UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

Physically Inspired Generation of Plant Environments

Karl Kraus

Fortgeschrittenen Praktikum
Universität Heidelberg
Heidelberg, den 4. Februar 2019

VISUAL
COMPUTING
GROUP
HEIDELBERG UNIVERSITY

Motivation

Source: commons.wikimedia.org

Rainforest in Australia

Trees generatedy by our algorithm

Outline

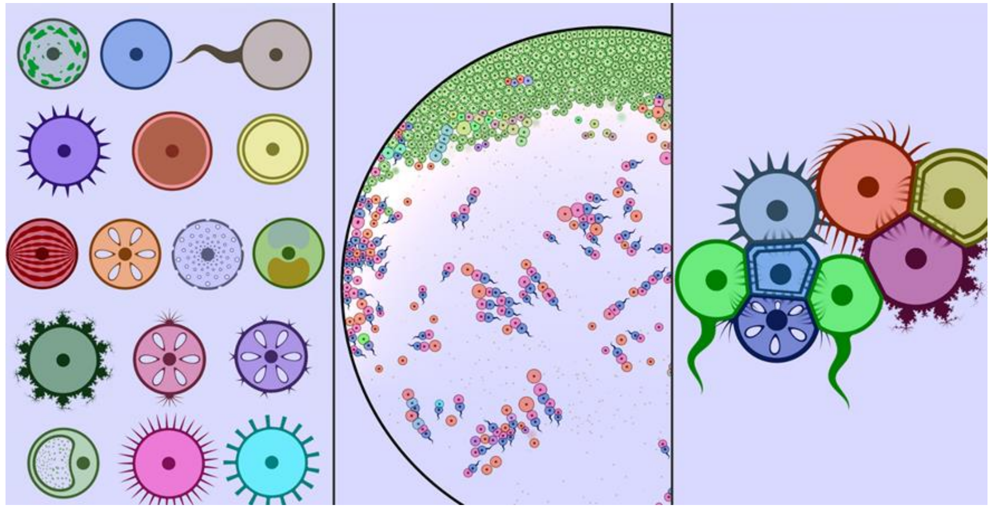
1. Related Work
2. Basic Concept
3. Environmental Influences
4. Result

Related Work: Simulated Ecosystems

- Many games have design elements with some sort of ecological influence
- Yet they usually mimic high-level behavior by description
- Our goal instead is to set low-level rules with many degrees of freedom
- Real-life-like behavior should emerge from these basic rules

Source: minecraft.gamepedia.com/

Minecraft

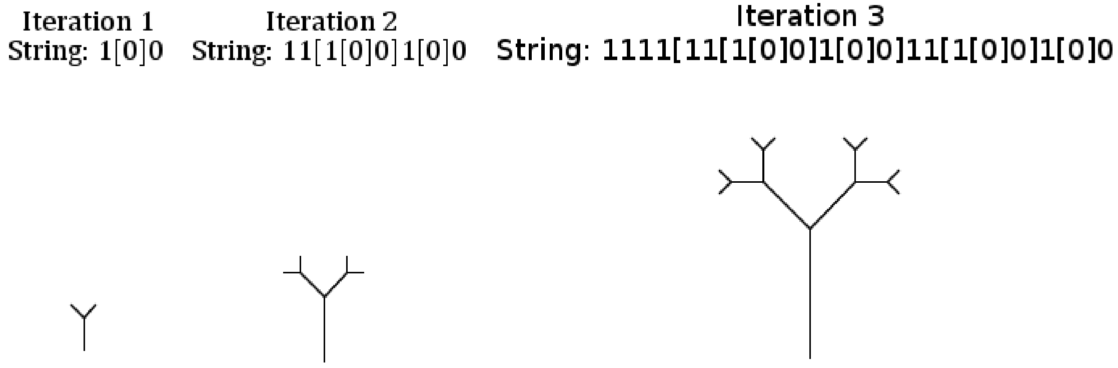


Source: cell-lab.net/

CellLab

Related Work: Procedural Plant Generation

- Procedural plant generation has great commercial relevance (e.g., SpeedTree)
- State of the art for generation are Lindenmayer systems (L-systems)
- L-systems are a string-manipulating system combined with simple drawing rules
- They are an excellent choice to express self-similarity
- Drawbacks: L-systems are unfit for
  - Low-level randomization
  - (Co)-development of different plants based on 3D environments
- Thus, the formal constaint of L-Systems to represent plant shapes by strings was dropped



