowest number of polygons per object will be the goal for everything to work as it should. If an object is tried to put on 1Millon of polygons, it would suspend the program or make it go slower. What would actually be the greatest would be not to overcome new objects that have more than 10-15 million polygons each, to achieve maximum optimization. Since the wind turbine has many parts, the scenario at the end will end up having so many polygons.

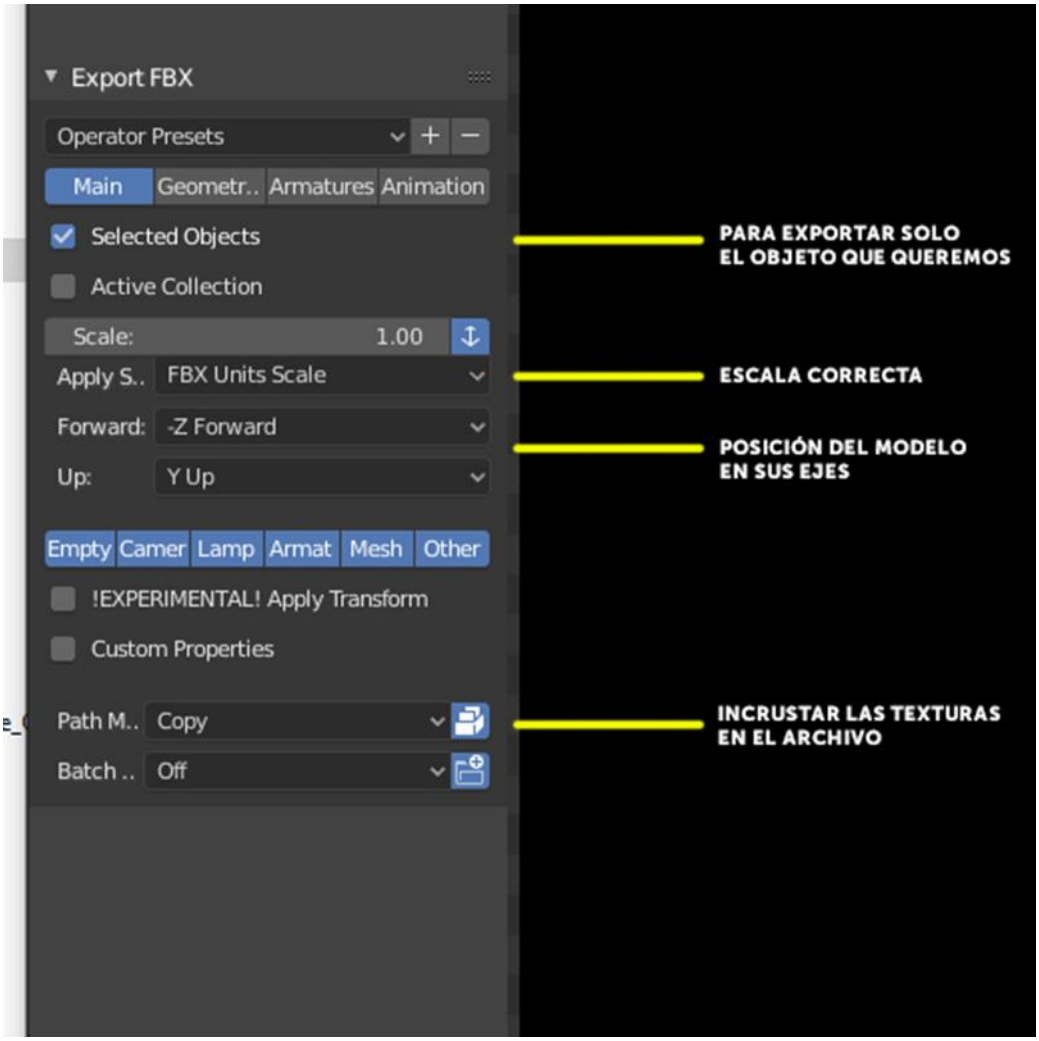
The objects must be in a single mesh without being nested in each other. This is thought this way to ensure that the area of interaction of the objects is correct.

