ECE4893A/CS4803MPG: Multicore and GPU Programming for Video Games

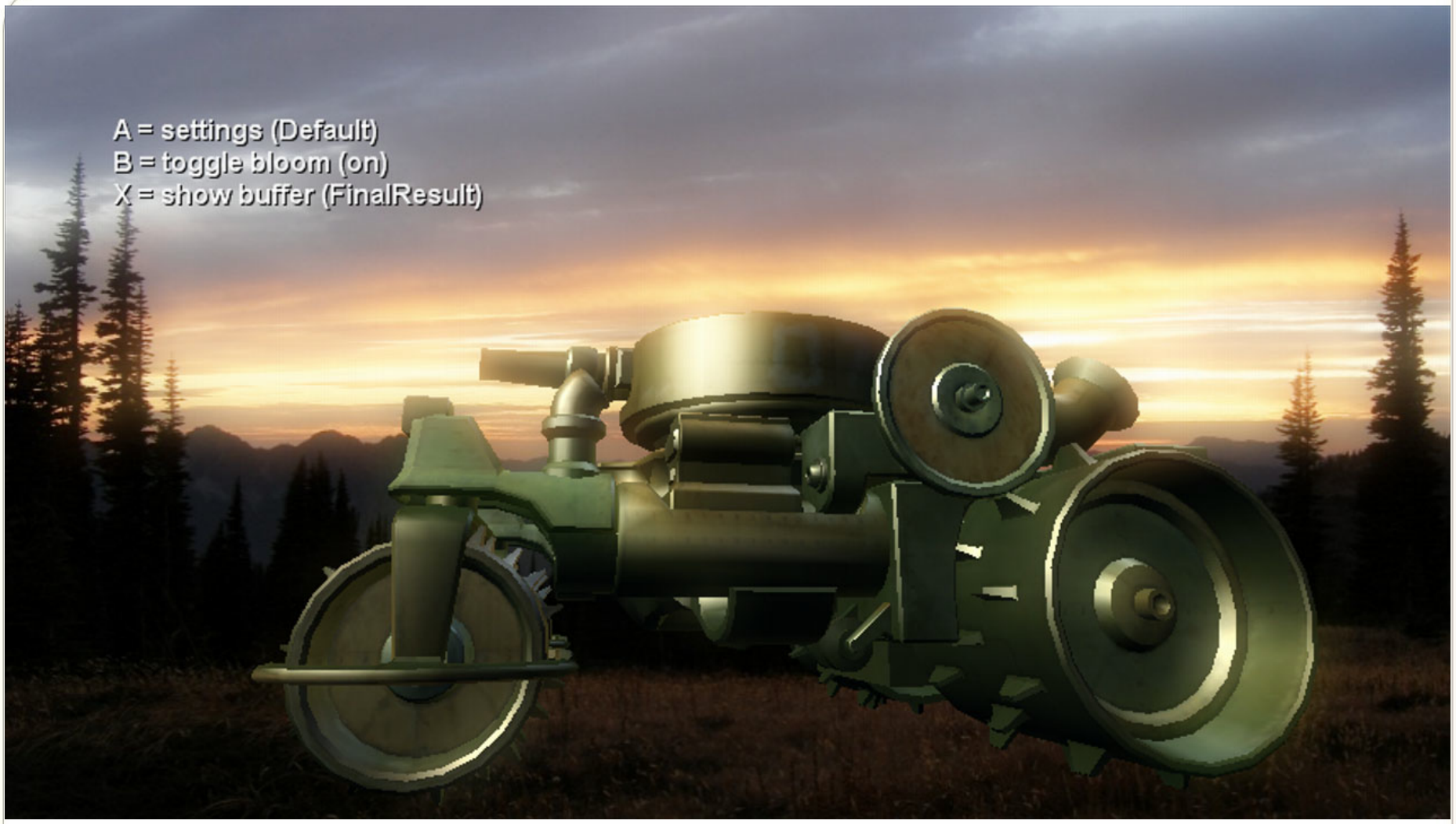
Postprocessing

Prof. Aaron Lanterman

School of Electrical and Computer Engineering

Georgia Institute of Technology

Bloom effect - before



