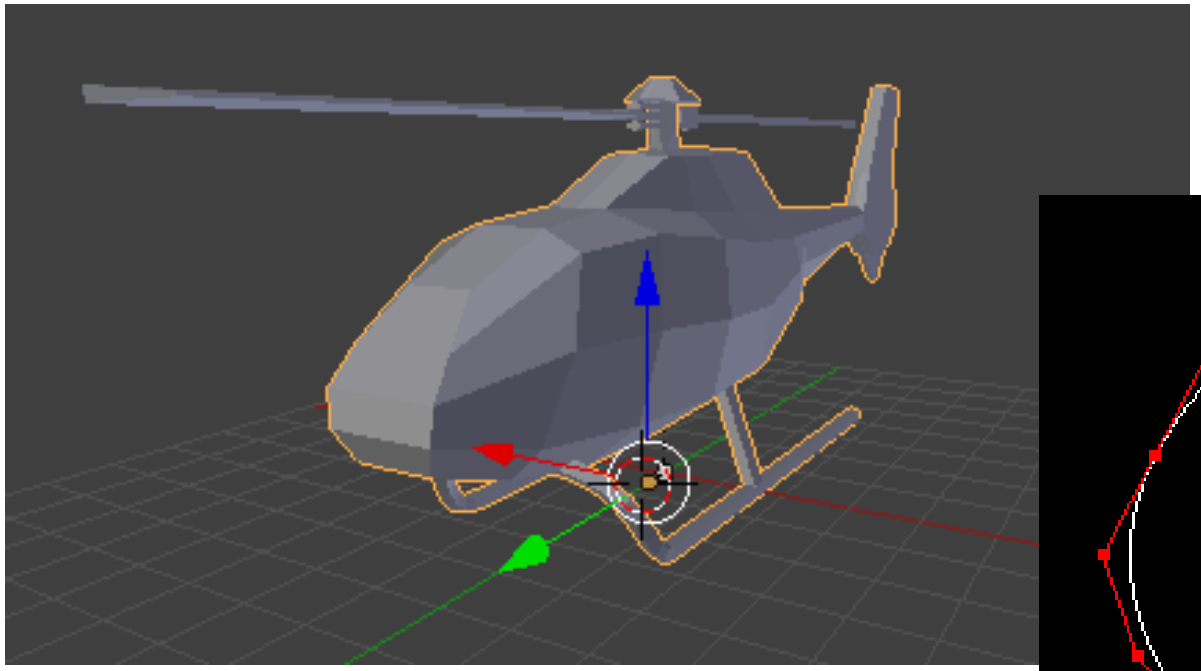
Computer Graphics

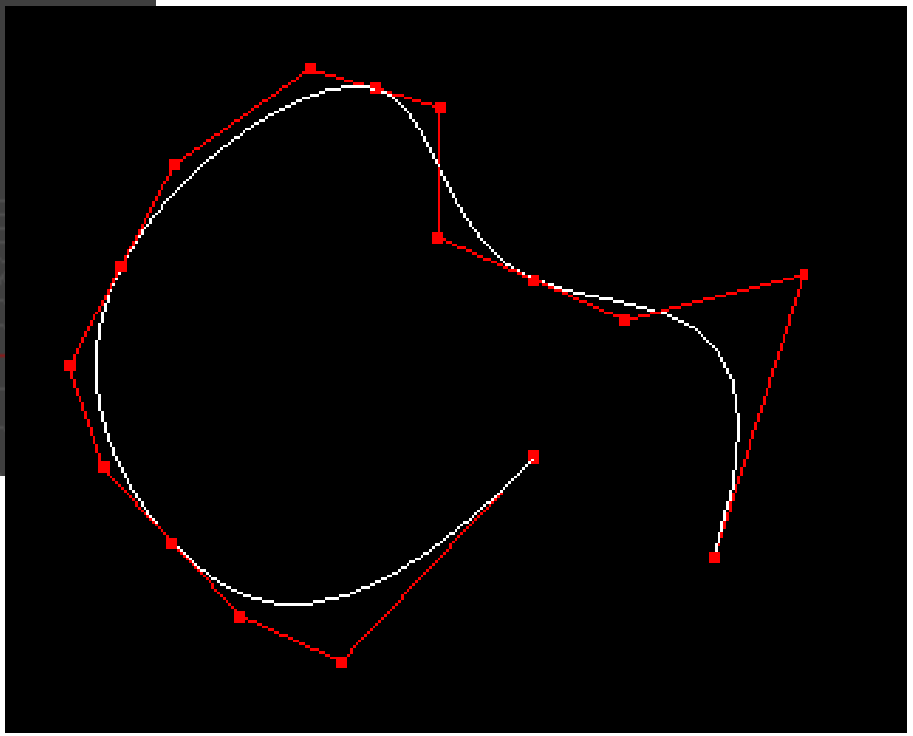
MTAT.03.015

Raimond Tunnel

Study IT in .ee

The Road So Far...





```
mtllib triangle.mtl
o Plane
v 1.007839 0.000000 -1.000000
v 1.000000 0.000000 0.978599
v -1.000000 0.000000 -0.588960
usemtl None
s off
f 3 2 1
```

Procedural Generation

- Generating objects algorithmically

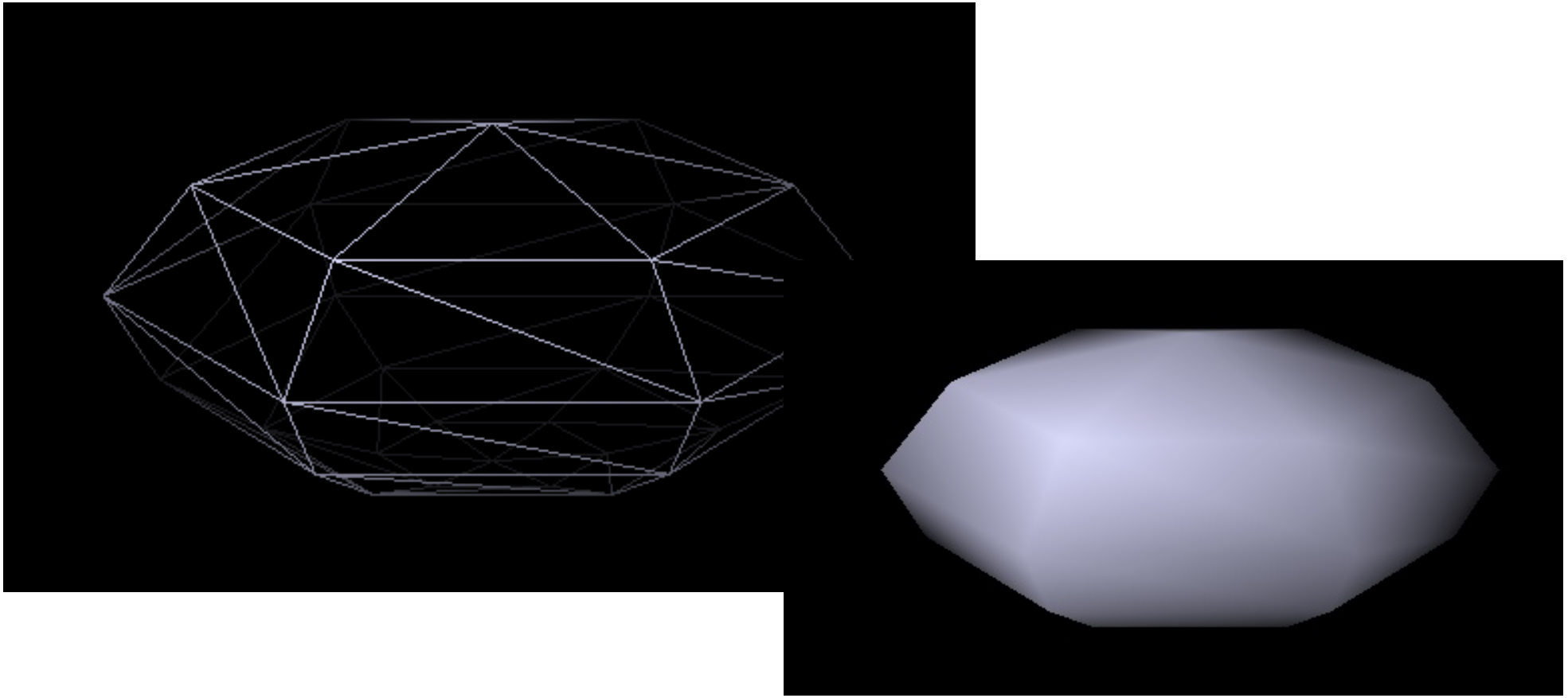
```
for(y = 0; y <= heightSegments; y++) {
    for(x = 0; x <= widthSegments; x++) {
        u = (float)x / widthSegments;
        v = (float)y / heightSegments;

        glm::vec3 vertex = glm::vec3(
            -radius * glm::cos(phiStart + u * phiLength) * glm::sin(thetaStart + v * thetaLength),
            radius * glm::cos(thetaStart + v * thetaLength),
            radius * glm::sin(phiStart + u * phiLength) * glm::sin(thetaStart + v * thetaLength)
        );

        vertices.push_back(vertex);
        normals.push_back(glm::normalize(vertex));
        colors.push_back(color);
    }
}
```

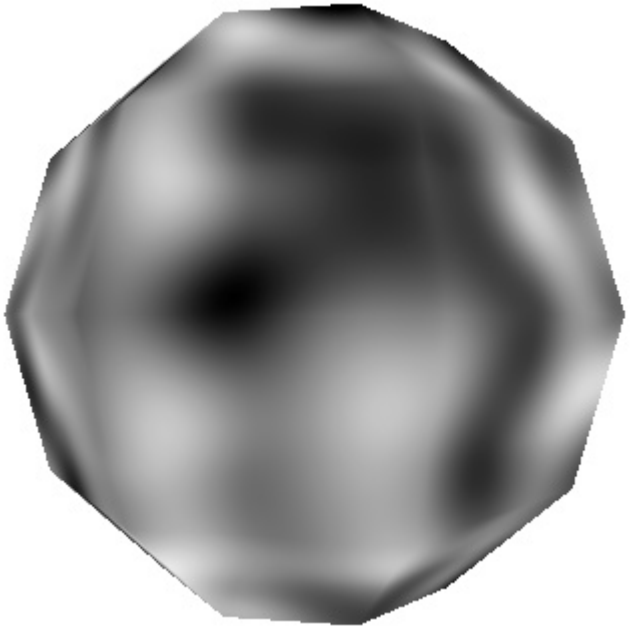
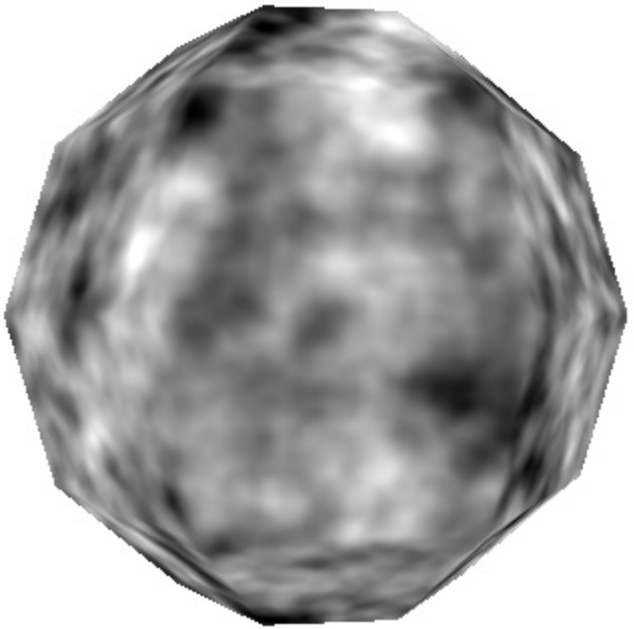
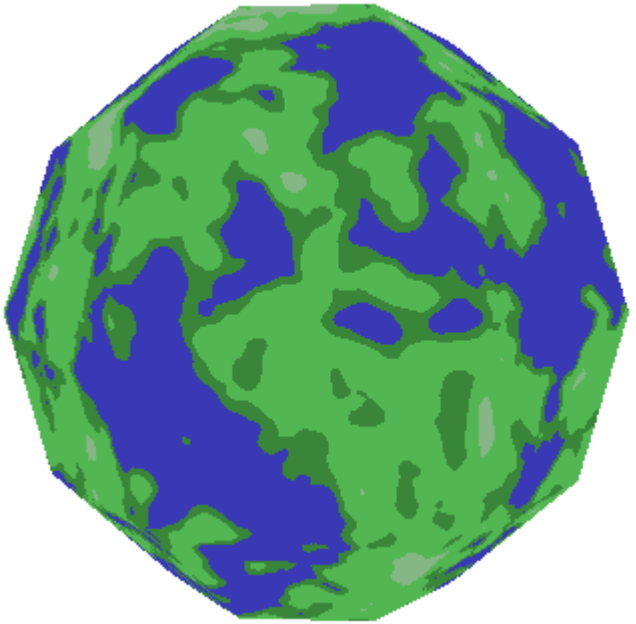
Procedural Generation

