# CSC5210_FinalProject

---

First-Person Cliff Diving Game.

User will be able to run and jump and perform flips. If the user lands in the water incorrectly an ambient light will make the screen glow red. The scene will include point lights that glow red when the user is unable to jump and green when it is okay to jump.

---

Project source code can be downloaded from
https://github.com/MC-CSC5210/CSC5210_FinalPoject.git

---

## Author and Contributor List

John Wyeth

Thomas Kelley

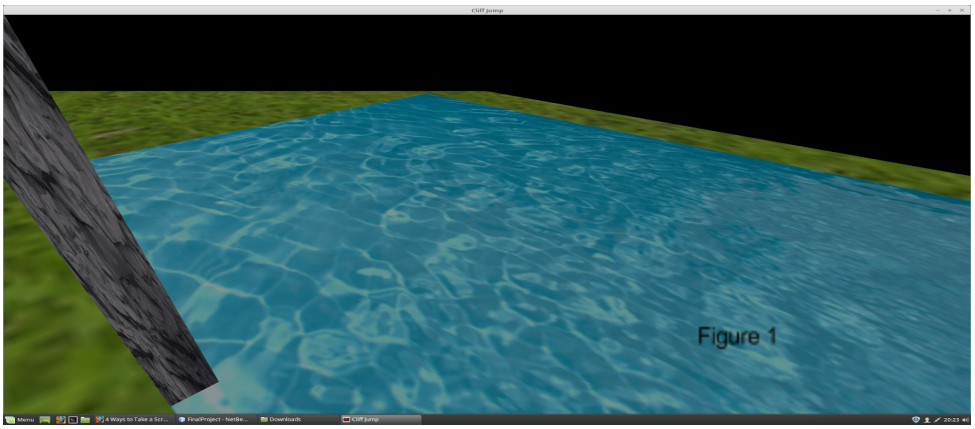
Pat Langille

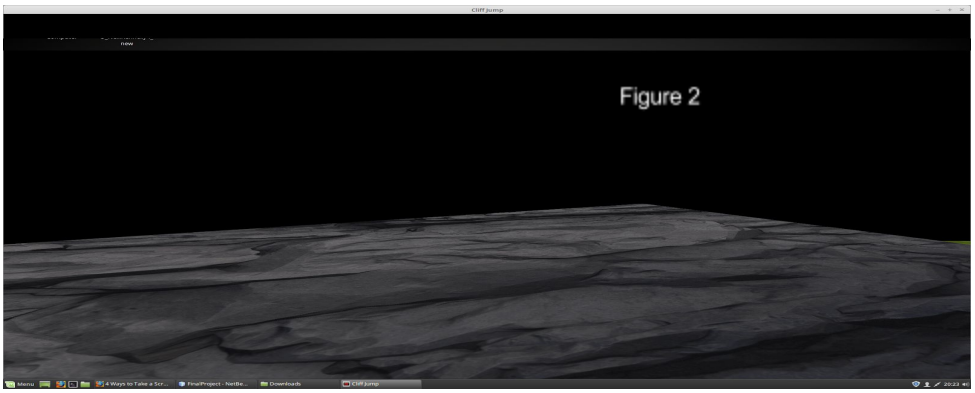
---

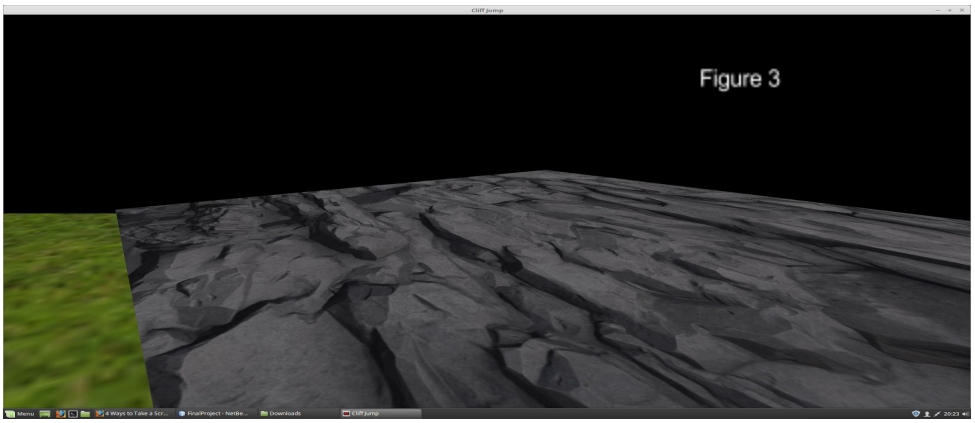
## Terrain:

The terrain consists of a number of textured Rectangular Prisms. The prisms are made up of 6 rectangles whose surfaces have geometry added by a geometry shader. Below you can see

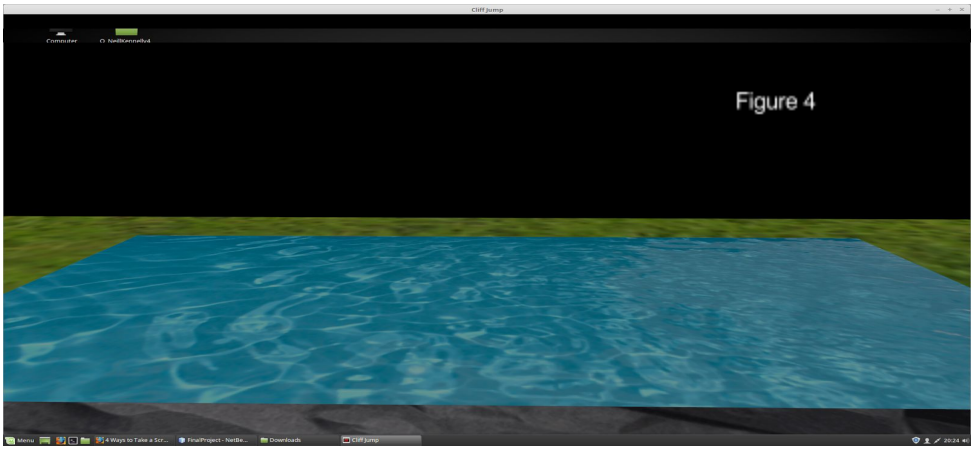
the basic terrain with textured models:

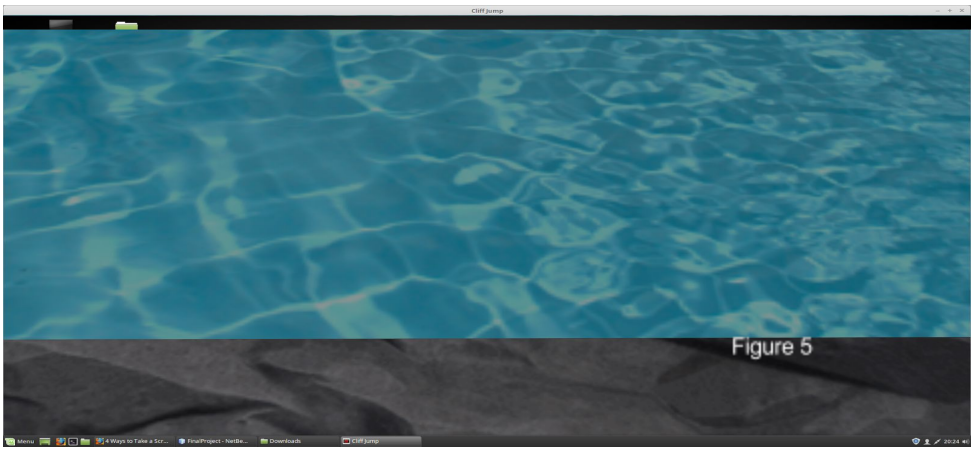





Figures 2 and 3:  Top of  cliff prior to the addition of the lights and sky box







Figures 4, 5, and 6: Top of cliff looking down at the water

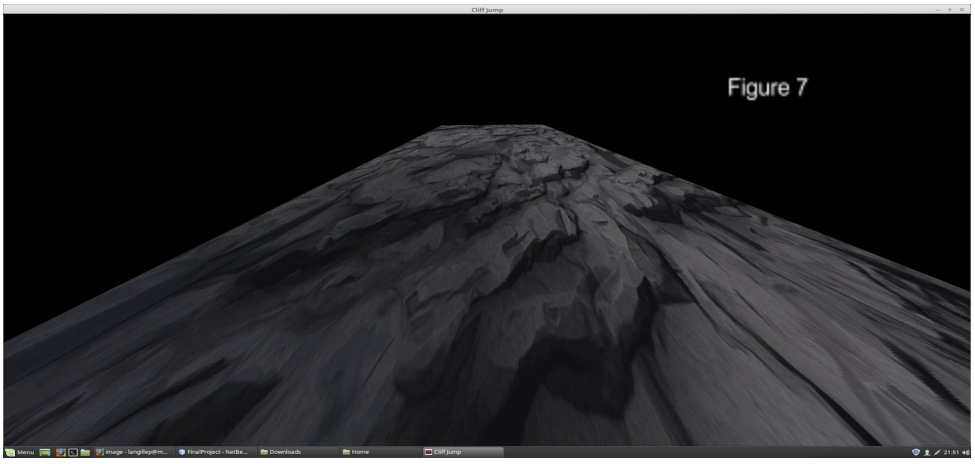



