Преглед НЦД 27 (2015), 27–34

Đorđe Manoilov, Nikola Gajić,
Miloš Stošić, Dušan Tatić
Faculty of Electronic Engineering, Niš, Serbia

A VIRTUAL TOUR OF THE MEDIANA ARCHEOLOGICAL PARK USING UNITY 3D ENGINE

Abstract. This paper describes a virtual tour through the archeological park Mediana developed by using the Unity3D Game Engine. The main goal of the application is to achieve better promotion of the archeological park through the use of modern information technologies. The solution is implemented as a multi-platform application for Windows, Mac, and Linux operating systems, as well as a Web application. The paper presents an analysis of basic concepts of Unity3D game engine and discusses the proposed architecture of the application. Special attention is paid to the interaction between the user and the application.

Keywords. Virtual tour, Unity, game engines, software implementation.

1. Introduction

In today's era of information technologies simple presentations in the form of images, schemas, or textual panels cannot keep up with the increasing demands of museum visitors. New technologies, such as mobile applications for Android and iOS smart phones, Web and desktop applications for Windows, Mac, Linux Operating systems (OS) improve museum exhibitions and make them far more interesting to visitors and help to develop the necessary interaction between visitors and museum artifacts. There are many examples of 3D virtual tours through museums on the Internet. A very good illustrative example is the Smithsonian National Museum of Natural History [6].

This museum is just one of many museums that provide virtual tours with 360 degree views. Also worth mentioning are the Warhawk Air Museum in Nampa (USA), the V&A Museum of Childhood in London (UK), and the Museum of Aviation in Belgrade [7, 8, 9]. All these applications allow users, which may never visit actual museums, to make a virtual tour through the museum from their homes. Main disadvantage of these tours is the lack of interaction the users and objects and the lack of object dynamics.

Another project is online multiplayer Serious Game ThIATRO that helps students to learn art history [5]. In contrast to previous examples, in this game there is much more interaction between the user and virtual objects, but all objects are still static.

Based on previous examples, we decided to make an improved application that enables a higher level of interaction between users and objects and allows more dynamic objects. The basic idea is that users can take a tour through the 3D reconstruction of objects in a museum or an archaeological park to see and interact with objects of their particular interest. For the purpose of this paper, we have developed a virtual tour through the reconstruction of the archaeological park Mediana in Niš, Serbia.

Mediana is a well known archeological site containing remains of several important Roman buildings. It is situated about 5 km from the present day city of Niš or

Naissus as it was known in Roman times. Mediana was built in the early 4th century AD, at the time of Constantine the Great. Up until now, about 80 buildings have been discovered in an area of 40 hectares. Some of the most important are: Roman villa suburbana - villa with the peristyle, thermae, granary (horreum), military barracks, south and north church, water supply system with aqueduct and water tower, tomb with frescoes, etc. The goal of our project is to allow a virtual visitor to walk around the 4th century Mediana and see it as it looked like in its prime. Our virtual reconstruction of Mediana involves the following objects: villa with the peristyle, south church, north church, horreum, military barracks, and bronze railing. Figure 1 shows renders of some of the above-mentioned 3D models.

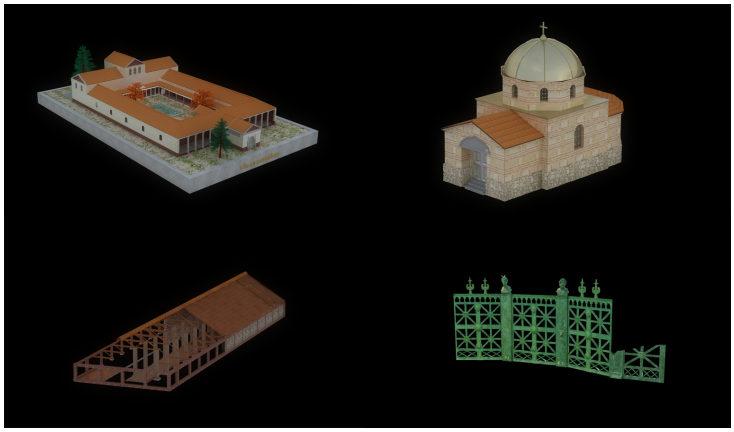

Figure 1 - Renders of 3D reconstructions of objects at the Mediana archeological park: villa with the peristyle (upper-left), the southern church (upper-right), the horreum (lower-left), and the bronze railing lower-right).