B) Embedded textures: To achieve an optimal effect, textures should be textured in PBR so that the object's display is not flat. Each object should follow the figure shown below for composition of the material and then the embedded textures come:



Textures quality: The recommended quality is the one that does not exceed 2048 pixels per texture, but if possible, it is also a good option to include 512 pixel textures so that then the external load can be faster in real time when using the application.

C) Export and parameters. It is recommended to use fbx although other formats such as GLB are also supported. (They are the only two formats that integrate the textura into the object).



4. MORE:

The Simulwind application has a high diversity of controls and options to then place the objects and edit them as to place or use them inside the wind turbine.

For example, it is possible to:

Say that the object is static or not. If it is static it means that you do not interact with it but you can both see and collide it. Otherwise, if it is dynamic you, can interact with it in each practice. In every single exercise we will have to tell you what kind of interaction we are interested in.

Leave invisible objects that have a collision area. This means that you can use objects such as the shoe that is physically in its correct position, but cannot be seen until you interact with it. It could also serve for other uses such as if you want to do a practice inside the blade to make an invisible surface so that the player in the 2D version does not fall into it. This property can also be defined within each exercise depending on the case we are interested in.

SIMULWIND TEMPLATE SPECIFICATION

3D MODELING AND ITS LOW POLY ADAPTATION FOR VR

3D modelling definition

3D modelling is a process that consists on developing a mathematical representation of any three-dimensional object (it doesn't really matter if it is either inanimate or not) through a specialized software. The name of the product that comes out of this job is "3D model". It can be visualized as a two-dimensional image through a process known as "3D rendering", as well as being used in a computer simulation of physical phenomena. The model can also be physically created using 3D printing devices.

Models can be created either automatically or manually. The manual process consisting on preparing geometric information for 3D graphics could easily be compared with the one related on plastic arts and sculpture.

3D modelling software is a kind of 3D graphics software that is used to produce three-dimensional models. Individual programs of this kind are usually called "Modelling applications" or "modelers." In our case, we will refer to it as the "Blender".

Modelling

The 3D models represent a three-dimensional object using a series of points in space within a 3D gap, connected by various geometric entities such as triangles, lines, curved surfaces, etc. Thus, by being a series of data (points and other information), 3D models can be handmade as well as made through algorithms or even scanned.

3D models are used in 3D graphics. In fact, its use of previous data is also extended to the 3D graphics field on computers. Some video games use pre-rendered images of 3D models as sprites even before a computer could render them in real time.

Nowadays, 3D models are used in a wide variety of fields. The medical industry uses detailed organ models; which can be created by multiple parts of 2-D images from an MRI or CT scanner. The film industry uses this as characters and objects both for

animation or motion pictures. Game industry uses it as a resource for video games. The scientific sector uses them to act like a highly detailed models of chemical components. The architecture industry uses them to show the proposals of new buildings and panoramas through Software Architectural Models. The engineering community uses it in the design of new artefacts, vehicles and structures as well as carrier of many other uses. In the last decades, the earth science community has begun to construct 3D geological models, becoming this a standard practice. 3D models can also be the basis for physical devices that are built with 3D printers or CNC machines.

Representation

The majority of 3D models could be divided into two categories:

Solids - These models are aimed to define the volume of the object that they are representing (like a rock). These are more realistic, but also harder to build. Solid models are usually used for non-visual simulations such as medical and engineering ones.

Housing / contour - These models represent the surface, ex. the outline of the object, not its volume (like an infinite shell). Working with this kind of representation is usually easier than with solid models.

Almost all visual models which are used in games and movies are protective models.