Example of simple L-system with strings

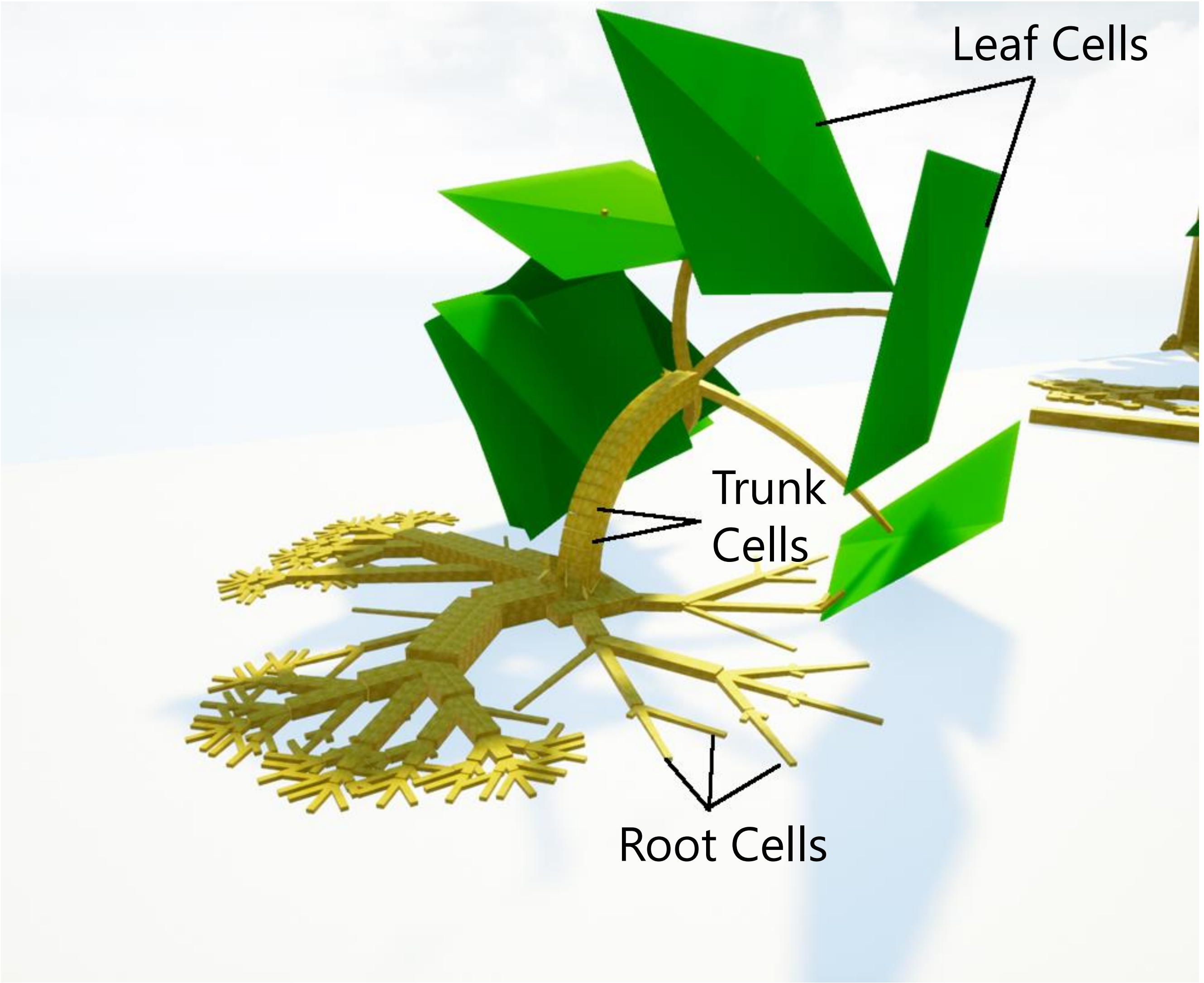
Source: en.wikipedia.org/

Source: commons.wikimedia.org/

Results from advanced L-systems

Basic Concept

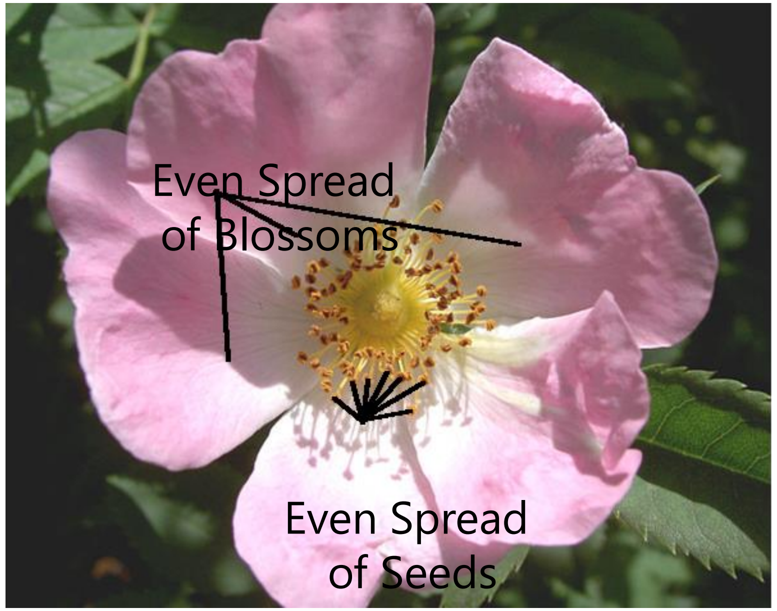
- Basis of the approach are cells with physical properties and their 3D representation
- An organism is the set of its cells
- Cells have types (e.g., “leaf”)
- Growth is modeled by cell division into cells of identical or different type
- Every cell (except the root) has an “attachement parent”, and the child’s position is calculated relatively to the parent



Plant and cell types example

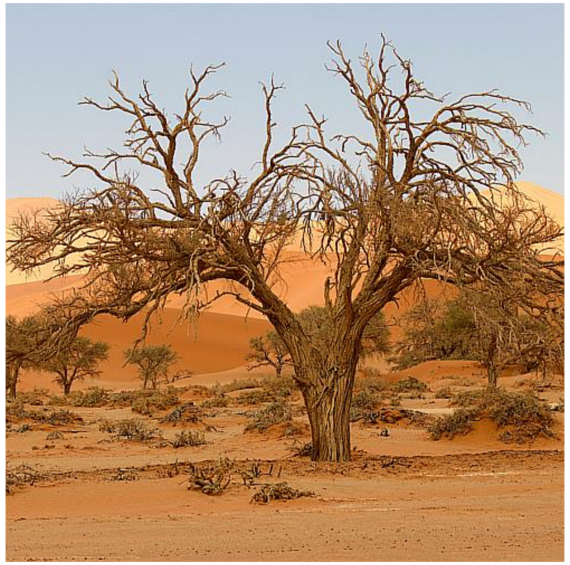
Even Spread and Self-Similarity I

- Many plant shapes are describable by even spreads with certain degrees of freedom, centered on an origin
- In general, this freedom is radially symmetric in growth direction
- Tree branching can be described by even spread of a small number of branches
- A tree is then a concatenation of branchings (self-similarity)
- Plant structures as different as rose blossoms, dandeilion seed heads and trees can be described by this concept



Source: commons.wikimedia.rog/

Different even spreads in a rose



Source: commons.wikimedia.org/

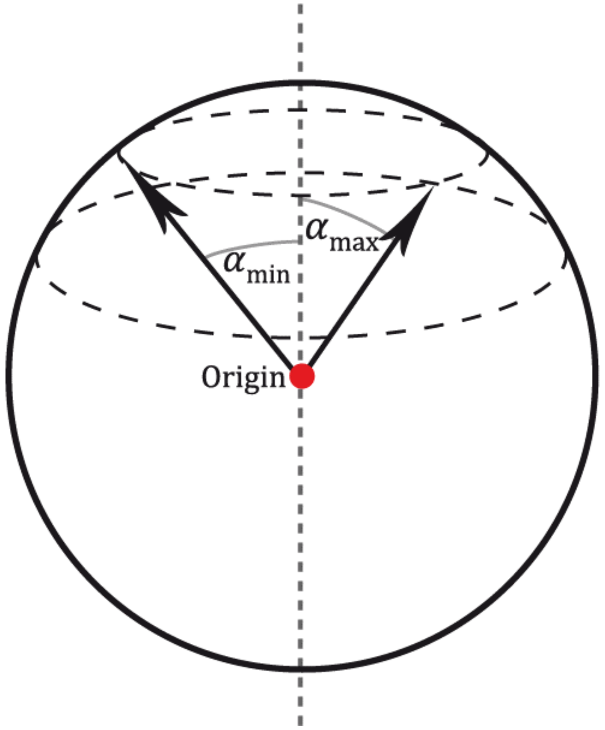
Biological tree branching

Source: commons.wikimedia.org/

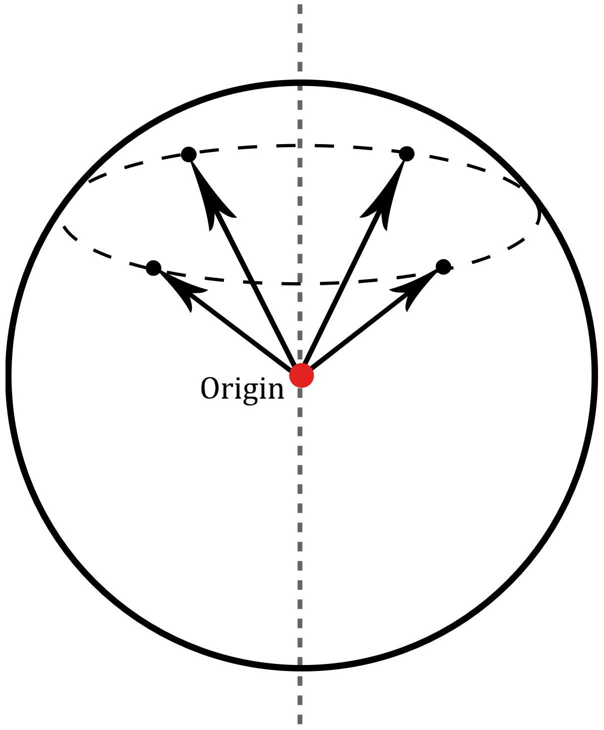
Dandelion

Even Spread and Self-Similarity II

- Mathematically, spread freedom can be defined by $\alpha_{\min}$ and $\alpha_{\max}$ (the allowed deviation in growth direction)
- The growth objects then take positions with maximal distance to each other
- For punctiform origins, the two angles represent two circles on a sphere
  - Allowed directions: vectors from the origin to points on the surface of the sphere between the circles
- E.g., for blossom leaves is $\alpha_{max} - \alpha_{min} \approx 0$



Visualization of allowed freedom



Even spread of four points ($\alpha_{\min} = \alpha_{\max}$)