Figure 8: Looking up at cliff prior to cloud and sky box

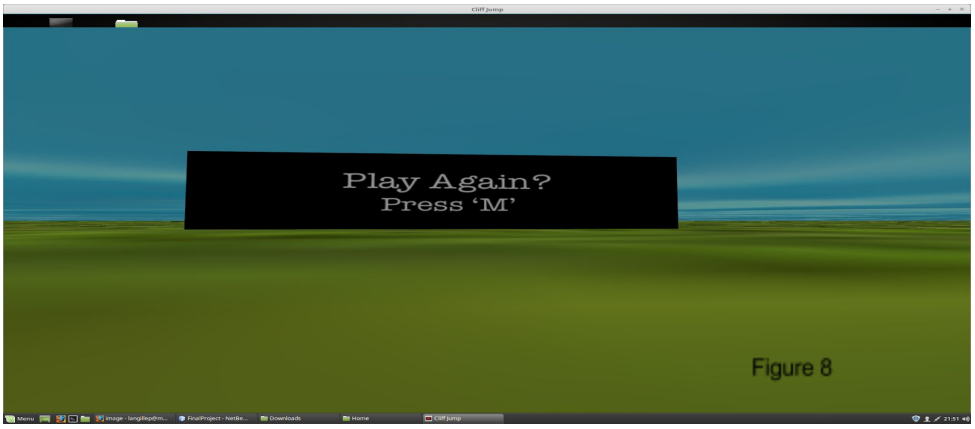
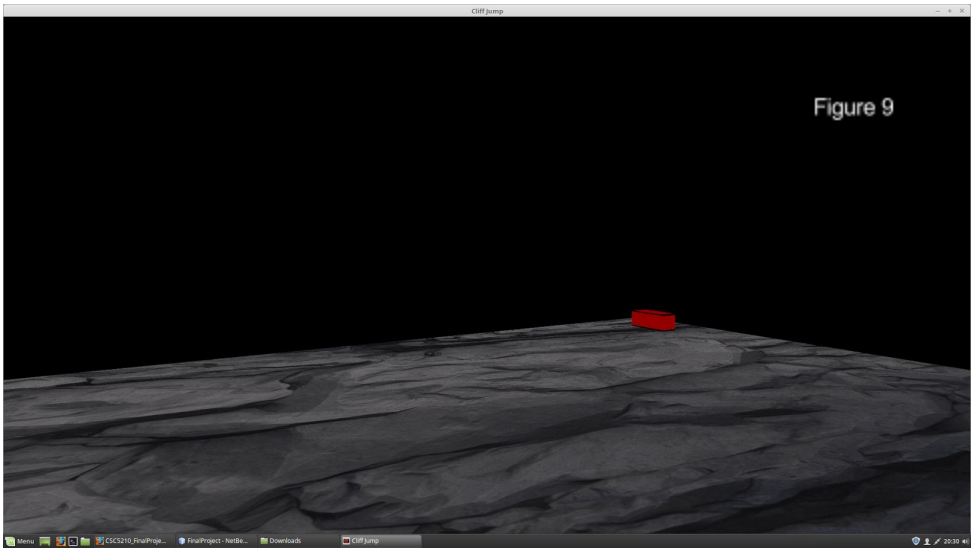
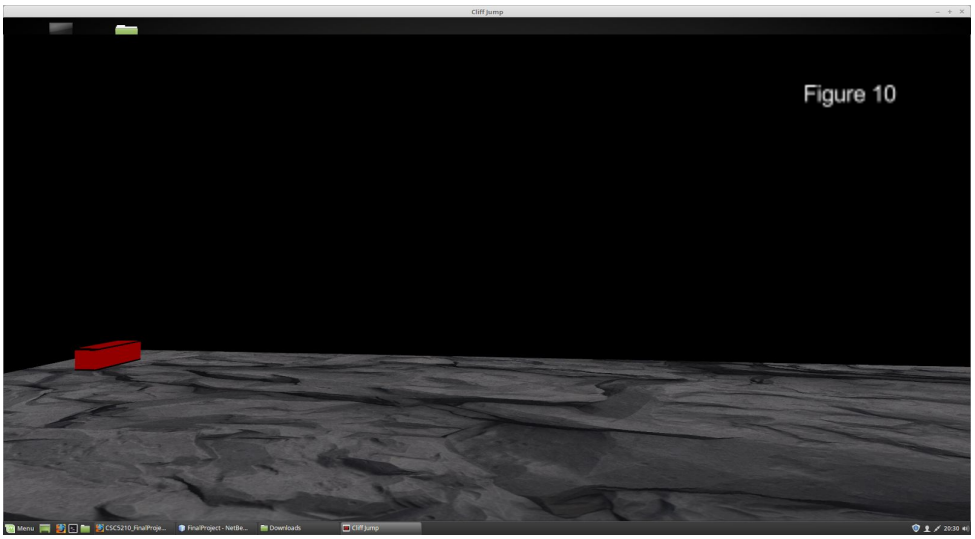
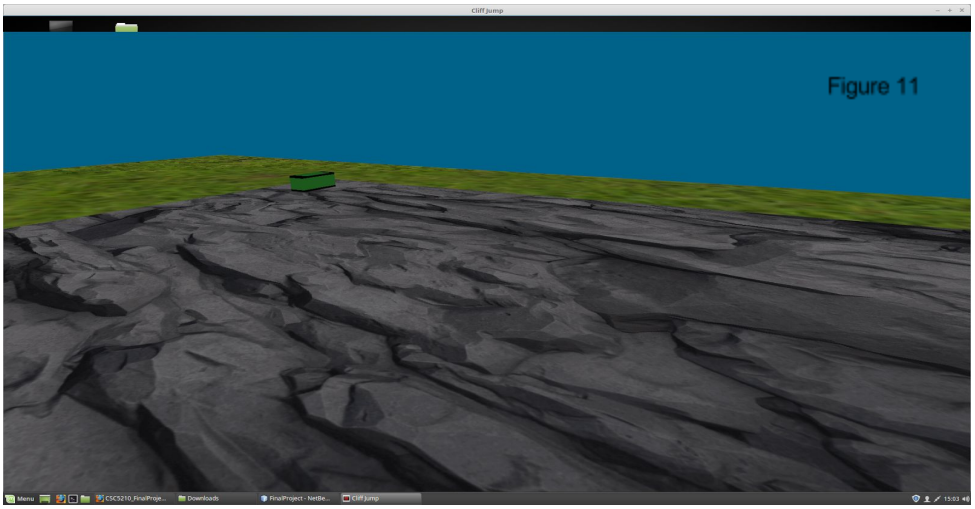