http://creators.xna.com/en-us/sample/bloom

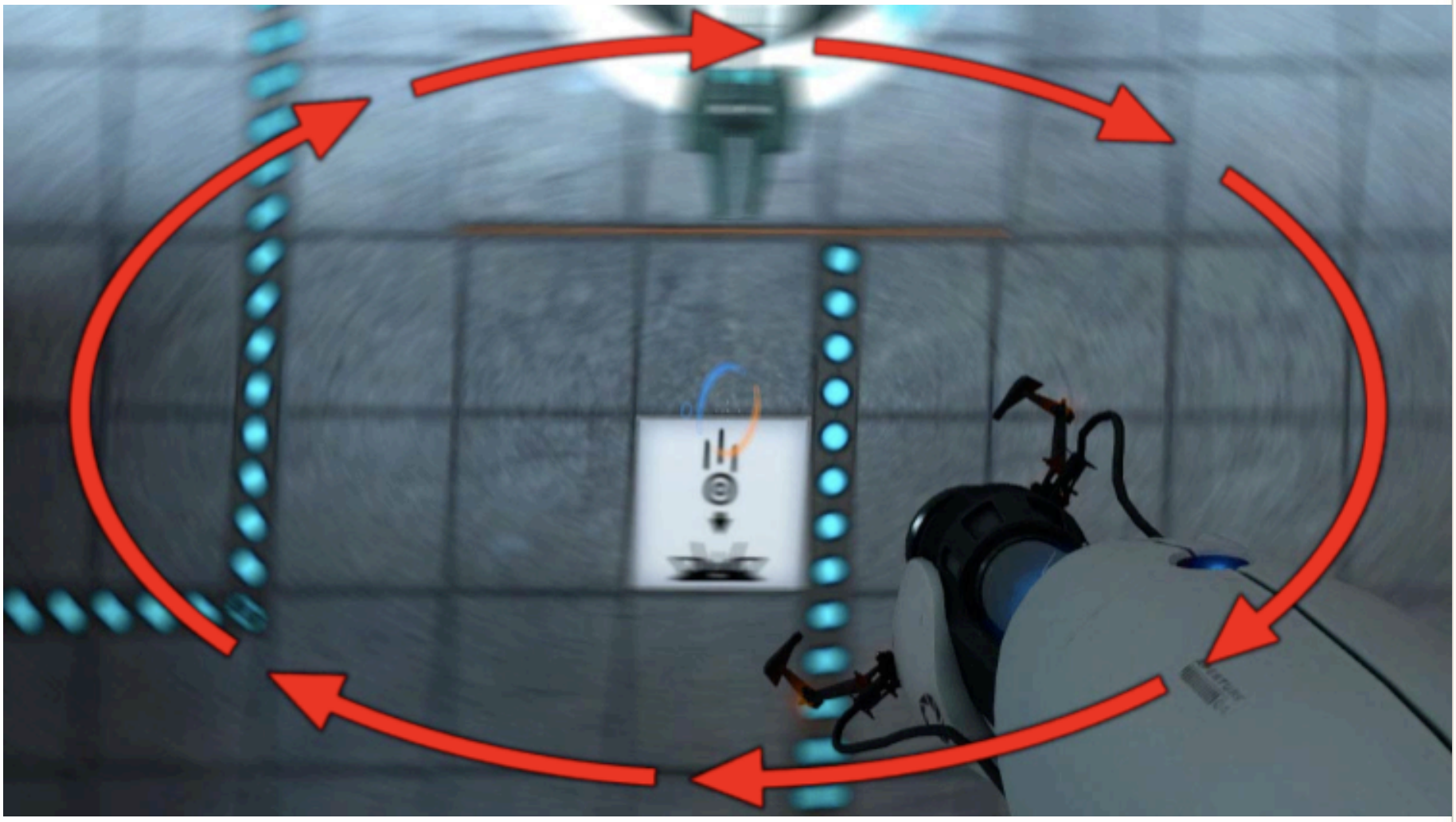
Georgia Institute of Technology

Bloom effect - after

http://creators.xna.com/en-us/sample/bloom

Georgia Institute of Technology

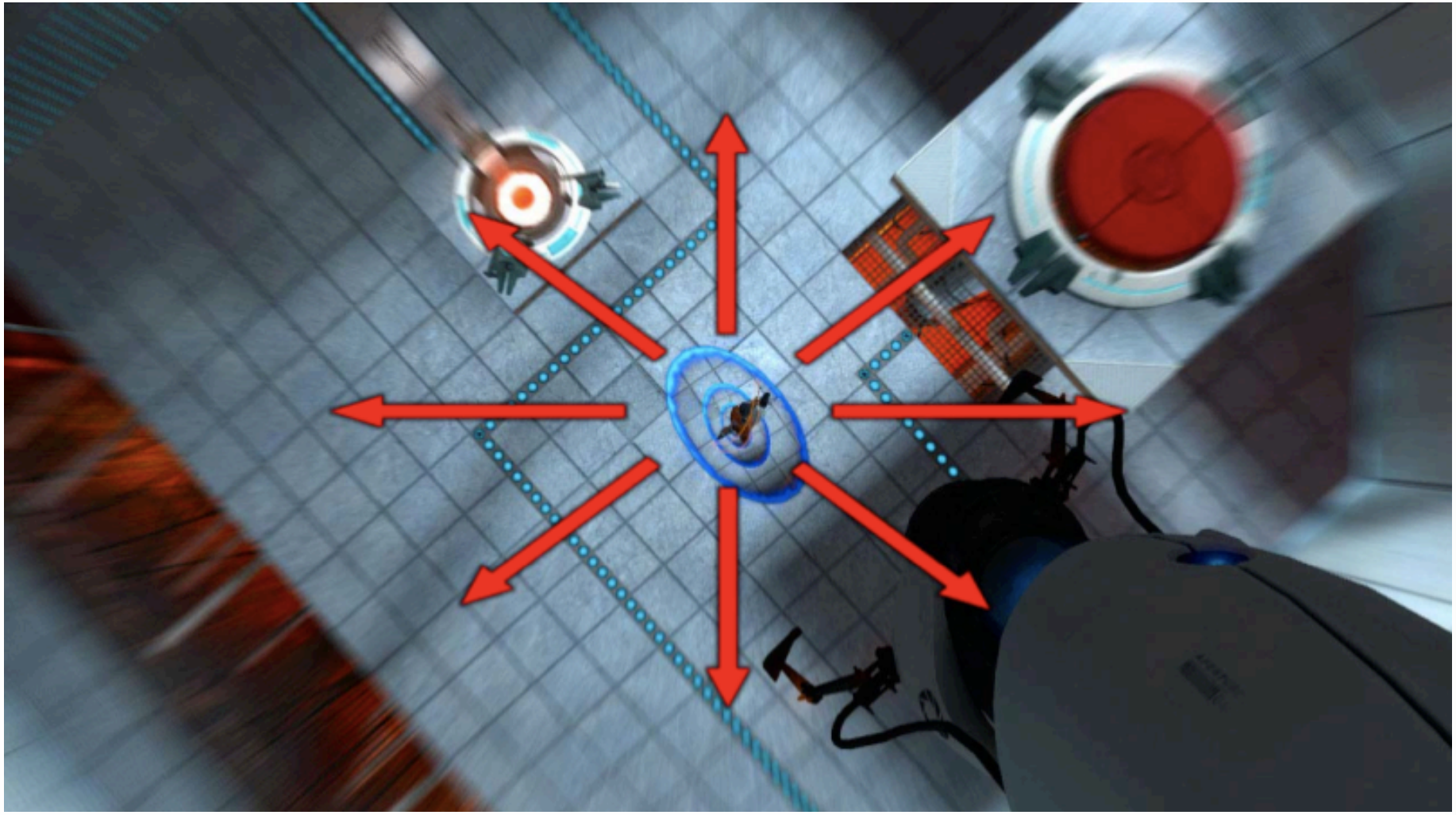
Motion blur in Valve's Portal - roll



http://www.valvesoftware.com/publications/2008/GDC2008_PostProcessingInTheOrangeBox.pdf

Georgia Institute of Technology

Motion blur in Valve's Portal - falling



http://www.valvesoftware.com/publications/2008/GDC2008_PostProcessingInTheOrangeBox.pdf

