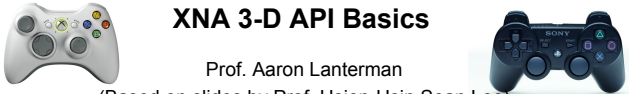


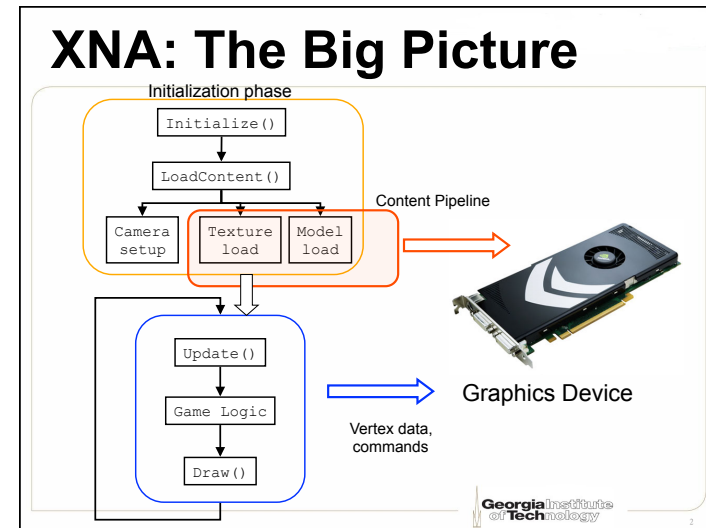
ECE4893A/CS4803MPG:  
**MULTICORE AND GPU**  
 PROGRAMMING  
 FOR VIDEO GAMES

**XNA 3-D API Basics**

Prof. Aaron Lanterman  
 (Based on slides by Prof. Hsien-Hsin Sean Lee)  
 School of Electrical and Computer Engineering  
 Georgia Institute of Technology



Georgia Institute of Technology



## Vectors in XNA

- Various types

- Vector2 vec2 = new Vector2(100, 100);
- Vector3 vec3 = new Vector3(10, 10, 10);
- Vector4 vec4 = new Vector4(1, 1, 10, 2);
- Vector4 vec4 = new Vector4(  
     new Vector2(1,1),10,2);

## Standard vector types

- Vector4 vec4 = Vector4.One;  
   vec4 = (1,1,1,1)
- Vector4 vec4 = Vector4.UnitX;  
   vec4 = (1,0,0,0)
- Vector4 vec4 = Vector4.Zero  
   vec4 = (0,0,0,0)

## Vector member functions

- `Vector4 vec4 = new Vector4(1, 1, 10, 2);`
- `vec4.X = vec4.Y + vec4.Z;`
- Result: `vec4 = (11,1,10,2)`

## Adding and subtracting vectors

- `Vector3.Add(a,b,c); // a=b+c`
- `a = Vector3.Add(b,c); // a=b+c`
- `a = b + c;` (look up “`op_Addition`” method)
- `Vector3.Subtract(a,b,c); // a=b-c`
- `a = Vector3.Subtract(b,c); // a=b-c`
- `a = b - c;` (look up “`op_Subtraction`” method)
- In general, there may be performance reasons to choose one over another or to “DIY”

## Multiplying & dividing vectors

- `c = a*b; c=a/b;`
- Works elementwise for vectors
- Work as you’d expect for a scalar and a vector
- Various corresponding explicit method calls

## Common vector functions


- `float len = v3.Length();`
- `float len = Vector3.Length(v)`
- `float dp = Vector3.Dot(va,vb)`
- `Vector3 cp = Vector3.Cross(va,vb)`
- `Vector3.Cross(cp,va,vb)`
- `Vector3 lerpv = Vector3.Lerp(va,vb,afloat)`
- `Vector3 dist = Vector3.Distance(va,vb)`
- Check out <http://msdn.microsoft.com> for more

2nd  
vector's  
weight

## Normalizing vectors

- `Vector3 nv = Vector3.Normalize(va)`
- `Vector3.Normalize(nv,va)`
- `va.Normalize();` // “destructive” call
- Can normalize other vector types too, but note you’ll most likely want to normalize 3-D vectors (usually for lighting calculations) and not 4-D vectors

VECTOR Example



Vectors  
 (See Demo in Visual Studio)


Georgia Institute of Technology

## Matrices (4x4) in XNA

- `Matrix Rmat44 =`  
`new Matrix(M11,M12,M13,M14,`  
`M21,M22,M23,M24,`  
`M31,M32,M33,M34,`  
`M41,M42,M43,M44);`
- Most often use one of the special matrix initialization routines
  - See if what you need is already there first
- `Rmat44.M13 = Rmat44.M43 + 3;`

## Common matrix functions

- `Vector4 Rmat14= Vector4.Transform(vec4, mat);`  
`// [1x4]*[4x4]`
- `Matrix Rmat44 = Matrix.Multiply(mat1, mat2);`  
`// [4x4]*[4x4]`
- Overloaded with operator `*`, thus `Rmat44=M1*M2;`
- `Matrix inv = Matrix.Invert(mat);` // find its inverse
- `Matrix tr = Matrix.Transpose(mat);` // transpose
- `Matrix im = Matrix.Identity;` // return an identity matrix



