Chapter 15

Texture Mapping

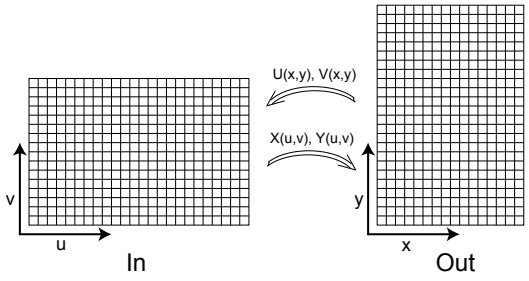
We have discussed forward mapping between images, where image coordinates (u, v) on an input image are mapped to coordinates (x, y) on an output image via a pair of functions

$$x = X(u, v),$$
$$y = Y(u, v).$$

and, we have seen how this can be inverted by defining functions

$$u = U(x, y),$$
$$v = V(x, y),$$

such that U and V exactly invert the maps X and Y:

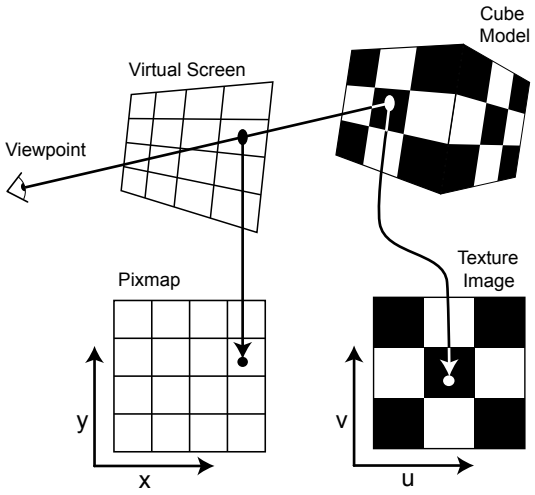


$$u = U[X(u, v), Y(u, v)],$$
$$v = V[X(u, v), Y(u, v)].$$

This allows the following inverse mapping algorithm, which is guaranteed to supply one output color for every output pixel:

```
for(x = 0; x <= x_max; x += Δx)
   for(y = 0; y <= y_max; y += Δy){
      u = U(x,y);
      v = V(x,y);
      Out[x][y] = In[u][v];
   }
```

Now, Let us look at the texture mapping problem, where we would like to use an image to supply color information across a surface, as in the figure below.



The idea is that we provide a mapping between our model and a texture image, so that wherever a ray hits our model, we can recover *texture coordinates* that tell us where to sample the texture image to get the color for that point on the model. The texture color is then used in the shader to provide the color for the pixel in the pixmap which originated the ray.

Note, that this is actually an inverse map, since what we want to do (i.e. the forward map) is to copy the colors in the texture image to the pixmap, but instead we start with the pixel in the pixmap and then determine what color in the texture image maps to it.

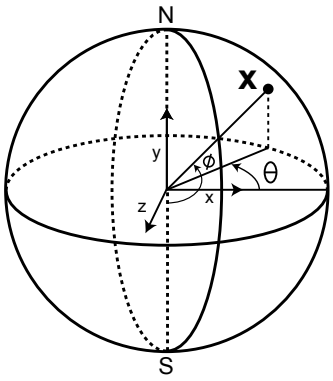
The raytracer supplies the part of the inverse map that goes from pixmap pixel to a point on the model. There are several ways to supply the rest of the inverse map from the point on the model to the texture map. The two most popular are:

1. Parameterize the surface so that we can compute (u, v) from the hit point $\mathbf{x}$.

2. For a polygonal surface, have the artist who creates the surface model supply explicit texture coordinates (u_i, v_i) for each vertex i in the model. Then compute (u, v) by interpolating the texture coordinates of nearby vertices.

15.1 Parametric texture mapping

A simple example of a texture map supported by a surface parameterization is that of a sphere. A sphere can be represented by the following parameterization. Let $-\pi \leq \theta \leq \pi$ be the latitude angle around the sphere in the horizontal plane. Let $0 \leq \phi \leq pi$ be the longitudinal angle, starting from 0 at the south pole **S** and going to π at the north pole **N**. Then, if θ is measured from the positive x axis, given θ, ϕ, and radius r, and with the sphere at the origin,



$$\mathbf{x} = \begin{bmatrix} r\cos\theta\sin\phi \\ -r\cos\phi \\ -r\sin\theta\sin\phi \end{bmatrix}.$$

Now, if we define

$$u = (\theta + \pi)/2\pi, v = \phi/\pi$$

then for all points on the surface of the sphere $0 \leq u, v \leq 1$ and can be used to index the texture map. All we need is a method to compute parameters θ, and ϕ) from a point $\mathbf{x}$ on the sphere's surface. Since

$$\frac{z}{x} = \frac{-r\sin\theta\sin\phi}{r\cos\theta\sin\phi} = \frac{-\sin\theta}{\cos\theta} = -\tan\theta,$$

we have

$$\theta = \tan^{-1}(-z/x),$$

and more simply

$$\phi = \cos^{-1}(-y/r).$$

To generalize, for a sphere of radius r and center at arbitrary location $\mathbf{c}$, we have in code form:

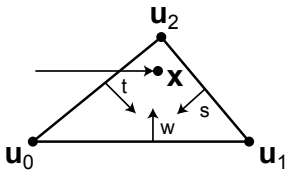
```
θ = atan2(-(z - c_z), x - c_x);
u = (θ + π) / 2 π;
φ = acos(-(y - c_y) / r);
v = φ / π;
```

This gives us a complete inverse map from pixmap to sphere to texturemap.

15.2 Explicit texture coordinates

If the surface is a polygonal surface the user can supply explicit texture coordinates with each vertex. The problem, then, is to convert from a hit point on a

particular polygon to (u, v) coordinates in the texture map.



If the surface is triangulated, we can use the barycentric coordinates of the hit point on the triangle to interpolate the texture (u, v) coordinates from the three triangle vertices. For each 3D vertex $\mathbf{v}_i$ we have supplied $\mathbf{u}_i$, the 2D (u, v) coordinates for that vertex. To avoid notation confusion, let us use (s, t, w) for the three barycentric coordinates of hit point $\mathbf{x}$. Thus, the interpolation of the vertex texture coordinates giving the (u, v) coordinates corresponding to the hit point $\mathbf{x}$ is

$$\mathbf{u}_x = s\mathbf{u}_0 + t\mathbf{u}_1 + w\mathbf{u}_2.$$

Similarly, a bilinear interpolation scheme can be used to get interpolated (u, v) coordinates for quadrilateral polygons.