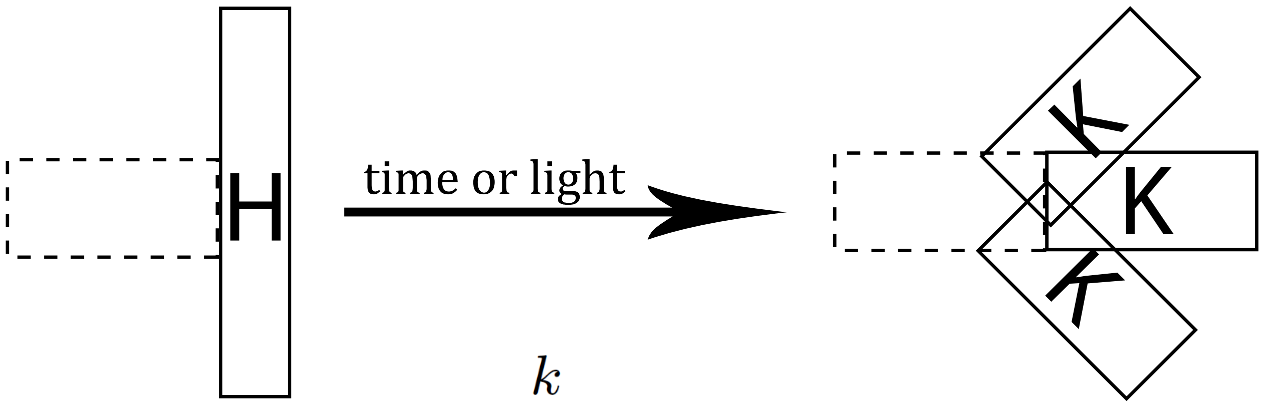
Division Rules: Vertical Division

$$F \xrightarrow{\text{time}} , F, F \quad \text{(vertical)}$$

$$K \xrightarrow{\text{time}} , F, H \quad \text{(vertical)}$$

- The basic definition scheme can be described by three cell types *F*, *K* and *H*
- Vertical division: length only growth of the organism
- There are multiple *F* states, so that plants grow faster farther away from the trunk origin
- Division of a cell with state F*n* results in two cells with state F*n+1* (capped at a maximum)

Division Rules: Horizontal Division

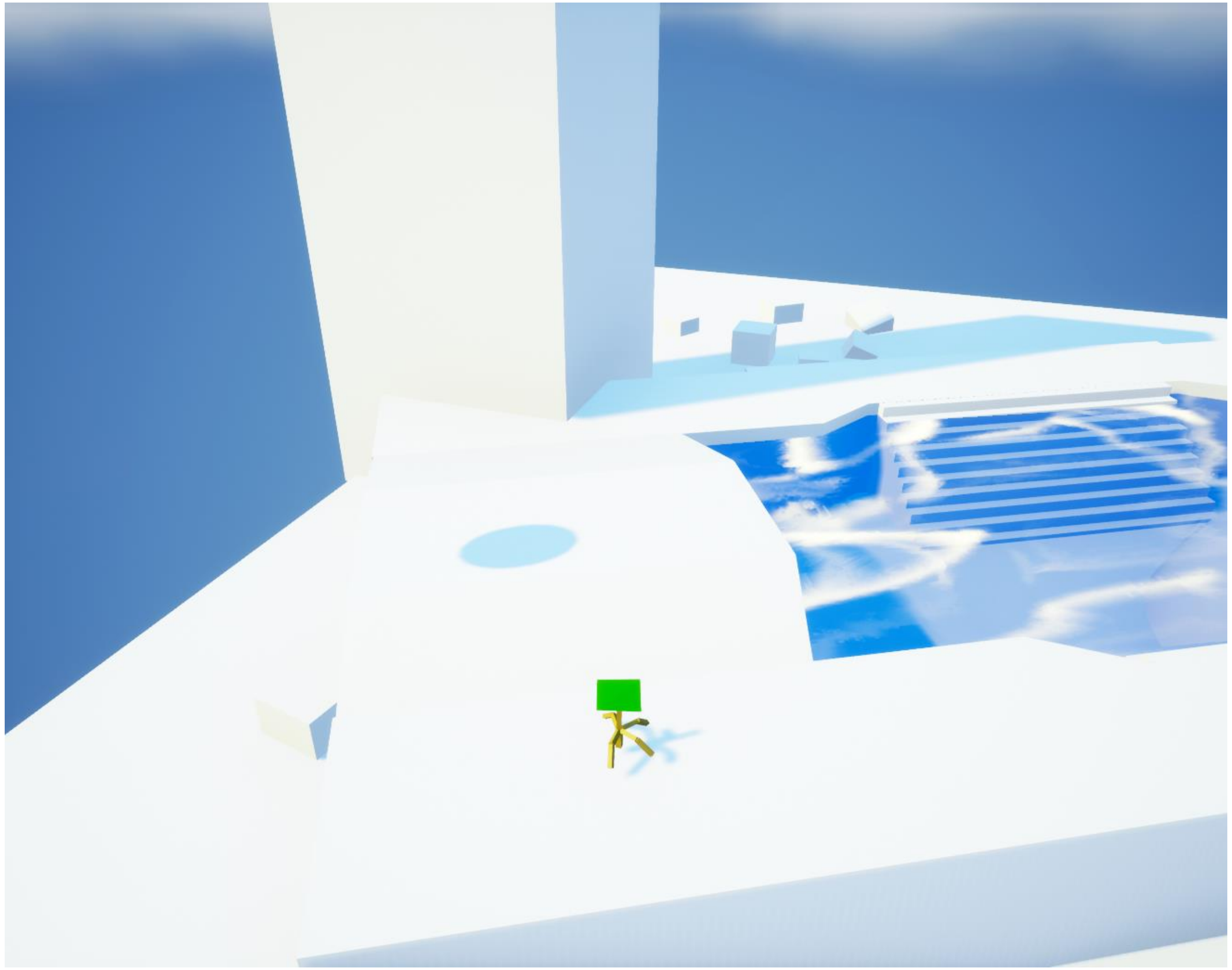


$$H \xrightarrow{\text{light}}, \overbrace{K, \ldots, K}^{k} \quad \text{(horizontal)}$$

$$H \xrightarrow{\text{time}}, \overbrace{K, \ldots, K}^{k'} \quad \text{(horizontal)}$$

- Horizontal division: creation of a fork by an even spread
- For non-ground cells, *H* are the leaf cells; for roots, *H* have the regular visual representation
- *H* divides into k cells when hit by light, or into k' cells after a given time
- In general: $k > k'$ such that plants grow "towards the light"

