

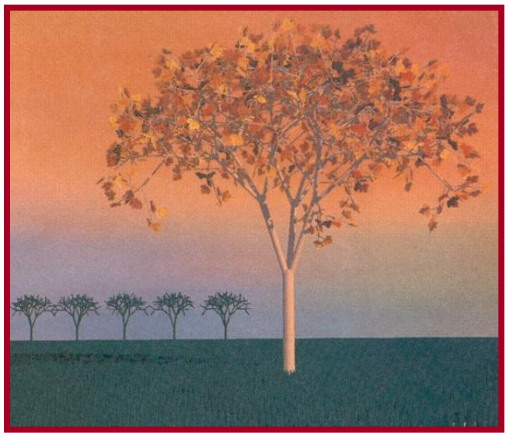
Modeling

Misha Kazhdan

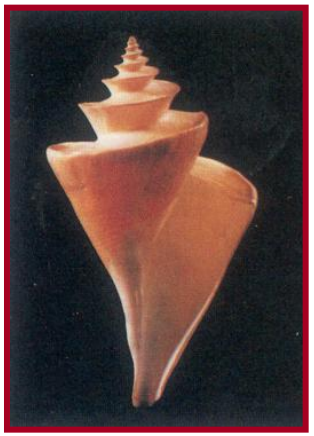
(601.457/657)

Modeling

How do we construct 3D models quickly and/or automatically with a computer?



H&B Figure 10.79



Fowler



H&B Figure 10.83b

Model Construction

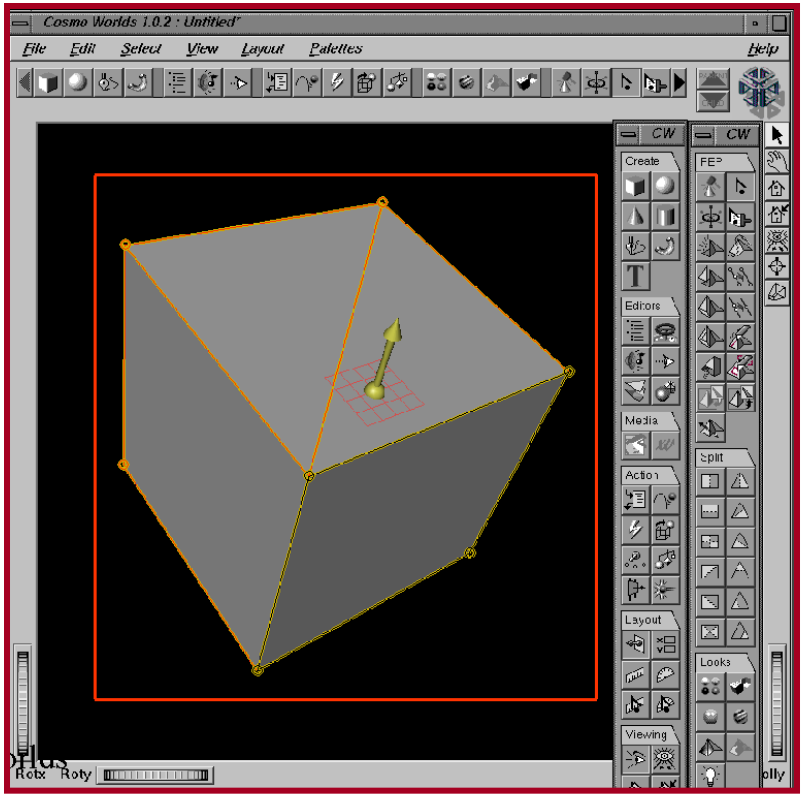


- Interactive modeling tools
  - CAD programs
- Active scanners
  - Light probes, triangulation/LiDAR/CAT/MRI scanners
- Passive scanners
  - Stereo, motion, etc.
- Procedural generation
  - Sweeps, fractals, grammars

Interactive Modeling Tools

User constructs objects with drawing program

- Menu commands, direct manipulation, etc.
- CSG, parametric surfaces, quadrics, etc.



Cosmoworlds, SGI

Model Construction



- Interactive modeling tools
  - CAD programs
- Active scanners
  - Light probes, triangulation/LiDAR/CAT/MRI scanners
- Passive scanners
  - Stereo, motion, etc.
- Procedural generation
  - Sweeps, fractals, grammars

Active Scanners: Touch Scanner

Articulated arm with:

- Angular sensors at the joints
- Touch probe at the end

Angles are recorded on contact, and use to compute the position of the head



Active Scanners: Laser Scanner

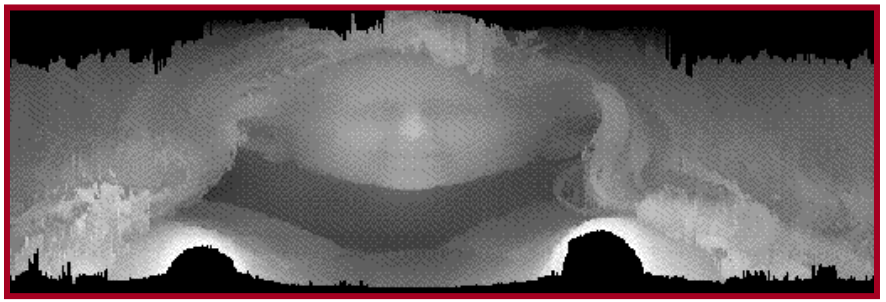
Emits a laser from one position

Images the laser-illuminated surface from another **calibrated** position

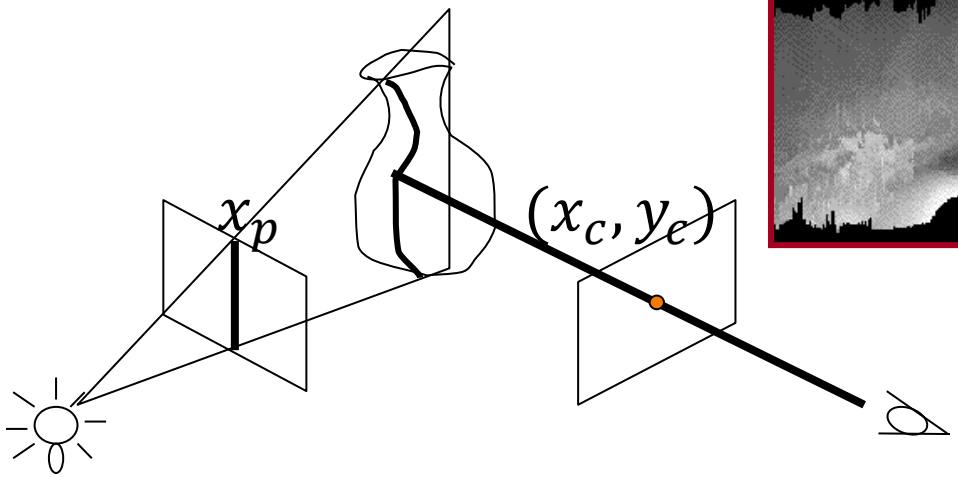
Triangulation used to determine positions.



Color



Depth



Active Scanners: IR Scanner

Emits IR light from one position

Images the IR-illuminated surface from another **calibrated** position

Triangulation used to determine positions.



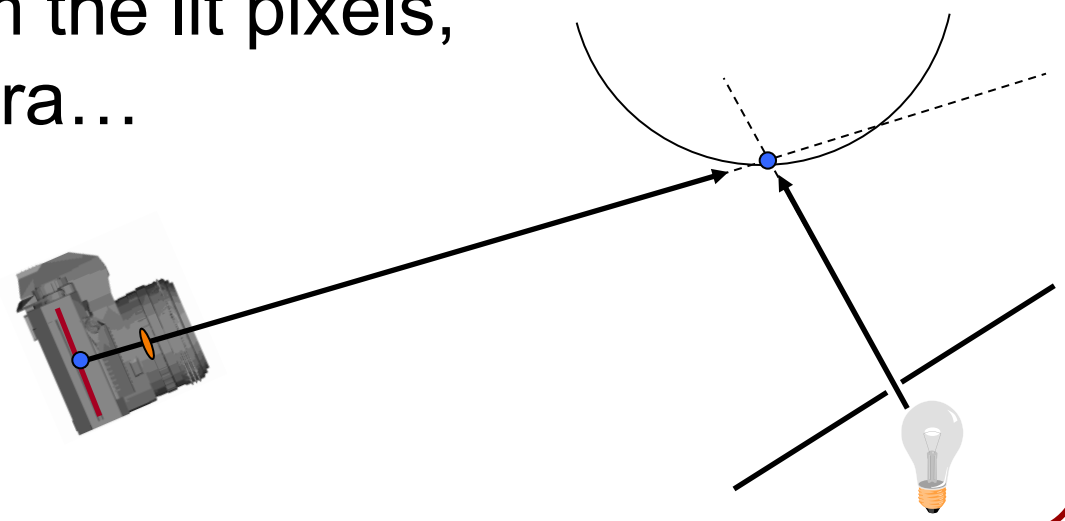


Active Scanners: Triangulation (2D)

