3D Topological Learning Game: Framework
by
Christian Donner

CS 6910
Project Documentation

Academic Supervisor:
Dr. Guy Zimmerman

Summer 2011
Department of Computer Science
Bowling Green State University, Ohio
3D Topological Learning Game: Framework

Christian Donner
Supervisor: Dr. Guy Zimmerman
Dept. of Computer Science
Bowling Green State University, Ohio
Reporting date: 2011

1 Abstract

Learning games have been a part of our everyday life for a long period, ranging from crossword puzzles, Sudoku games to high performance flight simulators; all of them helping us to improve certain aspects in our lives. The improvement of technology always shows us new ways to enhance the learning experience. This paper will discuss the development of the framework of a 3D Topological Learning Game based on the jMonkey 3D technology. The task was to create a working game framework for the game designer to use for a prototype. In the future this prototype can be used to create a game to enhance the learning experience of geology students.
2 Table of Contents

1 Abstract ........................................................................................................................................ 3
2 Table of Contents .......................................................................................................................... 5
3 Introduction ..................................................................................................................................... 6
  3.1 Goal ........................................................................................................................................... 6
  3.2 The 3D framework: jMonkey framework .................................................................................. 6
    3.2.1 Game Framework needs analysis ...................................................................................... 6
    3.2.2 jMonkey framework ............................................................................................................ 6
    3.2.3 Other considered 3D frameworks ..................................................................................... 8
    3.2.4 Decision for jMonkey .......................................................................................................... 9
    3.2.5 Advantages of JME2 .......................................................................................................... 9
4 Project description .......................................................................................................................... 9
  4.1 Project Plan and Phases ............................................................................................................. 9
    4.1.1 Project Pan ........................................................................................................................ 9
    4.1.2 Definition Phase ............................................................................................................... 10
    4.1.3 Implementation Phase ...................................................................................................... 10
    4.1.4 Documentation Phase ....................................................................................................... 12
5 Future research ............................................................................................................................. 12
6 Conclusion ..................................................................................................................................... 13
7 References ..................................................................................................................................... 14
Appendix A: Project schedule ......................................................................................................... 15
Appendix B: Presentation ................................................................................................................ 16
Acknowledgments ............................................................................................................................ 27
3 Introduction

The idea for this game was adopted from the research carried out by the Leventhal, Klopfer, Onasch, Zimmerman DUE research group. Philipp Venningen and I were approached by Prof. Leventhal and Prof. Zimmerman to create a prototype for a game that has the potential to augment geology education. Philipp Venningen got the task to design the game play for this game and the task to build up a usable framework for the game play designer was assigned to me. The first step for this project was to find a suitable 3D framework to work with. After this my task was to create all the essential tools that the game play designer needed and create the surroundings for this game.

3.1 Goal

The overall goal was to create the game framework for a game to augment geology education with the two key themes as stated in Kirriemuir, J., & McFarlane, A Literature Review in Games and Learning [6]:

1. “The desire to harness the motivational power of games in order to ‘making learning fun’
2. A belief that ‘learning through doing’ in games such as simulations offers a powerful learning tool”

The game framework needed to be easy to use and to have a straightforward functionality.

3.2 The 3D framework: jMonkey framework

3.2.1 Game Framework needs analysis

The game framework had the following requirements:

• Interpretation and displaying of topographical maps in a 3D game environment
• Interface for game play engine
• Way points handling
• Creation of surroundings

The game framework was developed to be a set of tools for the game play engine that is created by the game play designer. It consists of an interface for the communication with the 3D framework and a basic tool to create all the surroundings of the game. The resulting game framework was defined as a middleware between the game play engine and the jMonkey framework.

3.2.2 jMonkey framework

The jMonkey framework is a so-called low-level 3D development tool for computer games. A low-level 3D development tool is a framework that consists basically of separate libraries that can be used freely to create a program. Through this there are no limitations in the use cases. It is
a Java based framework that has the possibility to use different renderers, in this case the LWJGL (Lightweight Java Game Library) by Puppy Games. The possibility to use different renderers makes it highly portable as there is a renderer for every operating system.

jMonkey is a so-called scene graph based rendering system. A scene graph is a grouping of nodes according to their spatial location. This approach of sorting objects has great advantages for implementing a game. First of all it is a great way to manage big amounts of data as the spatial location is always visible. The second reason is that when using this method there is always a spatial reference for every node. In addition most of the objects that are used in computer games are already arranged this way. For example a toe is the child node of a foot; the foot is a child node of the ankle, and so on. Furthermore you can easily translate this data structure in other data structures such as XML.

