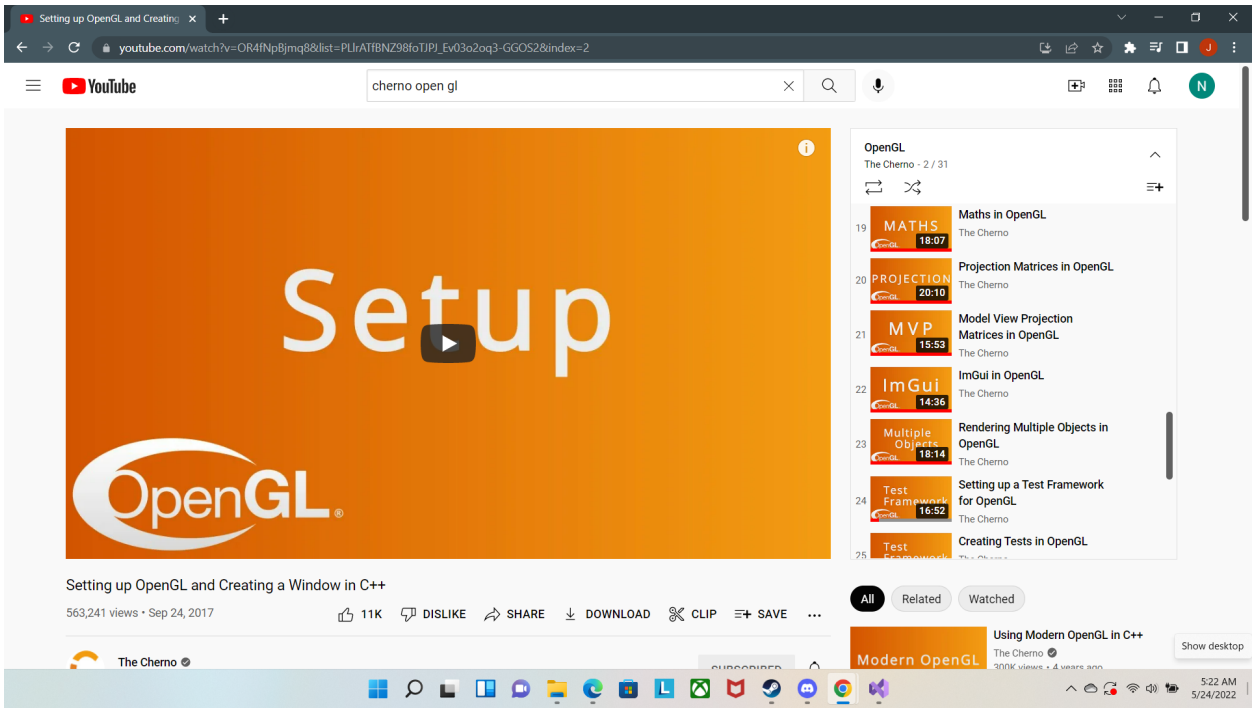
Jose Vidal
Final Project Report

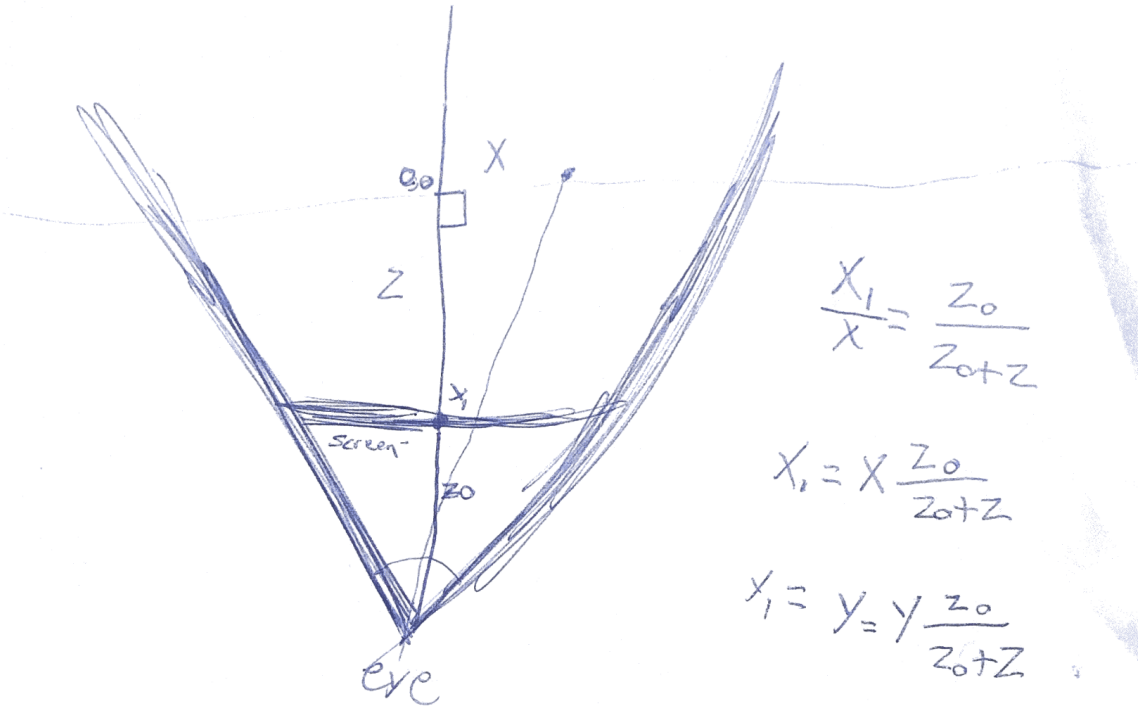
The First thing I did when I started the 3D rendering project was to research what I was going to program this on, which led me to various tutorials and websites, which took me from trying to make one that worked in the Terminal, to DirectX, all the way to Open GL. I spent many days sifting through many tutorials often going hours deep into them only to abandon them and the platforms they were using which were often too outdated, until I came across The Cherno on Youtube and proceeded to go through hours and hours of his tutorials, to both learn and implement Open GL code for the purposes of the project, however Open GL is notorious for being hard to debug, and graphics, especially the shader code is already pretty hard to debug, which is why most of my time was spent debugging, one example of this, is that on my desktop the graphics card was old enough that the type of fragment shader code on the tutorial was incompatible with it, which took me a week to figure out and circumvent. To highlight The Cherno even created his own debugging function which I implemented in my renderer to help with debugging. For the project part of what I wanted to do was abstract and streamline the rendering process because I would need to draw many points, and also connect them. Because of that I followed the tutorials and implemented an Index Buffer Class, Shader Class, Renderer, VertexBuffer, VertexArray, VertexBuffer, and