We chose the Unity game engine for application development, primarily because it allows cross-platform construction of software. While developing applications, developers can use objects which are provided by Unity (*First Person Controller*, *Terrain*, etc.). To use these objects, developers must set certain parameters which define their behavior. However, developers must implement additional functionalities such as tree colliders. Besides tree colliders, we had to develop several scripts primarily for calculations of distance between users and objects of interest. These calculations allow users to hear more information about the object or to enter it.

The paper is organized as follows. In the next section, we discuss the proposed architecture of the application. After that, we give some suggestions for solving problems that developers may face in the development of similar applications. Paper ends with some conclusions and directions for future work.

2. Implementation of the Application

The application is developed by using the Unity game engine (version 4.2), as a desktop application for Windows, Mac and Linux OS, as well as a Web application which runs in the users browser using the Unity Web Player. According to the definition, the term “game engine” is reserved for software that is extensible and can be used as the foundation for many different games without major modifications [10]. After launching the application, a user can take virtual walk through the reconstructed archaeological site by using standard peripherals (mouse and keyboard). Users can interact with a number of info-points which are placed throughout the map in vicinity of objects. In this

way, they can hear additional explanations about the objects. Users also have a possibility to open doors and enter buildings, to see their interiors. With this kind of application, a user can adapt the exhibition according to his interests and dictate the pace of the tour.

All 3D models of reconstructed buildings are positioned on their foundations by using the Google map as shown in Figure 2.

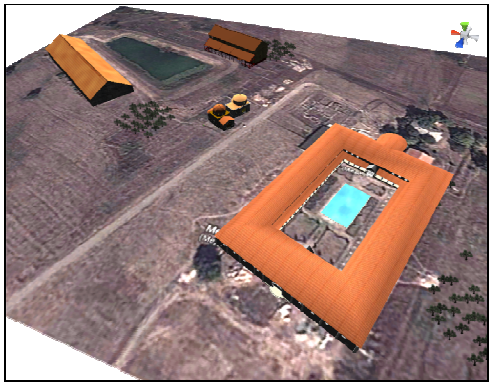

Figure 2 - Positioning objects using Google Maps.

2.1. The application architecture. Architecture consists of the application logic, developed by using game engine provided by Unity, and external resources, such as sounds, textures, models, and animations. Figure 3 gives an illustration of the application architecture.

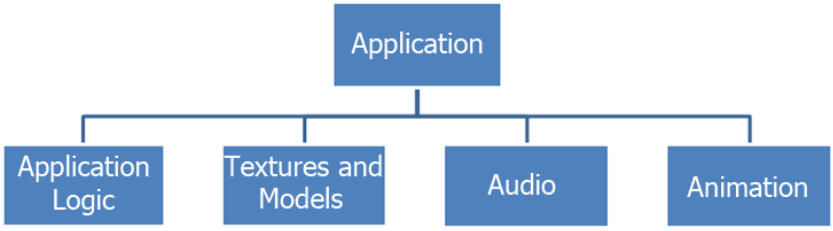

Figure 3 - The application architecture.

Unity is primarily a 3D development environment. Thus setting up the scene and the entire interface is similar as in other software tools for 3D modeling [4]. All objects, which are added to the scene, are represented as Game Objects on which Components (*Audio Source*, scripts, etc.) that define their behavior are added. In the rest of this section, we give a detailed description of Game Objects and Components that are used in our application.

