



## Basic Terrain Generation



# PA199 Advanced Game Design

## Lecture 7 Terrain Generation

Dr. Fotis Liarokapis

06<sup>th</sup> April 2016



## Terrain Data



- Terrain data relates to the 3D configuration of the surface of the Earth
- Map data refers to data located on the surface of the Earth (2D)
- The geometry of a terrain is modeled as a 2 ½-dimensional surface



## Terrain Models



- Global terrain model
  - Defined by a single function interpolating all data
- Local terrain models
  - Defined on a partition of the domain into patches
    - They represent the terrain by means of a different function on each of the regions in which the domain is subdivided
- In general it is very difficult to find a single function that interpolates all available data
  - Usually local models are used

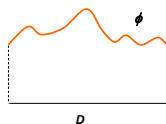


## Mathematical Terrain Models

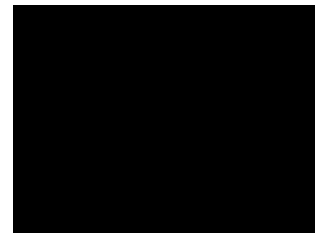


- A topographic surface or terrain can be mathematically modeled by the image of a real bi-variate function:  $z = \phi(x, y)$
- Defined over a domain  $D$  such that  $D \subseteq \mathbb{R}^2$
- The pair  $T=(D, \phi)$  is called a mathematical terrain model

Uni-dimensional  
profile of a  
mathematical  
terrain model



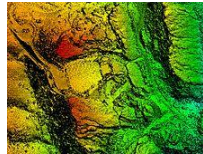
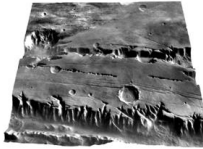
## Digital Terrain Models Video





## Digital Elevation Models (DEMs)

- DEM is set of regularly or irregularly spaced height values
  - Terrain elevation data
- No other information



## DEM Video



## Elevation Data Acquisition

- Elevation data can be acquired through:
  - Sampling technologies
    - i.e. on-site measurements or remote sensing techniques
  - Digitisation of existing contour maps
- Elevation data can be scattered (irregularly distributed) or form a regular grid
- The set of non-crossing lines can form a collection of polygonal chains

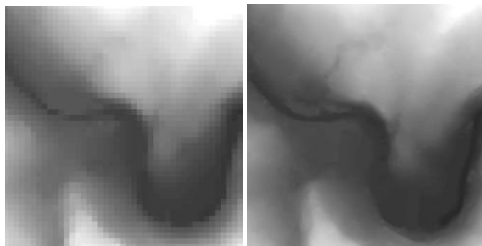


## UK DEM Data Sources

- Ordnance Survey: <http://www.ordnancesurvey.co.uk/>
  - Landform Panorama
    - Source scale: 1:50,000
    - Resolution: 50m
    - Vertical accuracy:  $\pm 3\text{m}$
  - Landform Profile
    - Source scale: 1:10,000
    - Resolution: 10m
    - Vertical accuracy:  $\pm 0.3\text{m}$



## Comparison



Landform Panorama

Landform Profile

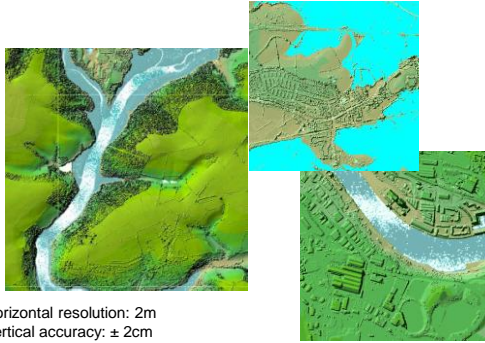


## Light Detection And Ranging (LIDAR)

- LIDAR is a remote sensing technology that measures distance by illuminating a target with a laser and analyzing the reflected light
  - Uses ultraviolet, visible or near infrared light to image objects
  - Can target a wide range of materials
    - i.e. non-metallic objects, rocks, rain, chemical compounds, aerosols, clouds, single molecules



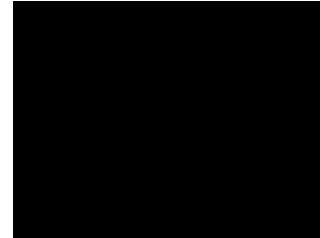
### LIDAR Examples



Horizontal resolution: 2m  
Vertical accuracy: ± 2cm



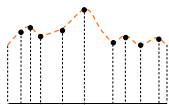
### LIDAR Video



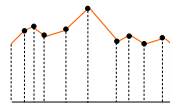
### Digital Terrain Models (DTMs)



- Digital terrain models represent an approximation of mathematical terrain models



