DH2323 Lab 1

Xuni Huang
xuni@kth.se

April 21, 2022

1 Implementation methods

1.1 Set up

The skeleton code from https://github.com/lemonad/DH2323-Skeleton was used. After modifying the path to SDL and GLM, the coding environment was successfully set up. Visual Studio Code IDE was used on a Mac laptop to modify and execute the code.

1.2 Part 1 - colour Interpolation

This part was completed by carefully following the lab instructions. Firstly drawing different colours on the screen to test my understanding of the colour model and the draw() function.

Secondly, to get the linear interpolation work, the numbers presented in the lab instructions were used to test. Instead of using coding to get these numbers, mathematical calculation was firstly used. The formula for colour interpolation is $colour\ after\ interpolation = colour\ A * (1 - t) + colour\ B * t$, where t is the fraction. Using the given numerical numbers (i.e., 5 6 7 8 9 10 11 12 13 14), I know that $t = i/size\ of\ the\ result\ array - 1$. Using this formula I get the linear interpolation work. When the size of the result array is 1, fraction $t = 0.5$.

Finally, when implementing the bilinear interpolation, four colours for four corners were firstly defined. After this, instead of interpolating at the y-axis first, I chose to interpolate the values of the very top and very down side of the screen. Then interpolation between the values of the top and down side of each column of the screen was implemented.

1.3 Part 2 - Starfield

Firstly, random numbers were generated to initialise the star positions. Since $float\ r = float(rand())/float(RAND\_MAX)$ can only generate random numbers from 0 to 1, For the positions at x-axis and y-axis, another random number generation formula was used. This formula is $-1.0\ +\ static\_cast < float > (rand())/(static\_cast < float > (RAND\_MAX/(1.0 - (-1.0))))$;

Secondly, the screen was firstly painted black. Then the star projection was done. As shown in 1, u_i and v_i can be calculated by multiplying focal length with the ratio between x_i and z_i, and between y_i and z_i. Besides, since the origin of the plane is in the left top corner as presented in 1, $W/2$ and $H/2$ need to be added respectively.

Finally, the update() function was updated to get the stars moving. To enable the stars to move towards the camera, the stars need to be updated at the z-axis. To update the value of the stars at the z-axis with a velocity. The velocity of the points is updated by multiplying by the frame time difference. Also, ensuring the value at the z-axis always stays between 0 and 1 enables the stars to move forever.

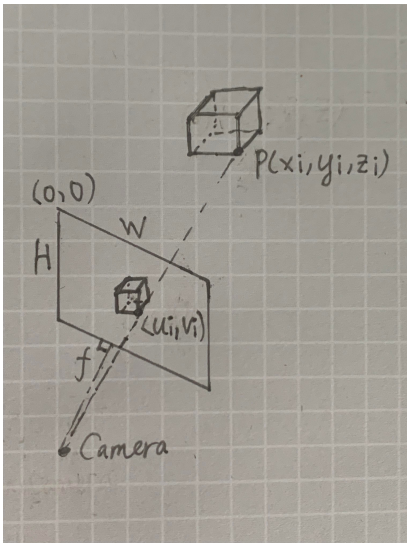

Figure 1: Pinhole camera model for 3D point projected on a plane

2 Experiences

It took some time to set up the lab. I encountered errors saying that cannot find SDL and GLM libraries. I solved it by using the skeleton code mentioned earlier and corrected the path to the SDL and GLM libraries.

Other than that, it also took some time for me to understand how all equations work and how to implement them.

3 Outputs

Everything worked pretty well. The output of the colour interpolation was presented in 2. The output of the colour interpolation was presented in 3.

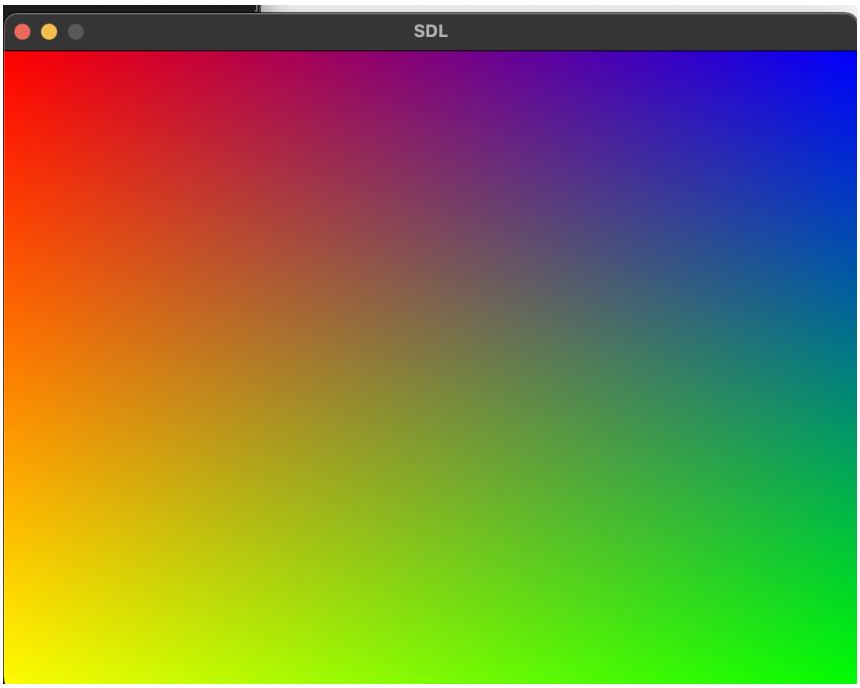

Figure 2: Output screenshot of Part 1

Figure 3: Output screenshot of Part 2