To figure out the position of a point we need:

1. The position of the camera
2. The position of the light source

Project the light through a slit onto the surface…
Find where the lit points project
onto the camera…
Consider the ray from the lit pixels,
through the camera…



Active Scanners: Triangulation 2D



To figure out the position of a point we need:

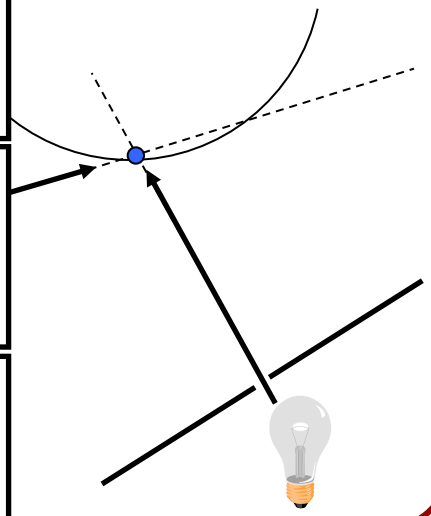
1. The position of the camera
2. The position of the light source

Project the light through a slit onto the surface…
Find where the lit points project
onto the camera…

For the lit point to project onto the pixel, it must lie along the ray.

The lit point is also constrained to lie on the plane from the light source.

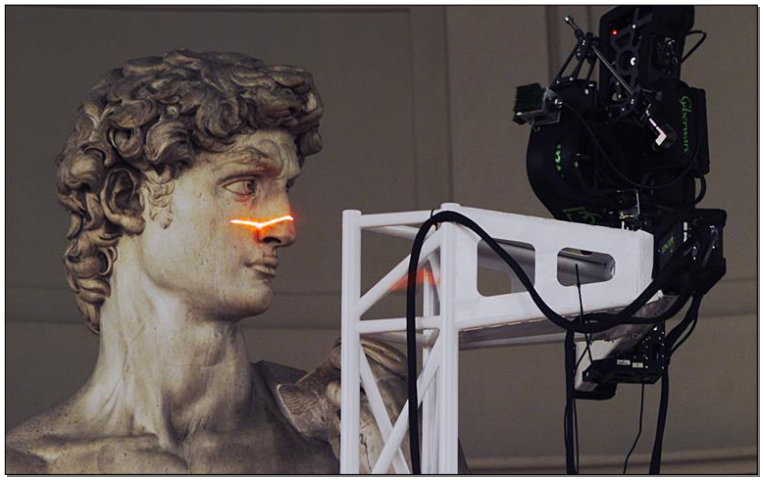
The position of the lit point has to be at the intersection of the ray and the plane.



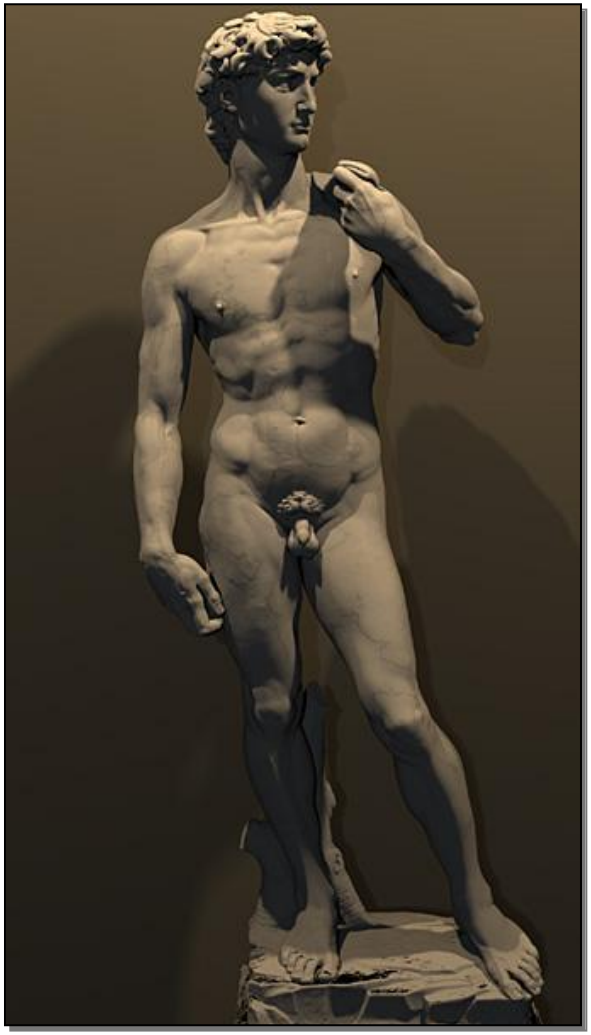
Laser Range Scanning

Example: Digital Michelangelo Project

480 individually aimed scans



Stanford Graphics Laboratory



Active Scanners: LiDAR

Emits laser light

Measures the time of flight – time it takes the light to reflect back to the sensor.

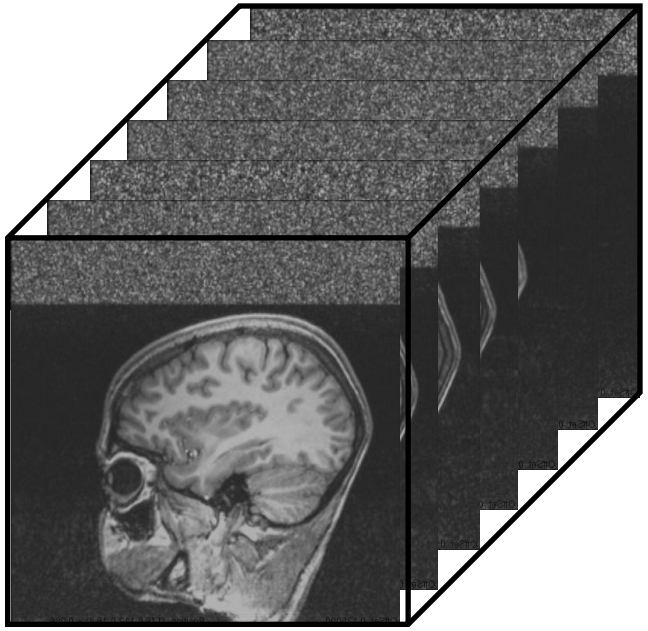
Knowing the speed of light gives (twice) the distance travelled.



Image courtesy of Wikipedia

Active Scanners: MRI Scanners

Magnetic fields + radio frequency



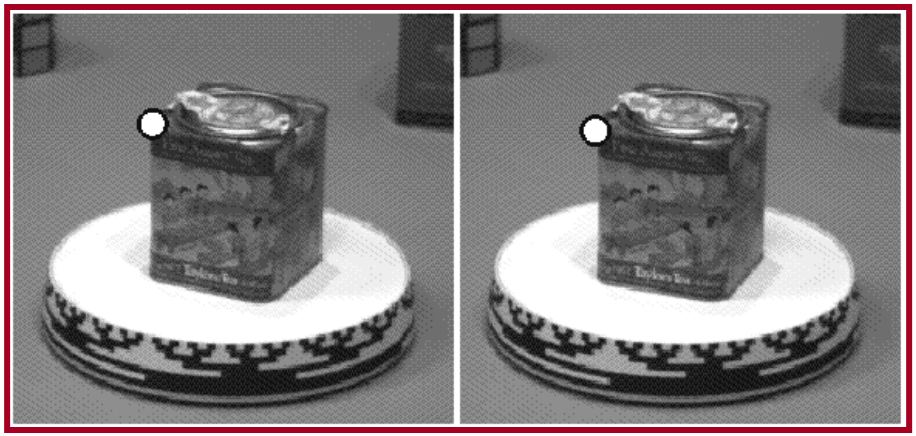
Model Construction

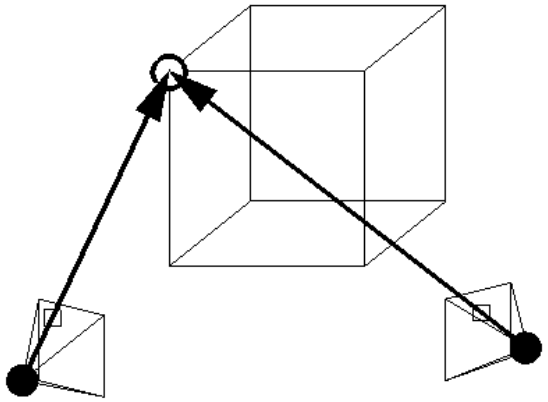


- Interactive modeling tools
  - CAD programs
- Active scanners
  - Light probes, triangulation/LiDAR/CAT/MRI scanners
- Passive scanners
  - Stereo, motion, etc.
- Procedural generation
  - Sweeps, fractals, grammars

