ECE4893A/CS4803MPG: Multicore and GPU Programming for Video Games

More XNA Basics

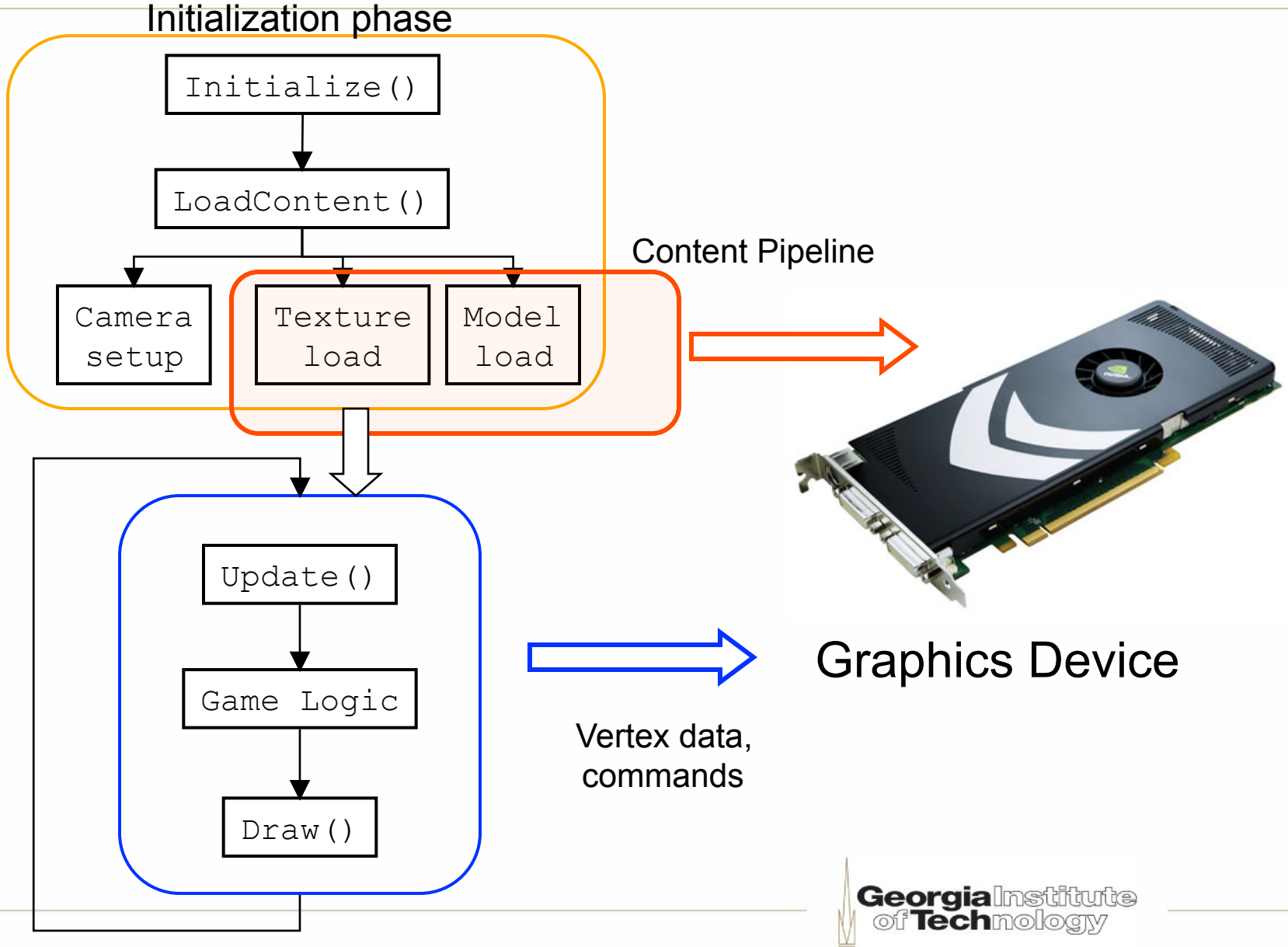
Prof. Hsien-Hsin Sean Lee

Prof. Aaron Lanterman

School of Electrical and Computer Engineering

Georgia Institute of Technology

The Big Picture



VECTOR

- Various types
  - VECTOR2 vec2 = new Vector2(100, 100)
  - VECTOR3 vec3 = new Vector3(10, 10, 10)
  - VECTOR4 vec4 = new Vector4(1, 1, 10, 2)

- Some Useful common vector functions
  - float len = v3.Length()
  - float dp = Vector3.Dot(Vector3, Vector3)
  - Vector3 cp = Vector3.Cross(Vector3, Vector3)
  - Vector3 lerp = Vector3.Lerp(Vector3, Vector3, float)
  - Vector3 dist = Vector3.Distance(Vector3, Vector3)
  - Vector3 nv = Vector3.Normalize(Vector3)
  - Check out http://msdn.microsoft.com for more

2nd vector's weight

VECTOR Example

Vectors

(See Demo in Visual Studio)

MATRIX Related

- MATRIX (4x4)

- Common matrix manipulations
  - Vector4 Rmat41= Vector4.Transform (Vector4, Matrix); // [4x1]*[4x4]

  - Matrix Rmat44 = Matrix.Multiply(Matrix, Matrix); // [4x4]* [4x4]
  - It is overloaded with operator *, thus <u>Rmat44=M1*M2</u>

  - Matrix INV = Matrix.Invert(Matrix); // find its inverse
  - Matrix TR = Matrix.Transpose(Matrix); // transpose
  - Matrix IM = Matrix.Identity; // return an identity matrix

MATRIX Example

SimpleMatrix

(See Demo in Visual Studio)

Plane

- PLANE plane1 = new Plane(Vector3, d)

  vector3 describes a normal vector

  d = plane's distance from the origin

- Useful plane function
  - Plane.intersects(boundingbox box)
  - Many others, check msdn

Difference Between Direct3D and XNA

- You always need an “effect file” (.fx)
- An effect file is a shader code consisting of
  - Vertex Shader
  - Pixel Shader
- Need to communicate varibales between your C# code and the shader code, e.g., matrices
- Even rendering a simple triangle requiring an effect file
  - Use a BasicEffect

Specifying an Effect

- Either use a BasicEffect or specify a provided .fx file

```
public class Game1 : Microsoft.Xna.Framework.Game
{
    GraphicsDeviceManager graphics;
    SpriteBatch spriteBatch;

    private BasicEffect effect;
```

```
#if LEE_EFFECT
            effect = Content.Load<Effect>("effects");
#else
            effect = new BasicEffect(graphics.GraphicsDevice, null);
#endif
```

