

Prozedurale Generierung von Baumstrukturen innerhalb der Unreal Engine 4

Bachelor Abschlussarbeit

David Liebemann, 17.03.17

Überblick

1. Einleitung
2. Lindenmayer-Systeme
3. Space Colonization Algorithmus
4. Ergebnisse
5. Zusammenfassung und Ausblick

1 Einleitung

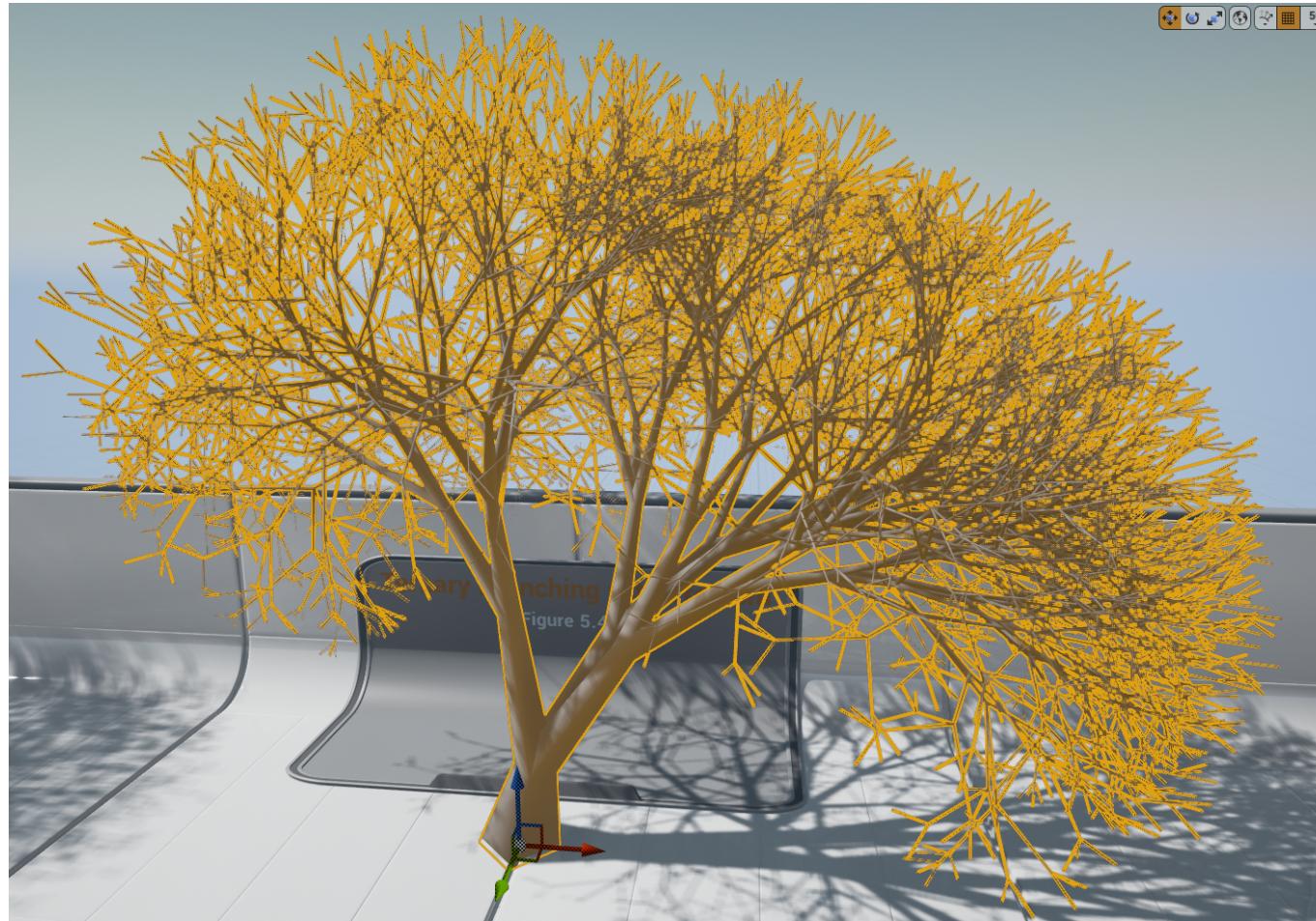


Ansatz

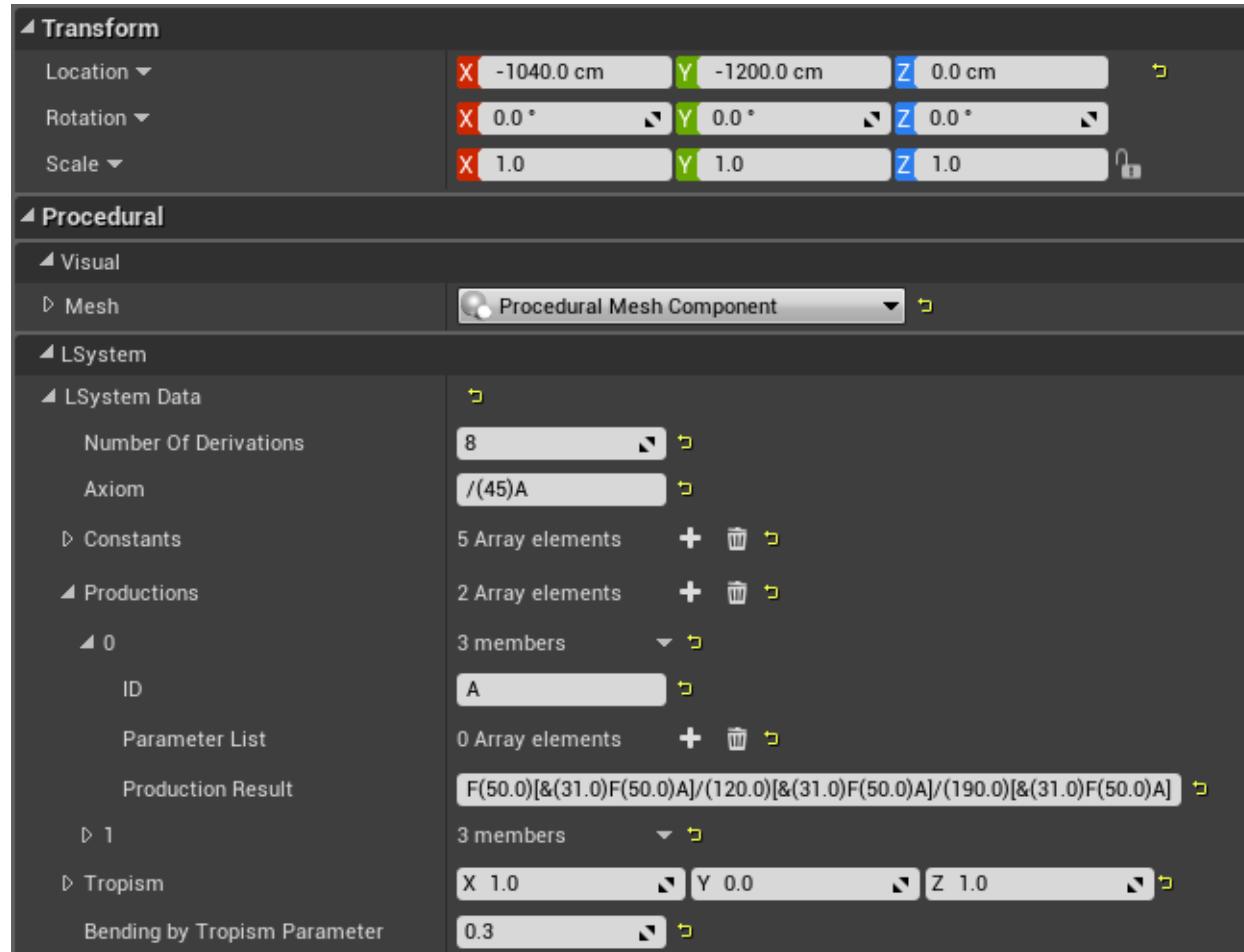
- Implementierung von zwei Verfahren zur prozeduralen Generierung von Baumstrukturen:
 - Lindenmayer-Systeme
 - Space Colonization Algorithmus
- Verwendung des Frameworks „Unreal Engine 4“
- Vereinfachte Darstellung von Ästen in Form von Zylindern

Unreal Engine 4

- Sammlung von Softwarewerkzeugen
- Erstellung von Inhalten mit C++ oder Blueprint und auf Basis von Framework-Klassen
- Verfügbarkeit eines visuellen Editor erlaubt:
 - Einfache Positionierung von Actors
 - Eingabe von Parametern über das Editor-UI



Positionierung eines Actors im Leveleditor



Eingabefenster für Actor-Parameter

2 Lindenmayer-Systeme

- Von Aristid Lindenmayer 1968 entwickelte Erweiterung von Ersetzungssystemen
- Weitere Ergänzungen durch Prusinkiewicz und Lindenmayer in 1990
- Funktionsweise basiert auf der Ersetzung von Zeichen in Zeichenketten
- Grafische Interpretation der Resultate ergibt Modelldaten

2.1 D0L-Systeme

Definition D0L-System: 2.1.1 Ein deterministisches, kontextfreies L-System (D0L-System) ist ein Tupel $G = (V, P, \omega)$ mit:

V : Ein nichtleeres, endliches Alphabet

P : Eine Menge von Produktionsregeln in der Form $P : a \rightarrow b$ mit $a \in V$ und $b \in V^*$

$\omega \in V^+$: Das Axiom, Startwort des L-Systems

Verwendete Begriffe:

Deterministisch: Es existiert genau eine Produktionsregel für jedes der Symbole in V

Kontextfrei: Ersetzung findet unabhängig von umgebenden Symbolen statt

Ableitung: Gleichzeitige Ersetzung aller Symbole eines Wortes anhand der Produktionsregeln

Beispiel: Simulation des Wachstums der Blaualgen-Gattung „Anabaena“

V besteht aus den Symbolen $\{a_l, a_r, b_l, b_r\}$

a und b: Größe und Teilungsbereitschaft einer Zelle

l und r: Zellenpolarität

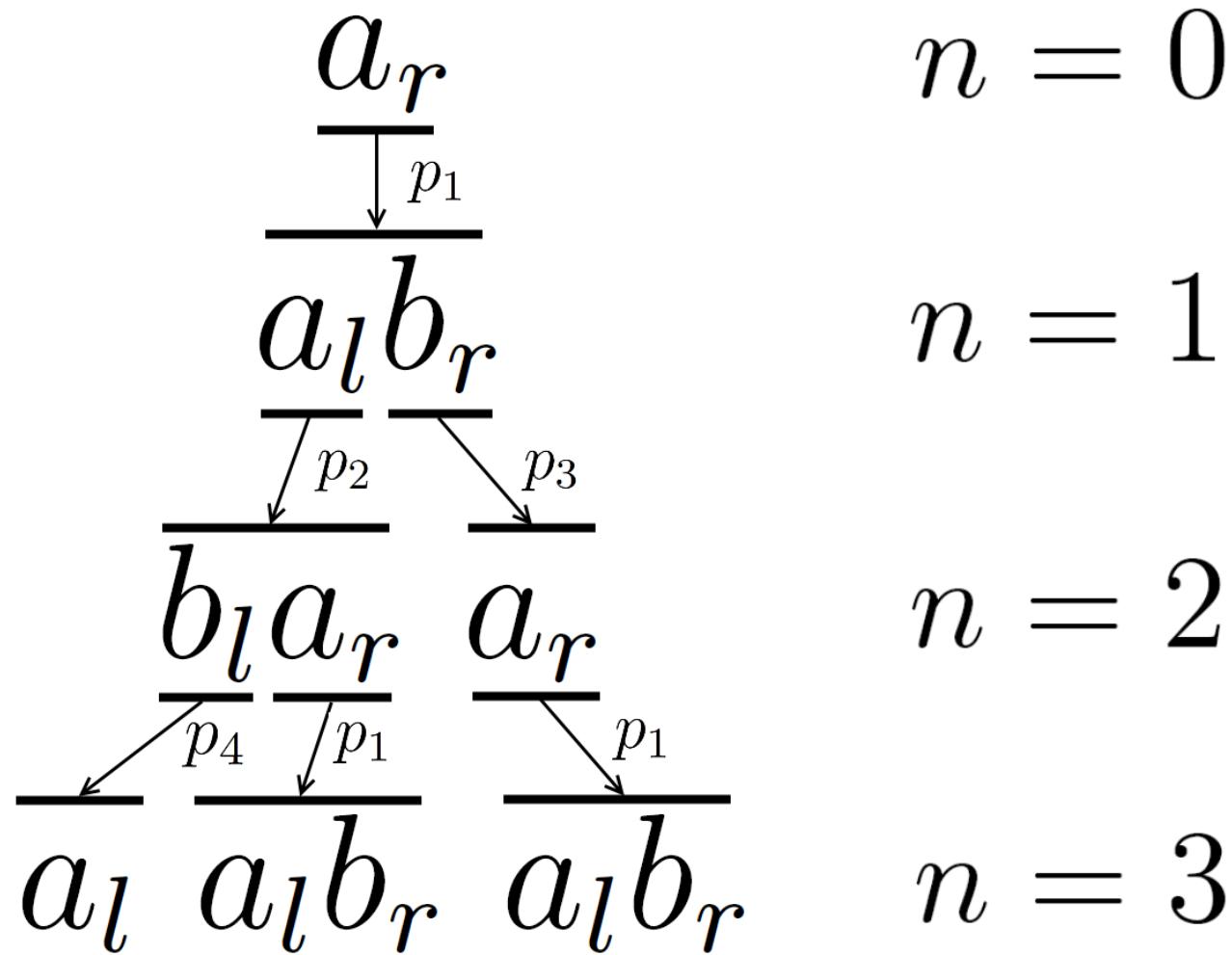
P besteht aus:

$$p_1 : a_r \rightarrow a_l b_r$$

$$p_2 : a_l \rightarrow b_l a_r$$

$$p_3 : b_r \rightarrow a_r$$

$$p_4 : b_l \rightarrow a_l$$



2.2 Parametrische L-Systeme

- Erweiterung der DOL-Systeme
- Verwendung von parametrischen Wörtern anstatt einfacher Symbole:

$A(a_1, \dots, a_n)$: Parametrisches Wort mit $A \in V$ und $a_1, \dots, a_n \in \Sigma$

Σ : Menge formaler Parameter

- Verwendung arithmetischer Ausdrücke im Nachfolger möglich

Beispiel: Definition und Ableitung eines parametrischen L-Systems

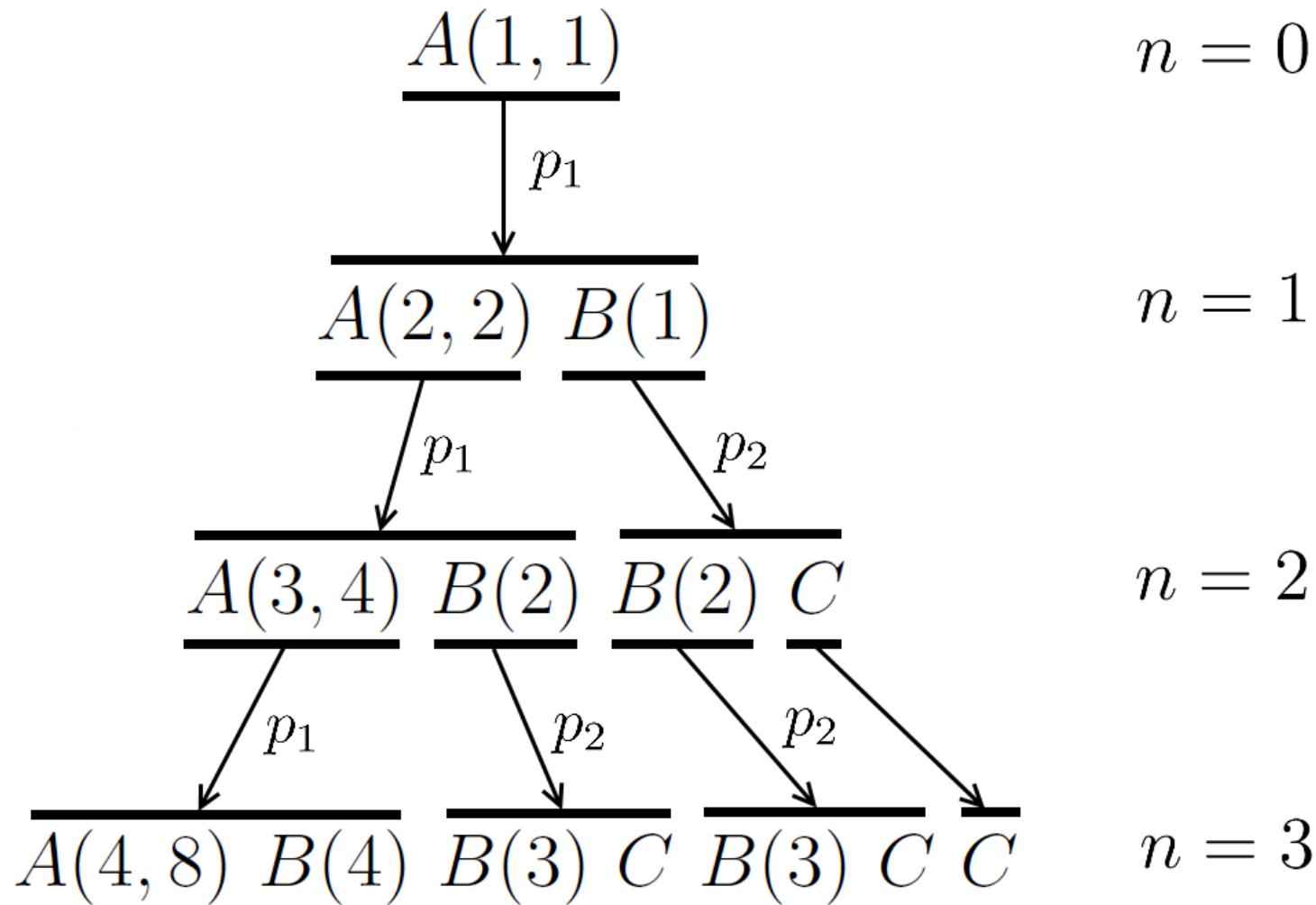
$$\begin{aligned}\omega &: A(1, 1) \\ p_1 &: A(x, y) \rightarrow A(x + 1, y * 2) B(y) \\ p_2 &: B(z) \rightarrow B(z + 1) C\end{aligned}\tag{1}$$

$$V = \{A, B, C\}$$

$$\Sigma = \{x, y, z\}$$

$$P = \{p_1, p_2\}$$

$$\omega = A(x, y) \text{ mit } x = 1 \text{ und } y = 1$$



2.3 Grafische Interpretation von L-Systemen

- Rückgabewerte von L-System-Ableitungen sind Zeichenketten, keine Modelldaten
- Grafische Interpretation von Zeichenketten: Turtle-Interpretation
- Zustand der Turtle ist ein Tupel (\vec{p}, \vec{H}) mit:
 - \vec{p} : Position der Turtle
 - \vec{H} : Blickrichtung (Heading) der Turtle

Turtle-Aktionen:

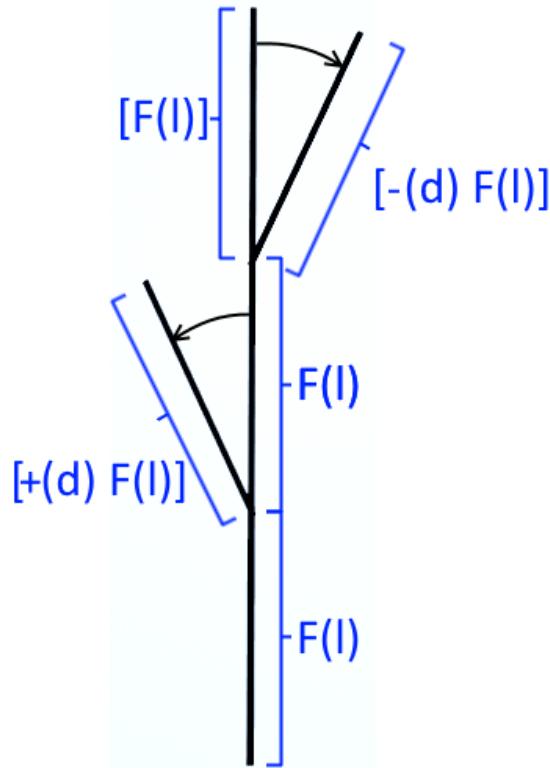
$F(l)$: Bewegung um $l > 0$ in Blickrichtung, Aktualisierung der Position und Zeichnung einer Linie

$+(d)$: Drehung um den Winkel d nach links, Aktualisierung der Blickrichtung

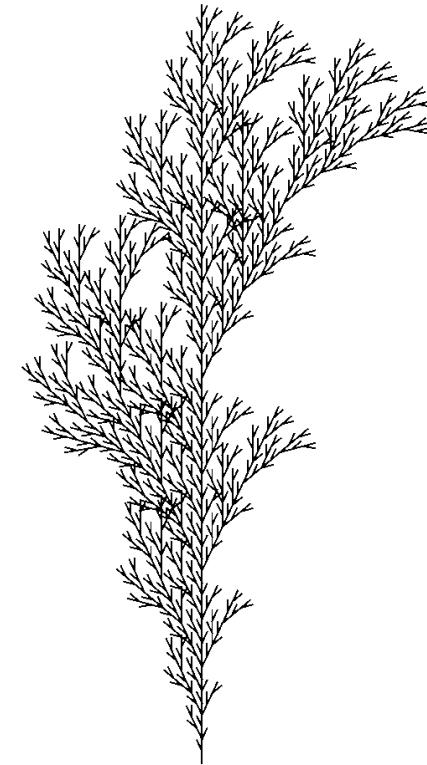
$-(d)$: Drehung um den Winkel d nach rechts, Aktualisierung der Blickrichtung

[: Ablage des Zustands auf einem Stack

] : Entnahme des obersten Zustands vom Stack und Aktualisierung des aktuellen Zustands