Multiple textures

- In your C# code:

```
graphics.GraphicsDevice.Textures[0] = firstTexture;
graphics.GraphicsDevice.Textures[1] = secondTexture;
```

- In your shader code:

```
sampler firstSampler : register(s0);
sampler secondSampler : register(s1);
```

From msdn2.microsoft.com/en-us/library/microsoft.xna.framework.graphics.graphicsdevice.textures.aspx

BloomComponent.cs (1)

```
public class BloomComponent : DrawableGameComponent
{

public class SpriteBatch spriteBatch;

Effect bloomExtractEffect;
Effect bloomCombineEffect;
Effect gaussianBlurEffect;

RenderTarget2D sceneRenderTarget;
RenderTarget2D renderTarget1;
RenderTarget2D renderTarget2;
```

http://creators.xna.com/en-us/sample/bloom

BloomComponent.cs (2)

```
public enum IntermediateBuffer
{
            PreBloom,
            BlurredHorizontally,
            BlurredBothWays,
            FinalResult,
}
```

http://creators.xna.com/en-us/sample/bloom

BloomComponent.cs – LoadContent (1)

```
spriteBatch = new SpriteBatch(GraphicsDevice);

bloomExtractEffect =
  Game.Content.Load<Effect>("BloomExtract");
bloomCombineEffect =
  Game.Content.Load<Effect>("BloomCombine");
gaussianBlurEffect =
  Game.Content.Load<Effect>("GaussianBlur");

// Look up the resolution and format of our main backbuffer.
PresentationParameters pp =
  GraphicsDevice.PresentationParameters;

int width = pp.BackBufferWidth;
int height = pp.BackBufferHeight;

SurfaceFormat format = pp.BackBufferFormat;
```

http://creators.xna.com/en-us/sample/bloom

SurfaceFormat enumeration

	Member name	Description
	Color	(Unsigned format) 32-bit ARGB pixel format with alpha, using 8 bits per channel.
	Bgr565	(Unsigned format) 16-bit BGR pixel format with 5 bits for blue, 6 bits for green, and 5 bits for red.
	Bgra5551	(Unsigned format) 16-bit BGRA pixel format where 5 bits are reserved for each color and 1 bit is reserved for alpha.
	Bgra4444	(Unsigned format) 16-bit BGRA pixel format with 4 bits for each channel.
	Dxt1	DXT1 compression texture format. The runtime will not allow an application to create a surface using a DXTn format unless the surface dimensions are multiples of 4. This applies to offscreen-plain surfaces, render targets, 2D textures, cube textures, and volume textures.

Screenshot from
msdn2.microsoft.com/en-us/library/microsoft.xna.framework.graphics.surfaceformat.aspx

BloomComponent.cs – LoadContent (2)

```
// Create a texture for rendering the main scene,
// prior to applying bloom.
sceneRenderTarget =
  new RenderTarget2D(GraphicsDevice,
                     width, height,
                     false,
                     format,
                     pp.DepthStencilFormat,
                     pp.MultiSampleCount,
                     RenderTargetUsage.DiscardContents);
```

http://creators.xna.com/en-us/sample/bloom

Under the hood of the Xbox 360

- Xenos GPU renders into only one physical rendertarget: 10 MB eDRAM
  - Cannot texture from eDRAM
  - Cannot render into main 512M RAM
- Must "resolve" to copy rendering in eDRAM back to main memory
  - Hardware designed to make this fast
  - Rendertarget in eDRAM is cleared

From Shawn Hargreaves, "XNA rendertarget semantics," blogs.msdn.com/shawnhar/archive/2007/02/04/xna-rendertarget-semantics.aspx

BloomComponent.cs – LoadContent (3)

```
width /= 2;
height /= 2;

renderTarget1 =
  new RenderTarget2D(GraphicsDevice,
                     width, height,
                     false, format,
                     DepthFormat.None);

renderTarget2 =
  new RenderTarget2D(GraphicsDevice,
                     width, height,
                     false, format,
                     DepthFormat.None);
```