An effect file called effects.fx is given

```
#if LEE_EFFECT
            effect.CurrentTechnique = effect.Techniques["NoTechniques"];
#endif
            effect.Begin();
            foreach (EffectPass pass in effect.CurrentTechnique.Passes)
            {
                pass.Begin();

                pass.End();
            }
            effect.End();
```

Passes specified in the shader will be processed

Set Up BasicEffect

```
private void InitializeEffect()
{
    LeeVertexDecl = new VertexDeclaration(graphics.GraphicsDevice,
                                          VertexPositionNormalTexture.VertexElements);

    effect = new BasicEffect(graphics.GraphicsDevice, null);
    effect.Alpha = 1.0f;
    effect.DiffuseColor = new Vector3(1.0f, 1.0f, 1.0f);
    effect.SpecularColor = new Vector3(0.25f, 0.25f, 0.25f);
    effect.SpecularPower = 5.0f;
    effect.AmbientLightColor = new Vector3(1.0f, 1.0f, 1.0f);

    effect.DirectionalLight0.Enabled = true;
    effect.DirectionalLight0.DiffuseColor = Vector3.One;
    effect.DirectionalLight0.Direction = Vector3.Normalize(new Vector3(1.0f, -1.0f, -1.0f));
    effect.DirectionalLight0.SpecularColor = Vector3.One;

    effect.DirectionalLight1.Enabled = true;
    effect.DirectionalLight1.DiffuseColor = new Vector3(0.5f, 0.5f, 0.5f);
    effect.DirectionalLight1.Direction = Vector3.Normalize(new Vector3(-1.0f, -1.0f, 1.0f));
    effect.DirectionalLight1.SpecularColor = new Vector3(0.5f, 0.5f, 0.5f);


    effect.LightingEnabled = true;
    effect.VertexColorEnabled = false;

    effect.TextureEnabled = true;
    effect.Texture = GT;

    effect.World = worldMatrix;
    effect.View = viewMatrix;
    effect.Projection = projectionMatrix;
}
```

BasicEffect: Just a Shader

- Internally set via the “ShaderIndex” parameter:
  - 0: Flat color
  - 1: Vertex colors
  - 2: Textured
  - 3: Textured + Vertex colors
  - 4: Lit
  - 5: Lit + Vertex colors
  - 6: Lit + Textured
  - 7: Lit + Textured + Vertex colors
  - 8: Per-pixel lit
  - 9: Per-pixel lit + Vertex colors
  - 10: Per-pixel lit + Textured
  - 11: Per-pixel lit + Textured + Vertex colors

http://creators.xna.com/en-us/utilities/basiceffectshader

BasicEffect.fx (the end of the file)

~/seanlee/Georgia_Tech/Courses/ECE4893GraphicsProgramming/2009_XNA/ClassExamples/BasicEffectShader

```
VertexShader VSArray[12] =
{
        compile vs_1_1 VSBasic(),
        compile vs_1_1 VSBasicVc(),
        compile vs_1_1 VSBasicTx(),
        compile vs_1_1 VSBasicTxVc(),

        compile vs_1_1 VSBasicNm(),
        compile vs_1_1 VSBasicNmVc(),
        compile vs_1_1 VSBasicNmTx(),
        compile vs_1_1 VSBasicNmTxVc(),

        compile vs_1_1 VSBasicPixelLightingNm(),
        compile vs_1_1 VSBasicPixelLightingNmVc(),
        compile vs_1_1 VSBasicPixelLightingNmTx(),
        compile vs_1_1 VSBasicPixelLightingNmTxVc(),
};


PixelShader PSArray[12] =
{
        compile ps_1_1 PSBasic(),
        compile ps_1_1 PSBasic(),
        compile ps_1_1 PSBasicTx(),
        compile ps_1_1 PSBasicTx(),
        compile ps_1_1 PSBasic(),
        compile ps_1_1 PSBasic(),
        compile ps_1_1 PSBasicTx(),
        compile ps_1_1 PSBasicTx(),

        compile ps_2_0 PSBasicPixelLighting(),
        compile ps_2_0 PSBasicPixelLighting(),
        compile ps_2_0 PSBasicPixelLightingTx(),
        compile ps_2_0 PSBasicPixelLightingTx(),
};


Technique BasicEffect
{
        Pass
        {
                VertexShader = (VSArray[ShaderIndex]);
                PixelShader      = (PSArray[ShaderIndex]);
        }
}

leehs@LENOVO-D84906F4 ~/seanlee/Georgia_Tech/Courses/ECE4893GraphicsProgramming/2009_XNA/ClassExamples/BasicEffectShader
$
```

BasicEffect Handles 3 Directional Lights

- Parameters
  - AmbientLightColor
  - DirLight0Direction
  - DirLight0DiffuseColor
  - DirLight0SpecularColor
  - DirLight1Direction
  - DirLight1DiffuseColor
  - DirLight1SpecularColor
  - DirLight2Direction
  - DirLight2DiffuseColor
  - DirLight2SpecularColor

http://creators.xna.com/en-us/utilities/basiceffectshader

More BasicEffect Parameters

- Material properties (colors range 0 to 1)
  - DiffuseColor
  - EmissiveColor
  - SpecularColor
  - SpecularPower
  - Alpha
- Fog properties
  - FogEnabled (0 to disable, 1 to enable)
  - FogStart
  - FogEnd
  - FogColor

http://creators.xna.com/en-us/utilities/basiceffectshader

DrawUserPrimitives

```
private VertexPositionColor[] vertex;
private VertexDeclaration LeeVertexDecl;
```

```
private void TriangleOne()
{
    vertex = new VertexPositionColor[3];
    vertexcount = 3;

    vertex[0] = new VertexPositionColor(new Vector3(-0.5f, -0.5f, 0.0f), Color.Red);
    vertex[1] = new VertexPositionColor(new Vector3(0.0f, 0.5f, 0.0f), Color.Green);
    vertex[2] = new VertexPositionColor(new Vector3(0.5f, -0.5f, 0.0f), Color.Blue);

    LeeVertexDecl = new VertexDeclaration(graphics.GraphicsDevice,
                                          VertexPositionColor.VertexElements);
}
```

Declare the type of vertices

```
            // before draw it, give the declaration to the gfx device
            graphics.GraphicsDevice.VertexDeclaration = LeeVertexDecl;


#if LEE_EFFECT
            effect.CurrentTechnique = effect.Techniques["NoTechniques"];
#endif
            effect.Begin();
            foreach (EffectPass pass in effect.CurrentTechnique.Passes)
            {
                pass.Begin();

                graphics.GraphicsDevice.DrawUserPrimitives(PrimitiveType.TriangleList, vertex, 0, 1);

                pass.End();
            }
            effect.End();
```

Inform the gfx device the type of vertices

Give vertex list

Offset

# of Triangles

Use Vertex Buffer (DrawPrimitives)