$n = 1, l = 240, d = 25^\circ$



$n = 5, l = 15, d = 25^\circ$

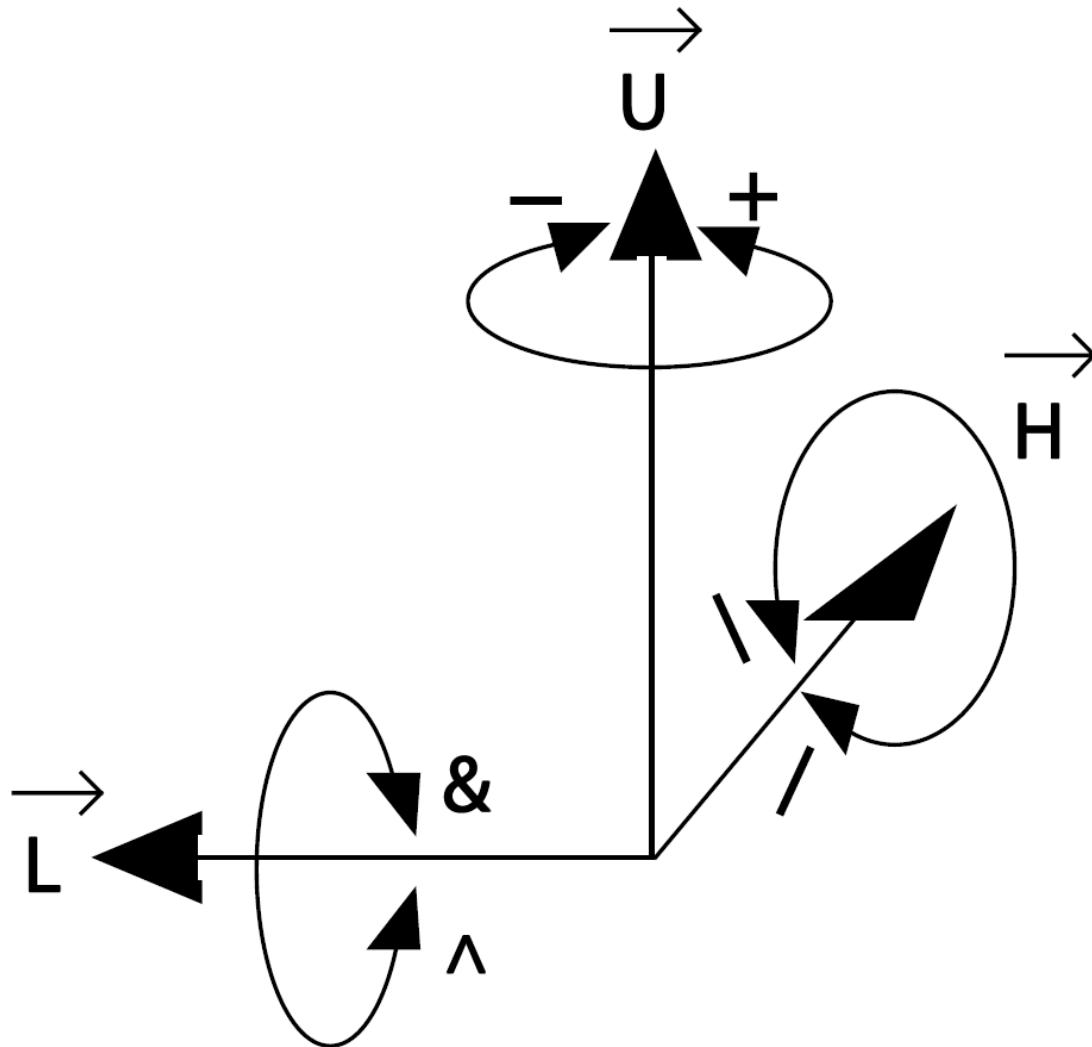
$$\omega : F(l)$$

$$p_1 : F(l) \rightarrow F(l) [+(d) F(l)] F(l) [-(d) F(l)] [F(l)]$$

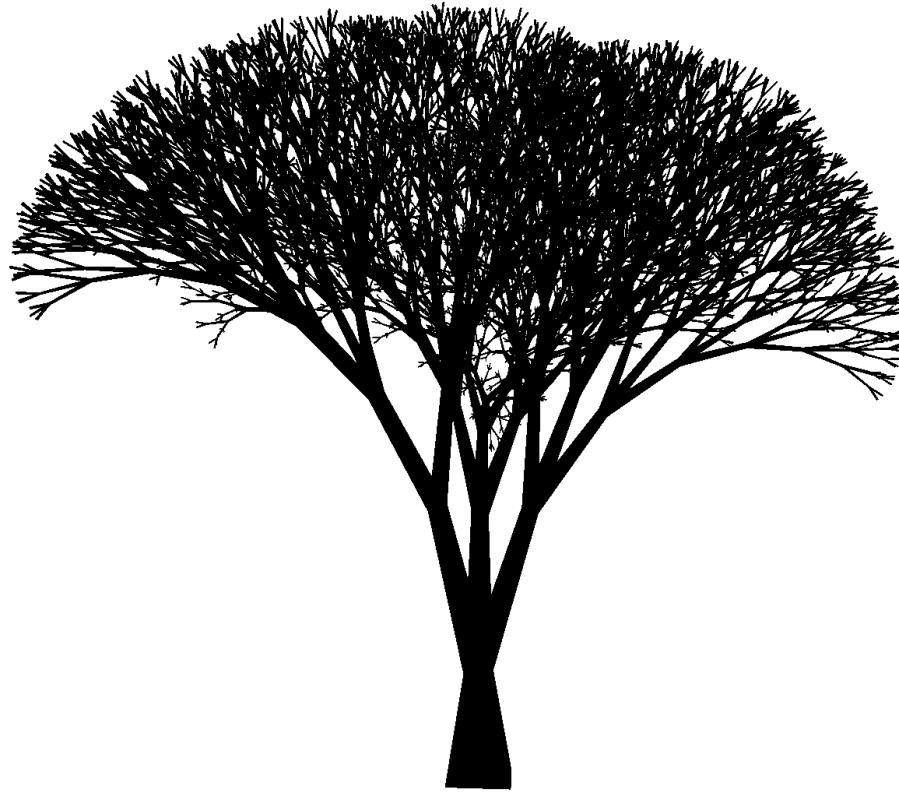
2.4 Anpassungen an Baumstrukturen

Erweiterung der Turtle-Interpretation in den dreidimensionalen Raum

- Zustand der Turtle ist ein Tupel (\vec{p}, \mathbf{R}) mit:
 - \vec{p} : Position der Turtle
 - \mathbf{R} : Rotationsmatrix der Turtle
- Einheitsvektoren $\vec{H}, \vec{L}, \vec{U}$ bilden das lokale Koordinatensystem der Turtle:
 - \vec{H} : Heading-Vektor
 - \vec{L} : Left-Vektor
 - \vec{U} : Up-Vektor



[PL90, S.19]



$$n = 8, l = 50, d = 137.5^\circ, \\ a = 18.95^\circ, l_r = 1.3$$

$$\omega : / (45) \ A$$

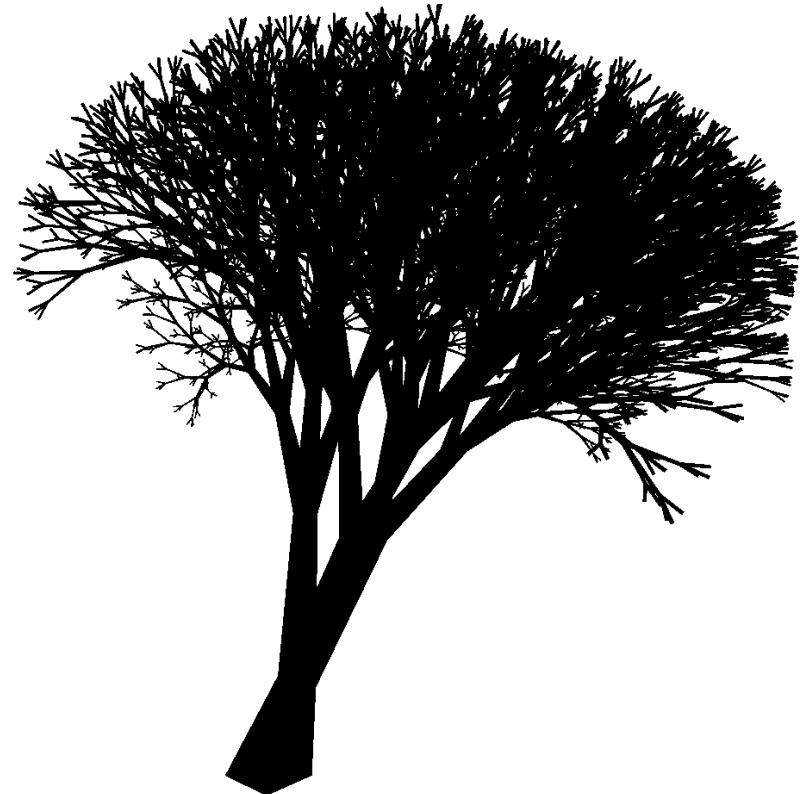
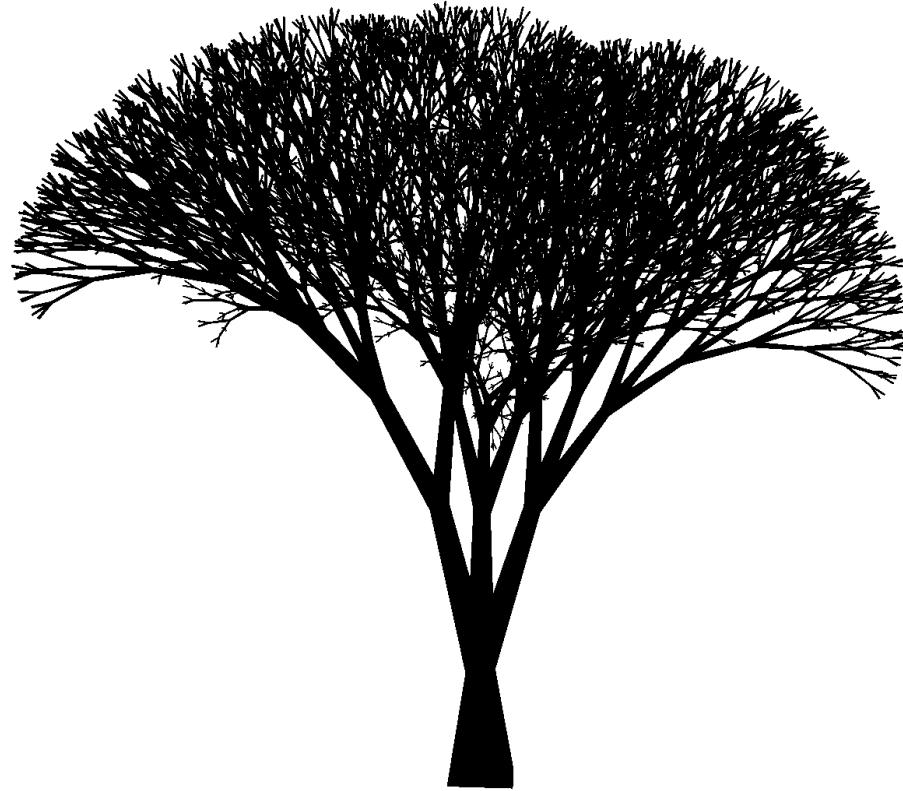
$$p_1 : A \rightarrow F(l) [\&(a)F(l)A] / (d) [\&(a)F(l)A] / (d) [\&(a)F(l)A] \quad (2)$$

$$p_2 : F(l) \rightarrow F(l * l_r)$$

Einfluss durch Tropismus

- Tropismus: Tendenz einer Pflanze in eine bestimmte Richtung zu wachsen
- Einfluss wird als Vektor $\vec{T} \in \mathbb{R}^3$ angegeben
- Beeinflusst die Bewegung der Turtle in Abhängigkeit des Beugungsfaktors $e \in \mathbb{R}$

2. L-Systeme – Anpassungen: Tropismus



$$\vec{T} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, e = 0$$

$$\vec{T} = \begin{pmatrix} 0 \\ 1 \\ -0.5 \end{pmatrix}, e = 0.27$$

3 Space Colonization Algorithmus

- Ursprünglich entwickelt zur Darstellung von Blattvenen
- Simuliert Konkurrenzverhalten von wachsenden Zweigen um Wachstumsraum
- Baut graphentheoretischen Baum auf
- Anschließende Nachbearbeitung und Visualisierung der Daten

3.1 Aufbau

Aufbau eines graphentheoretischen Baums $G = \langle V, E \rangle$ mit Eingaben:

S : Die Menge von Einflusspunkten

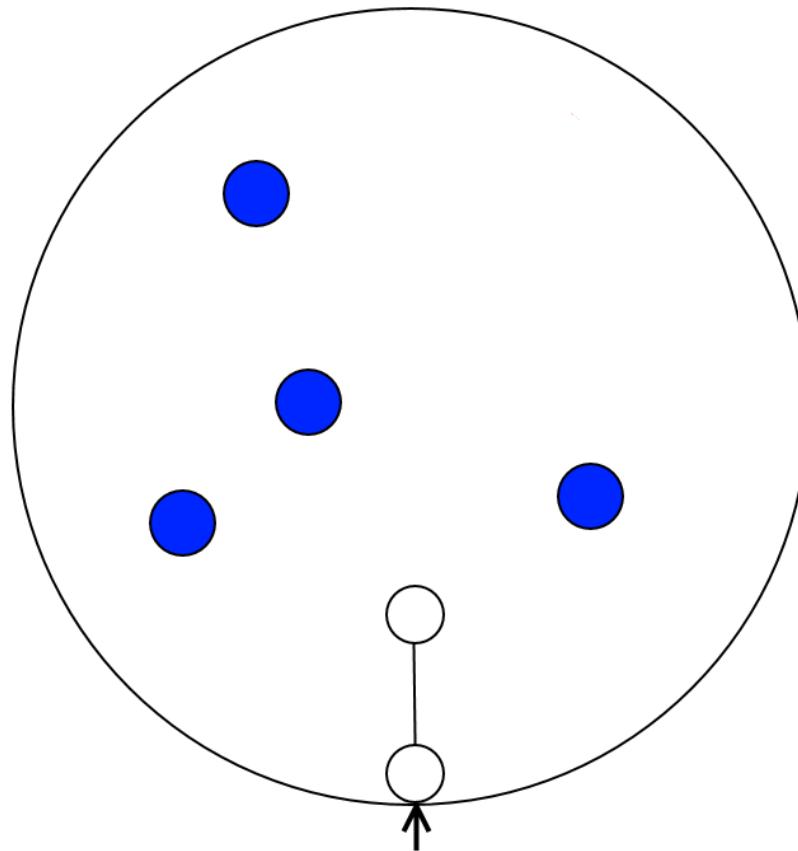
d_i : Der Einflussradius

d_k : Der Minimalradius (Kill-radius)

