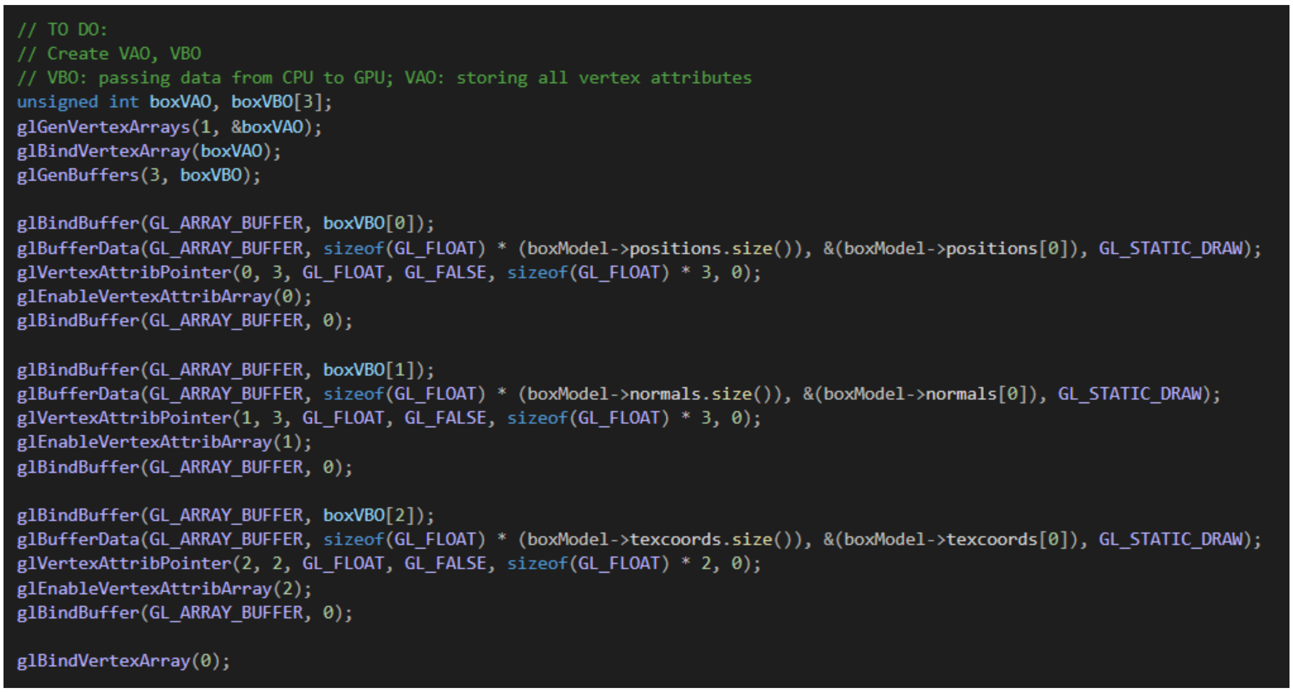
109550069_林宇柔

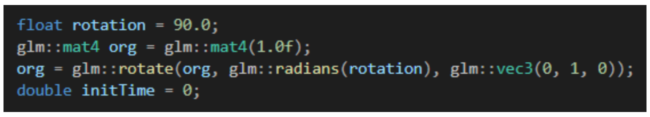
# HW2 Report

## I. Code



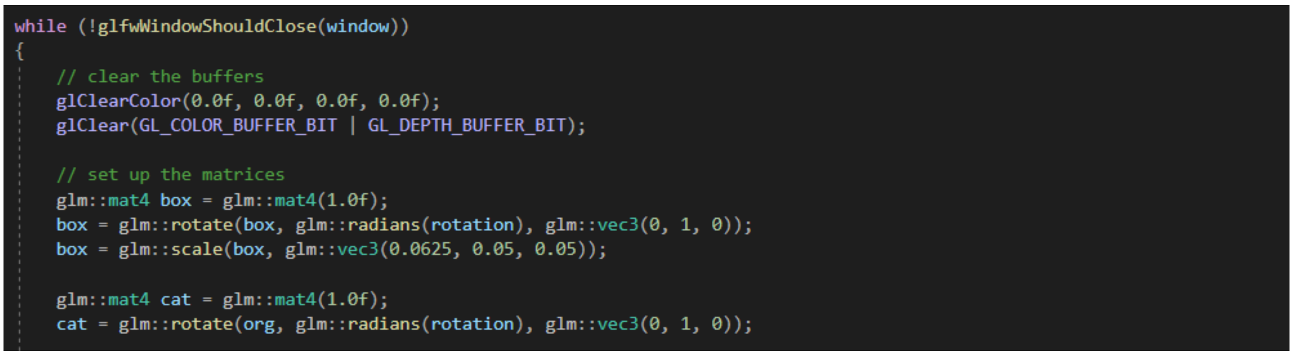
```
// TO DO:
// Create VAO, VBO
// VBO: passing data from CPU to GPU; VAO: storing all vertex attributes
unsigned int boxVAO, boxVBO[3];
glGenVertexArrays(1, &boxVAO);
glBindVertexArray(boxVAO);
glGenBuffers(3, boxVBO);

glBindBuffer(GL_ARRAY_BUFFER, boxVBO[0]);
glBufferData(GL_ARRAY_BUFFER, sizeof(GL_FLOAT) * (boxModel->positions.size()), &(boxModel->positions[0]), GL_STATIC_DRAW);
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, sizeof(GL_FLOAT) * 3, 0);
glEnableVertexAttribArray(0);
glBindBuffer(GL_ARRAY_BUFFER, 0);

glBindBuffer(GL_ARRAY_BUFFER, boxVBO[1]);
glBufferData(GL_ARRAY_BUFFER, sizeof(GL_FLOAT) * (boxModel->normals.size()), &(boxModel->normals[0]), GL_STATIC_DRAW);
glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, sizeof(GL_FLOAT) * 3, 0);
glEnableVertexAttribArray(1);
glBindBuffer(GL_ARRAY_BUFFER, 0);

glBindBuffer(GL_ARRAY_BUFFER, boxVBO[2]);
glBufferData(GL_ARRAY_BUFFER, sizeof(GL_FLOAT) * (boxModel->texcoords.size()), &(boxModel->texcoords[0]), GL_STATIC_DRAW);
glVertexAttribPointer(2, 2, GL_FLOAT, GL_FALSE, sizeof(GL_FLOAT) * 2, 0);