- Create a container Vertex Buffer for allocating vertices

```
private VertexBuffer VBuffer;
```

```
private void TriangleOne()
{
    vertex = new VertexPositionColor[3];
    vertexcount = 3;

    vertex[0] = new VertexPositionColor(new Vector3(-0.5f, -0.5f, 0.0f), Color.Red);
    vertex[1] = new VertexPositionColor(new Vector3(0.0f, 0.5f, 0.0f), Color.Green);
    vertex[2] = new VertexPositionColor(new Vector3(0.5f, -0.5f, 0.0f), Color.Blue);


    VBuffer = new VertexBuffer(graphics.GraphicsDevice,
                               vertexcount * VertexPositionColor.SizeInBytes, BufferUsage.WriteOnly);

    VBuffer.SetData<VertexPositionColor>(vertex);

    LeeVertexDecl = new VertexDeclaration(graphics.GraphicsDevice,
                                          VertexPositionColor.VertexElements);
}
```

Create VertexBuffer

Indicate source of data

```
            graphics.GraphicsDevice.VertexDeclaration = LeeVertexDecl;
            graphics.GraphicsDevice.Vertices[0].SetSource(VBuffer, 0, VertexPositionColor.SizeInBytes);


#if LEE_EFFECT
            effect.CurrentTechnique = effect.Techniques["NoTechniques"];
#endif
            effect.Begin();
            foreach (EffectPass pass in effect.CurrentTechnique.Passes)
            {
                pass.Begin();

                graphics.GraphicsDevice.DrawPrimitives(PrimitiveType.TriangleList, 0, 1);

                pass.End();
            }
            effect.End();
```

Set starting source vertex for gfx device

One less argument than DrawUserPrimitives

VertexBuffer

- Contains list of vertices
  - 4-D colors
  - 3-D positions
  - 2-D texture coordinates
  - 3-D normals
- Structures
  - VertexPositionColor
  - VertexPositionColorTexture
  - VertexPositionNormalTexture
  - VertexPositionTexture
  - Can make your own with VertexElement; See "How to: Create and Use a Custom Vertex" http://msdn.microsoft.com/en-us/library/bb976065.aspx

PrimitiveType Enumeration

Member name	Description
LineList	Renders the vertices as a list of isolated straight line segments; the count may be any positive integer.
LineStrip	Renders the vertices as a single polyline; the count may be any positive integer.
PointList	Renders the vertices as a collection of isolated points. This value is unsupported for indexed primitives.
TriangleFan	Renders the vertices as a triangle fan.
TriangleList	Renders the specified vertices as a sequence of isolated triangles. Each group of three vertices defines a separate triangle. Back-face culling is affected by the current winding-order render state.
TriangleStrip	Renders the vertices as a triangle strip. The back-face culling flag is flipped automatically on even-numbered triangles.

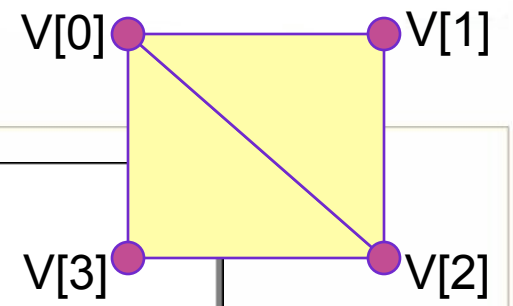
http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.graphics.priamitivetype.aspx

DrawPrimitves Example

FirstTriangle

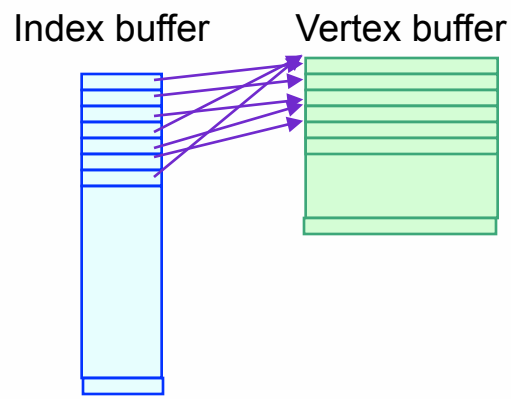
(See Demo in Visual Studio)

Create Indexed Vertex Buffer



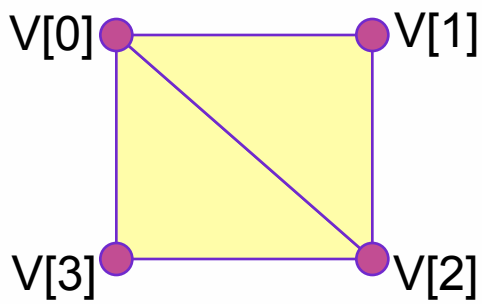

```
private void TriangleOne()
{
    Vector3 FrontNormal = new Vector3(0.0f, 0.0f, 1.0f);

    vertex = new VertexPositionNormalTexture[4];
    vertexcount = 4;

    vertex[0] = new VertexPositionNormalTexture(new Vector3(-0.5f,  0.5f, 0.0f), FrontNormal, new Vector2(0.0f, 0.0f));
    vertex[1] = new VertexPositionNormalTexture(new Vector3( 0.5f,  0.5f, 0.0f), FrontNormal, new Vector2(1.0f, 0.0f));
    vertex[2] = new VertexPositionNormalTexture(new Vector3( 0.5f, -0.5f, 0.0f), FrontNormal, new Vector2(1.0f, 1.0f));
    vertex[3] = new VertexPositionNormalTexture(new Vector3(-0.5f, -0.5f, 0.0f), FrontNormal, new Vector2(0.0f, 1.0f));

    triangleListIndices = new short[6] { 0, 1, 2, 2, 3, 0 };
}
```

```
effect.Begin();
foreach (EffectPass pass in effect.CurrentTechnique.Passes)
{
    pass.Begin();

    graphics.GraphicsDevice.DrawUserIndexedPrimitives(PrimitiveType.TriangleList,
                                        vertex, 0, 4, triangleListIndices, 0, 2);

    pass.End();
}
effect.End();
```



- To eliminate the repetitive definition of vertices
- Use a *vertex buffer* to store unique vertices
- Use an *index buffer* to store Index

DrawUserIndexedPrimitves




```
effect.Begin();
foreach (EffectPass pass in effect.CurrentTechnique.Passes)
{
    pass.Begin();

    graphics.GraphicsDevice.DrawUserIndexedPrimitives(PrimitiveType.TriangleList,
                                                      vertex,
                                                      0,
                                                      4,
                                                      triangleListIndices,
                                                      0,
                                                      2);

    pass.End();
}
effect.End();
```