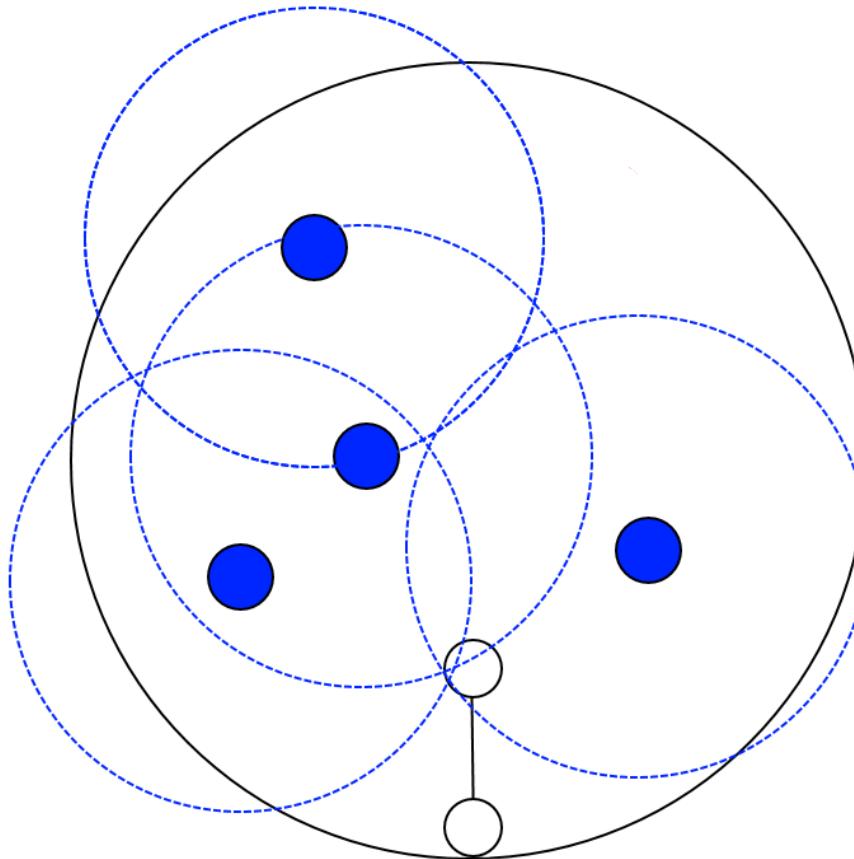
D : Die Schrittweite

\vec{T} Der Tropismusvektor

3.2 Ablauf

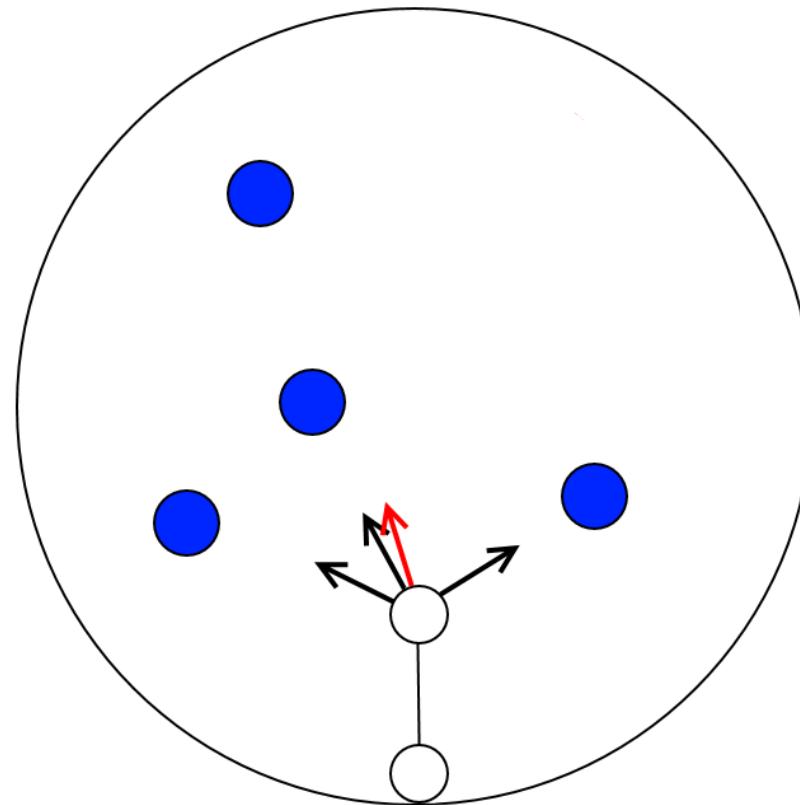


3. Space Colonization Algorithmus - Ablauf



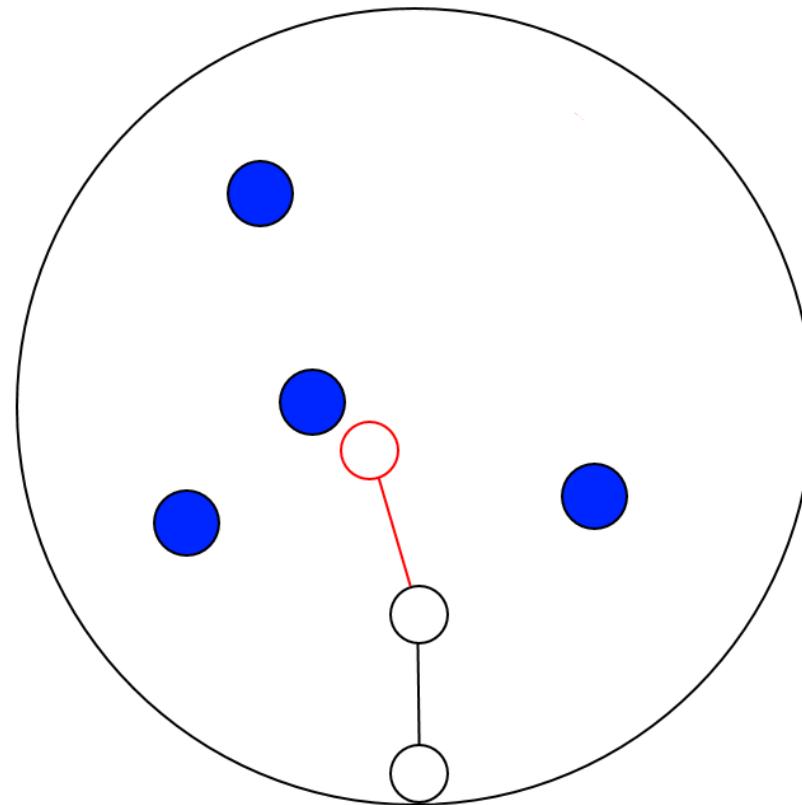
Bestimmung der minimal entfernten Knotenpunkte im Einflussradius

3. Space Colonization Algorithmus - Ablauf



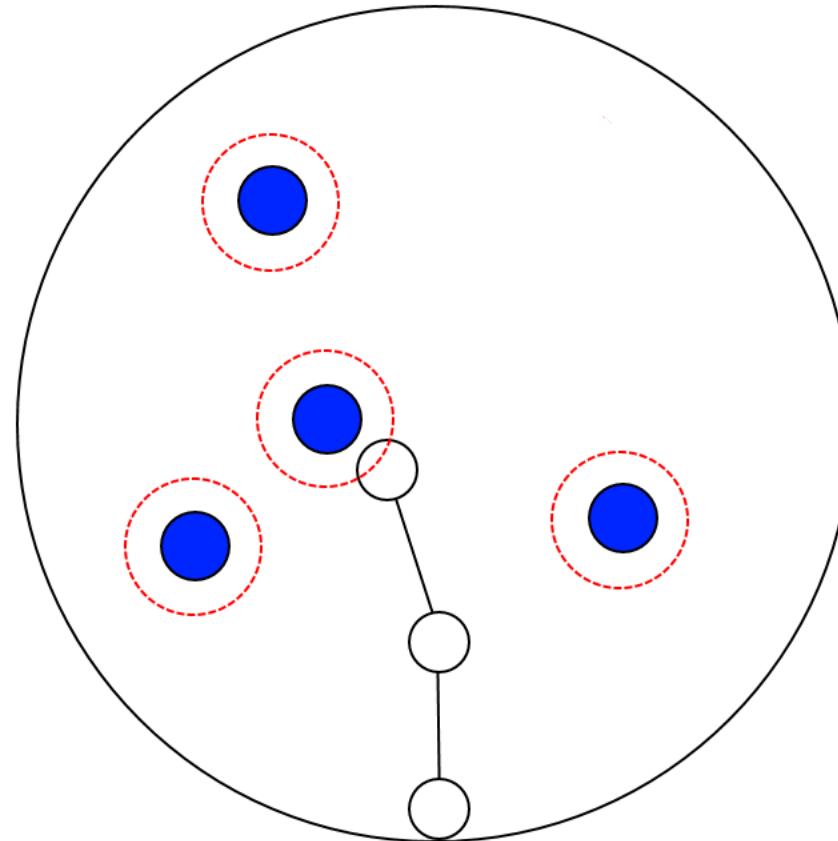
Berechnung der Wachstumsrichtung

3. Space Colonization Algorithmus - Ablauf



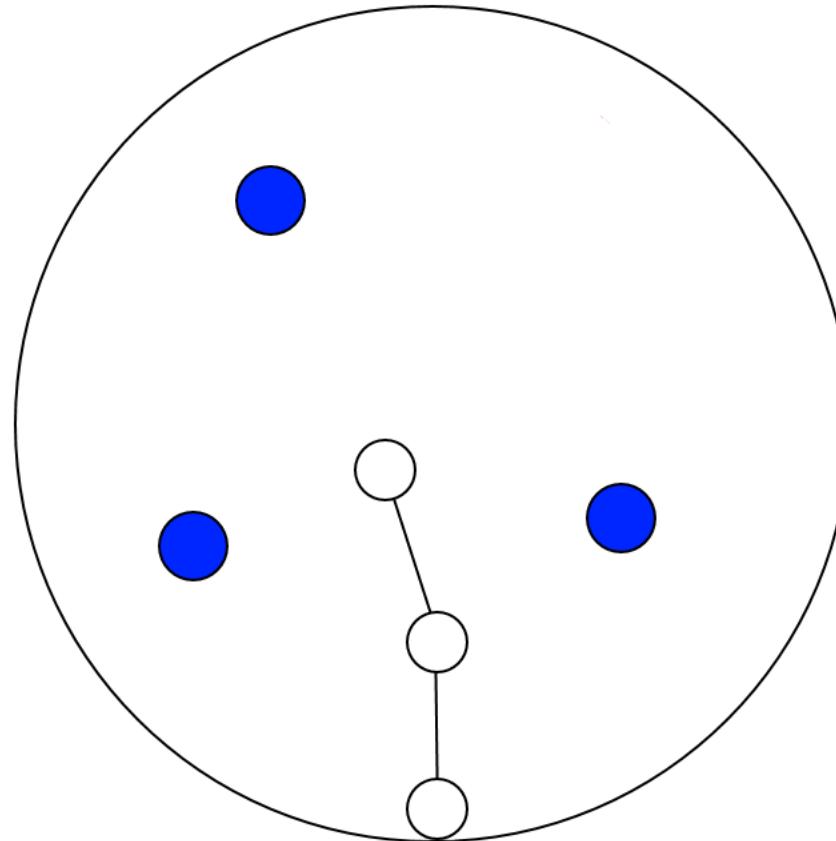
Platzierung des neuen Knotenpunkts

3. Space Colonization Algorithmus - Ablauf



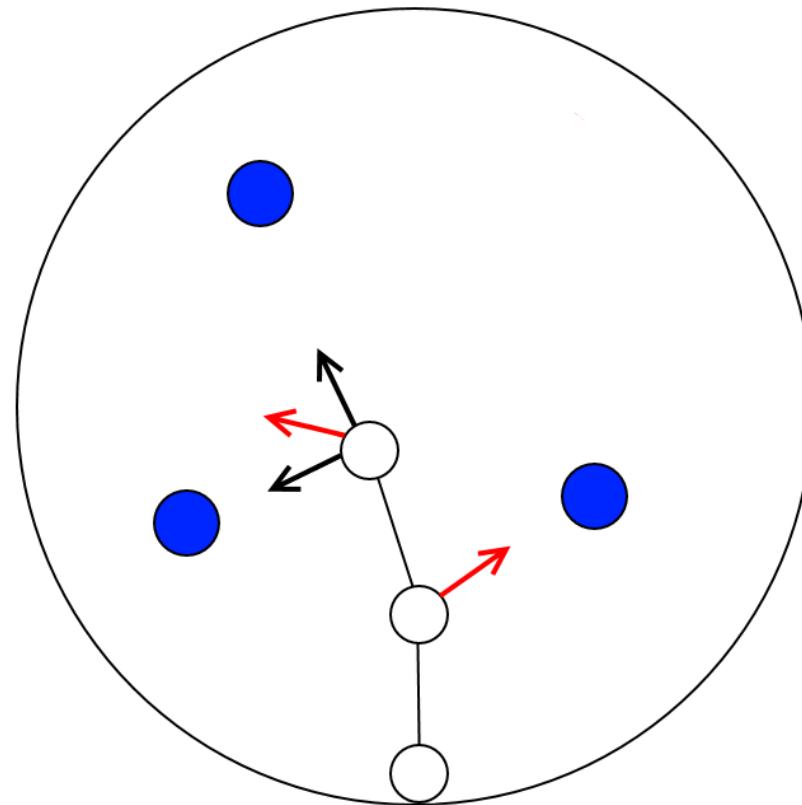
Bestimmung der Knotenpunkte im Minimalradius