### 3.2.2.1 Basic Design of a jMonkey Game

The jMonkey framework is based on the classic game design consisting of four steps: (1) Initialization, (2) Update Game State, (3) Draw Scene, and (4) Cleanup.

![Diagram of Basic Design of a jMonkey Game](image)

In the beginning there needs to be an initialization phase as in most programs. In this phase all the basic objects are created and added to the scene. The first phase is the initialization phase where the basic objects are created and added to the scene. In addition all the base variables are set.

After the initialization are two phases that will be looped throughout the game. These phases are called the “update game state phase” and the “draw scene phase”. As the name implies the “update game state phase” is used to update all the necessary elements of the game. The position of the avatar and all the resulting variables are updated. The camera position needs to be updated
and according to the game rules the action handlers need to be executed. In addition the necessary animations need to be executed.

The second phase in this loop block is the “draw scene phase” it is used to display the scene using the displaying system. This phase is very simple in jMonkey due to the easy to use display system.

In the end there is the “cleanup phase”. This phase will be called when the game is terminated. Its purpose is to clean up all the used objects and variables before the game ends.

### 3.2.2.2 Display System

The display system provides traditionally all the communication between the actual program and the windowing system. This can be done through various renderers in jMonkey. The jMonkey framework supports a number of renderers e.g. LWJGL for Windows of JOGL for UNIX. This makes jMonkey a highly portable framework and the easy to use display system is also a big advantage for jMonkey.

The displaying system is mainly responsible for two main tasks. The first is to create the actual game window. The second task is to create the renderer. After this it supplies an easy way to communicate through the renderer with the OpenGL (Open Graphics Library). There were no limitations in terms of developing visible throughout the implementation phase of the project. The way to communicate with LWJGL through the display system is not just easy but very intuitive due to the simple API provided by jMonkey.

### 3.2.3 Other considered 3D frameworks

There were several other frameworks considered. All the other frameworks were based on C or C++ based. The other considered possibilities aside from jMonkey were:

- OpenSG (C++) [2]
- Open Scene Graph (C++) [3]
- Irrlicht engine (C++) [4]

OpenSG and Open Scene Graph were considered as they were mentioned and showed great features in a Wikipedia article [5] about 3D gaming frameworks. The Irrlicht engine is a German 3D framework that has the drawback of not supporting some Windows systems. OpenSG was also not seen as suitable as it is mainly focused on virtual realities and it is based on C++. With the same reasoning Open Scene Graph was also not considered as the base framework. The problem with C++ that the team members saw was that there might come up problems with operations like file handling. The essential drawback of C++ was that problems with certain operations such as file handling could have occurred during the implementation process. These operations are not always executed the same way on different operating systems and during this phase it was not sure what would be needed.
3.2.4 Decision for jMonkey

jMonkey is a Java based low level development tool. The fact that it is JAVA based was very appealing for portability. Other programming languages might have some limitations to portability, which would have been a disadvantage in the future, e.g. file handling which will be used in a future stage for inserting the questions. As jMonkey is a Java based framework and Java handles these operations through the virtual machine this was seen as a big advantage for the framework. In addition both team members were very familiar on working with Java.

The second reason was that jMonkey provides a wide collection of tutorials and a very complete documentation, which makes it easy to work with. In addition the easy to use display system was very appealing and the fact that the framework is free was also taken into consideration.

3.2.5 Advantages of JME2

The reason why JME2 was chosen and not the more developed JME3 engine was that it is still in the developing phase and just an alpha version is currently available. The other essential reason was that JME3 only supports OpenGL2 and devices that use OpenGL cannot be used. As the possibility to switch to a netbook or another mobile device that only use OpenGL was very appealing. In addition the documentation of JME3 was not complete at the beginning of the project.

4 Project description

In this section we will discuss the project development and the different phases in detail. This section of the paper will be about the project development and the separate phases. It will describe the different steps that were taken to get a working prototype of a 3D game. In addition to this it will also describe the communication between the game play designer and the framework engineer.