Primitive Type

T[] vertex data

# of Vertices

Vertex offset

Index Offset

Short[] index data

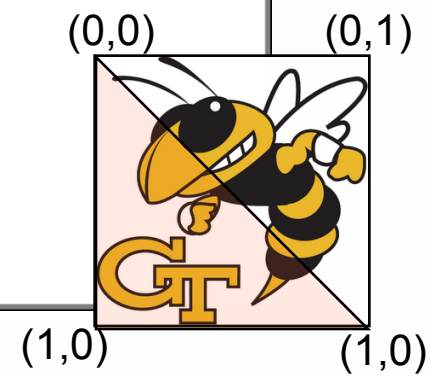
# of Primitive to render

Indexed Vertex Structure

Define vertices of a Cube
VertexPositionNormalTexture

Texture Coordinates

```
vertex = new VertexPositionNormalTexture[24];
vertexcount = 24;

// Front
vertex[0] = new VertexPositionNormalTexture(new Vector3(-0.5f,  0.5f, 0.0f), FrontNormal, new Vector2(0.0f, 0.0f));
vertex[1] = new VertexPositionNormalTexture(new Vector3( 0.5f,  0.5f, 0.0f), FrontNormal, new Vector2(0.5f, 0.0f));
vertex[2] = new VertexPositionNormalTexture(new Vector3( 0.5f, -0.5f, 0.0f), FrontNormal, new Vector2(0.5f, 0.5f));
vertex[3] = new VertexPositionNormalTexture(new Vector3(-0.5f, -0.5f, 0.0f), FrontNormal, new Vector2(0.0f, 0.5f));

// Right
vertex[4] = new VertexPositionNormalTexture(new Vector3(0.5f,  0.5f,  0.0f), RightNormal, new Vector2(0.0f, 0.0f));
vertex[5] = new VertexPositionNormalTexture(new Vector3(0.5f,  0.5f, -1.0f), RightNormal, new Vector2(1.0f, 0.0f));
vertex[6] = new VertexPositionNormalTexture(new Vector3(0.5f, -0.5f, -1.0f), RightNormal, new Vector2(1.0f, 1.0f));
vertex[7] = new VertexPositionNormalTexture(new Vector3(0.5f, -0.5f,  0.0f), RightNormal, new Vector2(0.0f, 1.0f));

// Top
vertex[8] =  new VertexPositionNormalTexture(new Vector3(-0.5f, 0.5f,  0.0f), TopNormal, new Vector2(0.0f, 1.0f));
vertex[9] =  new VertexPositionNormalTexture(new Vector3( 0.5f, 0.5f,  0.0f), TopNormal, new Vector2(1.0f, 1.0f));
vertex[10] = new VertexPositionNormalTexture(new Vector3( 0.5f, 0.5f, -1.0f), TopNormal, new Vector2(1.0f, 0.0f));
vertex[11] = new VertexPositionNormalTexture(new Vector3(-0.5f, 0.5f, -1.0f), TopNormal, new Vector2(0.0f, 0.0f));

//Left
vertex[12] = new VertexPositionNormalTexture(new Vector3(-0.5f,  0.5f,  0.0f), LeftNormal, new Vector2(2.0f, 0.0f));
vertex[13] = new VertexPositionNormalTexture(new Vector3(-0.5f, -0.5f,  0.0f), LeftNormal, new Vector2(2.0f, 2.0f));
vertex[14] = new VertexPositionNormalTexture(new Vector3(-0.5f, -0.5f, -1.0f), LeftNormal, new Vector2(0.0f, 2.0f));
vertex[15] = new VertexPositionNormalTexture(new Vector3(-0.5f,  0.5f, -1.0f), LeftNormal, new Vector2(0.0f, 0.0f));

//Back
vertex[16] = new VertexPositionNormalTexture(new Vector3( 0.5f,  0.5f, -1.0f), BackNormal, new Vector2(0.0f, 0.0f));
vertex[17] = new VertexPositionNormalTexture(new Vector3( 0.5f, -0.5f, -1.0f), BackNormal, new Vector2(0.0f, 4.0f));
vertex[18] = new VertexPositionNormalTexture(new Vector3(-0.5f, -0.5f, -1.0f), BackNormal, new Vector2(4.0f, 4.0f));
```



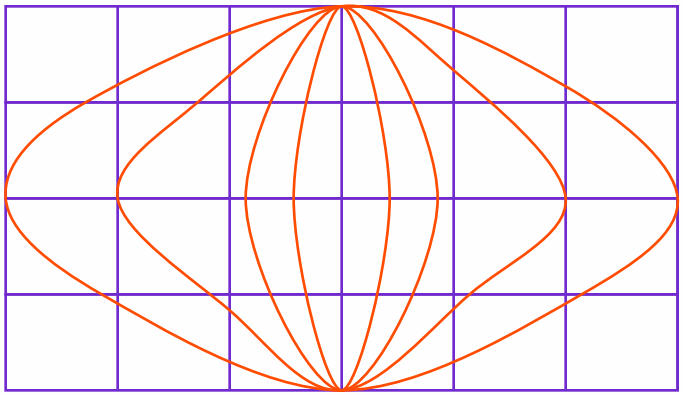

```
triangleListIndices = new short[36] {   0, 1, 2,
                                        2, 3, 0,
                                        4, 5, 6,
                                        4, 6, 7,
                                        8, 11, 10,
                                        8, 10, 9,
                                        12, 13, 14,
                                        15, 12, 14,
                                        16, 19, 17,
                                        19, 18, 17,
                                        20, 21, 22,
                                        23, 20, 22};
```

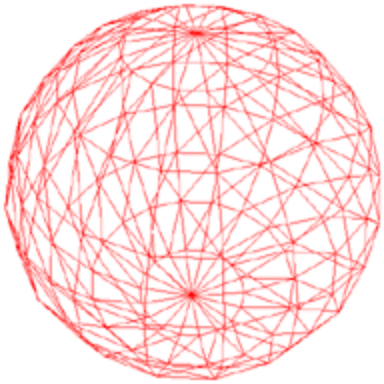
```
pass.Begin();

graphics.GraphicsDevice.DrawUserIndexedPrimitives(PrimitiveType.TriangleList,
                                                  vertex,
                                                  0,
                                                  24,
                                                  triangleListIndices,
                                                  0,
                                                  12);

pass.End();
```

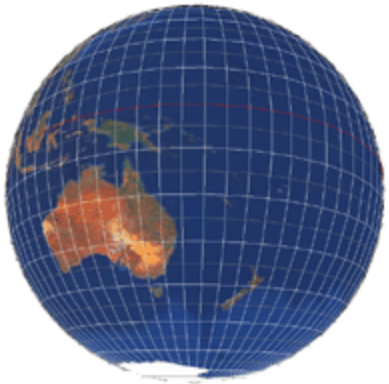
In Draw()

