

Computer Generated Trees

Implementation with Haskell
Using Voxel Geometry

Literary Research

ICE Team

Voxel Kernel

Voxel Geometry

- Typed Octree Structure
- Solid Geometry
 - As opposed to Sparse Geometry
- Three Node Types
 - Filled <some_type>
 - Empty
 - Node <some_type>
- Coordinate Space: Positive Integers

Ntree

- General Tree Structure with Dimension Parameter:
 - `data NTree a = Ntree {dim :: Int, size :: Int, nodes :: [NNode a]}`
 - 1 -> Binary Tree
 - 2 -> QuadTree
 - 3 -> Octree
- QuadTree needed for some calculations
- Dimension known at compile-time

Serialization

- Binary Coding:
 - 0: Empty cell
 - 10: Node, Children follow
 - 11: Leaf, Serialized Cell Data follows
- Leaf case:
 - 0 for Empty
 - 1 + <data> for Filled
 - Type () for Geometry only data (no payload)

Creating Voxel Objects

- By Function
- `shapeCon :: Cube -> NodeType`
- `NodeType`
 - Filled
 - Empty
 - Both
- Both case: Create Nodes, recurse

Rasterizing Curves

- Input Points, Weights
- Solve Spline Equations (Time-Wise)
- Iterate In Time-Steps corresponding to one Unit
- Add Voxel to Shape
- Use Octree data type to encode Time-Parameter of Spline at the Voxels

Extruding Shapes

- Basic Extrusion:
 - `extrude :: Shape -> Shape -> Shape`
- Shape to Extrude and Points to Extrude from
- Advanced: Shape Constructor
 - `extrude :: Shape data -> (Vector -> data -> Shape a) -> Shape a`
- Constructor Can Apply Local Rotations etc
 - Use data in Voxels

Decreasing Resolution

- At a given Tree level L
- Check Nodes of Input, Copy one Voxel to Output
- Strategies:
 - 1) Only Copy one Voxel from lowest Tree level
 - 2) Copy Both case as Filled
 - 3) Copy Both case as Empty

Extrusion Process

- Start at highest Tree Level L
 - Decrease Resolution of Shape to Extrude From with Method 1
 - Decrease Resolution of Shape to Extrude with method 3
 - for all vector in extrudeFrom : add shape construct extrudeWith vector
 - Recurse with L-1
- Result: Reduced Overwriting of Fine Details

Shape Modifications

- Boolean operations are quite trivial
- Simplify Octree
 - All eight nodes Filled: delete Children, set Parent to Filled
 - Same with empty
- Extract Surfels
 - Iterate Octree, output all voxels adjacent to empty space

Utilizing Mesh Input

- Octree Sort algorithm for Triangles
- Check Triangle equations against Rasterization Cubes
- Result is Sparse Voxel Octree

Fill Up SVO

- Sweep Virtual Plane through Bounding Cube
 - Check for intersections with material
- On lose material intersection
 - Mark Voxel in Quadtree
- On get material intersection
 - Unmark
- While sweeping, Materialize marked Quadtree Voxels to Output Shape
- Add to original SVO

Surface Extraction

- Similar to Surfel Extraction
- For every Node Adjacent to empty space, create Surface
- Marching Cubes Method: $2^8=256$ Configs
- Look up Triangle List for Config, Move to correct position
- Texturing ?

Surface Simplification

- Smooth MC output
- Find edges, corners
- Generate LOD
- Employ Volumetric LOD
- Reduce foliage to double sided tris

Output

- For Debug, use File Format STL
 - Triangle List
 - No Texture
 - No Per-Vertex Normal
 - Very Simple
 - But Very Good Support by 3D Programs
 - Used for 3D Printing
 - This is basically CAD
 - Can write script for gearwheels etc.

Trees

Processing Chain

- XML Tree Species Definition
- Instantiation (Haskell Program)
 - Also generate Bone Animation Params for Wind
- Shape Definitions (likely also XML)
 - Extra Definitions for Animation Bones, Texturing etc.
- Shape Creation for LOD Levels
- Mesh Output

Further Input Data

- May be needed
- Can be
 - Mesh (will be rasterized)
 - Foliage etc.
 - Serialized Binary Octree/Quadtree w/ known Type
 - Texture for foliage, trunk, roots...

Tree Theory #1

- Deutsches Paper
- Building Blocks
 - Branch (angles, weights for thickness)
 - Horn (straight branch)
 - ...
- Linked in a Graph
- Recursions Allowed, with counters
- Instantiate Graph to generate Geometry

Tree Theory #2

- Modifiers for Geometry Constructors
 - Gravity pulls down Branches
 - Shadowing suppresses Branch growth
 - Death of Branches acts randomly over time
- Aging
 - Simulate growth of tree
 - Time-dependend graph instantiation
 - Generate seamlessly aging tree with time-dependent modifiers

Tree Theory #3

- Example: Branching
- Definitions
 - number of branches
 - size distribution of branches
 - branching angles
 - special constraints on certain branches
- Might also be possible with graphical probability editor

Issues in Model

- Skipping nodes in graph is not possible
 - Even on the stem of a big tree, there might be leaves.
- “Horn” type is not good for finer branching details
 - A stem should be defined as a sequence of branches which are all oriented in the same direction
 - Creation should be top-down and fractal.
 - Then extrude shapes at each point

Statistics

- Model in Paper quite deterministic and quantized
- Use Gaussian Distributions for every Decision
- Make distributions dependent of Current Depth
- Constrain Output to specific Ranges
 - Floating-Point Output
 - Integer Output

Scripting

- Inject GHCi functions in every decision
- Make relevant Information available
- Scriptable Modifiers
- Script Geometry Modifiers which operate on voxel data
 - Create bark and other fine detail

Further Applications

- Generate a brick wall
 - Create simple Shape where the wall will be (e.g. A cube)
 - Define recursive space partitioning function
 - At node level: create / extrude a brick, obeying geometrical constraints
- Same principle applies to every building

References

- Botanische Bäume
- Mesh Simplification, Edge Detection
- The Discretized Polyhedra Simplification (DPS)