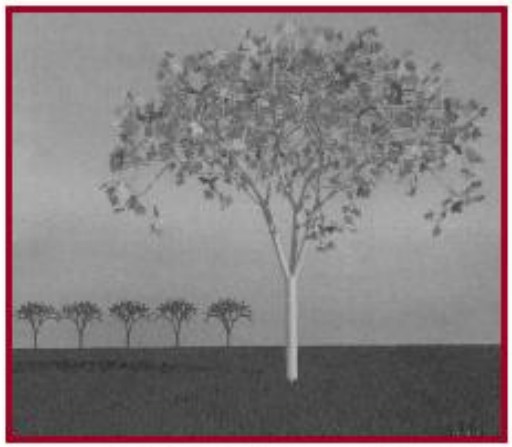
Procedural Modeling Fundamentals

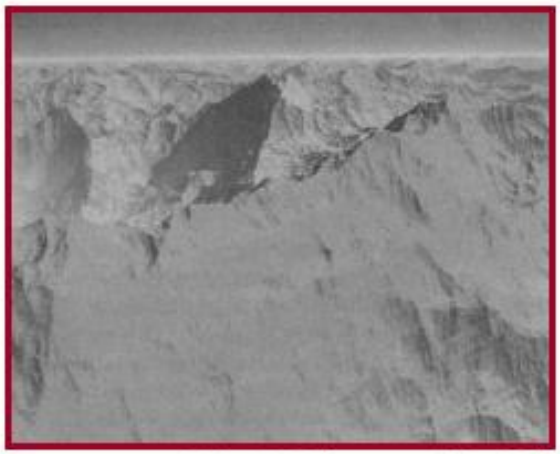
Modeling

- How do we …
  - Represent 3D objects in a computer?
  - Construct such representations quickly and/or automatically with a computer?
  - Manipulate 3D objects with a computer?



H&B Figure 10.79

Fowler



H&B Figure 10.83b

Modeling

- The subject of rendering covers techniques for generating images of complex models, but says little about the creation of those models
- Within computer graphics, the subject of *modeling* is as complex as the subject of rendering
- The demands of modern computer graphics call for the use of extremely complex models containing millions or billions of primitives
- Examples include scenes in modern special effects movies, or industrial models of buildings and vehicles

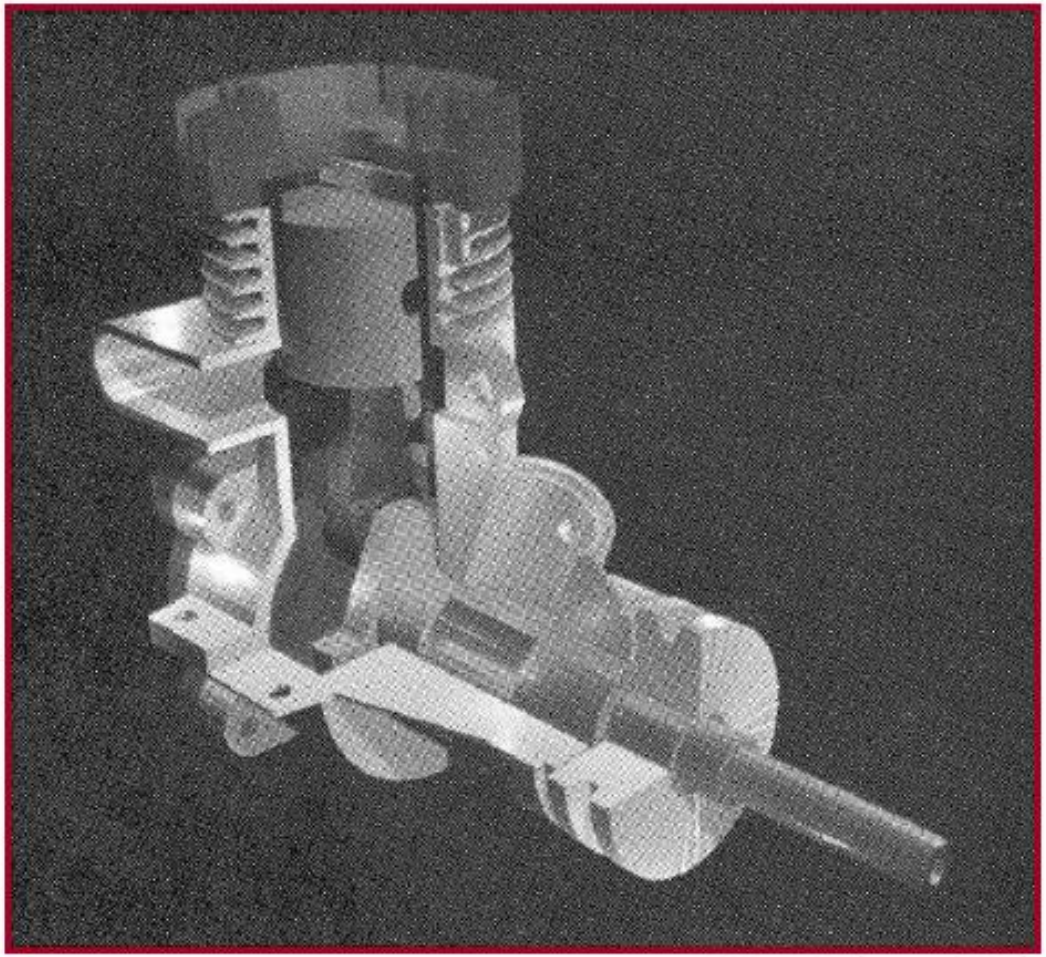
Model Creation

Models are typically built using one or more of the following methods

- *Interactive modeling*
  - Model is constructed by a human using a software modeling tool
- *Procedural modeling*
  - Model constructed by automatic procedure that may make use of randomness for variety
- *Scanning*
  - Model geometry is scanned from a real world example using a laser scanner or similar device
- *Computer vision*
  - Model geometry & material information is scanned from real world example using multiple photographic camera angles (or video sequences)