http://creators.xna.com/en-us/sample/bloom

BloomComponent.cs – DrawFullScreenQuad (5 arguments)

```
void DrawFullscreenQuad(Texture2D texture,
                        int width, int height,
                        Effect effect,
                        IntermediateBuffer currentBuffer)
{
  // If the user has selected one of the show intermediate
  // buffer options, we still draw the quad to make sure the
  // image will end up on the screen,
  // but might need to skip applying the custom pixel shader.
  if (showBuffer < currentBuffer) {
    effect = null;
  }

  spriteBatch.Begin(0, BlendState.Opaque,
                    null, null, null, effect);
  spriteBatch.Draw(texture,
                   new Rectangle(0, 0, width, height),
                   Color.White);
  spriteBatch.End();
}
```

http://creators.xna.com/en-us/sample/bloom

BloomComponent.cs – DrawFullScreenQuad (4 arguments)

```
void DrawFullscreenQuad(Texture2D texture,
                        RenderTarget2D renderTarget,
                        Effect effect,
                        IntermediateBuffer currentBuffer)
{

  GraphicsDevice.SetRenderTarget(renderTarget);

  DrawFullscreenQuad(texture,
                     renderTarget.Width,
                     renderTarget.Height,
                     effect,
                     currentBuffer);
}
```

http://creators.xna.com/en-us/sample/bloom

BloomExtract.fx

```
sampler TextureSampler : register(s0);

float BloomThreshold;

float4 PixelShaderFunction(float2 texCoord : TEXCOORD0) : COLOR0
{
    // Look up the original image color.
    float4 c = tex2D(TextureSampler, texCoord);

    // Adjust it to keep values
    // brighter than specified threshold.
    return saturate((c - BloomThreshold) /
                    (1 - BloomThreshold));
}

technique BloomExtract {
    pass Pass1 {
        PixelShader = compile ps_2_0 PixelShaderFunction();
    }
}
```

http://creators.xna.com/en-us/sample/bloom

BloomComponent.cs – Draw, Pass 1

```
GraphicsDevice.SamplerStates[1] = SamplerState.LinearClamp;

// Pass 1: draw the scene into rendertarget 1,
// using a shader that extracts only the brightest
// parts of the image.

bloomExtractEffect.Parameters["BloomThreshold"].
                            SetValue(Settings.BloomThreshold);

DrawFullscreenQuad(sceneRenderTarget,
                   renderTarget1,
                   bloomExtractEffect,
                   IntermediateBuffer.PreBloom);
```

http://creators.xna.com/en-us/sample/bloom

GaussianBlur.fx

```
sampler TextureSampler : register(s0);

#define SAMPLE_COUNT 15

float2 SampleOffsets[SAMPLE_COUNT];
float SampleWeights[SAMPLE_COUNT];

float4 PixelShaderFunction(float2 texCoord : TEXCOORD0) : COLOR0
{
    float4 c = 0;

    // Combine a number of weighted image filter taps.
    for (int i = 0; i < SAMPLE_COUNT; i++) {
        c += tex2D(TextureSampler,
                   texCoord + SampleOffsets[i])
             * SampleWeights[i];
    }

    return c;
}
```

http://creators.xna.com/en-us/sample/bloom

BloomComponent.cs – Draw, Pass 2

```
// Pass 2: draw from rendertarget 1 into rendertarget 2,
// using a shader to apply a horizontal gaussian blur filter.

SetBlurEffectParameters(1.0f / (float) renderTarget1.Width,
                        0);

DrawFullscreenQuad(renderTarget1,
                   renderTarget2,
                   gaussianBlurEffect,
                   IntermediateBuffer.BlurredHorizontally);
```

http://creators.xna.com/en-us/sample/bloom

BloomComponent.cs – Draw, Pass 3

```
// Pass 3: draw from rendertarget 2 back into rendertarget 1,
// using a shader to apply a vertical gaussian blur filter.

SetBlurEffectParameters(0,
                        1.0f / (float) renderTarget1.Height);

DrawFullscreenQuad(renderTarget2,
                   renderTarget1,
                   gaussianBlurEffect,
                   IntermediateBuffer.BlurredBothWays);
```

http://creators.xna.com/en-us/sample/bloom

BloomCombine.fx - globals

```
// Pixel shader combines the bloom
// image with the original
// scene, using tweakable intensity
// levels and saturation.
// This is the final step in applying
// a bloom postprocess.

sampler BloomSampler : register(s0);
sampler BaseSampler : register(s1);

float BloomIntensity;
float BaseIntensity;

float BloomSaturation;
float BaseSaturation;
```

