

1 Announcements

- Homework Project #1: Due Thursday

2 Double Buffering

Ever see your images flickering while they are redrawing? Why does this happen? How can you stop it?

The flickering is due to the fact that when you start to redraw a new frame, you usually first clear the screen, then redraw the image, but while that is all going on the monitor is still trying to display something on the screen. So, what does it show? Your current progress, which could be anywhere from the previous image to a blank screen to the first portion of your image. The result, huge flickering.

Double buffering is an excellent yet simple strategy to remove flickering. All drawing is done on a separate “canvas” (buffer). When you are done drawing your “scene,” you then tell the system (monitor) that the new canvas should be displayed instead of the old one. The result, the user either sees the previous one or the new one (once it is done) and none of the work in the middle, no flickering. The reason it is called double buffering is that you only need two separate canvases to do the job. The visible buffer and the current (background) buffer. When the current buffer is finished, you swap the two buffers and work on the new current buffer which was the old visible buffer. That’s about all there is to it.

The one drawback, you can still get a jumping feeling to the frames as there may be a long pause from one frame until the next one. This is only avoided by making the computer or algorithms faster, or preprocessing the whole scene in advance like a movie!

In OpenGL’s GLUT extension library the process to do Double Buffering is quite simple (see page 91 for further details):

```
// Use the following in place of GLUT_SINGLE
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);

// Whenever you are ready to display your NEW screen use
glutSwapBuffers();
```

IMPORTANT NOTE: Keep in mind that, when using double buffering, your new image will *not* show up until after you do the swap buffers command. So, if you forget to do it, the user will continue to see your old frame - which may be blank or garbage!

3 Vectors

When playing the latest animated video game or watching the most recent blockbuster CGI film, you should always keep in mind that every image you see is *nothing more than a bunch of numbers*. Every little pixel you draw boils down to a number. So, it should not be surprising that graphics relies heavily on... **Mathematics**.

One of the most powerful and simplest tools at our disposal is the vector. Vectors are basically directions, or displacements from one point to another. They can be used in any dimension and most properties of vectors apply to all dimensions. Of course, our focus will deal mostly with two-, three-, and **four-** dimensional vectors.

3.1 Coordinate system

It is important when dealing with vectors to realize that they are represented and displayed relative to some coordinate system. Each coordinate system has an origin and d axes (for d -dimensions). The main ones are two-dimensional (like a grid) and *right-handed* and *left-handed* systems. The distinction comes by making a fist with the right-hand curling the fingers from the x -axis to the y -axis, the thumb points in the direction of z . With the left-handed system, you use your left-hand to indicate the direction of z .

3.2 Properties Highlighted

To make it easier to identify the various and numerous properties of vectors, we shall look at a brief list of them for reference use.

- Symbols like P and Q represent points
- Bold faced letters like \mathbf{v} and symbols like \vec{v} represent vectors. These notes shall use the latter notation.
- **Difference:** $\vec{v} = Q - P$.
Difference in two points is a vector.
- **Sum:** $P + \vec{v} = Q$.
A point plus a vector is another point.
- **Representation:** Either: (**row matrix** form) $\vec{v} = (1.2, 3.4, 5.6)$ or (**column matrix** form) $\vec{v} = \begin{pmatrix} 1.2 \\ 3.4 \\ 5.6 \end{pmatrix}$.
- **Scalar multiplication:** $s\vec{v} = \vec{w}$.
Mutiplying a vector by a scalar, or real number, is a vector.
E.g., $2(1, 2, 3) = (2, 4, 6)$.
- **Vector addition:** $\vec{a} + \vec{b} = \vec{c}$.
Adding two vectors yields another vector.
E.g., $(1, 2, 3) + (4, 5, 6) = (5, 7, 9)$.
- **Linear combinations:** $a\vec{v} + b\vec{w} = \vec{u}$.
Combining scalar and vector additions. Can be generalized for any number of vectors.
 $\vec{w} = a_1\vec{v}_1 + a_2\vec{v}_2 + \dots + a_m\vec{v}_m$.
- **Affine combination:** A linear combination where the coefficients add up to unity (1).
 $a_1 + a_2 + \dots + a_m = 1$.
Special situation with two vectors: $(1 - t)\vec{v} + t\vec{w}$ is an affine combination.
- **Convex combination:** An affine combination where all the coefficients are also nonnegative.
 $a_i \geq 0$ for $1 \leq i \leq m$.
Notice $a_i \leq 1$ or the combination would not be affine.
- **Set of all convex combinations:** For two vectors \vec{v}_1 and \vec{v}_2 , the set would be all vectors formed by convex combinations of the two vertices:
 $\vec{v} = (1 - a)\vec{v}_1 + a\vec{v}_2$ for $0 \leq a \leq 1$.
This notion can also be extended to any number of vectors.
- **Magnitude:** Let $\vec{v} = (v_1, v_2, \dots, v_d)$, then $|\vec{v}| = \sqrt{v_1^2 + v_2^2 + \dots + v_d^2}$.
Notice this roughly tells you the **length** of the line segment defining the vector. That is if $\vec{v} = A - B$ then $|\vec{v}|$ is the distance between A and B .

