AstroGen - Procedural Generation of Highly Detailed Asteroid Models

X. Z. Li, R. Weller, G. Zachmann
University of Bremen, Germany

cgvr.informatik.uni-bremen.de

ICARCV’15th, Nov 19-21 2018, Singapore
Motivation

- Low-quality data from earth observation
  - Radar
  - Telescope
- Virtual testbed simulations
  - Time and cost efficient
  - Autonomous operation
    - Long distance scheduling latency
Challenges

- How to generate diverse but similar asteroid surfaces (i.e. virtual testbed) for simulation?
- How to reuse the data from previous space missions?
Previous Work

- Procedural hydrology terrain [Génevaux 2013]
  - Underlying hydrographic network
  - User defined terrain features (mountain, ...)
- Procedural terrain with real-world data [Parberry 2014]
  - Design terrain with real elevation data
  - Terrain details with value noise
- Sparse representation of terrain [Guérin 2016]
  - Procedural landform features (primitives)
  - Sparse construction tree
Our Contribution

- Automatic asteroid model generation
  - Given a predefined similarity distance to generate a variety of asteroid models from the given model
  - Add terrain features on the surface easily
- High performance
  - Parallel GPU implementation
- Arbitrary Resolution
  - Implicit representation of a given model
Approach – Overview

- Parameter training
- Surface detail transfer

Prototype Mesh

Implicit Representation
\[ S = \{(x, y, z)|F(x, y, z) = T\} \]

Training Pipeline

Surface detail parameters
Approach – Training Pipeline

Step 1: **Rough Shape**
- Prototype Mesh
- Metaball Modelling
- Optimization
- Fitness function
- Implicit Representation \( \{(x, y, z) | F(x, y, z) = T \} \)

Step 2: **Surface Details**
- Optimization
- Fractal Noise
- Fitness function
- Implicit Representation
- No
Approach

- Implicit surface
  - Define a series equation $F$ and compute for each grid point $P$
    - Implicit surface $S = \{(x, y, z) | F(x, y, z) = T\}$
    - $T$ is the iso-value of the implicit surface

- Optimization
  - Change the parameters in $F$ to generate an infinite number of shapes
  - Particle swarm optimization [Samal 2007] with a fitness function leads to target result
Step 1: Metaball Modelling

- Prototype surface
- Metaballs define the isosurface (implicit surface $S$ with isovalue $T_0$) to approximate the prototype surface
  - Skeleton of spheres (Sphere Packing [Weller 2010])
  - Potential field
  - Blending
Step 1: Optimization

- Protosphere
  - \( n \) is the number of spheres in the prototype shape
- Potential function \( f(r_p) \)
  - \( a \) is the tension factor
  - \( b \) is the softness factor
- Blend function for each metaball
  \[
  f(r_p) = \left( \frac{f^m(r_{pA}) + f^m(r_{pB})}{2} \right)^{\frac{1}{m}}
  \]
  - \( m \) is the overlapping factor

Ground truth shape

Rough shape
Step 2: Fractal Noise – Perlin & Simplex

- Fractal terrain
  - 3D Perlin noise
    - Fractal (summation of noises on different octaves)
    - Self-similarity
  - 3D Simplex noise
    - Less directional artifacts
Step 2: Fractal Noise – Worley

- **Primitive - Craters**
  - **3D Worley noise**
    - Points for a distance field
    - Randomly distribute feature points $X$ in space
    - Noise value is the distance to the-closest point $x \in X$
Step 2: Optimization – Surface Details

- Optimization parameters

<table>
<thead>
<tr>
<th>Number of Parameters</th>
<th>Perlin</th>
<th>Simplex</th>
<th>Worley</th>
<th>Gradient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Frequency</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Octave</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Amplitude</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Coords_w</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Coords_b</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ \sum = 31 \]

\[
T = T_0 + weight \cdot \sum_{i=0}^{octave} amplitude \cdot perlin((2^ix, 2^iy, 2^iz) \cdot f \cdot \vec{w} + \vec{b})
\]

\[
+ \sum_{i=0}^{octave} simplex(...) + \sum_{i=0}^{octave} worley(...)
\]

- Fitness function
  - Compute histograms [Li 2017] for all models
  - Minimize the histogram’s Euclidean distance
Results – Itokawa

Model from photogrammetry (Source 1,780k vertices)

“Flat” surface (1,986k vertices)

“Medium” surface (2,173k vertices)

“Steep” surface (2,335k vertices)
Results – Transformed Low-Poly Asteroids

Asteroid Lutetia
(710k vertices)

Asteroid Ceres
(1,063k vertices)

Asteroid Stein
(778k vertices)
Conclusions

- Optimization-based generation of 3D asteroid look-alikes

Major contributions:
- Create infinite numbers of asteroid shapes similar to prototype shape
- Users control the similarity/dissimilarity distance to generate different shapes
- Create arbitrarily high resolution from low-poly models
- Can be easily implemented on the GPU

Limitations:
- The randomness of noise make it hard to control and generate particular patterns
Future Work

- More naturalness
  - AstroGen integrated with physically-based noise such as flow noise and curl noise
  - Incorporate with reinforcement learning or other optimization algorithm to improve the result
  - Different similarity measurements can be compared

- More applications
  - AstroGen in virtual testbed to verify vehicle design
  - Mascon based gravity computing

- Better mesh quality
  - Enhance the visual fidelity by using dual marching cubes
Thank you!

Q&A