glEnableVertexAttribArray(2);
glBindBuffer(GL_ARRAY_BUFFER, 0);

glBindVertexArray(0);
```

In main.cpp, first we have to create VAO and VBO, taking the box for example, I first declared them and generate vertex arrays, for VBO it would be an array of length 3 to store position, normal, and texture coordinates, then bind the vertex array with VAO. Then I bound the three elements of the VBO to GL_ARRAY_BUFFER and copied the addresses of the elements in the arrays. Furthermore, assign and enable the indices and the size to be assigned, for instance the position is at index 0 and a size of 3 floats is needed. Finally we have to unbind them by passing 0.
The code for VAO and VBO of the cat is exactly the same.

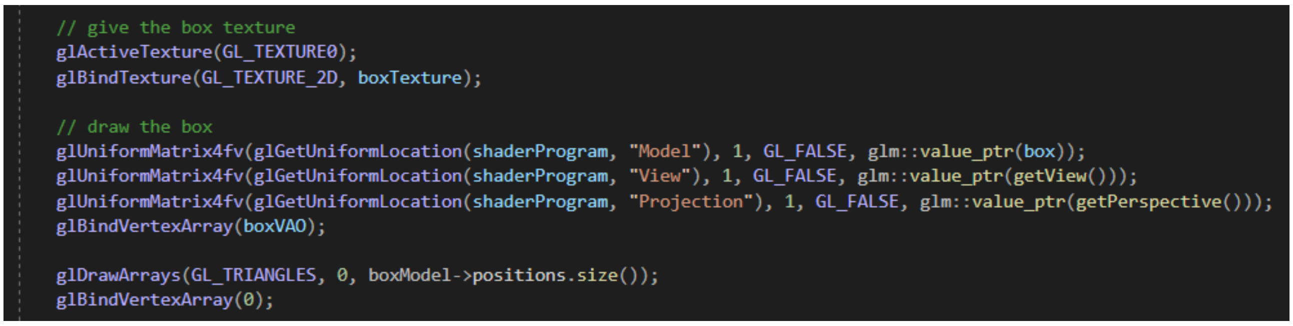


```
float rotation = 90.0;
glm::mat4 org = glm::mat4(1.0f);
org = glm::rotate(org, glm::radians(rotation), glm::vec3(0, 1, 0));
double initTime = 0;
```

Before getting into the while loop, there are several variables that have to be assigned first. The float rotation is for recording the angle increase in each frame. The matrix org is to make the cat begin in the correct direction. The double initTime is for recording the time passes in each frame.

