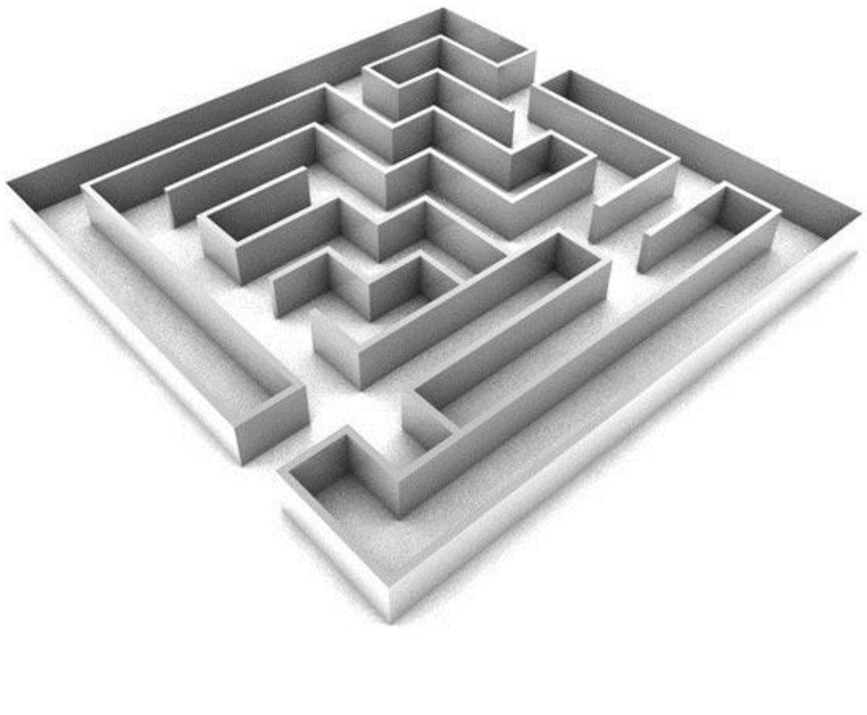
GRAPHICS

FINAL PROJECT

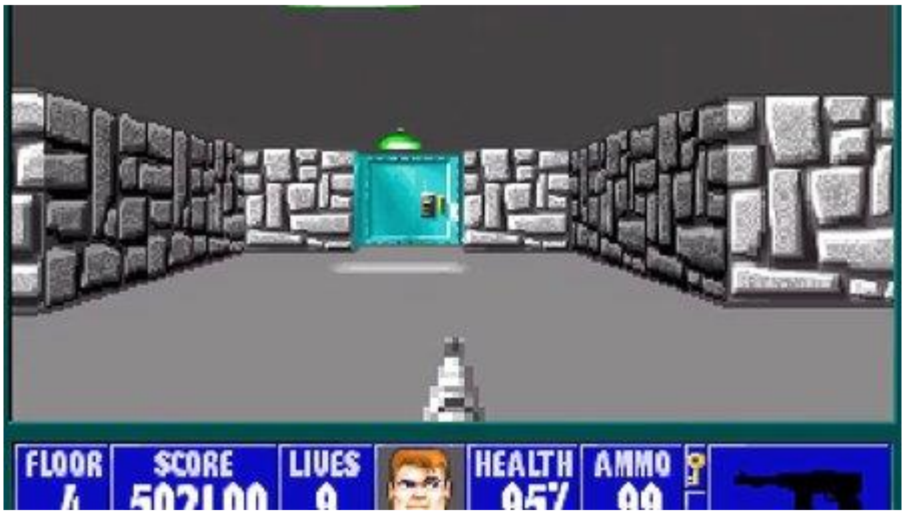
3D MAZE

Daoyu Tu

91473473



1. Goals of the project



Wolfenstein 3D

Wolfenstein 3D has always been my favourite vedio game in childhood, it is also the grandfather of all the FPS games like Battle Field and Call of Duty. This is the source of the reason why I came up with this idea of constructing a 3D maze just like this amazing scene of my childhood.