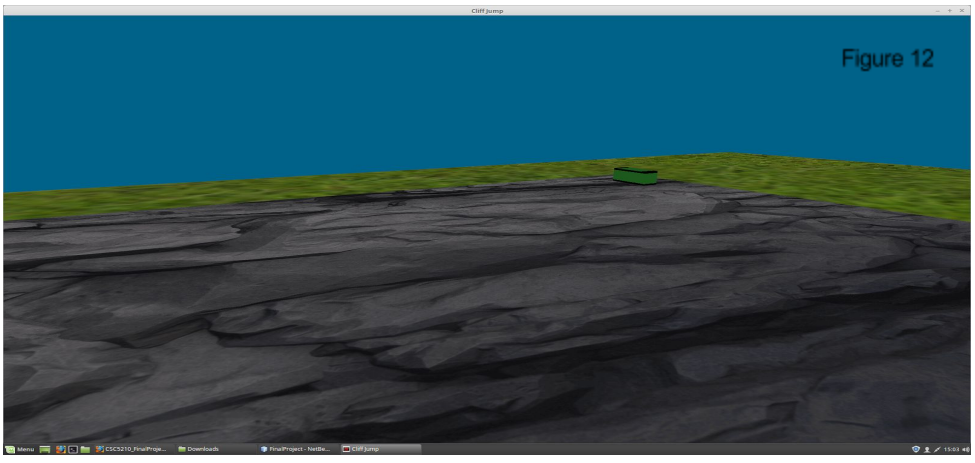

Figure 9: Rectangle added to prompt user for another jump

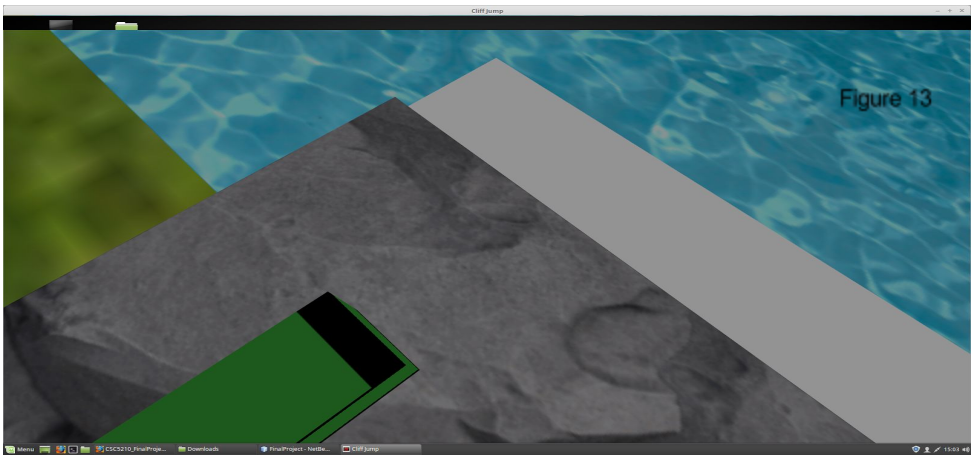
We then added two lights at the top of the cliff that indicate when the user can jump. The lights are changed from red to green with the press of the 'J' key which also initiates the run to the edge of the cliff. Once the user has jumped from the cliff the lights will be reset to red. This is shown in figures 9-13:

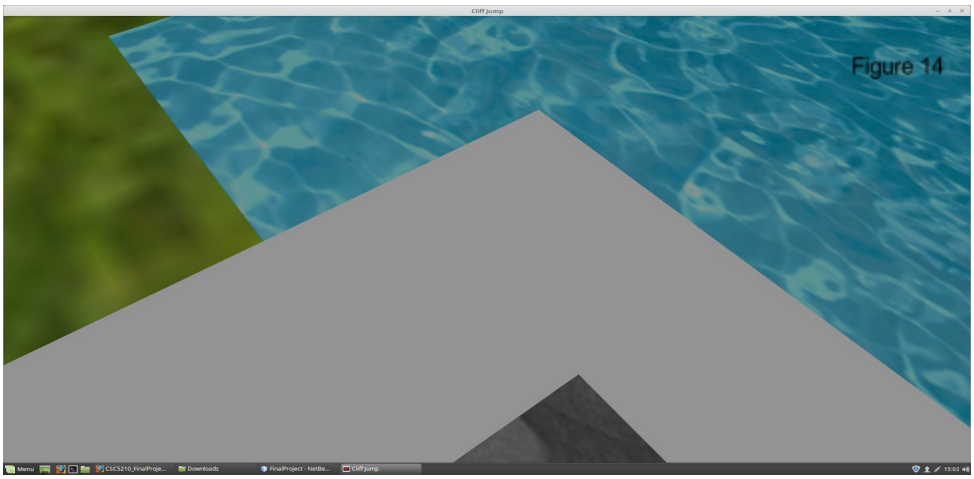



Cliff Jump
Figure 10

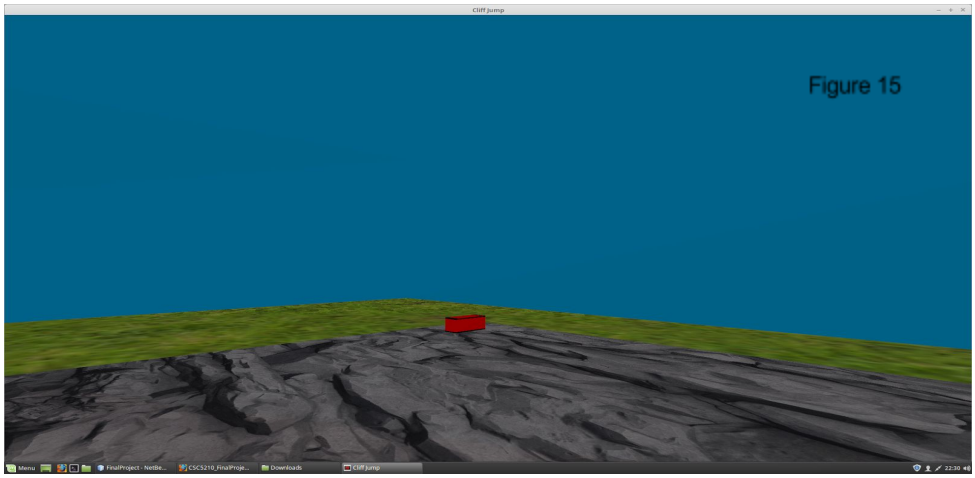

Cliff Jump
Figure 11

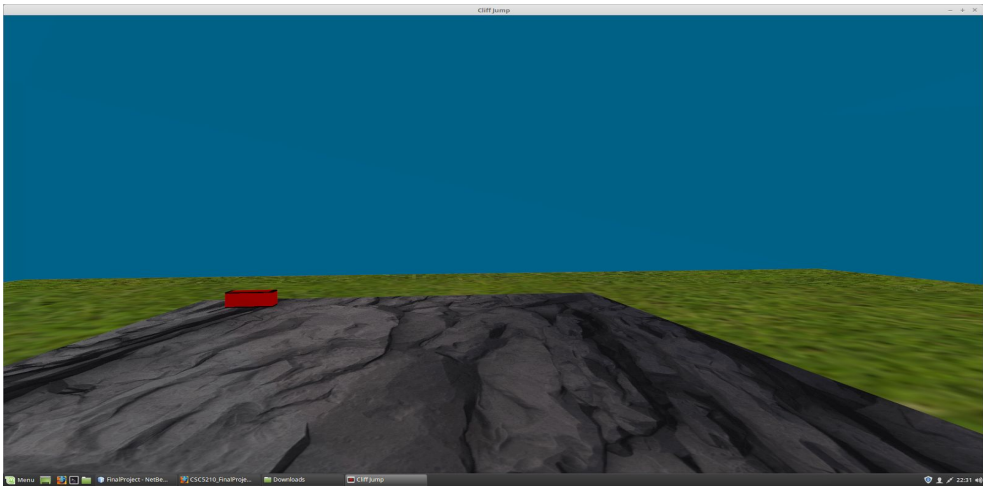

Cliff Jump
Figure 12




Figures 13 and 14 show the addition of the base geometry of the cloud which would later be modified by a geometry shader. A geometry shader is a shader program written in GLSL that governs the processing of OpenGL primitives. This is useful for making surfaces have more realistic geometries.





