Stockholms universitet

Laboration 1:
Introduction to 2D and 3D Computer Graphics

Danilo Catalan Canales

Daniloc@kth.se

DH2320 Introduction to Computer Graphics and Visualization

Content

Lab Introduction

The main goal of this lab was:

- To introduce computer graphics in 2D:
    - Image Representation
    - Colors
    - Pixels
    - Linear Interpolation

- To introduce computer graphics in 3D:
    - Using a “Pinhole Camera”
    - Projecting 3D points into 2D points
    - Starfield Effect, where there is lots of 3D points moving towards the observer.

Setup

First off, we had a number of choices on how to do the lab. I chose to use my own stationary computer with linux set up, where I used the linux terminal to run the source code provided. We used the SDL v.1.2. library (http://www.libsdl.org/) that we had to install and GLM (http://glm.g-truc.net/) library that was included in the source code directory.

Introduction to 2D Computer Graphics

We were given a “skeleton” program which was used as a base for the lab. We were encouraged to fill in our frame with different colors, to experiment a bit with the program before getting into the real assignment.
“Interpolation, a method of constructing new data points within the range of a discrete set of known data points”. This was our first assignment, to be able to store an amount of data between two points, including the given range. By computing the amount that we “skip” every step from the lower range to the upper range, we can calculate every point in between. The formula I made was:

$$(1) \quad \frac{x_2 - x_1}{n - 1}$$

We take the difference between the range and divide it by the number of points we want to store in between. However, it excludes the upper range from the equation, so we subtract a point from the number of points we want in between to add room for the upper range.

When we had this sorted out next part was to compute 4 secondary colors on each corner and interpolate between them for every pixel on the frame. So first off we adapt this to our vectors, where we compute for the vector coordinates. We first interpolate the left side of the frame, top to bottom, and then the right side. From there we Interpolate between the two vectors, to get the interpolated values from the left side to the right side.

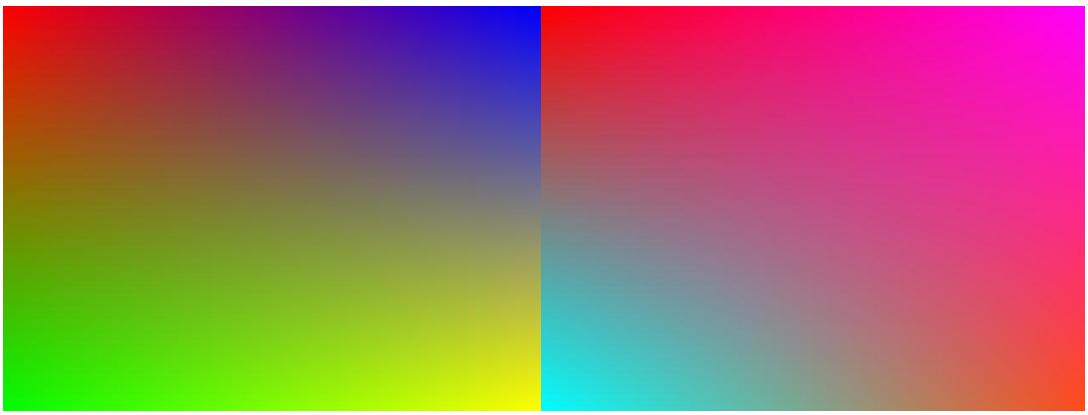


Figure 1: Some examples of my interpolated color vectors.

Introduction to 3D Computer Graphics

As introduction to 3D computer graphics we will be looking at geometry and in specific projections, as well as how to update parameters within the framework to create an animated image. So for this we will be creating a implementation on the star field effect. We make use of the “skeleton”-source code that we’ve been given for the previous part of the lab as a base for the next part of the lab.

If the “camera” is directed along the z-axis, then we can use this equation for a point $p^i = (x^i, y^i, z^i)$:

$$x_t^i = x_{t-1}^i$$
$$y_t^i = y_{t-1}^i$$
$$z_t^i = z_{t-1}^i - v * dt$$

Equation 1: Velocity, v, times the elapsed time, dt, will update the position of the pixel.