Plant Growth Example

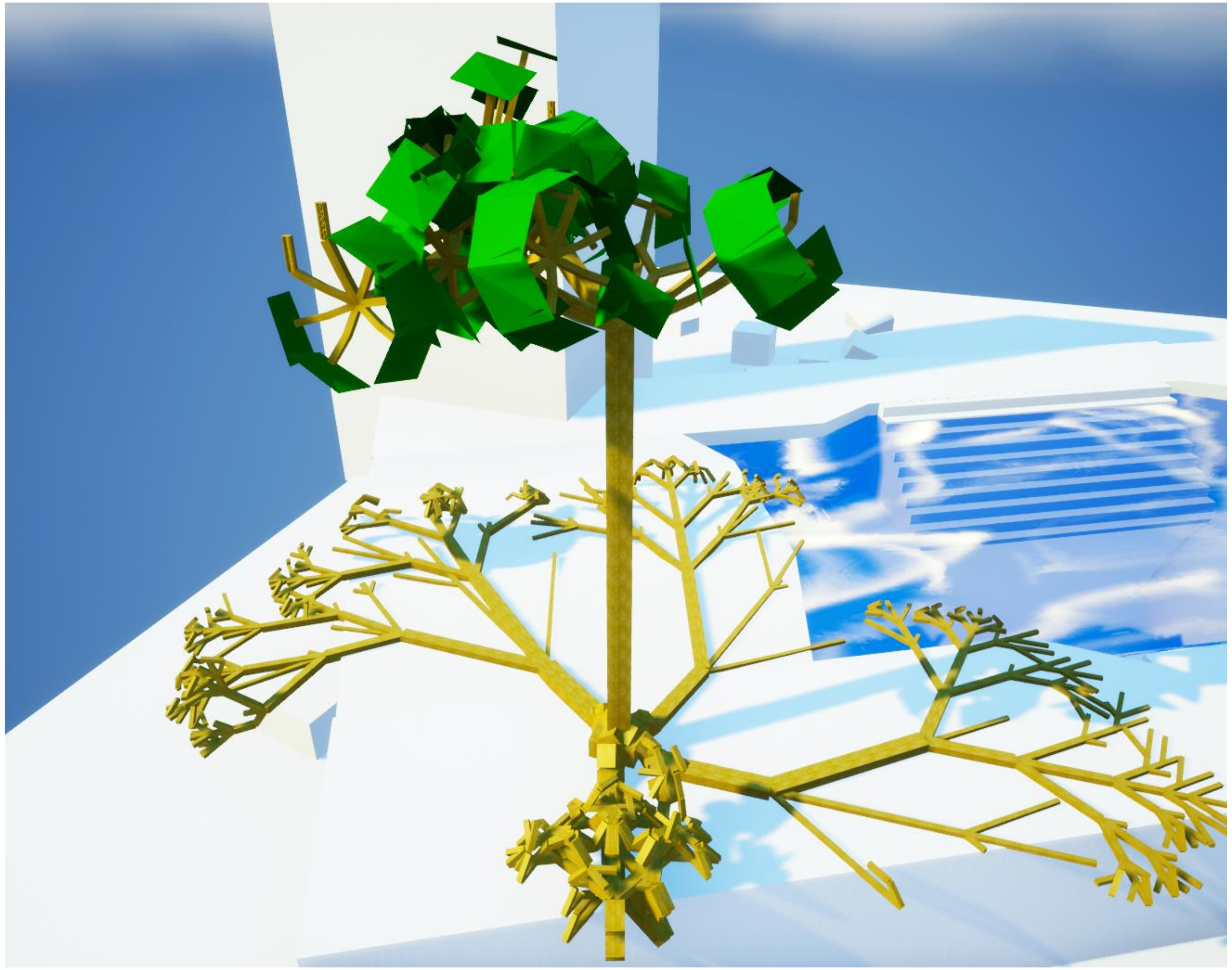


5 Iterations

Plant Growth Example

20 Iterations

Plant Growth Example



35 Iterations

Implementation: Unreal Engine

- The approach was implemented in Unreal Engine 4 (UE 4, UE)
- UE was originally designed for game development
- Offers a strong framework with many useful features
  - Efficient rendering
  - Collision detection
  - Physics simulation
- Most of the code was implemented in C++ for performance reasons
- Blueprints (visual scripting language of UE) was used occasionally

UNREAL ENGINE

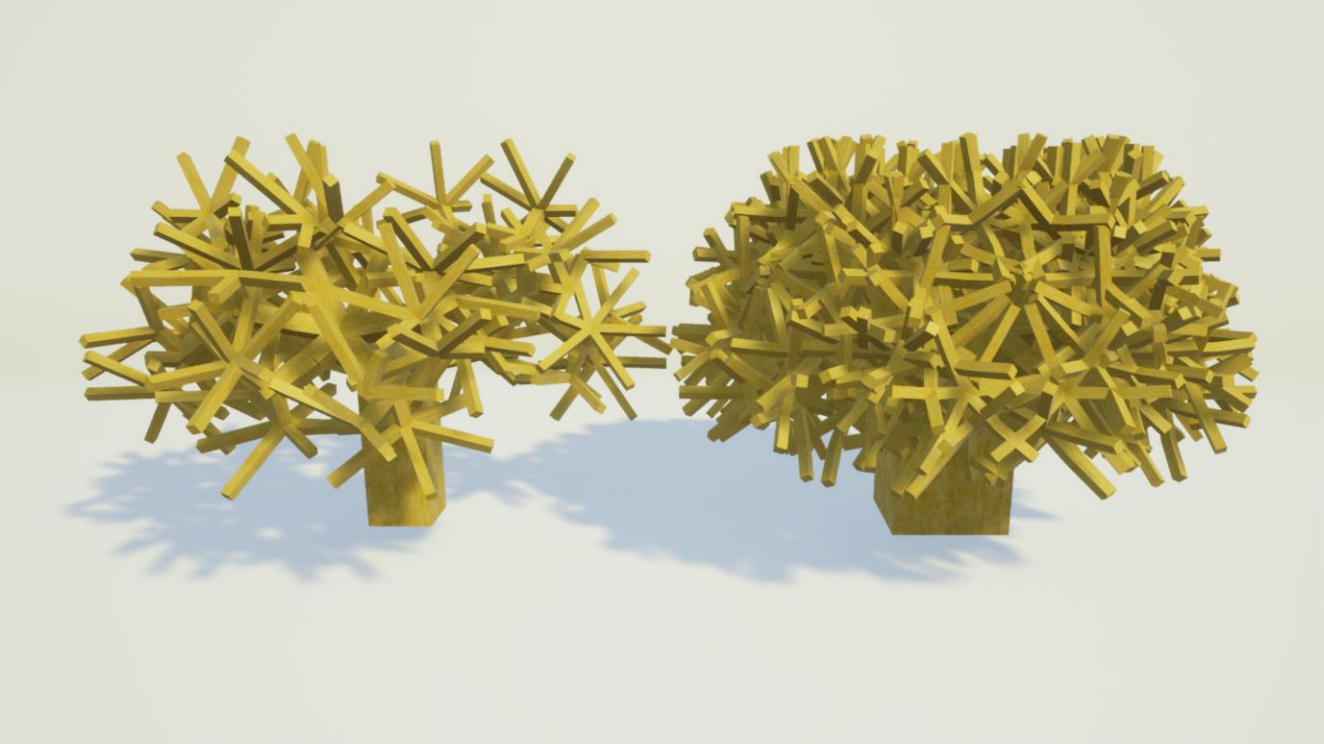
Source: unrealengine.com/

Modeled Behaviors

I. Plant growth is collision checked
II. Plants grow towards light
III. Water promotes tree growth
IV. Cells that carry more weight have a larger diameter
V. Growth can be influenced by gravity
VI. Roots grow along the ground
VII. Wind/storm can destroy parts of plants

I. Self-Collision

- In reality, branches cannot intersect
- Cells without children check for possbile collisions, and do not divide if one is found
- This greatly reduces, but does not completely prevent intersection
- Additionally adds deterministic, yet seemingly unpredictable structure