- Generating objects algorithmically
  - Mesh (geometry)



Procedural Generation

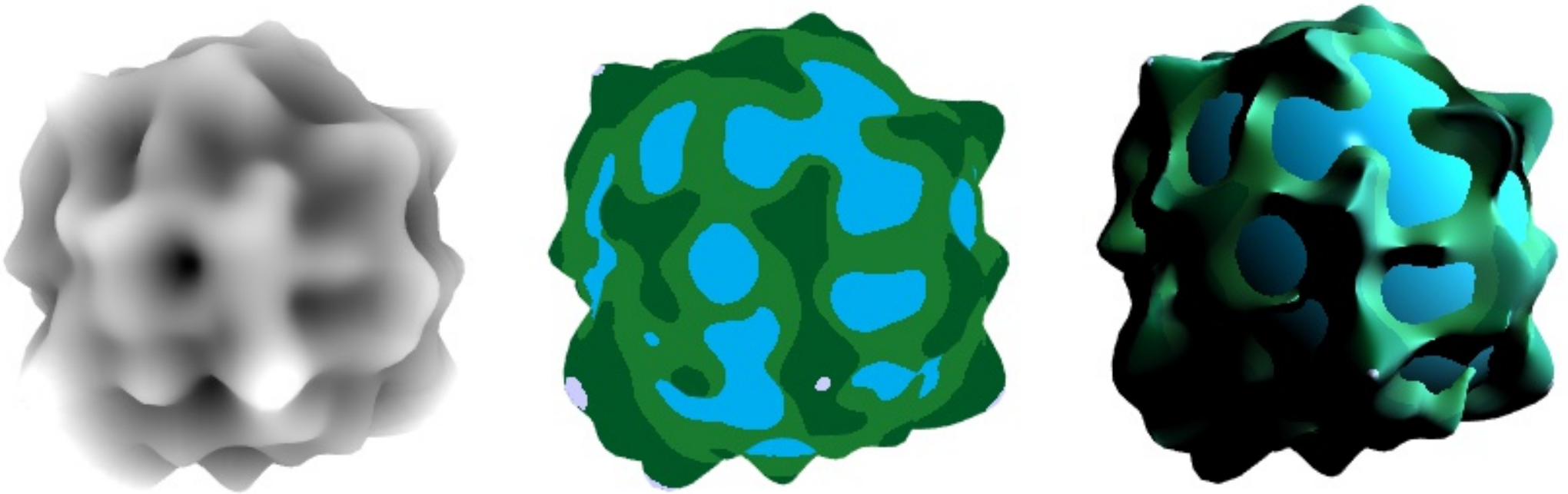
- Generating objects algorithmically
  - Mesh (geometry)
  - Material (texture)

Procedural Generation

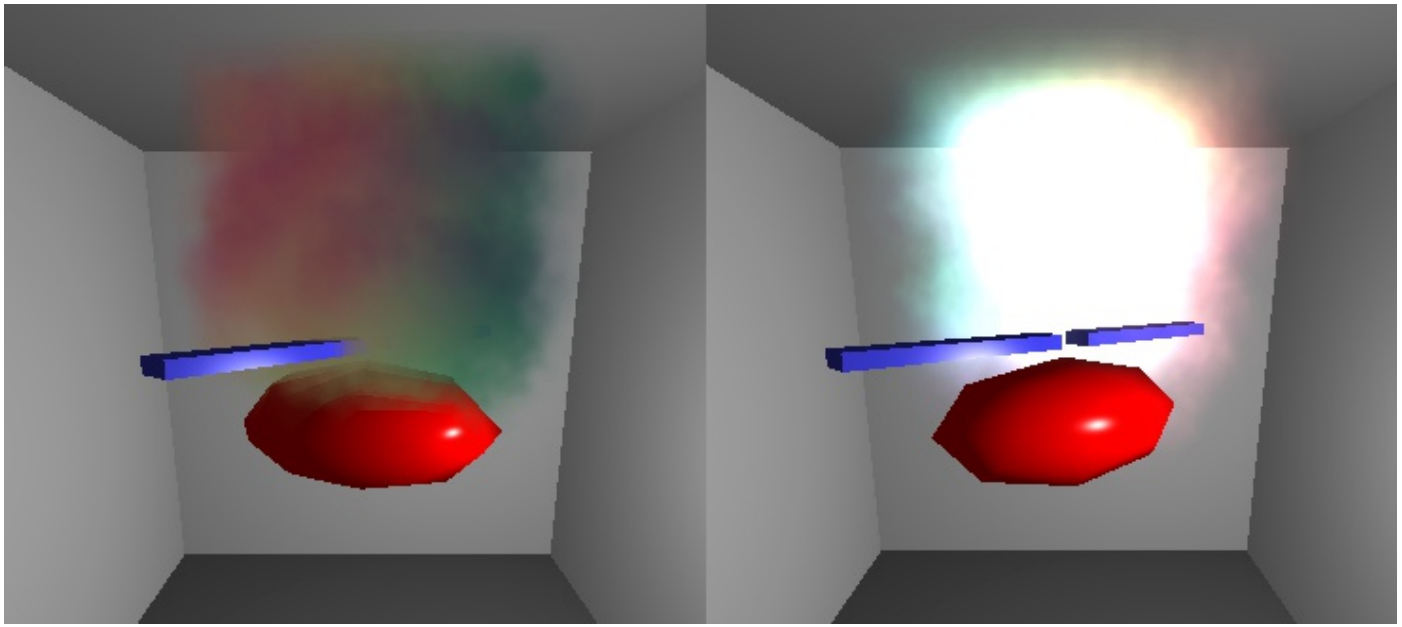
- Generating objects algorithmically
  - Mesh (geometry)
  - Material (texture)

Combination



Procedural Generation

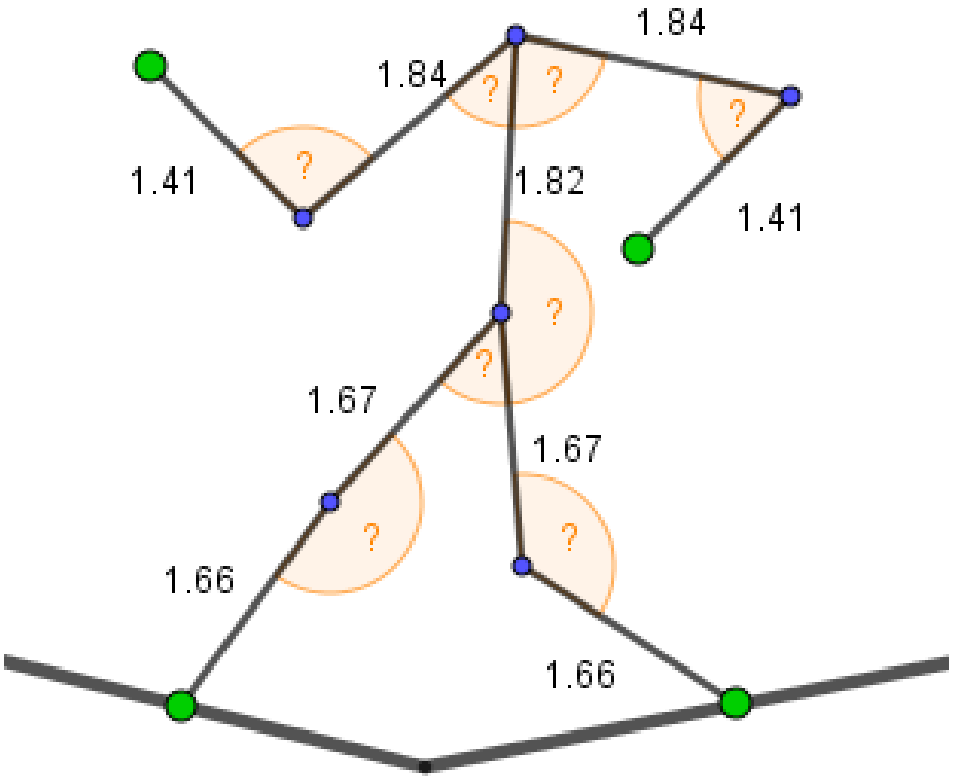
- Generating objects algorithmically
  - Mesh (geometry)
  - Material (texture)
  - Effects (particles)



