METHODOLOGY FOR DESIGNING VIRTUAL REALITY APPLICATIONS

Jiri Polcar, Michal Gregor, Petr Horejsi, Pavel Kopecek

University of West Bohemia, Faculty of Mechanical Engineering,
Department of Industrial Engineering and Management
Univerzitni 8, 306 14 Pilsen, Czech Republic

Abstract

This paper suggests a new virtual reality environment creation methodology, using game development IDEs as a platform. The methodology is based on previous experience with regular and virtual reality programming as well as on a compilation of other methodologies. The method was prepared for the development of a virtual therapy serious game, and was validated on more projects. The output of this method is the statement of the virtual environment scenario, the list of needed objects, their states and actions to change these states, which lead to a state machine creation and assets lists.

Keyword: virtual reality; virtual reality design approach; methodology; virtual reality development; virtual reality application

This Publication has to be referred as: Polcar, J[iri]; Gregor, M[ichal]; Horejsi, P[etr] & Kopecek, P[avel] (2016). Methodology for Designing Virtual Reality Applications, Proceedings of the 26th DAAAM International Symposium, pp.0768-0774, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-07-5, ISSN 1726-9679, Vienna, Austria
DOI:10.2507/26th.daaam.proceedings.107

1. Introduction

Development of a virtual environment is a specific task. Making small scale virtual reality projects does not require large amount of preparation or a methodical approach, but as the projects grow bigger and more complex, so grow the demands on the starting phases of the projects.

After unsuccessful attempts to start development of a virtual therapeutic serious game, we required a highly methodical approach. We needed a methodology for creating virtual reality projects consisting of scenario driven virtual environments. Unfortunately, we did not find a suitable one for our needs. We use Unity3D as a development environment for such projects, but the methodologies we found were focused rather on more general and fundamental software development approaches.

We required an approach for creating virtual worlds with interactive objects. These objects should be visualized in CAVE and also in HMDs, with interaction being the most important requirement. We weren't able to find a course for developing virtual environment that would be able to develop our experience from previous work.

Therefore, we started with small scale projects, helping us to improve our practices and technique. We gained experience and improved our methods to be able to find and avoid mistakes. Developing this methodology on more dedicated applications gave us enough experience for more complex projects. Our first projects were generally short virtual reality exposures, about 2 minutes long. Longer and more complex projects required much more sophisticated approaches and we had to re-evaluate our methodology to be able to make more stable and immersive virtual worlds.

2. Research

We based our methodology for developing VR applications on our previous experience. The first projects were the DIGITOV, a Source Engine based factory visualization tool, and various industrial visualizations made in VirTools. These development tools had quite a few technical and licensing problems, so first we needed to choose the engine that could be the base for our work.

There are virtual reality development tools out in the market, meaning there are several choices to pick from. Reference [1] describes and compares virtual reality development environments - namely the Blender, Quest3D, UDK and Unity3D. We found that Unity3D would be the best one in terms of license, ease of use and compatibility with VR hardware such as various HMDs, haptic interaction devices, etc.

Reference [2] validates the Unity3D on a university campus web-based visualization. The methodology described in this article is focused rather on creating the assets for the environment than on the logic and development of it. UDK was used in [3] for creating an edutainment game for safe grinder operation.

Valuable experience about the methodology of creation of the virtual environments are described in [4] on a ship simulator development. Reference [5] describes a methodical approach for designing the function of a virtual assembly engine. Acquiring and creating assets for a virtual world is described in [2]. A scenario-driven virtual reality method is described in [6] on an emergency management serious game. Similarly to DIGITOV, [7] suggests using Source Engine and its development tools for teaching computer science. The article contains a methodology for level design.

3. New Methodology

After several brainstorming and evaluating sessions with our colleagues who had been involved in previous virtual reality projects and analysis of the methodologies described in the references mentioned above, we put all this experience together and made a suggestion for a new methodology. After deployment on the first projects (see the Validation chapter), we made some corrections and this methodology was further enhanced. The methodology is divided into six phases, see fig. 1.

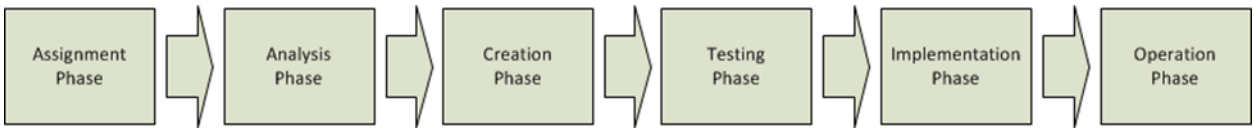

Fig. 1. Phases of the methodology.