- **Unit vector:** $\hat{v} = \frac{\vec{v}}{|\vec{v}|}$.

Notice that $|\hat{v}| = 1$ which is why this is called the *unit* vector. This process is known as **normalizing** the vector.

- **Dot Product:** $d = \vec{v} \cdot \vec{w} = \sum_{i=1}^n v_i w_i$

$$(2, 3) \cdot (4, 5) = (2 * 4 + 3 * 5) = 23.$$

Very useful with many properties and applications.

- Symmetry: $\vec{a} \cdot \vec{b} = \vec{b} \cdot \vec{a}$
- Linearity: $(\vec{a} + \vec{b}) \cdot \vec{c} = \vec{a} \cdot \vec{c} + \vec{b} \cdot \vec{c}$
- Homogeneity: $(s\vec{a}) \cdot \vec{b} = s(\vec{a} \cdot \vec{b})$
- $\vec{a} \cdot \vec{a} = |\vec{a}|^2$

- **Angle between two vectors:** The angle between two vectors can be computed as follows:

$\cos(\theta) = \hat{v} \cdot \hat{w}$. Notice the vectors are *normalized*.

- **Perpendicularity:** If $\vec{v} \cdot \vec{w} = 0$ then the two vectors are **orthogonal** or **normal** or **perpendicular** to each other.

If > 0 then angle is less than 90 degrees and if < 0 then angle is greater than 90 degrees.

$\vec{i}, \vec{j}, \vec{k}$ represent the three **standard unit vectors**: $(1,0,0)$, $(0,1,0)$, $(0,0,1)$.

Note: $(a, b, c) = a\vec{i} + b\vec{j} + c\vec{k}$. This will come in handy later.

- **Perp vectors:** $\vec{v} = (v_x, v_y)$ then $\vec{v}^\perp = (-v_y, v_x)$ is the **counterclockwise perpendicular** to \vec{v} .

How many unit vectors are perpendicular to \vec{v} in 2D? How about 3D?

- **Projecting a vector:** This is known as orthogonally projecting a vector to another vector. Figure 4.13(b) on page 158 illustrates this idea. In this case, we wish to project the vector \vec{c} onto the vector \vec{v} resulting in the vector $K\vec{v}$. Notice that this new vector is in the same direction as \vec{v} just with a different magnitude.

- **Resolving a vector:** Resolving a vector into its components is decomposing it into the sum of two other vectors. Again, in Figure 4.13(b), we wish to compute: $\vec{c} = K\vec{v} + M\vec{v}^\perp$.

- **Computing the distance between a point and a line:** The distance from a point to a line is shown in Figure 4.13(a-b). Here the distance from C to the line L is equal to $|M\vec{v}^\perp|$.

Solving the above three properties should be, without much surprise, related to each other. After doing a bit of math, we see that

$$\begin{aligned} \vec{c} &= K\vec{v} + M\vec{v}^\perp \\ \vec{c} \cdot \vec{v} &= K\vec{v} \cdot \vec{v} + M\vec{v}^\perp \cdot \vec{v} \\ K &= \frac{\vec{c} \cdot \vec{v}}{\vec{v} \cdot \vec{v}}, \text{ similarly,} \\ M &= \frac{\vec{c} \cdot \vec{v}^\perp}{\vec{v} \cdot \vec{v}} \end{aligned}$$

- **Reflections:** are computed easily by using the projection property above. This is useful for such things as simulating a billiard ball on a pool table. As the ball hits the objects, e.g. a side, it must bounce back. The process, ignoring spin and other such properties, is simulated by computing its reflection. Another use is in the reflection of light in a mirror or room of mirrors. This will most likely be a homework assignment so you will get practice with this one. This information can be found on pages 159-160, but the end result is that if \vec{a} is the initial ray and \vec{n} is the normal to the surface (that is being reflected off of), then \vec{r} is the reflected ray and has value:

$$\vec{r} = \vec{a} - 2(\vec{a} \cdot \hat{n})\hat{n}$$

- **Cross Products:** The cross product of two 3D vectors is quite useful in our graphics toolkit as we shall soon see. It is computed as follows:

$$\vec{a} \times \vec{b} = (a_y b_z - a_z b_y)\vec{i} + (a_z b_x - a_x b_z)\vec{j} + (a_x b_y - a_y b_x)\vec{k}$$

Whew! Quite a bit to remember. Fortunately, there is an easier way to learn this by using determinants (if that helps any). This illustration shall be shown in class.