Custom B. Chopper solution by Siim Raudsepp

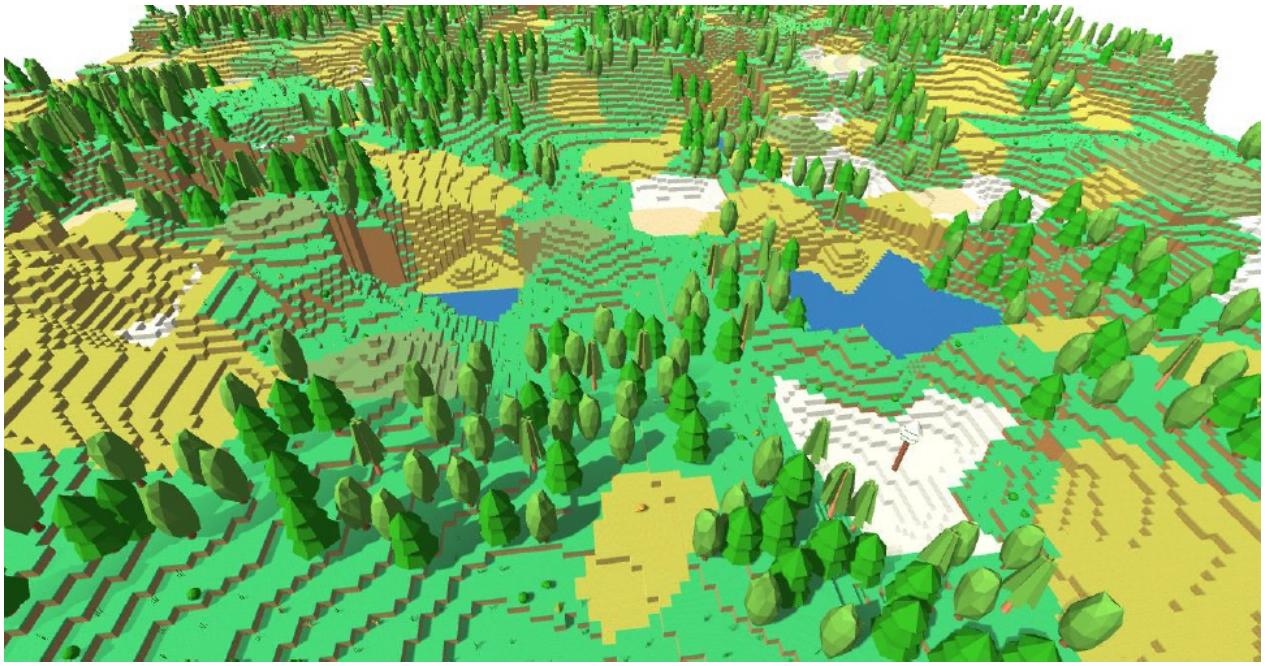
Procedural Generation

- Generating objects algorithmically
  - Mesh (geometry)
  - Material (texture)
  - Effects (particles)
  - Animation



Procedural Generation

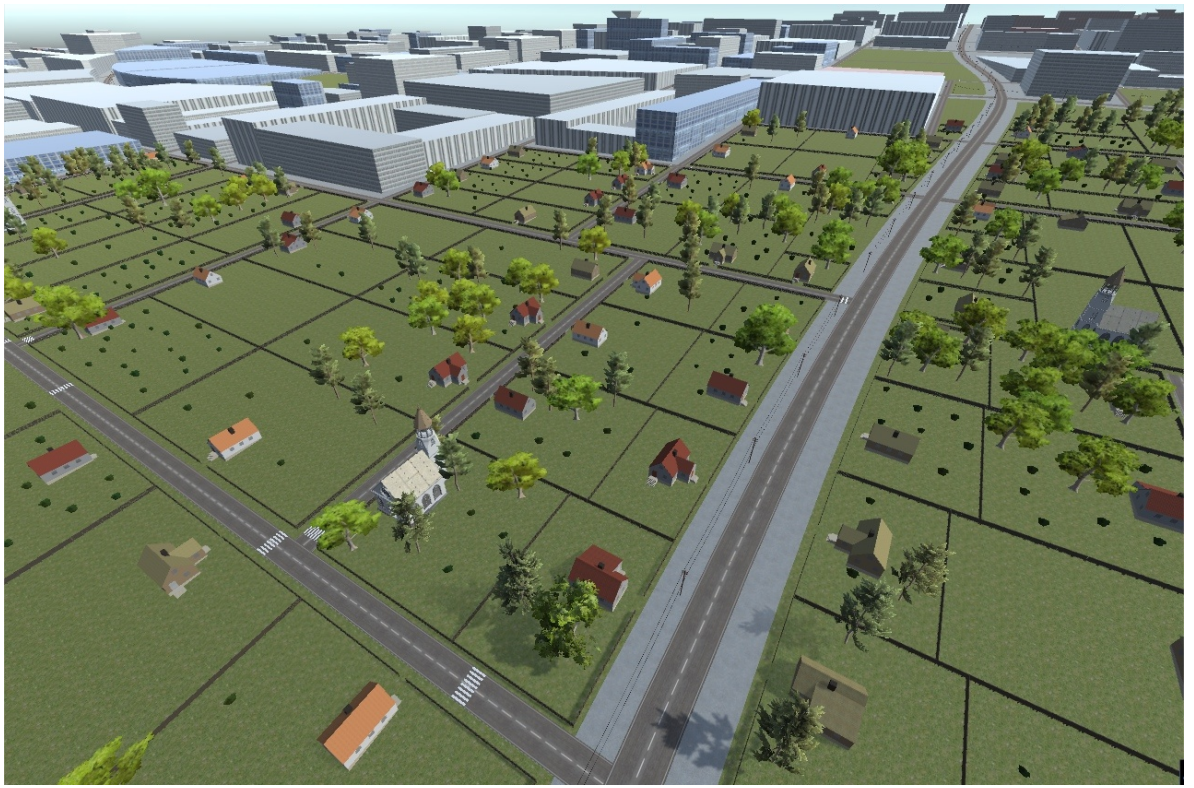
- Generating objects algorithmically
  - Mesh (geometry)
  - Material (texture)
  - Effects (particles)
  - Animation
  - Worlds



Procedural Infinite Terrain Generation (BSc thesis) by Andreas Sepp

Procedural Generation

- Generating objects algorithmically
  - Mesh (geometry)
  - Material (texture)
  - Effects (particles)
  - Animation
  - Worlds



Infinite Procedural Infrastructured World Generation (MSc thesis) by Andreas Sepp

Procedural Generation

- Generating objects algorithmically
  - Mesh (geometry)
  - Material (texture)
  - Effects (particles)
  - Animation
  - Worlds
  - Characters, weapons, space ships, ...

NPC Generator
by Jaanus Jaggo

Procedural Generation

- Generating objects algorithmically
  - Mesh (geometry)
  - Material (texture)
  - Effects (particles)
  - Animation
  - Worlds
  - Characters, weapons, space ships, ...
- More content, less repetative work for artists

Tree

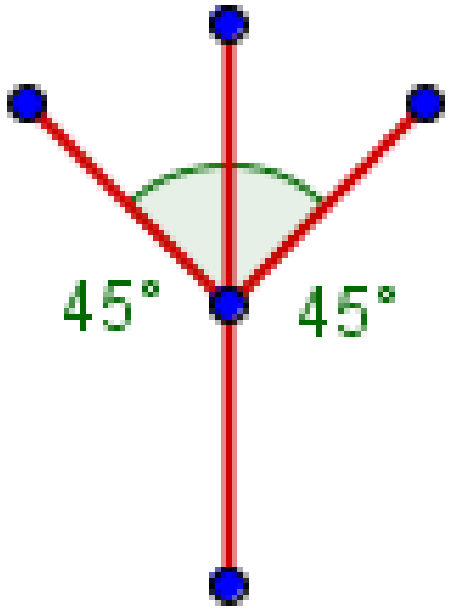
- Let's try to generate a tree branch structure.

Tree

- Let's try to generate a tree branch structure.
- We start with a trunk.

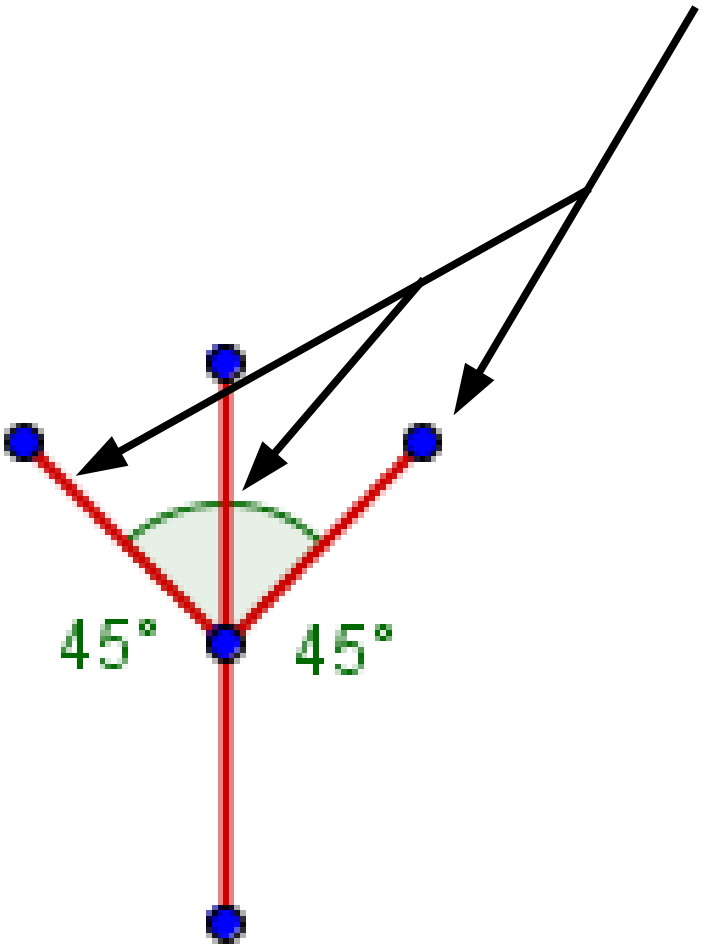
Tree

- From the trunk, we create two branches for either side.
- We also continue on the forward path.



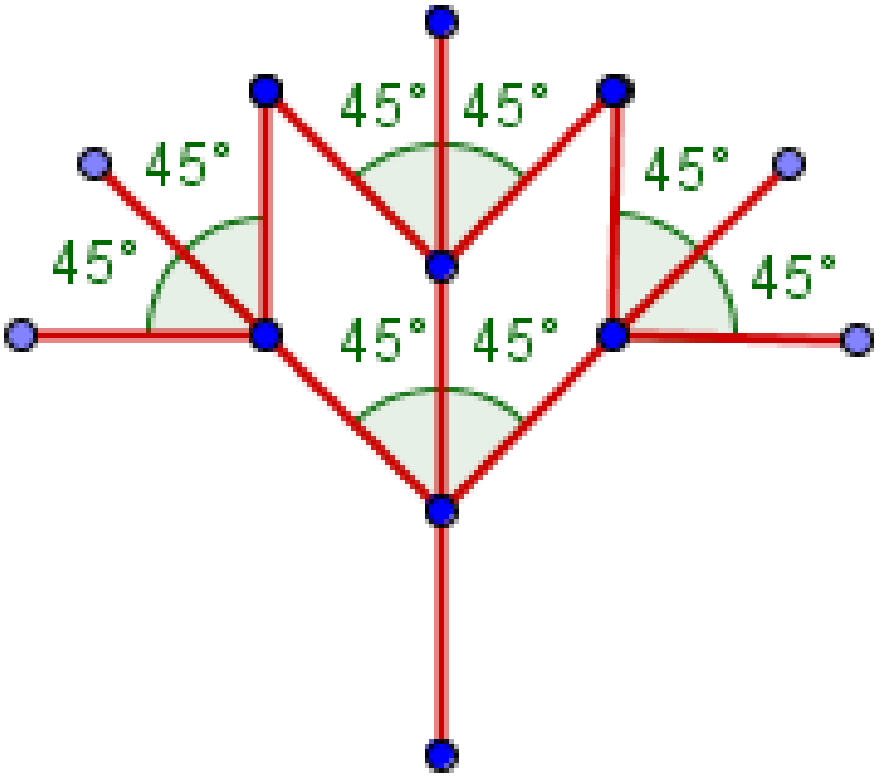
Tree

- We repeat the process for the new segments.



