Lightsaber Simulator Project

Miro Lahdenmäki
55089K
mlahdenm@cc.hut.fi

Esa Nuuros
54558L
enuuros@cc.hut.fi

Marko Luukkainen
55489L
msluukka@cc.hut.fi

23rd January 2005

Abstract

Our goal was to create a lightsaber simulator as an exercise project for the course T-111.400 Virtual Reality. Our application features mind blowing graphics, intuitive interaction and 3D sounds.

1 Introduction

The purpose of this project was to create a virtual reality application for the course T-111.400 Virtual Reality. This was done for a real VR environment to get a better view of actual software development in the field of virtual reality. The application runs in the CAVE-like environment of the Telecommunications Software and Multimedia Laboratory at the Helsinki University of Technology.

There were three requirements for the exercise.

- 1. Basic graphics
- 2. Basic interaction
- 3. Sounds

Fulfilling these basic requirements was mandatory. To achieve these objectives the application had to run. More sophisticated features were needed

for a better grade. Such features included: better graphics with textures, lights and shadows, an improved sound scene, a more realistic physics model and improved interaction.

Next we describe the application in detail and represent a UML class chart.

2 Functionality

The application was built iteratively. First the lightsaber was implemented. It is controlled using a wand since it is close to an actual sabre handle. The lightsaber has a glow around it.

It was planned that the lightsaber would come out of its hilt when a button was pressed on the wand. (Or if the button on the wand didn't work, we could have used the glove on the other hand for control: fist = blade in, hand open => blade out, by default the sabre would always be out). This feature was not considered important and was left out.

Next some sounds were to be integrated to the system. It was planned that the lightsaber woud make a sound as it moved. The faster the saber would move the louder the sound was going to be. The sound would come from the direction of the lightsaber. Due to some problems with the sound system that are discussed later we didn't implement a changing sound but a constant humming sound instead.

A practice droid would next be implemented that would hover and circle around the avatar. It was planned to make a hovering sound which was left out to make other sounds easier to perceive. It would shoot slow moving laser beams directed towards the avatar. It would be shaped like a sphere and parts of its surface material would glow.

The player would be able to deflect the laser beams by directing the lightsaber close enough to an incoming laser beam. A successfull block would increase the experience points of the avatar. A hit would decrease the hit points of the avatar. A hit would be recorded when a beam got inside a virtual cylinder that surrounded the position of the goggles. Experience points were to be shown in the upper right corner of the view while the hit points were to be shown in the upper left corner. In the final version both the experience points and the hit points were shown in the upper left corner of the screen. In addition a game over text appears after the player's hit points are up and a counter informs the player when the next game will begin.

The deflecting area of the lightsaber would be modeled as a cuboid and the laser beam as a simple line segment.

The surroundings in the holo deck would be textured to add a star wars feel to the simulation.

Additional features were planned in case we would have time to implement them:

- Droid will cast shadows on the walls.
- The user may select the color of the lightsaber.
- User can push the hovering droid away with his left hand.
- Better-looking environment where the avatar moves.
- Better moving algorithm for the droid.

3 UML Class Chart

Figure 1 shows the initial sketch of the class diagram.