Figures 15 and 16 show the addition of Directional Light to the scene. Directional Light is lighting that hits the scene at the same angle much like the sun.

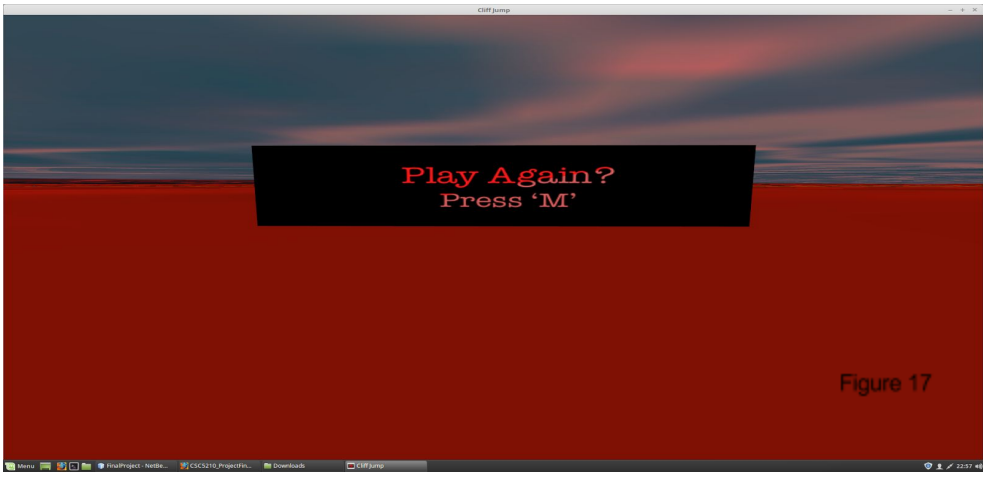



Figure 17

Figure 17 shows the addition of a red ambient light. This occurs when the user performs a poor landing into the water. This requires the use of the camera's view vector at the time of impact.

## First-Person View:

- 'F' rotates the camera to simulate a flip
- 'R' rotates the camera to simulate a backflip
- ‘J’ allows the user to run to the edge of the cliff
- ‘Space’ translates the user up in the y-plane and out off of the cliff, this triggers the gravity simulation
- camera is translated down toward the water based on simulated gravity calculations

## Lighting and Shaders:

- scene should be lit using ambient light (turns red as a result of a bad dive)
- scene should be lit using directional light from above the scene(like the sun)/will be toggled on and off using 'T' (maybe)
- scene should have two point lights at the top of the cliff which are red prior to the jump and green when the user can jump

## How to Use

1. 'J' initiates the run leading to the jump and the two lights at the top of the cliff will turn from red to green
2. 'Space' allows the user to jump from the cliff if they are in a given range along the z-axis
3. 'F' allows the user to perform a front-flip while falling toward the water
4. 'R' allows the user to perform a backflip while falling toward the water
5. Following a jump the user prompted, using a textured rectangle, if they would like to jump again. The user may select the ‘M’ key to place themselves back on the cliff.

---

# Challenges

1. Water Collision Detection and Ambient Light

Detecting a poor water entry relied on the camera's view vector. It require some playing around with the camera angle to find a range that would most likely represent a good landing versus a bad one. Once we had figured this out, we simply added a red ambient light once the poor landing requirements were met and then removed it when the user selects 'M'.

2. Gravity/Acceleration

This is difficult because the equations for velocity and acceleration require time, and because different operating systems have different clocks a timer can be difficult to implement. This required us to calculate antiderivatives in order to find the camera's position at time t.

3. Geometry Shader for the Cliff

We attempted to include a geometry shader for the cliff and the cloud, but unfortunately we were not able to successfully implement this type of shader. A geometry shader would have added extra geometry to our scene making the cliff look more jagged and the cloud a little more realistic.

4. Directional and Point Lighting

Implementing the directional light was a challenge. It was difficult to find the normals of each vertex within the scene. We decided that we would focus on calculating the normals of the surfaces rather than the individual vertices.