4.1 Project Plan and Phases

4.1.1 Project Pan

The project plan itself was created in cooperation with Philipp Venningen. It was decided to develop it in three separate stages (see Appendix A: Project schedule), namely the definition, implementation, and documentation phase. The tasks of the separate stages were defined in weekly meetings. The project plan was created in a retrograde fashion. The goal and the submission week were defined. After that all the different steps that would lead to the goal were defined. In order to get a valid plan all preconditions for the separate steps needed to be defined. After creating these steps the required duration to finish the task was defined with a tolerance for
bugs in the implementation and they were added to the project plan. Furthermore some buffer slots for unexpected events were defined and inserted.

The goal was to have a working prototype of an educational game for augmentation of geology education by the end of July. Due to the strict separation of the two fields of work every team member was able to create a solution for a given problem on his own. The only cross over were the preconditions for the separate tasks but due to the predefined project plan no collisions or deadlocks occurred.

### 4.1.2 Definition Phase

The first step was to define the several tasks for each team member to handle. At first the framework that will be used later was selected. jMonkey was selected as it is a JAVA based low level development tool. The reasons for using version 2 of the framework and not the more advanced version 3 were already described before. In JME3 the displaying system is limited to OpenGL2, which does not work on several mobile devices like netbooks. Another reason against JME3 was that it was still in the alpha phase and due to the availability of JME2 the risk of running into bugs was considered as critical with JME3. However, the main reason was that the documentation was just not complete at the time the decision had to be taken. JME2 provided a nearly complete documentation with a wide range of tutorials.

The next step was to define all the needed interfaces. This was a very important step as it defined all the following implementation work. It was decided to create a so-called “game handler” class which was used to mainly control the game. Within this class all the necessary implementation for the actions the game play designer would need to control the game should be created. It should be an easy to use tool that just provides a simple API for the game play designer.

The next big class to be created was the class controlling the avatar. It should provide the basic physical attributes of the avatar, which should be controllable from the outside. In addition it should include the basic physic movement implementation.

In addition to the previous mentioned definitions it was also necessary to create the surroundings, which did not influence the game play.

### 4.1.3 Implementation Phase

Throughout the implementation phase it was very important not only to consult the jMonkey documentation [1] but also to get up to speed again with the basic mathematical concepts of space. In other words the basic calculus procedures for vectors had to be refreshed. In addition the previously unknown concept of quaternion had to be discovered and used. This tool provides the possibility to rotate objects on three different axes, so this is a necessary tool in 3D programming.

The first important class that was developed was the game handler. This class is the central command center for the whole game. It includes all the necessary functionality for the game play
designer to use. In the game handler class, called BGHandler, the key bindings that were previously defined by the game play designer were created and matched to the corresponding actions that control the avatar. Furthermore, the additional key bindings for the answers are created and the “exit” and “map” key binding are created.

After creating the basic handler class the surroundings of the game were the next to be handled. The first objective was to create a horizon and to give the game a more natural feeling. The horizon or skybox is just a graphic spanned around the map with a sky. This makes the view more natural compared to the black background jMonkey provides.

Furthermore the lighting needed to be created. This part may be very hard in different 3D frameworks but it was quite easy in jMonkey. In jMonkey the light source needs to be defined and jMonkey creates the lighting.

In addition to the lighting the camera needed to be implemented. The camera is defined as a chase camera, which can be moved with the mouse. jMonkey again has this type of camera already implemented in the framework and one just needs to place the camera in the correct position corresponding to the avatar object.

As a third surrounding element an object for the checkpoints had to be created. The game play designer asked for a box like object that he would be able to place freely on the map. A box in a suitable size has been created with the functionality to let it appear in any place within the map. For testing purposes the box has the additional functionality to be placed randomly on the map.

The challenging part was the creation of a 3D terrain after creating these essential surrounding elements was at hand. jMonkey supports three different types of maps that can be used to create a 3D terrain:

1. random map
2. gray scale map
3. height values in an array

For testing purposes a simple random map was generated. This map had no texture and was inserted so the game play designer was able to test his changes. The next step was to make a decision which kind of map to use. The game play designer made this decision as he was responsible to actually create the map and the textures. It was decided to create the map using a gray scale map as it made it possible to create such a map from an actual satellite image, which is put in gray scale and then inverted.