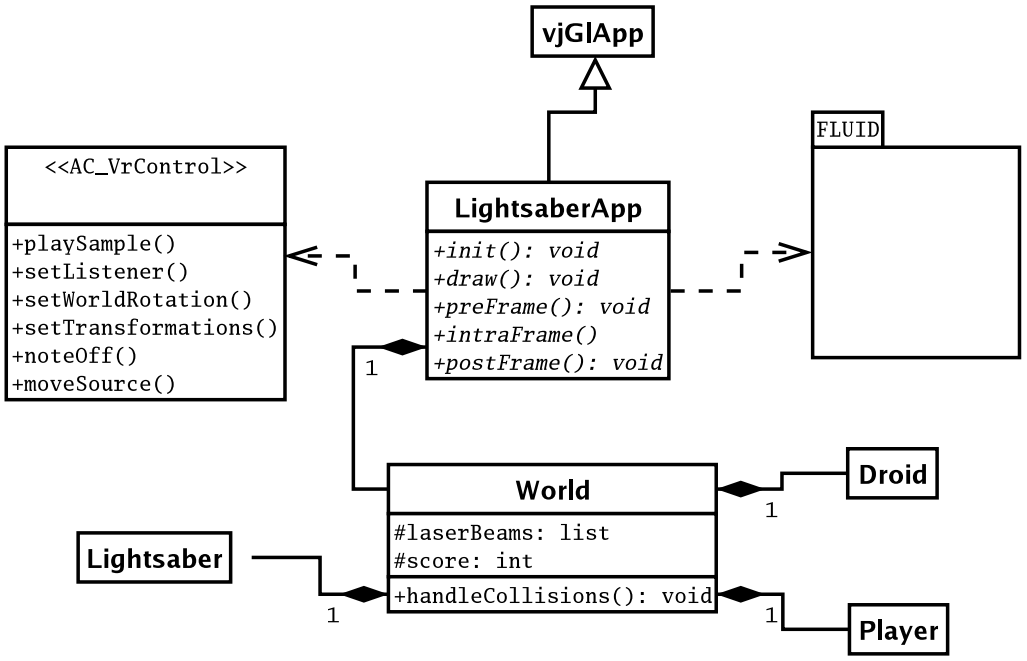

Figure 1: Class diagram

4 Schedule

A rough estimate on when features should be working was planned on a basic level.

- **Week 42:** Some basic VR application running using the given libraries
- **Week 43:** Basic lightsaber working
- **Week 44:** Light sabre sounds
- **Week 45:** Practice droid + sounds
- **Week 46:** Collision detection + sounds
- **Week 47:** Fine-tuning for the porting session
- **Week 48:** First porting session
- **Week 49:** Fixing issues found in porting session
- **Week 2:** Final touches
- **Week 3:** Second porting session
- **Week 4:** Final report

5 Results

We implemented all the mandatory requirements: the application has graphics, sound effects, and interaction.

The graphics include a room the player is situated in. The room contains a platform which the player stands on and fights the droid. The droid is hovering above the chasm and the player has his lightsaber displayed in front of him. The graphics also include the laser beams the droid shoots. We used textures and light sources in the scene to add to the visuals.

As additional graphical features that we didn't include in the project plan we implemented a trailing effect for the lightsaber and explosions/smoke effects for the laser hits. We also had a score counter displayed in the HUD. We decided not to implement the shadows because our scene was built in a way that the shadows would not have contributed to it. Only the droid could have cast a shadow and it would have always been cast on a wall. Probably the user wouldn't have even noticed it.

For sound effects we had a blaster sound for the droid when it shoots at the player. It was positioned at the droid's location. The saber makes a humming sound, but we didn't figure out a way to implement the change in pitch you here in the movies when the saber is whirled in the air. Blocking the laser beams with the saber also produces a sound effect that is positioned at the location of the impact point. We also had explosion sounds for the laser beams as they hit the platform or the walls of the room. These are also positioned correctly. One reason why we didn't implement the change in lightsaber pitch was because we couldn't get Mustajuuri to produce any sound on our own computers and there was no time to add features in the porting sessions.

As for interaction, the player can move around and control his lightsaber. The player can deflect the laser beams shot by the droid by moving his saber in front of the beam or dodge the beams by moving out of the their way.

6 Time usage

We mostly worked at the same time, everyone at their own computer and communicating with each other via IRC. The time usage per head went roughly as follows:

In the first part we designed the application and wrote the project plan. This took around 5 hours.

In the second part we started by setting up the programming environment for our own machines by compiling the various libraries for different machines. This took about 5 hours.

Then we wrote the basic skeleton for the application and tested it in the first porting session in EVE. We spent 3 hours in the porting session (we got one hour extra for being the first group and having to go confront the initial difficulties). The writing of the basic application took 10 hours.

After the porting we had time to fix the problems found and more or less finish the application for the second porting and demo session. The fixing and finishing of the application along with creating the models, textures and sound effects took around 15 hours. We spent 2 hours in the second porting session in EVE.

So we spent around 40 hours each for the project part of the course.

7 Problems we had

Mustajuuri provided some problems regarding the sound effects. There was no documentation of the used classes (we had to read the source). We also didn't get it to work in linux so we could've tested our sounds before the EVE porting sessions. This prevented us from implementing the change in the pitch of the humming sound of the lightsaber. The library compiled ok, but it just didn't produce any sound.

We also had problems with the various config files for both VRJuggler and Fluid. In the first porting session we couldn't get the image displayed on the CAVE walls because there wasn't a valid VRJuggler configuration file available. In the second porting session we had a problem where when the player moved the surroundings would move too much in relation to the player. We didn't have time to fix this in the porting session.

Fluid configuration files also changed during the project and we had problems in the second porting session with this. We had the old fluid configuration file and the application just crashed without any error message. Some basic error checking in the Fluid library would be appreciated. Fluid also didn't compile cleanly in EVE. We had to insert various preprocessor definitions in our own code in order to compile it.

8 Course feedback

Judging from the time we used for the project, the amount of credits you get for it is correct. We had the advantage that all of us are interested in computer graphics as a hobby and we had lots of code we wrote before the project we could just take and use.

Iikka Olli was very helpful and knew the environment in EVE. This really helped in tracking the problems in the porting sessions.