2.2. First Person Controller. *First Person Controller* (*FPC*) is the most important Game Object in the application. It is controlled by user by using mouse and keyboard for rotation and movement, respectively. The control is similar to any *First Person Shooting* (FPS) game. Movement logic is realized by using C# scripts that are assigned to the controller. These scripts are part of the Unity game engine. In the object hierarchy, *FPC* has two child objects the *Graphics* and *Main Camera*. As mentioned earlier, the parent-child relationship is very important in complex objects. In the case of the *FPC*, they are particularly pertinent, as wherever the parent object moves and rotates, the child objects, most importantly the *Main Camera*, moves with it [2].

FPC Object has the main scripting and *Character Controller* collider controls its behavior. The *Character Controller* acts as a collider (a component giving the object a physical presence that can interact with other objects) and it is specifically designed for the character movement and control within the world [2]. It consists of the following parameters: *height*, *radius*, *slope limit*, *step offset*, *skin width*, *min move distance* and *center* (Figure 4).

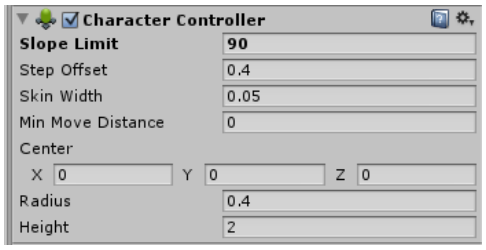

Figure 4 - Character Controller parameters.

The role of those parameters can be specified as follows:

- *Height* –defines the height of the character, i.e. defines how tall the capsule-shaped collider will be.
- *Radius* – determines how wide the capsule-shaped collider will be. This has a default radius that matches the radius of the *Graphics* child object.
- *Slope Limit* – while taking into account the uphill movement, this parameter allows us to specify how steep an incline can be before the character can no longer walk up to it. In our particular case, it is set to 90 to allow the character to climb the stairs.
- *Step offset* – specifies how far from the ground the character can step up, the higher the value, the larger the distance they can step up.
- *Skin Width* – this parameter defines how deeply other colliders may intersect with the character's collider without reacting. It is designed to help reduce conflicts with objects, the result of which can be a jittering (a slight but constant character shake) or the character getting stuck in walls. Unity Technologies recommends that you set skin width to 10 percent of your character's radius parameter.
- *Min Move Distance* – this is the lowest amount by which a character can be moved. Usually it is set to 0.
- *Center* – It allows positioning of the character collider away from its local central point. It is usually set to 0.

Graphics is simply a capsule primitive shape, which allows developers to see where they have placed the *FPC*.

Main Camera is positioned at the point where one would expect the player's eye level to be. The *Main Camera* is intended to provide the viewpoint and has scripting applied which allows looking upwards and downwards. The sky is realized using *SkyBox* component that is applied to the *Main Camera*.

2.3. Terrain. This is the base on which all other objects are placed. In addition to the reconstructed objects prepared in advance in 3D Studio Max, terrain has its own facilities such as grass and trees. There are some parameters such as height, width, length of the terrain, etc., that need to be set (Figure 5).

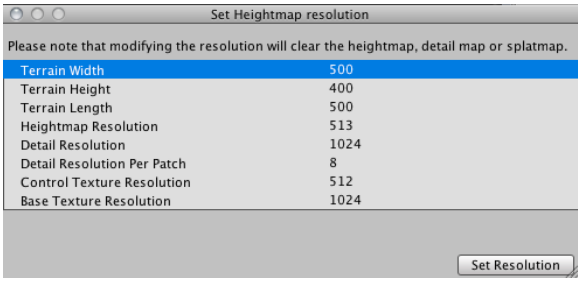

Figure 5 - Terrain parameters.

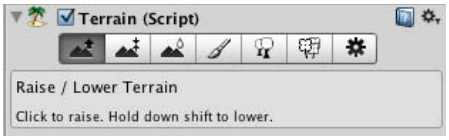

Figure 6- Terrain tools .

