Computer Graphics through C++ and Linear Algebra

Parker Greene

October 21, 2018

1 Introduction

The field of computer graphics studies methods by which the computer displays a scene to a computer screen. In the most common case, we can think of the scene as the collection of many objects in 3 dimensional space. Computer graphics algorithms are responsible for taking information about that scene (i.e. the location, color, and orientation of each object) and creating a 2 dimensional image to display on the screen. The field encompasses a large range of sub-disciplines and techniques, but at the very heart of computer graphics is linear algebra. In every aspect of computer graphics, vectors, matrices, rotations, transformations, etc. are present.

This course is meant to introduce the theoretical and practical linear algebra knowledge necessary for the study of computer graphics. It will also involve step-by-step implementation of the math and algorithms involved.

All of the implementation will be done in C++, however, no prior C++ or linear algebra knowledge is required or expected. Knowledge of another object-oriented language is helpful, but not required. The course will cover the basics of C++ as we walk through the implementation of a linear algebra library in Chapter 2.

2 Basics of Linear Algebra

For the purposes of a course concerning computer graphics, we will only need to worry about 3 dimensions and lower. Linear algebra can extend into an arbitrarily large number of dimensions, but since we have no way of visualizing 4 dimensions and higher, we won't bother worrying about those!

2.1 3-Dimensional Space

In two dimensional space, we have 2 axes: x and y. If we treat a flat piece of paper as our 2D space, then we could say that the horizontal direction is the x axis and the vertical direction is the y axis. Note that the axis directions are perpendicular to one another. When we add another dimension, we get a new axis, z, which extends directly out of the paper. The z axis direction forms a right angle with the paper, which means it also forms a right angle with the x axis direction and the y axis direction.

2.2 Vectors

2.2.1 Representation

A vector in n dimensions is a collection of n numbers. Each element of the vector defines the amount of weight the vector has on an axis. For three dimensions, we will always order the elements of the vector in the order x, y, z. In this course, we will notate a vector with an arrow above as such: $\vec{u}$. In computer graphics, vectors are primarily going to be used to represent **position** and **direction**. These three axes,

x, y, and z form the **basis** of 3D space, which means all 3D vectors can be made from a combination of these axes. The most common form of representing a vector is with three numbers in the following form:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

This is a column vector, which is written vertically. However, in this course, we will represent it with the notation (x, y, z) to mean the same thing, unless otherwise specified.

For a **position** in 3 dimensional space, we can store the three numbers that correspond to the placement of an object with respect to an **origin**, which is just our reference. Lets say that your eye was the origin, and some object was two meters to our right. We'd say that object is at position $(2, 0, 0)$. If the object was instead two meters to our left, then it would be at position $(-2, 0, 0)$. Similarly, if the object was one meter directly above us (this is the y direction), then its position is $(0, 1, 0)$. You can also see that your eye (the origin) is at position $(0, 0, 0)$.

The **direction** of an object can also be represented with a vector of three components with respect to the origin. In general, vectors representing directions should always be of length 1, so even if the object is at position $(0, 5, 0)$, its direction is $(0, 1, 0)$. The length of a vector is called its **magnitude**, which is discussed in the following section. Using the same example as above (with our eye being the origin), an object to your right is in the direction $(1, 0, 0)$, up is direction $(0, 1, 0)$ and forward is $(0, 0, -1)$.

Wait, wait, wait, why is that last one negative instead of positive? This is something we call the **handedness** of our coordinate system. For the purposes of this course, we will try to always use a **right-handed** coordinate system, with positive x being to the right, positive y being upwards, and positive z being behind. We will discuss this more later, but for now, just remember these directions.

2.2.2 Operations

Let's define some common vector operations that we will use throughout the course.

The **magnitude** of a vector can be thought of as its Euclidean length. Euclidean length is defined as the square root of each element of the vector squared. We will notate magnitude with the symbol $||\cdot||$. More formally,

$$||\vec{u}|| = \sqrt{u_x^2 + u_y^2 + u_z^2}$$

Where u_x is the first element of u, u_y is the second element, and u_z is the third. This is just a formal way to define ——.

The **dot product** operates on two vectors and computes the sum of each corresponding element multiplied with each other. We will use the notation $\cdot$ to mean dot product. More formally, the dot product of the three dimensional vectors $\vec{u}$ and $\vec{v}$ is

$$\vec{u} \cdot \vec{v} = u_x * v_x + u_y * v_y + u_z * v_z$$

Normalization on a vector turns it into a vector of magnitude (length) 1. To do this, we divide each component of the vector by the vector's length. Formally, to normalize a vector u:

$$u_{normalized} = \frac{u}{||u||}$$

The **angle** between two vectors, u and v is computed

The **cross product** operates on two vectors and computes a new vector. We will use the notation $\times$ to mean the cross product. Formally, we define the cross product between two three dimensional vectors $\vec{u}$ and $\vec{v}$ as

$$\vec{u} \times \vec{v} = ||\vec{u}||||\vec{v}|| \sin\theta \vec{n}$$

Where $\sin\theta$ is the angle between the vectors u and v, and $\vec{n}$ is a vector perpendicular to both u and v.