Left: Self-collision enabled; Right: Self-collision disabled

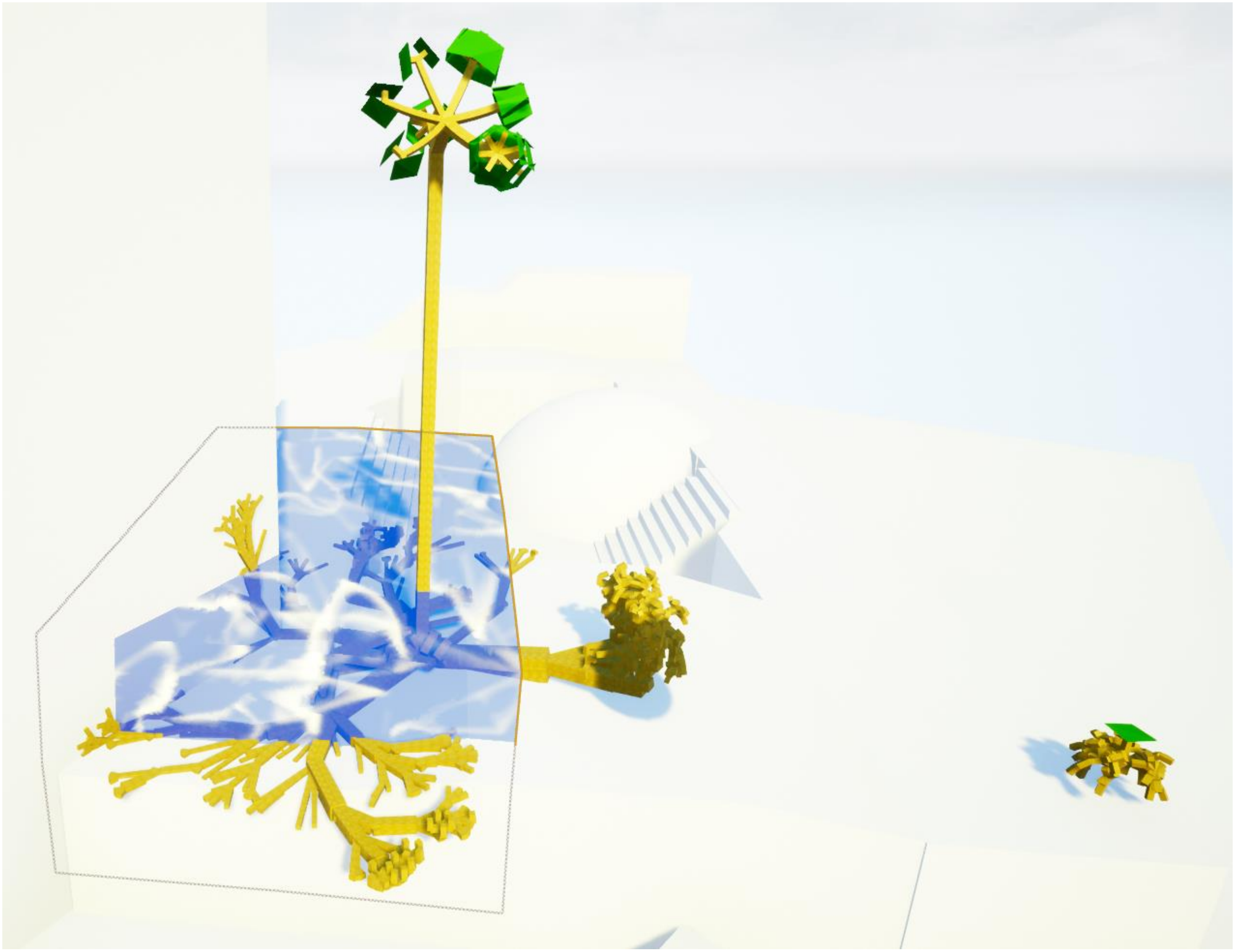
II. Growth to Light

- Plants grow "towards the light", as $k > k'$
- Precisely, leafs divide in less time and into more child cells if they are hit by light
- Thus, trees illuminated by light have more branches
- The more light hits a tree, the more cells it is allowed to have
- Trees that are hit by much light have smaller leafs (weigthed with a property)

Identical tree in light and penumbra

III. Influence of Water

- Plants with access to water grow more
- Water is not necessary, but promotes growth to diversify the vegetation
- In each iteration it is checked which plant cells overlap water
  - Therewith, the allowed total amount of cells for the correspdonding plants is raised



Identical tree with and without access to water

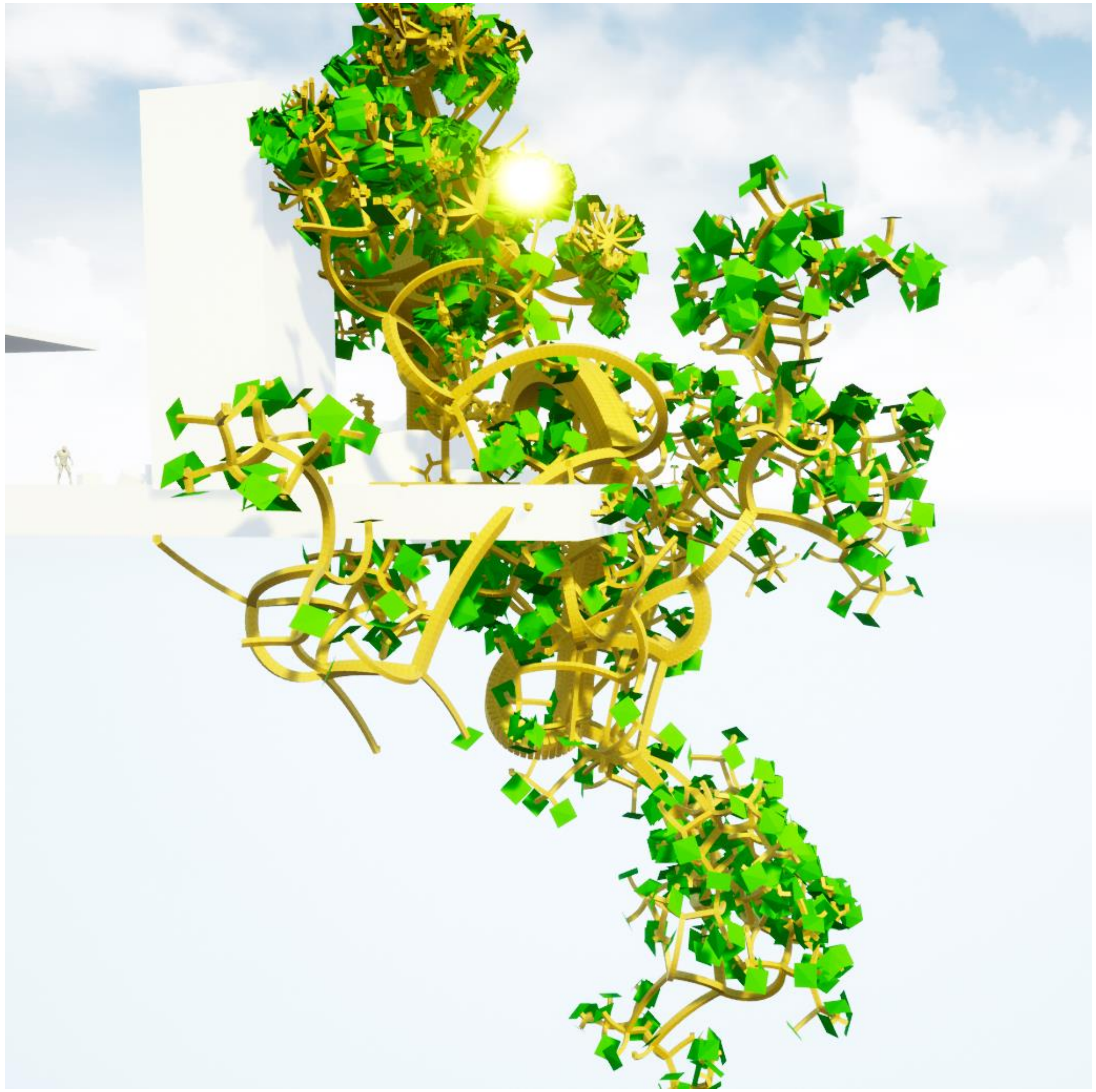
IV. Cell Diameter Growth

- Real-life trees are bigger where they have to carry a lot of weight (e.g., at the trunk)
- Cell diameter increases with the number of “attachement descendants” they have

