3D MODEL PREPARING PATTERNS FOR INTERACTIVE URBAN VISUALIZATION

GUIDELINES FOR GRAPHIC DESIGNERS PREPARING 3D MODELS FOR VIRTUAL REALITY APPLICATIONS

DOMINIK PIELAK, MATEUSZ KOWALSKI AND JACEK LEBIEDŹ

Gdansk University of Technology
Faculty of Electronics Telecommunications and Informatics
Department of Intelligent Interactive Systems
Narutowicza 11/12, 80–233 Gdansk, Poland

(received: 23 July 2018; revised: 24 August 2018;
accepted: 19 September 2018; published online: 2 October 2018)

Abstract: While working on architectural visualizations, the software developer often has to work with graphic designers who create models in a different environment what can cause many complications. For this reason, it is very important to have some guidelines which can protect both the developer and the designer from commixing mistakes. The paper presents a list of such guidelines based on the authors' experience. The reader can treat the paper as a first step in the development of a service based on cloud computing that verifies the correctness of graphical data in urban visualizations.

Keywords: architectural modeling, 3D modeling, architectural visualization, urban visualization

DOI: https://doi.org/10.17466/tq2018/22.4/a

1. Introduction

Architectural visualizations are useful in many cases because they allow people to imagine what the final result will look like much easier than 2D technical drawings. Nowadays, users expect to be able to interact with the visualization in real-time, *e.g.* moving objects, changing colors and many other tiny modifications. These expectations can be met by properly imposed requirements for both software [1] and data (3D models). This gives an opportunity for real-time rendering engines like Unity [2] or Unreal Engine [3], which are usually associated with the game development only.

An interactive 3D visualization is very sensitive to errors in models. These errors can occur at various stages. They can be generated by interpolation or approximation methods used for terrain representation (digital elevation models) [4] or by computational simulations that deal with the numerical solution of a set of some equations (*e.g.* partial differential equations) [5]. Errors can also occur due to simplification approaches which primarily use the geometric distance metric as simplification criteria [6] or 3D reconstruction in automatic building modeling [7]. However, mistakes can be made by humans. They can appear during the design process (architectural drawing errors) [8], but also during preparation of a virtual mock-up for prototype demonstration using advanced virtual reality displays (HMDs or CAVEs). In the paper we will deal with the latter kind of errors, especially since we did not find any publications on the subject. Mistakes are made by designers using both simple [9] and complex tools like Blender [10], 3ds Max [11] or SketchUp [12].

2. Research details

Since of the Immersive 3D Visualization Lab [13] was opened at the Gdansk University of Technology (GUT), students have been able to create architectural visualizations in which the user can be fully immersed in a virtual world. This possibility came with many requirements related to the production environment what resulted in the need of cooperation between students from different faculties. Students from the Faculty of Electronics, Telecommunications and Informatics at the GUT started creating visualizations, while models were provided by the Faculty of Architecture at the GUT and the Academy of Fine Arts in Gdansk. As a result of such cooperation guidelines with architectural modeling patterns for real-time rendering visualizations were created.

In the past few years students were working on many projects related to land development in Gdansk and its surroundings of which the following visualizations are worth mentioning:

- the Coal Market in Gdansk – the visualization is composed of three different variations of the Coal Market which were created as part of the competition organized by the City Hall in Gdansk. Three works created in 3D Max by students from the Faculty of Architecture at the GUT were later transferred to Unity and adapted to the Immersive 3D Visualization Lab [14];
- the Vistula Mouth Fortress and the Water Forge in Gdansk – a visualization of a spatial management design for both of these historical objects. Two 3D scenes prepared in SketchUp by students from the Academy of Fine Arts in Gdansk were imported to Unity and visualized in the Immersive 3D Visualization Lab [15]. A new similar project concerns the locality of Czarna Woda in the Tuchola Forest (study project "Slow City").

All the aforementioned visualizations were created in the Unity engine and adapted to the Immersive 3D Visualization Lab. This laboratory is a CAVE installation where image is displayed also on the walls, ceiling and floor. It allows

the user to be fully immersed in a virtual world. Additionally, special controllers allow the user to navigate through the visualization without decreasing the level of immersion [13].

3. Discussion of patterns

Simple architectural modeling guidelines were developed as a result of the research and experiments [16, 15]. They are an answer to the problems encountered when creating architectural visualizations in the Unity engine. Most of these issues are related to the external 3D graphics software and can be easily used whenever exported models have to be used in real-time rendering systems.