Interactive Modeling Tools

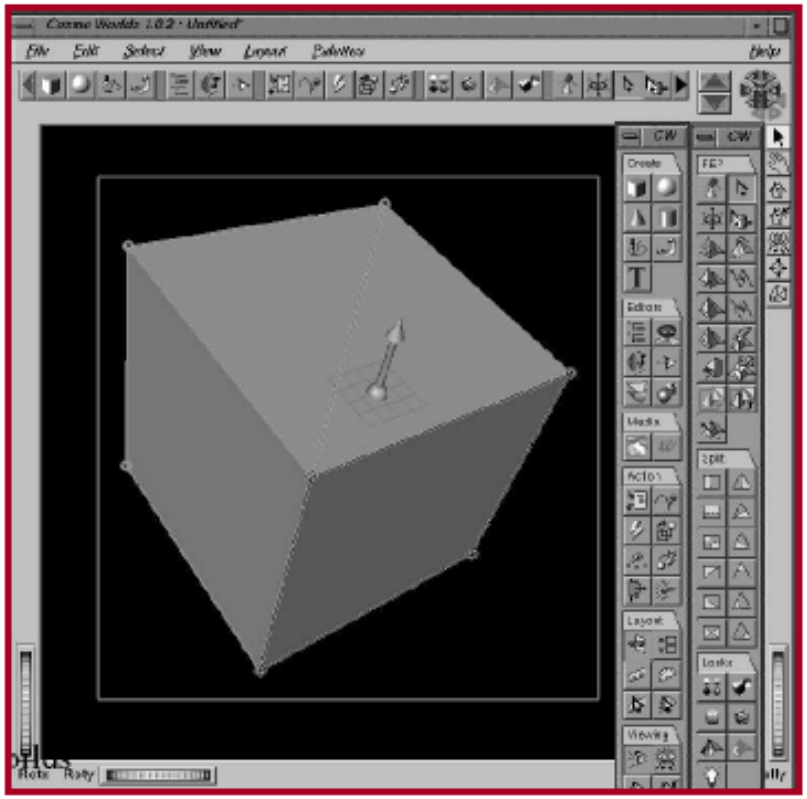
- Example: Mechanical CAD



H&B Figure 9.9

Interactive Modeling Tools

- User constructs objects with drawing program
  - Menu commands, direct manipulation, etc.
  - CSG, parametric surfaces, quadrics, etc.



Cosmoworlds, SGI

Procedural Modeling

Procedural Modeling

- *Procedural modeling* refers to a wide variety of techniques for automatic model creation
- Procedural modeling can be used to create models that are too complex (or tedious) for a person to build
- Common examples include natural objects such as trees, landscape (mountains…), clouds, etc.
- It is also possible to use procedural modeling for man-made objects such as buildings and architecture at the city scale
- Procedural models are often defined by a small set of data that describes the overall properties of the model. For example, a tree might be defined by some branching properties and leaf shapes
- The actual model is constructed by a procedure that often makes use of randomness to add variety. In this way, a single tree pattern can be used to model an entire forest

Model Create / Delete

- The most basic operations are:

```
Vertex *CreateVertex();
void DeleteVertex(int v);

Triangle *CreateTriangle();
void DeleteTriangle(int t);
```

- Just about all higher level modeling functions can be broken down into these basic operations
- All higher level functions go through these interfaces to create and remove data
- These functions need to be fast and reliable
- The ‘delete’ operations can be done in different ways and are not as simple as they might first look….

Procedural Modeling

- Goal:
  - Describe 3D models algorithmically
- Best for models resulting from ...
  - Repeating processes
  - Self-similar processes
  - Random processes
- Advantages:
  - Automatic generation
  - Concise representation
  - Parameterized classes of models

Shape Primitives

- Many real world objects contain basic shapes like spheres, boxes, cylinders, cones, etc.
- Sometimes, complex models can be built entirely from these simple shapes
- Modeling tools should have functions for creating a variety of primitive shapes like these

Copy

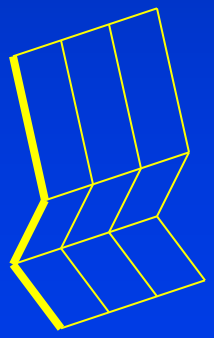
- One of the most basic modeling tools is the simple copy operation
- Models can be built up from multiple copies of simpler shapes
- A copy operation would probably take a source and destination object as well as a matrix as input
- It would add new vertices and triangles to the destination object by transforming the vertices of the source object by the matrix

Duplicate

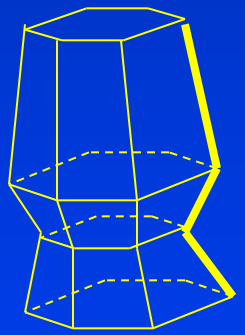
- The *dupe* or *duplicate* operation is a more complex variation on copy
- There are several variations on dupe operations and there really isn't any standard on this stuff
- A common dupe operation might take a source object as well as a group of points as input and generate a copy of the source object at every point
- More complex dupe operations might take several source objects as input and choose a random one to place at the point and might apply additional randomness such as a random rotation or slight variation in the scale
- This can be used to do things such as placing a bunch of trees along the side of a road, for example

Extrude / Lathe

- Many useful shapes can be constructed by extruding or lathing a line (or curve)
- The extrude operation generates a surface by connecting copies of the line that have been placed in a straight line
- The lathe operation works in a similar way, except the copies are rotated around a circle



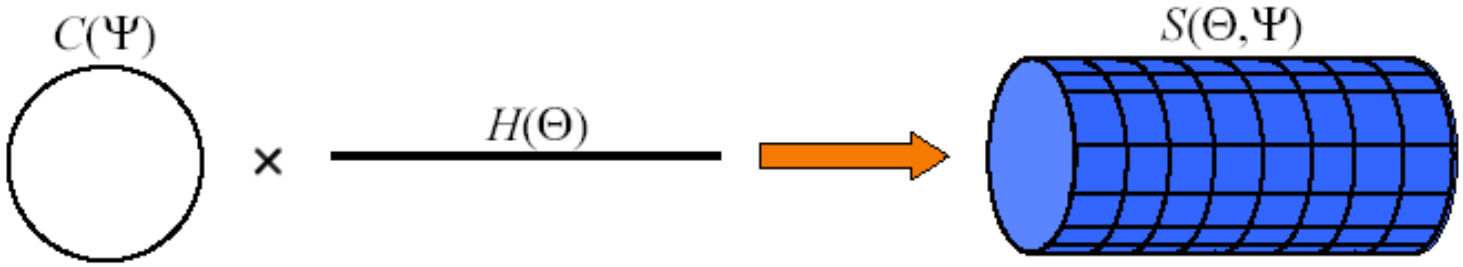
Extrude



Lathe

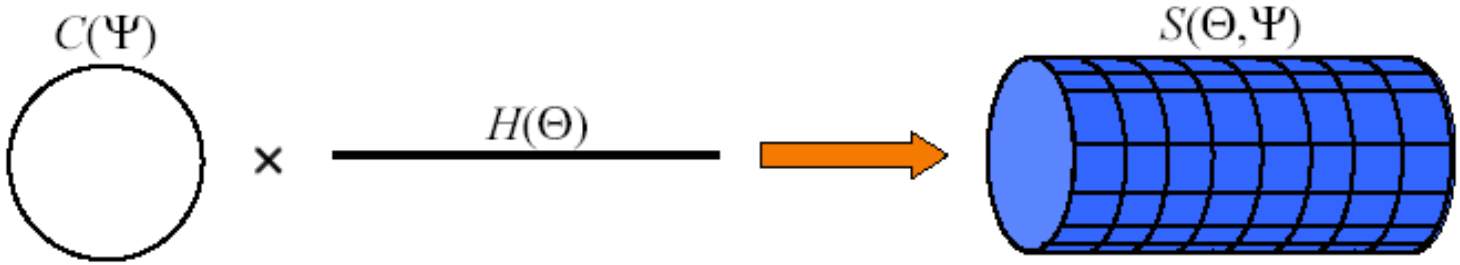
Sweeps

Given a 3D sweep curve $H(\Theta)$ and a 2D generating curve $C(\Psi)$, we define the sweep surface $S(\Theta,\Psi)$ as the sweep of C along H:



Sweeps

Given a 3D sweep curve $H(\Theta)$ and a 2D generating curve $C(\Psi)$, we define the sweep surface $S(\Theta,\Psi)$ as the sweep of C along H:



In this example, the sweep curve is simply used to translate the generating curve:

$$S(\Theta,\Psi) = H(\Theta) + C(\Psi)$$

We can define more complex sweep surfaces.

