Graphics pipeline, algorithms for 2D and 3D

Daniel Chatfield

November 25, 2014

Warmup Questions

1. Give as many reasons as possible why we use matrices to represent transformations. Explain why we use homogeneous coordinates.

Answer:

Matrices allow arbitrary transformations to be represented in a consistent way. By using matrices, transformations can be concatenated easily by multiplying the matrices together.

Some transformations cannot be represented by a 3×3 matrix. For example if you want to translate a point $p = (x, y, z)$ by an arbitrary translation vector $t = (t_x, t_y, t_z)$ then you have that the point after translation is given by $p' = (x + t_x, y + t_y, z + t_z)$. To represent this in matrix form we need to solve:

$$\begin{pmatrix} x + t_x \\ y + t_y \\ z + t_z \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \times \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

This can only be solved when the translation vector is the zero vector.

To represent these transformations we need to add a forth component and this is why *homogeneous coordinates* are used.

2. Are there any problems with texture mapping onto a sphere?

Answer:

The surface of a cylinder can be mapped onto a 2D coordinate space and thus it is easy to use *UV mapping* to map a 2D texture onto it.

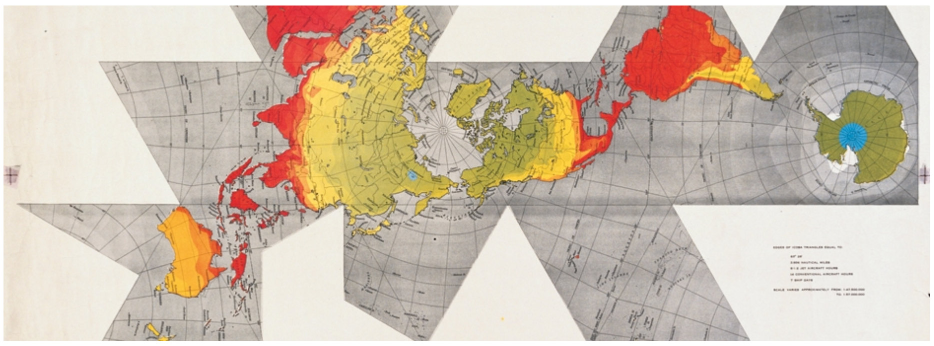
There are several different ways to map a 2D coordinate space onto a sphere:

Naïve approach Directly map long,lat onto sphere with u going round equator and v going from pole to pole. This is simple to implement but wastes texture resolution at the poles and requires that the texture is stretched at the top and bottom to look natural when mapped.

Cubic mapping Have 6 source textures, each of which correspond to the side of the sphere's bounding box. UV mapping is then used to project the surface of the sphere onto the sides of the cube. This has the advantage that there is a more even distribution of texels than the naïve approach and that it is often easy to generate these textures from photos of real spheres, however, because the projection is a non-linear warp it is computationally expensive.

Icosahedral mapping An *icosahedron* is a 20 sided polyhedron, the regular convex icosahedron is a reasonable approximation of a sphere, made from triangles. Pairs of triangular faces can be combined for 10 square source textures, UV mapping can then be used at the corners of each triangular face.

The earth can be arranged such that the land mass isn't split up (Dymaxion Projection):