MATRIX Example

SimpleMatrix  
(See Demo in Visual Studio)

Georgia Institute of Technology

## Planes

- Plane plane1 = new Plane(vec3, d)  
vec3 describes a normal vector  
d = plane's distance from the origin
- A useful plane function
  - Plane.Intersects(boundingBox)
  - Many others, check msdn

Georgia Institute of Technology

## Shaders in XNA

- You always need an “effect file” (.fx)
- An effect file is a shader code consisting of
  - Vertex Shader
  - Pixel Shader
- Need to communicate variables between your C# code and the shader code, e.g., matrices
- Even rendering a simple triangle requires an effect file
  - Use XNA's [BasicEffect](#) to “fake” a classic DirectX9 style fixed-function pipeline

Georgia Institute of Technology

## Specifying an effect (1)

- Either use a [BasicEffect](#) or specify a provided .fx file

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

    private BasicEffect effect;
```

Georgia Institute of Technology

## Specifying an effect (2)

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

An effect file called effects.fx has been dropped into the Content folder

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

## Specifying an effect (3)

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

```
#if LEE_EFFECT
    effect.CurrentTechnique = effect.Techniques["NoTechniques"];
#endif

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

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

## BasicEffect before XNA 4.0

- 12 Vertex Shaders
  - Lighting: none, per vertex, per pixel
  - Vertex color: off, on
  - Texture: off, on
- 4 Pixel Shaders
  - Per pixel lighting: off, on
  - Texture: off, on

## BasicEffect in XNA 4.0

- 20 Vertex Shaders
  - Lighting: none, 1 vertex light, 3 vertex lights, 3 pixel lights
  - Vertex color: off, on
  - Texture: off, on
  - Fog: off, on (*fog=off only for the versions that do not include lighting*)
- 10 Pixel Shaders
  - Lighting: none, per vertex, per pixel
  - Texture: off, on
  - Fog: off, on (*fog=off only for the versions that do not include per pixel lighting*)

## BasicEffect handles 3 directional lights

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

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



31

## BasicEffect material properties

- Colors range 0 to 1:
  - DiffuseColor
  - EmissiveColor
  - SpecularColor
  - SpecularPower
  - Alpha

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



32

## BasicEffect fog properties

- FogEnabled
  - 0 to disable, 1 to enable
- FogStart
- FogEnd
- FogColor

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



33

## Set up BasicEffect

```
effect = new BasicEffect(graphics.GraphicsDevice);

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.DirectionallLight0.Enabled = true;
effect.DirectionallLight0.DiffuseColor = Vector3.One;
effect.DirectionallLight0.Direction =
    Vector3.Normalize(new Vector3(1.0f, -1.0f, -1.0f));
effect.DirectionallLight0.SpecularColor = Vector3.One;
```



34

## VertexBuffer

- Contains a list of vertices
  - 4-D colors
  - 3-D positions
  - 2-D texture coordinates
  - 3-D normals

## VertexBuffer structures

- VertexPositionColor
- VertexPositionColorTexture
- VertexPositionNormalTexture
- VertexPositionTexture
- Can make your own if you really need to

## PrimitiveType enumeration

Member name	Description
<b>TriangleList</b>	The data is ordered as a sequence of triangles; each triangle is described by three new vertices. Back-face culling is affected by the current winding-order render state.
<b>TriangleStrip</b>	The data is ordered as a sequence of triangles; each triangle is described by two new vertices and one vertex from the previous triangle. The back-face culling flag is flipped automatically on even-numbered triangles.
<b>LineList</b>	The data is ordered as a sequence of line segments; each line segment is described by two new vertices. The count may be any positive integer.
<b>LineStrip</b>	The data is ordered as a sequence of line segments; each line segment is described by one new vertex and the last vertex from the previous line segment. The count may be any positive integer.

<http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.graphics.primitivetype.aspx>

## Set Up Vertex Buffer

```
private VertexBuffer VBuffer; Create a container Vertex Buffer for allocating vertices
```


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

    vertex[0] = new VertexPositionColor(new Vector3(-0.9f, -0.5f, 0.5f),
                                         Color.Red);
    vertex[1] = new VertexPositionColor(new Vector3(0.0f, 0.5f, 0.5f),
                                         Color.Green);
    vertex[2] = new VertexPositionColor(new Vector3(0.5f, -0.5f, 0.5f),
                                         Color.Blue);