Path Extrude

- A powerful variation on extrusion is the *path extrude* operation
- With this one, we have one or more lines (or curves) that make up a *cross section* and a second line (or curve) that makes up the *path*
- The path extrusion connects several copies of the cross section along the path that orient to the path as it turns
- This can be used to make a tree trunk, or a freeway overpass (or tunnel), for example
- The cross section could also vary along the path to allow for additional control

Lofting

- There are also a variety of *lofting* tools that can be used to create surfaces out of a set of input lines (or curves)
- For example, various lofting tools can be used to model shapes like boat hulls, airplane wings, and car bodies

Example: Seashells

- Create 3D polygonal surface models of seashells

"Modeling Seashells,"
Deborah Fowler, Hans Meinhardt,
and Przemyslaw Prusinkiewicz,
Computer Graphics (SIGGRAPH 92),
Chicago, Illinois, July, 1992, p 379-387.

Fowler et al. Figure 7

Example: Seashells

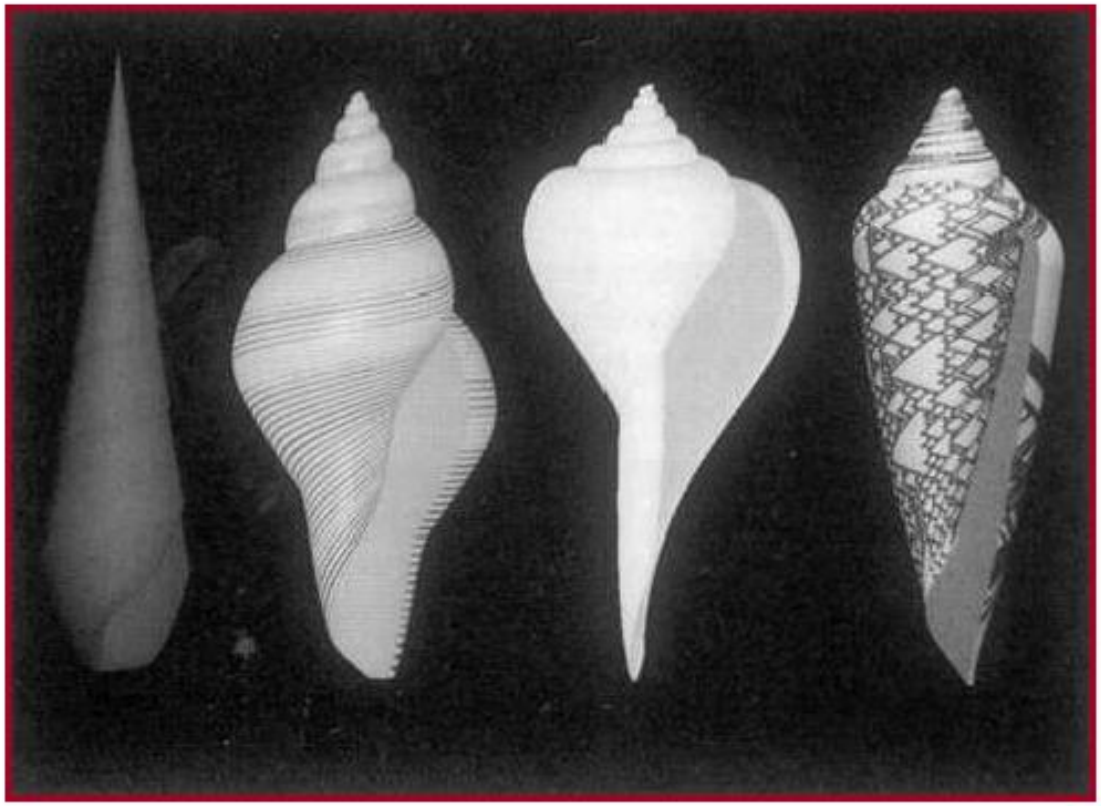
- Generate different shells by varying parameters

Different helico-spirals

Fowler et al. Figure 2

Example: Seashells

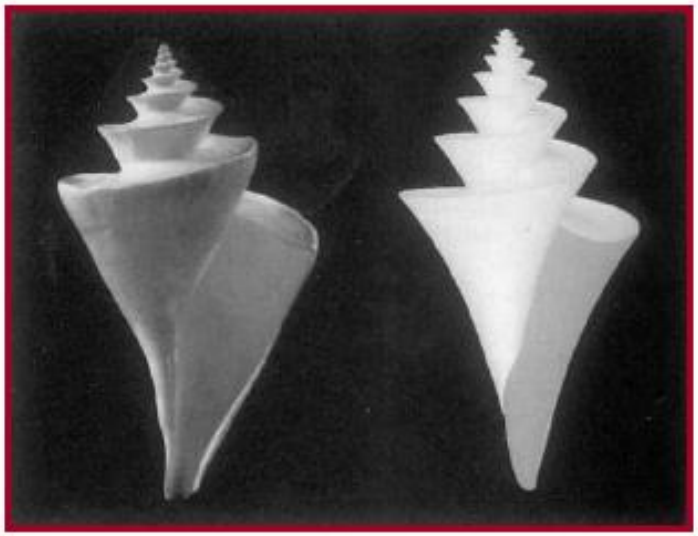
- Generate different shells by varying parameters