3.1. Naming Convention

The first aspect covered by be guidelines is related to the naming convention. The importance of this issue is often underestimated by artists, however, it is crucial for keeping the project clear and easy to maintain, no matter how much it will grow and how many people will be involved. The proposed practices included in the list are as follows:

- Objects should be named appropriately and in accordance with the context. It makes modification and review easier. Default names like "Plane0" or "Cube1" should be avoided, as well as names which can confuse the developer, *e.g.* walls or windows named "Roof" or "Floor";
- Names should not contain special characters, specific for a given language. It may cause problems when the engine does not support them correctly. Moreover, it can cause problems when the environment is updated to a new version. In this case objects may lose the references to geometry, materials or textures (reimported assets will have different names);
- Names should be in the same language throughout the whole project. The best choice is English because it is currently the standard language in IT, which is a strongly multicultural environment. This standard facilitates reading and searching in the hierarchy of objects.

3.2. Hierarchy

An equally important aspect as the naming convention is the model hierarchy and the way in which geometry is divided into smaller items. A wrong division may cause the model to be unusable in the production environment. While making a division it is worth paying attention to several things:

- The division of a model into subobjects should be made based on items which are unitary and connected. The designer should avoid a situation where the geometry of one subobject is composed of many separate parts. It may cause problems with matching the elements together, when the developer has found a small gap between two of them. When such an object is scaled or moved, gaps in different places of the model will be created;
- Intermediate nodes should have no geometry attached. Only leaves should contain geometry. It will allow the dependency between objects in a hierarchy

to be easily modified or removed. Intermediate nodes can only be used as containers which help in managing groups of objects;
- Pivots at each hierarchy level should be set in a correct place. In most cases the point can be placed in the center of mass or in a place where the object should connect with another one. Wrongly placed pivot points make rotation or scaling impossible. Moreover, the position of objects on the scene becomes meaningless. The developer cannot determine the actual distance between objects;
- Models should not contain more than one building inside. A single 3D model should be associated with a single building or element. With this approach the developer has multiple small files instead of one big file, which is difficult to maintain and develop.

3.3. UV Mapping

Good UV mapping is a key factor affecting the final appearance of visualization. The developer would have limited possibilities without it. The lack of standards in this area can make further work very difficult. Some important aspects to be taken into account during development are listed below:

- Models should use as few materials as possible. It forces the creator of models to split a model into sufficiently small pieces, what makes managing them much easier. Additionally, the developer can later group objects which use the same material together and quickly change the material in multiple objects at once. Moreover, materials have direct impact on the performance. While displaying an object, each material causes at least one draw call during which the whole model is processed. If models are smaller and use fewer materials, less geometry is processed what accelerates the visualization;
- UV mapping should be done in a specified order for all models in the visualization, *i.e.* in the case of many materials, they should be ordered in the same way (*e.g.* for windows – glass first, borders second). It makes modification of many objects at once to be much faster;
- The designer has to take care of the correctness of mapping when copying or modifying the existing geometry. Many transformations such as mirroring can cause incorrect mapping (Figure 1);

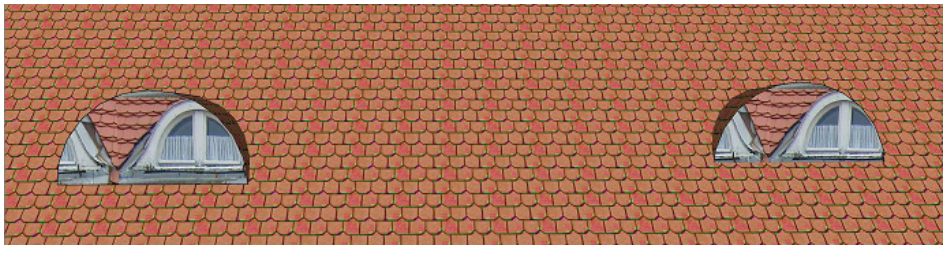

Figure 1. Incorrect texture mapping [15]

- It is worth exporting a model without textures and adding them separately to the project. This approach gives many benefits: the file size is smaller, the

import is faster, the engine does not create duplicates of textures and other garbage (*e.g.* Unity creates a folder `*.fbm` for all models exported from an earlier version of Autodesk 3ds Max);

- The texture resolution should depend on the size of the object. Big objects should have the texture in a higher resolution. Additionally, in the case of the texture which is repeated many times on the model, seams should not be visible. It means that it should be possible to merge edges without visible lines.