3. Space Colonization Algorithmus - Ablauf



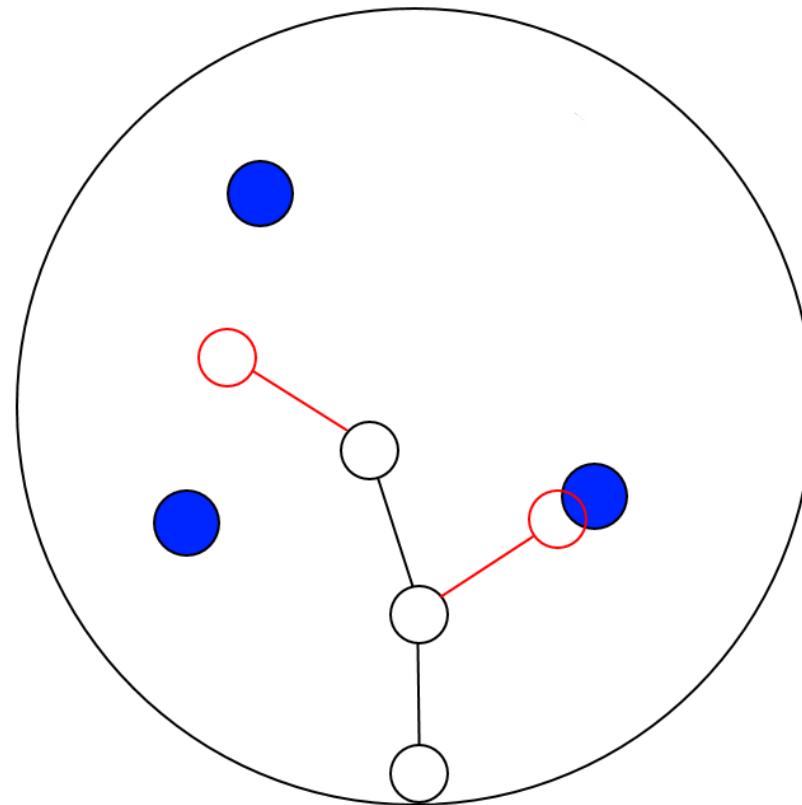
Ausgangssituation der nächsten Iteration

3. Space Colonization Algorithmus - Ablauf



Beeinflussung unterschiedlicher Knotenpunkte

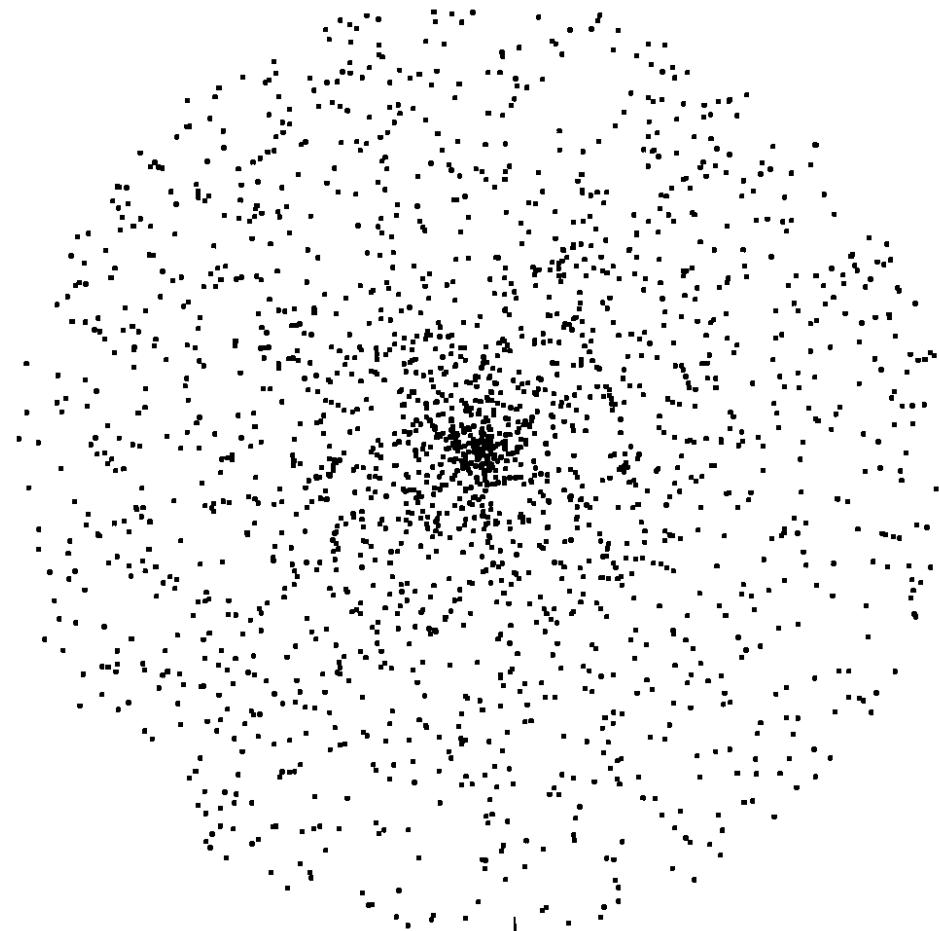
3. Space Colonization Algorithmus - Ablauf



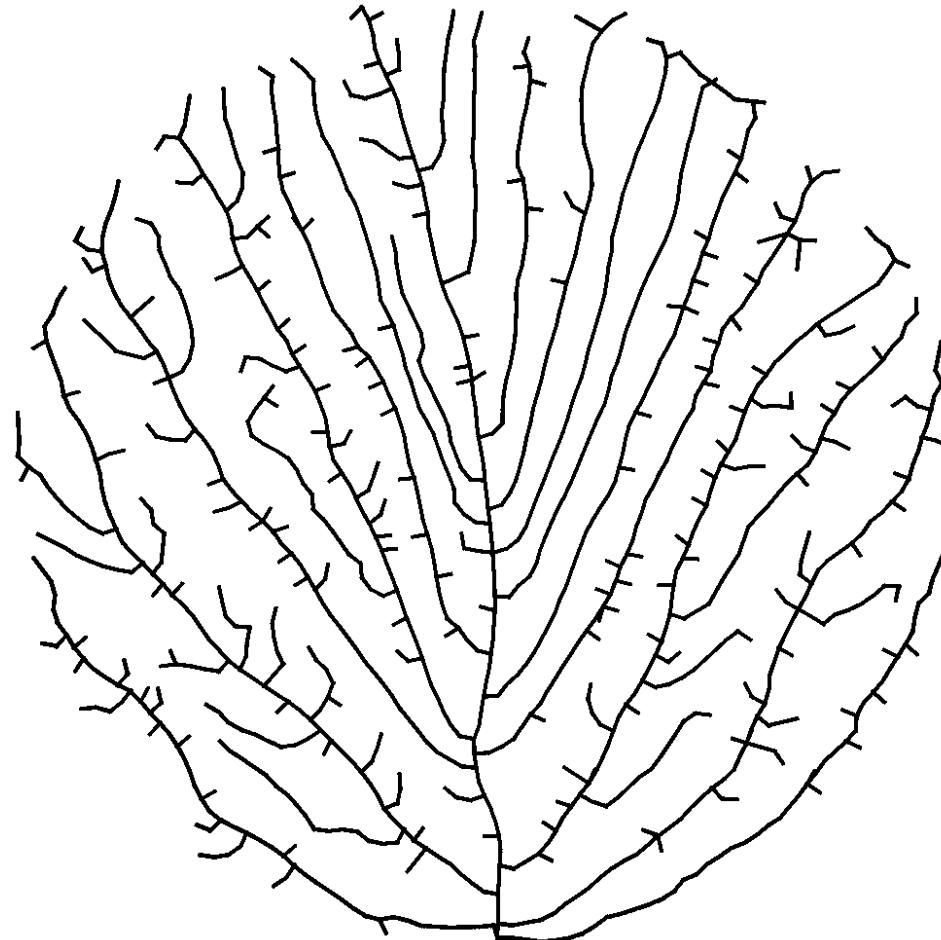
Entstehende Verzweigung

3.3 Generierung von Baumstrukturen

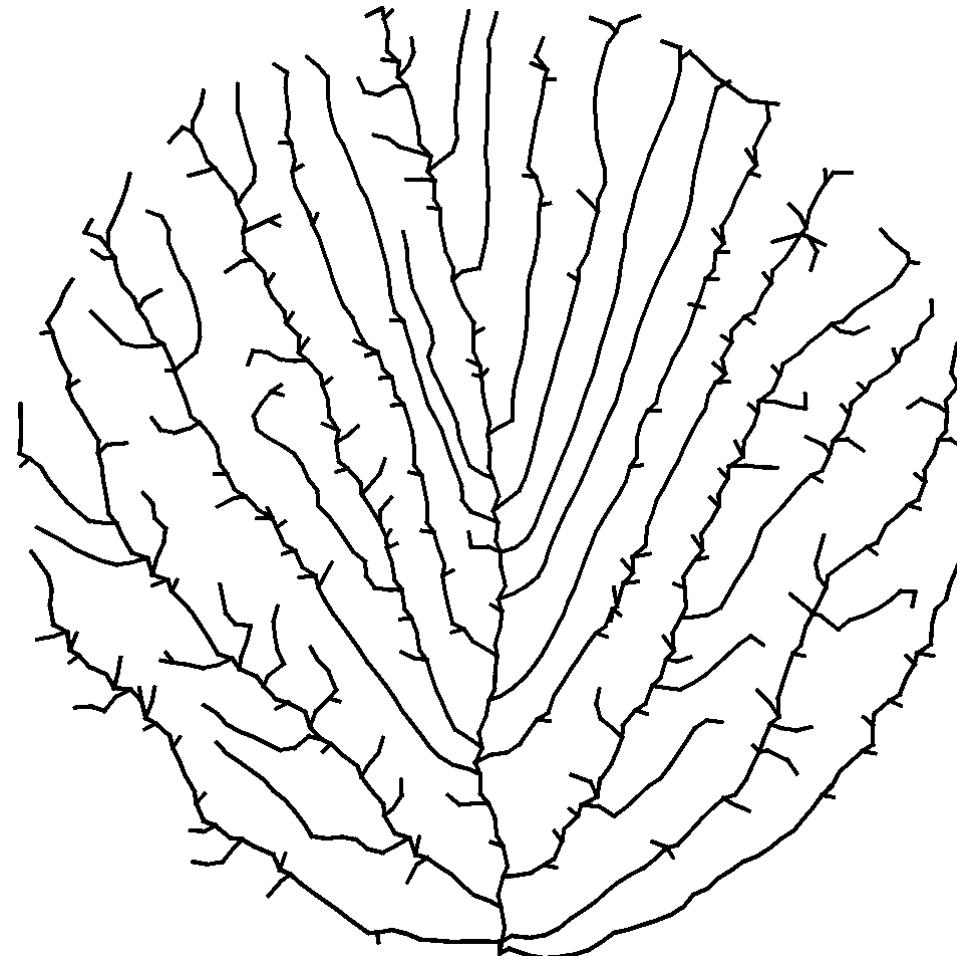
1. Befüllung des vorgegebenen Einflussbereichs
2. Iterative Generierung
3. Annäherung der Nachfolger jedes Knotenpunktes
4. Visualisierung der Kanten mithilfe von Zylindern



