# DUNGEON_DIVER.EXE

By Brianna Blain-Castelli,

Alexander “Sasha” Kharag,

And Ryan Fox

# Table of Contents

# Dependencies

## Dependency Instructions

For both of the operating systems to run this project installation of these three programs are required GLEW, GLM, and SDL2.

This project uses OpenGL 3.3. Some computers, such as virtual machines in the ECC, can not run this version. In in order to run OpenGL 2.7 follow the instructions at Using OpenGL 2.7

## 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

## 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. 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 add_custom_target function

COMMAND ${CMAKE_COMMAND} -E copy_directory ${PROJECT_SOURCE_DIR}/shaders/

${CMAKE_CURRENT_BINARY_DIR}/shaders

mkdir build

cd build

cmake ..

make

./Tutorial

## Makefile Instructions

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

mkdir build

cd build

cp ../makefile .

make

./Tutorial

## Ubuntu.cse.unr.edu

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 ./Tutorial

# Interaction

Standard WS Movement – Forward and Backwards respectively in the direction you are facing.

Left and Right arrow – Rotates the camera in the desired direction not inverted.

R – Raises and lowers the lantern.

F – Switches the game to per fragment lighting (fragment is the default lighting at game start).

V – Switches the game to per vertex lighting.

G and B – Increases and Decreases the brightness of the Ambient lighting respectively.

H and N – Increases and Decreases the brightness of the Specular lighting respectively.

# How to find the ball pit

After entering the maze take the first right. Turn right at the T junction. Turn left. Take the first right. Turn left. Then head straight and you will find paradise.

# Challenges Faced besides my crippling depression

Smaller issues that we faced included fixing our frankly broken lighting shaders, getting multiple light sources to work correctly, trying to locate the “delete material” button in blender after accidentally adding a multiple to the obj then saving it, and after discovering that said button does not exist, remaking the entire labyrinth again from scratch.

Remaking our labyrinth was an important challenge that presented itself because we discovered that scaling to a certain degree makes collisions fail spectacularly.

The greatest obstacle we encountered was getting the movement of our character to react properly to the inputs that we gave it. This included trying to get the camera to move in the correct direction. Followed swiftly by hours of math to get the movement to go in the direction the camera was ACTULLY FACING. Also fixing the movement when rotating in the “negative” direction by checking if the angle was less than pi and flipping the controls.