3.1. Assignment phase

In this phase, the customer and the developer work together. This phase should be performed as carefully as possible and everything should be taken into account, as it's the foundation for further work. The customer has to have an idea of the overall concept of the virtual reality application to be able to describe his requirements and wishes, so that the designer can be able to meet them.

The customer has an idea about final product. He knows what it should be doing, how it should look like and what should happen after inputs from the users of the application. So the developer can attempt to meet his requirements.

Defining the goal of the virtual reality application is the most important part and should be the first task to be completed. It must be clear why the virtual environment is being created.

Defining the target group is the second thing to be done. The virtual environment is very different, depending on whether it's for children, adults or retired persons or whether the target group is students, customers, workers, managers, gamers or patients etc.

The next step is to make the scenario draft. The customer of the VR project describes the plot, tasks and setting of the virtual environment.

The last step of the assignment phase is to define the level of immersion and the target hardware. They both directly influence each other and there is not always a need for the highest immersion level possible. The customer has to know what hardware will be used or sets limits based on the available hardware.

3.2. Analysis phase

The output of the assignment phase now needs to be analyzed to get the foundation to build the whole application upon. This must be conducted especially carefully, because minor faults in this phase will steer the direction of progress off course.

This phase begins with the detailed scenario. This phase should be done in consultation with the customer and it involves the stating of all the tasks, actions and story that drives the virtual world. This is the foundation for the next task.

List of actions and list of objects is derived from the scenario. All objects are listed and reduced to object classes - to objects with similar or the same functions and appearance. All the actions, activities and interactions are written on another list.

Defining objects' states is the next step. In the runtime, all objects find themselves in different states. For example, a man can be walking or sitting, or he can be silent or talking. This means that there can be different layers of such states. This phase consists also of deciding which states can be combined with others.

Now, we can assign actions to the objects. The actions should be the impulses leading to state changes. Now we are able to see which types of objects share the same actions (like walking for a human and a dog). But when the actual scripting is done the question is whether the action for two different kinds of object can be driven by the same script.

All of this is summarized in state diagrams. There are state diagrams for the user, for the NPCs, for the interactive and other non-static objects and for the scenarios present in the virtual world. Each state change is driven by one or more action. After finishing the state diagram, new actions can occur. This might lead to some iterations in this phase.

3.3. Creation phase

This phase consists of creating the assets of the virtual environment - the small bricks that build the whole virtual world. The requirements for some assets can be clear from earlier phases of the analyses or even from the scenario draft. During the creation phase, the customer should be involved in the process and should participate in regular reviews and consultations. This is an extreme programming-like approach. All the assets should be tested as soon as possible and documentation should be made accordingly. There are many kinds of assets, scripts, texts, graphics, animations, sounds and hardware. The form of the assets is highly influenced by the target group and level of immersion stated in the assignment phase.

Scripts drive the virtual world. All the actions are backed by various kinds of scripts. There are scripts to hold the state of the objects, scripts that change the states. Scripts should be written as universally as possible, although there will be some scripts that are intended for a small number of particular actions that don't need to be universal. The main foundation for the scripts is the state diagram. Such scripts will result in a logical engine powering the virtual world.

Dialogue trees, items descriptions and all other texts that will take part in the virtual environment. The foundation of most texts is the detailed scenario.

By graphics we mean 2D graphics like GUI or 3D models representing objects, architecture etc. Also, there are lights, renders, shaders, etc. The graphics assets are dependent on the detailed scenario (if the virtual world is based on reality, then the scenario draft can be used). People models are based on the detailed scenario and other models on the list of objects.

Animations are based on the state diagram and they are the basis for interactions in virtual environments. Almost all state changes are accompanied by corresponding animations. Some animations do not accompany a state change, but accompany the state itself (like a running fan rotation). Some animations are based on the very nature of the objects, like door opening animations.

Some sounds are based on texts (dubbing), other sounds accompany animations and there are also ambient sounds. The quality of sound is often underestimated, but sound making is a very difficult task. The sound quality has a big influence on the level of immersion [8].

