Lab 1: Linear interpolation

DH2323 Computer Graphics and Interaction

Author: Mohammad Javad Ahmadi

April 03,2017

1. Introduction to 2D computer graphics

  Our first task in this part of the lab was installing the SDL library in order to be able to use the existing skeleton code to draw colors on the screen.

Getting the program to work on Visual Studio turned out to be such a painful task therefore I decided to go with using Linux which worked like a charm and took couple of minutes to get the blue color picture on the screen.

2. Linear Interpolation

  After playing around with the code and printing out different colors on the screen it was time to write the interpolation function which given two 3D vectors as the starting and ending points as arguments alongside a vector with specified length, adds each step of the interpolation as elements of the vector filling it up.

To write a simple interpolation function we'll need to find the slope of the linear interpolation.

We have (vec3 a, vec3 b, vector<vec3>& result) as input and what we need to do is that depending on the size of our resulting vector we want to figure out the slope and interpolate through a to b filling every element of the resulting vector.

The formula which we use for finding the slope in each dimension, for example the **slope in the x axis** between two vectors is:

Difference or slope = x of b – x of a / length of result – 1;

By this formula we find the change which interpolation function needs to perform in each axis at each step and then after setting the first value “a” as our result vector’s first element then we loop through each element in the result adding a new value with regards of our slope.

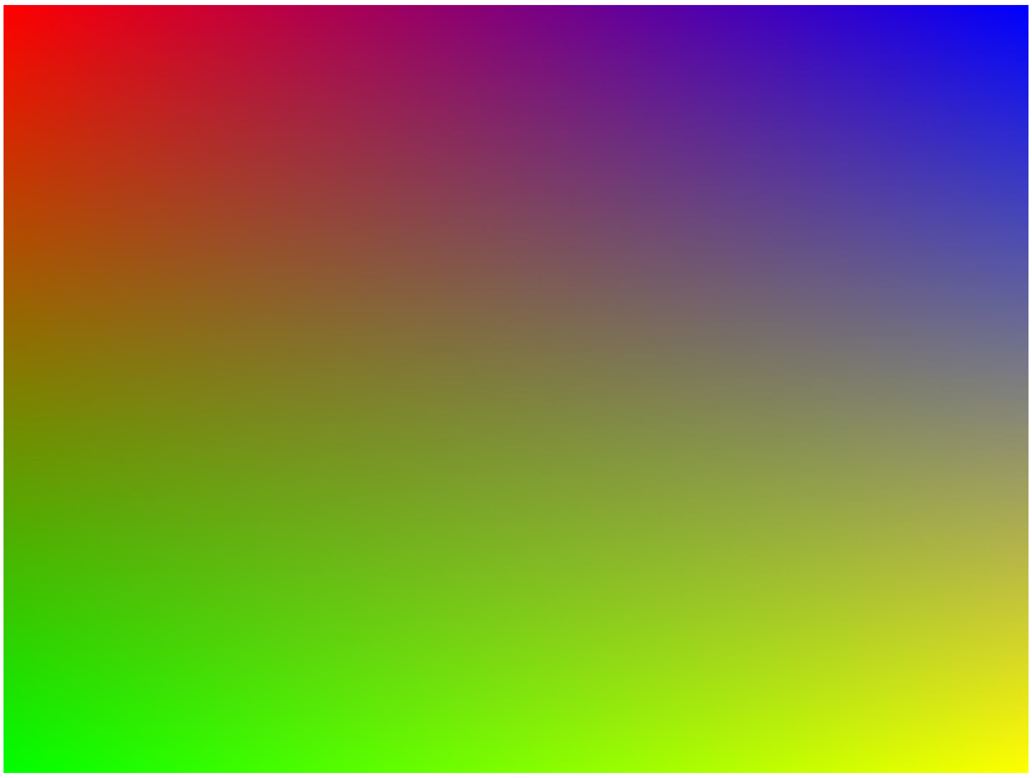
3. Interpolate colors across the screen

Now that we have a working interpolation function we want to use it to interpolate colors across the screen making a rainbow effect. We set a color for each corner of the screen and then interpolate from the top left color to the bottom left color and saving our interpolation in a vector called leftSide which has the size of the height of our canvas. We do the same thing for the rightSide inside of our draw function.

Now we need to interpolate colors from each element of our LeftSide vector to the opposite element of our rightSide vector. Saving the results in a vector which is as big as the width of our canvas.