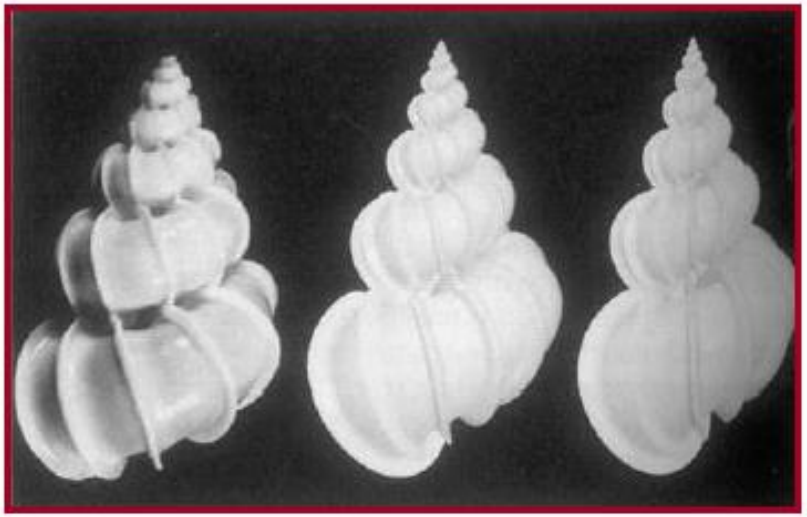
Different generating curves

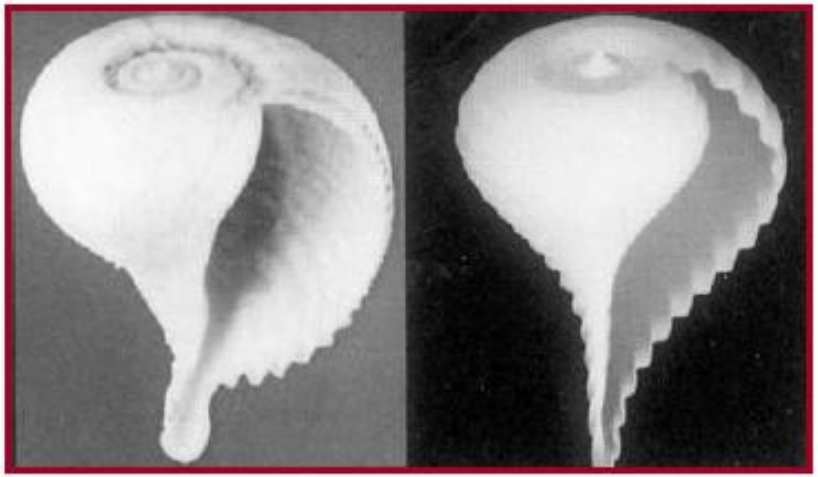
Fowler et al. Figure 3

Example: Seashells



Generate many interesting shells with a simple procedural model!





Fowler et al. Figures 4,5,7

Boolean

- Boolean operations can be used to compute unions, intersections, and subtractions with complex 3D shapes
- Many industrial models are build from Boolean operations

Basic Modeling Operations

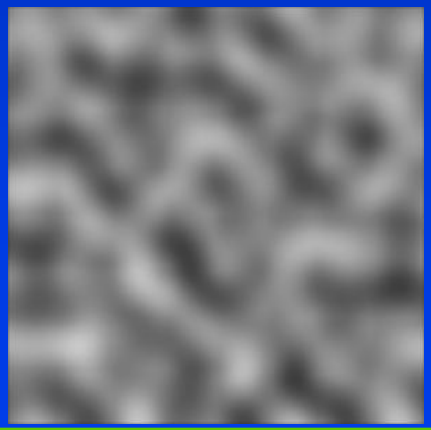
- The modeling operations we have discussed so far make up some of the most common functions found in interactive modeling tools
- They are also the foundation of more automated procedural modeling tools
- These operations have been used for many years and continue to be useful
- One reason for their popularity is that they directly relate to the way that many objects are designed and built in the real world

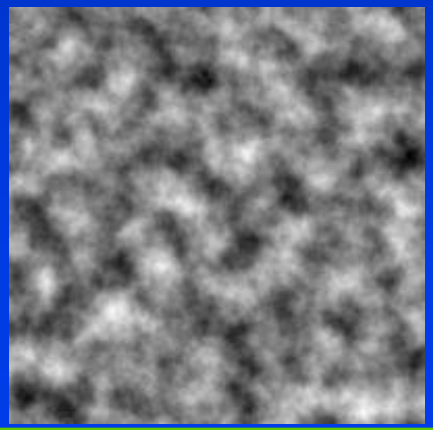
Randomness

- Procedural models often make use of some form of randomness
- A simple random number generator is usually sufficient for many operations
- As computers can not usually generate *true* random numbers, they typically make use of *pseudorandom* number generation algorithms
- A pseudorandom number generator outputs a sequence of (apparently) random numbers based on some initial seed value
- In this sense, the sequence is repeatable, as one can always reset the sequence
- For example, if a procedural model like a tree is built from by making use of several random numbers (maybe hundreds), then the entire tree can be rebuilt by just resetting the seed to its initial value
- If the seed is set to a different value, a different sequence of numbers will be generated, resulting in a slightly different tree

Noise

- Another form of randomness which is sometimes useful for procedural modeling is *noise*
- Noise represents a distribution of randomness over some space (usually 2D or 3D)
- Noise is not entirely random, as two points nearby will have a similar value
- In this way, noise has a frequency associated with it
- By combining noise patterns of different frequencies, one can make more complex *turbulence* patterns





Fractals

- A fractal is a geometric object that is self-similar when viewed at different scales
- For example, the shape of a coastline may appear as a jagged line when we view a map of Long Island. As we zoom in closer and closer, we see that there is more and more detail at finer scales. We always see a jagged line no matter how close we look at the coastline

Fractals

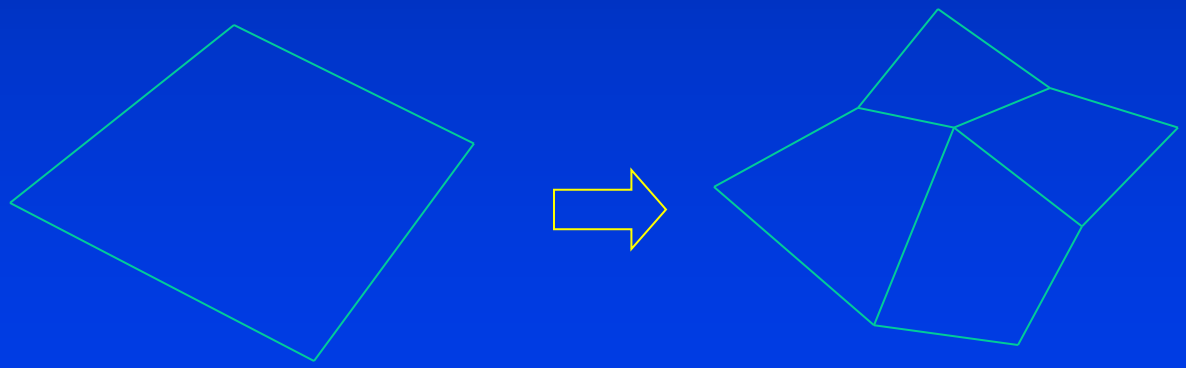
- Fractals can be regular repeated patterns, or can be irregular and incorporate randomness as well
- Random fractals are useful for creating a wide variety of natural shapes such as mountain landscapes
- Even trees can be thought of as a fractal, as the branching patterns are similar when one looks at the main trunk down to the finest branches
- For procedural modeling, we may borrow some fractal concepts, but we rarely deal with true mathematical fractals with infinite detail
- We usually think of fractals as techniques for generating randomness in some limited range of scales

