|  |
| --- |
| CPI 411 Graphics for Games |
| **Diffraction.fx**  This lab is to generate a diffraction shader file which shows the wavelengths of light through colors.   * Diffraction.fx : Shader file to show the diffraction shader on the models. * Game1 (or TermProject).cs : Sample game program to test the shader. * Torus.fbx : Object to display Diffraction.fx on.   **A. First Concept**  To compute the diffraction pattern the halfway vector between the light source and the view is projected onto the tangent vector. From this value and the distance, we can compute the interfering wavelengths. The Red, Blue, and Green components are calculated by bump functions using C which controls the appearance of the color map and y which shows the wavelengths. The sum of the diffraction and the anisotropic highlight, computed by finding the hi-light color, roughness, and the normal direction, is the final output color.  **B. Shader File**  This is the beginning  **// \*\*\* Diffraction.fx : Step 1**  **struct** VertexShaderInput  {  **float4** Position : POSITION0**;**  **float4** Normal: NORMAL0;  **float4** Tangent : TEXCOORD0;  };  **struct** VertexShaderOutput  {  **float4** Position : POSITION0;  **float3** Normal : TEXCOORD1;  **float3** Tangent : TEXCOORD0;  };  VertexShaderOutput DiffractionVertShaderFunction(VertexShaderInput input)  {  VertexShaderOutput output;  float4 worldPosition = **mul**(input.Position, World);  float4 viewPosition = **mul**(worldPosition, View);  output.Position = **mul**(viewPosition, Projection);  output.Normal = normalize(**mul**(input.Normal, WorldInverseTranspose).xyz);  output.Tangent = normalize(**mul**(input.Tangent, WorldInverseTranspose).xyz);  **return** output;  }  **float4** DiffractionPixelShaderFunction(VertexShaderOutput input) : COLOR0  {  **float3** L = normalize(lightPosition - input.Position.xyz);  **float3** V = normalize(CameraPosition - input.Position.xyz);  **float3** H = L + V;  **float3** N = normalize(input.Normal);  **float3** T = normalize(input.Tangent);  **float** u = dot(T, H) \* 1;  **float** halfNorm = dot(N, H);  **float** expon = 1 \* u / halfNorm;  **float3** shapeParam = exp(-expon \* expon);  **float4** anistropic = float4(1,1,1,1) \* float4(shapeParam.x, shapeParam.y, shapeParam.z, 1);  if (u < 0) u = -u;  **float4** colorDiff = float4(0, 0, 0, 1);  for (int i = 1; i < 8; i++)  {  float y = 2 \* u / i - 1;  float3 x = float3(4 \* (y - 0.75), 4 \* (y - 0.5), 4 \* (y - 0.25))  float3 y1 = 1 - x \* x;  y1 = max(y1, float3 (0, 0, 0));  colorDiff.xyz += y1;  }  **float4** color = colorDiff + anistropic;  return color;  }  **C. Main Program (Game1.cs)**  In the Content Load Method you will need to load the models, skybox textures, and the shader file.  **--In the ContentLoad() method**  CD = Content.Load<Model>("CD");  …  skybox = new SkyBox(skyboxTextures, Content, GraphicsDevice);  effect = Content.Load<Effect>("Diffraction");  }  **--In the Draw() method**  GraphicsDevice.Clear(Color.CornflowerBlue);  GraphicsDevice.DepthStencilState = new DepthStencilState();  graphics.GraphicsDevice.RasterizerState = originalRasterizerState;  effect.CurrentTechnique = effect.Techniques[0];  foreach (EffectPass pass in effect.CurrentTechnique.Passes)  {  foreach (ModelMesh mesh in model.Meshes)  {  foreach (ModelMeshPart part in mesh.MeshParts)  {  effect.Parameters["World"].SetValue(mesh.ParentBone.Transform);  effect.Parameters["View"].SetValue(view);  effect.Parameters["Projection"].SetValue(projection); effect.Parameters["CameraPosition"].SetValue(camPosition);  Matrix worldInverseTransposeMatrix =  Matrix.Transpose(Matrix.Invert(mesh.ParentBone.Transform));  effect.Parameters["WorldInverseTranspose"].SetValue(worldInverseTransposeMatrix);  …  }  }  }  base.Draw(gameTime);  }  **D. Main Exercise**  The sum of the diffraction and the anisotropic highlight, computed by finding the hi-light color, roughness, and the normal direction, is the final output color  --In Draw() Method  RasterizerState originalRasterizerState = graphics.GraphicsDevice.RasterizerState;  RasterizerState rasterizerState = new RasterizerState();  rasterizerState.CullMode = CullMode.None;  graphics.GraphicsDevice.RasterizerState = rasterizerState;  skybox.Draw(view, projection, camPosition);  …  --In the Shader File  float u = dot(T, H) \* ???;  …  float expon = ??? \* u / halfNorm;  …  float4 anistropic = ??? \* float4(shapeParam.x, shapeParam.y, shapeParam.z, 1);      **\*\*\* IMPORTANT \*\*\***  Complete the exercise in D section, and submit a zipped file including the solution (.sln) file and the project folders to course online site. The submission item is located in the "**Quiz and Lab**" section. Each lab has **10 points**. If you complete the exercise in class time, the full points will be assigned. The late submission is accepted just before the next class with 2 points reductions, because the solution is demonstrated in the next class. |
|  |