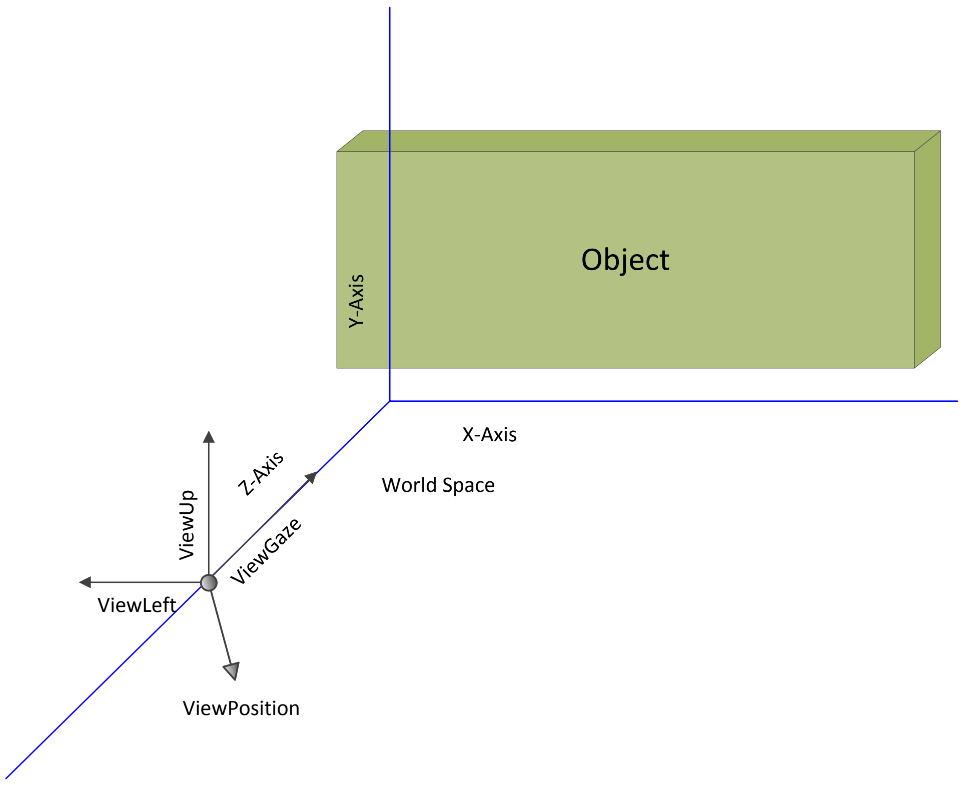
Our goal is to build a 3D maze for player to walk inside. And like in the homeworks, we are not going to use any glu function except the glDrawPixels. So the main idea is that for every pixel in the window, we are trying to culculate their RGB values by our implementation of the graphics pipeline that we learned in the whole semester.

2. My contribution to the project and some details of my job

Since we are using as less glu function as possible, we have to do a lot of basic implementations like designing data structure of triangle meshes, model/view transform, shading/ lighting, perspective transform, clipping, texture mapping and other gaming elements like movements, doors, cheating option and so on.

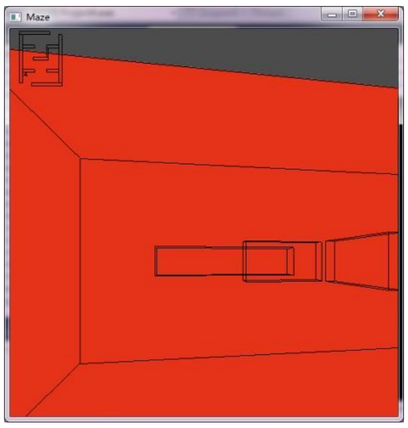
My part is basicly doing the framework of the program, shading/ lighting, rasterizing, perspective transform, clipping, movements, and cheating option.(Approximately 1500+ lines of codes.)(Texture mapping will be shown in Sijie`s report, here I will show red and green walls instead)

Shading is what we have done in homeworks, and perspective transform are very precisely discussed in the lectrues, and it`s not very hard to apply these. However, movement is what we need to design a bit.