Fractals

- Consider a simple line fractal
- We start with a single line segment and then split it in the middle, randomizing the height of the midpoint by some number in the [-*r*,*r*] range
- We then split each of the new line segments at the middle and randomize them by [-*r*/2,*r*/2]
- This process is repeated some desired number of steps, randomizing by half as much each step

Fractals

- A similar process can be applied to squares in the xy plane
- At each step, an xy square is subdivided into 4 squares, and the z component of each new point is randomized
- By repeating this process recursively, we can generate a mountain landscape



Height Fields

- Landscapes are often constructed as *height fields*
- In a height field, we assume a regular grid in the ground plane (for us, that is the xy plane)
- At each grid point, we store a height (z) value
- In this way, a large terrain can be stored in memory without explicitly storing the x & y coordinates of the vertices or the triangle connection information
- The terrain can be shaped by operations that modify the z coordinates
- In a lot of ways, shaping the terrain is like rendering an image, where the heights of the cells in the height fields can be compared to the pixel colors in an image
- Similar tools can be used to shape the height field to the tools used in rendering, such as the use of triangles or noise patterns

Height Fields

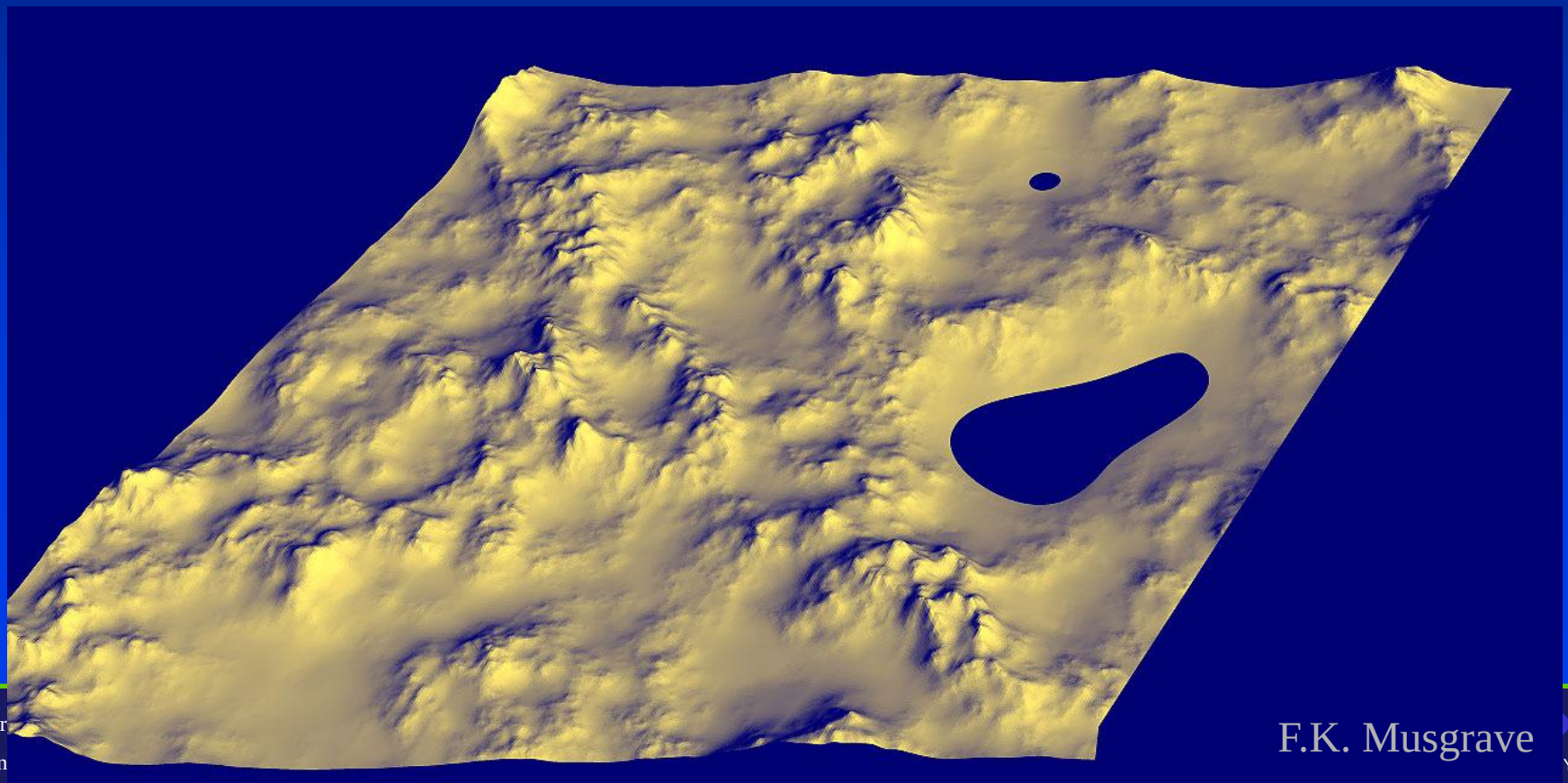
- The height field itself is an efficient data structure for storing the shape of the terrain, but it still must be converted to triangles to render
- We could simply generate a grid of triangles
- However, if we use a grid, we will end up with too many triangles in flat regions and too high of a triangle density off in the distance
- It would be better to perform some sort of adaptive tessellation of the height field, much like the tessellations used in patch rendering and displacement mapping

Quadtree Tessellation

- One way to triangulate height fields adaptively is through the use of a *quadtree*
- The quadtree is a 2D data structure that is usually based on rectangles or squares
- It works in a very similar way as the fractal subdivision we just covered, except it can be used for triangulating height fields (or Bezier patches…)
- We start with single square around our whole terrain
- We perform some sort of analysis on the square and determine if it contains more detail than is adequately represented by a square
- If the detail is insufficient, the square is split into four smaller squares, which are recursively tested
- Ultimately, squares are then split into two triangles

Generating Terrains

- **Use an elevation threshold ($z < z_{water}$)**



F.K. Musgrave

Landscape

- By combining a variety of tools such as fractals, noise patterns, triangle rasterization, and others, one can build up a set of tools for modeling natural terrains (and man made modifications to terrain)
- One can also run simulations of erosion to achieve additional realism
- One can also use real world data of the Earth to model specific regions
- Geographic data exists in many formats, but one of the more useful ones is the *DEM* or *digital elevation map*, which is essentially a height field for a rectangular region of the Earth's surface
- The USGS has DEM files for the entire continental US at 10 meter resolution, and for the entire world at 30 meter resolution, available for free downloading!