Identical tree with different shapes close to light and in water
(also an example for positive correlation with gravity)

V. Influence of Gravity

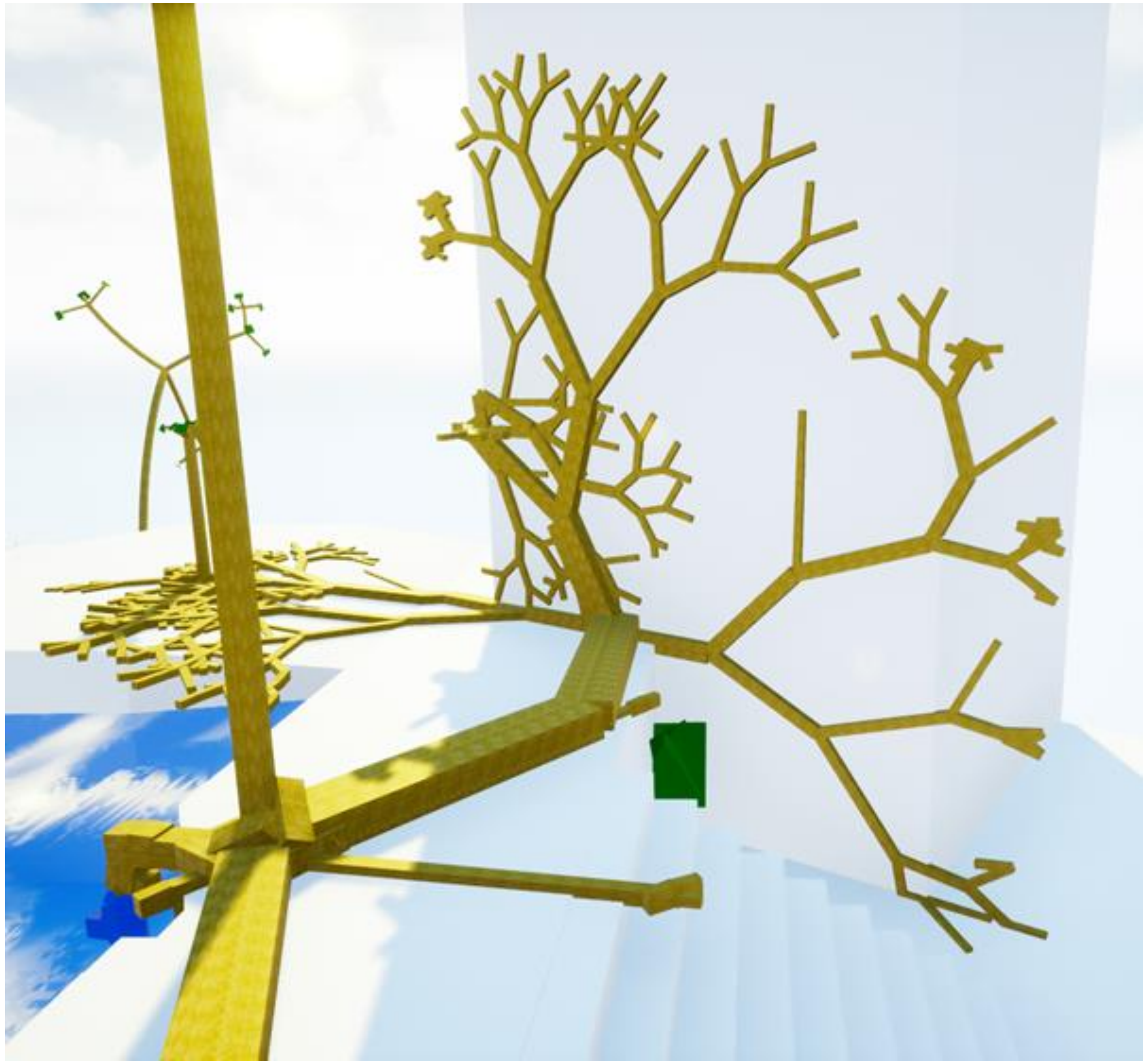
- Negative correlation with gravity leads to trees “bending over”
- Many plants, e.g. palms, ferns and willows, show this behavior
- The twisted structure of the shown plant emerges as a combination of downward growth and horizontal cell division
- This is also especially prone to provoke mistakes in collision detection



Massive generated tree

VI. Growth in a Plane

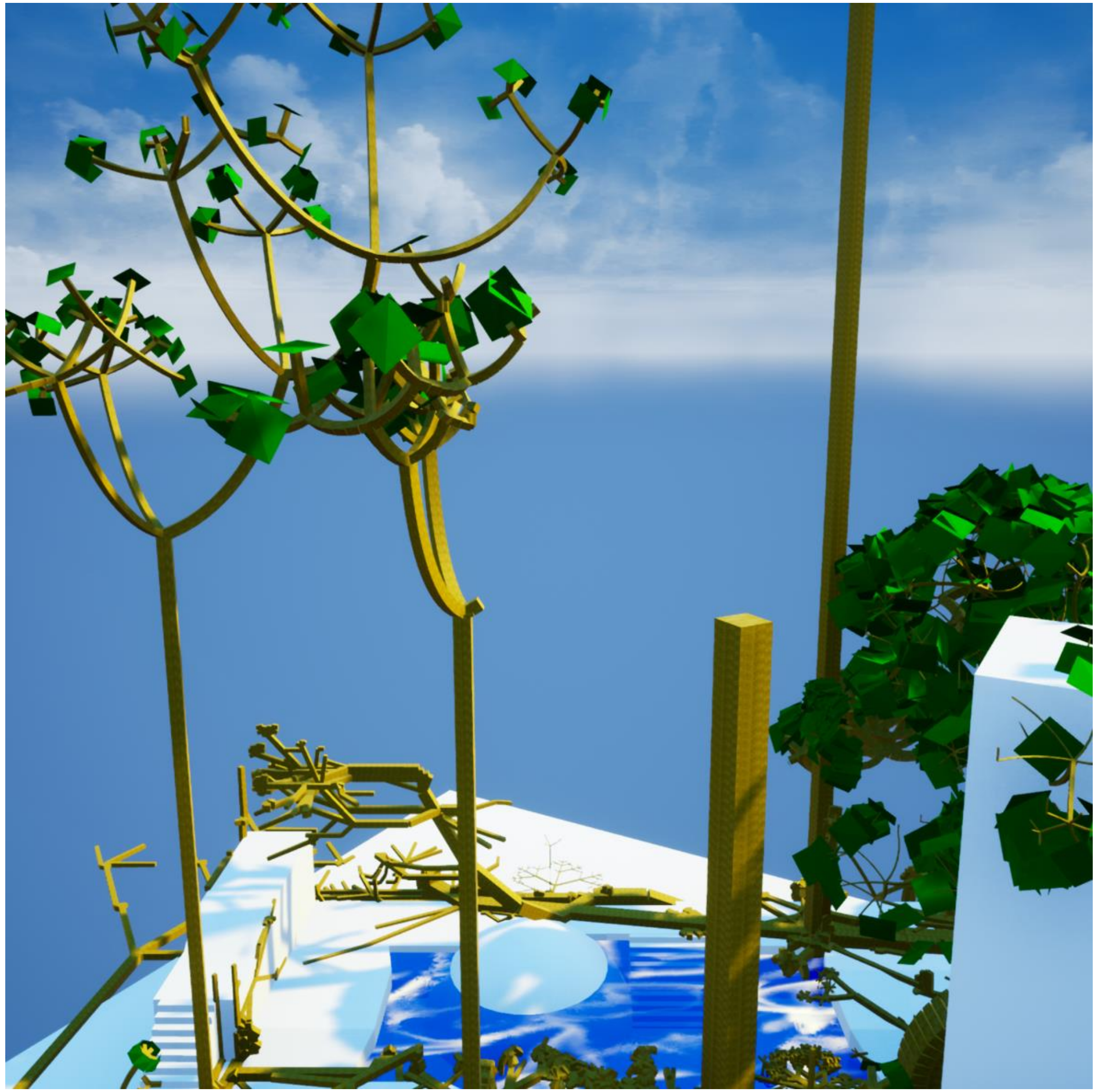
- Roots are an integral part of plants, visually and biologically
- Simulating growth in the earth is complicated, and not visually interesting
- Thus, we grow roots in a plane (the ground) which even spread is limited to
- Thus, we constrain growth and even spread of roots to the plane of the ground
- If roots hit an obstacle, a new plane is calculated from the face normal of the hit surface



