To do this we construct a scale which will map the bounds of the Viewport3D to the range [0,0] - [2,2]. We then apply a translation to move the origin from the upper left corner to the center. This puts our point in the range [-1,1] - [1,-1]. Finally we account for the Y axis pointing down instead of up in the 2D coordinate system.

```
// Scale bounds to [0,0] - [2,2]

double x = p.x / (width/2);

double y = p.y / (height/2);



// Translate 0,0 to the center

x = x - 1;



// Flip so +Y is up instead of down

y = 1 - y;
```

Now that we've found our x and y position on the sphere we can find z. Since our sphere is of *radius* = 1 we know that . Solving for *z* we get:

```
double z2 = 1 - x * x - y * y;

double z = z2 > 0 ? Math.Sqrt(z2) : 0;



Vector3D p = new Vector3D(x, y, z);

p.Normalize();
```

We now have the (*x*,*y*,*z*) coordinates of the point on the sphere beneath the mouse pointer.

2.2 Rotating Between the Points

On each mouse move we want to construct a rotation that will keep the same point on the sphere underneath the mouse pointer. We do this by remembering the previous point on the sphere from the last mouse move event and constructing a rotation that will transform it to the point currently under the mouse pointer.

To compute this rotation we need two things:

1. The axis of rotation
2. The angle of rotation θ

Figure 7

We need to find the axis of rotation and angle θ that will transform v_1 onto v_2.

Because our sphere is centered at the origin we may interpret our points as vectors. Doing so it is trivial to find the axis and angle of rotation using the cross product and dot product respectively:

```
Vector3D axis = Vector3D.CrossProduct(v1, v2);

double theta = Vector3D.AngleBetween(v1, v2);
```

Once we have the axis and angle all that remains is to apply the new rotation to the current orientation:

```
// We negate the angle because we are rotating the camera.
// Do not do this if you are rotating the scene instead.
Quaternion delta = new Quaternion(axis, -angle);

// Get the current orientantion from the RotateTransform3D
RotateTransform3D rt = (RotateTransform3D) camera.Transform;
AxisAngleRotation3D r = (AxisAngleRotation3D) rt.Rotation;
Quaternion q = new Quaternion(r.Axis, r.Angle);

// Compose the delta with the previous orientation
q *= delta;

// Write the new orientation back to the Rotation3D
r.Axis = q.Axis;
r.Angle = q.Angle;
```