http://creators.xna.com/en-us/sample/bloom

BloomCombine.fx - helper

```
// Helper for modifying the saturation of a color.

float4 AdjustSaturation(float4 color, float saturation)
{
    // The constants 0.3, 0.59, and 0.11
    // are chosen because the
    // human eye is more sensitive to
    // green light, and less to blue.

    float grey = dot(color, float3(0.3, 0.59, 0.11));

    return lerp(grey, color, saturation);
}
```

http://creators.xna.com/en-us/sample/bloom

BloomCombine.fx

```
float4 PixelShaderFunction(float2 texCoord : TEXCOORD0) : COLOR0
{
    // Look up the bloom and original base image colors.
    float4 bloom = tex2D(BloomSampler, texCoord);
    float4 base = tex2D(BaseSampler, texCoord);

    // Adjust color saturation and intensity.
    bloom = AdjustSaturation(bloom, BloomSaturation) * BloomIntensity;
    base = AdjustSaturation(base, BaseSaturation) * BaseIntensity;

    // Darken down the base image in areas where there is
    // a lot of bloom,
    // to prevent things looking excessively burned-out.

    base *= (1 - saturate(bloom));

    // Combine the two images.
    return base + bloom;
}
```

http://creators.xna.com/en-us/sample/bloom

BloomComponent.cs – Draw, Pass 4

```
// Pass 4: draw both rendertarget 1 and the original scene
// image back into the main backbuffer, using a shader that
// combines them to produce the final bloomed result.

GraphicsDevice.SetRenderTarget(null);

EffectParameterCollection parameters =
  bloomCombineEffect.Parameters;

parameters["BloomIntensity"].SetValue(Settings.BloomIntensity);
parameters["BaseIntensity"].SetValue(Settings.BaseIntensity);
parameters["BloomSaturation"].SetValue(Settings.BloomSaturation);
parameters["BaseSaturation"].SetValue(Settings.BaseSaturation);
```

http://creators.xna.com/en-us/sample/bloom

BloomComponent.cs – Draw, Pass 4 (con't)

```
GraphicsDevice.Textures[1] = sceneRenderTarget;

Viewport viewport = GraphicsDevice.Viewport;

DrawFullscreenQuad(renderTarget1,
                   viewport.Width,
                   viewport.Height,
                   bloomCombineEffect,
                   IntermediateBuffer.FinalResult);
```

http://creators.xna.com/en-us/sample/bloom

BloomComponent.cs – BeginDraw

```
public void BeginDraw()
{
  if (Visible)
  {
    GraphicsDevice.SetRenderTarget(sceneRenderTarget);
  }
}
```

http://creators.xna.com/en-us/sample/bloom

Game.cs – Draw

```
GraphicsDevice device = graphics.GraphicsDevice;
Viewport viewport = device.Viewport;

bloom.BeginDraw();

device.Clear(Color.Black);

// Draw the background image.
spriteBatch.Begin(0, BlendState.Opaque);

spriteBatch.Draw(background,
                 new Rectangle(0, 0,
                               viewport.Width, viewport.Height),
                 Color.White);

spriteBatch.End();
```

http://creators.xna.com/en-us/sample/bloom

Game.cs – Draw (con't)

```
// Draw the spinning model.
device.DepthStencilState = DepthStencilState.Default;

DrawModel(gameTime);

// Draw other components (which includes the bloom).
base.Draw(gameTime);

// Display some text over the top.
// Note how we draw this after the bloom,
// because we don't want the text to be
// affected by the postprocessing.

DrawOverlayText();
```

http://creators.xna.com/en-us/sample/bloom

Rendertarget sematics are tricky

- XNA 1.0: http://blogs.msdn.com/b/shawnhar/archive/2007/02/04/xna-rendertarget-semantics.aspx
- XNA 2.0: http://blogs.msdn.com/b/shawnhar/archive/2007/11/21/rendertarget-changes-in-xna-game-studio-2-0.aspx
- XNA 4.0: http://blogs.msdn.com/b/shawnhar/archive/2010/03/26/rendertarget-changes-in-xna-game-studio-4-0.aspx