Now, why is the updated position negative? It’s because we’re watching along the positive line of the z-axis which means when the pixels go in a negative direction, they go towards us, which is the effect we’re trying to create.

Following the instructions, we create a global variable that stores the location of the stars. After that we randomly generate the start position of every star within the range of:

$$-1 \leq x \leq 1$$
$$-1 \leq y \leq 1$$
$$0 \leq z \leq 1$$

Spawn limits for the coordinates

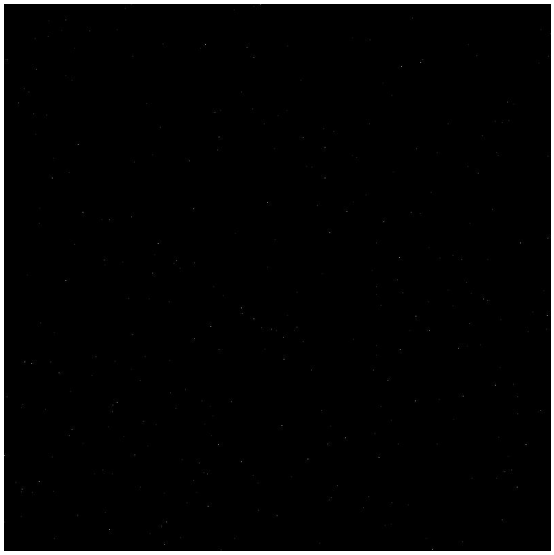
The range for the x and y coordinates tell us where in the frame they are allowed to spawn, and the range for z will be the “distance”. This will lead us to our next equation which will compute the updated position as the pixel travels towards the camera. Our equation:

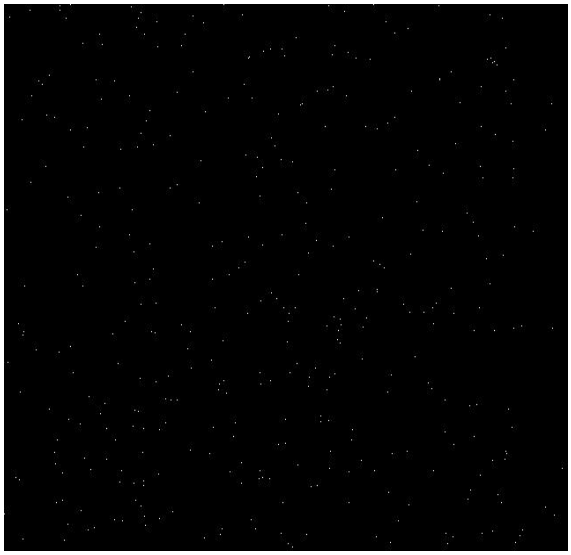
$$f = \frac{H}{2} \qquad u^i = f * \frac{x^i}{z^i} + \frac{W}{2} \qquad v^i = f * \frac{y^i}{z^i} + \frac{H}{2}$$

Equation 2: Projecting 3D coordinates to 2D coordinates

Where f is the focal length of the camera. W and H are the width and height of the frame, which takes us to the center of the frame.

The equations given where very straightforward and easy to comprehend. So all I had to do was to implement them in the code. I updated the draw function as instructed and implemented the equations accordingly to set the location for every star on a frame. Now, that was only for the first part. For the second part we had to update the first equation to make the stars move towards the camera. By adding another limit for the z value, we can reset the z value whenever it reaches the camera so we create an effect of infinite stars passing by! So we loop these limits and update z using Equation 1, and also putting the velocity value to 0.001 to not make the millenium falcon go too fast! The results:





We were asked to try and calculate the field of view, when the Vertical Field of View is 90 degrees. After doing some research I calculated the Horizontal Field of View to be... 90 degrees! Since we have a 500x500 dimension canvas.

$$H = 2arctan(tan(\frac{V}{2}) * \frac{w}{h})$$