Displaying 3D Models

Game Design Experience
Professor Jim Whitehead
March 4, 2009
• Days until Final Project Due: 12
  ▶ Due Monday, March 16
• 3D modeling homework
  ▶ Assignment text now on web
  ▶ Due Monday, March 9
  ▶ Assignment will involve:
    • Create a simple 3D model (e.g., in Blender)
      – Something slightly (but not by much) more complex than a cube will be fine
    • Make this model show up in XNA
    • Extra credit for making model rotate, applying bitmap textures
    • Goal is to exercise a model import pathway
  ▶ Intended to be a straightforward assignment
Triangles are fine, but models are better

• Today’s 3D games have a large number of objects
  ► Theoretically possible to create these objects by manually writing C# code to create each individual triangle
  ► In practice, this is rarely done.
  ► Far better to use a 3D modeling tool
    • Maya, XSI, 3DS Max, Blender
  ► Allows artists to create objects in world
  ► Allows programmers to focus on behavior of objects
  ► Modeling tools permit much faster creation and editing of models
Several ways to model a 3D object

- **Polygonal modeling**
  - Objects are subdivided into a series of polygons (triangles)
  - Can only approximate curved surfaces
  - Dominant modeling form in computer games and computer graphics due to speed of rendering

- **NURBS**
  - Surfaces are defined by spline curves
  - Curves defined and controlled by control points

- **Splines and patches**
  - Curved lines define surfaces. Between polygons and NURBS

- **Primitives modeling**
  - Objects built up from primitive shapes (balls, cubes, cylinders, etc.)
3D Model formats

• There is a huge number of 3D model formats
  ► No dominant standard
  ► Interchange among models is often lossy
• XNA supports two 3D model formats
  ► .X (DirectX)
  ► .FBX
    • Originally for FilmBox by Kaydara, then Alias, now Autodesk
Models in XNA

• Model
  ▶ Represents some entity
    • A person, a car, or potentially a complex scene with many parts
  ▶ Typically used to represent one logical object (e.g., a person) that has multiple parts (head, arms, legs, etc.)
  ▶ A model contains multiple meshes and bones

• Mesh
  ▶ Represents a single physical object
  ▶ Triangles, textures, shaders, etc.
  ▶ XNA ModelMesh class

• Bone
  ▶ Represents placement of each mesh relative to other meshes
  ▶ A transformation matrix
Models in XNA (cont’d)

• **ModelMesh contains:**
  ► List of ModelMeshPart
  ► List of Effects
  ► Verticies for triangles that comprise this mesh
    • VertexBuffer
  ► Also has a bounding sphere

• **ModelMeshPart**
  ► Represents a set of triangles that share the same materials (e.g., shader, or Effect)
  ► Has indexes into the ModelMesh
  ► Starting index, number of triangles, number of primitives to use from parent ModelMesh’s VertexBuffer
Working with Models in XNA

- Bring model into XNA project inside Visual Studio
- Load model into XNA via Content manager
- Draw model by iterating through all contained meshes
Importing Models into Visual Studio

• Import model into Content directory
  ► Copy .x or .fbx file, along with associated texture images, into Content directory in XNA project
  ► Inside Visual C#, right-click on Content directory
    • Add… Add Existing Item
    • Select .x or .fbx file
  ► Similar process to adding bitmap textures, etc.
Loading Model via Content manager

• Models are loaded via the content manager
  ▶ Model model = Content.Load<Model>("{name of my model without .x or .fbx extension}")

• XNA parses the model file, and fills in vertices, textures, and effects in Model, and ModelMeshes
  ▶ In XNA, this is a robust operation, big time savings
  ▶ In many open source 3D game engines, model import can be a **big** problem
  ▶ At present, typically is not safe to assume model import will work smoothly in a given 3D game engine
  ▶ Need to test your tool chain
Drawing a Model

- Iterate through all of the meshes
  - Iterate through each effect for each mesh
    - Set lighting, camera, and world for each effect
  - Draw each mesh using Draw() method on ModelMesh class

```csharp
foreach (ModelMesh mesh in model.Meshes) {
    foreach (BasicEffect be in mesh.Effects) {
        be.EnableDefaultLighting();
        be.Projection = camera.projection;
        be.View = camera.view;
        be.World = world * mesh.ParentBone.Transform;
    }
    mesh.Draw();
}
```

- Examine example code from Chapter 10 of Learning XNA 3.0
More on Models

• Explanation of parts of XNA models