Ch 10: Game Event Management

Quiz # 4
Discussion
Moving the Camera

Toggle between first and third person view
Translate object
namespace Class2b3DModelMoving
{
    /// <summary>
    /// This is the main type for your game
    /// </summary>
    public class Game1 : Microsoft.Xna.Framework.Game
    {
        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;
    
        //model
        Model model;

        Vector3 modelPosition;
        Vector3 cameraPosition;
        float modelRotation;
        Vector3 cameraForward;

        int toggle = 0;

        public Game1()
        {
            graphics = new GraphicsDeviceManager(this);
            Content.RootDirectory = "Content";
        }
    
    } // Game1
} // namespace Class2b3DModelMoving
/// Allows the game to perform any initialization it needs to before starting to run.
/// This is where it can query for any required services and load any non-graphics
/// related content. Calling base.Initialize will enumerate through any components
/// and initialize them as well.
///
protected override void Initialize()
{
    modelPosition = Vector3.Zero;
    cameraPosition = new Vector3(0, 20, -50);
    cameraForward = new Vector3(0, 0, -1);
    modelRotation = 0.0f;

    base.Initialize();
}

/// LoadContent will be called once per game and is the place to load
/// all of your content.
///
protected override void LoadContent()
{
    // Create a new SpriteBatch, which can be used to draw textures.
    spriteBatch = new SpriteBatch(GraphicsDevice);    
    model = Content.Load<Model>("warlock");
}

/// UnloadContent will be called once per game and is the place to unload
/// anything you don't need for execution.
///
protected override void UnloadContent()
{ }
///<summary>
///Provides a snapshot of timing values.
///</summary>

protected override void Update(GameTime gameTime)
{
    // Allows the game to exit
        this.Exit();

    KeyboardState keyboardState = Keyboard.GetState();

    if ((gameTime.TotalGameTime.Milliseconds % 100) == 0)
    {
        Keys[] keyspressed = keyboardState.GetPressedKeys();

        foreach (Keys key in keyspressed)
        {
            if (key == Keys.Up)
                modelPosition.Z = modelPosition.Z + 2;
            if (key == Keys.Down)
                modelPosition.Z = modelPosition.Z - 2;
            if ((key == Keys.F) && (toggle == 0))
            {
                toggle = 1;
                cameraPosition = modelPosition;
                cameraPosition.Y += 29;
                cameraPosition.Z += 1;
            }
            else
        
    }
else
{
    if ((key == Keys.F) && (toggle == 1))
    {
        toggle = 0;
        cameraPosition = modelPosition;
        cameraPosition.Y += 20;
        cameraPosition.Z = 50;
    }
}

base.Update(gameTime);

/// <summary>
/// This is called when the game should draw itself.
/// </summary>
/// <param name="gameTime">Provides a snapshot of timing values.</param>
protected override void Draw(GameTime gameTime)
{
    // adding drawing for the model
    Matrix[] transforms = new Matrix[model.Bones.Count];
    model.CopyAbsoluteBoneTransformsTo(transforms);
}
```csharp
// Adding drawing for the model
Matrix[] transforms = new Matrix[model.Bones.Count];
model.CopyAbsoluteBoneTransformsTo(transforms);

// Draw the model. A model can have multiple meshes, so loop.
foreach (ModelMesh mesh in model.Meshes)
{
    // This is where the mesh orientation is set, as well
    // as our camera and projection.
    foreach (BasicEffect effect in mesh.Effects)
    {
        effect.EnableDefaultLighting();

        effect.World = transforms[mesh.ParentBone.Index] *
            Matrix.CreateRotationY(modelRotation) *
            Matrix.CreateTranslation(modelPosition);

        effect.View = Matrix.CreateLookAt(cameraPosition,

            1.0f, 1000.0f);
    }

    // Draw the mesh, using the effects set above.
    mesh.Draw();
}
```
Transformations

 Quaternion: rotation around arbitrary axis

 Advantages:

 - They simply never suffer from gimbal lock (rotations cancelling each other out)
 - Simple
 - You can easily create smooth transitions between several rotations
 - You can concatenate them (like matrices) to do several rotations at once
Class Work

✧ Put several 3D models in the scene
✧ Create a simple environment
✧ Make a Camera Fly through the scene
Effects

Chapter 6 from Tom Miller’s book
Effects

- Textures and Texture Maps
- Lighting
- Light Maps
Effects

- Environment Maps: is an approximation of how the environment is reflected on a reflective object.
Effects

✨ Bump Maps:
Is a technique in graphics to approximate wrinkles and bumps
Effects

✨ Lighting:

✨ Many different effects

From: beyond3D

From: opengl programming
Effects

❖ Lighting:
   ❖ Many different effects

From: Joshua Andersen’s work at Utah
Effects

✧ Lighting:

✧ Many different effects

From: Joshua Andersen’s work at Utah
Review from Graphics

Color and Texture Mapping
Color Models

- RGB Color Model
- HSV/HSL Color Model
- CIE XYZ Color Model
RGB Color Model

- Point is described in 3D space (R, G, B)
- 0,0,0 is black
- 1, 1, 1 is white

From: http://www.clear.rice.edu/elec301/Projects02/artSpy/color.html
HSL/HSV Model

- Transformation of RGB into a double cone
- Lightness 0-1, white to black, correspond to diagonal in RGB model

http://prosjekt.ffi.no/unik-4660/lectures04/chapters/Introduction.html

HSL/HSV Model

- Lightness = \frac{1}{\sqrt{3}}(R + G + B)

- Saturation = (radial distance)

\[ \sqrt{x^2 + y^2} \]

\[ x = \frac{1}{k_x} (2R - G - B), y = \frac{1}{k_y} (G - B) \]

- Hue = \tan^{-1}(x, y)
CIE-Lab Color Space

- Defined in X, Y, Z space corresponding to how the retina behaves to light

- RGB to XYZ space is linear transformation:

  \[
  \begin{bmatrix}
  R \\
  G \\
  B \\
  \end{bmatrix}
  =
  \begin{bmatrix}
  3.24 & -1.53 & -0.49 \\
  -0.96 & 1.87 & 0.04 \\
  0.05 & -0.20 & 1.05 \\
  \end{bmatrix}
  \begin{bmatrix}
  X \\
  Y \\
  Z \\
  \end{bmatrix}
  \]

Texture Mapping – basic idea

✧ You don’t want to model everything mathematically

✧ To give the environment more depth and realism, we can use images imposed on the models.

http://radoff.com/blog/2008/08/22/anatomy-of-an-mmorpg/

3D Textures – Texture Mapping Approaches

✧ Develop a texture mathematically, using

\[ f(x, y, z) = c \]

✧ Store values in 3D table and look up values

http://www.siggraph.org/education/materials/HyperGraph/mapping/r_wolfe/r_wolfe_mapping_6.htm
Example – procedural texture

- Using Sine wave to develop procedural texture

ProcTexture (x, y, z)
{
    if (sin(x) > 0)
        return Color.Red;
    else
        return Color.White;
}

You can see how this can be extended to 3D
Example – procedural texture

- Using Sine wave to develop procedural texture

```
ProcTexture (x, y, z) {
    if (sin(z) > 0)
        return Color.Red;
    else
        return Color.White;
}
```

From Hurst's Lecture Notes
Texture mapping – 2D

Problem: map \( i,j \) (texel) – \( u,v \) (pixel)
Texture mapping – 2D

One Solution (tiling):

- Remove the integer portion of $u, v$ (e.g., $0.9, 2.2 \rightarrow 0.9, 0.2$)
- Interpolate to find exact $i, j$
Texture mapping – 2D

One Solution (tiling):
- Remove the integer portion of $u$, $v$ (e.g., $0.9, 2.2 \rightarrow .9, .2$)
- **Nearest-neighbor interpolation** to find exact $i$, $j$

$$C(i, j) = C\left(\left\lfloor un_x \right\rfloor, \left\lfloor vn_y \right\rfloor \right)$$
Lighting Design

