|  |
| --- |
| CPI 411 Graphics for Games |
| **RGB Image Processing / Filtering**  The main objective of this exercise is to be able to alter any image’s RGB color channels and to also add do this via a Filter over the image as well by using Pixel Editing. This was done similarly to that of Lab08 Image Processing  •ColorControl.fx: Shader file to alter images coloration.  •LabSears.cs: Sample game program to test the shader (Named after creator of lab)  Based on Chapter 22 of GPU Gems:  <http://developer.download.nvidia.com/books/HTML/gpugems/gpugems_ch22.html>  **A. First Concept**  The idea behind the shader is like a color wheel and choosing what colors to keep within an image and altering them. Secondly, to apply filters to images as well that change the colors too.  **B. Shader (ColorControls.fx)**  To start let’s do a *Sin City* or *The Spirit* movie style shader where the red in image will stay and the GB values will change to black/white.  Start by making a new fx file in Visual Studio and call it ColorControls, then make the following as a base for the shader file:  Step 1: Copy the code into ColorControls.  float4x4 MatrixTransform;  texture2D modelTexture;  texture2D filterTexture;  float imageWidth;  float imageHeight;  float colorRedValue;  float colorGreenValue;  float colorBlueValue;  float offset = 0;  sampler TextureSampler: register(s0) = sampler\_state  {  Texture = <modelTexture>;  AddressU = CLAMP;  AddressV = CLAMP;  MipFilter = LINEAR;  MinFilter = LINEAR;  MAgFilter = LINEAR;  };  sampler FilterSampler: register(s1) = sampler\_state  {  Texture = <filterTexture>;  AddressU = WRAP;  };  struct VS\_OUTPUT  {  float4 Pos: POSITION;  float2 UV0: TEXCOORD0;  float4 UV1: TEXCOORD1;  };  //Basic Vertex Shader  VS\_OUTPUT vtxSh(float4 inPos : POSITION, float2 inTex : TEXCOORD0)  {  VS\_OUTPUT output;  output.Pos = mul(inPos, MatrixTransform);  output.UV0 = inTex;  output.UV1 = float4(3 / imageWidth, 0, 0, 3 / imageHeight);  return output;  }  float4 pxlShRed(VS\_OUTPUT input) : COLOR  {  ...  }  ...(More to be added)...  technique MyShader1  {  pass P0  {  VertexShader = compile vs\_4\_0 vtxSh();  PixelShader = compile ps\_4\_0 pxlShRed();  }  }  Step 2: Make the following Pixel Shader: Think of what color we are wanting to keep, and the other colors change to black/white.  float4 pxlShRed(VS\_OUTPUT input) : COLOR  {  float4 colorin, colorout;  colorin = tex2D(TextureSampler, input.UV0 + input.UV1.xy);  colorout = colorin;  if ((??? + ???) < ???)  {  ??? = ???;  ??? = ???;  ??? = ???;  }  //This part makes the other colors black and white  else  {  colorout.rgb = dot(colorin.rgb, colorin.rgb) \* dot(colorin.rgb, float3(2, 1, 2));  }  return colorout;  }  Step 3: Now that you have successfully completed changing the images coloring red and black/white. Try doing the same but for the green and blue color channels as well.  For Ensuring proper color: Also if you want to make sure that your image is changing the right color values, this piece of code will show you what the original image is supposed to look like:  float4 pxlShNormal(VS\_OUTPUT input) : COLOR  {  float4 colorin;  float4 colorout;  colorin = tex2D(TextureSampler, input.UV0 + input.UV1.xy);  colorout = colorin;  return colorout;  }  The remaining shaders will be a combination of doing the G values and then altering the RB, or keeping the B values and the RB values to have different effects on the images you load into your project.  **C. Main Program (LabSears.cs)**  For the main program you will need to add in the following Template:  using System;  using Microsoft.Xna.Framework;  using Microsoft.Xna.Framework.Graphics;  using Microsoft.Xna.Framework.Input;  using CPI411\_2020.SimpleEngine;  namespace LabSears  {  public class LabSears : Game  {  GraphicsDeviceManager graphics;  SpriteBatch spriteBatch;  Effect effect;  Texture2D texture, filter;  public LabSears()  {  graphics = new GraphicsDeviceManager(this);  Content.RootDirectory = "Content";  graphics.GraphicsProfile = GraphicsProfile.HiDef;  ScreenManager.Initialize(graphics);  }  protected override void Initialize()  {  ScreenManager.Initialize(graphics);  base.Initialize();  }  protected override void LoadContent()  {  spriteBatch = new SpriteBatch(GraphicsDevice);  ScreenManager.Setup(true, 1920, 1080);  //Choose any photo here for your texture, more can be added later on  texture = Content.Load<Texture2D>("???");  filter = Content.Load<Texture2D>("filter");  effect = Content.Load<Effect>("ColorCorrection");  ...  }  protected override void UnloadContent() { }  protected override void Update(GameTime gameTime)  {  if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed || Keyboard.GetState().IsKeyDown(Keys.Escape))  Exit();  if (Keyboard.GetState().IsKeyDown(Keys.F1))...  base.Update(gameTime);  }  protected override void Draw(GameTime gameTime)  {  GraphicsDevice gd = graphics.GraphicsDevice;  gd.Clear(Color.CornflowerBlue);  spriteBatch.Begin(0, null, null, null, null, effect);  spriteBatch.Draw(texture, Vector2.Zero, null, Color.White, 0, Vector2.Zero, 0.5f, SpriteEffects.None, 0);  spriteBatch.End();  base.Draw(gameTime);  }  }  }  **D. Main Exercise**  Now you will start to edit both **ColorCorrection and LabSears** so they can both display and alter the images in the following ways:  (The first 4 should be simple to do)  **#1**: Red with B/W **#2**: Green with B/W **#3**: Blue with B/W  A picture containing text, photo, looking, man  Description automatically generatedA picture containing covered, snow, blue, water  Description automatically generated A picture containing cake, dark, green, lit  Description automatically generated  **#4**: Norma1  A person holding a sign  Description automatically generated  (**HINT**: For #5 – 7 think about what colors when mixed become those)  **#5**: Cyan **#6**: Magenta **#7**: Yellow  A picture containing dark, holding, man, sign  Description automatically generatedA screen shot of a person in a dark room  Description automatically generatedA picture containing outdoor, man, photo, dark  Description automatically generated  #**8**: Red focused #**9**: Green focused #**10**: Blue focused  A picture containing dark, sitting, computer, lit  Description automatically generatedA picture containing sitting, looking, light, dark  Description automatically generatedA picture containing sitting, dark, lit, computer  Description automatically generated  #**11**: Negative of Image  A picture containing colorful, lit, decorated, light  Description automatically generated  #**12**: Red focus with Cyan #**13**: Green focus with Yellow #**14**: Blue focus with Magenta  A picture containing red, orange, sign, hydrant  Description automatically generatedA picture containing clock  Description automatically generatedA picture containing outdoor, sign, man, dark  Description automatically generated  #**15**: Gradient map (The image can be changed bases on the offset – See video)  A picture containing colorful, cake, sitting, green  Description automatically generated  #**16**: Red Gradient Map with Cyan offset (The image can be changed bases on the offset – See video)  A sign in the dark  Description automatically generated  #**17**: Green Gradient Map with Magenta offset (The image can be changed bases on the offset – See video)  A picture containing flower, rain, water  Description automatically generated  #**18**: Blue Gradient Map with Yellow offset (The image can be changed bases on the offset – See video)  A picture containing outdoor, dark, black, sign  Description automatically generated  #**19**: Modifying just the red and keeping other colors BW **(Right image is offset at 0 and Left is at about 0.5)**  **\*\*\*Warning - If Your image has brown in it won’t process the color appropriatley due to the color of brown.\*\*\*\***    #**20**: Modifying just the green and keeping other colors BW **(Right image is offset at 0 and Left is at about 0.5)**    #**21** Modifying just the blue and keeping other colors BW **(Right image is offset at 0 and Left is at about 0.5)**  :  **These are what your outputs should look like before you manipulate the offset.**  **\*\*\* 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. |
|  |