Texture Mapping for Sphere

- Warping the map could lead to distortion
- Geometric calculation involved







Globe image source: P. Bourke from uwa.edu.au

MATRIX Method Examples

- Translation (by amount to translate)

```
Matrix.CreateTranslation(Vector3 position);
```

- Scaling (by amount to scale in x, y, and z)

```
Matrix.CreateScale(Vector3 scales);
```

- Rotation

```
Matrix.CreateRotationX(float radians);
Matrix.CreateRotationY(float radians);
Matrix.CreateRotationZ(float radians);
```

- Special function

```
Matrix.CreateShadow(Vector3 lightDir, Plane plane);
```

Movement Control

```
if (keyboard.IsKeyDown(Keys.Up))
{
    zoom -= 0.1f;
}
else if (keyboard.IsKeyDown(Keys.Down))
{
    zoom += 0.1f;
}
else if (keyboard.IsKeyDown(Keys.Left))
{
    move -= 0.1f;
}
else if (keyboard.IsKeyDown(Keys.Right))
{
    move += 0.1f;
}
```

Composite (*) into “one matrix” for effect file

```
Movement();
rot += offset*speed;

worldMatrix = Matrix.CreateRotationY(rot.Y) * Matrix.CreateRotationX(rot.X) * Matrix.CreateRotationZ(rot.Z)*
                Matrix.CreateTranslation(move, updown, zoom);
effect.World = worldMatrix;
```

Be cautious on operation ordering

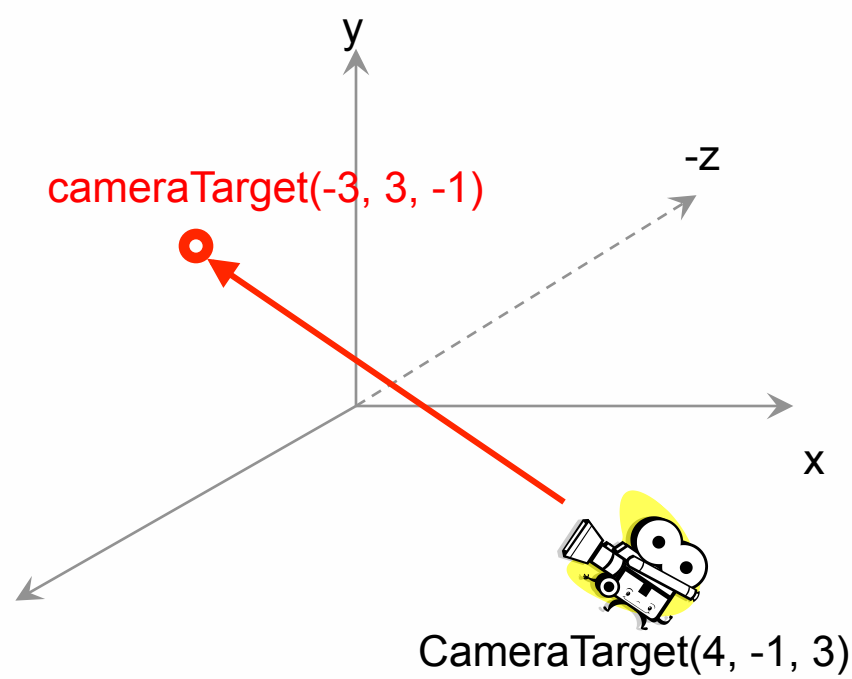
MATRIX Method for View Transformation

- View Transformation

```
Matrix.CreateLookAt(Vector3 cameraPosition,
                    Vector3 cameraTarget,
                    Vector3 cameraUpVector);
```

Usually (0, 1, 0) → the world is upward

Use Vector3.Up

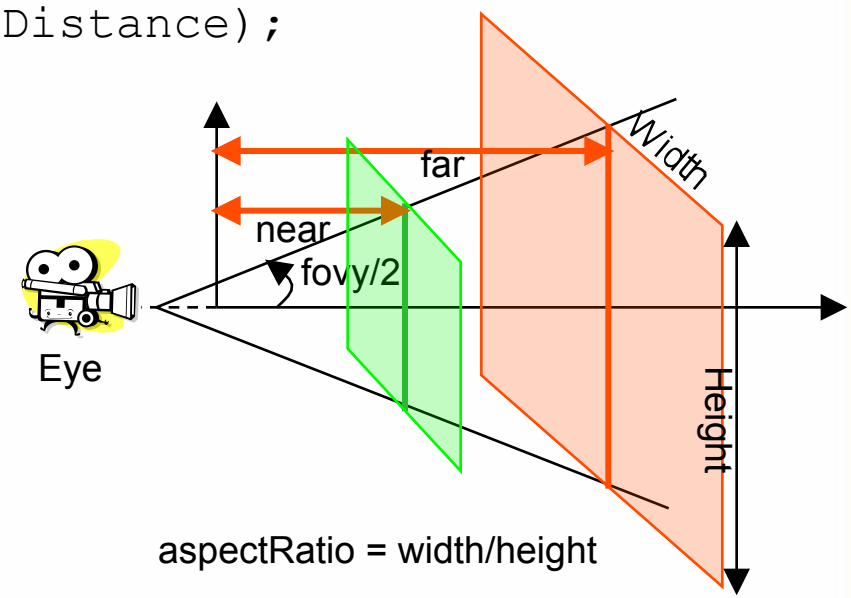


MATRIX Method for Projection Transformation

- Build a Look-at Matrix

```
Matrix.CreatePerspectiveFieldOfView(
                    float fieldOfView, // In radians
                    float aspectRatio,
                    float nearPlaneDistance,
                    float farPlaneDistance);
```

- Can use MathHelper.ToRadians(degree) for fieldOfView



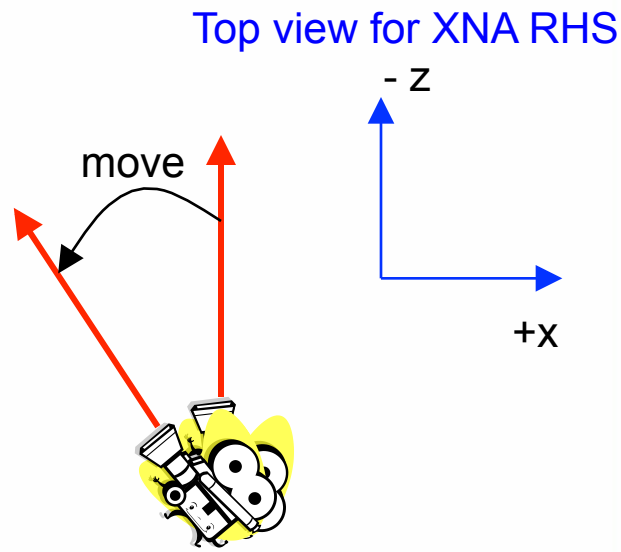
DrawUserIndexedPrimitves Example

DrawIndexedCube
(See Demo in Visual Studio)

Note that, in this example, all controls move the "Object," *not* the viewer.