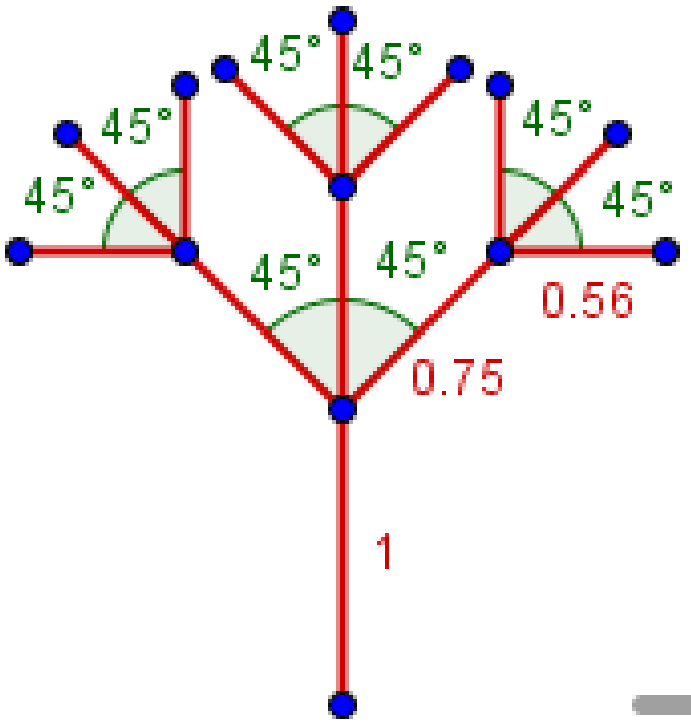
Tree

- We repeat the same process for all of the new segments.



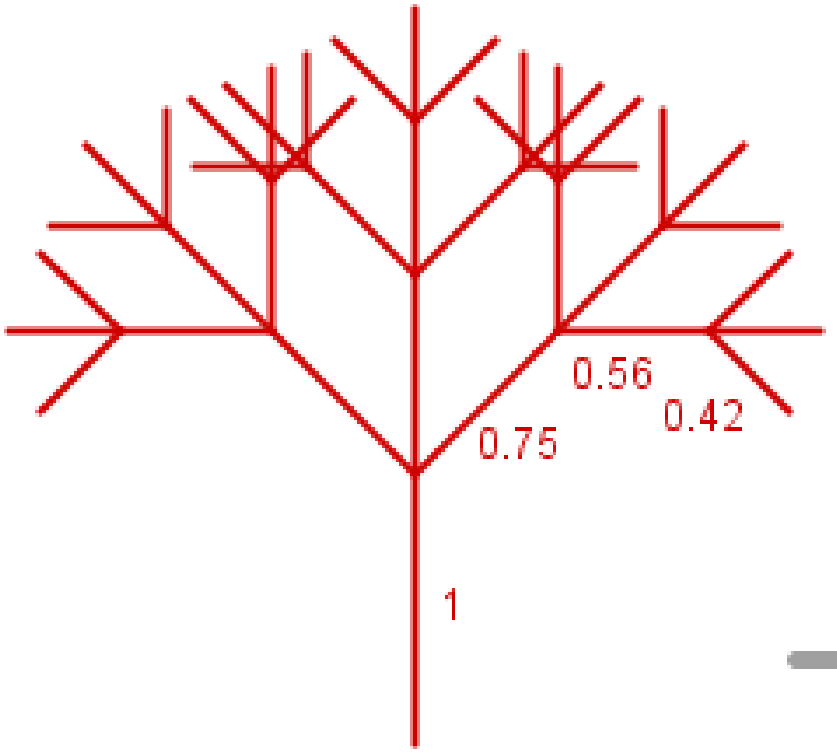
Tree

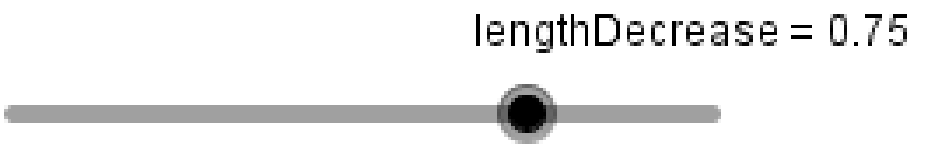
- Decrease the length of the segments each time.



Tree

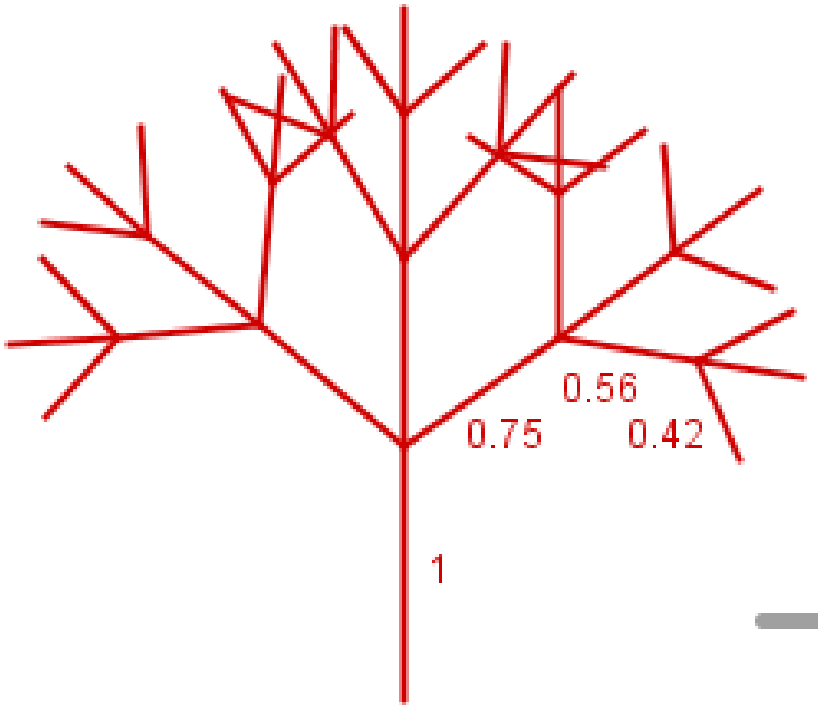
- Repeat again the same process.

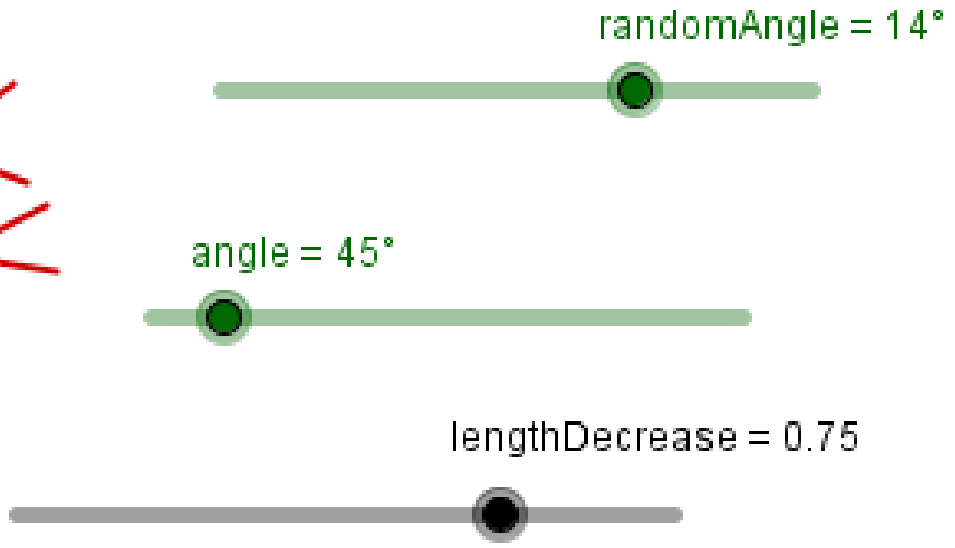




Tree

- Introduce randomness.





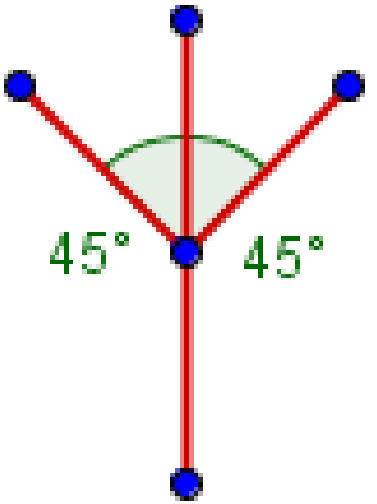
Show this in action...

Tree

- What if we want to store the generated structure?

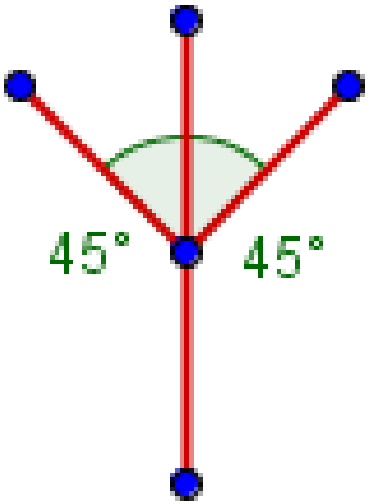
Tree

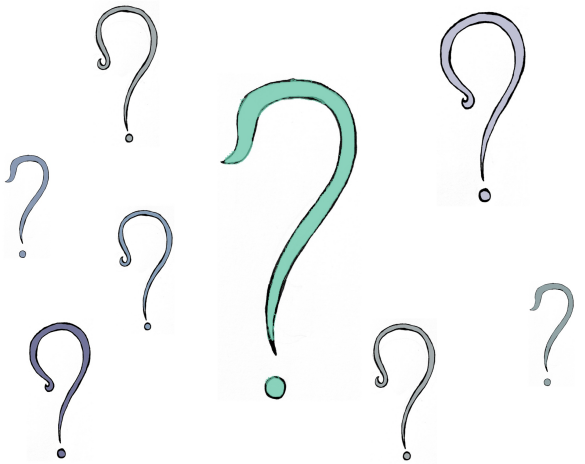
- What if we want to store the generated structure?
- For example, this smaller tree:



Tree

- What if we want to store the generated structure?
- For example, this smaller tree:
- We should specify the strucutre and the parameters (length, angle).





Formal Grammar (Chomsky)

- Formal grammar consists of:
  - Set of nonterminal symbols N.

Formal Grammar (Chomsky)

- Formal grammar consists of:
  - Set of nonterminal symbols N.

Nonterminals can be changed by production rules.

Formal Grammar (Chomsky)

- Formal grammar consists of:
  - Set of nonterminal symbols N.

Nonterminals can be changed by production rules.

They do not „terminate“ the derivation.

Formal Grammar (Chomsky)

- Formal grammar consists of:
  - Set of nonterminal symbols N.
  - Set of terminal symbols Σ .

Formal Grammar (Chomsky)

- Formal grammar consists of:
  - Set of nonterminal symbols N.
  - Set of terminal symbols Σ.

Terminals can not be changed by production rules.

Formal Grammar (Chomsky)

- Formal grammar consists of:
  - Set of nonterminal symbols N.
  - Set of terminal symbols Σ.

Terminals can not be changed by production rules.

They do „terminate“ the derivation.

Formal Grammar (Chomsky)

- Formal grammar consists of:
  - Set of nonterminal symbols N.
  - Set of terminal symbols Σ .
  - Set of production rules.

Formal Grammar (Chomsky)

- Formal grammar consists of:
  - Set of nonterminal symbols N.
  - Set of terminal symbols Σ .
  - Set of production rules.