3.4. Modeling

In this section the general rules related to the 3D model are described. They will ensure that the model will be correctly exported from the modeling software and imported to the production environment. Moreover, they will help to avoid the most visible shortcomings:

- Faces should not overlap (even partially). Failure to meet this condition causes flickering of overlapping faces which "fight" to be visible on the screen. This effect is called Z-fighting and is especially evident when moving around the 3D scene;
- Normal vectors of faces should be set correctly *i.e.* in the direction from which the user can see the model. One of the most common mistakes is related with/to inverted normal vectors, because modeling software usually displays both front and back faces of the model. Real-time rendering engines (depending on the shader) display only the front face which can make a part of the model invisible if normal vectors are inverted (Figure 2);
- Modeled objects should have a volume. It allows easy matching of elements and adjusting their parameters. By increasing slightly the number of vertices, the designer decreases the visibility of many shortcomings like small gaps between the geometry or floating fragments. This solution helps with Z-fighting problems as well;
- While modeling the author should think which fragments of the model will be visible for the user to avoid situations when back faces of the geometry are seen. When the building has semi-transparent windows or doors, the interior should be modeled as well. Otherwise the user will be able to see through the walls;
- Modeled objects should be fitted exactly to the surrounding elements of the 3D scene. When the whole roof or the floor of a building is modeled as one object, the developer may have a problem with fitting them in the final destination without any defects or intersections with walls. In this case it is better to split them into smaller pieces which will be easy to use. Objects should not intersect each other. This eliminates elements of other objects protruding incorrectly above the faces (Figure 3);
- There should not be any holes in a 3D scene (Figure 4). Each face should touch other faces by all edges. There should be no gaps between adjoining objects. Buildings and roads cannot levitate above the ground (Figures 5–6);

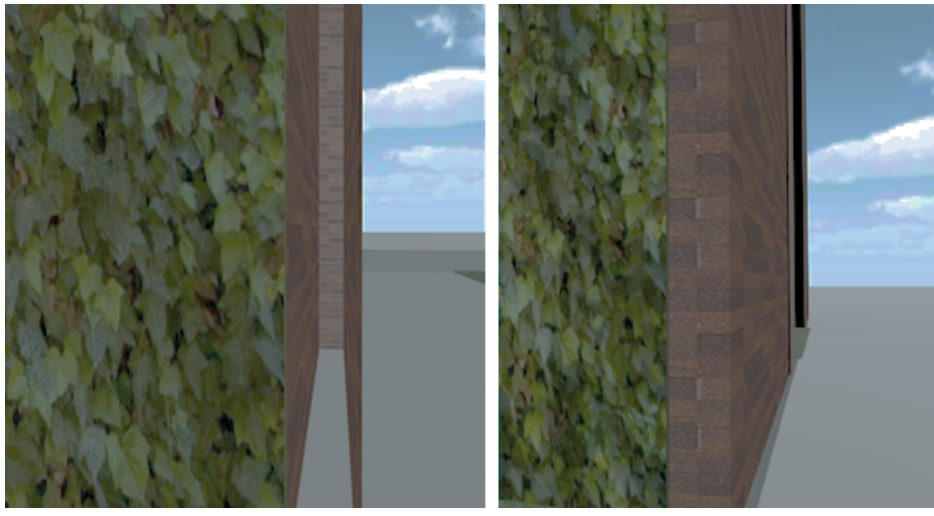

Figure 2. Fragment of a building with an incorrect (left) and correct (right) normal vector [16]

Figure 3. A tree penetrating the walls of a building [15]

- The geometry of an object should be as simple as possible because its complexity has direct impact on the performance of visualization. The creator of a model should consider how many details are needed and from what distance the user will be able to see the object. It is important especially while modeling decorations and furniture which are a side element of visualization. In this case the developer can create guidelines with limitations about the acceptable