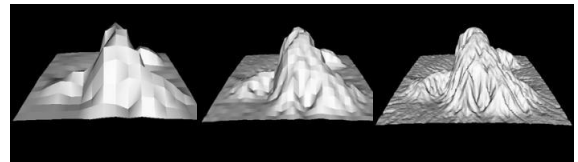
Sampled model



Digital terrain model



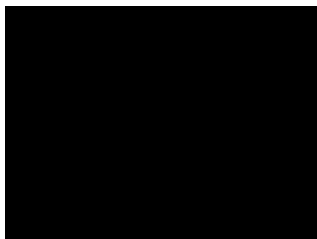
### DTM Examples



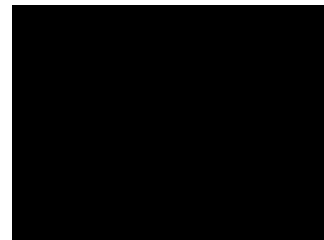
In general, a larger number of sampled points allows for a better representation: multiresolution terrain models



### DTM Video 1



### DTM Video 2





## DTM Types

- Polyhedral terrain models
- Gridded elevation models
- Contour maps



## Polyhedral Terrain Models (PTM)

- A PTM for a set of sampled points  $V$  can be defined on the basis of:
  - A **partition** of the domain  $D$  into polygonal regions having their vertices at points in  $V$
  - A **function  $f$**  that is linear over each region of the partition
    - The image of  $f$  over each polygonal region is a planar patch to guarantee continuity of the surface along the common edges



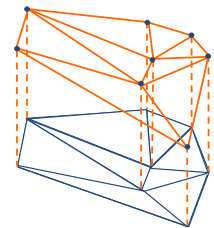
## PTM Properties

- Can be used for any type of sampled pointset
  - Regularly and irregularly distributed
- Can adapt to the irregularity of terrains
- Represent continuous surfaces

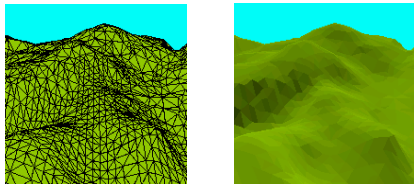


## Triangulated Irregular Networks (TINs)

- TINs are the most commonly used PTMs
  - Each polygon of the domain partition is a triangle



## TINs Example

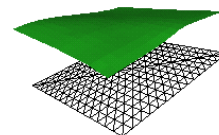


Example of a TIN based on irregularly distributed data



## TINs for Regular Data

- Regular sampling is enough in areas where the terrain elevation is more or less constant





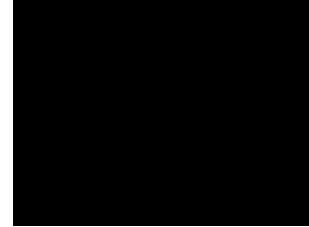
## TINs Properties



- They guarantee the existence of a planar patch for each region (triangle) of the domain subdivision (three points define a plane)
  - The resulting surface interpolates all elevation data
- The most commonly used triangulations are Delaunay triangulations



## TIN Video



## Why Delaunay Triangulations



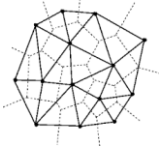
- They generate the most equiangular triangles in the domain subdivision
  - Thus minimising numerical problems
    - e.g. Point location
- Their Dual is a Voronoi diagram
- Therefore, some proximity queries can be solved efficiently



## Why Delaunay Triangulations .



- It has been proven that they generate the best surface approximation independently of the  $z$  values
  - In terms of roughness
- Several efficient algorithms to calculate them exist!



## Delaunay Triangulations



- Intuitively: given a set  $V$  of points, among all the triangulations that can be generated with the points of  $V$ , the Delaunay triangulation is the one in which triangles are as much equiangular as possible
  - Delaunay triangulations tend to avoid long and thin triangles



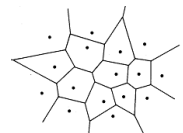
Does  $P$  lie inside  $t$  or on its boundary?



## Voronoi Diagrams



- Given a set  $V$  of points in the plane, the Voronoi Diagram for  $V$  is the partition of the plane into polygons such that each polygon contains one point  $p$  of  $V$  and is composed of all points in the plane that are closer to  $p$  than to any other point of  $V$





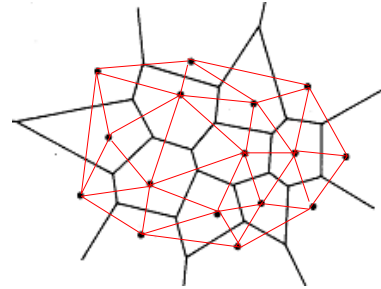
## Voronoi Diagrams .



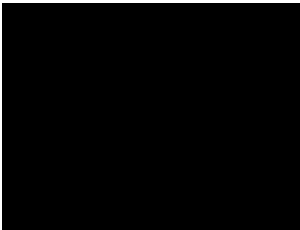
- Property
  - The straight-line dual of the Voronoi diagram of  $V$  is a Delaunay triangulation of  $V$
- Dual
  - Obtained by replacing each polygon with a point and each point with a polygon
- Connect all pairs of points contained in Voronoi cells that share an edge



## Voronoi Diagrams Example



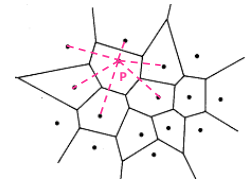
## Voronoi Diagrams Example



## Voronoi Diagrams Usage



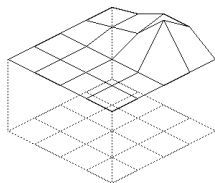
- Used as underlying structures to solve proximity problems:
  - Nearest neighbor
    - What is the point of  $V$  nearest to  $P$ ?
  - $K$ -nearest neighbors
    - What are the  $k$  points of  $V$  nearest to  $P$ ?