There are a number of tools for changing terrain relief, for adding textures, trees, grass and flowers on the terrain (Figure 6). It is important to note that trees don't have colliders when they are added to terrain, so developers must create them. A solution for this problem that we applied in the application is the following:

1. Find the desired tree in the *Project* panel.
2. Drag it out as a prefab somewhere onto the map to see it.
3. Select the tree, and in the *Components* menu, under *Physics*, add a capsule collider.
4. Set the radius of the collider to be between 0.8 and 1. Make sure the height is about 20.
5. Make a new prefab and call it *BigTreePrefab* and drag your new tree into it.
6. Under terrain tree painter, add the tree called *BigTreePrefab* (Fig. 7).
7. Place trees with collider on terrain.

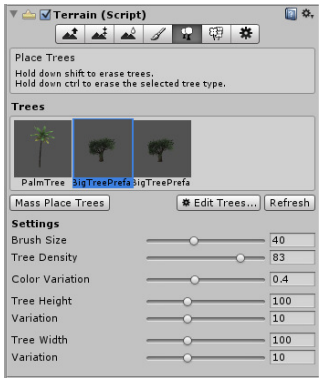

Figure 7 - Creating tree.

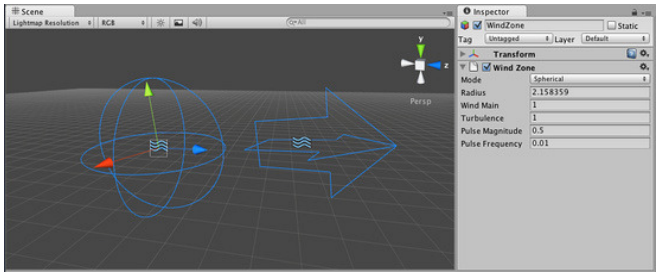

Figure 8 - Creating wind.

Trees are swaying under the influence of the object *Wind Zone* (Figure 7). Wind strength can be regulated by setting the appropriate parameters (Figure 8).

To produce a softly changing general wind, we propose the following procedure:

1. Create a directional wind zone.
2. Set *Wind Main* to 1.0 or less, depending on how powerful the wind should be.
3. Set *Turbulence* to 0.1.
4. Set *Pulse Magnitude* to 1.0 or more.
5. Set *Pulse Frequency* to 0.25.

The parameter setup for the considered application is shown in Figure 9.

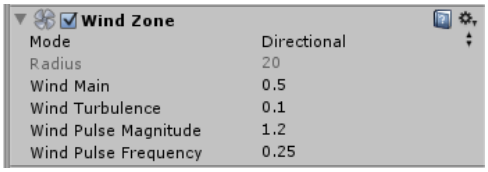

Figure 9 - Wind parameters.

2.4. Including objects constructed in 3D Studio Max. All reconstructed objects and info-points are *.FBX* models which are realized as low-polygon models using 3D Studio Max. When imported into Unity and added to the scene, these objects are represented as *Game Objects*. Info-points are *Game Objects* to which *Audio Source* component is added. These components allow users to hear some additional information about reconstructed objects (horreum, army barracks etc.). Further, the *Animator* component is added to info-points. This component is used to animate the letter "i" that is rotating above them. The *Animator* component is also added on some of the reconstructed objects (North and South Church). It is used for animating the opening of the doors. Water in impluvium is realized by using *Daylight Simple Water* object. Intensity of the water fluctuations can be changed by setting parameters in the inspector.

2.5. C# scripts. Besides *Game Objects* and *Components*, for the purpose of this application, we have developed a number of scripts:

- *Distance-to-board:* a script that is added to the *FPC* and used to calculate the distances between the controller and info-points. If the user is close enough to any info-point on the screen, a message to press the button is displayed. This allows hearing more information about the object (Figure 11).
- *Distance-to-door:* a script that is added to the *FPC* and used to calculate the distances between the controller and doors of some object. If the user is close enough to any door object on the screen, a message to press the button to open the door is displayed (Figure 10).
- *CrossHair:* a script that is added to the *Main Camera*. This script is used to display crosshair in the screen center, in order to navigate through the archeological park.