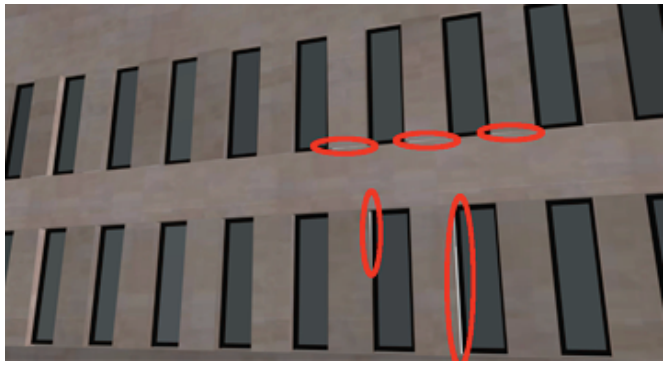

Figure 4. Undesirable slots in the building's wall [16]

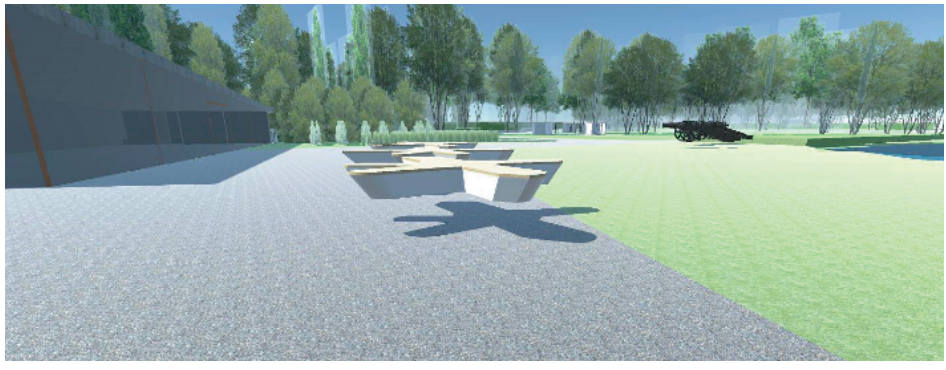

Figure 5. Levitating benches [15]

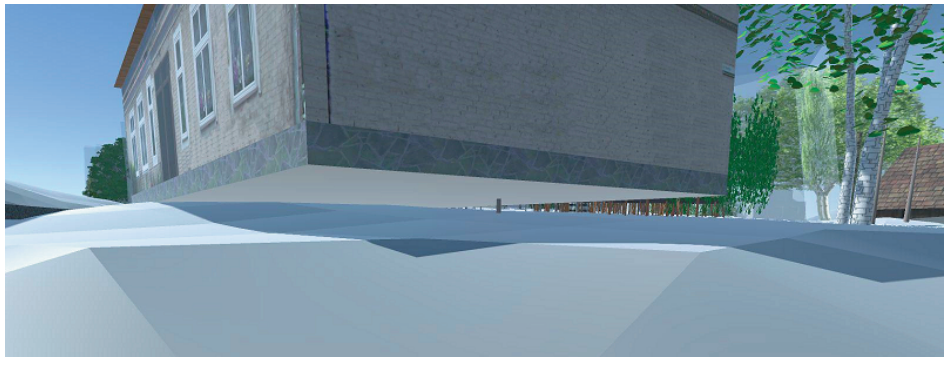

Figure 6. Levitating building [15]

number of vertices per object. It is recommended to model the vegetation (every tree or bush) as two intersecting sprites.

3.5. Unity Project Organization

When the developer receives a well prepared model, he or she can still follow a few good practices which make the model easier to use and maintain in Unity.

This section is mostly related to a proper configuration of the Unity importer and the project structure:

- While importing the model, it is worth disabling the import of animations if objects do not have them. It decreases the number of components on the scene and may slightly improve the performance. Moreover, basic actions in the editor like loading/saving of scenes can be faster as well;
- The scale of objects in the import settings should be set correctly in such a way that the object in scale (1, 1, 1) represents its real size;
- Sometimes it is worth disabling the import of materials. This import increases the integration time of a new model because the developer has to set up them manually and create a prefab, however, it allows precise management of all resources and easy detection and removal of duplicates;
- In the case of many identical fragments in the exported object (*e.g.* chairs, decorations or furniture), the designer can export them separately. The developer may later duplicate these objects many times and place them manually on the scene. It will decrease the memory usage and improve the performance because less data will be processed by the graphics card;
- The latest versions of Unity allow importing 3D scenes directly in the SketchUp format (`*.skp`). It is pointless to export a scene prepared in SketchUp to another format, because many details may be lost in conversion. However, if the designer uses a 3D graphics tool other than SketchUp, saving 3D scenes in the `*.fbx` format is suggested. On the other hand, Unity supports many different formats of 3D models, including files like `*.3ds`, `*.blend` or `*.skp`. This allows designers to quickly iterate and check how a model looks like in the engine, without exporting it.

