Quiz 2

Victoria Deng

August 2020

Lecture 5 - Fractal Geometry and Landscapes

Fractal: A geometric shape generated using a series of recursive rules. They have a property called self similarity.

Self similarity: A pattern that repeats at different scales. A fern is a good example of this.

- Exact self-similarity: The strongest type of self-similarity. The fractal appears identical at different scales. Fractals defined by iterated function systems often display exact self-similarity.
- Quasi self-similarity: The fractal appears approximately (but not exactly) identical at different scales. Quasi-self-similar fractals contain small copies of the entire fractal in distorted and degenerate forms.
- Statistical self-similarity The weakest type of self-similarity. The fractal has numerical or statistical measures which are preserved across scales. Random fractals (eg. fractal landscapes) are examples of fractals which are statistically self-similar but neither exactly nor quazi-self similar.

Brownian Motion: Describes the random movement of small particles of matter in a fluid (liquid or gas). In the context of computer graphics, Fractional Brownian surfaces get smoother the larger the *Hurst Parameter* is.

Diamond Square Algorithm: Used to generate heightmaps from a 2D grid.

- Initializing: Start by setting the inital values for the corner points of the grid. Perform the diamond and square steps alternately until complete.
- Diamond Step: For each square in the array, set the midpoint of that square to be the average of the intersecting points plus a random value.
- Square Step: For each diamond in the array, set the midpoint of that diamond to be the average of the intersection points plus a random value.

Lecture 6 - Surface Rendering and Shading

Illumination: Determines how the light sources interacts with object surfaces (eg. intensity of the light that is being reflected)

Shading: Determines how to render the faces of polygons in a scene, given the illumination.

Light Source Models:

- Point Source: All light rays originate at a point, and radially diverge.
- Parallel Source: Light rays are all parallel, and can be modelled as a point source at an infinite distance (like the sun).
- Distributed Source: All light rays originate at a finite area in space. It models a nearby source, such as a flourescent light. Radial, but spread out (unlike a point source).

Shading Techniques:

- Flat Shading: No interpolation
- Smooth Shading: Linear change (interpolation)

Flat Shading: Same values are used to render an entire polygon. The result can be very 'block like' as an entire square would be white for highlight. Drawbacks are that it doesn't look smooth or realistic. Benefits is that it has a low computation cost, and can be realistic if polygons are small or the object is far.

Mach Band Effect: Exaggerates the contrasts between edges of the differing shades of colours as the human eye draws a boundary when they contact each other. I.e Two greys touching each other would look more stark than if they were seperated.

Smooth Shading: Light is computed at multiple points on each polygon, hence it has no Mach Band Effect. The two most popular methods of implementing smooth shading is Gouraud Shading and Phong Shading.

Gouraud Shading: Lighting is calculated for each polygon vertex, and colour is interpolated for the pixels in between. The result is that there is a linear change from vertices. The main issue is if the polygon shading is not linear(like a point source), or the surface is curved.

- Algorithm
- Determine the normal at each polygon vertex (the average of the normal of adjacent faces)
- Apply an illumination model to each vertex to calculate the vertex intensity
- Linearly interpolate the vertex intensities over the surface polygon

Phong Shading: Similar to Gouraud, but rather than being linear, does the shading calculating at every pixel. Slower and has a higher computational cost, but generally more realistic in handling highlights.

- Algorithm
- Determine the normal at each polygon vertex (average of the normal of adjacent faces)
- Linearly interpolate the vertex normals over the surface polygon
- Apply the illumination model along each scan line to calculate the intensity of each surface point

Texture Mapping: Maps a planar image (typically 2D) onto a 3D object. Has some issues, such as flat image going on a curved object, distortion or repetition if image is too small etc.

Texture Synthesis: Generate synthetic textures of arbitrary size on the characteristics of a sample input image (generally only works on random looking textures)

Displacement Mapping: Displaces each point on the surface, texture values gives the amound to move in direction normal to the surface. Eg. making a sphere spiky

Bump Mapping: Used to generate rough surfaces, without increasing the number of polygons. The surface itself does not change, but the shading makes it look like it did. Convincing at a distance, but not up close. Eg. Adding bumps to an orange, we can physically change the shape, or wrap a sphere in an orange texture.