The appearance or semblance of an object depends in great measure on the exterior of the object, boundary representations commonly in computer graphics. The two dimensions surfaces use to be a good analogy of the objects used in graphics, although quite often these are non-manifold. Since surfaces are not finite, we require a discrete digital approximation: polygonal meshes (and to a lesser extent subdivision surfaces) are by far the most common representation, even though point-based representations have been gaining some popularity in the last few years. Level sets become a useful representation to deform surfaces where to undergo various typological changes such as fluids.

This case model is the one which we will use for Low Poly and VR

The transformational process of the representation of objects, such as the coordinated midpoint of the sphere and a point on its circumference in a polygon in the representation of a sphere, is known as tessellation. This step is used in a rendering polygon base, where each and every object is divided into abstract ("primitive") presentations such as spheres, cones etc. They are called "meshes", which are networks of interconnected triangles.

Modeling Process

There are 3 popular ways to represent a model: Curve Modeling, Digital Sculpture and Polygonal Modeling. For our case of Low Poly we will use Polygonal Modeling.

Polygonal Modeling

They are points in a 3D space, called vertices, are connected to form a polygonal mesh. The vast majority of 3D models today are built as polygonal texture models, because they are flexible and because computers can render them very fast. However, the polygons are flat and can only be approximated to curved surfaces using several polygons.

Low Poly Explanation

Low poly is a mesh of polygons in 3D computer graphics that has a relatively small number of polygons. Low poly meshes are produced in real-time applications (for example, games) in contrast to high polyethylene meshes in animated films and special effects from the same era. The term low poly is used both technically and descriptively; The number of polygons in a mesh is an important factor to optimize performance, but it can give the resulting graphics an undesirable appearance.

Polygonal meshes are one of the main methods for modeling a 3D object for a computer to display. Polygons can, in theory, have any number of sides, but they are generally divided into triangles for viewing. In general, the more triangles in a mesh, the more detailed the object is, but the more computationally it requires to display. To

reduce rendering times (that is, increase frame rate), the number of triangles in the scene should be reduced, using low polyethylene meshes.

Computer graphics techniques, such as normal and relief mapping, are designed to compensate for the decrease in polygons by making a low poly object appear to contain more detail than it does. This is done by altering the shading of the polygons so that they contain internal details that are not in the mesh.

Polygon Budget

A combination made out of the game engine or the rendering method and the computer which is being used defines the polygon's budget; the number of polygons that can appear in a scene and still be rendered at an acceptable frame rate. As a consequence, the use of low-density polyethylene meshes is mainly limited to computer games and other software where the user must manipulate 3D objects in live, since the processing capacity is limited to that of a personal computer or characteristic game console and frame rate should be high. Computer-generated images, for example, for movies or fixed images have a higher polygon budget because rendering does not need to be done at the time, which is what requires higher frame rates.

In addition, the processing capacity that computers use to have in these situations is usually less limited, since a large computer network or what is known as a “processing server farm” is often used. Each frame may take hours to create, despite the huge power of the computer. A common example of the difference this makes is for instance the video sequences in motion in computer games that, as they can be pre-rendered, they also look much smoother than the games themselves, that are not in the mesh.

Polygon budget for our application.

For the specific case of our application, being a job for Virtual Reality, it will have a low limit on the number of models’ polygons. It will be necessary to make a distinction between static and dynamic models.

We will define as static models those objects that have 3d representation but do not require user interaction. By not having to animate in real time, more polygons could be added to these meshes. As a recommendation, the objects in this section should not exceed 150,000 polygons.

We will define as dynamic models those objects that have 3d representation could be a part of interaction with the user in real time. For these ones we will fix the low polygon limit between 25,000 and 50,000 polygons.

Furthermore, we will have a limit of polygons per scene in Occusion mode. In other words, we will fix a general limit of how many polygons the application will have to render in a frame. This does not mean that these are all the polygons that are in a 3d scene but, in fact, they will be all those that the Virtual Reality application will have to render by frame. In an ideal assumption, the total limit of polygons rendered by the application should be between 300,000 and 500,000.