Passive Scanners: Stereo

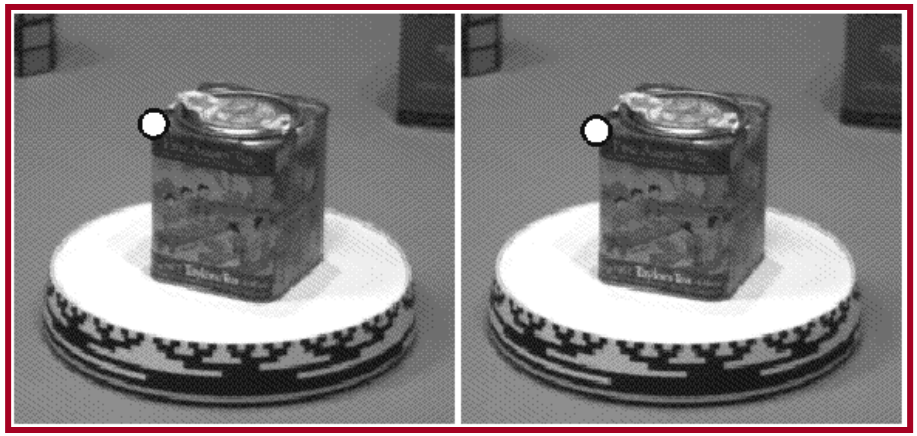
Image the scene from multiple **calibrated** locations





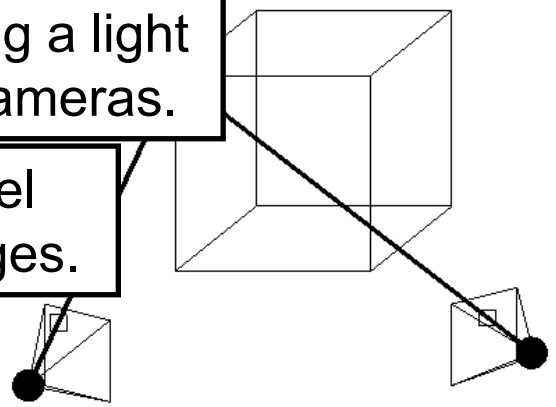
Passive Scanners: Stereo

Image the scene from multiple **calibrated** locations



Similar to scanners, but instead of using a light and a camera, triangulate using two cameras.

The challenge is to determine pixel correspondences between the images.



Passive Scanners: Motion

Image the scene from **uncalibrated** locations

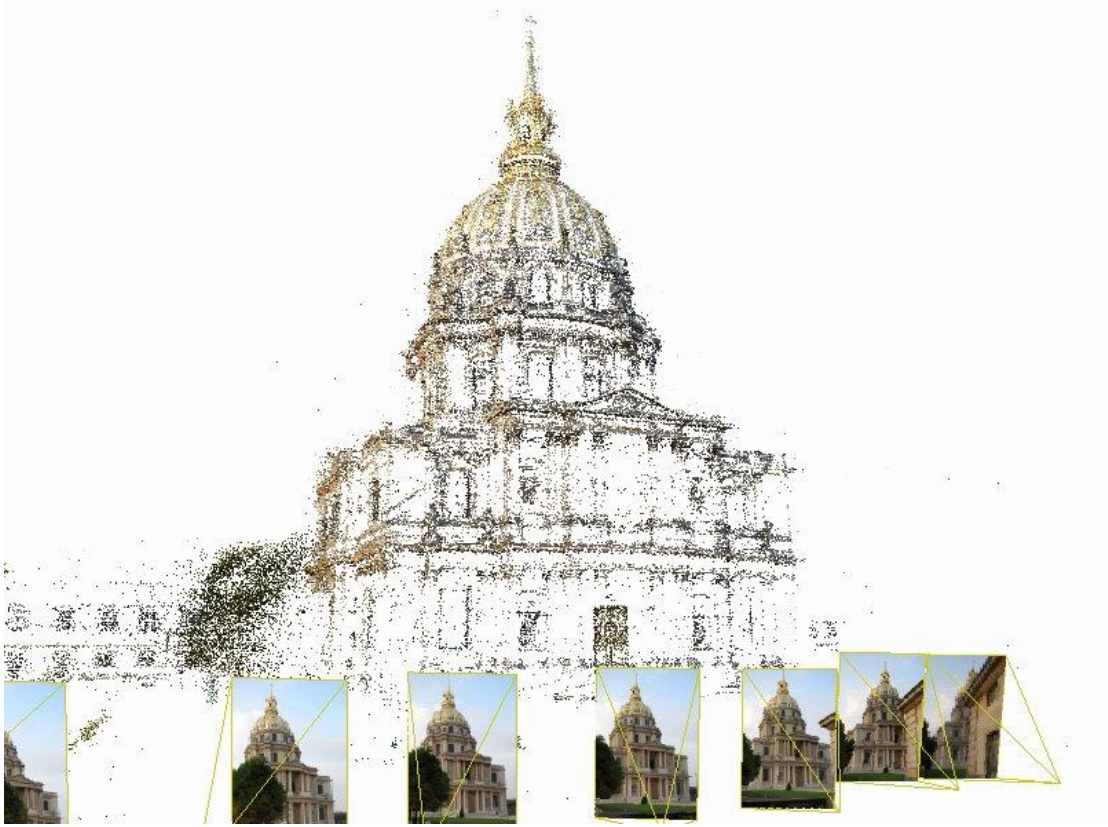
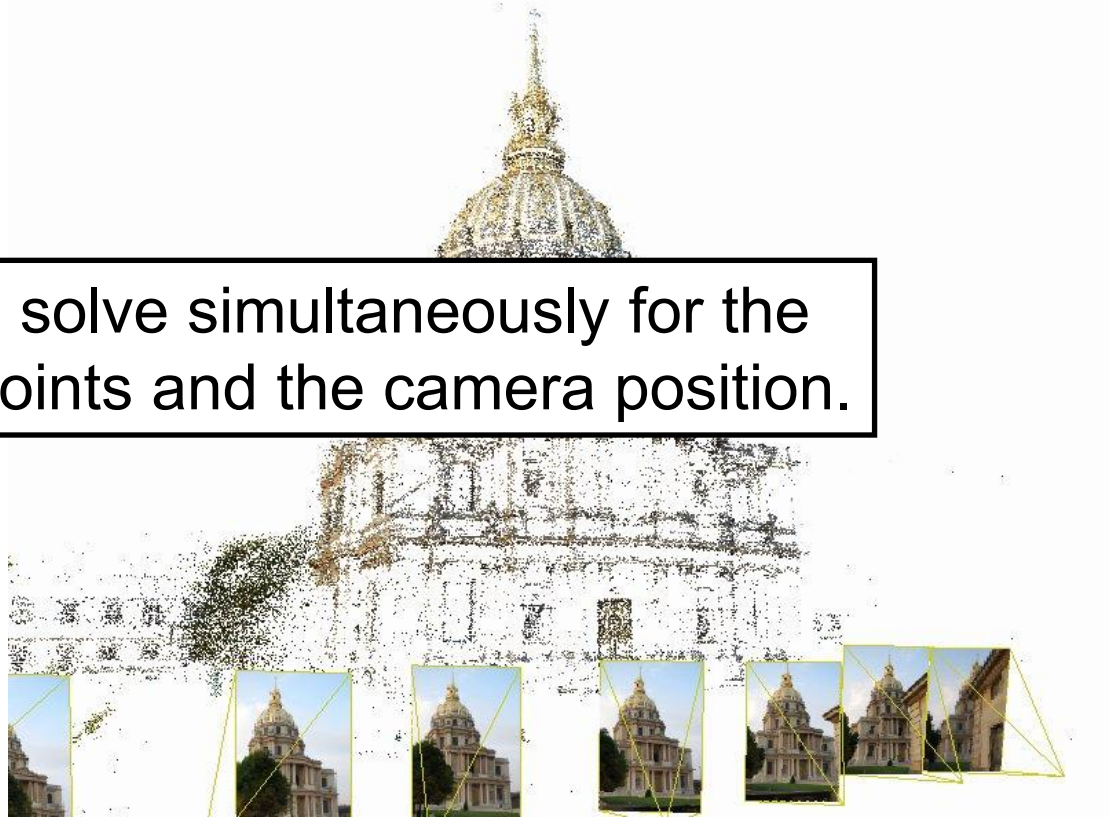


Image courtesy of http://www.maths.lth.se/matematiklth/personal/calle/

Passive Scanners: Motion

Image the scene from **uncalibrated** locations



Need to solve simultaneously for the surface points and the camera position.

