XNA Drawing

305890
Spring 2014
3/11/2014
Kyoung Shin Park

Drawing

- Vertex Buffer & Index Buffer
  - Creating vertex buffer & index buffer
  - Accessing vertex & index buffer memory
  - Getting vertex buffer & index buffer information
- Render State
- Drawing Preparations
  - Vertex buffer drawing
  - Vertex buffer& index buffer drawing
  - Example
- Geometry Object

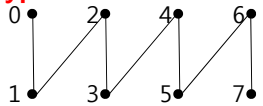
Vertex Buffer / Index Buffer

- Vertex buffer & Index buffer
  - Vertex buffer is simply a chunk of contiguous memory that contains vertex data
  - Index buffer is a chunk of contiguous memory that contains index data
- Creating a vertex buffer
  - This example creates a static vertex buffer that has enough memory to hold **8 vertices of Vertex type**:

```
VertexPositionColor[] vertices = new VertexPositionColor[8];
vertices[0] = new VertexPositionColor(new Vector3(x, y, z), Color.White);
//...
VertexBuffer vertexBuffer = new VertexBuffer(GraphicsDevice,
                                    vertexDeclaration, 8, BufferUsage.None);
vertexBuffer.SetData<VertexPositionColor>(vertices);
```

Vertex Buffer / Index Buffer

- Creating an index buffer
  - This example shows how to create a dynamic index buffer that has enough memory to hold **14 short-type indices**:




```
short[] lineIndices = new short[14];
private void InitializeLineList()
{
    for (int i=0; i<7; i++)
    {
        lineIndices[i * 2] = (short)(i);
        lineIndices[(i * 2) + 1] = (short)(i + 1);
    }
}
```

```
0 1 // line 1
1 2 // line 2
2 3 // line 3
3 4 // line 4
4 5 // line 5
5 6 // line 6
6 7 // line 7
```

```
GraphicsDevice.DrawUserIndexedPrimitives<VertexPositionColor>(
    PrimitiveType.LineList, pointList, 0, 8, lineIndices, 0, 7);
```

Render State

- Render state
  - "SetRenderState" is used to specify rendering states other than default value
  - Enum of many state variables about 100

```
// to draw wireframe mode rendering
RasterizerState rasterizerState = new RasterizerState();
rasterizerState.FillMode = FillMode.Wireframe;

// to draw solid fill mode rendering
rasterizerState.FillMode = FillMode.Solid;

// to set culling
rasterizerState.CullMode = CullMode.None;
```

Vertex Declaration

- Vertex Declarations
  - We need to create a vertex declaration to describe the format of the vertex we are using.

```
// POSITION/COLOR vertex
vertexDeclaration = new VertexDeclaration(new VertexElement[]
    {
    new VertexElement(0, VertexElementFormat.Vector3,
    VertexElementUsage.Position, 0),
    new VertexElement(12, VertexElementFormat.Color,
    VertexElementUsage.Color, 0)
    }
);
```

Vertex Declaration

- Vertex Declarations

```
// POSITION/NORMAL/TEXTURE vertex
vertexDeclaration = new VertexDeclaration(new VertexElement[]
    {
    new VertexElement(0, VertexElementFormat.Vector3,
    VertexElementUsage.Position, 0),
    new VertexElement(12, VertexElementFormat.Vector3,
    VertexElementUsage.Normal, 0),
    new VertexElement(24, VertexElementFormat.Vector2,
    VertexElementUsage.TextureCoordinate, 0)
    }
);
```

Drawing

- GraphicsDevice.DrawPrimitives
  - http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.graphics.graphicsdevice.drawprimitives.aspx
- GraphicsDevice.DrawIndexedPrimitives
  - http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.graphics.graphicsdevice.drawindexedprimitives.aspx
- GraphicsDevice.DrawUserPrimitives
  - http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.graphics.graphicsdevice.drawuserprimitives.aspx
- GraphicsDevice.DrawUserIndexPrimitives
  - http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.graphics.graphicsdevice.drawuserindexedprimitives.aspx

Vertex Buffer Drawing

- DrawPrimitives
  - renders non-indexed geometric primitives of the specified type from the current set of data input streams

```
GraphicsDevice.DrawPrimitives<T>(
    PrimitiveType primitiveType,        // primitive type
    int startVertex,                    // index of the first vertex
    int primitiveCount          // number of primitives to draw
);

// draw 4 triangles
GraphicsDevice.SetVertexBuffer(vertexBuffer);
GraphicsDevice.DrawPrimitives(
                PrimitiveType.TriangleList, 0, 4);
```