Rotate Camera (Looking Around)

```
protected void CamMove()
{
    KeyboardState keyboard = Keyboard.GetState();

    if (keyboard.IsKeyDown(Keys.Left))
    {
        move -= 0.01f;
    }
    else if (keyboard.IsKeyDown(Keys.Right))
    {
        move += 0.01f;
    }

    camLookat.X = radius * (float) Math.Sin(move);
    camLookat.Z = 0.0f - radius * (float) Math.Cos(move);
    camLookat.Y = 0.0f;
}
```

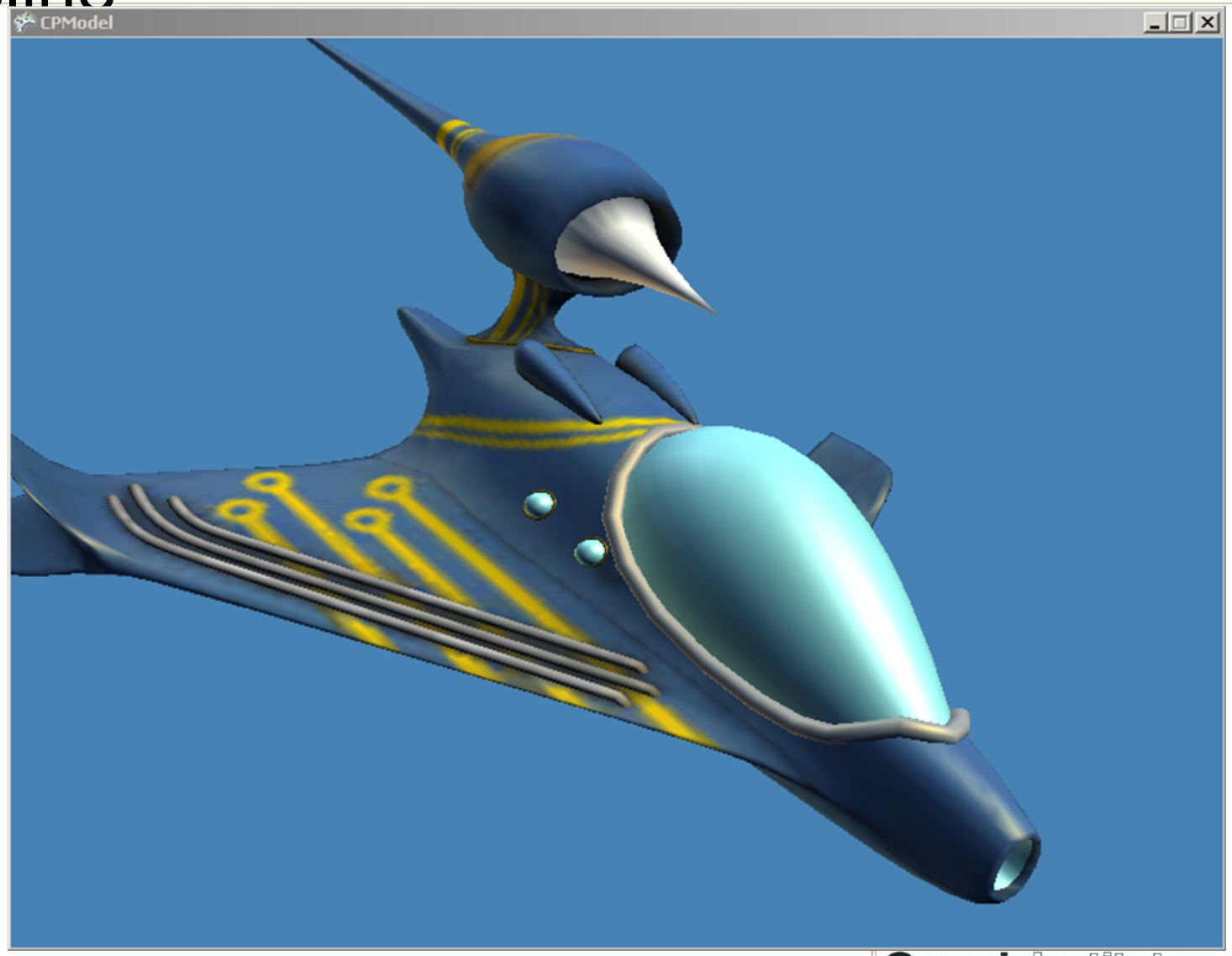



```
// Camera movement
CamMove();

viewMatrix = Matrix.CreateLookAt(camPos, camLookat, Vector3.Up);
effect.View = viewMatrix;
```

Moving Camera Example

MovingCam
(See Demo in Visual Studio)

How to Render a Model in XNA Content Pipeline



http://msdn.microsoft.com/en-us/library/bb203933.aspx

How to Peek at the Model Structure

```
protected override void LoadContent()
{
    gameShip = Content.Load<Model>("ship");

    System.Diagnostics.Debugger.Break();

}
```

.x model's
Asset name

Locals

Name	Value	Type
this	{CPModel.Game1}	CPModel.Game1
base	{CPModel.Game1}	Microsoft.Xna.Framework.Game {CPModel.Game1}
gameShip	{Microsoft.Xna.Framework.Graphics.Model}	Microsoft.Xna.Framework.Graphics.Model
gameWorldRotation	{ {M11:0 M12:0 M13:0 M14:0} {M21:0 M22:0 M23:0 M	Microsoft.Xna.Framework.Matrix
graphics	{Microsoft.Xna.Framework.GraphicsDeviceManager}	Microsoft.Xna.Framework.GraphicsDeviceManager
Position	{X:1 Y:1 Z:1}	Microsoft.Xna.Framework.Vector3
RotationX	0.0	float
RotationY	0.0	float
Zoom	2500.0	float

- The asset name of input model "wedge_player1.x" has been renamed to "ship" in the "properties" window (Content Pipeline)