Hardware in methodology is not intended to be made, but rather to be bought, installed and prepared to power the virtual environment. The customer may require using a particular piece of hardware which is already at his disposal or will order the resulting application installed on new hardware. The basis for choosing the hardware is primarily the level of immersion and the complexity of the virtual environment.

3.4. Testing phase

This phase is done thoroughly during the whole project. Everything should be tested as soon as possible. Not only the scripts should be tested, but also all the assets together. The tests should focus not only errors and unhandled exceptions in the code, but also for the overall feel of the virtual environment.

From time to time, the partially complete virtual environment should be shown to the customer, to confirm that it is progressing in accordance with requirements. The team of creators should test every build as soon as it is made. For beta testing, uninvolved people should be looked for. Such people are able to test the understandability of the virtual environment and the controls.

3.5. Implementation phase

This phase consists primarily of deploying the software application with the hardware and calibrating it for light and acoustic conditions. Calibration is very important for stereoscopic projection and haptic interaction - the user must see the touch in the actual position. Final testing is of course conducted after deployment.

3.6. Operation phase

The project is not closed after implementation. The developers should have the opportunity to monitor the implemented virtual environment and to collect data for further debugging or modifications. A completed project brings with it a wealth of experience. A meeting between the developers and the customers should be held to talk about the experience learned from this project. It should also be stated whether the assets can be used for other projects in the future. The virtual reality application can have a built-in logging system. Such logs can be collected and analyzed. If possible, it's good to watch users' actions in order to check whether there are any problems with orientation, difficulties with controls, etc.

4. Example of use

A simplified example of using this methodology follows with a description of an edutainment serious game. This game was intended to be played by first year Bachelor students at the Faculty of Mechanical Engineering to learn the basics of machining by operating virtual machines and carrying out manufacturing operations. This should be a standalone application for multiple platforms with realistic graphics.

According to the scenario, the player receives technological process documentation and goes to the workshop. There, s/he finds the machines and starts to manipulate them so that the manufacturing operations are carried out. After finishing the whole manufacturing process, s/he puts the product to a box. If the operations are carried out in the wrong way, the player will be stopped and a message will show up. Let's crop the detailed scenario to just one machine, for example a lathe and the manufacturing process on the lathe.

After completing the scenarios, the list of objects and list of actions are filled. In case, the objects are: the player, the lathe, the box for material, the box for the finished product, message box, material and the finished product. This is a list of all of the objects in the scene. After some considerations, the list of objects is reduced to classes - the player, product, box and lathe. The actions will are: player walking, picking up and putting down an object, attaching product to machine, machining, reading message. Now, it's time to assign the actions to the appropriate objects as fig. 2 shows.

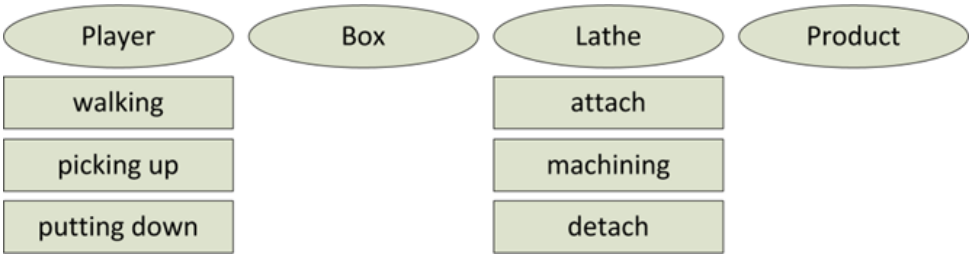

Fig. 2. Actions assigned to object classes.

Note that some object classes are passive and do not carry any actions. Because there are several solutions for this task, we recommend consultations with the script programmers. After assigning the actions to the objects, the state diagram can be made, see fig. 3 for the lathe and fig. 4 for the player. Note, that an unfinished product can't be detached from the lathe, resulting in no additional states for the product, making the state diagram much simpler.