Image courtesy of http://www.maths.lth.se/matematiklth/personal/calle/

Model Construction



- Interactive modeling tools
  - CAD programs
- Active scanners
  - Light probes, triangulation/LiDAR/CAT/MRI scanners
- Passive scanners
  - Stereo, motion, etc.
- Procedural generation
  - Sweeps, fractals, grammars

Procedural Modeling



Goal:

- Describe 3D models algorithmically

Best for models resulting from...

- Repeating/self-similar/random processes

Advantages:

- Automatic generation
- Concise representation
- Parameterized classes of models

Perlin noise is an example

Procedural Modeling: Sweeps

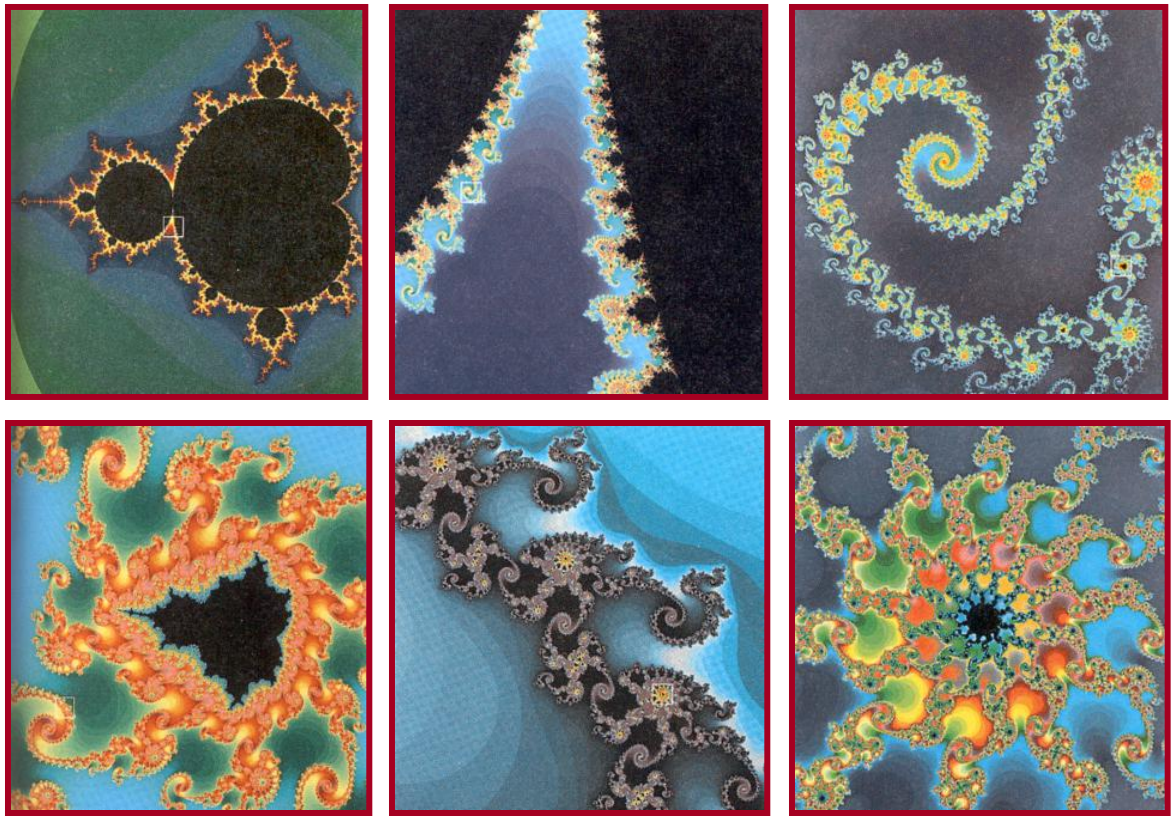


Transform a generating curve along a sweep curve

Procedural Modeling: Fractals

Defining characteristics:

- Self-similar with infinite resolution



Mandelbrot Set

H&B Figure 10.100

Procedural Modeling: Fractals



Deterministically self-similar fractals

- Parts are scaled copies of original

Statistically self-similar fractals

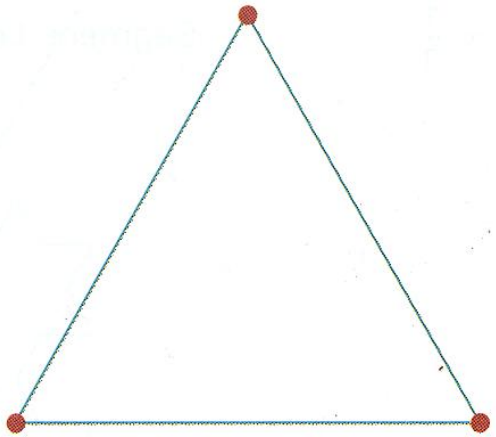
- Parts have same statistical properties as original

Procedural Modeling: Fractals

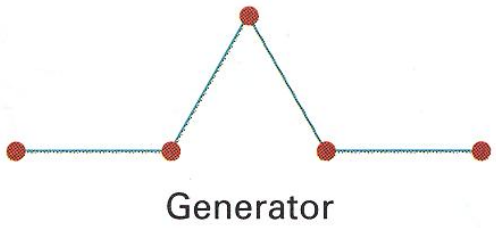


Deterministic fractal generation:

- **Initiator**: start with a shape
- **Generator**: replace subparts with scaled copy of original
- **Repeat**



Initiator



Generator

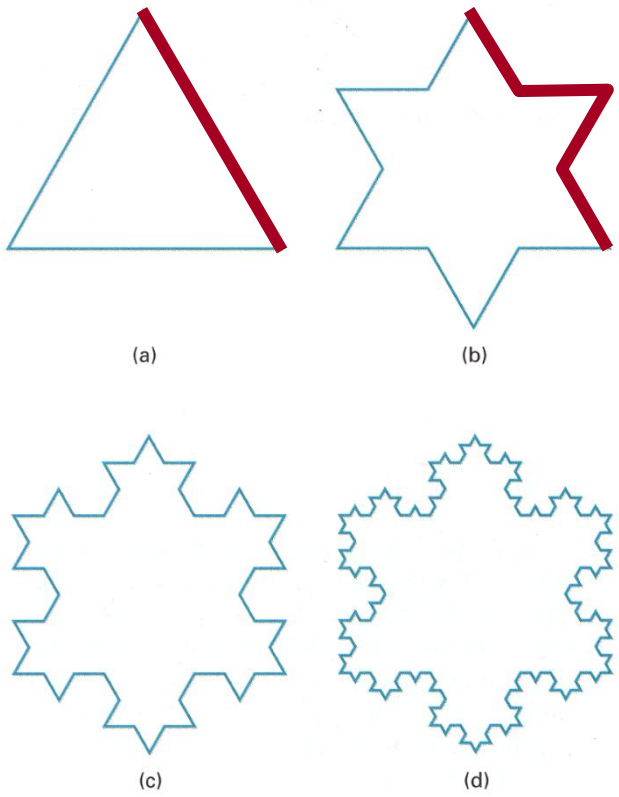
H&B Figure 10.68

Procedural Modeling: Fractals



Deterministic fractal generation:

- **Initiator**: start with a shape
- **Generator**: replace subparts with scaled copy of original
- **Repeat**



Koch Curve

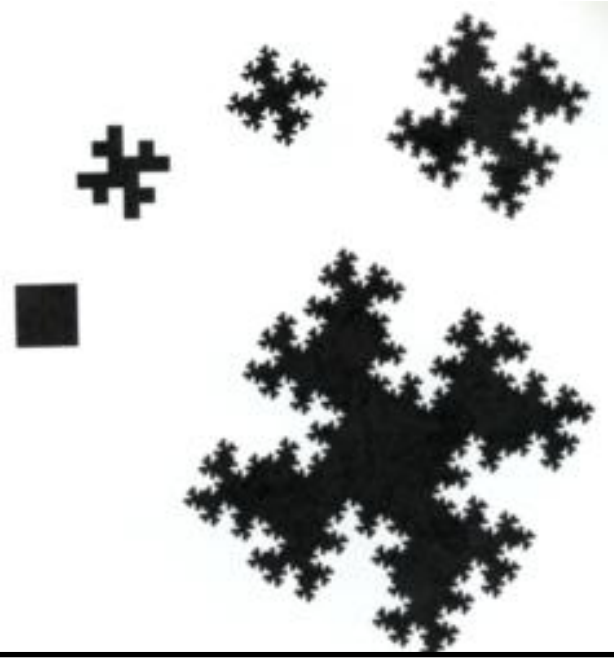
H&B Figure 10.69

Procedural Modeling: Fractals



Deterministic fractal generation:

- **Initiator**: start with a shape
- **Generator**: replace subparts with scaled copy of original
- **Repeat**



"Useful" for creating interesting shapes.