Bones and Meshes

gameShip	{Microsoft.Xna.Framework.Graphics.Model}	Microsoft.Xna.Framework.Graphics.Model
Bones	Count = 2	Microsoft.Xna.Framework.Graphics.ModelBoneColle
[0]	{Microsoft.Xna.Framework.Graphics.ModelBone}	Microsoft.Xna.Framework.Graphics.ModelBone
[1]	{Microsoft.Xna.Framework.Graphics.ModelBone}	Microsoft.Xna.Framework.Graphics.ModelBone
Raw View		
Meshes	Count = 1	Microsoft.Xna.Fra
[0]	{Microsoft.Xna.Framework.Graphics.ModelMesh}	Microsoft.Xna.Framework.Graphics.ModelMesh
BoundingSphere	{Center:{X:0.8337776 Y:197.7493 Z:-513.1478} Radiu	Microsoft.Xna.Framework.BoundingSphere
Effects	Count = 3	Microsoft.Xna.Framework.Graphics.ModelEffectCol
IndexBuffer	{Microsoft.Xna.Framework.Graphics.IndexBuffer}	Microsoft.Xna.Framework.Graphics.IndexBuffer
MeshParts	Count = 3	Microsoft.Xna.Framework.Graphics.ModelMeshPart
Name	"single_mesh_object"	string
ParentBone	{Microsoft.Xna.Framework.Graphics.ModelBone}	Microsoft.Xna.Framework.Graphics.ModelBone
Tag	null	object
VertexBuffer	{Microsoft.Xna.Framework.Graphics.VertexBuffer}	Microsoft.Xna.Framework.Graphics.VertexBuffer
Non-Public members		
Raw View		
Root	{Microsoft.Xna.Framework.Graphics.ModelBone}	Microsoft.Xna.Framework.Graphics.ModelBone
Tag	null	object
Non-Public members		

A Model

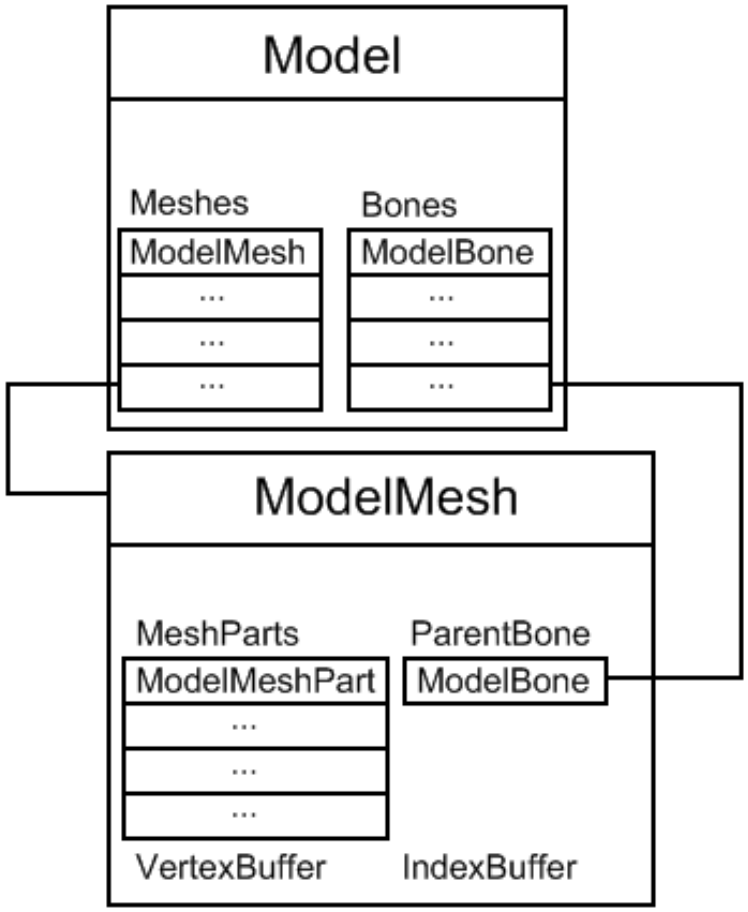
Meshes
Collection of geometry that move together

Bones
Representation of the Hierarchical Structure

- The model is “flattened” to a single list of meshes.
- The bones retain the tree structure from the scene graph
- Each mesh can have a related matrix defining its own transformation

Source: Charles Owen from MSU

Model Bone and Model Meshes



- Each ModelMesh has a ParentBone
- Each Bone has a transformation matrix
- Rotate the ParentBone will also rates the children and their children, etc.

In meshes: BoundingSpheres and Effects

Meshes	Count = 1	Microsoft.Xna.Framework.Graphics.ModelMeshColl
[0]	{Microsoft.Xna.Framework.Graphics.ModelMesh}	Microsoft.Xna.Framework.Graphics.ModelMesh
BoundingSphere	{Center:{X:0.8337776 Y:197.7493 Z:-513.1478} Radiu	Microsoft.Xna.Framework.BoundingSphere
Center	{X:0.8337776 Y:197.7493 Z:-513.1478}	Microsoft.Xna.Framework.Vector3
Radius	1609.95264	float
Effects	Count = 3	Microsoft.Xna.Framework.Graphics.ModelEffectCo
[0]	{Microsoft.Xna.Framework.Graphics.BasicEffect}	Microsoft.Xna.Framework.Graphics.Effect {Microso
[Microsoft.Xna.Fran	{Microsoft.Xna.Framework.Graphics.BasicEffect}	Microsoft.Xna.Framework.Graphics.BasicEffect
Creator	"D3DX Effect Compiler"	string
CurrentTechnique	{Microsoft.Xna.Framework.Graphics.EffectTechnique}	Microsoft.Xna.Framework.Graphics.EffectTechniqu
Annotations	{Microsoft.Xna.Framework.Graphics.EffectAnnotationC	Microsoft.Xna.Framework.Graphics.EffectAnnotati
Name	"BasicEffect"	string
Passes	{Microsoft.Xna.Framework.Graphics.EffectPassCollecti	Microsoft.Xna.Framework.Graphics.EffectPassColl
Non-Public memb		
EffectPool	null	Microsoft.Xna.Framework.Graphics.EffectPool
Functions	{Microsoft.Xna.Framework.Graphics.EffectFunctionColl	Microsoft.Xna.Framework.Graphics.EffectFunction
GraphicsDevice	{Microsoft.Xna.Framework.Graphics.GraphicsDevice}	Microsoft.Xna.Framework.Graphics.GraphicsDevice
IsDisposed	false	bool
Parameters	{Microsoft.Xna.Framework.Graphics.EffectParameterC	Microsoft.Xna.Framework.Graphics.EffectParamet
Techniques	{Microsoft.Xna.Framework.Graphics.EffectTechniqueC	Microsoft.Xna.Framework.Graphics.EffectTechniqu
Non-Public members		
[1]	{Microsoft.Xna.Framework.Graphics.BasicEffect}	Microsoft.Xna.Framework.Graphics.Effect {Microso
[2]	{Microsoft.Xna.Framework.Graphics.BasicEffect}	Microsoft.Xna.Framework.Graphics.Effect {Microso
Raw View		