## Gridded Elevation Models



- A Gridded Elevation Model is defined on the basis of a domain partition into regular polygons



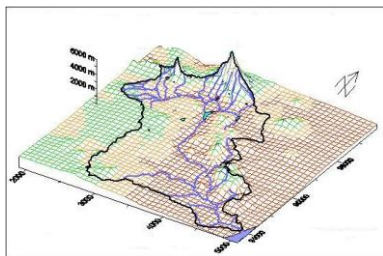
## Regular Square Grids (RSGs)



- The most commonly used gridded elevation models are Regular Square Grids (RSGs)
  - Where each polygon in the domain partition is a square
- The function defined on each square can be a bilinear function interpolating all four elevation points corresponding to the vertices of the square



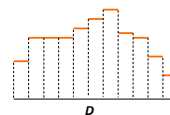
## RSG Example



## RSG Stepped Model

- Alternatively, a constant function can be associated with each square (i.e., a constant elevation value)
- This is called a stepped model
  - It presents discontinuity steps along the edges of the squares

Unidimensional profile  
of a stepped model



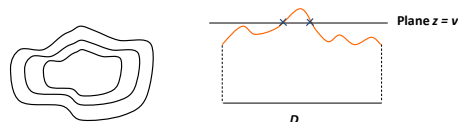
## TINs vs RSGs

- Both models support automated terrain analysis operations
  - RSGs are based on regular data distribution
  - TINs can be based both on regular and irregular data distribution
- Irregular data distribution allows to adapt to the “variability” of the terrain relief
  - More appropriate and flexible representation of the topographic surface



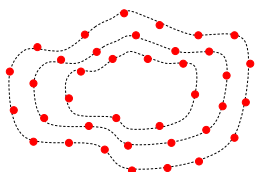
## Digital Contours

- Given a sequence  $\{v_0, \dots, v_n\}$  of real values, a digital contour map of a mathematical terrain model  $(D, \phi)$  is an approximation of the set of contour lines
- $\{(x, y) \in D, \phi(x, y) = v_i\} \quad i = 0, \dots, n$



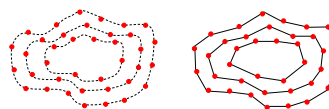
## Digital Contour Maps

- Contours are usually available as sequences of points



## Digital Contour Maps .

- A line interpolating points of a contour can be obtained in different ways
- Typical examples
  - Polygonal chains
  - Lines described by higher order equations





## Digital Contour Maps Properties



- They are easily drawn on paper
- They are very intuitive for humans
- They are not good for complex automated terrain analysis



## Problems with DEMs



- Issues worth considering when creating/using DTMs
  - Quality of data used to generate DEM
  - Interpolation technique
  - Give rise to errors in surface such as:
    - Sloping lakes and rivers flowing uphill
    - Local minima
    - Stepped appearance
    - etc



## Example Applications



- Visualisation
  - Terrain and other 3D surfaces
- Visibility analysis
  - Intervisibility matrices and viewsheds
- Hydrological modeling
  - Catchment modeling and flow models
- Engineering
  - Cut & fill, profiles, etc



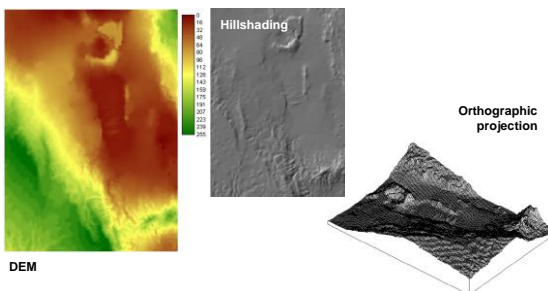
## Terrain Visualisation



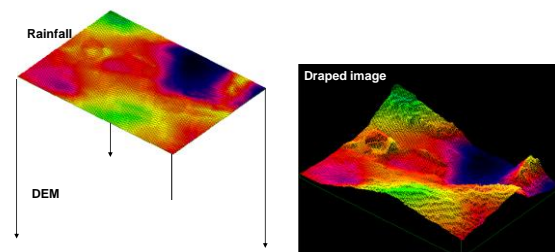
- Analytical hillshading
- Orthographic views
  - Any azimuth, altitude, view distance/point
  - Surface drapes (point, line and area data)
- Animated 'fly-through'
- What if? modeling
  - Photorealism
  - Photomontage
  - CAD



## Examples of Hillshading and Orthographic Projection



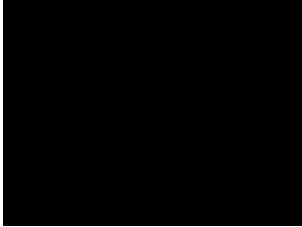
## Example Surface Drape







## Hillshading Video



## Fractal Terrain Generation



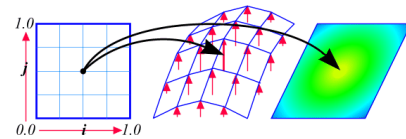
## Terrain Map

- Height Map:  $z = f(x, y)$ 
  - $x$  and  $y$  are sampled on a 2D integer grid
- Real data
  - Satellite, Elevation maps (previous lecture)
- Synthetic
  - Texture map, Noise functions