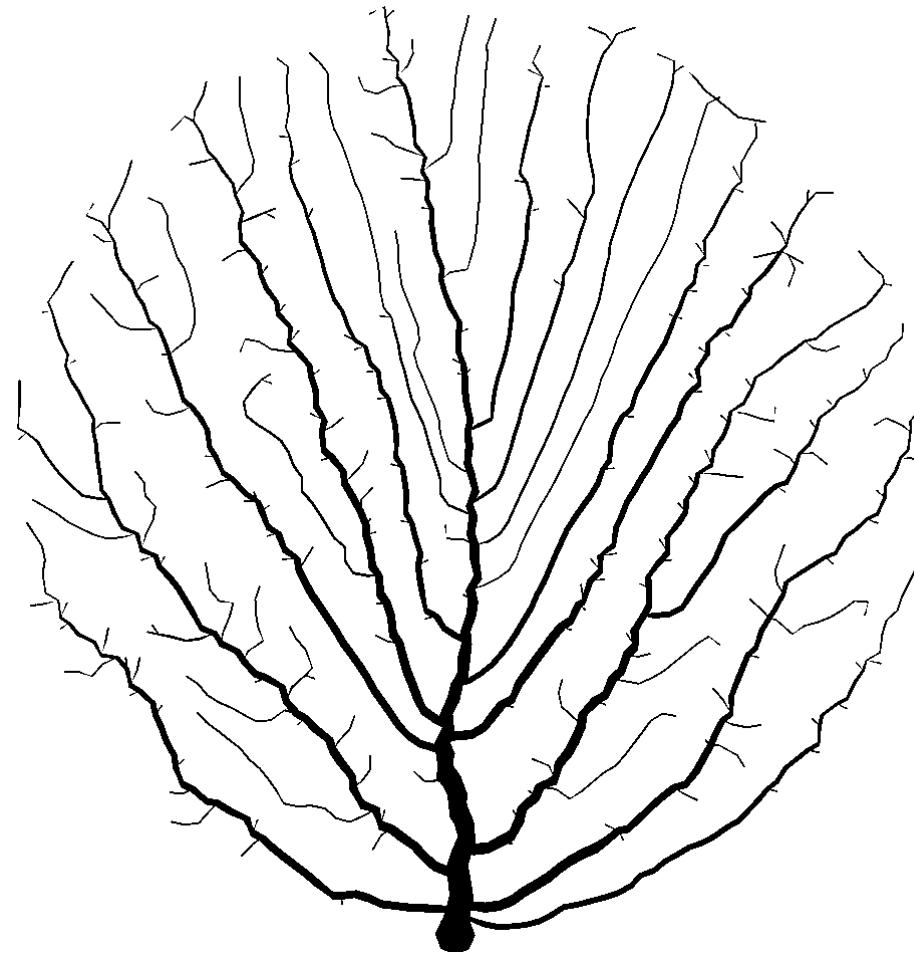
Verteilung von 2000 Einflusspunkten in einem Rings mit Radius $r = 500$



Ergebnis der iterativen Generierung mit $d_i = 100$, $d_k = 20$, $D = 15$, $\vec{T} = \vec{0}$



Annäherung der Nachfolger jedes Knotenpunktes



Visualisierung mithilfe von Zylindern

4 Ergebnisse

5 Zusammenfassung und Ausblick

- Lindenmayer-Systeme
- Space Colonization Algorithmus
- Ergebnisse

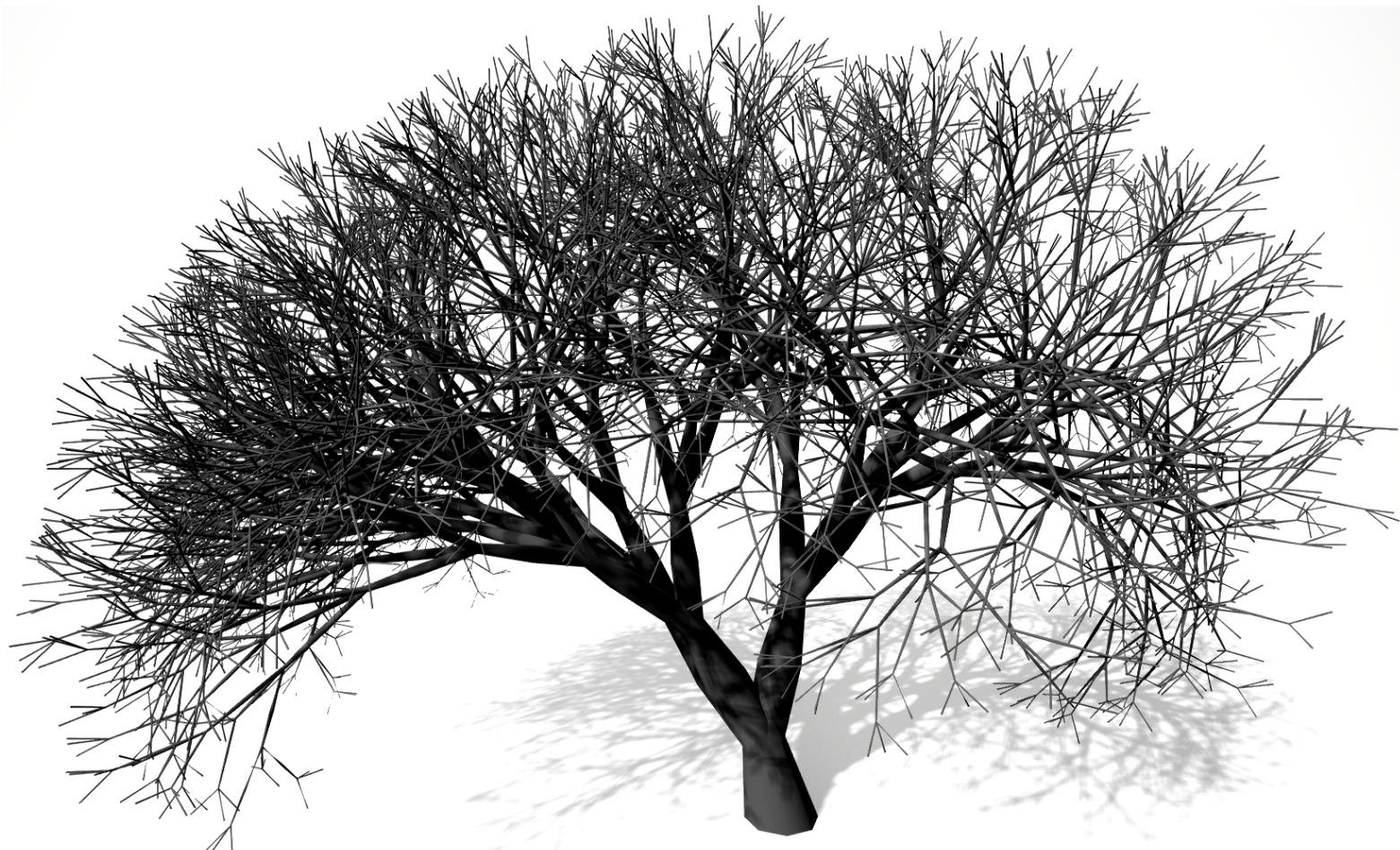
5.1 Bewertung der Ergebnisse - Visuell

L-Systeme

- + Genaue Kontrolle über generierte Baumstrukturen
- + Unterschiedliche Strukturen bei gleichbleibender L-System Definition möglich
- Muster und Regelmäßigkeiten sind erkennbar

Space Colonization Algorithmus

- + Für die Darstellung von unregelmäßigen, komplexen Baumstrukturen geeignet
- + Generiert auch ohne Ergänzungen realistische Formen
- + Anpassung der Baumstruktur an einschränkende Bedingungen
- Für die Darstellung vieler Baumsorten nicht geeignet



Effizienz – Generierungszeit = $0.233s$, 206658 Vertizes



Effizienz – Generierungszeit = 2.416s, 103725 Vertizes

5.2 Bewertung der Ergebnisse - Benutzerfreundlichkeit

L-Systeme

- Verlangen vom Benutzer:
 - Genaue Vorstellung der Baumstruktur
 - Kenntnisse über L-System Definitionen
 - Verständnis für die Umsetzung als L-System

Space Colonization Algorithmus

- Klare Zusammenhänge zwischen Parametern und Baumstrukturform
- „Trial and Error“-Vorgehen möglich

5.3 Wünschenswerte Erweiterungen

- Generierung zur Laufzeit
- Level-of-Detail (LOD)
- Oberflächenbeschaffenheit
- Blätter
- Verteilung der Baumstrukturen

Literatur

- [Bak] BAKER, MARTIN JOHN: *Maths - Angle between vectors.* <http://www.euclideanspace.com/maths/algebra/vectors/angleBetween/index.htm>.
- [Bal98] BALZERT, HELMUT: *Lehrbuch der Software-Technik : Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung.* Spektrum Akademischer Verlag GmbH, 1998.
- [Bec05] BECKER, PETE: *Working Draft, Standard for Programming Language C++,* 2005. <http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/n1905.pdf>.
- [Blo85] BLOOMENTHAL, JULES: *Modeling the mighty maple.* Computer Graphics Laboratory, New York Institute of Technology, Old Westbu-

ry, New York, 1985. <https://pdfs.semanticscholar.org/00d3/4582edd116a23d4d574ad2c90e9ebf01d74d.pdf>.

- [DL05] DEUSSEN, OLIVER und BERND LINTERMANN: *Digital Design of Nature - Computer Generated Plants and Organics*. Springer-Verlag Berlin Heidelberg 2005, 2005.
- [EKh10] EBERHARDT, HENNING, VESA KLUMPP und UWE D. HANEBECK: *Density Trees for Efficient Nonlinear State Estimation*, 2010. http://isas.uka.de/Publikationen/Fusion10_EberhardtKlumpp.pdf.
- [Eng] *Unreal Engine Documentation : Engine Features*. <https://docs.unrealengine.com/latest/INT/Engine/index.html>.
- [FGR] FINK, PROF. DR. SIEGFRIED, JÖRG GRÜNER und DR. CHRISTIAN RABE: *Skript zum Kernblock „Forstbotanik und Baumphysiologie II“ – Forstbotanischer Teil*. <https://www.forstbotanik.uni-freiburg.de/Lehre/Skripten/Skript%20Forstbotanik%20II>.

7. Literatur

- [Gre] *Green One - A landmark render of XfrogPlants by Jan Walter Schliep.* http://xfrog.com/gallery/landscapes/green01_big-1600small.jpg.php.
- [GSJ04] GOLDMAN, RON, SCOTT SCHAEFER und TAO JU: *Turtle Geometry in Computer Graphics and Computer Aided Design*, 2004. <http://www.cs.wustl.edu/~taoju/research/TurtlesforCADRevised.pdf>.
- [LN02] LEFEBVRE, SYLVAIN und FABRICE NEYRET: *Synthesizing Bark*, 2002. <http://www-evasion.imag.fr/Publications/2002/LN02/bark.pdf>.
- [Lux14] LUX, PROF. DR. ANDREAS: *Algorithmen und Datenstrukturen - Vorlesungsskript Kapitel 4*, 2014.
- [Man83] MANDELBROT, BENOIT B.: *The Fractal Geometry of Nature*. W. H. Freeman and Company, 1983.

- [PL90] PRUSINKIEWICZ, PRZEMYSLAW und ARISTID LINDENMAYER: *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, eBook Auflage, 1990. <http://algorithmicbotany.org/papers/abop/abop.pdf>.
- [Proa] *Procedural Mesh Component in C++ : Getting Started*. https://wiki.unrealengine.com/Procedural_Mesh_Component_in_C%2B%2B:_Getting_Started.
- [Prob] *Profiling, How To Count CPU Cycles Of Specific Blocks Of Your Game Code*. https://wiki.unrealengine.com/Profiling,_How_To_Count_CPU_Cycles_Of_Specific_Blocks_Of_Your_Game_Code.
- [Ran] *Unreal Engine 4 Documentation : Random Streams - Initial Seed*. <https://docs.unrealengine.com/latest/INT/Engine/Blueprints/UserGuide/RandomStreams/#initialseed>.
- [RFL⁺05] RUNIONS, ADAM, MARTIN FUHRER, BRENDAN LANE, PAVOL FEDERL, ANNE-GAËLLE ROLLAND-LAGAN und PRZEMYSLAW PRUSIN-