Rules tell you what nonterminals can be replaced with other nonterminals or terminals.

Formal Grammar (Chomsky)

- Formal grammar consists of:
  - Set of nonterminal symbols N.
  - Set of terminal symbols Σ .
  - Set of production rules.
  - Starting axiom.

Formal Grammar (Chomsky)

- Formal grammar consists of:
  - Set of nonterminal symbols N.
  - Set of terminal symbols Σ .
  - Set of production rules.
  - Starting axiom.

The initial „word“ of symbols / system state.

Formal Grammar (Chomsky)

- Formal grammar consists of:
  - Set of nonterminal symbols N.
  - Set of terminal symbols Σ.
  - Set of production rules.
  - Starting axiom.
- Example:

$N = \{A\}$

Formal Grammar (Chomsky)

- Formal grammar consists of:
  - Set of nonterminal symbols N.
  - Set of terminal symbols Σ.
  - Set of production rules.
  - Starting axiom.
- Example:

$N = \{A\}$

$\Sigma = \{a\}$

Formal Grammar (Chomsky)

- Formal grammar consists of:
  - Set of nonterminal symbols N.
  - Set of terminal symbols Σ.
  - Set of production rules.
  - Starting axiom.
- Example:

$$N = \{A\}$$

$$\Sigma = \{a\}$$

$$R = \begin{Bmatrix} A \rightarrow AA \\ A \rightarrow a \end{Bmatrix}$$

Formal Grammar (Chomsky)

- Formal grammar consists of:
  - Set of nonterminal symbols N.
  - Set of terminal symbols Σ.
  - Set of production rules.
  - Starting axiom.
- Example:

$$N = \{A\} \qquad Axiom = A$$

$$\Sigma = \{a\}$$

$$R = \begin{Bmatrix} A \rightarrow AA \\ A \rightarrow a \end{Bmatrix}$$

Formal Grammar (Chomsky)

- Formal grammar consists of:
  - Set of nonterminal symbols N.
  - Set of terminal symbols Σ.
  - Set of production rules.
  - Starting axiom.
- Example:

$N = \{A\}$ $\quad Axiom = A$

$\Sigma = \{a\}$

$$R = \begin{Bmatrix} A \rightarrow AA \\ A \rightarrow a \end{Bmatrix}$$

Generates words

$A \rightarrow a$

Formal Grammar (Chomsky)

- Formal grammar consists of:
  - Set of nonterminal symbols N.
  - Set of terminal symbols Σ.
  - Set of production rules.
  - Starting axiom.
- Example:

$N = \{A\}$ $\quad Axiom = A$

$\Sigma = \{a\}$

$R = \begin{Bmatrix} A \rightarrow AA \\ A \rightarrow a \end{Bmatrix}$

Generates words

$A \rightarrow a$

$A \rightarrow AA \rightarrow aA \rightarrow aa$

Formal Grammar (Chomsky)

- Formal grammar consists of:
  - Set of nonterminal symbols N.
  - Set of terminal symbols Σ.
  - Set of production rules.
  - Starting axiom.
- Example:

$N = \{A\}$ $\quad Axiom = A$

$\Sigma = \{a\}$

$R = \begin{Bmatrix} A \rightarrow AA \\ A \rightarrow a \end{Bmatrix}$

Generates words

$A \rightarrow a$

$A \rightarrow AA \rightarrow aA \rightarrow aa$

$A \rightarrow AA \rightarrow AAA \rightarrow aAA \rightarrow aaA \rightarrow aaa$

Formal Grammar (Chomsky)

- Formal grammar consists of:
  - Set of nonterminal symbols N.
  - Set of terminal symbols Σ.
  - Set of production rules.
  - Starting axiom.
- Example:

$N = \{A\}$ $\quad Axiom = A$

$\Sigma = \{a\}$

$R = \begin{Bmatrix} A \rightarrow AA \\ A \rightarrow a \end{Bmatrix}$

Generates words

$A \rightarrow a$

$A \rightarrow AA \rightarrow aA \rightarrow aa$

$A \rightarrow AA \rightarrow AAA \rightarrow aAA \rightarrow aaA \rightarrow aaa$

...

Formal Grammar (Chomsky)

- Used for:

Formal Grammar (Chomsky)

- Used for:
  - Natural language processing

Formal Grammar (Chomsky)

- Used for:
  - Natural language processing
  - Program code processing (compiler, interpreter)

Formal Grammar (Chomsky)

- Used for:
  - Natural language processing
  - Program code processing (compiler, interpreter)
- Hierarchy of types
  - **Type 0: Unrestricted** – $N = \Sigma$

Formal Grammar (Chomsky)

- Used for:
  - Natural language processing
  - Program code processing (compiler, interpreter)
- Hierarchy of types
  - **Type 0: Unrestricted** – $N = \Sigma$
  - **Type 1: Context sensitive** – non-terminal symbol on the left side, can be surrounded by a context

Formal Grammar (Chomsky)

- Used for:
  - Natural language processing
  - Program code processing (compiler, interpreter)
- Hierarchy of types
  - **Type 0: Unrestricted** – $N = \Sigma$
  - **Type 1: Context sensitive** – non-terminal symbol on the left side, can be surrounded by a context
  - **Type 2: Context free** – left side contains only a single non-terminal symbol

Formal Grammar (Chomsky)

- Used for:
  - Natural language processing
  - Program code processing (compiler, interpreter)
- Hierarchy of types
  - **Type 0: Unrestricted** – $N = \Sigma$
  - **Type 1: Context sensitive** – non-terminal symbol on the left side, can be surrounded by a context
  - **Type 2: Context free** – left side contains only a single non-terminal symbol
  - **Type 3: Regular** – right side is empty, single terminal, or single terminal follower by non-terminal

Lindenmayer System

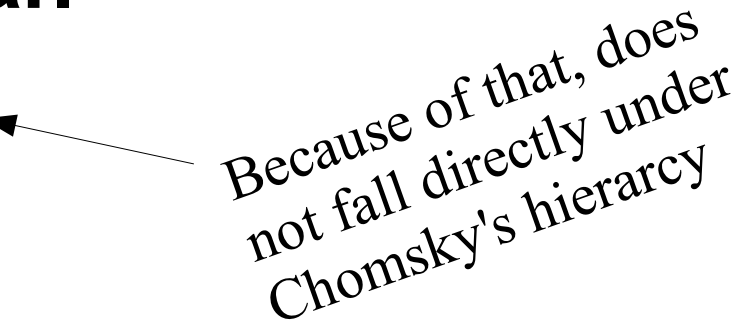
- **Variant of a formal grammar.**

Lindenmayer System

- **Variant of a formal grammar.**
- **Parallel rewriting system.**

Lindenmayer System

- **Variant of a formal grammar.**
- **Parallel rewriting system.**



Lindenmayer System

- **Variant of a formal grammar.**
- **Parallel rewriting system.**
- We will look at one, that is:
  - Bracketed system.

Because of that, does not fall directly under Chomsky's hierarcy

Lindenmayer System

- **Variant of a formal grammar.**
- **Parallel rewriting system.**
- We will look at one, that is:
  - Bracketed system.
  - Stochastic system.

Because of that, does not fall directly under Chomsky's hierarcy

Lindenmayer System

- Variant of a formal grammar.
- Parallel rewriting system.
- We will look at one, that is:
  - Bracketed system.
  - Stochastic system.
  - Context free (0L-system).

Because of that, does not fall directly under Chomsky's hierarcy

Lindenmayer System

- Variant of a formal grammar.
- Parallel rewriting system.
- We will look at one, that is:
  - Bracketed system.
  - Stochastic system.
  - Context free (0L-system).
  - Parametric system.

Because of that, does not fall directly under Chomsky's hierarcy

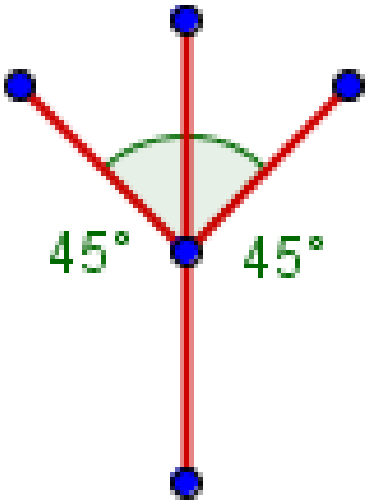
Lindenmayer System

- **Bracketed system** – we use brackets to indicate branches.

Lindenmayer System

- **Bracketed system** – we use brackets to indicate branches.
- Using following symbols:

Symbol	Meaning
F	Segment
+	Rotate left 45°
-	Rotate right 45°
[	Start of a branch
]	End of a branch



Can we write our tree using those?

Lindenmayer System

- **Parallel rewriting system** – all the rules will be applied in parallel to rewrite the entire word.

Lindenmayer System

- **Parallel rewriting system** – all the rules will be applied in parallel to rewrite the entire word.

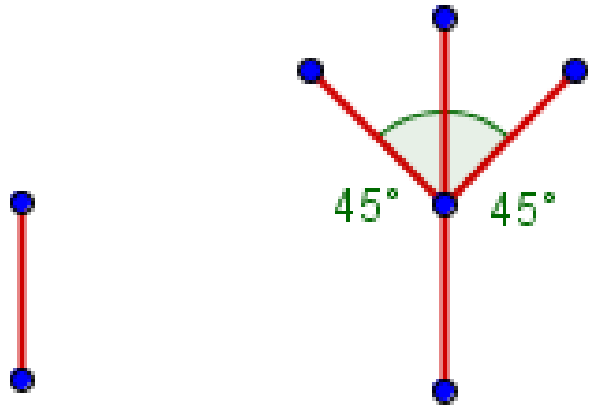
What would be the rules to create the following?

Axiom: F

Lindenmayer System

- **Parallel rewriting system** – all the rules will be applied in parallel to rewrite the entire word.

What would be the rules to create the following?

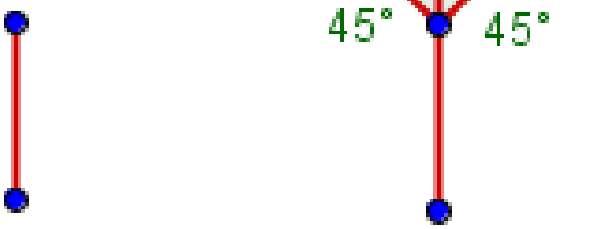


Axiom: F 1. iteration: F[+F][-F]F

Lindenmayer System

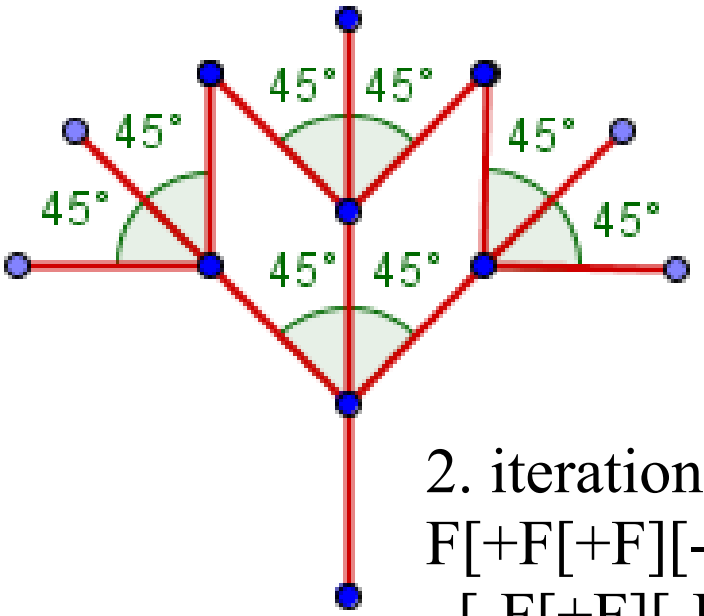
- **Parallel rewriting system** – all the rules will be applied in parallel to rewrite the entire word.