Mandelbrot Figure X

Procedural Modeling: Fractals



Deterministically self-similar fractals

- Parts are scaled copies of original

Statistically self-similar fractals

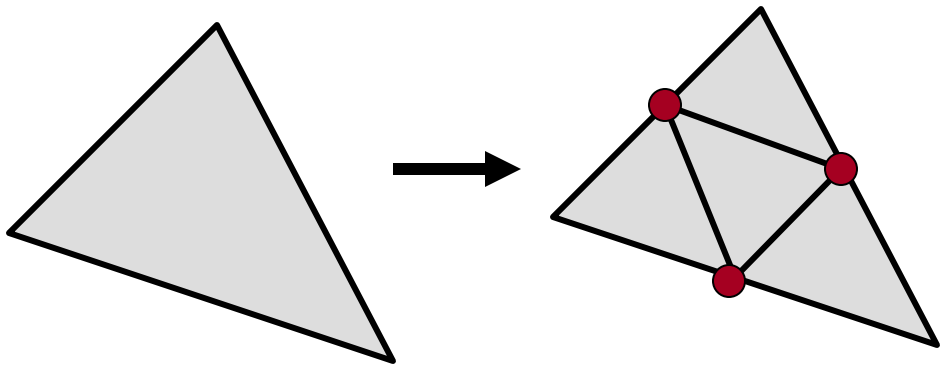
- Parts have same statistical properties as original

Procedural Modeling: Fractals



Statistical fractal generation:

- **Initiator**: start with a shape
- **Generator**: replace subparts with a scaled copy
- **Repeat**



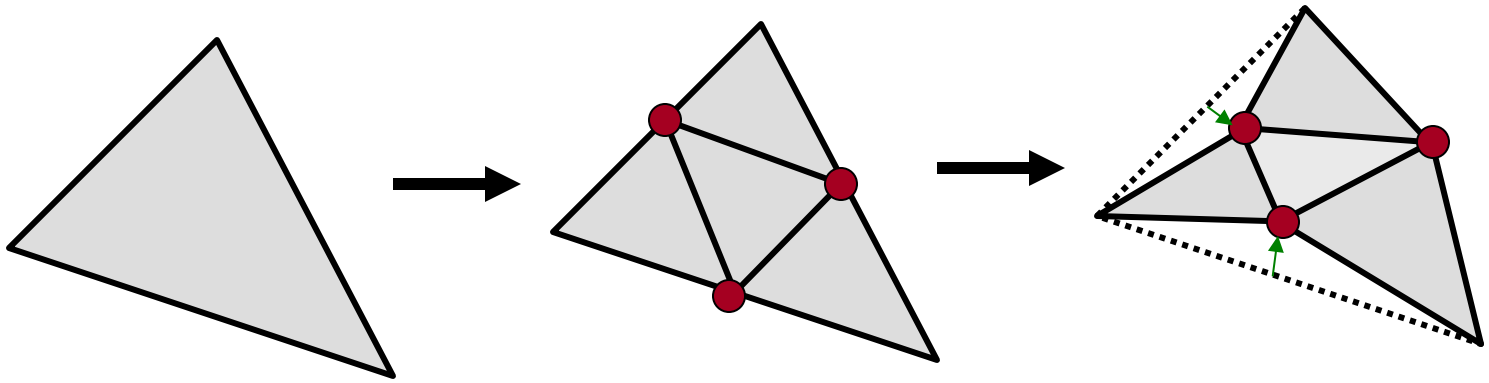
Procedural Modeling: Fractals



Statistical fractal generation:

- **Initiator**: start with a shape
- **Generator**: replace subparts with a scaled copy
- **Repeat**