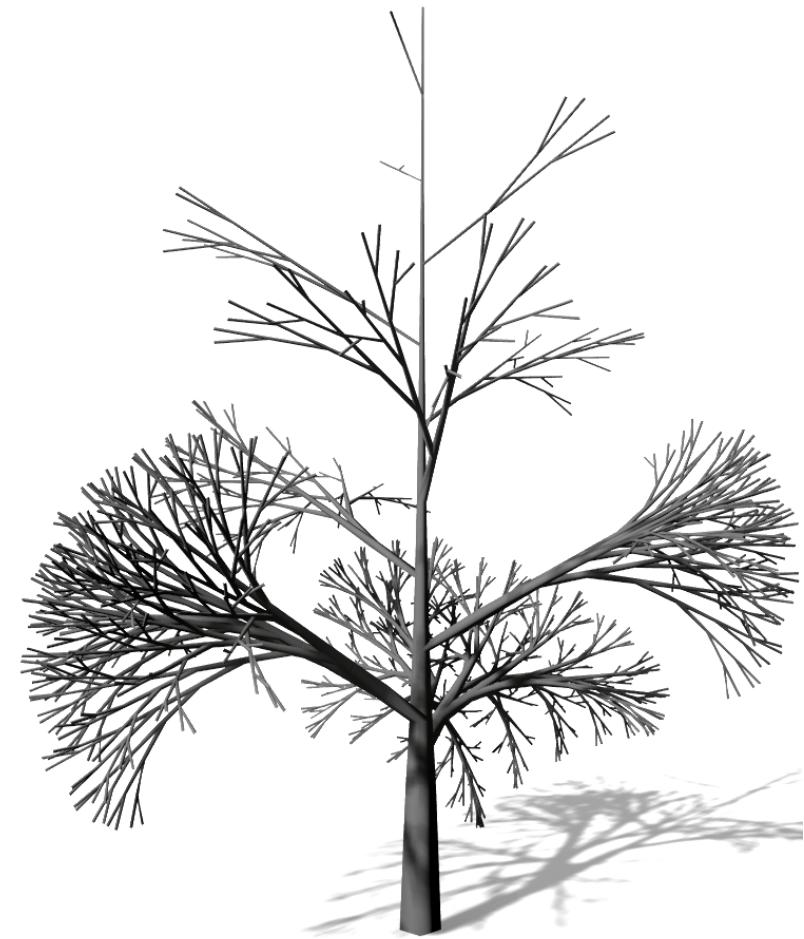
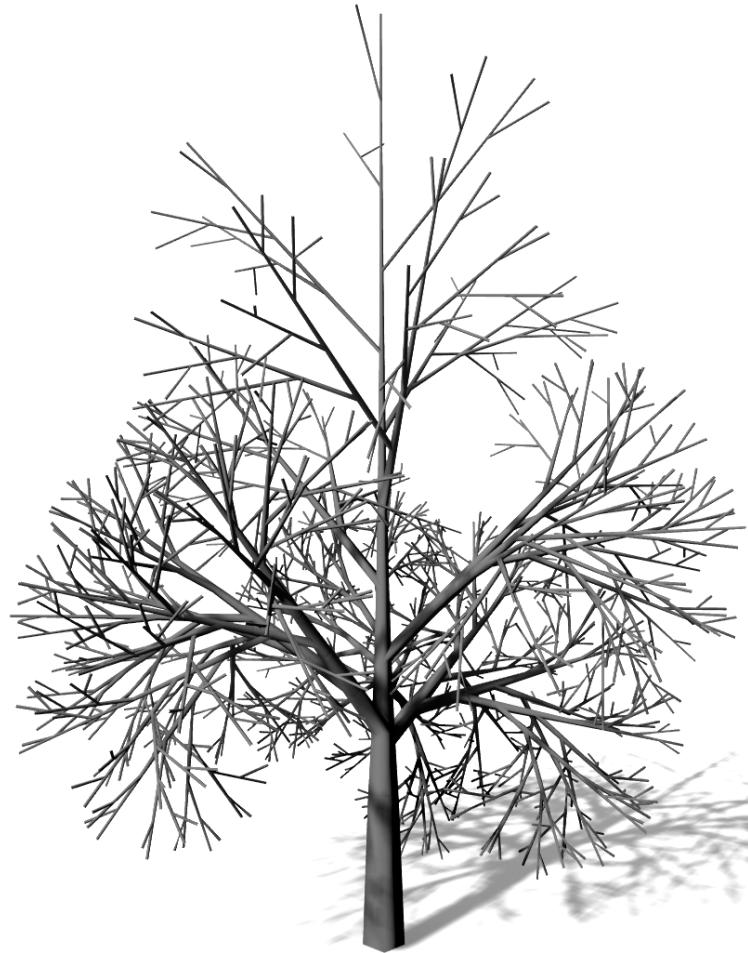
KIEWICZ: *Modeling and visualization of leaf venation patterns*, 2005. <http://algorithmicbotany.org/papers/venation.sig2005.pdf>.

- [RLP07] RUNIONS, ADAM, BRENDAN LANE und PRZEMYSLAW PRUSINKIEWICZ: *Modeling Trees with a Space Colonization Algorithm*, 2007. <http://algorithmicbotany.org/papers/colonization.egwnp2007.pdf>.
- [Sch14] SCHMITZ, PROF. DR. HEINZ: *Theoretische Informatik - Vorlesungsskript*, 2014.
- [STN16] SHAKER, NOOR, JULIAN TOGELIUS und MARK J. NELSON: *Procedural Content Generation in Games*. Springer International Publishing Switzerland 2016, 2016.
- [Sura] SURIDGE, JAYELINDA: *Modelling by numbers: Part One A: An introduction to procedural geometry*. http://www.gamasutra.com/blogs/JayelindaSuridge/20130903/199457/Modelling_by_numbers_Part_One_A.php.

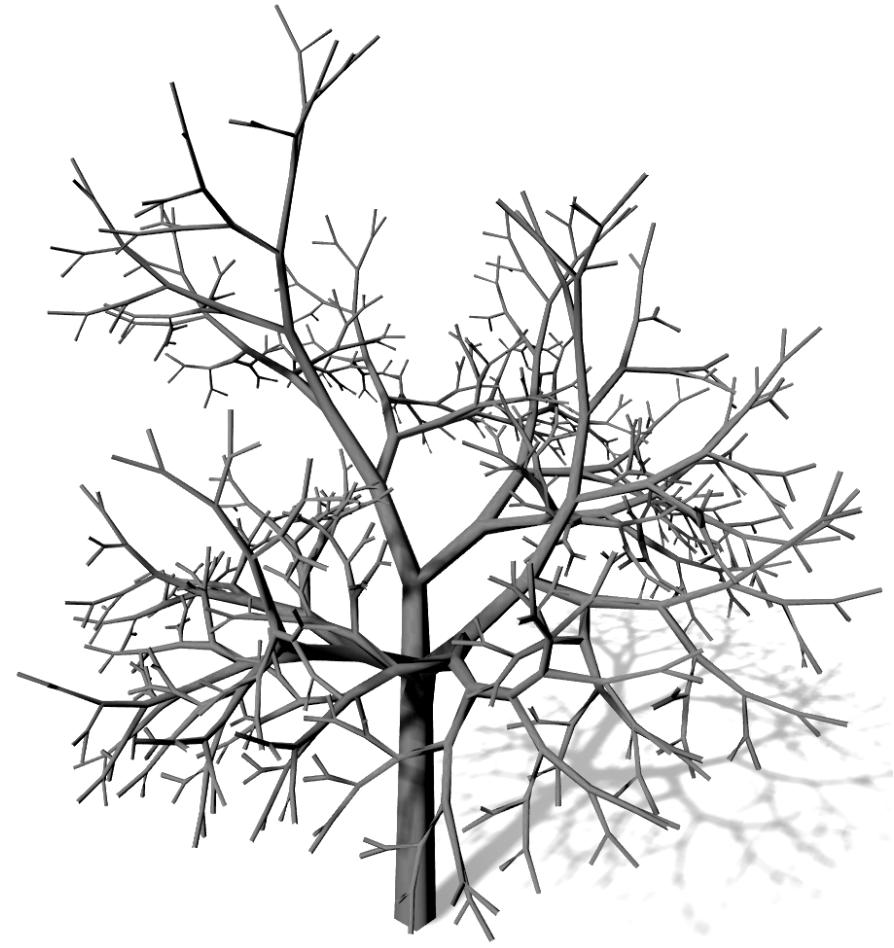
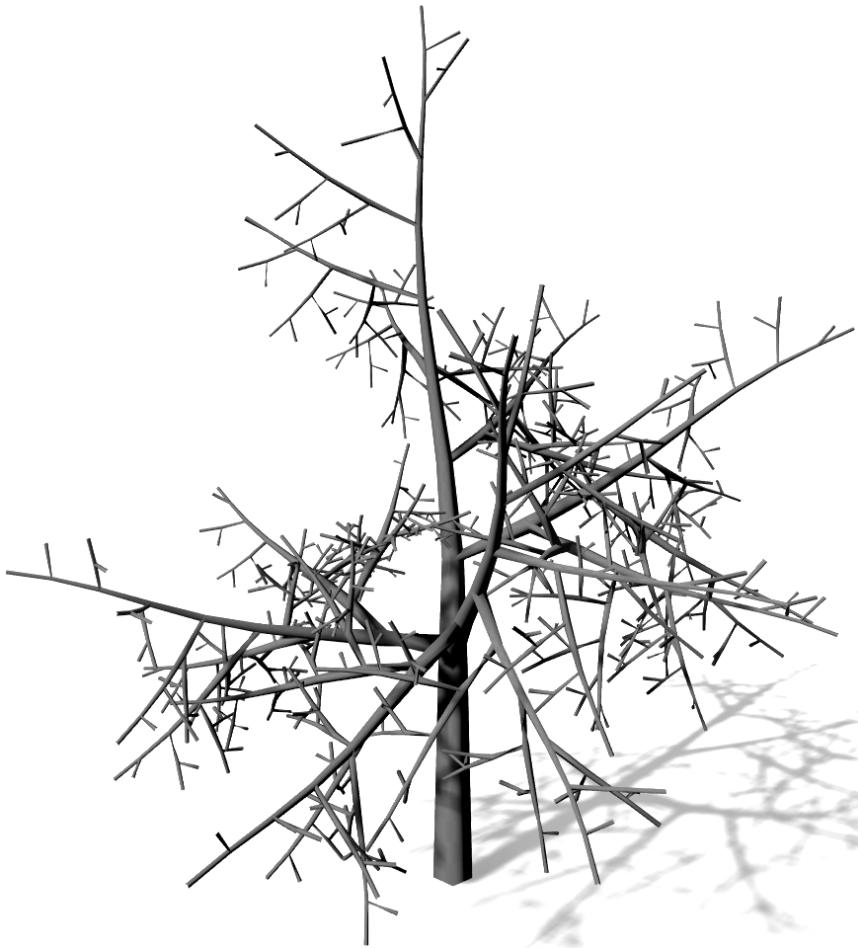
- [Surb] SURIDGE, JAYELINDA: *Modelling by numbers: Part Two A: The cylinder.* http://www.gamasutra.com/blogs/JayelindaSuridge/20130905/199626/Modelling_by_numbers_Part_Two_A.php.
- [TKSY] TOGELIUS, JULIAN, EMIL KASTBJERG, DAVID SCHEDL und GEORGIOS N. YANNAKAKIS: *What is Procedural Content Generation? Mario on the borderline*. <http://julian.togelius.com/Togelius2011What.pdf>.
- [Unra] *Unreal Engine Documentation : Content Examples.* <https://docs.unrealengine.com/latest/INT/Resources/ContentExamples/>.
- [Unrb] *Unreal Engine Documentation : Unreal Engine 4 Terminology.* <https://docs.unrealengine.com/latest/INT/GettingStarted/Terminology/index.html>.
- [Wha] *Unreal Engine Features.* <https://www.unrealengine.com/unreal-engine-4>.