Roots growing on the floor and up a wall

VII. Wind

- Wind is another important influence that defines biological growth of plants
- This was modelled by UE-`Raytraces` that put "wind weight" on hit cells, destroying them if over a threshold
- Initially the system was designed to make damaged branches fall down by gravity
- Unfortunately, Unreal does not support physics for non-static `InstancedStaticMeshes`



Example scene with cells destroyed by wind

Comparison: Tree-Growth

- The tree in the front (*A*) is close to the light
- The tree in the back (*B*) has access to water
- *A* reaches max cell count earlier, and builds a broader tree crown
- B grows higher

10 Iterations

Comparison: Tree-Growth

- The tree in the front (*A*) is close to the light
- The tree in the back (*B*) has access to water
- *A* reaches max cell count earlier, and builds a broader tree crown
- *B* grows higher

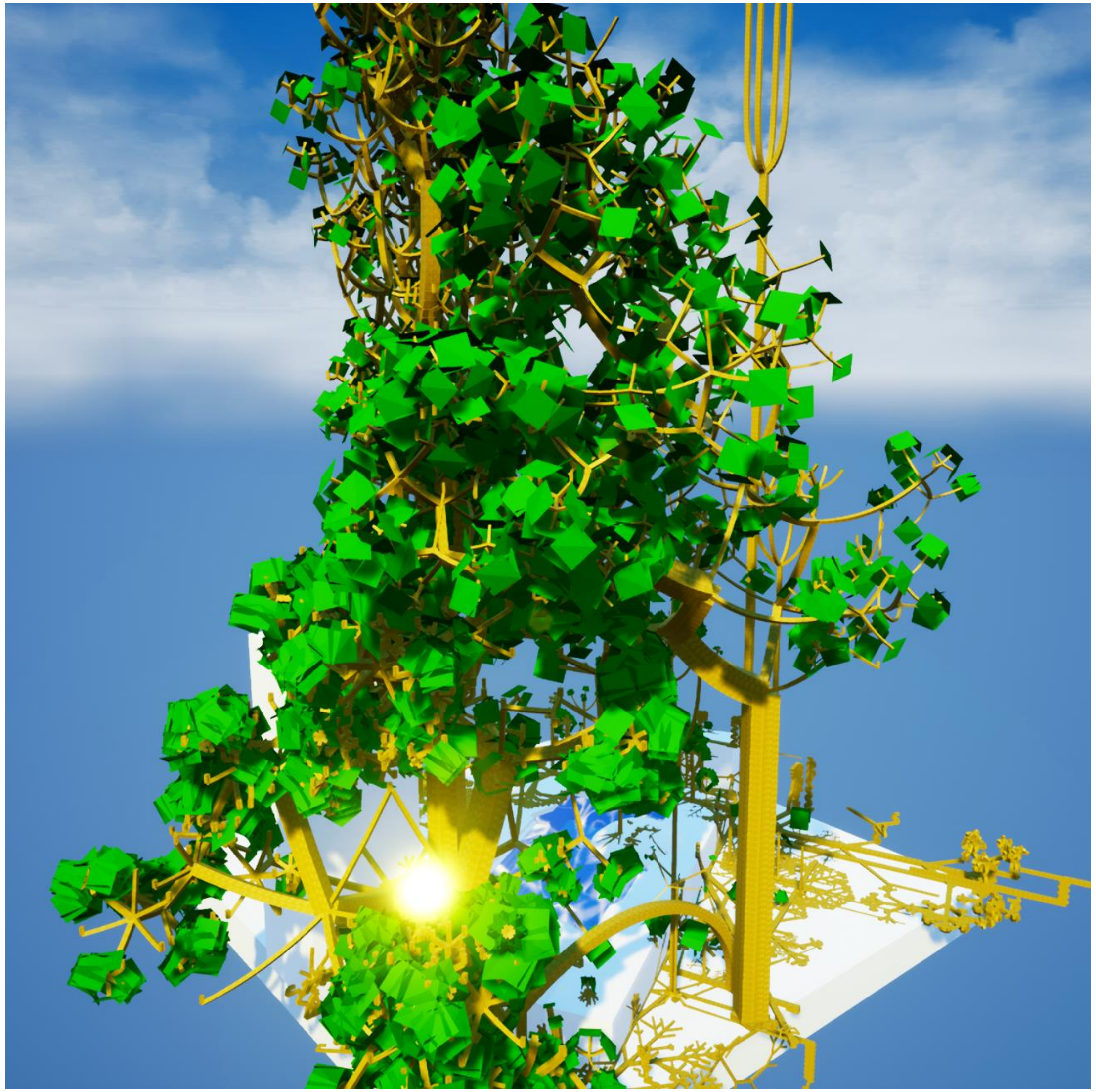
30 Iterations

Comparison: Tree-Growth

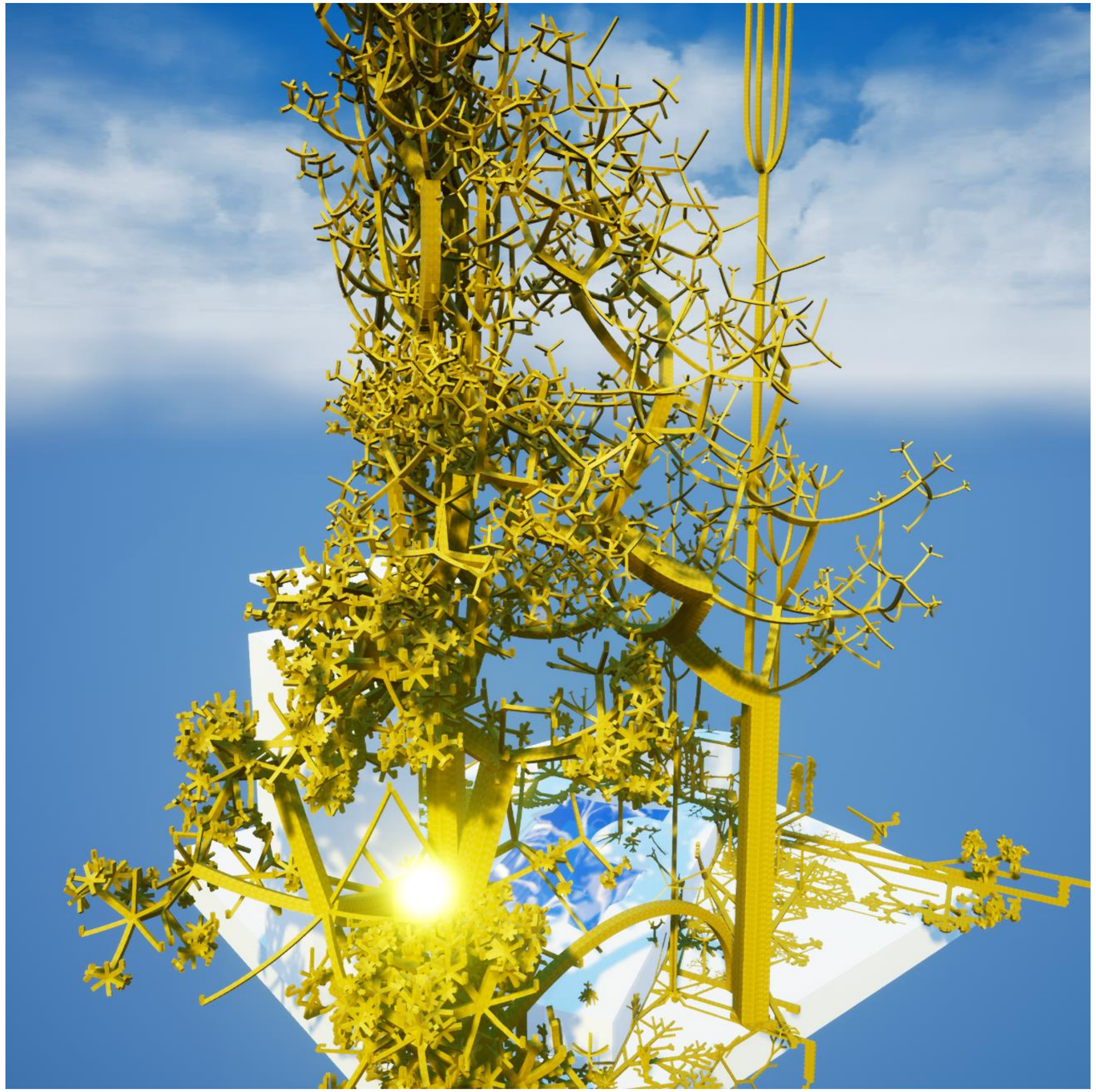
- The tree in the front (*A*) is closer to the light
- The tree in the back (*B*) has access to water
- *A* reaches max cell count earlier, and builds a broader tree crown
- *B* grows higher

