<model filename="objects/cube.obj" name="cube" material="Brass">

<scale x="0.3" y="0.3" z="0.3"/>

<rotation x="0.0" y="0.0" z="0.0"/>

<position x="-1.3" y="-0.2" z="0.0"/>

</model>

<model filename="objects/cube.obj" name="cube" material="White">

<scale x="1.6" y="1.0" z="1.0"/>

<rotation x="0.0" y="0.0" z="0.0"/>

<position x="-1.6" y="0.0" z="1.0"/>

</model>

<triangle name="Triangle">

<scale z="5.0" y="5.0" x="5.0"/>

<rotation z="0.0" y="0.0" x="0.0"/>

<position z="0.0" y="-0.6" x="0.0"/>

<!-- First Vertex -->

<vertex material="Red" index="0">

<position x="1.0" y="0.0" z="1.0"/>

<normal x="0.0" y="1.0" z="0.0" />

<texture v="0.0" u="0.0"/>

</vertex>

<!-- Second Vertex -->

<vertex material="Red" index="1">

<position x="-1.0" y="0.0" z="1.0"/>

<normal x="0.0" y="1.0" z="0.0"/>

<texture v="0.0" u="0.0"/></vertex>

<!-- Third Vertex -->

<vertex material="Red" index="2">

<position x="-1.0" y="0.0" z="-1.0"/>

<normal x="0.0" y="1.0" z="0.0"/>

<texture v="0.0" u="0.0"/>

</vertex>

</triangle>

<triangle name="Triangle">

<scale z="5.0" y="5.0" x="5.0"/>

<rotation z="0.0" y="0.0" x="0.0"/>

<position z="0.0" y="-0.6" x="0.0"/>

<!-- First Vertex -->

<vertex material="Red" index="0">

<position x="1.0" y="0.0" z="1.0"/>

<normal x="0.0" y="1.0" z="0.0" />

<texture v="0.0" u="0.0"/>

</vertex>

<!-- Second Vertex -->

<vertex material="Red" index="1">

<position x="-1.0" y="0.0" z="-1.0"/>

<normal x="0.0" y="1.0" z="0.0"/>

<texture v="0.0" u="0.0"/></vertex>

<!-- Third Vertex -->

<vertex material="Red" index="2">

<position x="1.0" y="0.0" z="-1.0"/>

<normal x="0.0" y="1.0" z="0.0"/>

<texture v="0.0" u="0.0"/>

</vertex>

</triangle>

if(closestObj->IsTriangle())

{

SceneTriangle \*myTriangle= (SceneTriangle\*)closestObj;

//find the normal of the intersection position. take the center of the object minus the object intersection

//Finding normal for a sphere

Vector edge1 = myTriangle->vertex[1]-myTriangle->vertex[0];

Vector edge2 = myTriangle->vertex[2]-myTriangle->vertex[0];

Vector Cross = edge1.Cross(edge2).Normalize();

Vector Reflect = (Cross \* (View.Dot(Cross))\* 2.0 - View).Normalize();

//Vector Reflect = (Normal \* (Light.Dot(Normal))\* 2.0 - Light).Normalize();

// light vector,calculated ray vector, background ambientlight, material, view\_vector (v), normal vector (n)

// Vector R = (Normal \* (Light1.Dot(Normal))\* 2.0 - Light1).Normalize();

Color1 = Color1+ Phong(Light,R,m\_Scene.GetBackground(),m\_Scene.GetMaterial(myTriangle->material[0]),View,Cross)\*isShadow+ Trace(PoI+Reflect\*.0001,Reflect,n+1);// +Trace(Reflection) +Trace(Refraction);

}