- Dramatic Tension
- Visual Focus
- Style
- Feel of the space
- Time of Day
- Period
- Visibility
Lighting in Traditional Media
Light Maps: Non-Interactive

Lighting through dual textures

Advantages:

❖ realistic
❖ Controllable

Disadvantages:

❖ Don’t adapt to variations in the environment
❖ Requires much memory

Images from Lightmaps (static shadowmaps) article written by Kurt Miller from:
http://www.flipcode.com/articles/article_lightmaps.shtml
Use of Dynamic Lighting
Dynamic lighting

This is mostly done through shaders and lighting effects
Lighting Models – Implementing Lighting Effects

- Shader Language
  - Cg (Nvidia)
  - HLSL (Microsoft derived from Nvidia)
  - GLSL (OpenGL)
Types of Shaders

- **Vertex Shaders**
  - Execute per vertex
  - Transform 3D to 2D
  - Manipulate position, color, texture
  - Cannot add new Vertices

- **Geometry Shaders**
  - Can add/remove vertices
  - Can Procedurally generate geometry

- **Pixel Shaders**
  - Execute per pixel
  - Calculate color for individual pixels
  - Calculates lighting, textures, etc.
BasicEffect simplifies the problem and allows you to quickly render basic effects without having to program shaders,

So let’s look at what the BasicEffect allows you to do
Public interface IEffectMatrices
{
    Matrix Projection { get; set; }
    Matrix View { get; set; }
    Matrix World { get; set; }
}
Lighting for basic effect

- BasicEffect uses per vertex lighting
- `effect.LightingEnabled = true;` will enable lighting on the model
- `effect.LightingEnabled = false;` switches off the lighting
Public interface IEffectLights

{ Vector3 AmbientLightColor {get; set; }

DirectionalLight DirectionalLight0 {get;}

DirectionalLight DirectionalLight1 {get;}

DirectionalLight DirectionalLight2 {get;}

bool LightingEnabled { get; set; }

Void EnableDefaultLighting(); }
Types of Lights

- Ambient
- Directional
- Point
- Spot Light

- Color: RGB color model (1,0,0)
Ambient Light

- Uniform color for all the scene
- Not very realistic and can be dull
- By default color is 0,0, 0 (black)
- Can be used to fill in shadow areas so they won’t be completely black, one way to simulate light reflections off of surfaces (global illumination)
Point Light

- Light emitted from a point in 3D space
- Light is emitted uniformly

To calculate, need angle between Normal and light direction:

Dot product, which determines how much of the light color is applied on vertex or pixel:

\[
\text{Color} = \text{base color} \times (\text{light intensity} \times \cos(\text{angle}) + \text{ambient});
\]

If parallel to normal: full color is applied
if Perpendicular to normal: no color
Directional Lights

- Light emitted using an angle spread
- Thus, they have a direction
- Used to simulate: sun light
Spot Light

- Light emitted using an angle spread
- Light attenuates over distance
Quality of Lighting

✩ Diffuse vs. Specular

✩ In XNA:

- Effect.SpecularColor = Color.White.ToVector3();
- Effect.SpecularPower = 100.0f;
- Effect.DirectionalLight0.SpecularColor
- Effect.DirectionalLight0.DiffuseColor