Now what matters here is to know where we should place this very last interpolation. At first I made the mistake of placing it inside the loop for our x axis in the canvas while drawing as I wanted to use it to draw the pixels on the screen but the result was too slow and the cpu was working really hard to render the image. Then I figured out that the interpolation was happening over and over again because it was in two loops when it really didn’t need to be. So I moved it up to the loop for y axis “ for(int y=0; y<SCREEN_HEIGHT; ++y) “ and as our interpolation needs two points on the same axis in our case here so it made the code

run faster and draw the picture faster resulting in this beautiful image.



4. Projection

The first step in our next part of the lab is initiating and positioning some random stars as 3D points which will be drawn later on a canvas with black background and creating the effect of the stars moving towards the screen making it look like we are traveling along the space.

We then use the projection formula in our draw function in order to draw each star on the galaxy in perspective to our camera which is set at the origin of the screen with a focal value of screen height over two “ f = H / 2” which results in a vertical and horizontal field of view of 90 degrees.

Using our focal length and dimensions we can calculate the angular field of view by the following formula.

$$\alpha = 2 \arctan \frac{d}{2f}.$$

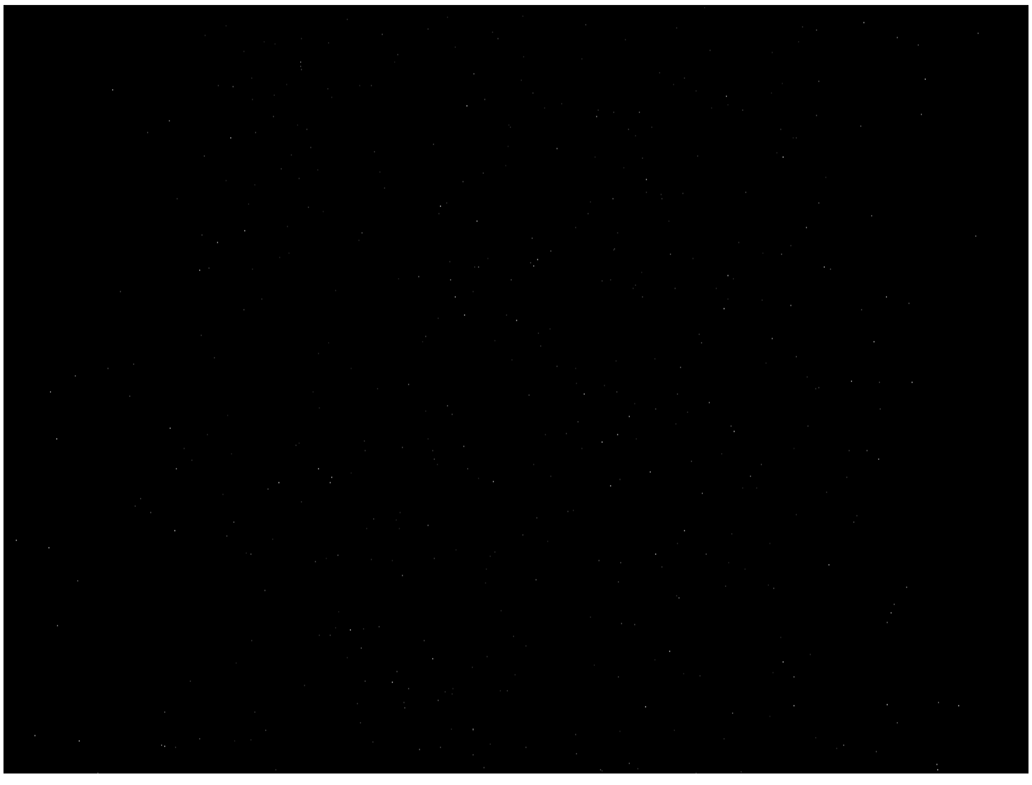
By using the projection formula as below we draw our stars on the screen.

$$u^i = f . \frac{x^i}{z^i} + \frac{Width}{2}$$

$$v^i = f . \frac{y^i}{z^i} + \frac{Height}{2}$$

5. Motion

Lastly we need a function which updates the position of each star by a given time and velocity so that when we re-draw them they would pear in their new position. Repeating this procedure over and over again, updating the position and drawing and reseting the position of the star when it leaves the screen makes an effect which seems that the stars are moving towards you endlessly.



6. Summary

In this lab we learned how to draw colors on the screen and interpolate between the colors to make a rainbow effect picture across our canvas. We also learned how to use the projection formula to render 3D points on our 2D canvas and animate them.