VertexBufferLayout. My idea was if I learned each of these aspects and abstracted them I would be able to easily render what was needed for the project. What I did not account for was how hard it would be to debug when going from 2D to 3D, because even though the tutorial was great, the tutorial was for 2D, and built on top of itself. I was able in the end to get everything running well, however I could not get the positions to update dynamically on a larger scale with texture maps, and without texture maps and uniforms I couldn’t figure out how to update my positions in time, due to the complexities of shader code, and that if any other aspect such as the vertex buffer, uniform buffer, vertex attrib array ect. Had

an error I would just get a weird and unexpected graphical output that could be from an issue derived from any aspect of the renderer.



Hours of Tutorial Videos I went through

Outside of Coding, I researched matrix algebra, and projection matrices to implement for once I would be able to render the machine learning .



$$\frac{X_1}{X} = \frac{Z_o}{Z_o + Z}$$

$$X_1 = X \frac{Z_o}{Z_o + Z}$$

$$X_1 = Y_1 = Y \frac{Z_o}{Z_o + Z}$$

$$Z_a = \frac{Res/2}{Tan(Fov/2)}$$

Can have object with set points then have a position within Class to show where appear

Projection Research
Project Overview

Libraries
Uses GLFW Library to Create Window
Uses GLEW for modern GLFW and Open GL functionality
Uses GLM for vector multiplication
Uses STBI to import Images for Texture Maps
Uses IMGUI for debugging, and Presentation Purposes