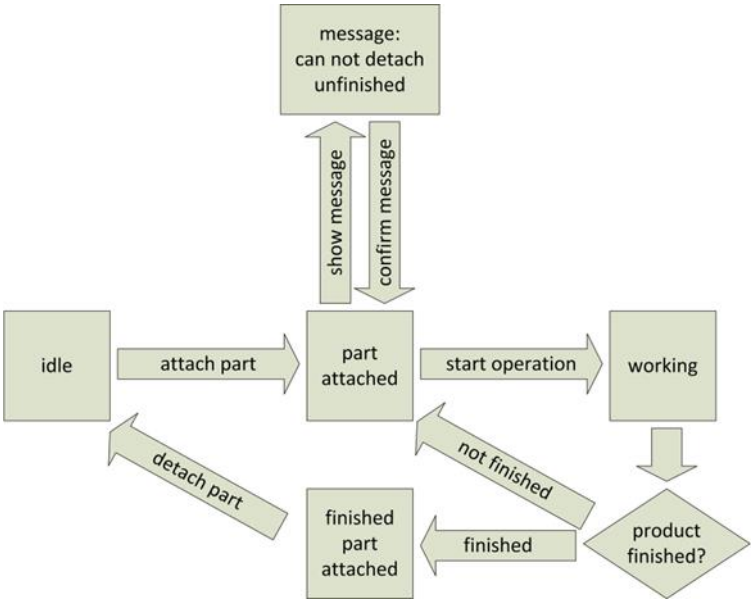

Fig. 3. The state diagram for the lathe described in the example.

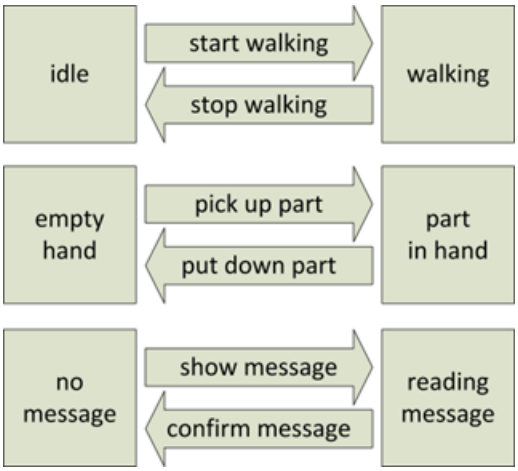

Fig. 4. State diagram for the player (the user).

Note that the impulse for displaying the message occurs in the machine's state diagram, but it changes another object's state. Such situations occur very often and must be taken into account while scripting.
According to the methodology, the following assets will be necessary for such a virtual reality application:

- models: the lathe, unfinished and finished parts, product boxes
- texts: objects labels, messages
- animations: picking part, putting down part, attaching part, detaching part, lathe turning
- sounds: machining, picking up, putting down
- scripts: state carrier for the lathe, state carrier for the player, player controls, message viewer

5. Validation

This methodology is in use and it has been validated on several virtual reality projects. We have been working on a virtual version of the Minnesota Dexterity Test (see fig. 5), which is used while hiring new employees for positions where agility and speed of hands is required. This application is ready to be used in a CAVE-like projection with a haptic device. It can also be used as a learning tool for precise and quick mouse movement.

Fig. 5. Virtual version of the Minnesota Dexterity Test.

Another virtual application is a point cloud visualisation tool for visualising data from LiDAR (LIght Detection and Ranging) scanned point clouds. These point clouds can be visualized either on a computer screen or in the Oculus Rift DK2 head mounted display, as seen in fig. 6. The LiDAR pointcloud was used to substitute the models and background environment, saving about 90% of designing time compared to the common modelling based method, although point clouds rendering lowers the performance of the resulting application.

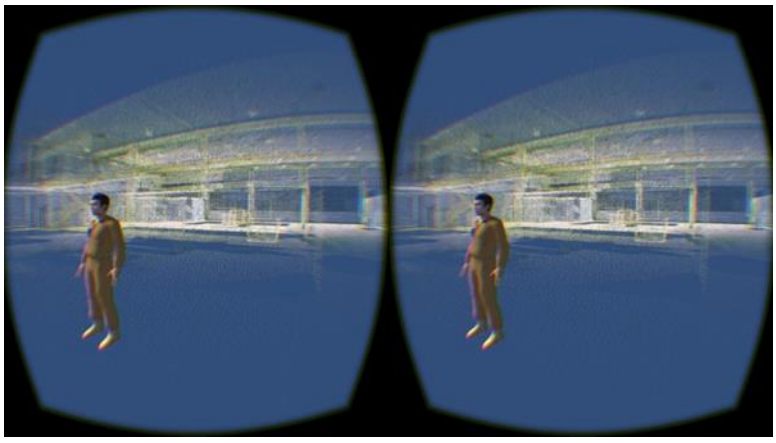

Fig. 6. An Oculus Rift viewed LIDaR point cloud with interaction capabilities.