## Terrain Map .

- Connect samples into a mesh



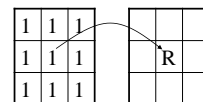
## Fake Terrain

- Generate the height-field
- Random process
  - This can be controlled
- Reflects 'realistic' terrain in some way



## Random Terrain

- Simple:
  - $\text{Terrain}(x,y) = \text{rand}(\text{MAX\_HEIGHT})$
  - Results in random noise
- Next step:
  - Smooth the terrain generated above
  - Finite Impulse Response (FIR) filter:





## Procedural Modeling With Fractals

- Procedural Modeling
  - Compute geometry “on-the-fly”
- Fractals
  - Model Natural Phenomena - Self Similarity
    - Mountains, fire, clouds, etc
  - Scales to infinity
    - Add or “make up” natural looking details with mathematical tools



## Fractals - A Definition

- A geometrically complex object, the complexity of which arises through the repetition of some shape over a range of scales
  - Sufficient definition for describing terrains



A hybrid multifractal made from Worley's Voronoi distance basis



## Fractals in Nature

- Fractals are common in:
  - Mountains, clouds, trees, turbulence, circulatory systems in plants and animals
- Wide variety of other phenomena such as:
  - Noise in transistors
  - Fluctuations in river fluxes



## Fractal Geometry

- Fractal geometry is very powerful!
  - But not sufficient for describing the complex forms found in Nature
- Mathematics are simple
  - Based on the Euclidean geometry of lines, planes, spheres and cones
- Fractal geometry has very little use in describing man-made objects

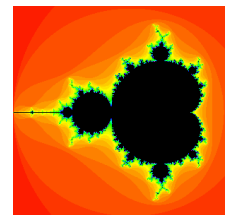
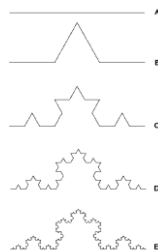


## Fractal Properties

- Two properties:
  - Self-similarity
  - Fractal Dimension



## Self-Similarity





## Fractal Dimension



- Euclidean dimensions
  - 1, 2, 3, 4, ...
- Fractal
  - 1.2342, 2.7656
- Measure of detail or roughness of a fractal
  - $D = (\ln N)/(\ln 1/s)$



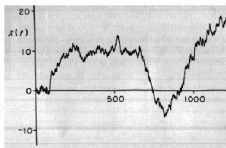
## Brownian Motion



- One of the most realistic ways of representing natural objects is the Brownian motion method because of the ability of creating peculiar curves
- Brownian method has been used to describe the chaotic and random manner in which a particle moves in a fluid
  - Application in terrains



## Brownian Motion Examples



Graph of a Brownian Sample Function



A simulation of a Brownian Path in 2-Dimensions



## Fractional Brownian motion (fBm)



- fBm consists of steps in a random direction and with a step-length that has some characteristic value
  - Also known as the Random Walk Process
  - Hence the random walk process
- A key feature to fBm is that if you zoom in on any part of the function you will produce a similar random walk in the zoomed in part



## Differences



- The main difference between fBm and regular Brownian motion is that while the increments in Brownian Motion are independent they are dependent in fBm
  - This dependence means that if there is an increasing pattern in the previous steps, then it is likely that the current step will be increasing as well



## Code Example of fBm



```
total = 0.0f; //for each pixel, get the value

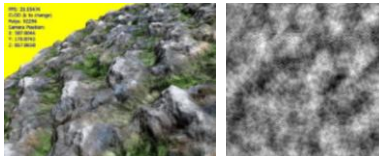
frequency = 1.0f/(float)hgrid;
amplitude = gain;

for (i = 0; i < octaves; ++i)
{
    total += noise((float)x * frequency,
                 (float)y * frequency) * amplitude;
    frequency *= lacunarity;
    amplitude *= gain;
}
map[x][y]=total; //now that we have the value, put it in
```



## Terrain using fBm Noise

- A simple approach is to generate a heightmap using fBm noise
  - Results look ok but not very realistic
    - Since fBm is homogeneous and isotropic

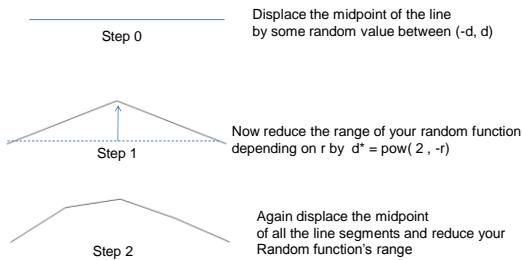


## Midpoint Displacement 1D

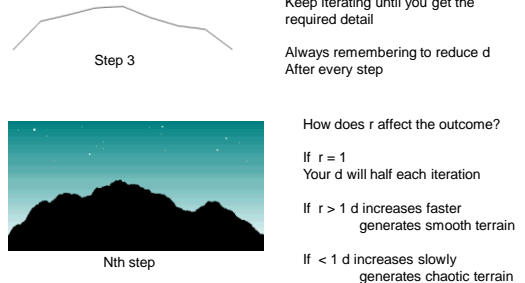
- Type of polygon subdivision algorithm, also a fractal function
- Created to simulate tectonic uplift of mountain ranges
- One of its main input parameters is the roughness constant  $r$