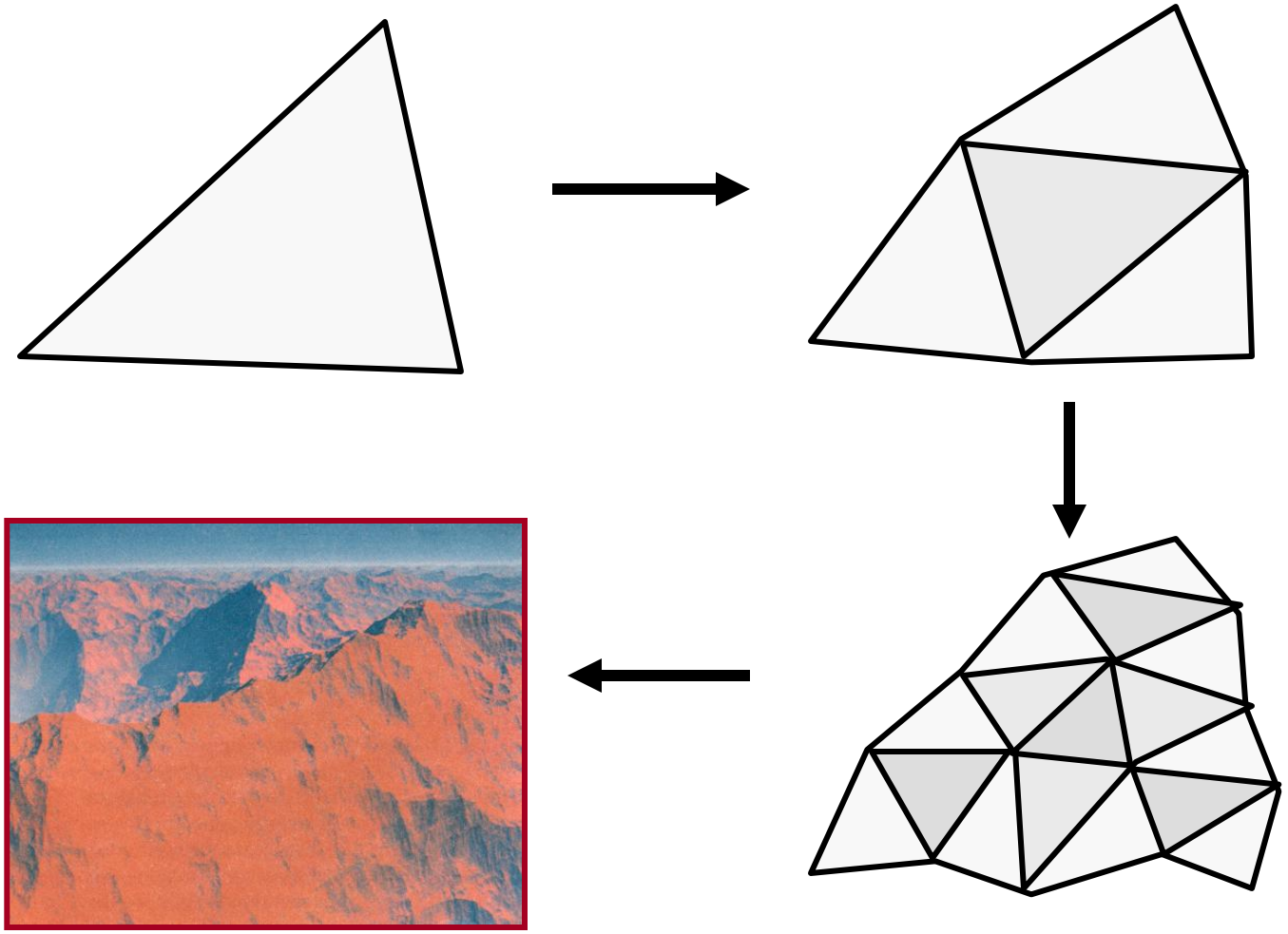
statistically self-similar



Random Midpoint Displacement

Procedural Modeling: Fractals

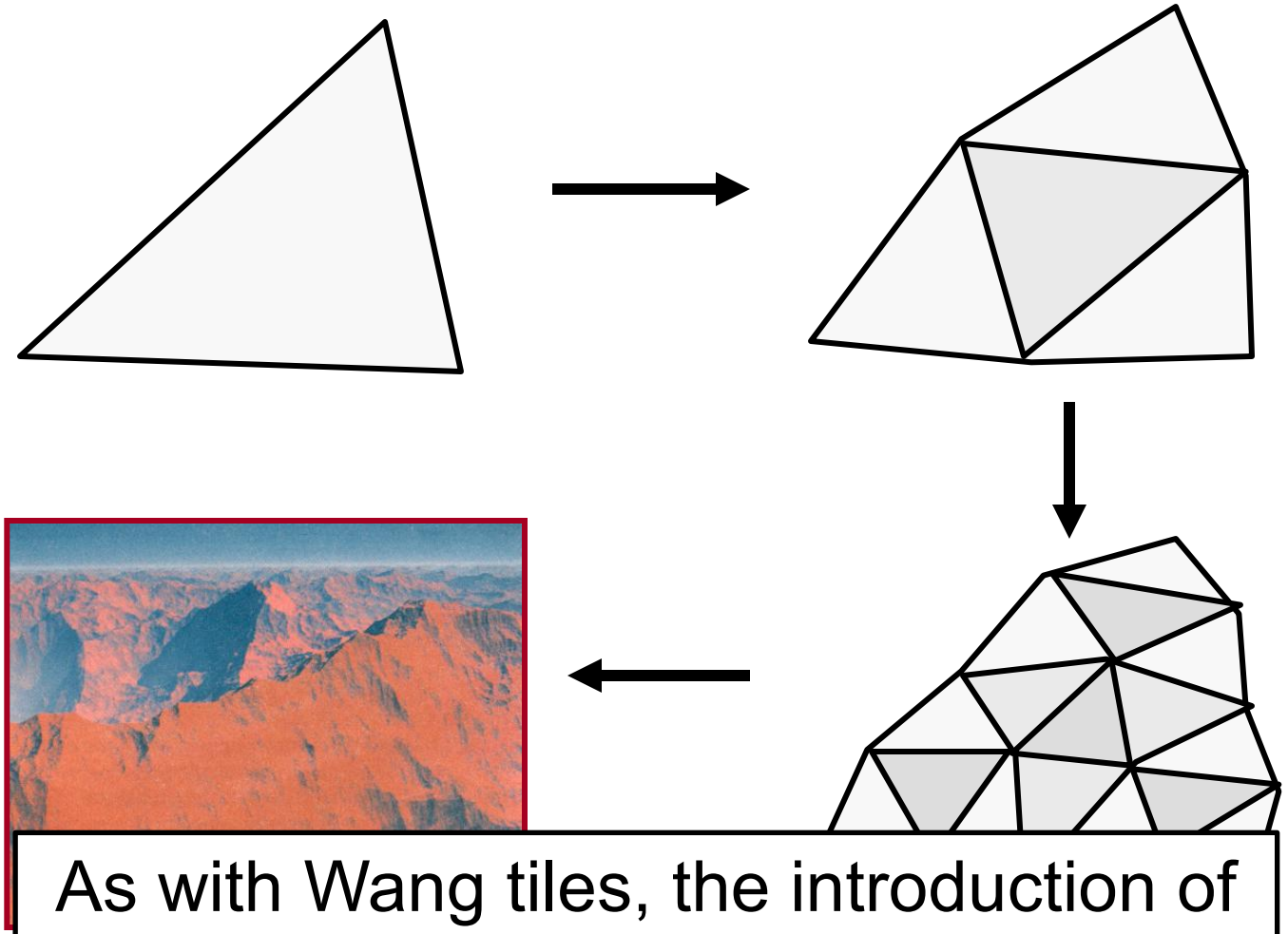
Example: terrain



H&B Figure 10.83b

Procedural Modeling: Fractals

Example: terrain



As with Wang tiles, the introduction of randomness removes repetitiveness

Statistical Fractal Generation

Useful for creating mountains



H&B Figure 10.83a

Statistical Fractal Generation

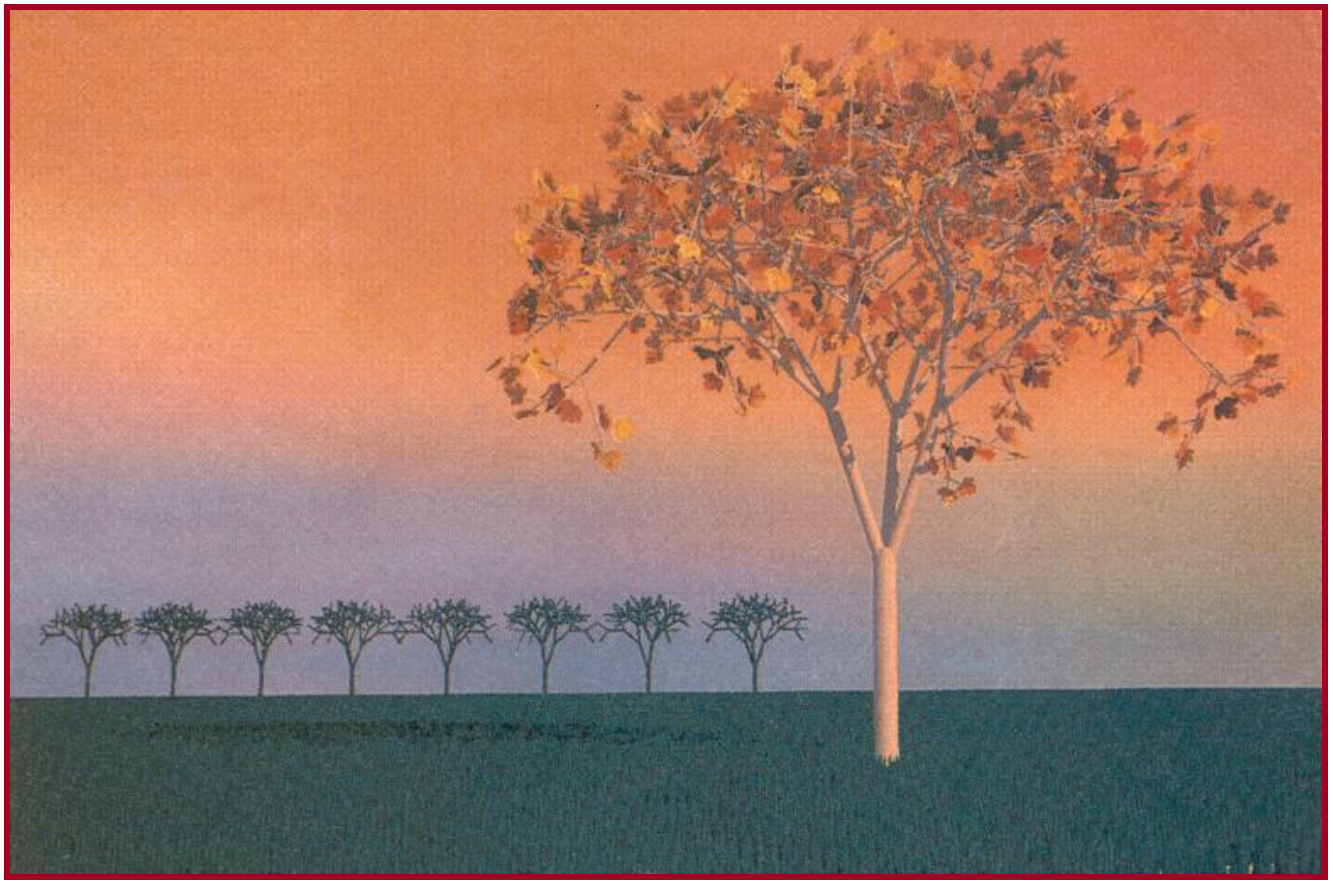
Useful for creating 3D plants



H&B Figure 10.82

Statistical Fractal Generation

Useful for creating 3D plants



H&B Figure 10.79

Procedural Modeling: Grammars



Generate description of geometric model by applying production rules

$$
\begin{aligned}
S &\rightarrow AB \\
A &\rightarrow Ba \mid a \\
B &\rightarrow Ab \mid b
\end{aligned}
$$

Procedural Modeling: Grammars



Generate description of geometric model by applying production rules

$$S \rightarrow AB$$
$$A \rightarrow Ba \mid a$$
$$B \rightarrow Ab \mid b$$

AB

Procedural Modeling: Grammars



Generate description of geometric model by applying production rules

$$S \rightarrow AB$$
$$A \rightarrow Ba \mid a$$
$$B \rightarrow Ab \mid b$$

AB
BaB

Procedural Modeling: Grammars



Generate description of geometric model by applying production rules

$$S \rightarrow AB$$
$$A \rightarrow Ba \mid a$$
$$B \rightarrow Ab \mid b$$

AB
BaB
BaAb

Procedural Modeling: Grammars



Generate description of geometric model by applying production rules

$$S \rightarrow AB$$
$$A \rightarrow Ba \mid a$$
$$B \rightarrow Ab \mid b$$

AB
BaB
BaAb
AbaAb

Procedural Modeling: Grammars



Generate description of geometric model by applying production rules

$$S \rightarrow AB$$
$$A \rightarrow Ba \mid a$$
$$B \rightarrow Ab \mid b$$

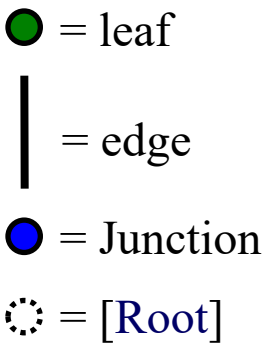
AB
BaB
BaAb
AbaAb
⋮

Procedural Modeling: Grammars



Can be used for modeling plants

Start → [Root]
Root → Junction + Root | leaf
Junction → [Root] | edge

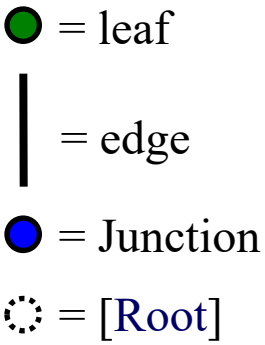


Procedural Modeling: Grammars



Can be used for modeling plants [R]