What would be the rules to create the following?



Axiom: F

1. iteration: F[+F][-F]F



2. iteration:
F[+F[+F][-F]F]
[-F[+F][-F]F]
F[+F][-F]F

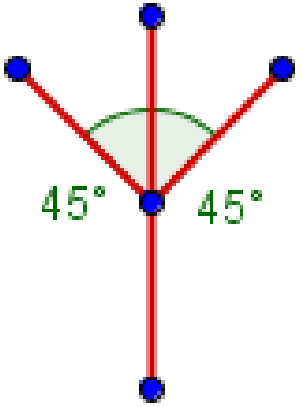
Lindenmayer System

- **Parallel rewriting system** – all the rules will be applied in parallel to rewrite the entire word.

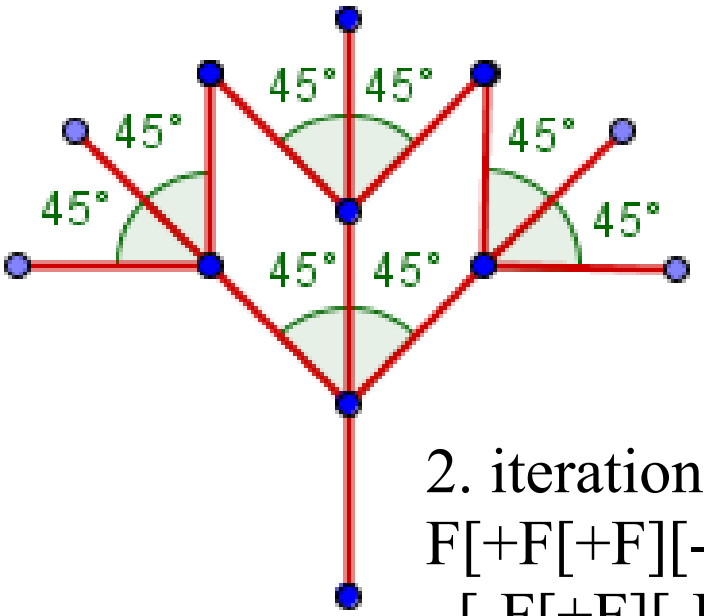
What would be the rules to create the following?

This is a trick question.

Axiom: F



1. iteration: F[+F][-F]F



2. iteration:
F[+F[+F][-F]F]
[-F[+F][-F]F]
F[+F][-F]F

Lindenmayer System

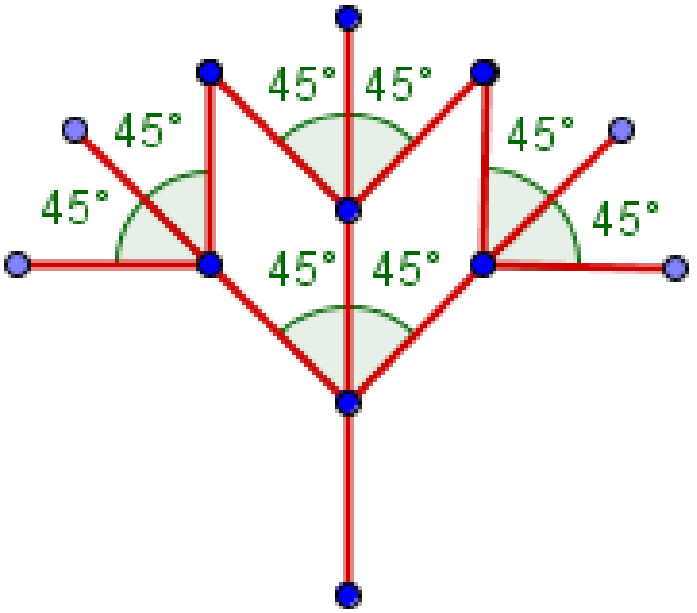
- **Parametric system** – we can specify parameters for some of the symbols.

Lindenmayer System

- **Parametric system** – we can specify parameters for some of the symbols.
  - The length, the angle etc

Lindenmayer System

- **Parametric system** – we can specify parameters for some of the symbols.
  - The length, the angle etc

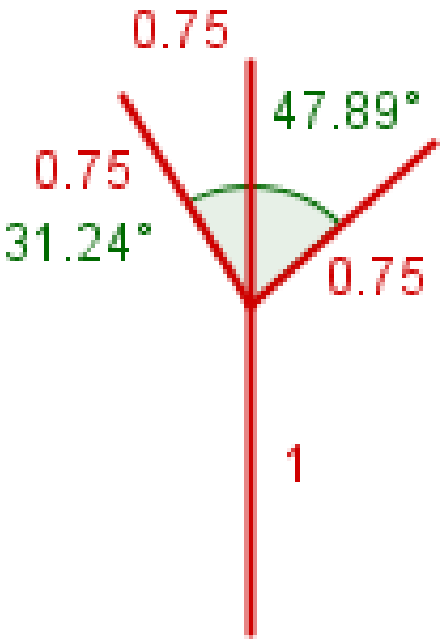


F[+(45)F[+(45)F][-(45)F]F]
[-(45)F[+(45)F][-(45)F]F]
F[+(45)F][-(45)F]F

Every + or - is followed by the angle of rotation.

Lindenmayer System

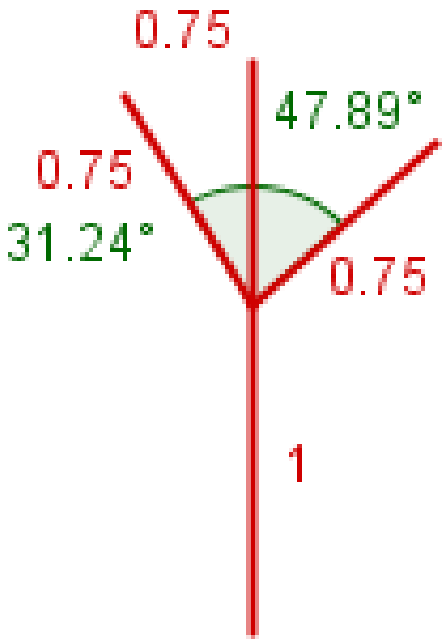
- We can generate **angles** with some variance.



F[+**(31.24)**F][-**(47.89)**F]F

Lindenmayer System

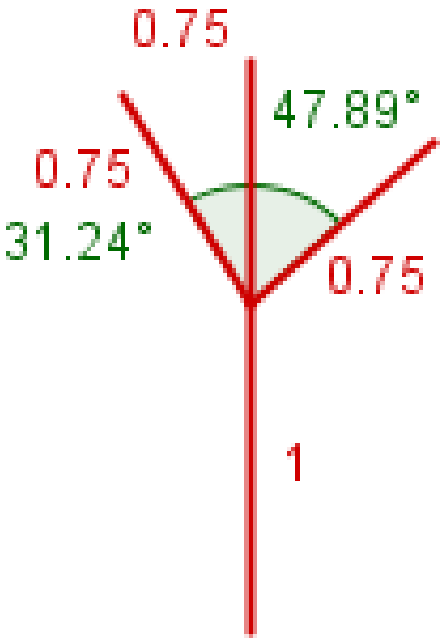
- We can generate **angles** with some variance.
- Also specify the **lengths** of the segments.



F(1)[+(31.24)**F(0.75)**][-(47.89)**F(0.75)**]**F(0.75)**

Lindenmayer System

- We can generate **angles** with some variance.
- Also specify the **lengths** of the segments.



If the decrease of lengths is deterministic, we could consider it only, when drawing the tree...

F(1)[+(31.24)**F(0.75)**][-(47.89)**F(0.75)**]**F(0.75)**

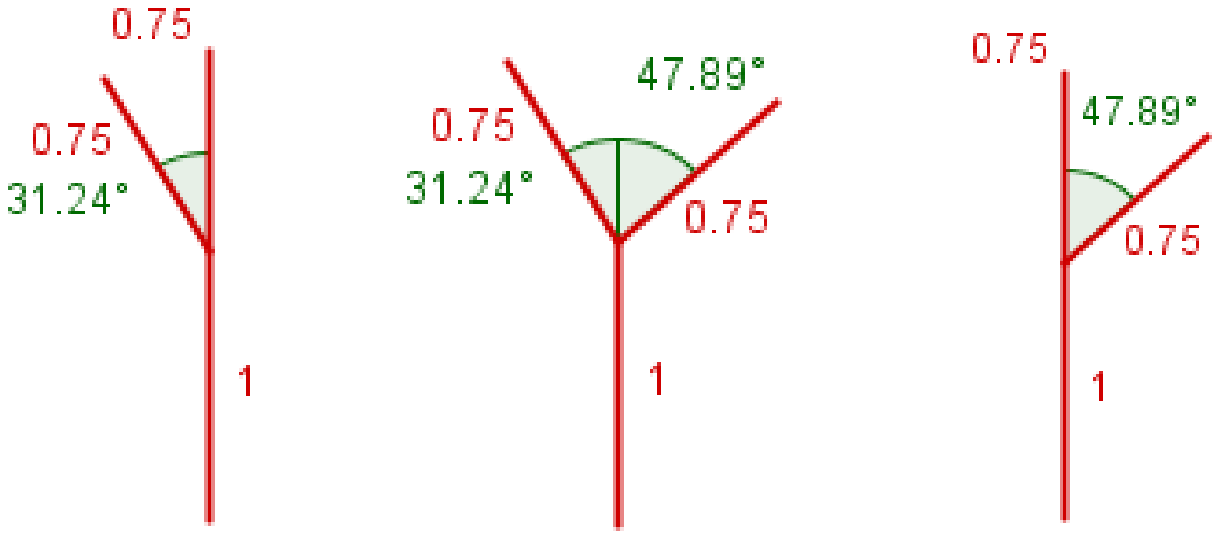
Lindenmayer System

- **Stochastic system** – we can have many rules, with the same left-hand side.

$A \rightarrow F[+A]A$

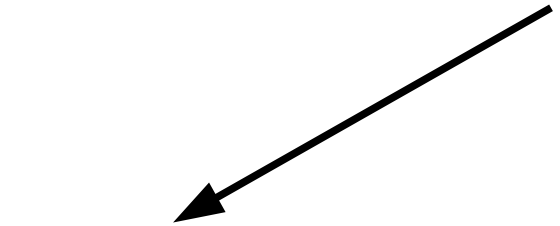
$A \rightarrow F[-A]A$

$A \rightarrow F[+A][-A]$



Lindenmayer System

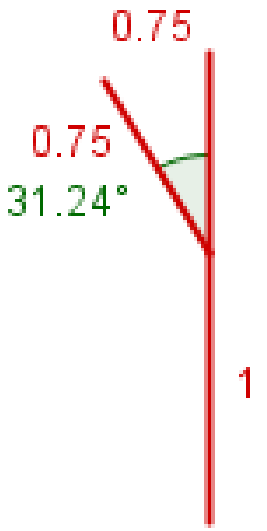
- **Stochastic system** – we can have many rules, with the same left-hand side.
- Each rule has a probability.