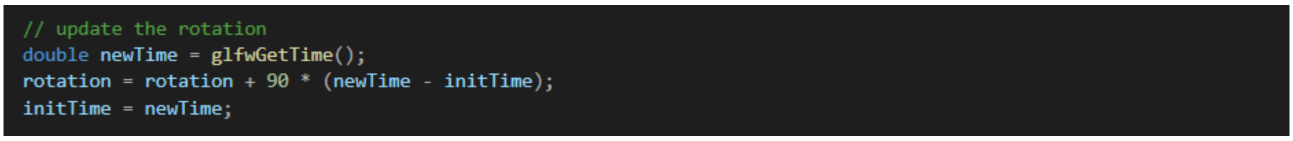


```
while (!glfwWindowShouldClose(window))
{
    // clear the buffers
    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    // set up the matrices
    glm::mat4 box = glm::mat4(1.0f);
    box = glm::rotate(box, glm::radians(rotation), glm::vec3(0, 1, 0));
    box = glm::scale(box, glm::vec3(0.0625, 0.05, 0.05));

    glm::mat4 cat = glm::mat4(1.0f);
    cat = glm::rotate(org, glm::radians(rotation), glm::vec3(0, 1, 0));
```



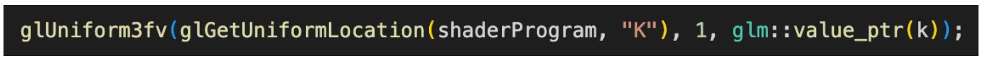
```
// give the box texture
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, boxTexture);

// draw the box
glUniformMatrix4fv(glGetUniformLocation(shaderProgram, "Model"), 1, GL_FALSE, glm::value_ptr(box));
glUniformMatrix4fv(glGetUniformLocation(shaderProgram, "View"), 1, GL_FALSE, glm::value_ptr(getView()));
glUniformMatrix4fv(glGetUniformLocation(shaderProgram, "Projection"), 1, GL_FALSE, glm::value_ptr(getPerspective()));
glBindVertexArray(boxVAO);

glDrawArrays(GL_TRIANGLES, 0, boxModel->positions.size());
glBindVertexArray(0);
```

In the while loop, first I clear the buffers and set up the matrices for rotating the models and scaling the box. Then I activated and bound the texture 0 with the texture that is being used. After that, I declared uniform matrices for Model, View, Projection with their values, and with the VAO bound it is able to draw.
The code for giving texture and drawing the cat is exactly the same.



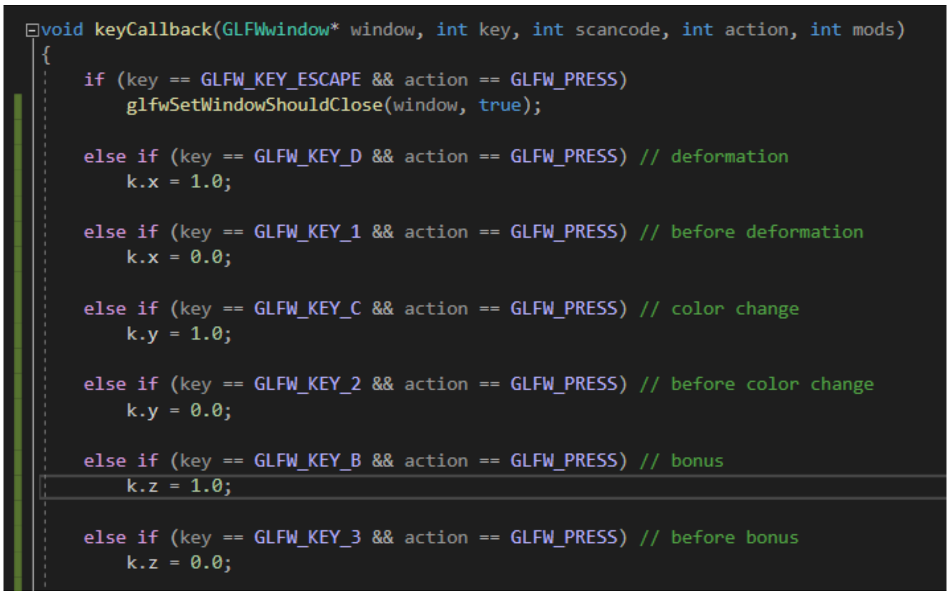
```
// update the rotation
double newTime = glfwGetTime();
rotation = rotation + 90 * (newTime - initTime);
initTime = newTime;
```