Figure 10 - Opening of doors.

Figure 11 - Playing sounds

2.6. Application testing. The roof covering of the Mediana archeological park is currently being constructed and, therefore, the site is closed for visitors until the spring of 2015. For this reason, we were unable to perform the on-site test of the application and it is still not included as part of the exibition at Mediana. We have a preliminary agreement with the management of the park that our application will become a pilot project in the new exibition, after the completion of reconstruction.

In order to gather feedback from users, which we could use to improve the project, we offered a group of students at our faculty, as well as several guests of our laboratory, to test the application. Based on these test results, we can state that the interaction between users and our application is intuitive. Test users also confirmed that the idea of the application seems very interesting. Several users also proposed that, in addition to existing functionality and content, we should include even more information about Mediana, either as text or video. A number of test users also expressed a wish to have an option to see the present-day appearance of the archaeological park using our application.

3. Conclusions and Future Work

In this paper, we presented an application of the Unity game engine to the implementation of virtual tours on the example of archeological park Mediana, in the vicinity of Niš, Serbia. We discussed the architecture of the application, composed of application logic, textures and models, audio files, animations. We described in detail the use of Unity's Game Objects and Components in the construction of virtual tours. For the purposes of the presented research, we constructed three new scripts which improve the functionality of the *First Person Controler* and *Main Camera* components of the Unity development environment.

The development of the discussed solution is currently in its first demo phase and it is available in the form of Windows desktop and Web applications. The next step is to port the application into Android and iOS. We also plan to add dynamics to *SkyBox* in order to create the day-night cycle. Interior of all of the objects will also be created in the near future. In this way, we will complete the virtual tour through Mediana from the time of Constantine the Great.

4. Acknowledgments

The authors would like to thank Prof. Radomir Stanković for giving guidance and advice throughout this research. The authors would also like to thank Dušan Gajić for all audio materials used in the application. The research presented in the paper was supported by the Serbian Ministry of Education and Science (project ON174026).

References

[1] Wikipedia – Unity (game engine), http://en.wikipedia.org/wiki/Unity_(game_engine), last access on November 18, 2013.
[2] W. Goldstone. *Unity 3.x Game Development Essentials – Game development with C# and JavaScript,* 2nd edition, Packt Publishing Ltd., Birmingham, UK, 2011.
[3] Wikipedia – Unity Technologies, http://en.wikipedia.org/wiki/Unity_Technologies, last access on November 18, 2013.
[4] M. Smith, C. Queiroz. *Unity 4.x Cookbook*, Packt Publishing Ltd., Birmingham, UK, 2013.
[5] Josef Froschauer, Max Arends, Doron Goldfarb, Dieter Merkl, Towards an Online Multiplayer Serious Game Providing a Joyful Experience in Learning Art History, *Games and Virtual Worlds for Serious Applications (VS-GAMES*), 2011 Third International Conference, Athens, Greece, May 2011, 160–163
[6] Smithsonian National Museum of Natural History, http://www.mnh.si.edu/panoramas/, last access on December 10, 2013.
[7] Warhawk Air Museum in Nampa, http://warhawkairmuseum.org/, last access on December 10, 2013.
[8] V&A Museum of Childhood, http://www.museumofchildhood.org.uk/, last access on December 10, 2013.
[9] Museum of Aviation in Belgrade, http://www.muzejvazduhoplovstva.org.rs/ , last access on December 10, 2013.
[10] J. Gregory, *Game Engine Architecture,* Taylor & Francis, Boca Raton, FL, USA, 2009.

manoilov88@gmail.com
foxalfa@gmail.com
milosstosic88@gmail.com
dule_tatic@yahoo.com