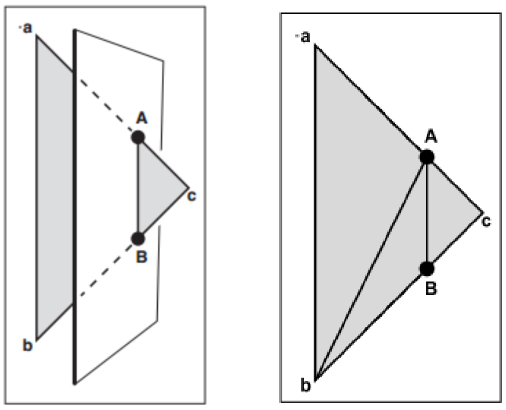
Like shown in the picture above, I used 3 factor to characterize the position and viewing direction of the player in the 3D world space, which are 2 normed vectors, ViewGaze and ViewUp, and 1 point, ViewPosition. Then we can have another normed vetocr ViewLeft based on the inner product of ViewUp and ViewGaze. For example, if we want to move forward, instead of only changing the z coordinate of ViewPosition, which seems right in the picture, we make a augmentation alone the ViewGaze vector. Since they are all normed vector, we can make sure that the movement is always one unit. And to make it more like in game, we also need to change the gaze direction by rotate the ViewGaze about ViewUp, and since ViewLeft is based on these two, they three will always be orthogonal.

With moving and rotating, we can travel almost everywhere in the maze, yet we need some constrain to make sure the player will not hit the wall nor go throught it. We did this by checking if our ViewPosition will move outside some area.



We can not move further if a wall is ahead us, the picture in the left show that in cheating mode(we will discuss that later) while we can see wire frames of walls in a transparent way.

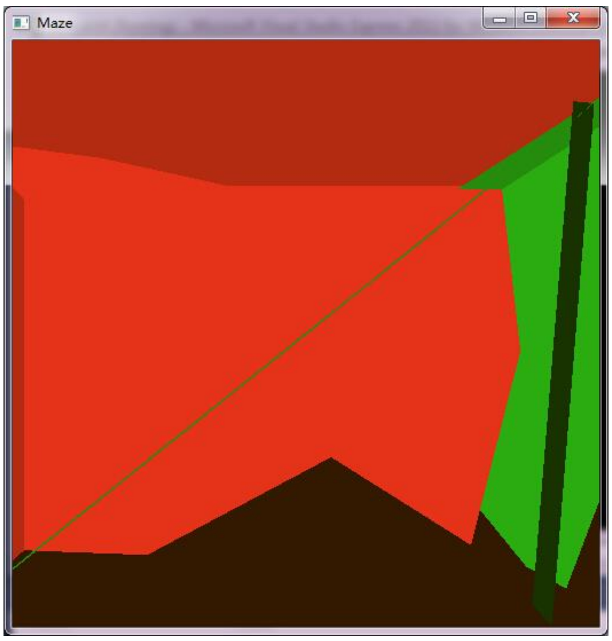
The most challenging part is clipping, the basic idea shown in lecture is very intuitive. However, since there are a lot of boudary and float problem, I spent more than half of the time trying to find a correct and efficient algorithm in practice.

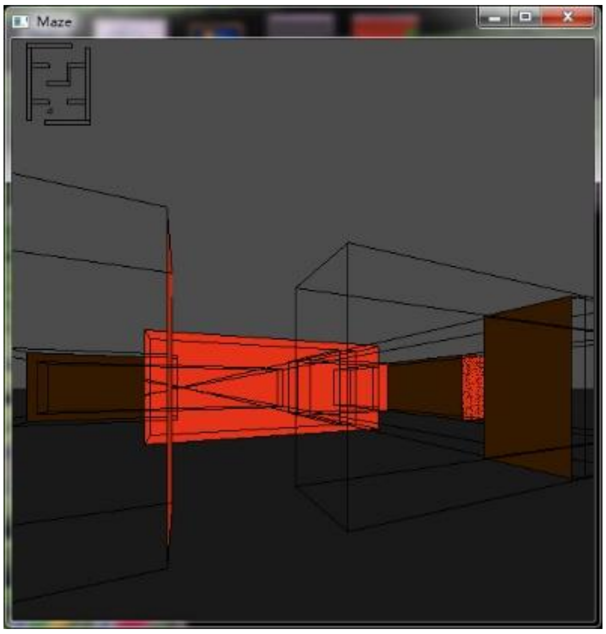


basic idea of clipping

The first was a very na ïve algorithm, it was for every triangle, we use 6 faces to clip it. Since one triangle may break into two, like cell divition, we can potentially have 2^6 triangles, it is a huge number it is applied to triangle meshes, and it did act very slow in the test of one wall.