Using glfwGetTime to get the time elapsed, the angle of rotation is to add the time elapsed times 90 since it has to rotate 90 degrees per second, after updating the angle, update the initTime.



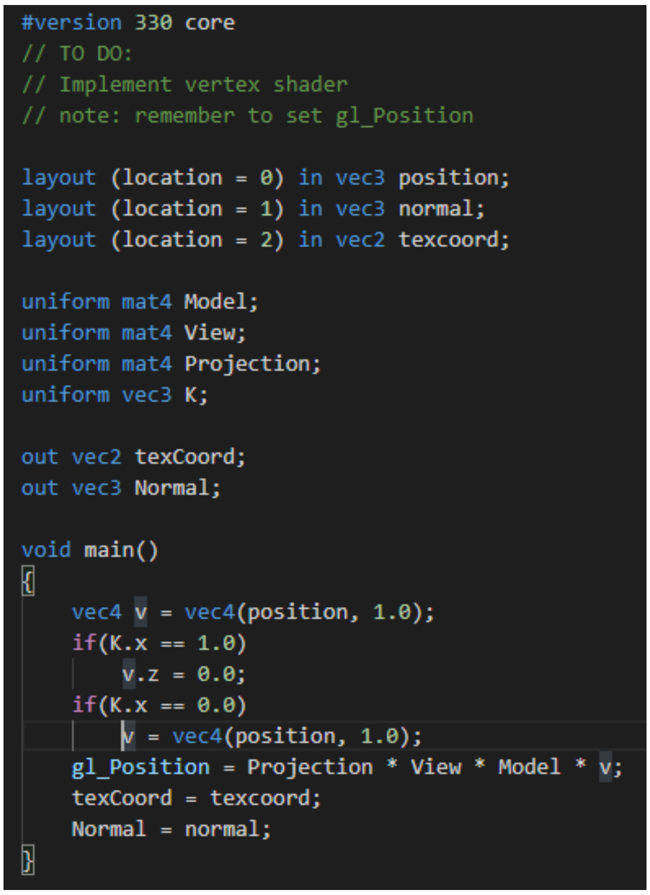
```
glUniform3fv(glGetUniformLocation(shaderProgram, "K"), 1, glm::value_ptr(k));
```

For controlling the key press, I declared a uniform 3-element vector to store the changes and initialize them to 0, x is for deformation, y is for color change, and z is for bonus.