Start → [Root]
Root → Junction + Root | leaf
Junction → [Root] | edge

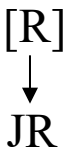




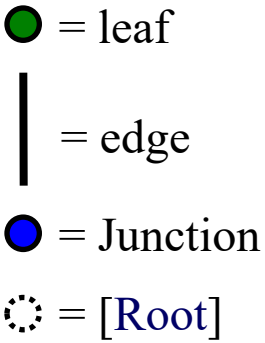
Procedural Modeling: Grammars

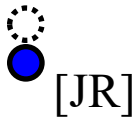


Can be used for modeling plants



Start → [Root]
Root → Junction + Root | leaf
Junction → [Root] | edge

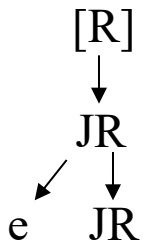




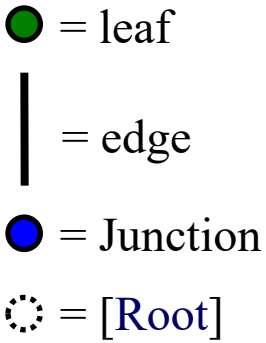
Procedural Modeling: Grammars

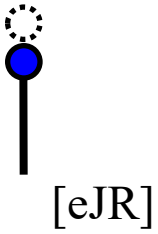


Can be used for modeling plants



Start → [Root]
Root → Junction + Root | leaf
Junction → [Root] | edge

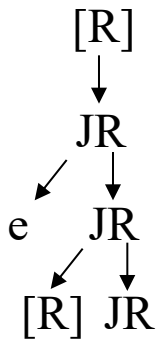




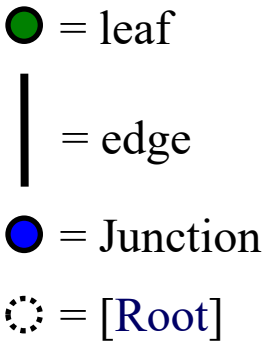
Procedural Modeling: Grammars



Can be used for modeling plants



Start → [Root]
Root → Junction + Root | leaf
Junction → [Root] | edge





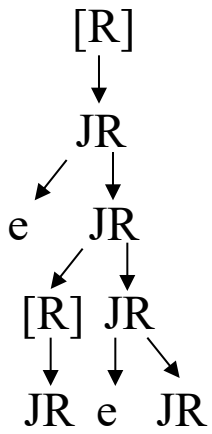
[e[R]JR]

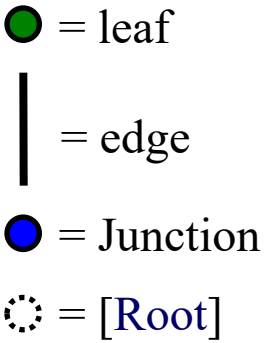
Procedural Modeling: Grammars

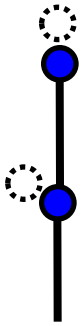


Can be used for modeling plants

Start → [Root]
Root → Junction + Root | leaf
Junction → [Root] | edge





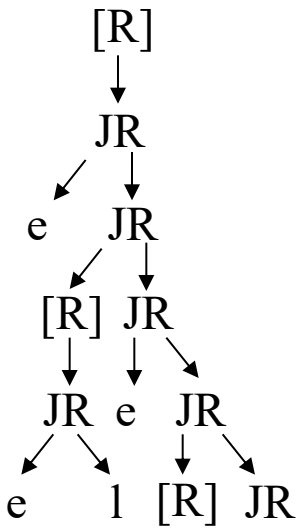


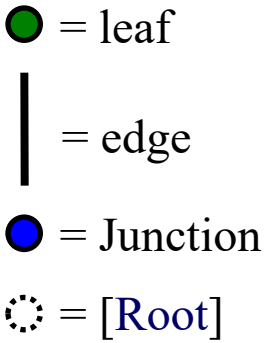
[e[JR]eJR]

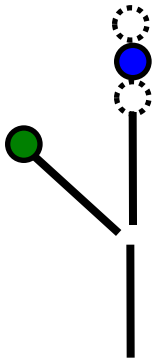
Procedural Modeling: Grammars

Can be used for modeling plants

Start → [Root]
Root → Junction + Root | leaf
Junction → [Root] | edge





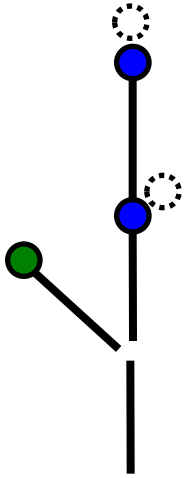


[e[el]e[R]JR]

Procedural Modeling: Grammars

Can be used for modeling plants

Start → [Root]
Root → Junction + Root | leaf
Junction → [Root] | edge



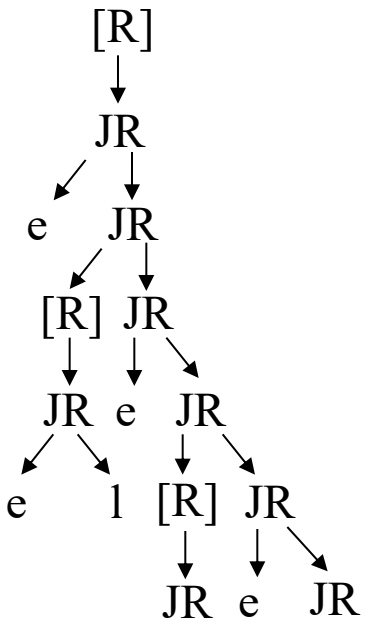
[e[el]e[JR]eJR]

= leaf

= edge

= Junction

= [Root]

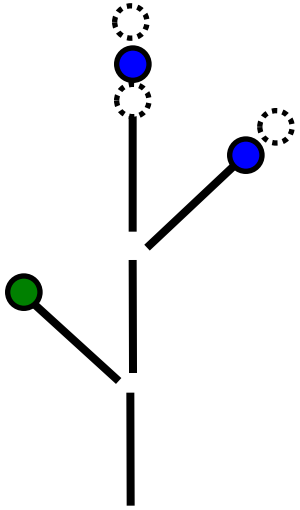


Procedural Modeling: Grammars

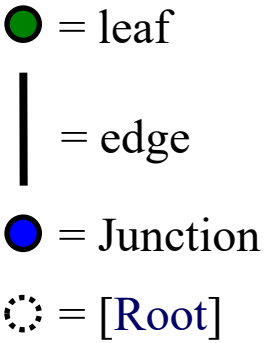


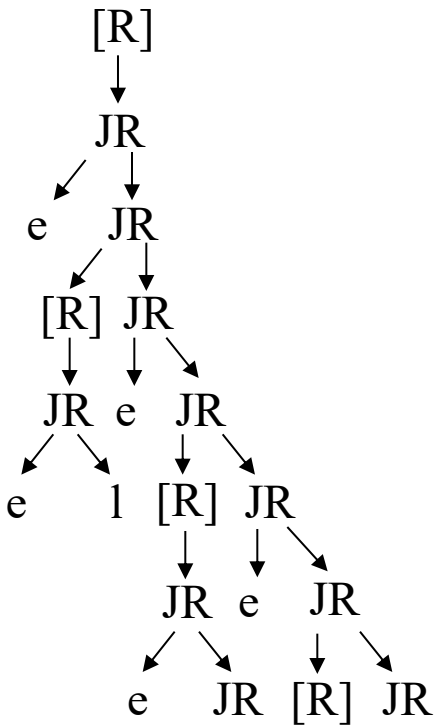
Can be used for modeling plants

Start → [Root]
Root → Junction + Root | leaf
Junction → [Root] | edge



[e[el]e[eJR]e[R]JR]