## Midpoint Displacement 1D .



## Midpoint Displacement 1D ..



## Diamond - Square Algorithm

- Also called the cloud fractal, plasma fractal or random midpoint displacement
- The 2D version of the original Midpoint displacement algorithm
  - Therefore it also has a roughness constant
- The algorithm works best if it is run on square grids of width  $2^n$ 
  - This ensures that the rectangle size will have an integer value at each iteration

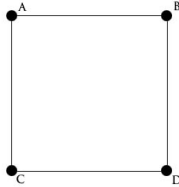


## Diamond - Square Algorithm Example



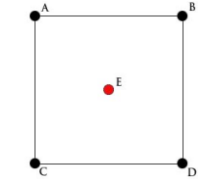
## Diamond - Square Algorithm Methodology

- Algorithm starts with a 2 x 2 grid
- The heights at the corners can be set to either:
  - Zero, a random value or some predefined value



## Diamond - Square Algorithm Methodology .

- The first step involves calculating the midpoint of the grid based on its corners and then adding the maximum displacement for the current iteration
  - Called the Diamond step
- Because if you render this terrain you will see four diamond shapes

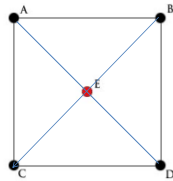


$$E = (A+B+C+D)/4 + \text{Rand}(d)$$

Rand(d) can generate random values between -d and +d

## First Step: Diamond Step

- The first step involves:
  - Calculating the midpoint of the grid based on its corners and then
  - Adding the maximum displacement for the current iteration

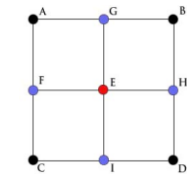


$$E = (A+B+C+D)/4 + \text{Rand}(d)$$

Rand(d) can generate random values between -d and +d

## Second Step: Square Step

- Calculate the midpoints of the edges between the corners
- Since the first iteration is complete, now d is reduced by:
  - $d *= \text{pow}(2, -r)$ 
    - where r is the roughness constant

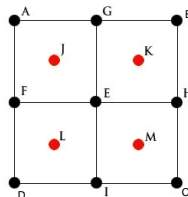


wrapping  
 $G = (A+B+E)/4 + \text{rand}(d)$   
 $H = (B+D+E)/4 + \text{rand}(d)$   
 $I = (D+C+E)/4 + \text{rand}(d)$   
 $F = (A+C+E)/4 + \text{rand}(d)$

Non-wrapping  
 $G = (A+B+E)/3 + \text{rand}(d)$   
 same for H,I,F

## Third Step: Second Iteration

- Start the second iteration
- Again perform the diamond step



$$J = (A+G+F+E)/4 + \text{rand}(d)$$

$$K = (G+B+E+H)/4 + \text{rand}(d)$$

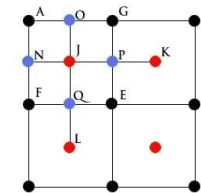
$$L = (F+E+D+I)/4 + \text{rand}(d)$$

$$M = (E+H+I+C)/4 + \text{rand}(d)$$

Remember this d is smaller than the one in the first iteration

## Fourth Step: Square Step

- Do the square step and continue subdividing
  - Until you reach the desired level of detail



wrapping  
 $O = (A+G+J+I)/4 + \text{rand}(d)$   
 $P = (J+G+K+E)/4 + \text{rand}(d)$   
 $Q = (J+E+L+F)/4 + \text{rand}(d)$   
 $N = (A+F+J+I)/4 + \text{rand}(d)$

Non-wrapping  
 $O = (A+G+J)/3 + \text{rand}(d)$   
 $N = (A+F+J)/3 + \text{rand}(d)$



## Diamond - Square Algorithm Summary



- While length of square sides  $> 0$ 
  - Pass through the whole array and apply the diamond step for each square
  - Pass through the whole array and apply the square step for each diamond
  - Reduce the range of the random displacement



## Smoothness of Terrain



- Lorentz and Gaussian distributions for the height array can control the smoothness of the terrain

$$height = y = (random\ number) \times \frac{D^2 / 8}{(x - x_0)^2 + (z - z_0)^2 + D^2 / 8}$$

- where  $(x_0, z_0)$  is the position of the peak and  $D^2$  is the length of the square
- The value of the width affects the smoothness of the terrain
  - By decreasing the width of the bell-shaped distribution the terrain becomes steeper



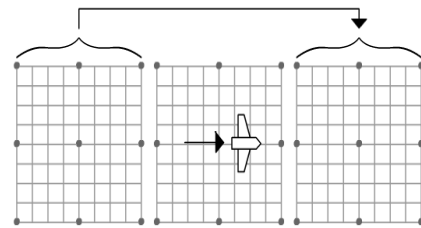
## Outer Boundaries



- Three solutions can be considered:
  - Large terrain
    - The user would never reach the outer boundary
  - Loop the old terrain
    - When the user reaches a boundary the user reenters the map on the opposite side
  - Infinite terrain
    - When the user reaches a boundary an extension to the map is added