Georgia Institute of Technology

In meshes: IndexBuffer & VertexBuffer

Meshes	Count = 1	Microsoft.Xna.Framework.Graphics.ModelMeshCo
[0]	{Microsoft.Xna.Framework.Graphics.ModelMesh}	Microsoft.Xna.Framework.Graphics.ModelMesh
BoundingSphere	{Center:{X:0.8337776 Y:197.7493 Z:-513.1478} Radiu	Microsoft.Xna.Framework.BoundingSphere
Effects	Count = 3	Microsoft.Xna.Framework.Graphics.ModelEffectC
IndexBuffer	{Microsoft.Xna.Framework.Graphics.IndexBuffer}	Microsoft.Xna.Framework.Graphics.IndexBuffer
base	{Microsoft.Xna.Framework.Graphics.IndexBuffer}	Microsoft.Xna.Framework.Graphics.GraphicsResc
BufferUsage	None	Microsoft.Xna.Framework.Graphics.BufferUsage
IndexElementSize	SixteenBits	Microsoft.Xna.Framework.Graphics.IndexElemen
SizeInBytes	96708	int
Non-Public members		
MeshParts	Count = 3	Microsoft.Xna.Framework.Graphics.ModelMeshPa
Name	"single_mesh_object"	string
ParentBone	{Microsoft.Xna.Framework.Graphics.ModelBone}	Microsoft.Xna.Framework.Graphics.ModelBone
Tag	null	object
VertexBuffer	{Microsoft.Xna.Framework.Graphics.VertexBuffer}	Microsoft.Xna.Framework.Graphics.VertexBuffer
base	{Microsoft.Xna.Framework.Graphics.VertexBuffer}	Microsoft.Xna.Framework.Graphics.GraphicsResc
BufferUsage	None	Microsoft.Xna.Framework.Graphics.BufferUsage
SizeInBytes	285632	int
Non-Public members		

Georgia Institute of Technology

Loading an .X model

```
gameShip = Content.Load<Model>("ship");
```

.x model's Asset name

```
foreach (ModelMesh mesh in m.Meshes)
{
  foreach (BasicEffect effect in mesh.Effects)
  {
        effect.EnableDefaultLighting();

        effect.View = view;
        effect.Projection = projection;
        effect.World = gameWorldRoattion *
                     transforms[mesh.ParentBone.Index] *
                     Matrix.CreateTranslation(Position);
   }
   mesh.Draw();
}
```

Draw all the meshes in the model

DrawModel (Initialization)

```
private void DrawModel(Model m)
{
        // copy any parent transform
        Matrix[] transforms = new Matrix[m.Bones.Count];
        m.CopyAbsoluteBoneTransformsTo(transforms);

        float aspectRatio =
                graphics.GraphicsDevice.Viewport.Width /
                graphics.GraphicsDevice.Viewport.Height;

        Matrix projection =
        Matrix.CreatePerspectiveFieldOfView
(MathHelper.ToRadians(45.0f), aspectRatio, 1.0f, 10000.0f);
        Matrix view =
                Matrix.CreateLookAt(new Vector3(0.0f, 50.0f, Zoom),
                 Vector3.Zero, Vector3.Up);
```

CopyAbsoluteBoneTransformsTo

Remarks

In an absolute transform, each bone is transformed according to the position of all parent bones.

This is the same as iterating the Bones collection and applying the transformation matrix of every parent of a ModelBone to the Transform property of each ModelBone and copying the results into an array that can be indexed into by the bone index.

An array of transformation matrices for the meshes of any model can be obtained by calling CopyAbsoluteBoneTransformsTo or CopyBoneTransformsTo. The resulting array contains the transforms that describe how each ModelMesh is located relative to one another in the Model. The transformation matrix that should be applied to each ModelMesh can be obtained using the index of the **CopyAbsoluteBoneTransformsTo** to retrieve a transformation matrix from this collection.

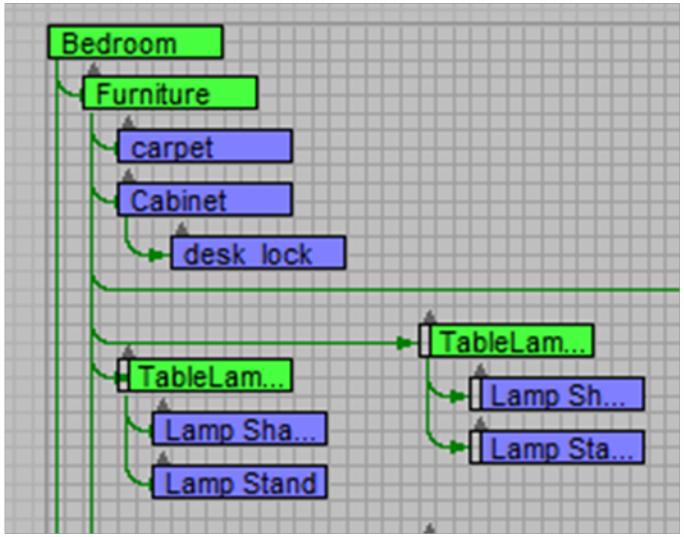
http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.graphics.model.copyabsolutebonetransformsto.aspx

- In an absolute transform, each bone is transformed according to the position of all parent bones

The Absolute Transform for a Lamp Shade

Product of 4 transformation matrices

LampShade * TableLamp *
Furniture * Bedroom



Source: Charles Owen from MSU

Ship Example

CPModel Effect

(See Demo in Visual Studio)

Ducky Example

DuckBasicEffect
(See Demo in Visual Studio)

Draw a Subset of a Model

Set up mesh IB and VB yourself

```
                graphics.GraphicsDevice.Indices = mesh.IndexBuffer;
                graphics.GraphicsDevice.Vertices[0].SetSource(mesh.VertexBuffer, 0, VertexPositionNormalTexture.SizeInBytes);

                // HHLee: the following code is pain in the ass to draw the first 2 meshes
                // because the 3rd mesh of this stupid model is a rectangle which is kind of annoying
                int i = 0;
                foreach (ModelMeshPart meshpart in mesh.MeshParts)
                {
                    i++;
                    if (i == 3)
                        break;
                    effect.Begin();
                    foreach (EffectPass pass in effect.CurrentTechnique.Passes)
                    {
                        pass.Begin();

                        graphics.GraphicsDevice.DrawIndexedPrimitives(PrimitiveType.TriangleList,
                            meshpart.BaseVertex,
                            meshpart.StreamOffset,
                            meshpart.NumVertices,
                            meshpart.StartIndex,
                            meshpart.PrimitiveCount);
                        pass.End();
                    }
                    effect.End();
                }
                //mesh.Draw();
            }
        }


        base.Draw(gameTime);
    }
}
}
```

Draw only MeshPart[0] and [1]

Use your own Draw call instead of mesh.Draw()

Teapot Example

TeapotBasicEffect
(See Demo in Visual Studio)