Low poly as a relative term

There is no defined threshold for a mesh to be considered as a low poly; Low poly is always a relative term and depends on (among other factors):

The moment the meshes were designed and for which hardware where they aimed.

The detail required in the final mesh.

The shape and properties of the object in question.

As the computing power inevitably increases, the number of polygons that can be used also increases. For example, Super Mario 64 would be considered as a low poly today, but it was considered an impressive achievement when it was launched in 1996. The same way, in 2018, using hundreds of polygons on a sheet at the bottom of a scene would be considered high poly, whereas the use of extra polygons in the main character would be considered low poly. In this way, there is a relativism between the importance of objects and their graphic quality. The most important objects, such as non-player characters, usually contain a greater amount of detail than less important

objects, which are usually small or in the background, such as a sliver of grass. This relativism is subverted when low-poly is used as an intentional aesthetic style: each object shares the simplicity provided by a low number of polygons, which makes the distinction between important features and unimportant features less clear.

Blender and Unity coordinates

For what is specifically about the blender and unity, we will have to take into consideration that different modelling programs and graphic engines may have different coordinate systems (XYZ). Thus, when exporting the objects we will have to take into account that they must have the same coordinate system. Normally this option can be set when exporting the object.

It is important to remind this point, because the visual representation between one program and another may vary. What also affects their normal (described in the next chapter).

Normals Mapping

In 3D computer graphics, normal mapping, also known as Dot3 relief mapping, is a technique used to simulate the lighting of bumps and dents, an implementation of buffer mapping. Its function is to add a higher detail quality without using more polygons. A common use of this technique is to greatly improve the appearance and details of a low polygon model by generating a normal map from a high polygon model or a height map. Normal maps are normally stored as regular RGB images where the RGB components correspond to the X, Y and Z coordinates, respectively, of the normal surface.

In Low Poly, the mapping of normals is used to eliminate those parts of the object that are not to be seen on the model and for which we do not specifically need representation. Special care must be taken into what is related on modeling so that all faces of the object have their well-defined normals and it does not result in strange visual representations of the object.

Bump mapping

Relief mapping is a computer graphics technique whose function is to achieve a rendered surface to look more realistic by simulating small surface displacements. However, unlike displacement mapping, the surface geometry is not modified.

However, just the normal surface is modified as if the surface had been displaced. The normal of the modified surface is then used to reach lighting calculations (using, for example, the Phong reflection model) to give an appearance of detail rather than a smooth surface.

Relief mapping is much faster and consumes fewer resources to reach the same level of detail compared to displacement mapping because the geometry remains unchanged.

There are also extensions that modify other surface features and also increase the feeling of depth. Parallax mapping is one of those extensions.

The limitation that we consider is the most important about the relief map is that it only disturbs the surface normals without changing the underlying surface. Silhouettes and shadows, therefore, are not affected, which is especially noticeable for larger simulated movements. Anyway, this limitation can be outstripped by techniques that include displacement mapping in which protrusions are applied to the surface or by using an isosurface.

Materials (Shaders)

Although we have already seen the normal map and the bump map, these two terms are closely related to the Materials definition of the 3D object and its texturing. In Computer Graphics, Materials are an important improvement in the field of texture mapping (and a prerequisite for advanced shading effects) that allow objects in 3D modelling packages and games to simulate different kind of materials we can find in real life. They are quite frequently used to improve the realism of polygonal meshes and other forms of 3D model data.

They gather additional properties such as: advanced rendering parameters (for example, specularity, BRDF); physical behavioural properties (such as friction, density); or the sound fires along with the texture information for the surfaces. For example: if one determinate texture makes an object look like wood, it will also sound like wood (if something hits it or scrapes along a surface), it breaks like wood and even floats like wood. If it was made of metal, it will sound like metal, it would also be dented like metal in case of, and even would sink like metal. This allows more flexibility when making objects in games.

