The term ‘procedural modeling’ refers to the automated (or partially automated) generation of 3D geometry.

Often times, procedures are created to model complex natural objects such as plants, or mountains, but procedural modeling can apply to man-made geometry as well.
Advantages

- General
  - Build more complex data than what can be built by hand
  - Scalability
- Game Specific
  - World editor feature for player
Primitives

- Rendering systems may support a variety of different primitives:
  - Triangles
  - Patches, NURBS, subdivision surfaces…
  - Spheres, ellipsoids, boxes…
- Triangles tend to be the most general, so we will mainly focus on them, but most of the stuff in this lecture can take advantage of different primitives if they are available.
Meshes

- Rather than treat triangles as individual objects, it's nice to collect them into a higher level 'mesh' data structure.
- A mesh is a collection of vertices, triangles, materials, and possibly some other stuff.

Vertices

- For simplicity, we will assume that a 'vertex' contains all of the following information:
  - Position (3D vector)
  - Normal (3D vector)
  - Texture coordinate (2D vector)
  - Color (4D vector (RGBA))
- The mesh contains an array of vertices and individual triangles connect 3 vertices.
- There are various more complex indexing strategies one might use in a mesh, but we will stick to this simple description.
Vertex class

class Vertex {
    Vector3 Position;
    Vector3 Normal;
    Vector2 TexCoord;
    Vector4 Color;
};

Triangles

- A triangle needs to store the index of each of its three vertices
- It also needs some sort of index to its material (texture)

class Triangle {
    int Index[3]; // could also use Vertex*'s
    int Material;
};
Triangle Strips & Fans

- Triangle strips & fans are important for fast real time rendering
- A general purpose mesh library should support them as primitives in addition to individual triangles
- It’s nice to have functions that split strips & fans up into individual triangles, and also to have functions that take collections of triangles and generate optimal strips and/or fans
- For simplicity, however, we will not consider these any further

Materials

- For simplicity, we will just assume that a material contains an integer texture index, but more complex materials could contain lighting information such as diffuse & specular properties, etc.

```cpp
class Material {
    int TexIndex;
};
```
Mesh

- A very basic mesh might look something like:

```cpp
class Mesh {
    void AddVertex(Vector3 &pos, Vector3 &norm, Vector2 &tex, Vector4 &col);
    void AddTriangle(int index0, int index1, int index2, int mtl);
    void AddMaterial(int tex);

    int GetVertexCount(), GetTriangleCount(), GetMaterialCount();
    void RemoveVertex(int v), RemoveTriangle(int t), RemoveMaterial(int m);

    Vertex &GetVertex();
    Triangle &GetTriangle();
    Material &GetMaterial();
};
```

Core Modeling Operations

- Most (all?) complex modeling operations boil down to the following very simple operations:
  - AddVertex(), AddTriangle(), AddMaterial()
  - RemoveVertex(), RemoveTriangle(), RemoveMaterial()

- The ‘add’ functions are very simple, but the ‘remove’ functions may tend to be a little more complex

- In practice, it is often more efficient to tag items for removal and then run a clean up operation once on the entire mesh
Basic Modeling Operations

- Cut, copy, paste
- Triangle manipulation: split, join, subdivide…
- Boolean: intersection, cutting planes, union, subtraction
- All of these can be built upon the core add & remove operations

Mid-Level Modeling Operations

- Some very useful mid-level operations include:
  - Duplication (dupe)
  - Extrude / lathe
- These sound simple at first, but can have many variations and additional features
- They tend to be particularly useful tools used by higher level procedural modeling functions
Duplication

- The basic idea behind the ‘dupe’ operation is to generate several copies of an initial object (or objects) and position them appropriately.
- Objects can be duped along a set of points, or duped across a terrain, or duped along a line, curve, surface… they can be randomly scattered, positioned in patterns… the duped objects can be rotated, scaled, colored, or modified in other ways…

Extrusion

- The basic idea behind an extrusion is to take a (usually 2D) cross section and extrude it along some path in space to generate a 3D surface.
- Simple extrusions can just be along straight lines, but more complex extrusions can use curved paths, angled paths, branching paths, etc…
- A ‘lathe’ is a special case of an extrusion around a circular path (for example, constructing a wine glass by rotating a 2D cross section).
Cross Sections

- A cross section might store various pieces of information needed to build the extruded surface:
  - Geometry
  - Texture mapping info
  - Normals & smoothing info
- The basic extrusion operation essentially generates several copies (dupes) of the cross section and then connects them up with triangles. The individual copies might be translated, rotated, or otherwise modified along the way.

Extrusion Paths

- The cross section is extruded along some sort of path.
- The path can just be a user defined line or curve, or can be more complex shape with branching, etc.
- Fancy features:
  - Miter
  - Interpolate cross section
  - Cross section orientation rules
  - Etc.
Mesh Optimization

- Procedural modeling operations may tend to generate a lot of data, so it is often nice to combine them with mesh optimization (LOD reduction) strategies.
Randomness

Random Number Generation

- Random
- Pseudo-Random
- Quasi-Random
- Noise (Perlin, gradient, sparse-convolution)
- Turbulence
- Fractals
- Fourier synthesis
Fractals

Procedural Texture

- Procedural texture generation is a subset of procedural modeling and shares a lot of common components
- It’s too big of a subject for me to get into here…
Modeling Nature
Trees & Plants

L-Systems
- Lindenmayer (1968)
Water

Skies & Clouds
Modeling Urban Areas
Cities

- “Procedural Modeling of Cities”
- Parish, Müller
- SIGGRAPH, 2001
Cities

“Feature-Based Cellular Texturing for Architectural Models”
Legakis, Dorsey, Gortler
SIGGRAPH, 2001

Masonry

Low-level mesh (geometry)  High-level mesh (features)
Masonry

Masonry