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

    VBuffer.SetData(vertex);
}
```

### DrawPrimitives Example



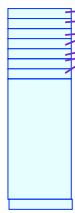
FirstTriangle  
 (See Demo in Visual Studio)

Georgia Institute of Technology

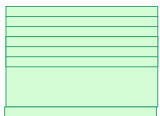
## Indexed vertex buffers

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

Index buffer



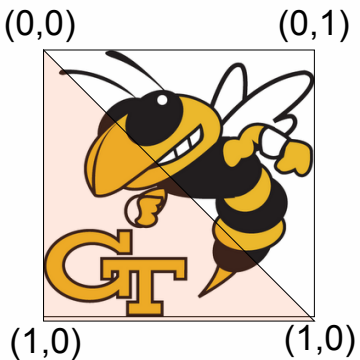
Vertex buffer



(Purple arrows show indices from the index buffer pointing to vertices in the vertex buffer.)

Georgia Institute of Technology

## Let's make a Buzz cube (1)



Georgia Institute of Technology

## Let's make a Buzz cube (2)

```

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));
    
```

Define vertices of a Cube  
 VertexPositionNormalTexture

Texture Coordinates

Georgia Institute of Technology



### A more complex example (3)

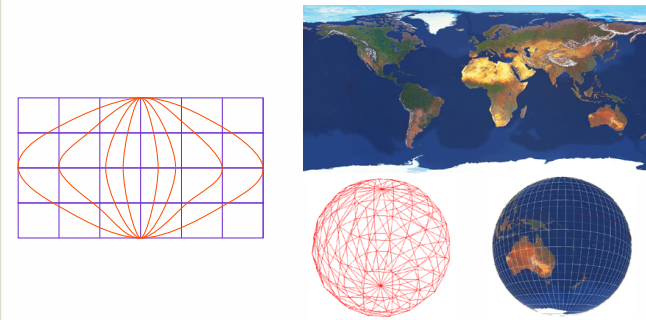
```
triangleListIndices = new short[36] {
    0, 1, 2,
    2, 3, 0,
    4, 5, 6,
    4, 6, 7,
    8, 11, 10,
    8, 10, 9,
    12, 13, 14,
    15, 12, 14,
    16, 19, 17,
    19, 18, 17,
    20, 21, 22,
    23, 20, 22};
```

```
foreach (EffectPass pass in effect.CurrentTechnique.Passes)
{
    pass.Apply();
    graphics.GraphicsDevice.DrawUserIndexedPrimitives
        (PrimitiveType.TriangleList, vertex, 0, 24,
         triangleListIndices, 0, 12);
}
```

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

### Classic transformation matrices

- 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;
```

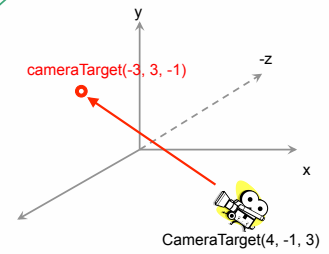
Be cautious on operation ordering

## Create view transformation matrix

- View Transformation

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

Usually (0, 1, 0) → the world is upward  
Use Vector3.Up

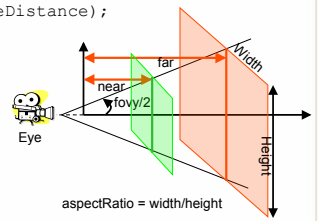


## Create perspective transformation matrix

- Build a "look-at matrix"

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

- Can use `MathHelper.ToRadians(degree)` for fieldOfView



## DrawUserIndexedPrimitives Example



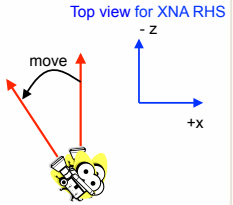
DrawIndexedCube  
(See Demo in Visual Studio)

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

## Rotate camera (looking around)

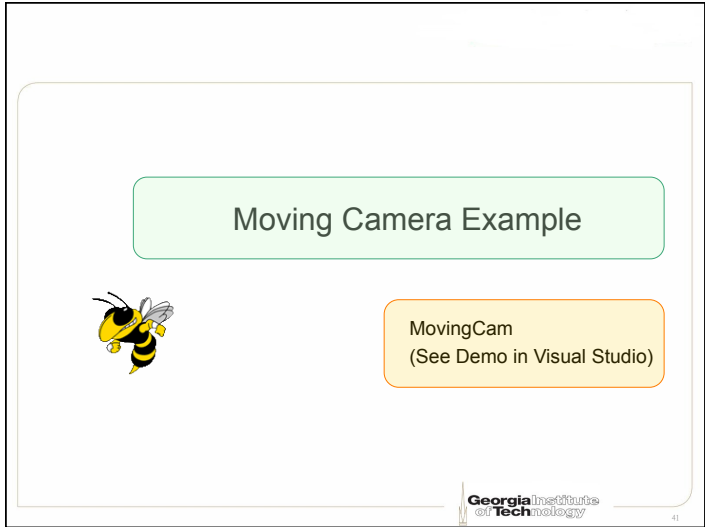
```
protected void CamMove()
{
    KeyboardState keyboard = Keyboard.GetState();
    if (keyboard.IsKeyDown(Keys.Left))
    {
        move -= 0.01f;
    }
    else if (keyboard.IsKeyDown(Keys.Right))
    {
        move += 0.01f;
    }

    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)

Georgia Institute of Technology

41