# Report & Introduction

I choose a topic that “A sophisticated 3D interactive tutorial to illustrate concepts from the course (a super cool and informative demo).

**In the most of time, I like to play the Game Minecraft, thus, I choose to implement a Minecraft Demo in this course (always my dream)** and use the concept in the graph.

The material is refer to the [1], I download a pig,grond material. The pig I also download the obj code. In addition to this, I download the cy, cy3, and image background by myself from the google image.

In the source code the Final\_Project is programmed by myself.

I add the lib to the common folder.

* Description

1. I set two camera view in the game.
2. //camera
3. var cameraControl1=0;
4. var cameraControl2=0;
5. var cameraControl3=0;
6. var cameraControl4=0;
7. var cameraControl5=0;
8. var cameraControl6=0;

This variables shows the code to control the camera.

1. I set two light a parallel light and a dot light to satisfy the requirement Two or more light sources must be activated to illuminate the scene;
2. uniform mat4 modelViewMatrix;
3. uniform mat4 projectionMatrix;//projectionMatrix
4. uniform mat4 u\_world;//parallel light
5. uniform mat4 modelView;//
6. uniform vec3 u\_lightWorldPosition;//light position
7. //light
8. var light1=0;
9. var light2=0;
10. var light3=0;
11. var light4=0;
12. var light5=0;
13. var light6=0;

This code show the basic light we set. Then we will rely on the library.

This code show the basic parallel light and dot light

//switch light

gl.uniform1i(gl.getUniformLocation(program, "sunControl"), sunControl);

gl.uniform1i(gl.getUniformLocation(program, "dotControl"), dotControl);

gl.uniform1i(gl.getUniformLocation(program, "limitControl"), limitControl);

this function to switch the light

1. I use looked at function to the system and matrix.js to process the matrix computationA system for efficient, intuitive navigation must be presented, using the lookAt() function or a matrix equivalent.
2. modelViewMatrix = lookAt(eye, at, up);
3. }
5. if(view==1)
6. {
7. eye = vec3(radius\*Math.sin(theta)\*Math.cos(phi), radius\*Math.sin(phi), radius\*Math.cos(theta)\*Math.cos(phi));
8. at=vec3(0,0,0);
9. modelViewMatrix = lookAt(eye, at, up);
10. }

Above I use look at function to look at the eye of the human

1. I also using both keyboard and mouse functionality; to process the game up and down.

I use it to control the camera moving and human moving.

Also I can use mouse to click button to control the light.

1. if(e && e.keyCode==40)
2. {
3. //down
4. backview=-1;
5. }
6. if(e && e.keyCode==37)
7. {
8. //left
9. leftview=-1;
10. }
11. if(e && e.keyCode==39)
12. {
13. //right
14. rightview=-1;
15. }
16. if(e && e.keyCode==38)
17. {
18. //up
19. forwardview=-1;
20. };
21. if(e && e.keyCode==87)
22. {
23. // w
24. upview=-1;
25. };
26. if(e && e.keyCode==83)
27. {
28. // s
29. downview=-1;
30. };
31. Textures;

I set a texture by myself in the variable

Texcoord

var texCoord = [

//human

vec2(0,1),

vec2(0,1),

vec2(0.25,1),

vec2(0.5,1),

vec2(0.75,1),

vec2(1,1),

vec2(0,0.75),

vec2(0.25,0.75),

vec2(0.5,0.75),

the details can be seen in the js code line 170.

I also use webgl to bind the buffer and texture image

tBuffer1 = gl.createBuffer();

gl.bindBuffer(gl.ARRAY\_BUFFER, tBuffer1);

gl.bufferData(gl.ARRAY\_BUFFER, flatten(texCoordsArray.slice(0,object1Len)), gl.STATIC\_DRAW);

vTexCoord = gl.getAttribLocation(program, "vTexCoord");

gl.enableVertexAttribArray(vTexCoord);

//texture image

var image1 = document.getElementById("cy");

configureTexture1(image1);

1. elements are moving in a hierarchical relationship with other elements

There are multiple elements in the game. So there are fully of the hierachical relationships.

1. A “novel component“

I add a pig in the game. We can run the game in a grass.

Also my novelty is whether the light source is spilled on the target, it sounds like there is collision detection! Let ’s recall the collision detection in 2D. Our detection is based on the AABB method, which is based on the object ’s bounding box (object top, left, width, height), and then the coordinates (x, y) and Calculate to determine the collision. This method has a drawback, that is, there may be errors in the detection of non-rectangular shapes, such as circles, triangles, etc. After all, the bounding box is rectangular. The AlloyPage game engine developed by dntzhang has a painter's algorithm that perfectly solves this defect, and the detection granularity is changed from object to pixel. Interested students can go to study ~ Let me not mention it here, we are talking about 3D detection.

* User Manual

There are 6 key to press

Up, down, left,right, w,s to control the camera

There are 6 button to control the human move and bigger or Smaller **CYXXX**

There are 6 buttons to control the **pigXXX**

There are 6 buttons to control the camera (different camera) **Cameraxxx left right up down**

**Light XXX** to control the light

**Sun light** to control the sun light

**Dot light** to control the dot light

* acknowledgment of sources and references

I use website [1] to get the image, picture and texture

I use [2],[3],[4],[5],[6] to help with my webgl code. These code is downloaded from Internet.

* Screenshots

A picture containing grass, building, green, field

Description automatically generated

Normal Screenshot

A screenshot of a computer

Description automatically generated

Button

A large green field

Description automatically generated

Different view

A picture containing grass, field, green, man

Description automatically generated

Sun arise

A picture containing grass, field, green, building

Description automatically generated

Another Face



Only the dot light

* special features

1. I add a **facexxx button to add a face to the humand and pig**
2. I add a reset button to the game
3. I want to memory my Minecraft Game experience.

## Reference

[1] <https://minecraft.gamepedia.com/Materials>

[2] webgl-utils.js: standard utilities from google to set up a webgl context

[3] MV.js: our matrix/vector package. Documentation on website

[4] initShaders.js: functions to initialize shaders in the html file

[5] initShaders2.js: functions to initialize shaders that are in separate files

[6] Bootstrap