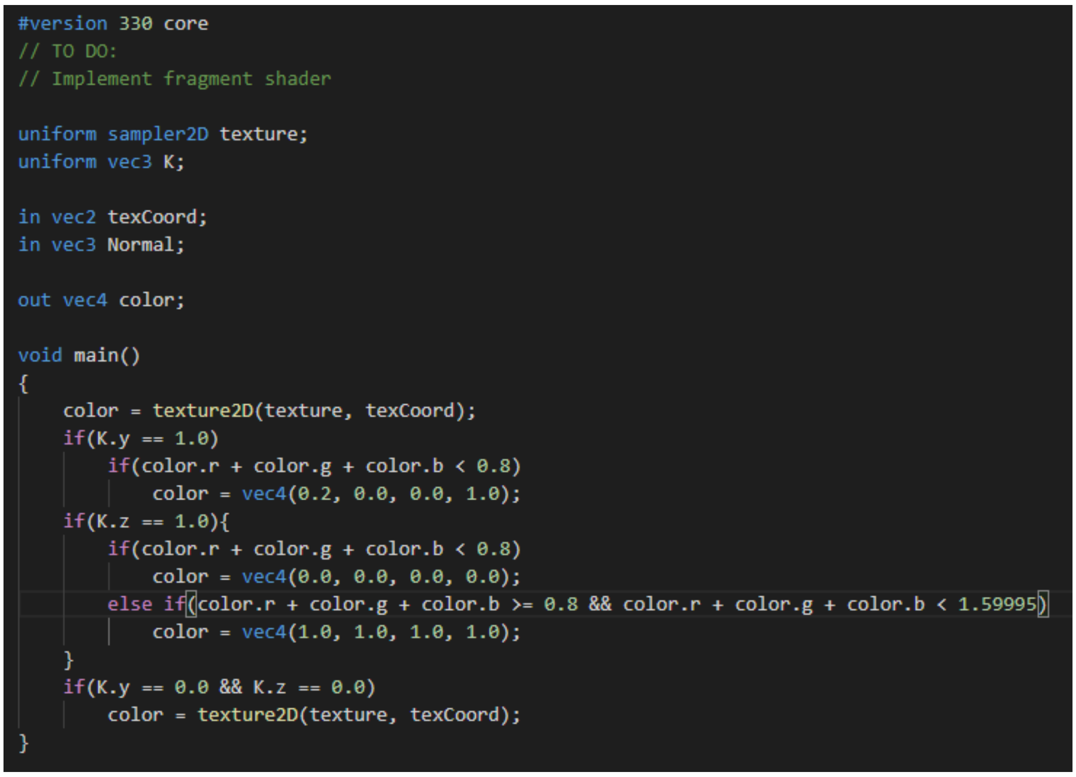
```
void keyCallback(GLFWwindow* window, int key, int scancode, int action, int mods)
{
    if (key == GLFW_KEY_ESCAPE && action == GLFW_PRESS)
        glfwSetWindowShouldClose(window, true);

    else if (key == GLFW_KEY_D && action == GLFW_PRESS) // deformation
        k.x = 1.0;

    else if (key == GLFW_KEY_1 && action == GLFW_PRESS) // before deformation
        k.x = 0.0;

    else if (key == GLFW_KEY_C && action == GLFW_PRESS) // color change
        k.y = 1.0;

    else if (key == GLFW_KEY_2 && action == GLFW_PRESS) // before color change
        k.y = 0.0;

    else if (key == GLFW_KEY_B && action == GLFW_PRESS) // bonus
        k.z = 1.0;

    else if (key == GLFW_KEY_3 && action == GLFW_PRESS) // before bonus
        k.z = 0.0;
```

1 means the effect is turned on, and the effect can be shut down by another key press:
For deformation: press D to activate and 1 to deactivate.
For color change: press C to activate and 2 to deactivate.
For bonus: press B to activate and 3 to deactivate.



```
#version 330 core
// TO DO:
// Implement vertex shader
// note: remember to set gl_Position

layout (location = 0) in vec3 position;
layout (location = 1) in vec3 normal;
layout (location = 2) in vec2 texcoord;

uniform mat4 Model;
uniform mat4 View;
uniform mat4 Projection;
uniform vec3 K;

out vec2 texCoord;
out vec3 Normal;

void main()
{
    vec4 v = vec4(position, 1.0);
    if(K.x == 1.0)
        v.z = 0.0;
    if(K.x == 0.0)
        v = vec4(position, 1.0);
    gl_Position = Projection * View * Model * v;
    texCoord = texcoord;
    Normal = normal;
}
```

In the vertex shader, set the gl_Position with the uniform matrices Model, View, Projection. And also check the key.x, if it is 1 means the effect is activated, I set the z coordinate of the position to 0 to implement flattening, if it is 0, set it back to the original position. Also assign the texture coordination and normal.