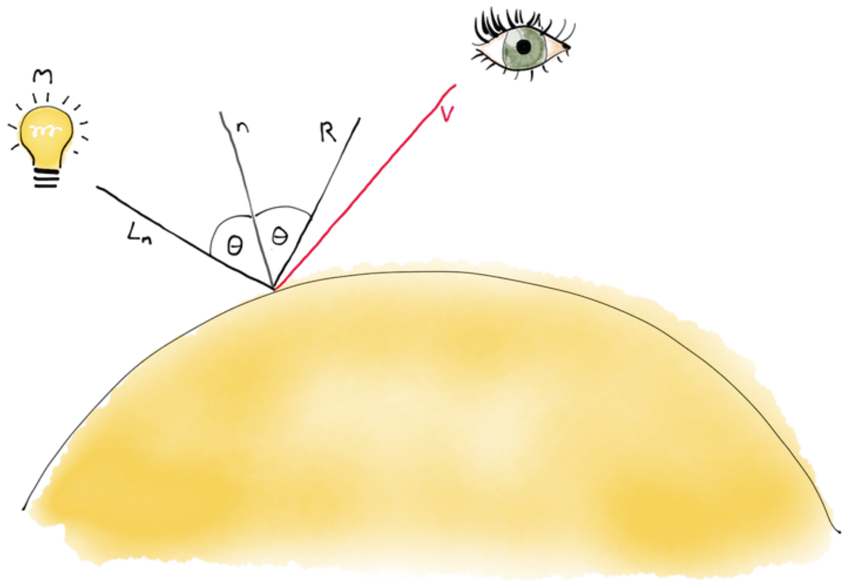


3. Describe how Phong's specular reflection models real specular reflection. Why is it only a rough approximation? Why is it useful?

Answer:

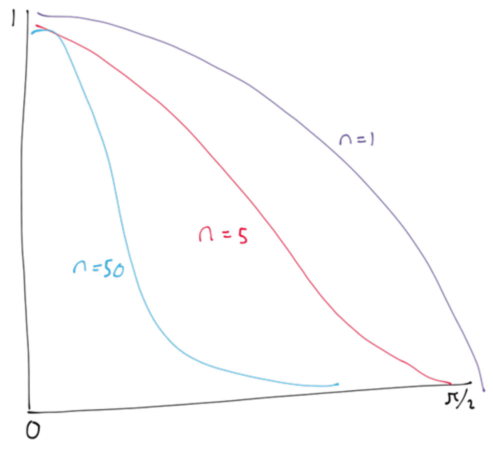
Specular reflection is when the reflection of a light source is more intense at the angle of reflection (in contrast to diffuse reflection where the light is scattered in all directions).

If L_m is the direction of light from a light source m then the direction of specular reflection R is such that the normal to the surface bisects the angle subtended by the two directions.



For a perfect reflector then the reflection will only be visible in the direction R. For real objects (which are not perfect) the light can be seen even if the direction to the viewer V does not coincide with R (i.e. the reflection is visible within a cone around R). The shinier the surface the smaller this cone.

Phong's model lets the intensity of the reflection by modelled by $cos(a)^n$, as this met the physical observation that for a shinier surface (larger n) the intensity drops off more rapidly.



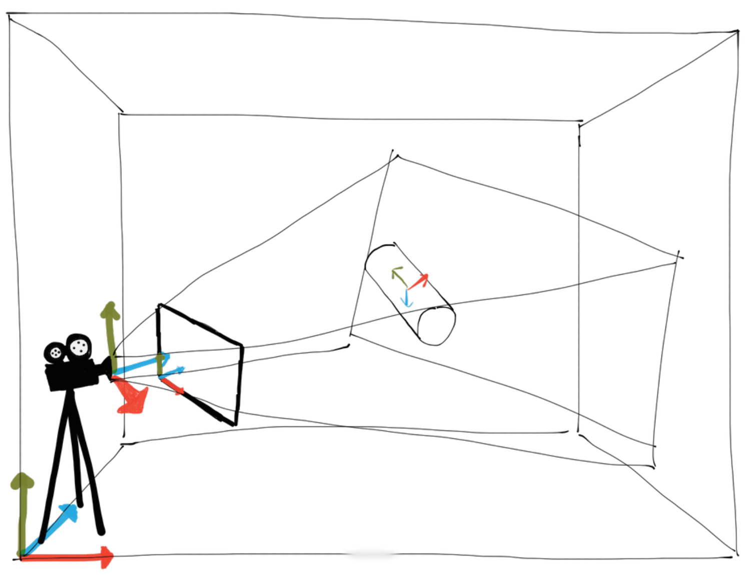
Phong's model was based solely on physical observation rather than physics itself and the true rate at which the intensity falls off may not exactly match any power of $cos(a)$.