Vertex/Index Buffer Drawing

- DrawIndexedPrimitives

```
GraphicsDevice.DrawIndexedPrimitives<T>(
    PrimitiveType primitiveType,        // primitive type
    int baseVertex,     // offset to add each vertex index
    int minVerticeIndex,// min vertex index of vertices
    int numVertices,    // number of vertices
    int startIndex,     // location in the index array
    int primitiveCount // number of primitives to draw
);
// draw a geometry consisting of 12 triangles and 8 vertices
GraphicsDevice.SetVertexBuffer(vertexBuffer);
GraphicsDevice.Indices = indexBuffer;
GraphicsDevice.DrawIndexedPrimitives(
                PrimitiveType.TriangleList, 0, 0, 8, 0, 12);
```

Vertex Buffer Drawing

- DrawUserPrimitives
  - This method is used to draw primitives that do not use index

```
GraphicsDevice.DrawUserPrimitives<T>(
    PrimitiveType primitiveType,        // primitive type
    T[] vertexData,                     // vertex data
    int startVertex,                    // index to an element in the vertex
                                        // buffer for starting point
    int primitiveCount          // number of primitives to draw
);

// draw 4 triangles
GraphicsDevice.DrawUserPrimitives<VertexPositionColor>(
                PrimitiveType.TriangleList, vertices, 0, 4);
```

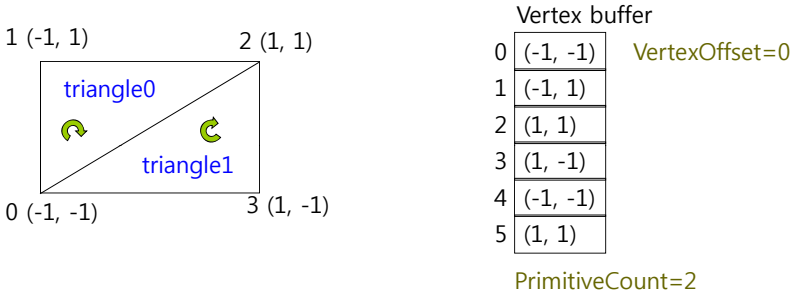
Vertex/Index Buffer Drawing

- DrawUserIndexedPrimitives

```
GraphicsDevice.DrawUserIndexedPrimitives<T>(
    PrimitiveType primitiveType,        // primitive type
    T[] vertexData,     // vertex data
    int vertexOffset,   // offset (in bytes) from the beginning of the vertex
                        // buffer to the first vertex to draw
    int numVertices,    // number of vertices to draw
    short[] indexData,  // index data
    int indexOffset,    // offset (in bytes) from the beginning of the first
                        // index to use
    int primitiveCount // number of primitives to draw
);
// draw a geometry consisting of 12 triangles and 8 vertices
GraphicsDevice.DrawUserIndexedPrimitives<VertexPositionColor>(
                PrimitiveType.TriangleList, vertices, 0, 8, indices, 0, 12);
```

Drawing Example

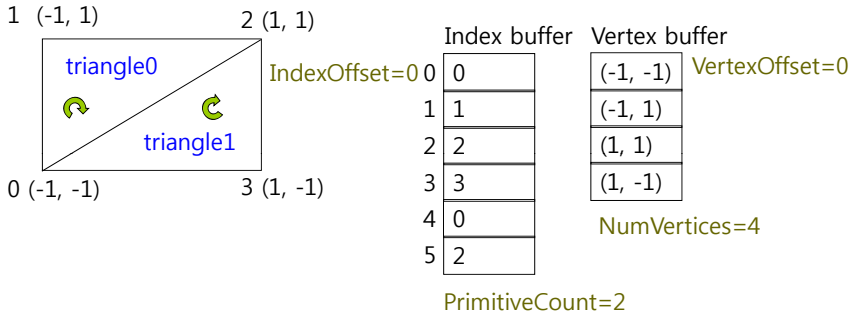
- Draw 2 triangles using DrawUserPrimitives




```
GraphicsDevice.DrawUserPrimitives<VertexPositionColor>(
        PrimitiveType.TriangleList, vertices, 0, 2);
```