Roads

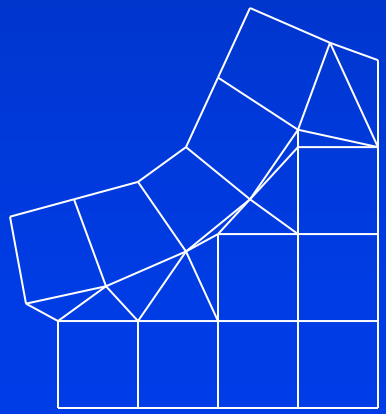
- Roads can be modeled as cross sections that get path extruded along some curve
- The cross section can include lanes, curbs, sidewalks, and center islands
- Intersections require special handling, but can still be generated using a set of procedural techniques
- As with using DEMs for creating height fields, it is possible to find road map data online that contains maps of many key metropolitan areas. Often, the road map data is defined as a graph of connected lines with additional information for each line segment such as the number of lanes and the address range

Roads

- Roads can be placed on height fields by placing the control points of the road curves on the height field
- This will still require some local flattening for the road and the area to the side of the road
- This can be achieved by essentially rendering road triangles onto the height field, where a low detail road is extruded and ‘rendered’ into the cells of the height field to set their heights. Sides of roads can be blended using techniques similar to alpha blending
- These operations can be used to modify the existing shape the height field in a very similar way to how real roads are constructed

Roads

- Ideally, the road surface would be an extrusion and the open terrain would be a height field
- They can be sewn together into a single triangle mesh in a similar way to how trim curves are used in patch tessellation

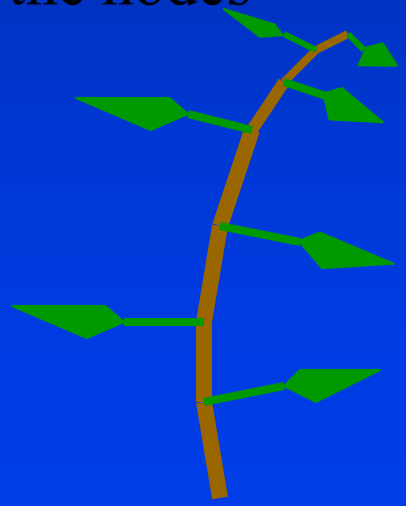


Trees

- Trees are a classic example of complex natural objects that can be procedurally modeled
- There have been numerous research papers published on various aspects of botanical modeling
- One recent paper focused on creating the detailed sub-millimeter scale vein patterns seen on leaf surfaces
- By varying a number of key parameters, one can model a wide variety of plants and trees to any level of detail desired
- Even with about 10 parameters, one can model a wide variety of overall plant shapes, but real plant modeling systems may allow hundreds of parameters as well as the inclusion of custom geometric data to define leaf shapes or branch cross sections….

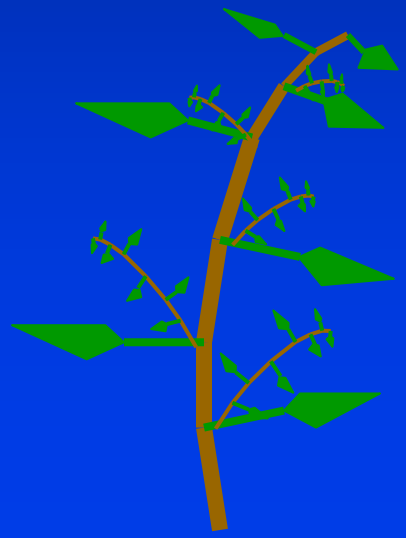
Primary Shoot Growth

- All plants grow according to the same basic pattern, although there is a huge amount of variation within that pattern
- Growth takes place at tips of stems, where new leaves are formed
- Primary growth includes development of these new leaves at relatively regular intervals as well as elongation of the stem between the nodes



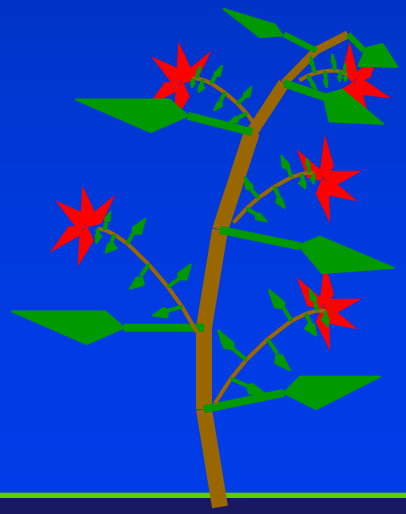
Axillary Growth

- At each leaf node, an axillary bud is formed, which may remain dormant, or may develop into a whole new stem



Flowers

- Flowers will form at the tips of younger stems at certain times of the year, triggered by seasonal properties such as day length or temperature
- Flower petals and other floral components are modified leaves themselves

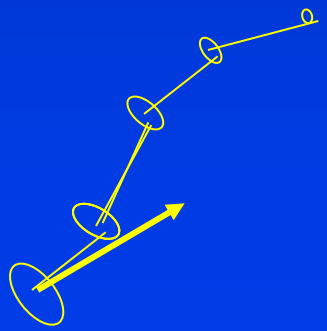


Plant Shapes

- The variety of plant shapes is mainly due to variations in stem shapes, branching properties, and leaf shapes, and these can be broken down into specific properties that can be used to control a procedural plant model
- A simple way to model a plant is to start by thinking of it as a bunch of branches (stems) and leaves

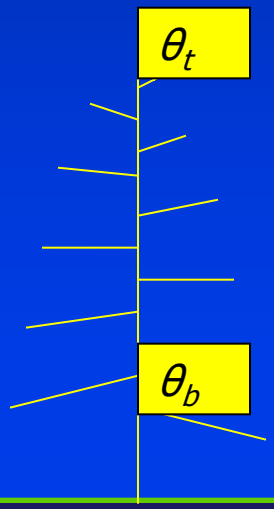
Branches

- We can think of a branch as being a circular extrusion along some path, but it would certainly be possible to use non-circular cross sections as well
- Overall, the radius of the cross section will either remain constant or may taper from being thicker at the base and thinner at the tip. We can simplify this by just specifying a radius at each end and assume a linear interpolation along the branch
- The path of the branch can just be a set of points
- At the simplest, these points could form a straight line, but it wouldn't be hard to add some curvature and randomness
- The path could be randomly created just from a starting point, an initial direction, and a desired length



Branching

- Each branch can spawn off new branches
- The new branches would be placed at points along the original branch with some rules to define their initial direction
- For example, it is nice to allow new branches to start at some percentage along the original branch
- The length of the new branches can be defined by two other percentages, one which describes the length of new branches at the base of the original branch and one that describes the length at the tip
- The branching angle can be described similarly by values at the base and tip