Then I tried to clip it after the projection in 2D window which can be easier. In 2D, a triangle can be clipped to a convex, the vertice of the convex consist of 3 kind:1. the vertex of triangle that is inside the window; 2. the vertex of the window that is inside the triangle; 3. intersection of edge of triangle and window. Then my algorithm became: find these points to consist a convex, then break this convex into triangles to rasterize them. This time it looks and works fine until I walked into the maze, it became a huge mess like shown below. To make it visual to see the problem, I

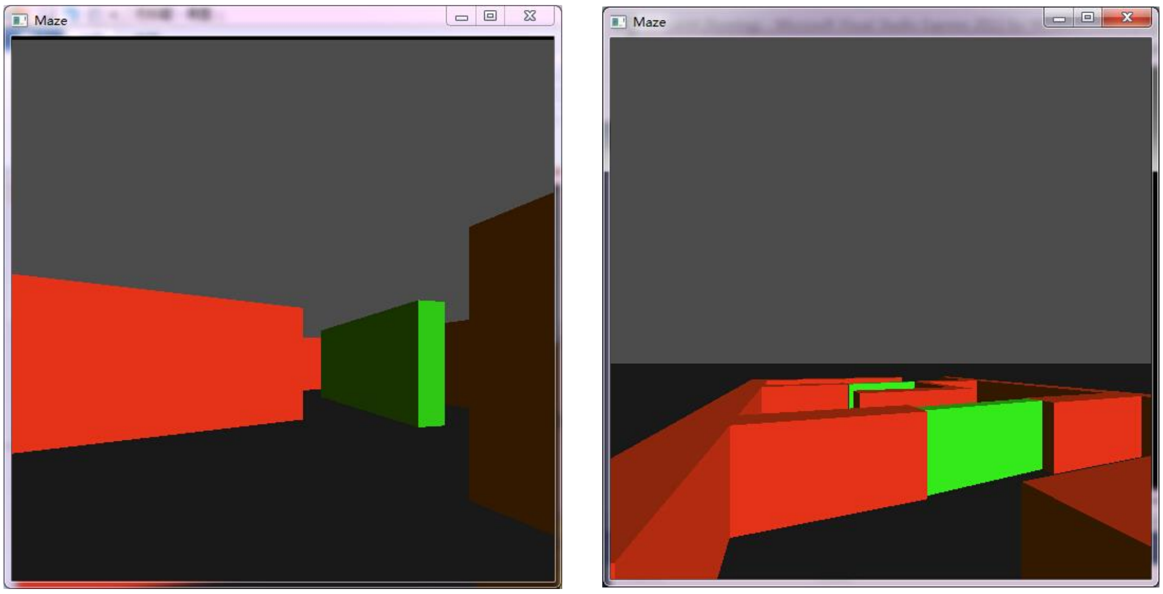




a big mess after entering the maze

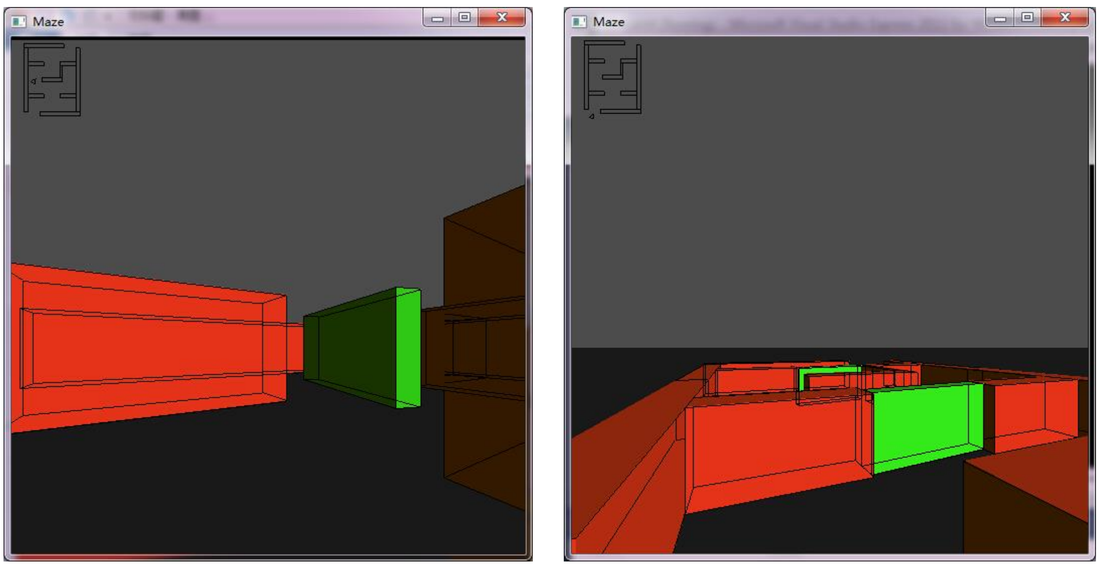
a wall behind the back appers

draw the wire frame of the triangle that need to clip instead of coloring them. Then I found out that a wall that should be behind the back appears in front and there is no bug if everything is in front of the player. I think this is happening because if some point are near the plane that is characterized by ViewPosition and ViewGaze can be projected to very large x,y coordinates which might make the image look strange, especially points lies in that plane. I eliminate this phenomenon by doing a clipping in 3D before clipping in 2D, which can get rid of the triangles that are behind the player. After that, the clipping works not perfect, but way better.

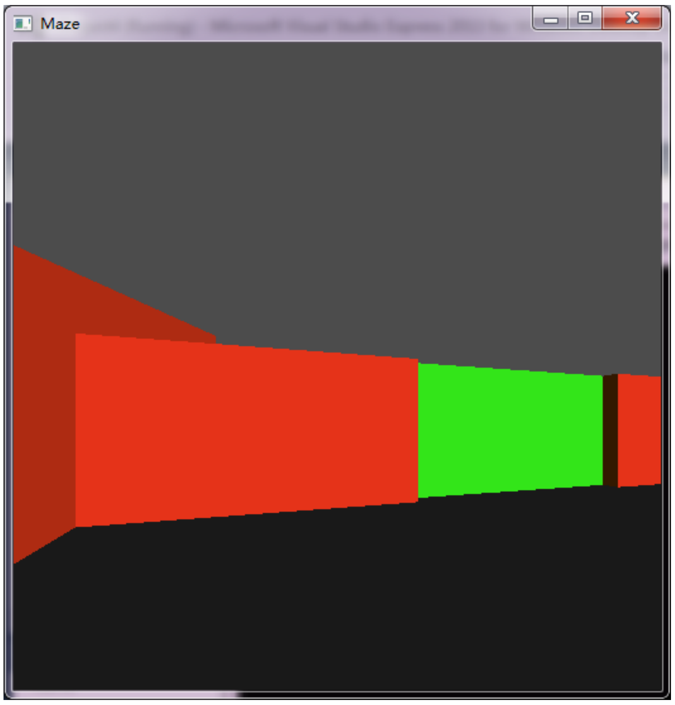


clipping works at last

Next thing is the cheating option. When the player changes into this mode, he will have the ability to see walls behind walls and small map that tells where he is at and which direction he is in. Showing the small map is essentially do another porjection form above, scale it down and draw the wire frame of the walls as long as the player himself.



I have to admit that there are still some bugs while clipping, it happens in some specific positions and directions. I believe it is when some triangle hit boundaries, I should be more careful about boundary problems.



3. Learning outcomes

We have learned a lot about how the graphic pipeline works this semester, I think this project is implementing the whole graphic pipeline. Though its is not a very freshing field or difficult topic, it contains a lot of practice of coding, solving bugs, design a program and teamwork, we did feel very happy when we joint our codes togather and saw our maze first came out of the screen.

The presentation was my first time talking English in front of people. I was very nervous in the presentation because there are still critical bugs in clipping that I can not show them and we could not either show the texture part, I think I did very bad, but I believe this failure will help me in the future which makes it another important lesson learned.

I am glad I practiced a lot in the homeworks, project and the presentation. And I really appreciate the help of professor Alireza and TA Tushar, and the contribution of my teammates.

4. Thoughts on future work:

First, we can solve the bugs about clipping and foreshortening in texture mapping.
Second, we can optimize our algorithms because right now it is lagging a bit.
Third, we can add more interesting elements like some signs that could influence the player to make it harder, or add time constrain or even add someone chasing.