5.4 Ergebnisse (2)

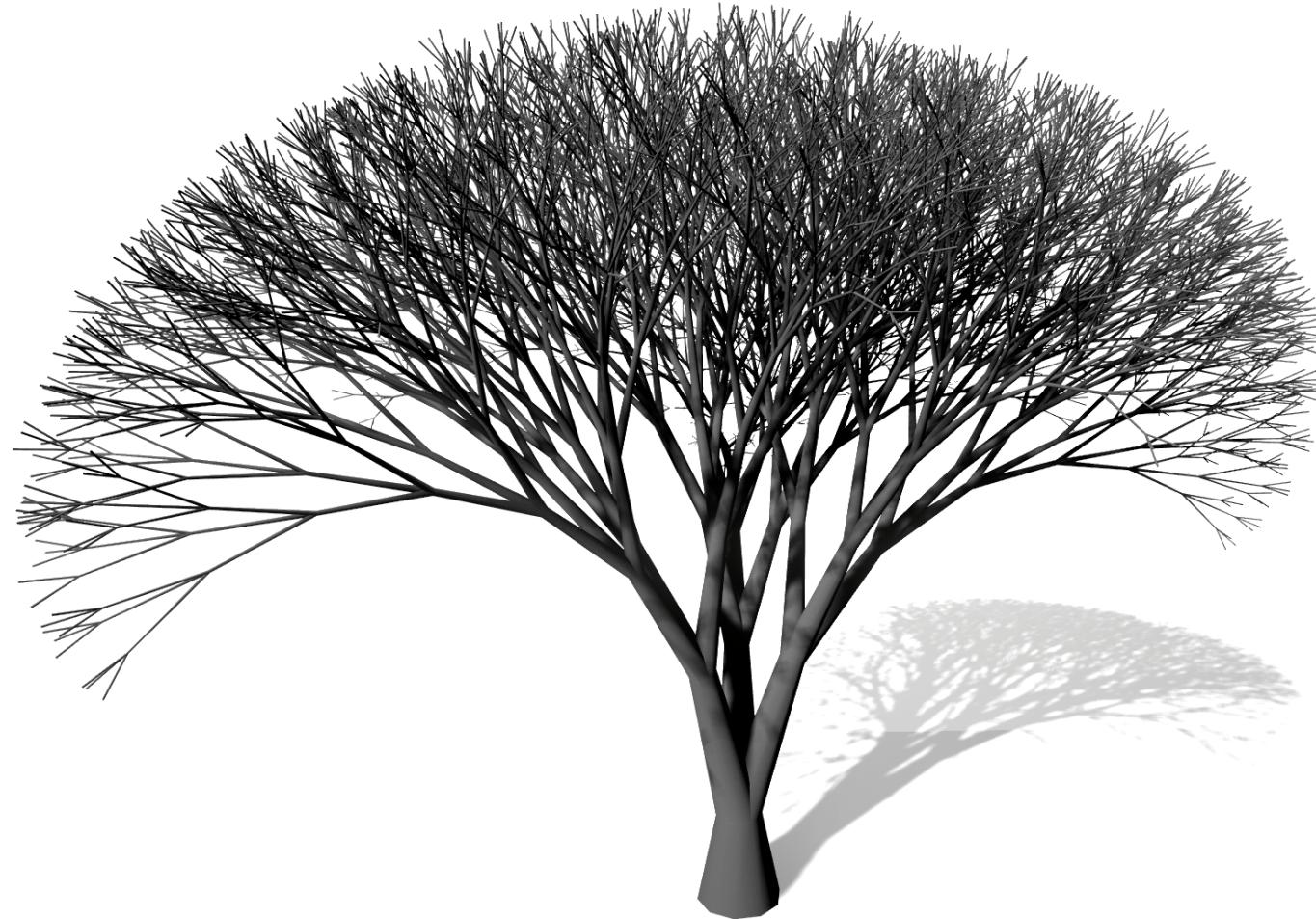
5. Ergebnisse - L-Systeme



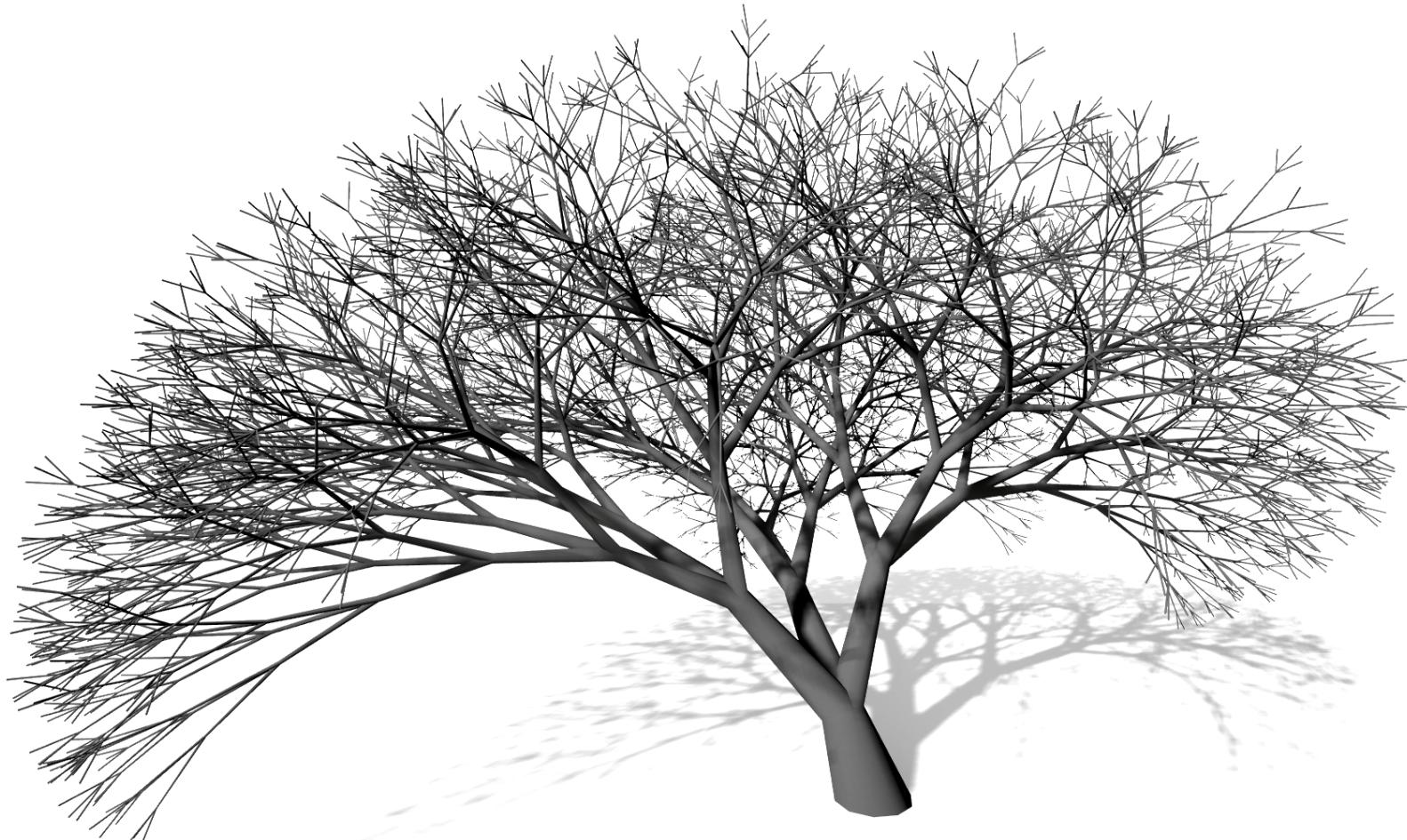
Monopodiales Wachstum



Sympodiales Wachstum

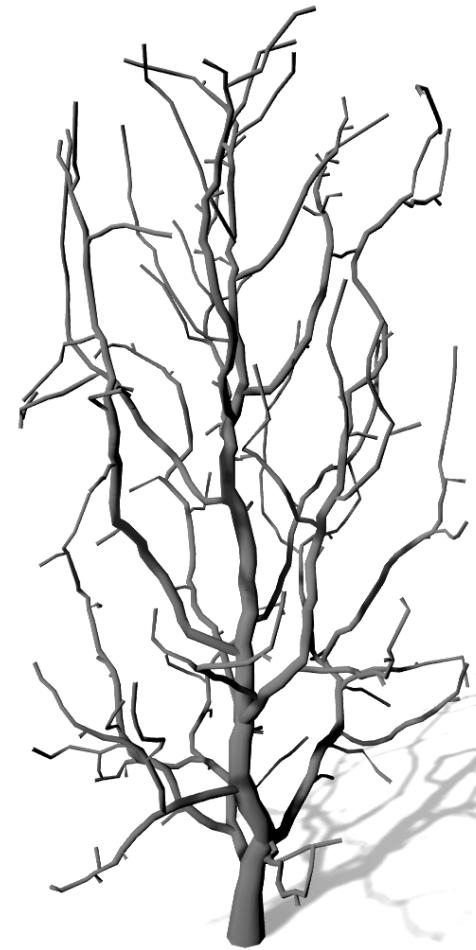
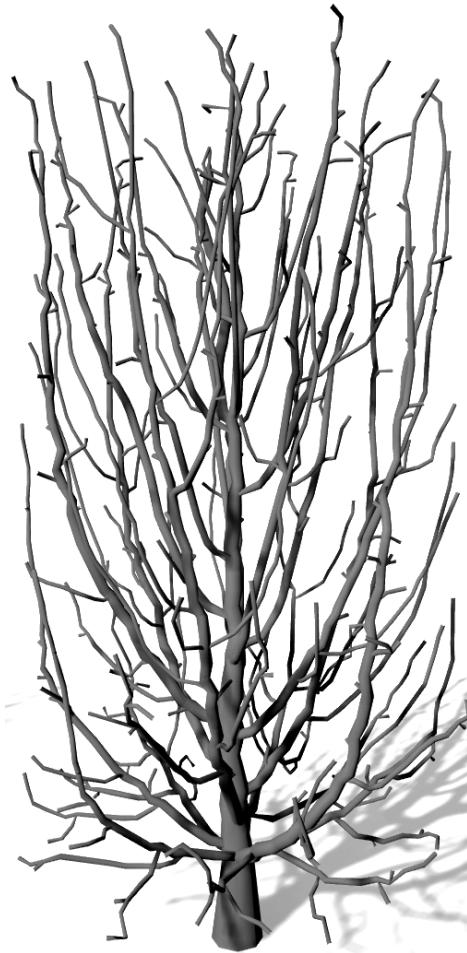


Ternäre Verzweigungen ohne Tropismus



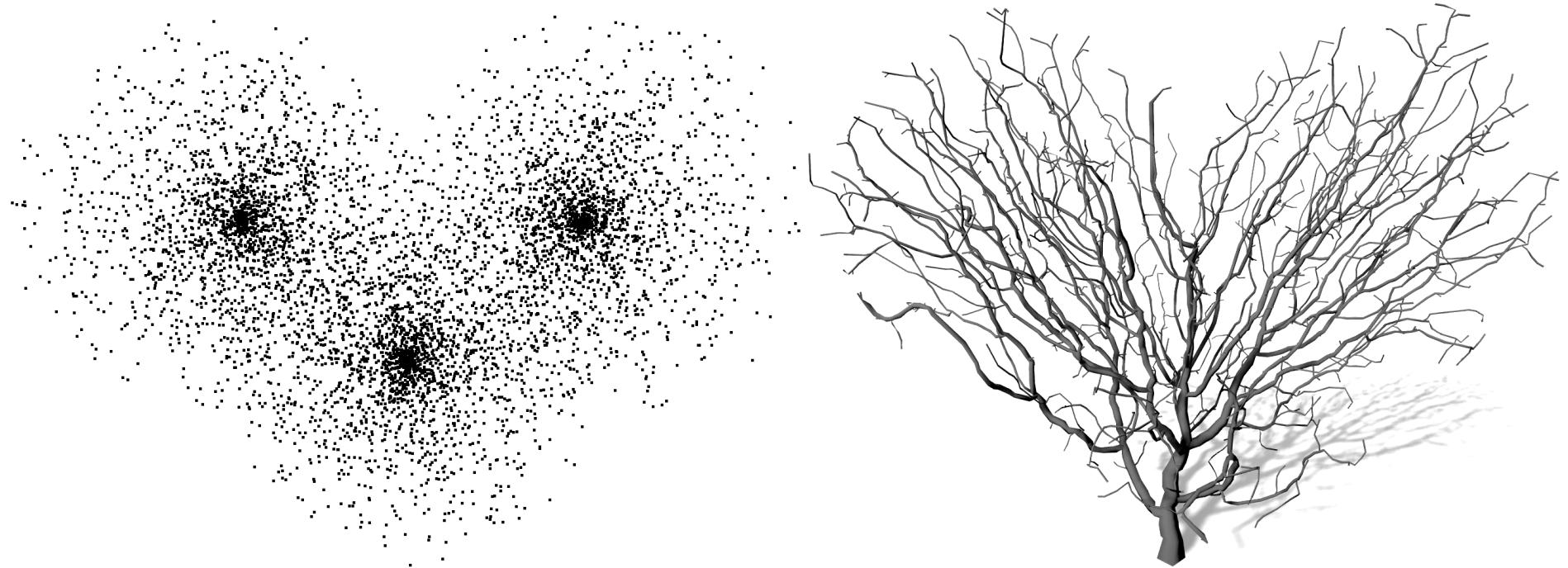
Ternäre Verzweigungen mit Tropismus: $\vec{T} = (0, -0.5, -1)^T$, $e = 0.5$

5. Ergebnisse - Space Colonization Algorithmus



Form des Einflussbereiches und Anzahl der Einflusspunkte

5. Ergebnisse - Space Colonization Algorithmus



Zusammenführen mehrerer Einflussbereiche

5. Ergebnisse - Space Colonization Algorithmus



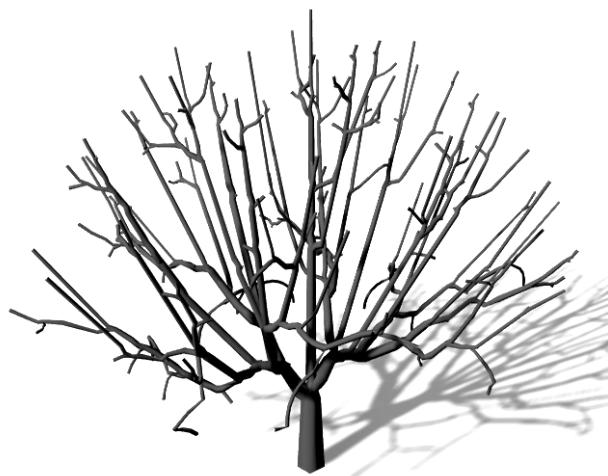
Unterschiedliche Einflussradien

5. Ergebnisse - Space Colonization Algorithmus



Einfluss des Minimalradius

5. Ergebnisse - Space Colonization Algorithmus



Optimale Verwendung des gewichteten Wachstums