Branching and Leaves

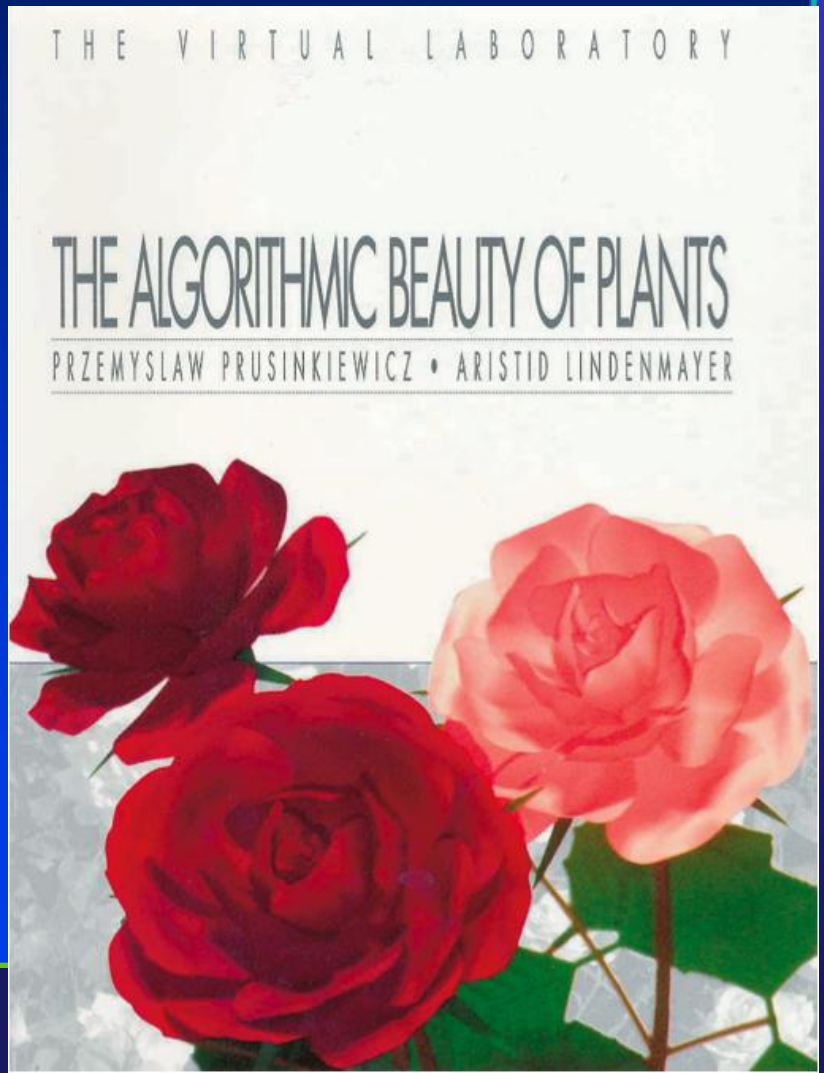
- The branching is usually repeated two or three times so that we get sub-branches on sub-branches on branches
- At that point, we branch one final time, but create leaves instead of new branches
- The leaves can be placed along the branch according to similar rules as sub-branches

L-Systems for Plants

- L-Systems can capture a large array of plant species
- Designing rules for a specific species can be challenging

Algorithmic Botany

- http://algorithmicbotany.org/papers/
- **Free 200pg ebook**
- **Covers many variants of L-Systems, formal derivations, and exhaustive coverage of different plant types**

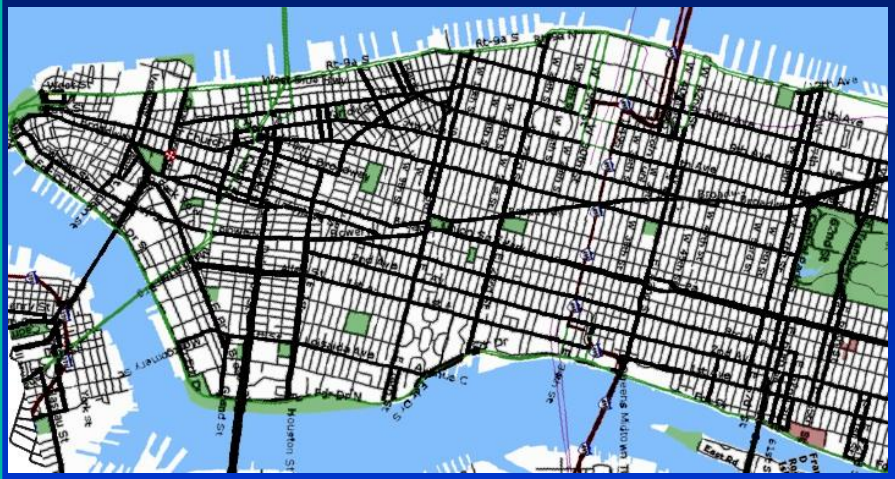


- **Fast procedural foliage is important for real-time applications**
- http://www.speedtree.com/