Such materials are also valuable for the procedures' generation, where a high-level description of a model could even be increased by processing layers to produce a more detailed result (for example, by adding weather or vegetation). The material properties can also be linked to animation channels to show that its behaviour is time-dependent.

A system of materials allows a digital artist or game designer to think on objects in a different way or from a different point of view. Instead of the object being a model with an applied texture; the object (or part of the object) is made of a material.

Examples of main materials found in a video game can be: wood, concrete (or stone), metal, glass, earth, water and clothing (such as carpets, curtains or the clothes that a character with skeletal animation is wearing).

Unity and Blender materials

The materials can be very different and diverse depending on their purpose. For the 3D application of wind turbines, we will rely on a type of material known as Standard in Unity.

This material uses the following layers and their combination will achieve the visual representation of the combination of any object (metallic, transparent, plastic, etc). The materials we use in Blender and then import into unity will have to have the following ideal structure:

- Diffuse layer.

- Metallic layer

- Normal Layer

- Height layer

- Occlusion layer

We recommend to have at least the Diffuse, Normal and Metallic layer for our models.

The Unity Standard Shader is a built-in shader with a complete set of features.

It can be used to represent "real life" objects, such as stone, wood, glass, plastic and metal, and supports a wide range of types and combinations of shaders. Its functions can be easily enabled or disabled simply using or not the different spaces and texture parameters in the material editor.

The standard shader also incorporates an advanced lighting model known as “physically based shading”. Physics-based shading (PBS) simulates the interactions between materials and light in a way that tries to mimic reality. PBS is a new release on what refers to real-time graphics. It shows the better results when light and materials must exist together intuitively and realistically.

The project behind our Physics Based Shader is to create an easy-to-use, intuitive way to achieve a coherent and plausible appearance in each and every lighting condition. Model how light behaves in reality, without using multiple ad hoc models that could work or not.

In instance to achieve this purpose, we work to follow the principles of physics, including energy conservation (meaning that objects never reflect more light than the one they receive), Fresnel reflections (all surfaces become more reflective at grazing angles) and taking also into account that many surfaces are occluded (what is known as Geometry Term).), among others.

The Standard Shader is designed having in mind hard surfaces (also known as "architectural materials"), and can work with most “real-life” materials such as stone,

glass, ceramics, brass, silver or rubber. It will even accomplish a great job with non-hard materials such as skin, hair or fabric.

Standard Shader for unity. Extended Parameters

The 3D model imported by the application known as Unity, the one which has been made by, Blender, must meet the following parameters:

- Albedo Color:

The Albedo parameter controls the basic colour of the surface. It may be useful sometimes to specify a single colour for the Albedo value, but it is much more common to assign a texture map for the Albedo parameter. This is what will represent the surface colours of the object. It is of the utmost importance to remind that the texture of Albedo will not have to contain any lighting, since the lighting will be added according to the context in which the object is seen.

- Metallic:

When working on the metallic workflow (which would be te opposed to specular workflow), the reflectivity and the response of the surface light are modified in direct relation with the metallic level and the level of smoothness.

Specular reflections continue to be generated when this workflow is used, but they arise as a spontaneous phenomenon depending on the settings established for the Metallic and Smoothness levels, rather than being explicitly defined.

The metallic mode is not just used for materials that are supposed to be metallic! This mode has this name; “metallic”, because of the way it has control over the metallic or non-metallic surface.

The metallic parameter of a material determines how "metal-like" the surface is. When a surface is more like a metallic kind, it reflects with higher precision the environment and its albedo colour become less visible. At all metallic levels, the colour of the

surface is totally driven by environmental reflections. When a surface is not that metallic, its albedo colour will be lighter and any reflection of the surface will be visible at the top of the surface colour, instead of kept hidden.