Recall that $\vec{i} = (1, 0, 0)$, etc... There are some properties that you should be aware of with cross products as one is different from the dot product property.

- $\vec{i} \times \vec{j} = \vec{k}$,
- $\vec{a} \times \vec{b} = -\vec{b} \times \vec{a}$,
- $\vec{a} \times (\vec{b} + \vec{c}) = \vec{a} \times \vec{b} + \vec{a} \times \vec{c}$,
- $(s\vec{a}) \times \vec{b} = s(\vec{a} \times \vec{b})$.
- The vector $\vec{a} \times \vec{b}$ is perpendicular to both \vec{a} and \vec{b} . Why?
- Figure 4.15 (page 162) shows a good illustration of what the cross product looks like geometrically. In particular, keep in mind that the magnitude of the cross product, $|\vec{a} \times \vec{b}|$ computes the *area* of the parallelogram formed by the two vectors \vec{a} and \vec{b} .

- **Computing Normals:** As we saw in reflection and, in many other properties in graphics, the normal of a face is extremely important to know. However, when developing models it is not easy to simply compute the normals by hand for each face of the object. To the rescue comes the cross product. If \vec{a} and \vec{b} are two vectors on the plane (notice that any two vectors that are not scalar multiples of each other define a plane), then the normal vector is simply $\vec{n} = \vec{a} \times \vec{b}$, or any scalar multiple of \vec{n} .

4 Representing Geometric Objects

This is hard to explain without some good illustrations so you should probably refer completely to section 4.5 for this information. But, to highlight, it is important to realize that vectors, points, lines, planes are usually stated as numbers but these numbers mean very little unless we have a frame of reference. This reference is the coordinate frame that the objects lie in. There can be multiple coordinate frames, different ones for different objects. And moving the individual frames around can move certain models around in a larger coordinate system.

The key element to remember about this section is the introduction of what are called **homogeneous representations**. Here we represent vectors and points as they truly are with one extra value. For example, the vectors and points in 3-D are represented as follows:

$$\vec{v} = (v_1, v_2, v_3, 0)$$

$$P = (P_1, P_2, P_3, 1)$$

The fourth “coordinate” represents if the object is with respect to a given origin or not. Points depend on the location along each of the three axes plus the origin. Whereas vectors, which are directional rather than locational, depend on simply the direction with respect to the three axes and do not rely on the location of the origin at all. To see this, imagine that the origin is relocated, but the directions of each axis remain the same. In the global world, the point P should move, but the vector \vec{v} should still remain in the same direction, it has no location.

This property will be exploited much more when we get to 3-D moving and projections.

4.1 Affine Combinations of Points

Given the new homogeneous representation we can now illustrate how to combine two points in a linear combination.

$$fP + gR = (fP_1 + gR_1, fP_2 + gR_2, fP_3 + gR_3, f + g)$$

Notice that in order for this to remain a proper point, $f + g$ must equal 1. This is known as an **affine combination**. A similar combination yields the following property:

$$P = A + t(B - A), P = tB + (1 - t)A$$

That is a point is a combination of one point and the scaled vector $t(B - A)$. The uses for this?

- Computing the **centroid** of a triangle, its center of gravity. See Example 4.5.1 (pg. 169) for information on this.
- Linear interpolation of Two Points:
as we mentioned in parametric curves, the process of ranging t from 0 to 1 computes the line segment from A to B . This is known as interpolating the line.
- Tweening:
A very cute benefit is the ability to extend this to multiple polylines. By interpolating a polyline drawing from one to another you get a simple “morphing” effect that looks very interesting (depending on the mapping of the vertices between each other). That is, you draw a new polyline where each point is *between* the corresponding two points from the original images. Figure 4.22 (page 171) illustrates this technique very effectively, as do the subsequent figures.
- Interpolation vs. Extrapolation:
Picking a value of t between 0 and 1 (inclusive) is considered **interpolating** between the two points. Picking a value outside of this range, either negative or greater than 1, is called **extrapolating** because you are going away from both images (in one direction or the other). Otherwise, the two are identical concepts - the images are just different.
- Quadratic and cubic tweening and Bezier Curves
We will be hopefully covering this topic later in the semester, depending on how fast we get through topics. Read this very short section for a warmup of the topic to come.