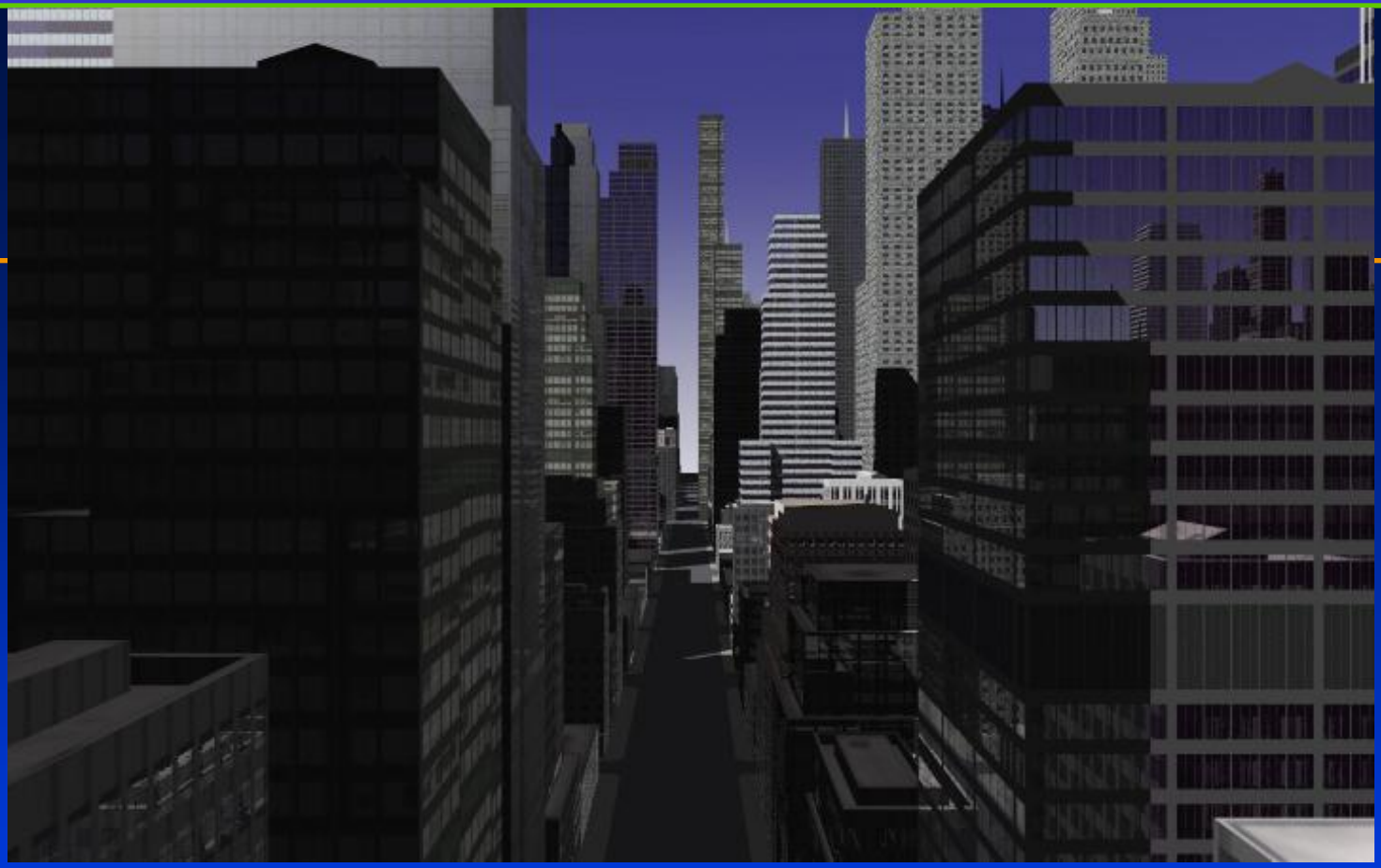
Buildings and Cities

- Buildings and other man-made structures can also be procedurally modeled
- In addition to the overall shapes of buildings, there have been papers for details such as exact brick placement including a variety of patterns
- There have also been research papers on automatically generating city road map layouts based on terrain height fields
- From the road maps, city blocks are then subdivided into lots, which have procedural buildings placed on them
- Details like street lights, trees, etc. can be placed along the roads
- In this way, entire cities can be build automatically
- Cities (and other complex models) can either be generated completely randomly, or as a mix of random and non-random processes
- Additional data exists for cities that describe locations and overall outlines of buildings, placement of power & telephone lines, train tracks, and other data

L-Systems for Cities [Parish01]



- Start with a single street
- Branch & extend w/ parametric L-System
- Parameters of the string are tweaked by goals/constraints
- Goals control street direction, spacing
- Contraints allow for parks, bridges, road loops
- Once we have streets, we can form buildings with another L-System
- Building shapes are represented as CSG operations on simple shapes



- Once we have streets, we can form buildings with another L-System
- Building shapes are represented as CSG operations on simple shapes

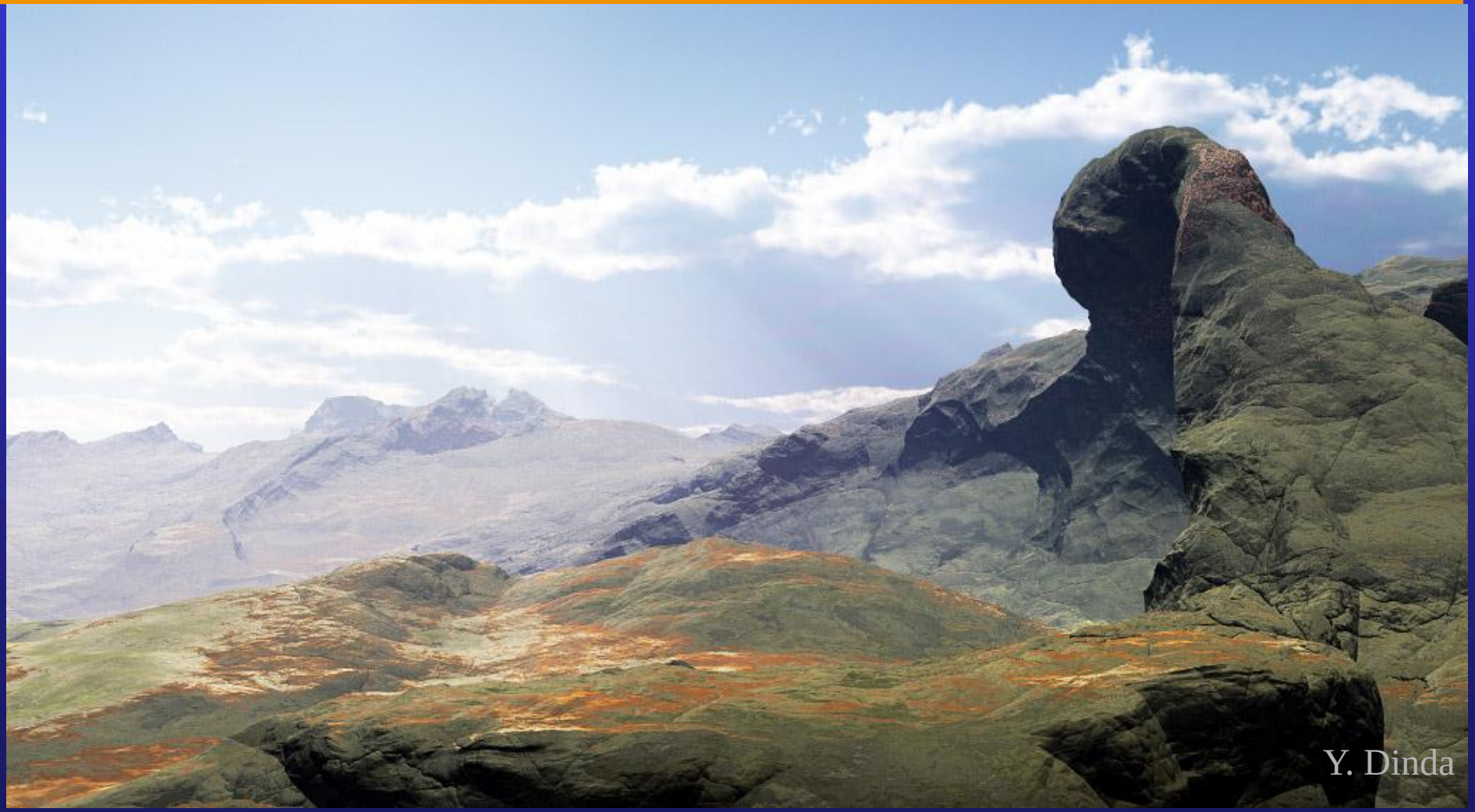
Proper Placement

- There have been various research papers that have addressed the issue of prop placement as a procedural modeling tool
- To place plants, for example, we do not just want to randomly scatter them around
- Models have been designed that take the shape of the terrain into consideration and determine plant locations based on properties such as wetness, light exposure, and other geographic properties
- In addition, simulations can be run that model the changes in the ecosystem over time and allow for different plant groups to spread about the terrain
- Man-made objects can also be automatically placed in terrains. Objects such as street lights, traffic lights, houses, street signs, and more can be placed automatically in a city based on the basic road map and terrain layouts

Procedural Planets

Procedural Planets

Procedural Planets



F.K. Musgrave