By default, with no texture assigned, the Metallic and Smoothness parameters are controlled by a slider each. This is enough for some materials. However, if the surface of your model has areas with a mixture of surface types in the albedo texture, you can use a texture map to control how the metal levels and smoothness in the surface of the material vary. For example, if its texture contains a character's clothing, including some metal buckles and zippers. You would like the buckles and zippers to have a higher metallic value than the fabric of the clothes. To achieve this, instead of using a single slider value, you can assign a texture map that contains a lighter pixel colours in the areas of the buckles and zippers, and the darker values of the fabric.

With a texture assigned to the Metallic parameter, the Metallic and Smoothness sliders will disappear. However, the metallic levels of the material are controlled by the values that appear in the red channel of the texture, and the softness levels of the material are controlled by the alpha channel of the texture. (This means that the green and blue channels are ignored.) Thus, we can conclude it has a unique texture that can define areas as rough or smooth, and metallic or non-metallic, which is very useful when working on texture maps that cover many areas of a model with different requirements, for example, a map of single character texture. It often includes multiple surface requirements: leather shoes, clothing, skin that is part of hands and face and metal buckles.

- Normal map (relief mapping):

Normal maps are a type of relief map. They consist on a special type of texture that allows the user to add surface details such as bumps, grooves and scratches to a model that catches the light as if they were represented by real geometry.

For example, you may want to show a surface that has grooves and screws or rivets across it, such as the hull of an airplane. One of the ways to achieve this would be to model these details as geometry, as shown below.

It uses to be a bad idea to have details as small modelled as "real" geometry. On the right you can see the polygons that are necessary to compose the detail of a single screw. On a large model with many fine surface details, this would require also a large number of polygons to draw. To avoid this, we must use a normal map to represent the detail of the fine surface and a lower resolution polygonal surface for the larger form of the model.

However, if we represent this detail with a relief map, the surface geometry can be much simple, and the detail could then be represented as a texture that modulates the way light is reflected on it. This is something that modern graphic hardware are able to do extremely fast. Its metal surface can now be a flat low-density polyethylene plane, and the screws, rivets, grooves and scratches will trap the light and seem to have depth due to the texture.

In the art lines of a modern game development, artists will use their 3D modelling applications to generate normal maps based on very high-resolution source models.

Then, the normal maps are assigned to a version of the game-prepared model with a lower resolution, so that the original high-resolution details can be represented using the normal map.

For more information:

https://docs.unity3d.com/Manual/StandardShaderMaterialParameterNormalMap.html

The rest of the layers increase the visual representation of the material but are not as important as the previous three that have already been described above. For a brief summary of what each one brings:

- Height layer.

Height mapping (also known as parallax mapping) could be assumed as the concept of normal mapping. However, this technique is in fact more complex and, therefore, also more expensive when talking about performance. Height maps are generally used in conjunction with normal maps, and are often used to give additional definition to surfaces where texture maps are responsible for representing both large and normal bumps.

- Occlusion layer.

The occlusion layer's function is to provide information on which areas of the model must receive either high or low indirect lighting. This kind of lighting comes from ambient lighting and reflections, so the steep concave parts of your model, such as a crack or crease, would not receive much indirect light, thus being quite the same as if it would be a real-life object.

Occlusion texture maps are usually calculated using 3D applications directly from the 3D model using the modeler or the third-party software.

An occlusion map is a grayscale image, where white indicated areas should receive full indirect illumination and black ones supposing an indirect lighting. Sometimes, this is as simple as a grayscale height map, for simple surfaces (such as the texture of the knotted stone wall shown in the previous height map example).

It also could be that generating the correct occlusion texture would be somehow a bit more complex. For example, if a character in your scene is wearing a hood, the internal edges of the hood should be set to very low indirect lighting, or none of it. In these situations, artists will often produce occlusion maps, using 3D applications to automatically generate an occlusion map based on the model.

- Emission layer.

The emission layer controls the colour and intensity of the light emitted from the surface where it is acting. When an emissive material is used in the scene, it automatically appears as a visible source of light.

