PA11: Wizard Duel

By Team Tessellation
Natalie Arnold
Terra Williams
Kurtis Rodrigue

December 17, 2018

Table of Contents

Sections

Overview

Dependencies, Building, and Running

Dependency Instructions

For both of the operating systems to run this project installation of these three programs are required GLEW , GLM , and SDL2 . (http://glew.sourceforge.net/) (http://gl m.g-truc.net/0.9.7/index.html) (http://glm.g-truc.net/0.9.7/index.html)
This project uses OpenGL 3.3. Some computers, such as virtual machines in the ECC, cannot run this version. In in order to run OpenGL 2.7 follow the instructions at Using OpenGL 2.7 (https://github.com/HPC-Vis/computer-graphics/wiki/Using-OpenGL-2.7)

This project uses the Assimp library. To download and install Assimp . (http://www.assimp.org/)

This project uses the ImageMagick 7.0.8. To download, install, and build for linux, follow the instructions at ImageMagick . (https://linuxconfig.org/how-to-install-imagemagick-7-on-ubuntu-18-04-linux)

This project uses the Bullet Physics Library, installation instructions can be found here for Bullet . (https://github.com/bulletphysics/bullet3)

Ubuntu/Linux

sudo apt-get install libglew-dev libglm-dev libsdl2-dev
Mac OSX
Installation of brew is suggested to easily install the libs. Ensure that the latest version of the
Developer Tools is installed.
brew install glew glm sdl2

User Manual

Building and Running

To build this project there are two options. One is to use CMake which makes including new libraries easier, and handles new files added automatically to the src and include directory. CMake is a small new learning curve but makes things easier in the future. The second option is to use the provided Makefile which is used as usual. Running the make in a separate directory will allow easy cleanup of the build data, and an easy way to prevent unnecessary data to be added to the git repository.

CMake Instructions

The building of the project is done using CMake, installation with apt-get or brew may be necessary. Later use with CMake and Shader files will be require the copy of a directory where those files are stored (ex. shaders). To do this in the a dd_custom_target function place

COMMAND ${CMAKE_COMMAND} -E copy_directory
${PROJECT_SOURCE_DIR}/shaders/
${CMAKE_CURRENT_BINARY_DIR}/shaders
mkdir build
cd build
cmake ..
make
./Wizard_Duel

Makefile Instructions

The makefile works as expected and must be updated with new files added in.

```
mkdir build
cd build
cp ../makefile  .
make
./Wizard_Duel
```

Ubuntu.cse.unr.edu5 OpenGL 3.3 will run on the ubuntu.cse.unr.edu website. To do so follow the build instructions, but when running the Tutorial executable use this line to execute.

```
/usr/NX/scripts/vgl/vglrun ./Wizard_Duel
```

Controls

W - Move Forward
A - Strafe Left
S - Move Backward
D - Strafe Right
E - Shoot Spell

Figures

Figure 1. Map from Main Menu with Day Skybox

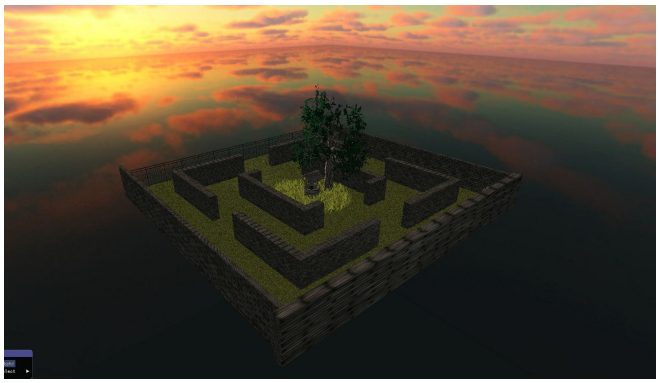

Figure 2. Map from Main Menu with Sunset Skybox

Figure 3. The evil wizard chases the player down a corridor.

Figure 4. Middle of the battle field day skybox

Figure 5. Middle of the battle field sunset skybox

Technical Manual

Issues

We struggled to get sound co-operating and encountered a strange problem with the spell shooting sound effect. After firing a spell 15 times the effect would simply stop loading and we were never able to figure out why. We also struggled to get background music to loop through multiple games which was one of many variables affecting our ability to make the game playable for multiple rounds. We wanted to have a health bar above the enemy sprite that shrank when it's health went down but unfortunately scaling the bar gave us some undefined behavior and we had to go without it. The tree model we found for our map is a little high-poly making it difficult for the game to run when streaming to the projector. We ran out of time to implement the game as multi-player and again, did not make the game playable for multiple rounds.

Things We Would Have Done Differently

The more obvious things include implementing split-screen and having some sort of HUD for the player. We had normal mapping working (for positive z facing models) and decided that the resulting look was not worth piecing through the math of tangent spaces given how much time we had. The spells would have been much more efficient if they had been rendered as individual instances but we ran out of time researching the OpenGL drawInstance functions. We would have liked to implement a couple more kinds of spells. We wanted to debug game replay but ran out of time and nicer menus would have been a must. We also would have liked to clean up our general code structure as, once again, all semblance of object oriented practices have been mercilessly eradicated.