4.2 Lines and Planes

Parametric formulas are also useful in representing lines (as we have seen) and planes (in any dimension - but mostly three-dimensions). First some terminology to keep everyone up to speed:

- **line**: Defined by any two points and infinite in length. Hopefully, I don't have to explain more than this.
- **line segment**: The portion of a line between **two** points.
- **endpoints**: The two points defining the boundary of the line segment.
- **ray**: Half of a line defined by **one** endpoint and a direction. It extends from that endpoint to infinity (or negative infinity).
- **plane**: The line extended to higher dimensions. The plane is uniquely defined by any three non-collinear points (why is this)?
- **co-planar**: A set of points is considered co-planar if all the points in the set lie in the same plane. Notice that any set of 3 or fewer points is by definition co-planar.
- **co-linear**: Same except the points all lie on same line.

Lines, line segments, and rays can all be represented using the same parametric formula. Let C and B be two endpoints defining the line, etc. and let $\vec{b} = B - C$. Then $L(t)$ represents the point on the line, segment, etc. at position t and is defined as follows:

$$L(t) = C + (B - C)t, \text{ or } L(t) = C + \vec{b}t$$

. The range of t determines the particular type of object.

- If t has no bounds, it is the entire **line**.
- If t ranges from 0 to 1, it is the **line segment**. Notice that $L(0) = C$ and $L(1) = B$.
- If t ranges from 0 to positive infinity, we have the **ray** extending from C in the direction towards B .

Notice that this version has no dependency on the dimension. That is, it can also represent a line, line segment, or ray in three dimensions.

The **point normal form** is another way to represent a line but since it is really only convenient in two-dimensions we will not focus on it but we *will* be using it often to represent a plane. Just for information, the point normal form of a line is defined by two values, an endpoint C and a normal \vec{n} to the line. In 2-D, any value R satisfying the following equation is considered to be on the line:

$$\vec{n} \cdot (R - C) = 0$$

Typically if \vec{b} is the vector defined by two endpoints on the line (as noted earlier), then \vec{b}^\perp would be a suitable (but not the only) normal.

4.2.1 Planes

We can of course extend the two above representations for planes. In particular we have the following:

- **Parametric form:**
A point C and two (nonparallel) vectors \vec{a} and \vec{b} lying in the plane. Then all points $P(s, t)$ in the plane are defined as follows:

$$P(s, t) = C + \vec{a}s + \vec{b}t$$

Notice that the points on the plane have two parametric values, s and t , corresponding to the two directions (“axes”) on the plane.

- **Point Normal Form:**
A point C and a normal \vec{n} to the plane. Then, in 3-D, all points P on the plane satisfy the equation:

$$\vec{n} \cdot (R - C) = 0$$

Planar patches represents just a small (parallelogram) region of the plane which is easily defined using the parametric form by bounding s and t (*usually* to be between 0 and 1). Notice it is the extended version of the line segment. Planar patches will become useful when (or if) we get into parametric curved surfaces and surface modelling.

5 Line Segment Intersections

Vector properties and the parametric form help us find the intersection of two lines or line segments. This is an essential operation as we have already seen with clipping but its uses go far beyond this. Let AB (resp. CD) be the line defined by two endpoints A and B (respectively C and D). And let $\vec{b} = B - A$ and $\vec{d} = D - C$. In parametric form we have:

$$AB(t) = A + \vec{b}t, \text{ and } CD(u) = C + \vec{d}u$$

Notice that we have two different values of t and u , to indicate that they are different parameters. The intersection occurs simply at the point where t and u satisfy:

$$A + \vec{b}t = C + \vec{d}u$$

. Let $\vec{c} = C - A$. We now get:

$$\vec{b}t = \vec{c} + \vec{d}u$$

There are two cases,

- They are not parallel, then $\vec{d}^\perp \cdot \vec{b} \neq 0$ and we can do the following:

$$\vec{b}t \cdot \vec{d}^\perp = \vec{c} \cdot \vec{d}^\perp + \vec{d}u \cdot \vec{d}^\perp$$

$$t = \frac{\vec{c} \cdot \vec{d}^\perp}{\vec{b} \cdot \vec{d}^\perp}$$

We can do the same to find the solution to u yielding:

$$u = \frac{\vec{c} \cdot \vec{b}^\perp}{\vec{b} \cdot \vec{d}^\perp}$$

To see if the two line segments intersect then t and u would have to both be between 0 and 1. Same idea works for intersection of two rays. Plugging t into the formula for $AB(t)$ yields the intersection point (on the line, line segment, or ray).

- They are parallel
Hmmm... an intersection here only happens if the two lines are actually identical! For example, if A lies on the line segment CD . Solve it in a similar way as above.

6 Further Reading

Since I can't lecture about *everything* in class. You are also responsible for the following concepts:

- Computing the circle defined by three points.
This simply uses the notion of line segment intersections to compute the center and then radius.
- Lines intersecting Planes (Clipping)
- The Cyrus-Beck Clipping Algorithm
In particular, the CSC629 students should review this material. It is another powerful approach to clipping lines.