DH2323 Lab1 Report

Shuang Qiu (shuangq@kth.se)

1. Objective

Lab1 includes three parts:

1. Set-up of the lab environment.
2. Introduction to 2D computer graphics. This part covers image representation, colors, pixels and linear interpolation.
3. Introduction to 3D computer graphics. This part covers the pinhole camera and how it projects 3D points to 2D. The result will be a program that produces a starfield effect.

2. Introduction to 2D Computer Graphics

2.1 Linear interpolation

The Interpolation function that works for glm::vec3:

```
void Interpolate(vec3 a, vec3 b, vector<vec3>& result){
    size_t length = result.size();

    //if size = 1, pick the color on the left corner
    if( length <= 1 ){
        result[0] = a;
    }else{
        float stepX = (b.x - a.x)/( length - 1 );
        float stepY = (b.y - a.y)/( length - 1 );
        float stepZ = (b.z - a.z)/( length - 1 );

        for(int i=0; i < length; ++i){
            result[i].x = a.x + stepX*i;
            result[i].y = a.y + stepY*i;
            result[i].z = a.z + stepZ*i;
        }
```

```
    }
}
```

2.2 Biliear Interpolation of Colors

First I interpolated linearly between the four corner color values, and store the value into two vectors `leftside` and `rightside`. Then, for each row I picked the values in the left and right sides, then interpolated linearly between these two values to get the color for each pixel on this row.

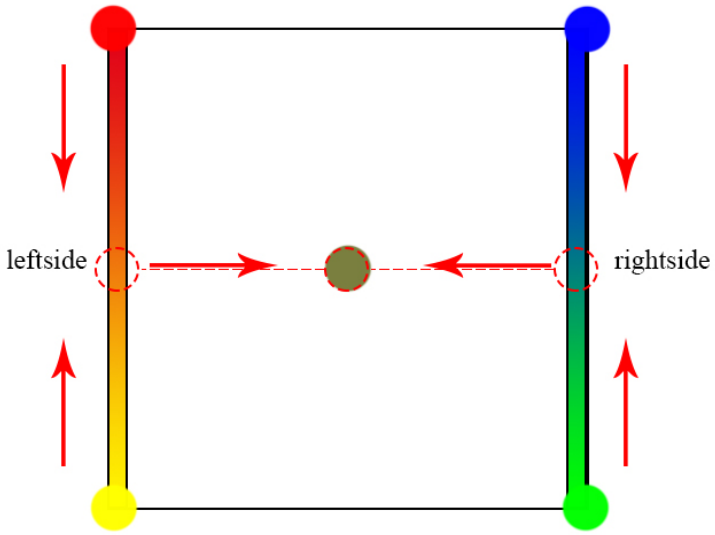
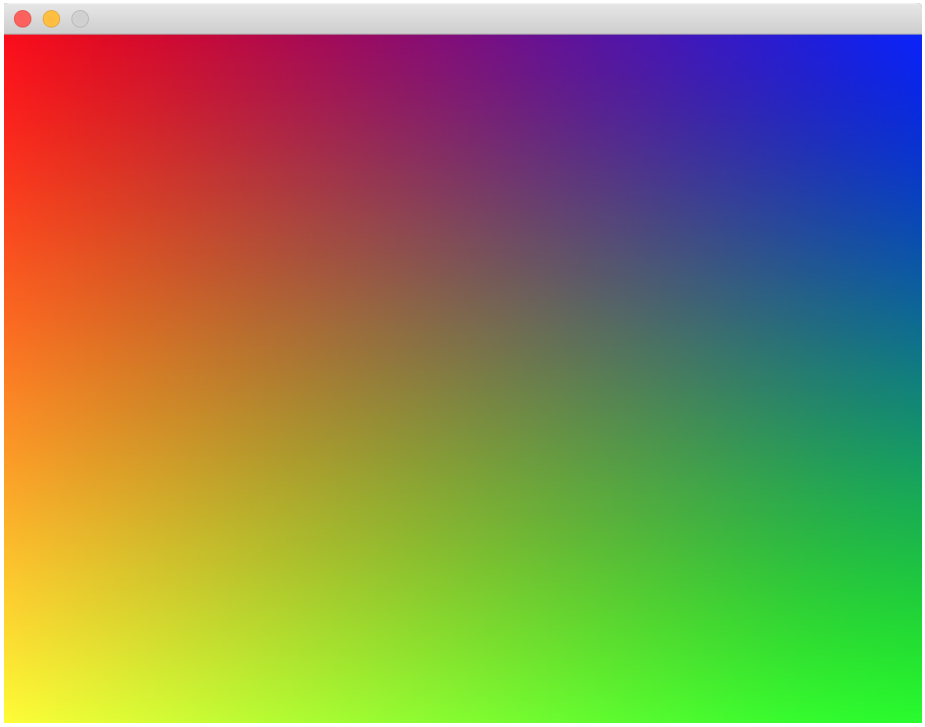


Figure 1: Bilinear Interpolation of Colors

The result of part 2 is shown as figure 2.