4. Conclusions

Collaboration between developers and designers creates many problems. Designers usually are not well prepared for requirements of real-time visualizations and have to modify and adjust their models many times. The most popular problems described in the paper are related to:

- textures and UV mapping – too low quality, missing or mismatched textures, wrong UV mapping, too many materials;
- hierarchy and names – misleading names, unsupported names, bad division, wrong pivots;
- models – floating fragments, robust geometry, incorrect normal vectors, Z-fighting, lack of precision, gaps in the geometry, intersection between pieces;
- import to the visualization engine – import of needless features of object like animations, false scale, duplicated identical objects.

The problems described in the paper may be a starting point to establish a service based on a computing cloud, *e.g.* in the Academic Computer Center CI TASK at the Gdansk University of Technology. This service could check 3D

models prepared by graphic designers before their visualization. In the further future, it could make adjustments of such 3D models to correct the visualization. It can be expected that the existing direct high speed fiber optic connection between the Immersive 3D Visualization Lab and the Academic Computer Center CI TASK would allow visualization in real-time of scenes corrected in this way.

Acknowledgements

The authors would like to thank Professor Iwona Dzierżko-Bukal from the Academy of Fine Arts in Gdansk and Karolina Życzkowska, Ph.D. from the Faculty of Architecture at the Gdansk University of Technology for supervising the preparation of architectural models by students.

This work was supported in parts by the Entity Grant to Finance the Maintenance of a Special Research Device (SPUB) from the Ministry of Science and Higher Education (Poland) & DS Funds of the Faculty of ETI at the Gdansk University of Technology.

References

[1] Petit M 2016 *Procs. 21st International Conference on Web3D Technology*, ACM, New York 61 doi: https://doi.org/10.1145/ 2945292.2945301
[2] Unity 2018 *Unity 2017.3* [online] https://unity3d.com
[3] Epic Games 2018 *Unreal Engine 4.18* [online] https://www.unrealengine.com
[4] Gousie M B 2005 *The 4th Workshop on Dynamic & Multi-dimensional GIS* 42
[5] Oberkampf W L, DeLand S M, Rutherford B M, Diegert K V and Alvin K F 2002 *Reliability Engineering and System Safety* **75** 333
[6] Rajput D S, Kishore R R 2012 *IJCSES* **3** (2) 39 doi: https://doi.org/10.5121/ijcses.2012.3205
[7] Michelin J-C, Tierny J, Tupin F, Mallet C and Paparoditis N 2013 *Proc. ISPRS Conference on SSG*
[8] Kazaz A, Acıkara T, Ulubeyli S and Koyun H 2017 *Procedia Engineering* **196** 1018 doi: https://doi.org/10.1016/j.proeng.2017.08.044
[9] Yeh Andy 2017 *Procs. 22nd International Conference on 3D Web Technology (Web3D'17)*, ACM, New York doi: https://doi.org/10.1145/3055624.3075953
[10] Blender 2018 *Blender 2.8* [online] https://www.blender.org
[11] Autodesk 2018 *3ds Max* [online] https://www.autodesk.eu/products/3ds-max
[12] SketchUp 2018 *Think in 3D. Draw in 3D* [online] https://www.sketchup.com
[13] Lebiedź J and Redlarski J 2016 *24th International Conference on Computer Graphics, Visualization and Computer Vision WSCG 2016 – Poster Papers Procs.*, Skala V ed., Plzeň 69
[14] Lebiedź J and Szwoch M 2016 *Procs. Federated Conference on Computer Science and Information Systems*, Sikorski M ed., Gdansk 1641 (4th Conference on Multimedia, Interaction, Design and Innovation MIDI'16)
[15] Kowalski M, Motyl M, Nidzgorski A and Wantka W 2018 *Virtual sightseeing of the Vistulamouth Fortress*, Gdansk University of Technology, Faculty of Electronics Telecommunications and Informatics (Student Group Project, in Polish: Wirtualne zwiedzanie Twierdzy Wisłoujście)
[16] Pielak D 2017 *Architectural Visualization in Unity, MSc. Thesis*, Gdansk University of Technology, Faculty of Electronics Telecommunications and Informatics (in Polish: Wizualizacja architektoniczna w Unity)