True specular reflection is also a function of the incidence angle θ. For example, glass reflects a lot more light when $\theta > 80°$). Some substances (e.g. copper) actually change colour with incidence angle.

Phong approximates the *reflectance distribution function* ($BRDF$) as the specular coefficient (k_s) and assumes it is constant (a reasonable approximation for many materials but not all).

4. Draw pictures to show what is meant by:
   (a) object coordinates
   (b) world coordinates
   (c) viewing coordinates
   (d) screen coordinates

Answer:



5. We use a lot of triangles to approximate stuff in computer graphics. Why are they good? Why are they bad? Can you think of any alternatives?

Answer:

Triangles are guaranteed to be planar which simplifies the maths involved with calculating normals. Any polygon can be split up into triangles.

Not sure about disadvantages

Hexagons?

Longer Questions

3. Draw and explain two different scenes which have the same projection as seen by the viewer. What other cues can you give so that the viewer can distinguish the depth information?

Answer:

Focus can be used to show depth (make items in the background out of focus).

6. We often use triangles to represent a sphere. Describe two methods of generating triangles from a sphere.

Answer:

Not sure about this

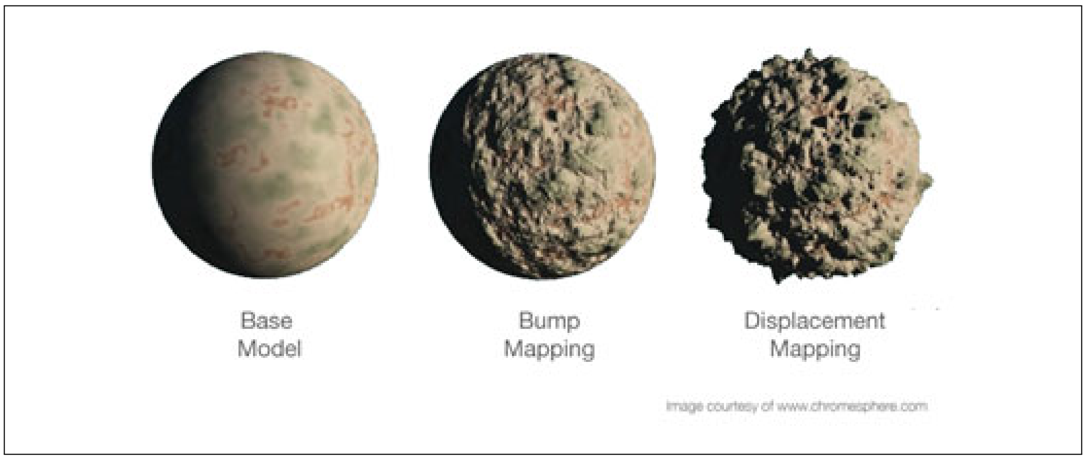
7. Compare and contrast:
   - texture mapping
   - bump mapping
   - displacement mapping

Answer:

Texture mapping is where the surface of a *3D* is mapped onto a *2D* coordinate grid. A *2D* texture can then be applied to the object.

Bump mapping is a technique for simulating bumps or wrinkles on the surface of an object. A*normal map* is mapped to the surface (in the same way a texture is mapped). This alters the normal at points on the surface and thus when the lighting is calculated it simulates bumps on the surface. It is useful when modelling an object that closely resembles a regular shape (e.g. a sphere) but has small disturbances on the surface. It is not suitable for large bumps as it doesn't alter the actual shape and thus at the edges of the sphere it will still be perfectly circular and thus will look strange. Also, since it only affects the normals it can't simulate self shadowing.

Displacement mapping uses a map that actually changes the geometric position of points on the surface and thus in contrast to bump mapping it does allow for *self-shadowing*.



10. (a) Show how to perform 2D rotation around an arbitrary point.