## Loop the Old Terrain



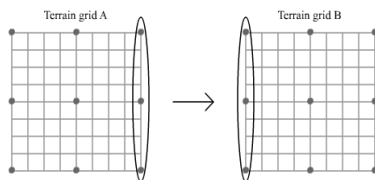
Upon reaching the right side of the landscape, a tile of terrain is moved in front of the player to provide the illusion of endless terrain



## Infinite Terrain



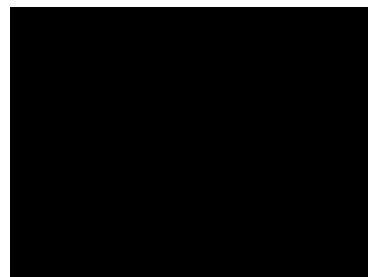
Upon reaching the right side of terrain grid A, the values of the far-right points are copied to the far left points of terrain grid B



The other points of terrain grid B are then calculated via randomization and mid-point displacement, as done for grid A



## OpenGL Fractal Terrain Video





## Other Techniques



- Cracked terrains for dry lakes and riverbeds
- Throw random points onto the plane
- Construct the Voronoi diagram and use the graph to etch into the terrain



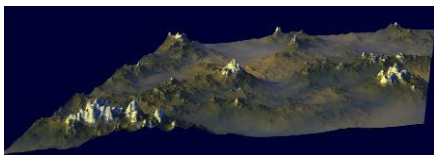
## Cracked Terrains



## Multifractals



- A better choice would be to use Multifractals
  - Proposed by Kenton Musgrave
- These are fractals whose dimension/roughness varies with location



## Multifractal Code Example



```

/*
 * Procedural multifractal evaluated at "point."
 *
 * Parameters:
 * "H" determines the highest fractal dimension
 * "lacunarity" is gap between successive frequencies
 * "octaves" is the number of frequencies in the fBm
 * "offset" is the zero offset, which determines multifractality
 */
double
multifractal( Vector point, double H, double lacunarity,
              int octaves, double offset )
{
    double    value, Noise();

    value = 1.0;

    for (int i=0; i<octaves; i++) {
        value *= (Noise( point ) + offset) * pow( lacunarity, -H*i );
        point += lacunarity;
    }
    return value;
}

```



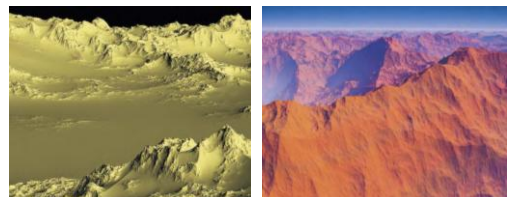
## Other Multifractals



- Hybrid multifractals
  - Called hybrid because they are both additive and multiplicative multifractals
- Ridged multifractals
  - Similar to Perlin's turbulence noise
  - They calculate  $1 - \text{abs}(\text{noise})$  so that the resulting "canyons" from  $\text{abs}(\text{noise})$  become high ridges



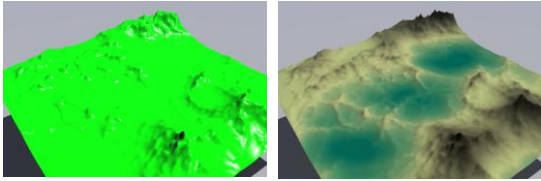
## Other Multifractal Examples



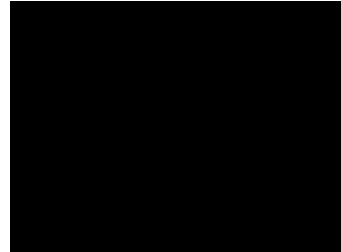
Ridged multifractal terrains : taken from Texturing and Modeling:  
A Procedural Approach pg 518 (left) pg 480(right)



## Multifractal with a 1D Texture



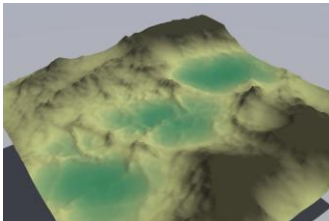
## Multifractal Video



## More Controls



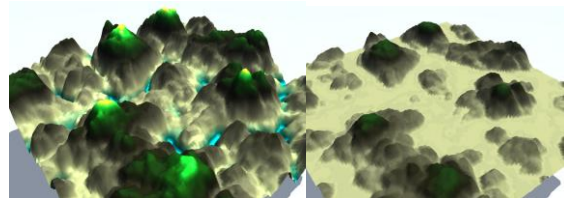
- To create plateaus, add a min function to flatten out high areas



## More Controls .



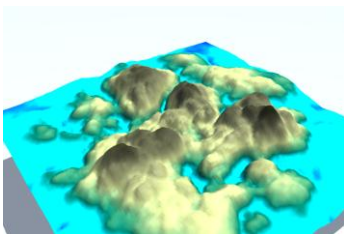
- To create a plain, add a max function to flatten out low areas