This methodology was used in the creation of the exterior and interior of an industrial plant in Unity3D. This was intended to be an advertisement for a brewery, with the ability to examine states of intoxication by users. When the user selected the degree of intoxication the virtual environment reacts to his senses and the immersion caused the feeling of inebriation.

Another large scale project is the virtual workshop. It is intended for students at the Faculty of Mechanical Engineering who don't have experience with real industrial processes. There are a few basic machines that can be virtually operated and a part is manufactured according to the manufacturing process, see fig. 7.

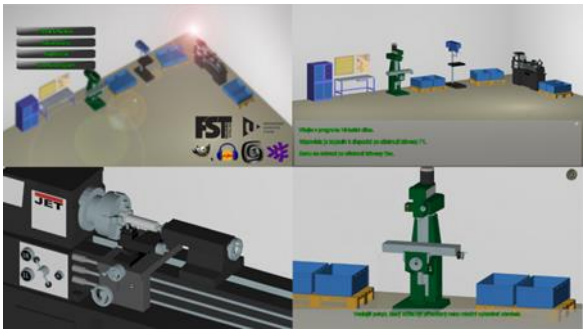

Fig. 7. The virtual workshop serious game.

The current project we are working on using this methodology is a virtual therapeutic environment for psychiatric treatment support. So far the most complex project has experienced no major faults in 9 months spent in making it, thanks to this methodology.

6. Conclusion

This paper described a process of designing a new methodology for creating virtual environments in game development tools like Unity3D or UDK. This methodology is a compilation of our long term experience with different kinds of virtual reality projects, especially with the theme of industrial plants and of collected experience from different articles describing virtual environment creation. The methodology does not describe how to prepare an engine for rendering, physical collision and such fundamental functions, but it describes how to fill such engine with the content and logic.

Unfortunately, other methodologies cited in other articles were only partially suitable for our case. They deal with a fundamental programming approach on how to build an actual engine and not how to use it.

This methodology can be applied to event driven virtual environments, where a high level of interaction between the user and the environment and the objects in the environment is needed. This applies for most games and serious games, presentations, visualizations, therapies etc.

The methodology was and is being utilized in making different virtual reality projects, so far preventing major faults that would force the creators to start over again. However, the use of this methodology does not completely exclude the possibility of such a situation arising.

7. Acknowledgements

This paper was prepared with the subsidy of the Internal Science Foundation of the University of West Bohemia SGS–2015-065.

8. References

[1] Petridis, P.; Dunwell, I.; de Freitas, S.; Panzoli, D., An Engine Selection Methodology for High Fidelity Serious Games, in Games and Virtual Worlds for Serious Applications (VS-GAMES), 2010 Second International Conference on , vol., no., pp.27-34, 25-26 March 2010.

[2] Sa Wang; Zhengli Mao; Changhai Zeng; Huili Gong; Shanshan Li; Beibei Chen, A new method of virtual reality based on Unity3D, in Geoinformatics, 2010 18th International Conference on , vol., no., pp.1-5, 18-20 June 2010.

[3] Ge Jin; Nakayama, S., Virtual reality game for safety education, in Audio, Language and Image Processing (ICALIP), 2014 International Conference on , vol., no., pp.95-100, 7-9 July 2014.

[4] Seo, J; Kim, G, Design for Presence: A Structured Approach to Virtual Reality System Design, in Presence , vol.11, no.4, pp.378-403, Aug. 2002.

[5] Yao Lili; Xueqing Li; Chen Jiayao, Research and implementation of virtual assembly training system, in IT in Medicine & Education, 2009. ITIME '09. IEEE International Symposium on , vol.1, no., pp.641-647, 14-16 Aug. 2009.

[6] Hao Liu; Arafa, Y.; Boldyreff, C.; Dastbaz, M., Cost-effective virtual world development for serious games, in Games Innovation Conference (IGIC), 2011 IEEE International , vol., no., pp.48-51, 2-3 Nov. 2011.

[7] Emam, A.; Mostafa, M.G., Using game level design as an applied method for Software Engineering education, in Computer Games (CGAMES), 2012 17th International Conference on , vol., no., pp.248-252, July 30 2012-Aug. 1 2012.

[8] Cowan, B.; Kapralos, B., Spatial sound rendering for dynamic virtual environments, in Digital Signal Processing (DSP), 2013 18th International Conference on , vol., no., pp.1-6, 1-3 July 2013.