Drawing Example

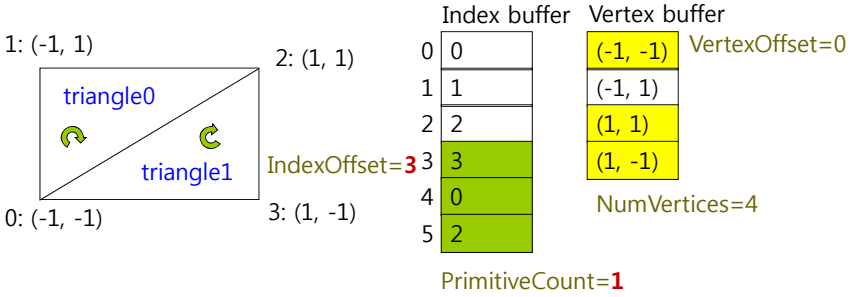
- Draw 2 triangles using DrawUserIndexedPrimitives




```
GraphicsDevice.DrawUserIndexedPrimitives<VertexPositionColor>(
        PrimitiveType.TriangleList, vertices, 0, 4, indices, 0, 2);
```

Drawing Example

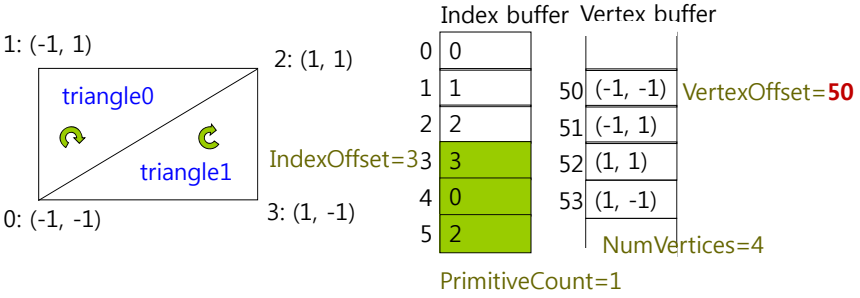
- Draw 1 triangle (i.e., 2nd one) specifying IndexOffset in DrawUserIndexedPrimitives




```
GraphicsDevice.DrawUserIndexedPrimitives<VertexPositionColor>(
        PrimitiveType.TriangleList, vertices, 0, 4, indices, 3, 1);
```

Drawing Example

- Draw 1 triangle specifying VertexOffset in DrawUserIndexedPrimitives




```
GraphicsDevice.DrawUserIndexedPrimitives<VertexPositionColor>(
        PrimitiveType.TriangleList, vertices, 50, 4, indices, 3, 1);
```

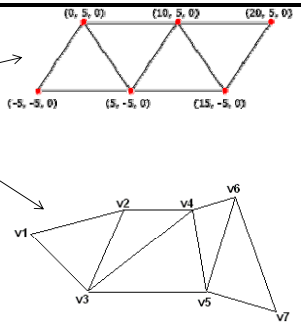
BeginScene / EndScene

- Drawing methods must always be called inside effect Begin and End pair.

```
foreach(EffectPass pass in effect.CurrentTechnique.Passes)
{
    pass.Apply();
    ...
    GraphicsDevice.DrawUserPrimitives<VertexPositionColor>( ... );
    ...
}
```

Primitive Types

- XNA4 primitive types
  - **TriangleList = 0**
  - **TriangleStrip = 1**
  - LineList = 2
  - LineStrip = 3

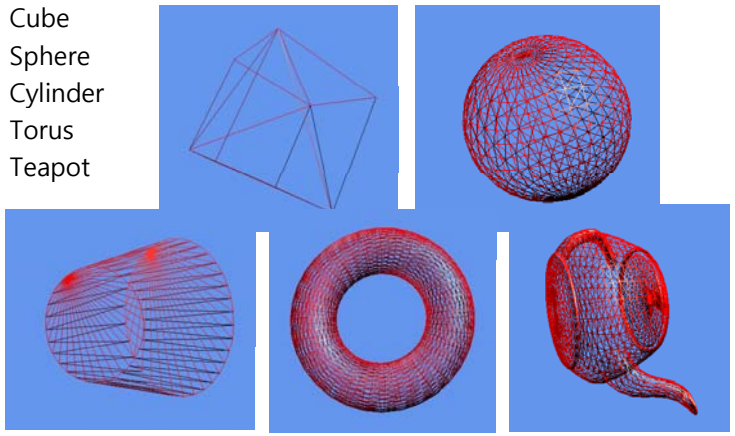


Primitive Types

```
namespace Microsoft.Xna.Framework.Graphics
{
    public enum PrimitiveType {
        TriangleList = 0,
        TriangleStrip = 1,
        LineList = 2,
        LineStrip = 3,
    }
}
```

3D Geometry Object

- XNA basic 3D geometric primitive objects:
  - Cube
  - Sphere
  - Cylinder
  - Torus
  - Teapot



http://create.msdn.com/en-US/education/catalog/sample/primitives_3d