Answer:

To rotate θ° about the origin you perform the following:

$$\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \end{aligned}$$

To translate along vector (t_x, t_y) you perform:

$$\begin{aligned} x' &= x + t_x \\ y' &= y + t_y \end{aligned}$$

A rotation about an arbitrary point is equivalent to a translation (such that the arbitrary point becomes the origin) then the rotation followed by the reverse translation.

It is convenient (computationally) to represent these transformations as matrices as they can then be combined (using matrix multiplication). Since it is common to apply a transformation to several points it is often useful to be able to do some work once that is then used for every point.

Since a translation transformation cannot be represented as a matrix when using a traditional coordinate representation we will use *homogeneous coordinates*.

The matrix for a rotation in homogeneous coordinates is:

$$\begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The matrix for a translation in homogeneous coordinates is:

$$\begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

Thus, the transformation matrix representing a rotation of $\theta°$ about an arbitrary point (p_x, p_y) is given by:

$$M = \begin{pmatrix} 1 & 0 & -p_x \\ 0 & 1 & -p_y \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & -p_x \\ 0 & 1 & -p_y \\ 0 & 0 & 1 \end{pmatrix}$$

(b) Show how to perform 3D rotation around an arbitrary axis parallel to the *x-axis.*

Answer:

The solution for a 3D rotation is largely the same as the 2D (just with an extra dimension). In the interest of brevity I will omit the steps and just show the final matrix calculation.

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -p_y \\ 0 & 0 & 1 & -p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(c) Show how to perform 3D rotation around an arbitrary axis.

Answer:

Not sure about this? I think it is a translation, 3 rotations (one around each orthogonal axis) followed by t but can't get it to work.

Advanced Questions

3. Define and then compare and contrast the following methods of specifying rotation in 3D.

   You will need to look these up.

   (a) Quaternions

Answer:

The Quaternions H are equal to $\mathbb{R}^4$, a four-dimensional vector space over the real numbers. H has three operations:

Addition Defined as their sum as elements of $\mathbb{R}^4$

Scalar multiplication Defined as the product of each element by the scalar.

Quaternion multiplication

(b) Euler Angles

Answer:

Euler angles do not interpolate in a consistent way and are thus less useful for animation where the interpolation matters (e.g. should animate smoothly between keyframes).

Euler angles also suffer from *Gimbal lock* (where a degree of freedom is lost when the axes of two of the gimbals are driven into a parallel configuration. This can be mitigated by altering the gimbal order but not completely avoided in some circumstances and requires additional thinking to get right.

4. Find out about the Cook-Torrance shading model and explain how this improves on the naïve diffuse+specular+ambient model.

Answer:

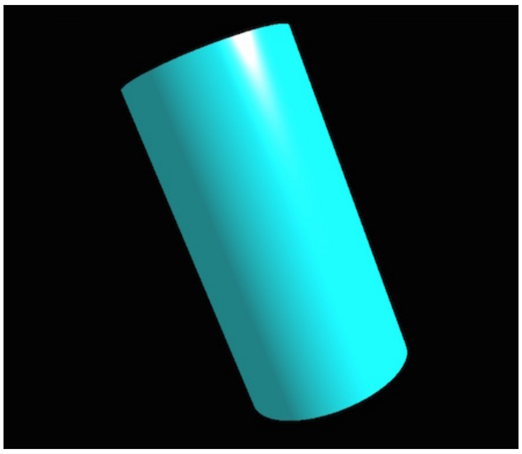
The *Cook-Torrance* shading model models specular reflections using *micro-facets*. Essentially it models a surface as a "sea" of V-shaped facets that act like perfect mirrors. It takes into account the Fresnel effect (that materials become more *specular* at glancing angles and are more *Lambertian* towards normal incidence).

This model allows for shadowing at the micro scale and thus more accurately represents the reflection intensity at large angles of incidence.

Practical Programming

1. Write a program using OpenGL to display the scene that you rendered with your ray-tracer. Explore some different surface models in the fragment shader.

Answer:



Source can be seen at:
`https://github.com/danielchatfield/cst1b-computer-graphics/`