45 Iterations

Example of Complex Branching Structure



Regular rendering



Rendering without leafs

Program Features and Controls

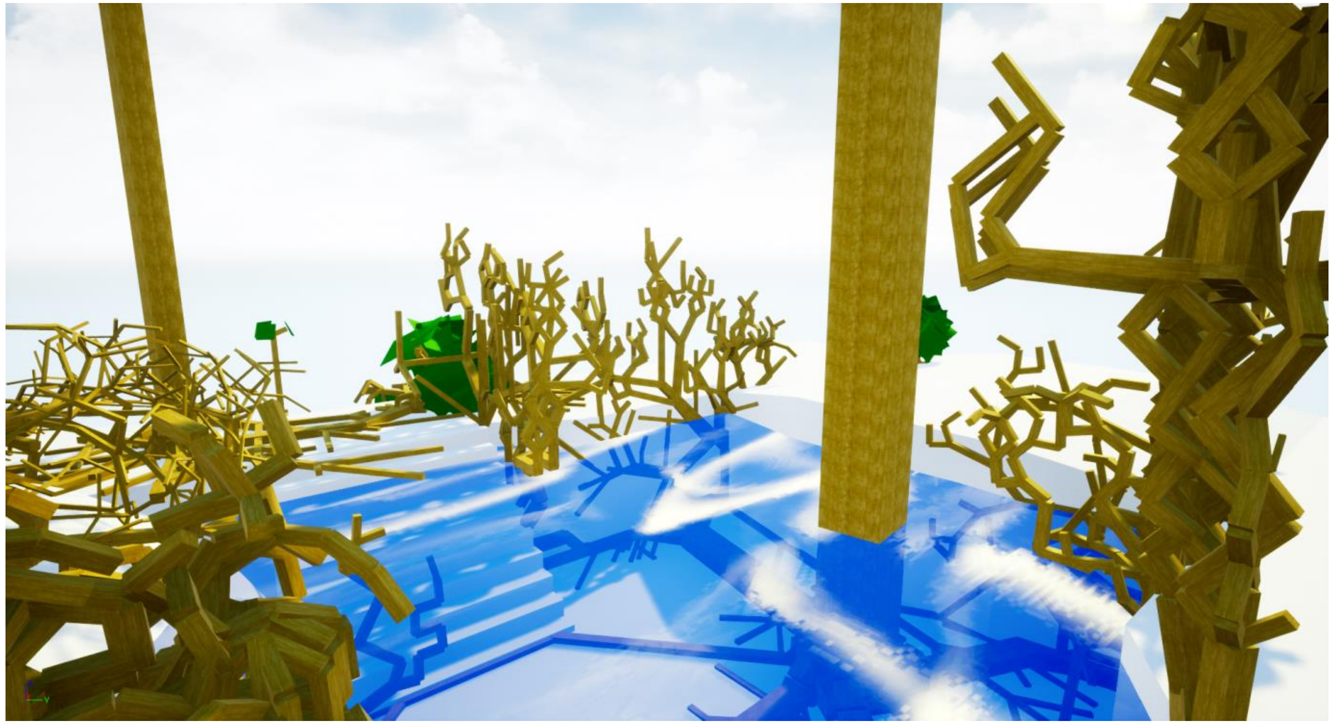
- One (game) level where exact values for a plant can be set
- One (game) level that generates many plants at once (a “forest”)
- Character control by industry-standard `WASD`/mouse
- Jumping and flying by `Space`
- `I` regenerates the single tree, and adds ten iterations for the forest
- `M` switches levels
- `P` toggles leaf rendering
- `O` re-randomizes the tree and resets the forest

Performance: Iteration Subtask Times

	Run A	Run B
Meshes Calculation & Setup	109.137 ms	17.924 ms
Cell Division in Data Structure	51.654 ms	4.699 ms
Raytrace for Light	19.551 ms	15.450 ms
Raytrace for Wind	8.910 ms	9.611 ms
Weight & Wind Burden	2.146 ms	4.769 ms
Water Influence	0.016 ms	1.064 ms
Total Iteration Time	191.473 ms	53.595 ms

- GPU: Nvidia GTX 760
- CPU: Intel i5-4670K 3.40GHz
- When not growth iterating, the program always hits 60 fps
- Iteration time and distribution on subtasks varies greatly

Impressions: Roots



Impressions II: With Player Character

Conclusion

- In this project, we presented a novel approach to abstract plant growth
- Based on
  - Even spreads
  - Environmental influences
  - Interactions between cells in a plant
  - Interactions between different plants
- Our method allows for low-level randomization
- The goal of unexpected and interesting behavior emergence worked out

Future Work

Simulated Ecosystems

- Our work constitutes a basis to simulate ecosystems
- For example, further work could include nutrient circulation
- This would require periodic plant death, and decomposers to return nutritients
- Completely differently structured organisms („animals“) could be added

Even Spread and Self-Similarity

- The possbilities of the plant description model were only scratched on the surface
- Current algorithms yield unsatisfactory results for non-circular degrees of freedom
- Other plantal objects (e.g., blossoms) could be mimiced with the model
- The model could also be extended to allow non-punctiform origins (e.g., for pine cone shaped behavior)

Thank you for your attention. Any Questions?

References

Literature

- Johan Knutzen, “Generating Climbing Plants Using L-Systems”

Games

- Minecraft Wiki: “Tree”: minecraft.gamepedia.com/Tree, viewed on 01/13/19
- Wikipedia: “Spore”: en.wikipedia.org/wiki/Spore_(2008_video_game), viewed on 01/13/19
- CellLab website: cell-lab.net/, viewed on 01/13/19
- Eco website: strangeloopgames.com/eco/, viewed on 01/13/19

- Other
  - Wikipedia: “L-systems”: , viewed on 01/30/19
  - SpeedTree webiste: store.speedtree.com/, viewed on 01/13/19
  - Unreal Engine 4 website: unrealengine.com/en-US/what-is-unreal-engine-4