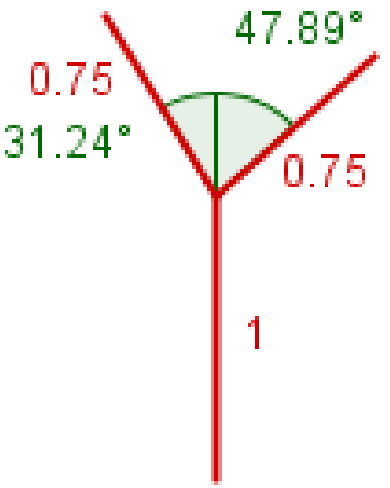


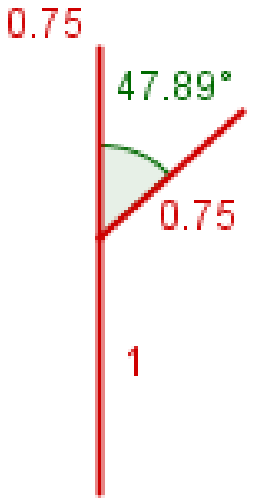
$$A \xrightarrow{1/3} F[+A]A$$

$$A \xrightarrow{1/3} F[-A]A$$

$$A \xrightarrow{1/3} F[+A][-A]$$







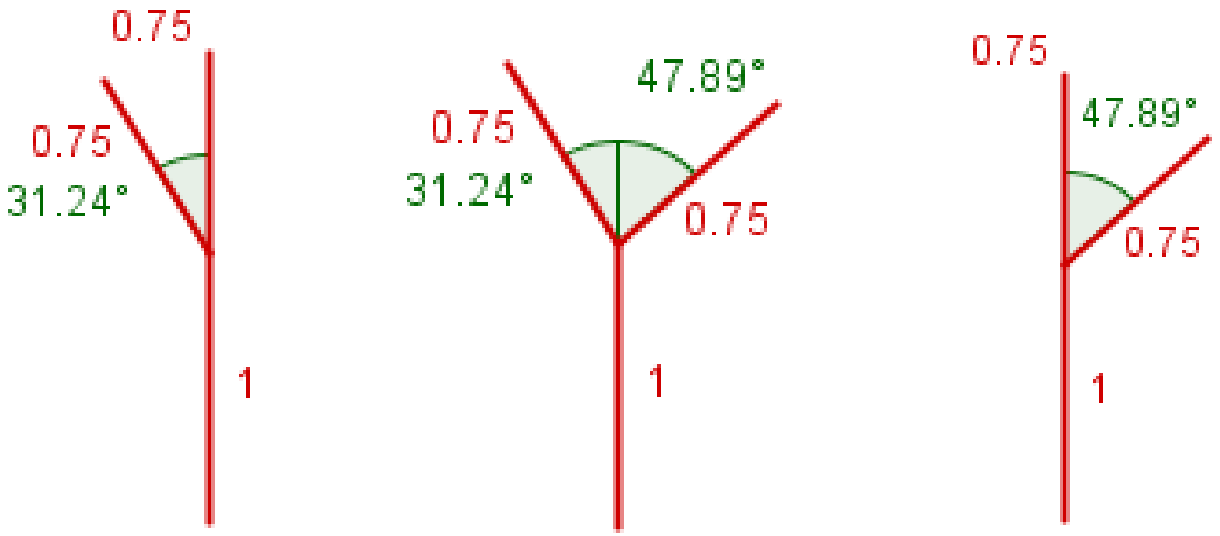
Lindenmayer System

- **Stochastic system** – we can have many rules, with the same left-hand side.
- Each rule has a probability.
- The **sum** of the probabilities of all the rules, with the same left-hand side, **has to be 1**.

$$A \xrightarrow{1/3} F[+A]A$$

$$A \xrightarrow{1/3} F[-A]A$$

$$A \xrightarrow{1/3} F[+A][-A]$$



Lindenmayer System

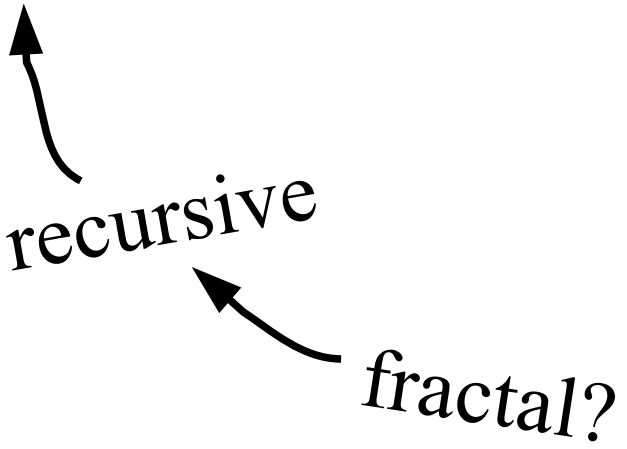
- Rigorous way to specify a mechanism for a **self-similar** structure generation.

Lindenmayer System

- Rigorous way to specify a mechanism for a **self-similar** structure generation.

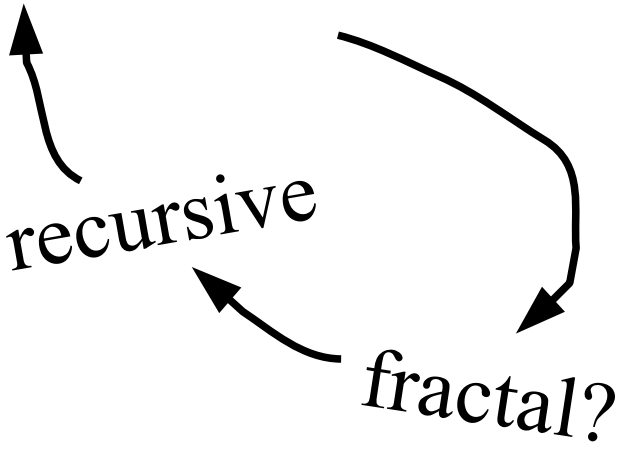
Lindenmayer System

- Rigorous way to specify a mechanism for a **self-similar** structure generation.



Lindenmayer System

- Rigorous way to specify a mechanism for a **self-similar** structure generation.



Lindenmayer System

- Rigorous way to specify a mechanism for a **self-similar** structure generation.
- Lot of research and different possibilities.

Lindenmayer System

- Rigorous way to specify a mechanism for a **self-similar** structure generation.
- Lot of research and different possibilities.
- *The Algorithmic Beauty of Plants*, A. Lindenmayer, P. Prusinkiewicz. http://algorithmicbotany.org/papers/abop/abop.pdf

Lindenmayer System

- Rigorous way to specify a mechanism for a **self-similar** structure generation.
- Lot of research and different possibilities.
- *The Algorithmic Beauty of Plants*, A. Lindenmayer, P. Prusinkiewicz. http://algorithmicbotany.org/papers/abop/abop.pdf
- Try out 2D online: http://www.kevs3d.co.uk/dev/lsystems/

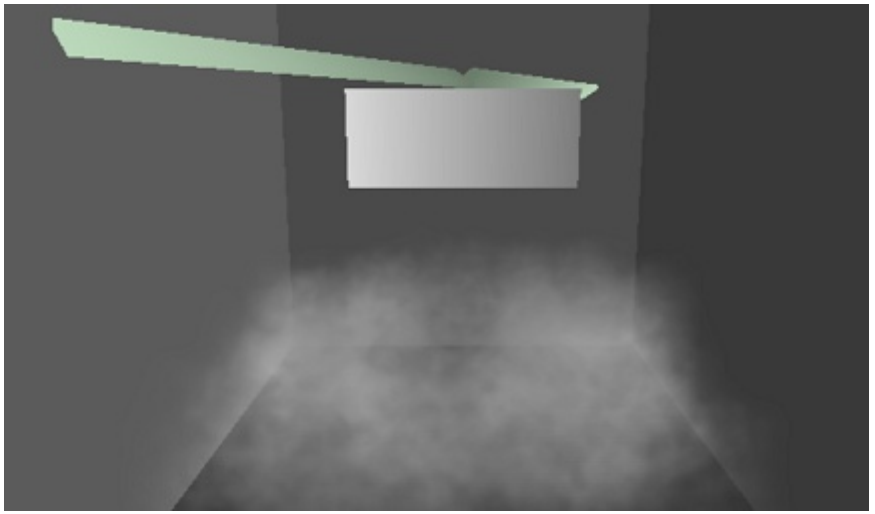
Lindenmayer System

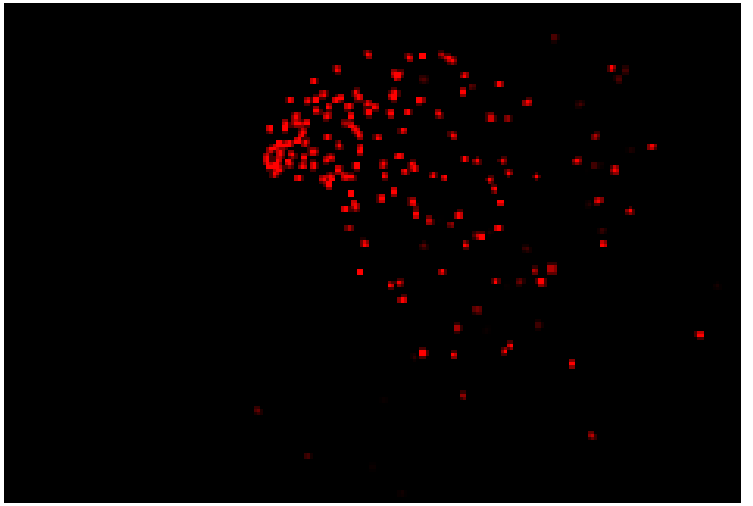
- Rigorous way to specify a mechanism for a **self-similar** structure generation.
- Lot of research and different possibilities.
- *The Algorithmic Beauty of Plants*, A. Lindenmayer, P. Prusinkiewicz. http://algorithmicbotany.org/papers/abop/abop.pdf
- Try out 2D online: http://www.kevs3d.co.uk/dev/lsystems/
- Questions?

Particle System

- Used for different effects
  - Fire, fluid, wind, smoke
  - Precipitation (rain, snow)
  - Groups of objects with behaviour (birds, NPC-s)

This you did in the Soft Particle Chopper.





Particle System

- Particles can have a transparency that varies over time.

Particle System

- Particles can have a transparency that varies over time.
- Particles can be generated from an object pool.
  - If a particle dies, return it to the object pool.