## More Controls ..



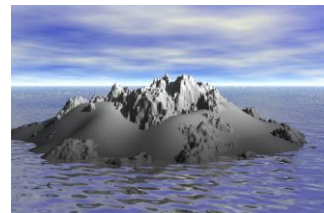
- Multiply by a Gaussian filter to limit the mountain range



## More Controls ...



- To create valleys, create a lower amplitude and rather smooth terrain to use as the max operator

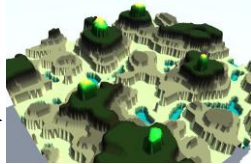
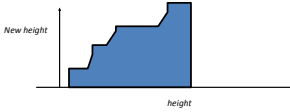






## Ridges

- Quantize a terrain to create ridges
  - Use directly or as the min function
  - Can also be done as a transfer function that maps  $f(x) \rightarrow g(x)$



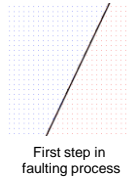
## Fault Line Algorithm

- This algorithm is a very simple one, yet its results, although not the best, are pretty good
- The technique is not limited to planar height fields, being also applicable to spheres to generate artificial planets
  - Can also approximate real world terrain features such as escarpments, mesas, and seaside cliffs



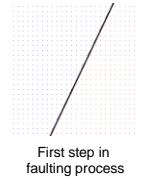
## Fault Line Algorithm .

- Start with a planar height field
  - All points have zero height
- Then select a random line which divides the terrain in two parts
  - The points to one side of the line will have their height displaced upwards
  - The points on the other side will have their heights displaced downwards

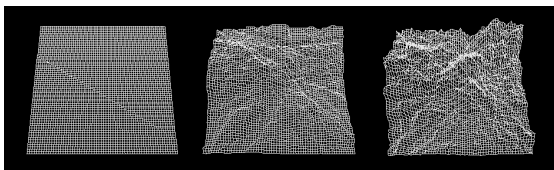


## Fault Line Algorithm ..

- The red points will have their height decreased, whereas the blue points will have their height increased
  - So the terrain has two distinct heights
- If we keep dividing the terrain can get something that has valleys, mountains and so on



## Fault Line Algorithm Examples



1 iteration

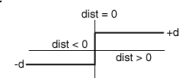
100 iterations

400 iterations



## Generating Fault Lines in a Height Field Grid

- Randomly pick two grid points  $p_1$  and  $p_2$
- Calculate the line between them
- Go through all the points in the height field and add or subtract an offset value depending on what side of the line they are located
- Before the next fault is drawn reduce the range of the offset by some amount

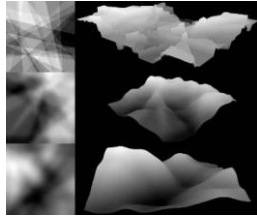




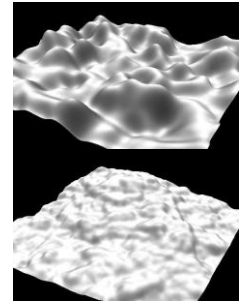
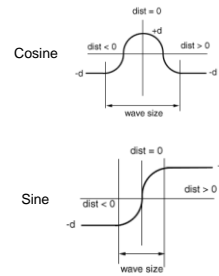
## Filtering Height Fields



- Height fields generated by this algorithm need to be filtered to look more realistic
  - A low pass filter can be used



## Variations to the Fault Line Algorithm



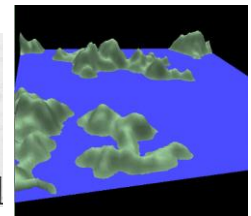
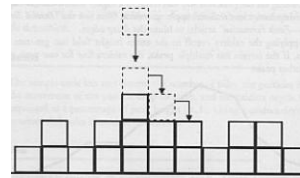
## Particle Deposition



- Simulates volcanic mountain ranges and island systems
  - Drop random particles in a blank grid
  - Determine if the particle's neighboring cells are of a lower height
    - If true, increment the height of the lowest cell
    - Keep checking its surrounding cells for a set number of steps or until it is the lowest height among its surrounding cells
    - If not, increment the height of the current cell



## Particle Deposition .



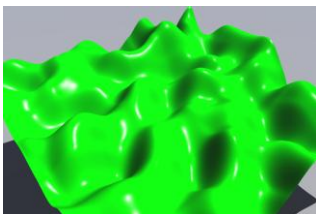
Generated after 5 series of 1000 iterations



## Perlin Noise



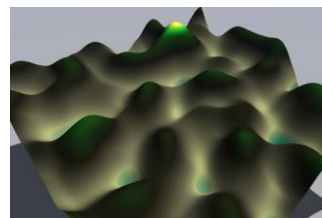
- Using a sampling of 2D Perlin Noise provides smooth hills



## Terrain Coloring



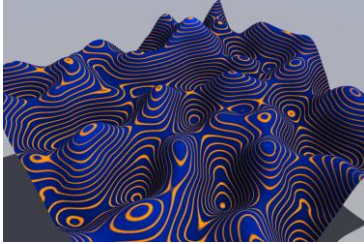
- Using a 1D texture map based on the altitude can provide many useful mapping





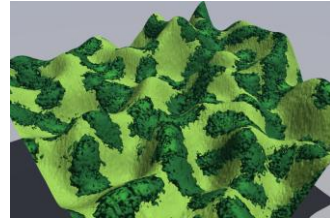
## Terrain Coloring .