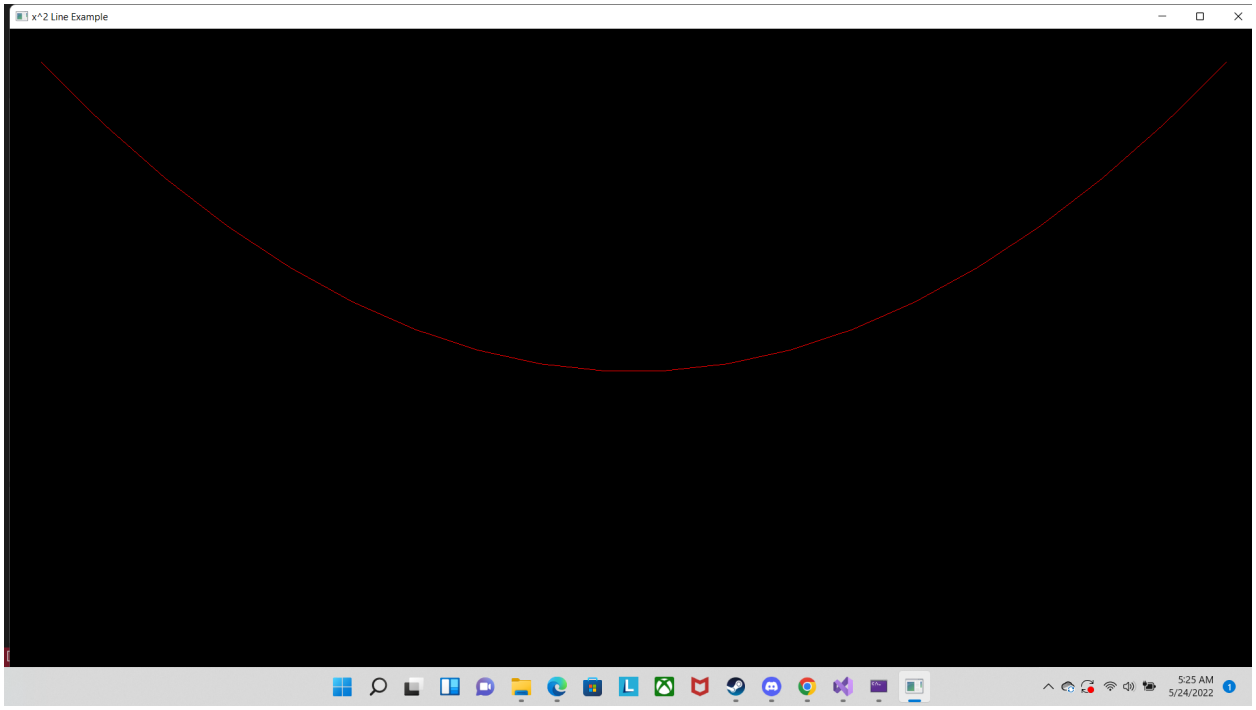
Vertex Buffers and Index Buffers are abstracted into their own classes for efficient rendering, and creation of these buffers. Vertex Buffer Layout Saves the Layout of this information for use of the Vertex Array. Vertex array plus the Index Buffer, and the Shader class are then Used by The Renderer class to draw the objects.

For the Parabola presentations the way I did it was have an array of 20 xPositions, and an initial xPosition and I would assign values to each of the xPositions iteratively by multiplying number it was in the array by the min delta and then adding the initial xPos to each of the points. In this example here the min delta was .05f, with an initial x of -.95. I had a similar system for the y, however since it is a function of x there was no need for an initial y.