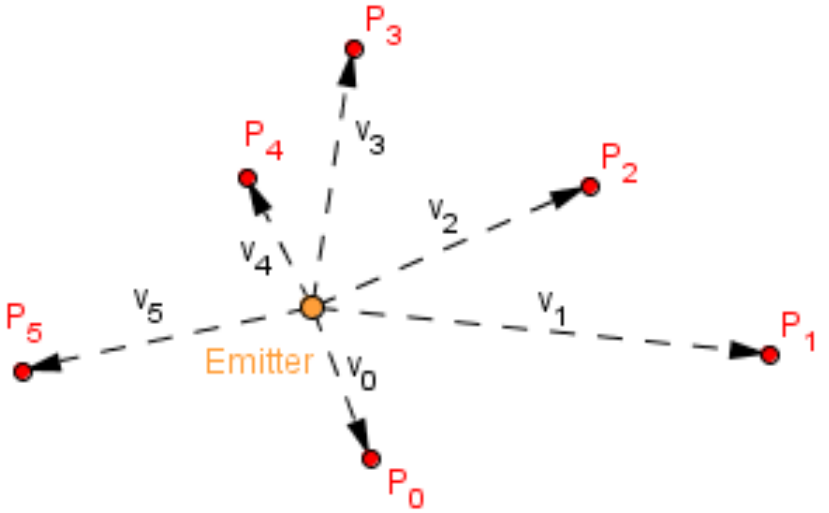
Particle System

- Particles can have a transparency that varies over time.
- Particles can be generated from an object pool.
  - If a particle dies, return it to the object pool.
- Particle can be 1 pixel in size, or have an image.

Particle System

- Particles can have a transparency that varies over time.
- Particles can be generated from an object pool.
  - If a particle dies, return it to the object pool.
- Particle can be 1 pixel in size, or have an image.
- Particle system has an emmitter of particles.

Emitter can also be a line,
a surface, a volume etc.

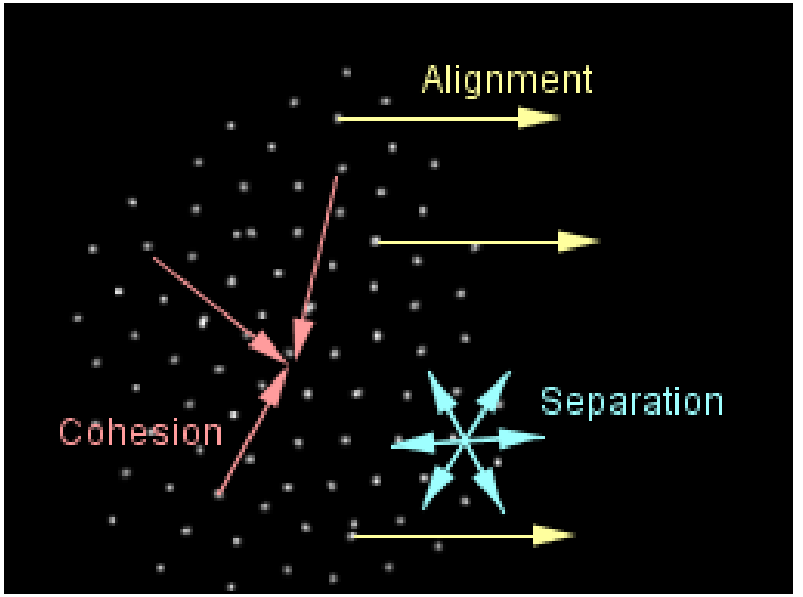


Boids Algorithm

- Used to model flocking (*eg* of birds).

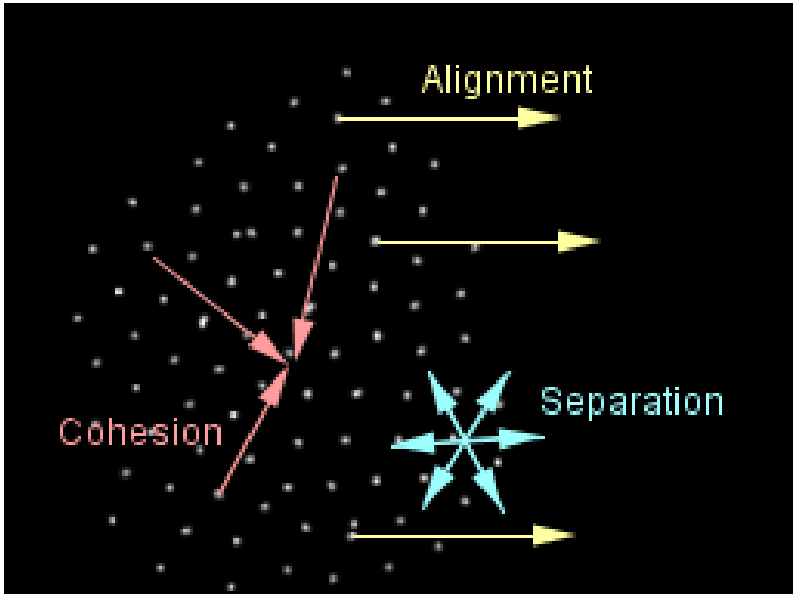
Boids Algorithm

- Used to model flocking (*eg* of birds).
- Each particle follows a set of rules:
  - **Cohesion** – Move towards the center of mass.



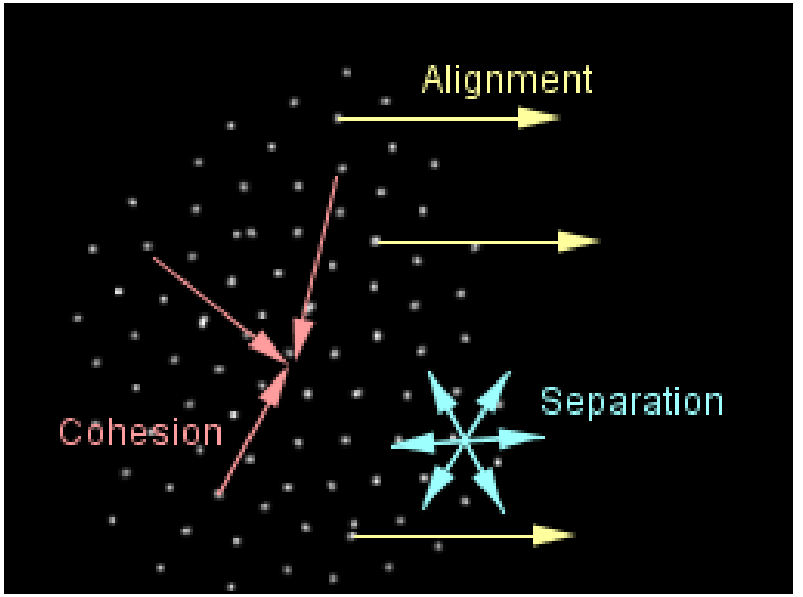
Boids Algorithm

- Used to model flocking (*eg* of birds).
- Each particle follows a set of rules:
  - **Cohesion** – Move towards the center of mass.
  - **Separation** – Keep distance from other particles.



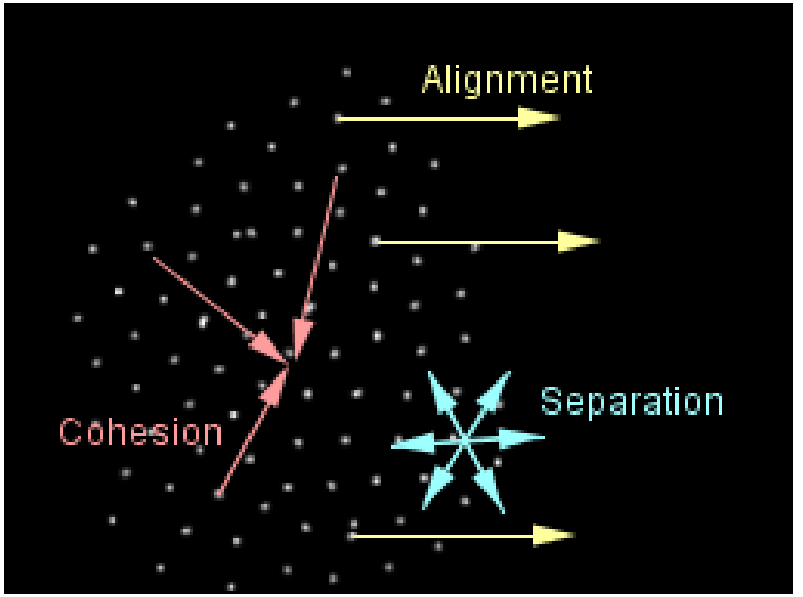
Boids Algorithm

- Used to model flocking (*eg* of birds).
- Each particle follows a set of rules:
  - **Cohesion** – Move towards the center of mass.
  - **Separation** – Keep distance from other particles.
  - **Alignment** – Follow the average direction.



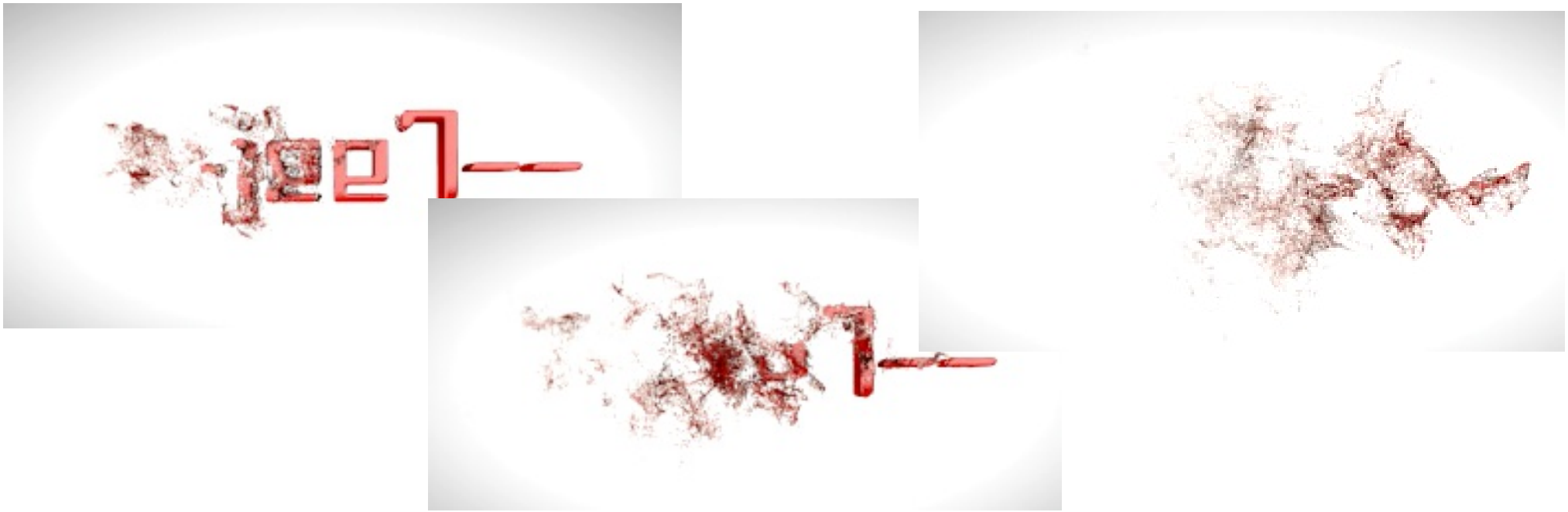
Boids Algorithm

- Used to model flocking (*eg* of birds).
- Each particle follows a set of rules:
  - **Cohesion** – Move towards the center of mass.
  - **Separation** – Keep distance from other particles.
  - **Alignment** – Follow the average direction.
- There can be other rules.



Particle Systems

- Blender has particle systems



- Example of scar generation via particles: https://www.youtube.com/watch?v=e3FpG3CFlfQ

What was new for you today?

What more would you like to know?

Next time: Ray Casting, Ray Tracing,
Space Partitioning, BVH