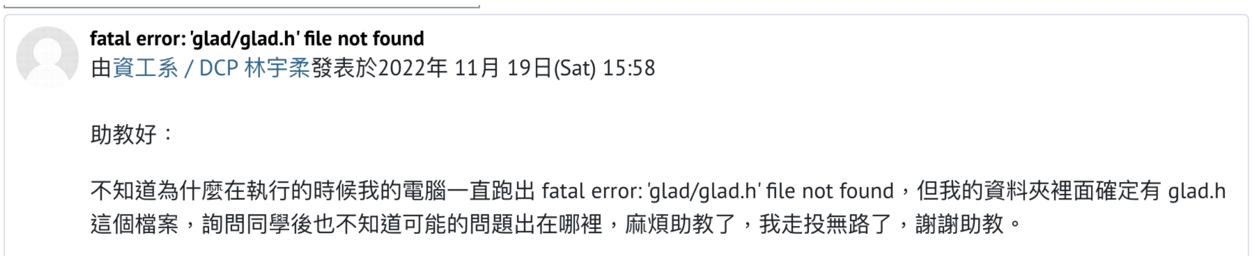


```
#version 330 core
// TO DO:
// Implement fragment shader

uniform sampler2D texture;
uniform vec3 K;

in vec2 texCoord;
in vec3 Normal;

out vec4 color;

void main()
{
    color = texture2D(texture, texCoord);
    if(K.y == 1.0)
        if(color.r + color.g + color.b < 0.8)
            color = vec4(0.2, 0.0, 0.0, 1.0);
    if(K.z == 1.0){
        if(color.r + color.g + color.b < 0.8)
            color = vec4(0.0, 0.0, 0.0, 0.0);
        else if(color.r + color.g + color.b >= 0.8 && color.r + color.g + color.b < 1.59995)
            color = vec4(1.0, 1.0, 1.0, 1.0);
    }
    if(K.y == 0.0 && K.z == 0.0)
        color = texture2D(texture, texCoord);
}
```

In the fragment shader, map the texture with the uniform sample texture and the texture coordinate passed in. if key.y is 1 means the color effect is activated, check where the stripes of the cat are by summation of the color elements, I set the threshold to 0.8, if it is a stripe, I paint it with a darker color. If key.y is set back to 0 it is deactivated and will resume to the original color.

## II. Problems Encountered

When I first start my homework, there was a problem with my computer, although all my header files are put in the correct folder, the terminal kept saying that glad.h is not found, I tried several ways to fix it but the error still exists, and I tried to compile HW1 but it is still not able to run, I think I broke my computer. At last I finish this assignment at the computer center.



**fatal error: 'glad/glad.h' file not found**
由資工系 / DCP 林宇柔發表於2022年 11月 19日(Sat) 15:58

助教好：

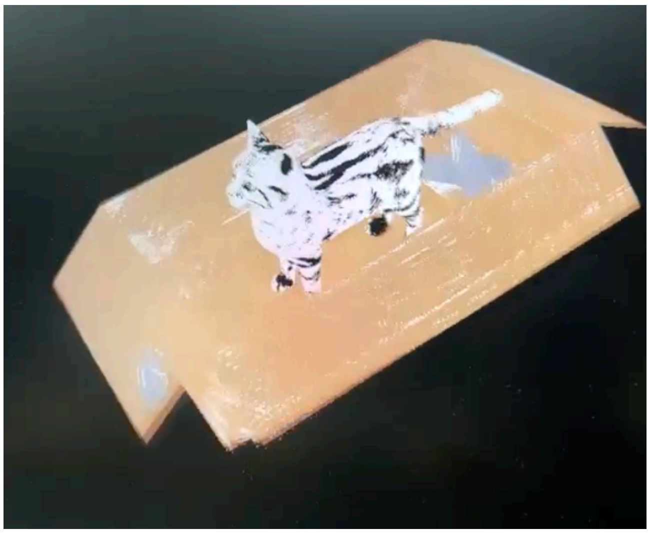
不知道為什麼在執行的時候我的電腦一直跑出 fatal error: 'glad/glad.h' file not found，但我的資料夾裡面確定有 glad.h 這個檔案，詢問同學後也不知道可能的問題出在哪裡，麻煩助教了，我走投無路了，謝謝助教。

我走投無路了

The second big problem was that the my box wasn't colored with the texture, instead, it was pure gray, I tried to fix the vertex shader and the fragment shader, but that didn't fix the problem, finally I found that the problem was within binding the VAO.

## III. Bonus

I made a zebra.



109550069_林宇柔

Press B to change the cat into a zebra, and press 3 to bring the cat back.
In the fragment shader, if key.z is activated, I changed the color of the stripes to black and the other parts to white by examining the summation of color values.
I like zebras.