From: Joshua Andersen’s work at Utah
Basic Effects using XNA

```csharp
// drawing the model
foreach (ModelMesh mesh in warlock.Meshes)
{
    foreach (BasicEffect effect in mesh.effects)
    {
        effect.PreferPerPixelLighting = true;
        effect.LightingEnabled = true;
        effect.DirectionLight0.Enabled = true;
        // effect.AmbientLightColor = Color.White.ToVector3();
        effect.DirectionLight0.Direction = new Vector3(0.0f, 0.0f, 0.75f);
        effect.DirectionLight0.DiffuseColor = new Vector3(0.0f, 1.0f, 1.0f);
        // effect.DirectionLight0.SpecularColor = new Vector3(1.0f, 1.0f, 1.0f);
        effect.SpecularPower = 100.0f;
        effect.SpecularColor = new Vector3(1.0f, 0.0f, 0.0f);
        // effect.EnableDefaultLighting();
        // effect.DirectionLight0.Direction = Vector3.Left;
        // effect.DirectionLight0.DiffuseColor = new Vector3(255, 0, 0);
        effect.World = transforms[mesh.ParentBone.Index] *
            Matrix.CreateRotationY(MathHelper.PI) *
            Matrix.CreateTranslation(modelpos);
        effect.View = Matrix.CreateLookAt(cameraPos, new Vector3(cameraPos.X, cameraPos.Y, cameraPos.Z+100), Vector3.Up);
        effect.Projection = Matrix.CreatePerspectiveFieldOfView(MathHelper.PiOver4, GraphicsDevice.Viewport.AspectRatio, 10.0f, 100.0f);
        mesh.Draw();
    }
}
```

The field 'Class3aEffects.Game1.modeRotation' is never used
Basic Effects using XNA
In-class Exercise

✦ Take a model that you used and play around with the BasicEffect properties

✦ EnableDefaultLighting
✦ Add/Remove and change colors for Directional Lights
✦ Change Diffuse Light and Specular Light properties
  ✦ DiffuseColor
  ✦ SpecularColor in both the directional light and on effect
  ✦ SpecularPower
Per Pixel vs. Per Vertex Lighting

- In XNA: `Effect.PreferredPerPixelLighting = true;`
Other Effects

Public interface IEffectFog
{
    Vector3 FogColor {get; set;}
    bool FogEnabled {get; set;}
    float FogEnd { get; set;}
    float FogStart {get; set;}
}

Other Effects

- DualTextureEffect (Light Maps)
- EnvironmentMapEffect
Custom Shaders

Chapter 13 from Aaron’s Book
Effect File

float4x4 World;
float4x4 View;
float4x4 Projection;

// TODO: add effect parameters here.

struct VertexShaderInput
{
    float4 Position : POSITION0;

    // TODO: add input channels such as texture
    // coordinates and vertex colors here.
};

struct VertexShaderOutput
{
    float4 Position : POSITION0;

    // TODO: add vertex shader outputs such as colors and texture
    // coordinates here. These values will automatically be interpolated
    // over the triangle, and provided as input to your pixel shader.
};

VertexShaderOutput VertexShaderFunction(VertexShaderInput input)
{
    VertexShaderOutput output;

    float4 worldPosition = mul(input.Position, World);
    float4 viewPosition = mul(worldPosition, View);
    output.Position = mul(viewPosition, Projection);
float4 worldPosition = mul(input.Position, World);
float4 viewPosition = mul(worldPosition, View);
output.Position = mul(viewPosition, Projection);

    // TODO: add your vertex shader code here.
    
    return output;
    }

float4 PixelShaderFunction(VertexShaderOutput input) : COLOR0
{
    // TODO: add your pixel shader code here.
    return float4(1, 0, 0, 1);
}

technique Technique1
{
    pass Pass1
    {
        // TODO: set renderstates here.
        VertexShader = compile vs_2_0 VertexShaderFunction();
        PixelShader = compile ps_2_0 PixelShaderFunction();
    }
}
In XNA

✧ Add the effect into the content
✧ Effect myEffect; //declare the effect
✧ In Loading function:
  ✧ myeffect = Content.Load<Effect>("SampleEffect");
// drawing the model
foreach (ModelMesh mesh in warlock.Meshes)
{
    myeffect.CurrentTechnique = myeffect.Techiniques["Technique1"];  

                          * Matrix.CreateTranslation(modelpos));


foreach (ModelMeshPart meshpart in mesh.MeshParts)
{
    meshpart.Effect = myeffect;
}

mesh.Draw();
}
Assignment #2

✧ Scene graph for representing your scene

✧ Think about the architecture:
  ✧ Events?
  ✧ Effects?
  ✧ Movement?