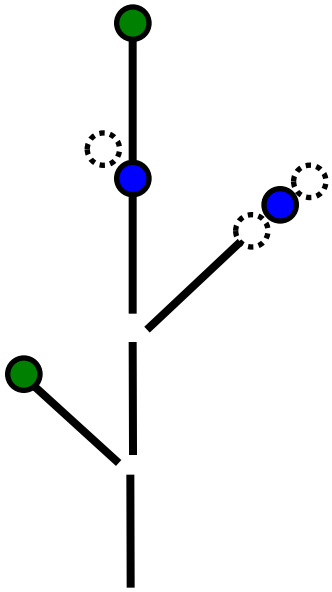


Procedural Modeling: Grammars

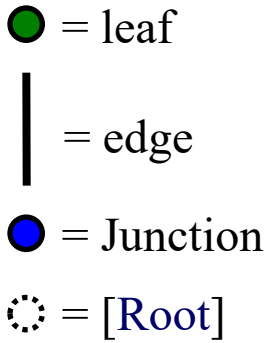


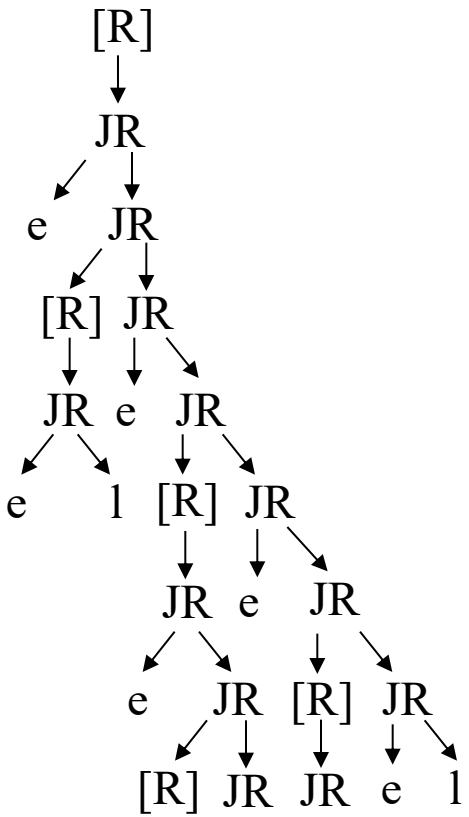
Can be used for modeling plants

Start → [Root]
Root → Junction + Root | leaf
Junction → [Root] | edge



[e[el]e[e[R]JR]e[JR]el]

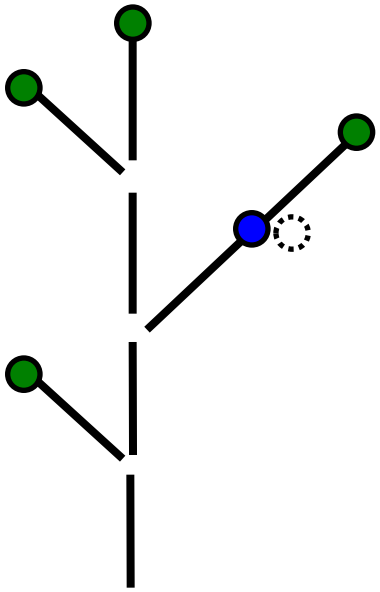




Procedural Modeling: Grammars

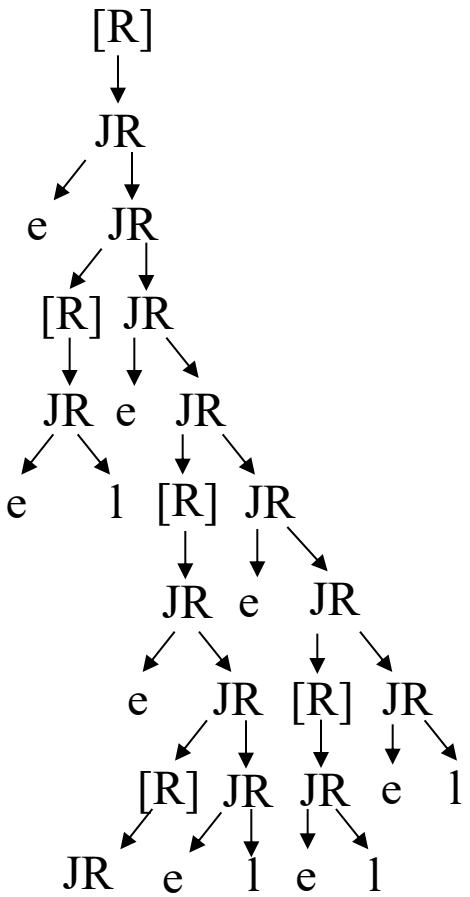
Can be used for modeling plants

Start → [Root]
Root → Junction + Root | leaf
Junction → [Root] | edge



= leaf
= edge
= Junction
= [Root]

[e[el]e[e[JR]el]e[el]el]

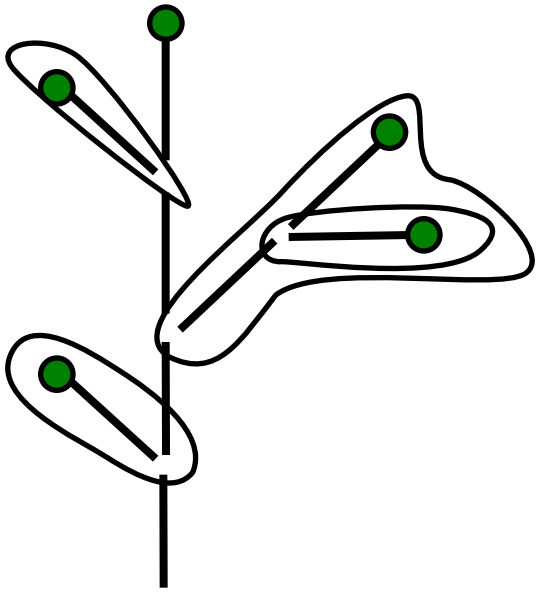


Procedural Modeling: Grammars



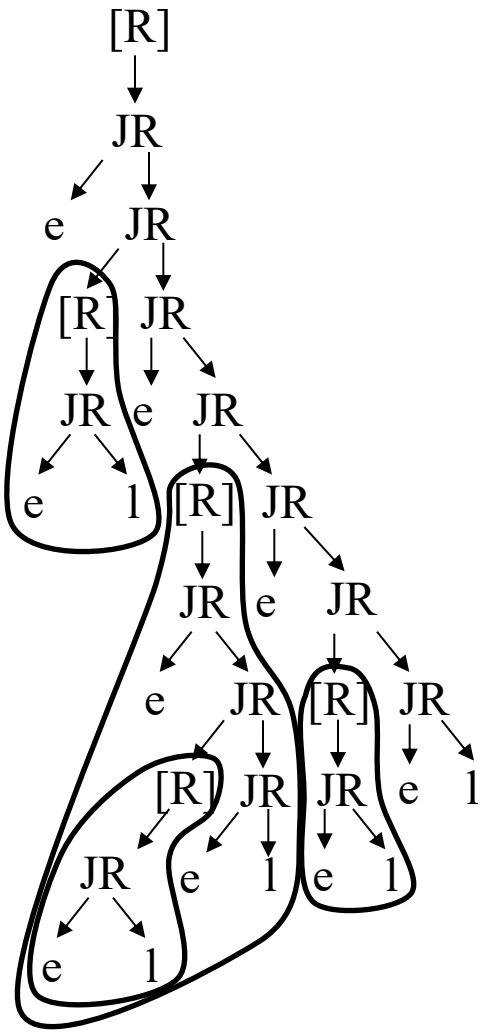
Can be used for modeling plants

Start → [Root]
Root → Junction + Root | leaf
Junction → [Root] | edge



= leaf
= edge
= Junction
= [Root]

[e[el]e[e[el]el]e[el]el]

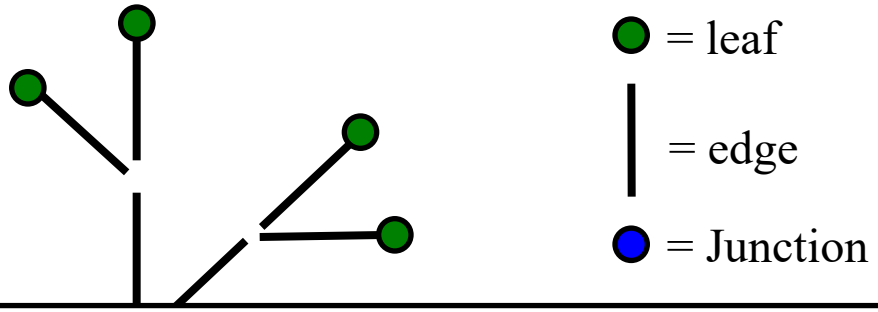


Procedural Modeling: Grammars



Can be used for modeling plants

Start → [Root]
Root → Junction + Root | leaf
Junction → [Root] | edge



Like Wang tiles:

- Choices are discrete
- Ability to choose creates variability and removes repetition