Procedural Geometry

Jaanus Jaggo
Procedural geometry uses

- Sprite fonts
- 2D sprites geometrization
- Procedural landscapes
- Procedural flora
- Procedural destruction
- Meshing point clouds
- Procedural growth

Pandora from Avatar movie - uses a lot of procedural geometry
Sprite Fonts

Archmage Antonidas

Whenever you cast a spell, add a 'Fireball' spell to your hand.
Sprite fonts generation

!"#$%&'()*+,-./
0123456789;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ[]^_`
abcdefghijklmnopqrstuvwxyz{|}~
gold:0 wood:0 cristals:0 hp:10
2D sprite geometrization

- GPU can handle loads of vertices
- limited in **Pixel Fill-Rate**

![Graph showing performance comparison between Static SpritePolygon and Static Sprite](image)

**Performance:** (number of sprites rendered within 40 FPS)

<table>
<thead>
<tr>
<th></th>
<th>Static SpritePolygon</th>
<th>Static Sprite</th>
<th>Dynamic SpritePolygon</th>
<th>Dynamic Sprite</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPhone 6 plus</td>
<td>661</td>
<td>268</td>
<td>+146.64%</td>
<td>653</td>
</tr>
<tr>
<td>Samsung 9100</td>
<td>1080</td>
<td>404</td>
<td>+167.32%</td>
<td>780</td>
</tr>
<tr>
<td>rMBP early 2013</td>
<td>8851</td>
<td>2718</td>
<td>+225.64%</td>
<td>9538</td>
</tr>
</tbody>
</table>

Source: [http://discuss.cocos2d-x.org/t/new-feature-meshsprite-polygonsprite-updated/21153](http://discuss.cocos2d-x.org/t/new-feature-meshsprite-polygonsprite-updated/21153)
2D sprite geometrization

Unity does it automatically

Can we do better?
How it’s done procedurally?

Algorithm by Erik Grahm
Edge detection

Erosion (shrink)

substracted

Edge detection
Edge detection
Find closed path around the object

Problems: loops and deadlocks
Simplify the polygon

Convex hull algorithm
Simplify the polygon

Convex hull algorithm

kick out points that produce triangles
Triangulation


This algorithm can triangulate polygons with holes
Triangulation complexity

- Ear clipping method - $O(n^2)$
- Using monotone polygons $O(n \log n)$
- Can be done within $O(n)$ \(<-\) some complicated algorithms
UV’s

\[ u = \frac{x}{\text{width}}; \quad v = \frac{y}{\text{height}}; \]
Optimizing it further

2-layered approach

2 layers == 2 draw calls
Procedural landscape

- Height field terrain
- Vector field terrain
- Voxel based terrain
Heightmap based landscapes

- Main game engines -> built in

Height field vs. Vector field
Voxel based procedural landscapes
Smooth Voxels

http://nerdkingdomofficial.tumblr.com/post/119558446658/dev-blog-44-voxelize-this
Isosurface extraction

- Marching cubes
- Marching Tetrahedra
- Surface nets
- Dual contouring

Demo: http://mikolalysenko.github.io/Isosurface/

Marching cubes

$2^8 = 256$ different possibilities

Problem: some cases are ambiguous
Marching cubes

Break contour

Join contour
Marching Tetrahedra

Solves ambiguity problem

\[2^4 = 16\] different possibilities

Problem: may produce artificial “bumps”
Surface nets

Minecraft ;)

26
Comparison

- **Marching cubes**
  - + pretty fast
  - + most widely known
- **Marching tetrahedra**
  - + solves ambiguity problem
  - - much slower
  - - far larger meshes
- **Surface nets**
  - + fast
  - + slightly smaller meshes than first two
  - - non-manifold vertices

http://0fps.net/2012/07/12/smooth-voxel-terrain-part-2/
Procedural flora

- L-systems
- Superformula
L-systems

Formal method for describing self similar objects.

http://buildingindustry.org/l-systems
Superformula

Mathematical formula used to describe many complex shapes in nature.

\[ r(\varphi) = \left( \left| \frac{\cos\left(\frac{m\varphi}{4}\right)}{a} \right|^{n_2} + \left| \frac{\sin\left(\frac{m\varphi}{4}\right)}{b} \right|^{n_3} \right)^{-\frac{1}{n_1}}. \]

parameters: \( a = b = 1 \)
\( m, n_1, n_2, n_3 \)
Procedural destruction

Supports slicing

Keep cutting up objects as much as you like!

Assets in Unity
assets store
Shattering glass

before: https://www.youtube.com/watch?v=eQXCHfmsHQA

after: https://www.youtube.com/watch?v=7U6kPGgGe1s
Shattering glass

http://cowboyprogramming.com/2007/01/05/shattering-reality/
Voronoi diagrams

https://www.youtube.com/watch?v=XYs5WOc4Etg

http://forum.unity3d.com/threads/fracture-dynamic-destruction-system.177066/

Partitioning of a plane into regions based on distance to points (any point inside a region is closest to the origin)

Euclidean distance  Manhattan distance
Fortune’s algorithm

Sweep line algorithm for making voronoi diagrams

- Vertical sweep line moves left to right
- Beach line is a curve left to sweep line
- Left of beach line all points are known
- beach line progresses by keeping each parabola base exactly half way between points initially swept and the sweep line
- Beach line is maintained in **binary search tree**
- Future points are maintained in **priority queue**
- Algorithm repeatedly removes next event from priority queue and updates the data structure.

\[ O(n) \text{ events \& } O(\log n) \text{ time to process an event} = O(n \log n) \text{ time} \]

https://en.wikipedia.org/wiki/Fortune%27s_algorithm
More uses for voronoi
More uses for voronoi
Supernoi (multiple voronoi)

Meshing point clouds

https://www.youtube.com/watch?v=_RHXqG7HIEM

1. Subsampling
   a. reducing the dataset
2. Normal reconstruction
3. Surface reconstruction
   a. marching cube algorithms
   b. quadric edge collapse simplification
4. Recovering original color
5. Cleaning up and assessing

There are also other methods for example Voronoi filtering for steps 2 and 3

Procedural growth

https://www.youtube.com/watch?v=9HI8FerKr6Q

https://vimeo.com/27913396
Growing citys

A roadmap generated:

https://josauder.github.io/procedural_city_generation/
Growing citys

Rules:
## Growing rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Forward</th>
<th>Turn</th>
<th>pForward</th>
<th>pTurn</th>
</tr>
</thead>
<tbody>
<tr>
<td>grid</td>
<td>vert + prev_vec</td>
<td>vert + rot 90</td>
<td>1</td>
<td>0.09</td>
</tr>
<tr>
<td>organic</td>
<td>vert + random rot</td>
<td>vert + rot (60,120)</td>
<td>1</td>
<td>0.8</td>
</tr>
<tr>
<td>radial</td>
<td>vert + adjust vec</td>
<td>vert + adjust vec</td>
<td>1</td>
<td>0.07</td>
</tr>
<tr>
<td>minor road</td>
<td>vert + prev_vec</td>
<td>vert + rot 90</td>
<td>0.05</td>
<td>0.85</td>
</tr>
</tbody>
</table>
Vertex snapping
Thank you for listening

Here is a kitten for you!