3. Starfield

The starfield effect is created by modeling a set of 3D points that moves through space (in time) towards the camera. First, initialize the locations of all stars:

```
//loop through all stars and initialize their locations
 for( int i=0; i < stars.size(); ++i){
     // this will produce a random number between 0 ~ 1
     float starX = float(rand()) / float(RAND_MAX)  ;
     float starY = float(rand()) / float(RAND_MAX);
     float starZ = float(rand()) / float(RAND_MAX);

     // set the value between -1 ~ 1
     stars[i].x = starX * pow(-1, floor(starX*10));
     stars[i].y = starY * pow(-1, floor(starY*10));
     stars[i].z = starZ;
```

3.1 Projection - Pinhole Camera

The vertical and horizontal field of view can be written:

$$vfv = 2 * arctan(\frac{H/2}{f})$$

$$hfv = 2 * arctan(\frac{W/2}{f})$$

If the focal length of the camera is set to $H/2$, the horizontal field of view is equal to 106.26.

3.2 Motion

To let the stars "fly" towards us we need to change their `z` value every frame. Therefore we have an `Update()` function to do that.

```
void Update(){
    int t2 = SDL_GetTicks();
    float dt = float(t2 - t);
    t = t2;

    for(int s=0; s<stars.size(); ++s){

        stars[s].z -= VELOCITY * dt;
        //update of stars

        //If the stars are outside of the volume specified in
        // equation 2 after the update they should be wrapped
        // around to the other side
        if(stars[s].z <= 0){
            stars[s].z += 1;
        }
        if( stars[s].z > 1){
            stars[s].z -= 1;
        }
    }
}
```

For the motion blur effect, I created two global arrays to store the stars positions in the previous two frames. So at every frame, the program will draw the stars in three different position (the current frame as well as the previous two frames with a faded color), as seen in Figure 3.

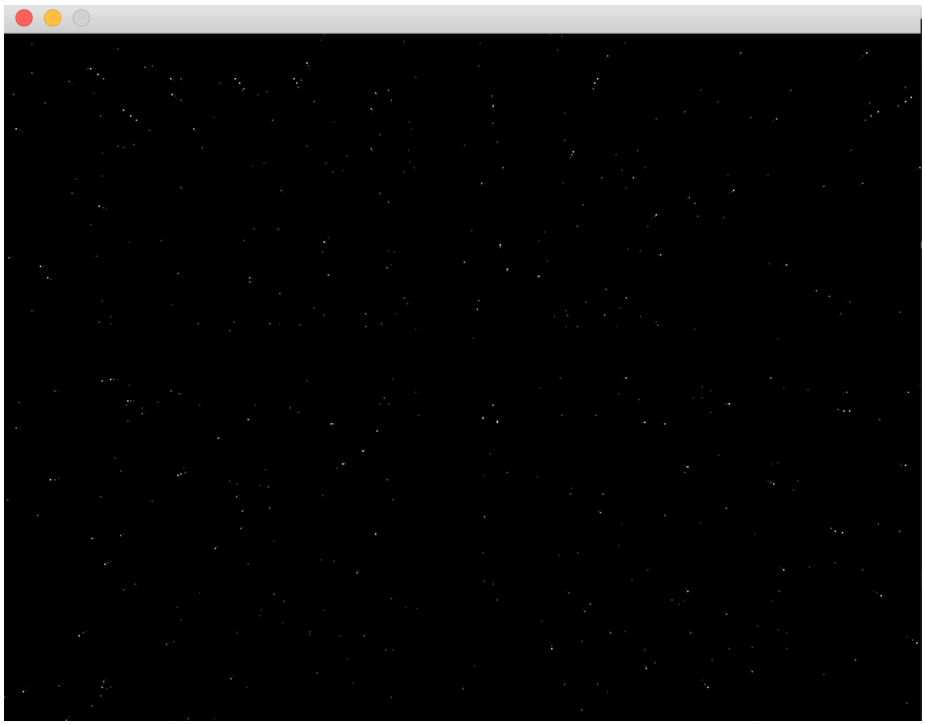


Figure 2: Final Result