3D Model's Export Formats

The application will import objects with the following formats: Each and every of them can be easily exported from Blender.

- FBX. Recommended.

- OBJ.

- Blender

Nomenclature of 3D objects.

To reach the proper performance of the application, a hierarchical structure defined according to the use that the engine will make in real time will be constructed. In a more advanced phase of this project the basic study of how this hypothetical hierarchy of 3D objects will be carried out, and it will also be adapted to the program already considered.

3D scene

Each new tutorial or practice to perform from the virtual reality application will import a 3D scene with two separate files.

On the first hand, the total 3D model, with his sub-models correctly scaled, positioned and with their well-defined materials according to the parameters previously described in this document.

On the second hand, the file with the 3D tools that will be used in virtual reality practice. These tools must be scaled correctly and also represent a real size so that they can be used by users in practice.

Blender / Unity units of measure

One of the most important points to take into account would be the units of measurement in Unity and Blender, so that the scaling and proportion of the objects can be represented in an adequate way.

The units are a little bit arbitrary in 3D engines, even in Unity. Even though, sometimes there will be aspects of the engine that expect a certain unit on a real scale.

Choosing the relation as if 1 unit corresponded to 1 meter is a safe bet on Unity for de high majority of the projects. So, in what concerns to Blender, the optimum would be to count each unit of measurement as 1 meter.

Template and operation

The operation of the template and its configuration for the easy acquisition of the data that will be necessary for modelling and its subsequent integration into the functional basis of the application has represented one of the points that has generated more debate in the internal side of the development of the project we are currently talking about. The search for balance and weight of the three main areas: Engineering (Technical and Configuration), Programming and Graphic Design have led to a consensus that gathers the bases to improve the easy-to-use concept without neglecting the formality required by a project of these characteristics.

The result of this, states:

As a basis for the obtainment, there must be one (1) BLENDER file containing each and every part of the wind turbine model to be represented, reflecting thus with maximum fidelity its operational representation.

It is also established that it will be the SIMULWIND application which automatically will import (after identifying the file described above) all the elements considered necessary to model virtual reality and 3D (depending on the access mode), as well as to generate the corresponding library, decomposing the elements of unit form in BLENDER files, and identifying or classifying them according to the following coding:

AERO-nn-xxxxxxxx-REVn

In this code, AERO- represents the fixed part, while “nn” is showing the correlative number of imported wind turbine (represented in a single BLENDER file), starting from 00 on our project.

xxxxxxxx will be related to a numerical sequence with which SIMULWIND will automatically and uniquely identify each of the wind turbine parts represented in the BLENDER file.

REVn is the way we understand the last revision of the file in this code (by default it will be 0).

Thus, for SIMULWIND to be able to do the import in a successful way, the template to be used must be (in our case and for the first wind turbine, object of this project):

BLENDER file containing all the parts and complete representation of the wind turbine that must also fulfil the characteristics recommended in this document with the duty to keep the LOW POLY with the following file name:

AERO-01-00000000-REV0

SIMULWIND (once executed in administrator mode) will allow the file to be imported by performing the following actions in an automatic way:

- Full set capture.

- Capture each element of the set separately.

- Generation of “n” files (where “n” is understood as the number of pieces contained in the main file) in BLENDER identifying them following the numbering described above.

This process will generate a structure of folders and subfolders within the files corresponding to the application in the system as follows:

“AERO-01” folder

Subfolder "Main File"

AERO-01-00000000-REV0

Subfolder "Library Elements"

AERO-01-nnnnnnnn-REVn / and successive

SIMULWIND will allow that every single model related to a turbine will be named, which will be useful for an easy search, as well (in later stages of the process described in other documents), it will be necessary to identify the model parts of the main file by free text, or if necessary to import them from the library, which will be used for interactions during training activities.

ANNEX 1

SIMULWIND EXPLORATION