The next challenging step was to create the human like avatar. The aim was to create the avatar with the basic attributes of a real-life human being. This avatar needed to be created with the basic attributes that a real life human also has. The main focus was on the actual movement of the character. In order to implement a movement it was necessary to define a parameter for the velocity. This parameter needed to be of the style

\[ V = \frac{D}{T} \]

Which stands for velocity is equal to distance over time. The measuring unit for distance is a normalized vector in the coordinate system. Hence, the velocity is defined in this case as normalized vectors over second. Further attributes had to be defined besides the velocity. After defining the velocity there needed to be some more attributes defined. The change in velocity is defined either as acceleration if it is positive or breaking in the case that it is negative.
Specifically, the acceleration is the rate of change of velocity which is defined in physics as the change of velocity over time. Both values for acceleration and breaking will be variables changeable from outside through function calls. The maximum speed, maximum turn speed, and weight are the last variables needed for this class. The maximum speed will be used as a limitation for the acceleration. The maximum turn speed is used as a limitation for the rotation of the character and the weight will be used for the drift action which will be called upon when the player stops pressing the move forward key. The last needed object is the actual model which is used to display the avatar. All the variables can be manipulated from the outside.

After defining and creating the variables to control the avatar there is a need to call the animations that were predefined by the gameplay designer. There are basically two steps in creating this animation and they will be explained with creating the “walk” animation. The first basic step is to change the location of the avatar in space, which is defined by the velocity. The second step is to create a MeshAnimationController. With this MeshAnimationController all the available animations are loaded. In our case we just need the walk animation defined by the gameplay designer. To execute the animation an AnimationChannel must be created and all the necessary bones defined by the game play designer have to be added and with this AnimationChannel the actual “walk” can be called.

The last very important step in the implementation phase was the code cleanup. The review of the code showed some unused elements that were left over during implementation changes. During this phase there is a need to check again the complete implementation to discover some illogical or false implementations.

4.1.4 Documentation Phase

The documentation phase mainly consisted of creating the presentation (see Appendix B: Presentation), and the project report. Furthermore the planning of the presentation took also place during this phase. As both Philipp Venningen and I were not residing in Bowling Green, Ohio, the project presentation needed to be done remotely via video conferencing. Sony DADC Austria AG supported this by letting us use one of their video conference rooms in their main facility in Anif, Salzburg, Austria.

5 Future research

As a proposal for future work the first task should be to port the program to JME3. During the project JME3 documentation increased and the drawback that OpenGL2 as the only available 3D graphic display engine decreased through several driver updates that added OpenGL2 support to graphic chipsets. If the situation would have been like this at the beginning of the project, JME3 would have been chosen. In addition JME3 provides a lot of additional tools like a map creator. This would be a great opportunity for the game play designer as it provides a lot of freedom in this case.
In addition a detailed research about the usability of the game would bring new conclusions that could lead to an improvement of the game. Furthermore the implementation of the usage of several Avatars might give the game an additional appeal. Most of these tasks would need to be rather done by a designer than by a software engineer.

The implementation of sounds should also be looked considered as after some tries this element was not considered anymore due to the fact that the JME2 audio player had some unusual behavior. However, in JME3 the audio player seems to be operating as it should. This would also be a possibility to create a cross over project with the College of Musical Arts.

6 Conclusion

The goal was to create a prototype in the time period between March and July 2011. The framework was intended to be an easy way to create the communication from the game play design to the actual 3D framework. In the end a playable game prototype was created and the goal of creating the rudimentary framework between the game play designer and jMonkey has been created. This project showed that programming such a framework is not a standard software engineering project. It showed that a lot of mathematical concepts are used that are not normally applied in a standard software project. The development itself had a lot of challenges. One of the biggest was the communication between the game play designer and the framework engineer. A lot of assumptions were made which resulted in reprogramming some parts of the code as the needed functionality was not implemented. Another challenge was to create the preconditions of some parts of the game play design. The programming itself was more difficult as expected as it required an understanding of the 3D framework and most of its functionality. This was aside from actually creating the framework one of the biggest parts of the project. Overall the result of the project is a valuable basis to develop a learning game to augment geology education.
7 References

[1] jMonkey Documentation:  

[2] OpenSG  
http://www.opensg.org/

[3] Open Scene Graph  
http://www.openscenegraph.org

[4] Irrlicht engine:  
http://irrlicht.sourceforge.net/