Assignment code of xPos: `xCoord[i] = xCoord[i] + initialX + (i*delta);`

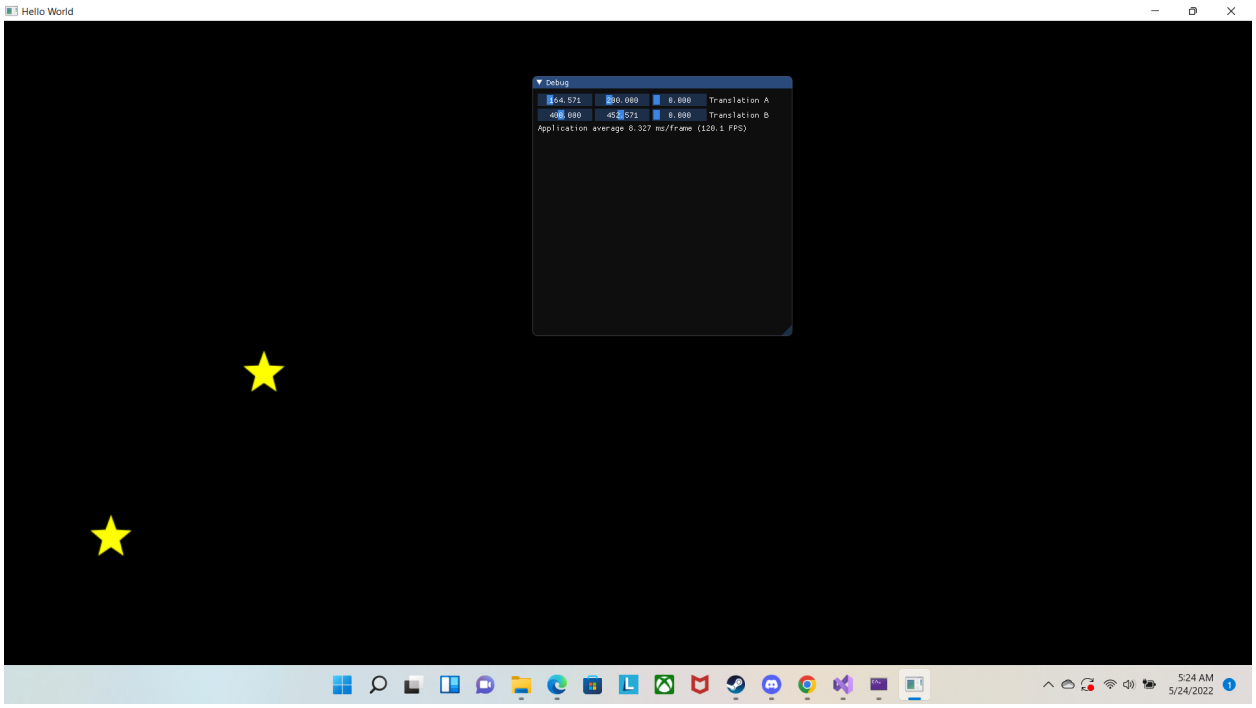
Assignment code of yPos : `yCoord[i] = pow(xCoord[i], 2);`

Results:

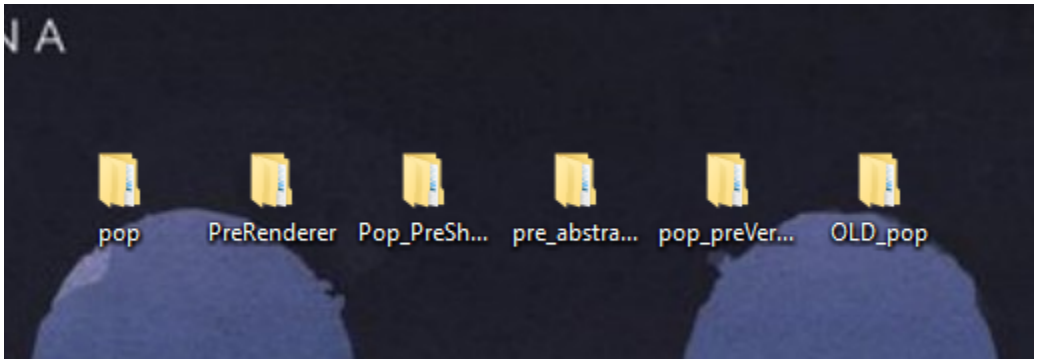


Drawn x^2 Parabola

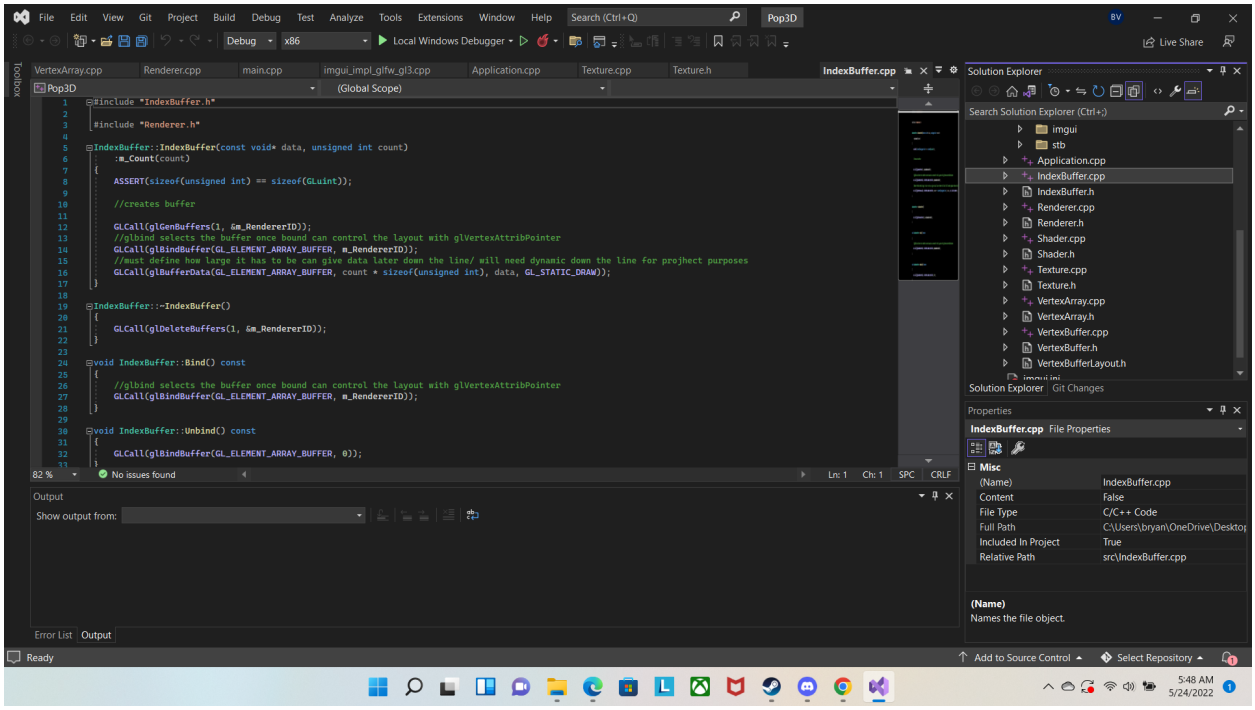
For the Star presentation I created a model projection matrix by multiplying an orthographic projection, view projection (camera), and model translation. After that via uniform “shader.SetUniformMat4f("u_MVP", mvp);” , I would pass this matrix to the shader, and dynamically update position inside of the shader with gl_Position = u_MVP * position;
And I would dynamically change the translation matrix to change how much I would be translating the stars, however due to the nature of this I needed an individual translation matrix, otherwise it would just be camera projection since the entire world would be translated if I only had one, which is why I have a translation A, and translation B here.



2 Stars Using Texture Maps That Can Be Moved With Translation Matrices



Different Iterations


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
Pop3D
Debug
x86
Local Windows Debugger
Live Share
VertexArray.cpp
Renderer.cpp
main.cpp
imgui_impl_glfw_gl3.cpp
Application.cpp
Texture.cpp
Texture.h
IndexBuffer.cpp
Pop3D
(Global Scope)
#include "IndexBuffer.h"
#include "Renderer.h"
IndexBuffer::IndexBuffer(const void* data, unsigned int count)
:m_Count(count)
{
ASSERT(sizeof(unsigned int) == sizeof(GLuint));
//creates buffer
GLCall(glGenBuffers(1, &m_RendererID));
//glbind selects the buffer once bound can control the layout with glVertexAttribPointer
GLCall(glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, m_RendererID));
//must define how large it has to be can give data later down the line/ will need dynamic down the line for projhect purposes
GLCall(glBufferData(GL_ELEMENT_ARRAY_BUFFER, count * sizeof(unsigned int), data, GL_STATIC_DRAW));
}
IndexBuffer::~IndexBuffer()
{
GLCall(glDeleteBuffers(1, &m_RendererID));
}
void IndexBuffer::Bind() const
{
//glbind selects the buffer once bound can control the layout with glVertexAttribPointer
GLCall(glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, m_RendererID));
}
void IndexBuffer::Unbind() const
{
GLCall(glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0));
}
82 %
No issues found
Ln: 1
Ch: 1
SPC
CRLF
Output
Show output from:
Error List
Output
Solution Explorer
Search Solution Explorer (Ctrl+;)
imgui
stb
Application.cpp
IndexBuffer.cpp
IndexBuffer.h
Renderer.cpp
Renderer.h
Shader.cpp
Shader.h
Texture.cpp
Texture.h
VertexArray.cpp
VertexArray.h
VertexBuffer.cpp
VertexBuffer.h
VertexBufferLayout.h
Solution Explorer
Git Changes
Properties
IndexBuffer.cpp File Properties
Misc
(Name) IndexBuffer.cpp
Content False
File Type C/C++ Code
Full Path C:\Users\bryan\OneDrive\Deskto
Included In Project True
Relative Path src\IndexBuffer.cpp
(Name)
Names the file object.
Ready
Add to Source Control
Select Repository
5:48 AM
5/24/2022

Code