- Striped 1D texture map



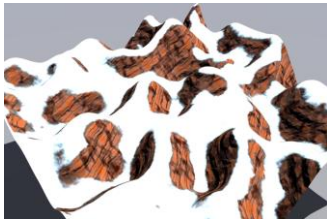
## Terrain Coloring ..

- Using a 2D texture map provides richer detail, but is independent of the terrain



## Terrain Coloring ...

- More advanced coloring is based on altitude and slope



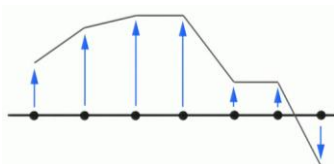
## Rolling Hills

- Scaling in one dimension gives smooth rolling hills



## Height Fields Issues

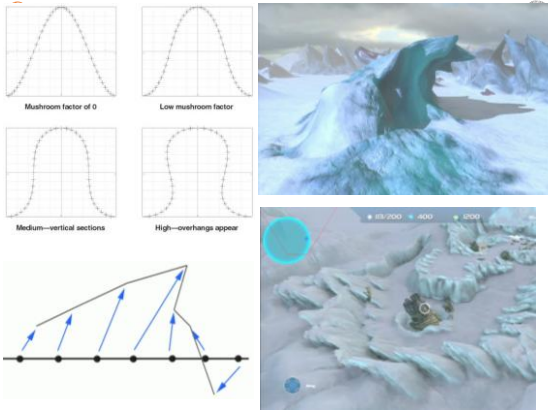
- They cannot generate overhangs or caves



## Height Fields Potential Solutions

- “Mushrooming” effects that involve the manipulation of vertex normals
  - To render height field textures with overhangs
- The game Halo Wars implemented a new type of height field called a vector height field
  - Stored a vector to displace a vertex instead of a height value





### Realistic Terrain Video 1



### Realistic Terrain Video 2



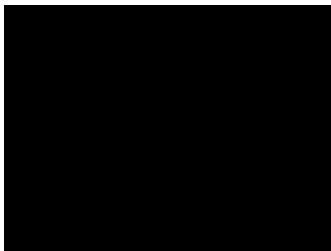
### Erosion



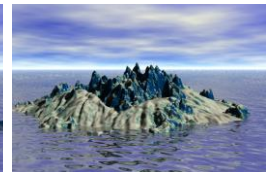
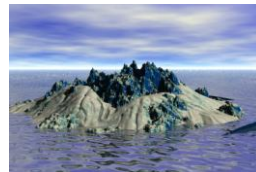
- Hydraulic Erosion
  - Water (rain) depositing to settle in height field
- Thermal Weathering
  - “Any material that knocks material loose, which then falls down to pile up at the bottom of an incline.”



### Thermal Erosion Video



### Erosion Examples

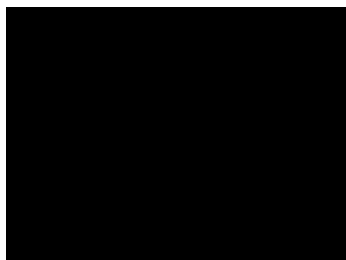


Erosion by Water

Erosion by Wind and Water



## Hydraulic Erosion Video



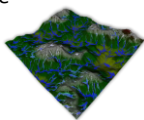
## Case Study - Fractal Nature



## Procedural Fractal Nature



- Simulate a selection of naturally occurring phenomena into a user friendly tool
- Provide a set of techniques that can be used for creating realistic terrains for games
  - i.e. flight simulators, action and strategy games
- Emphasis on:
  - Generating a realistic random terrain
  - Implement hydraulic and thermal erosion
  - Evaluate the tool



## Architecture



- Developed on Unity 3D game engine
  - Can export the terrain into a file format
- Two visualisation modes:
  - 'Flight Simulator' and 'Free View'



## GUI Overview



Filters menu      Erosion menu      Mesh control menu



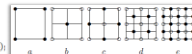
## Fractal Turbulence



- Firstly define a terrain based on the Diamond-Square subdivision:

$$\text{newValue} = \frac{\text{neighborsSum}}{\text{neighborsCount}} + \text{rand} * \text{roughness};$$

$$\text{rand} = \text{Random.Range}(-\text{randomValue}, \text{randomValue});$$



- Next the centre of the selected area is heightened

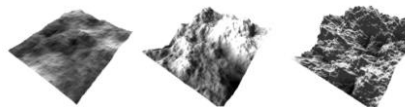
for each iteration:

$$\text{newValue} = \frac{\text{neighborsSum}}{4} + \text{rand} * \text{roughness};$$

$$\text{rand} = \text{Random.Range}(-\text{randomValue}, \text{randomValue});$$

$$\text{randomValue} = \text{randomValue} / 2;$$

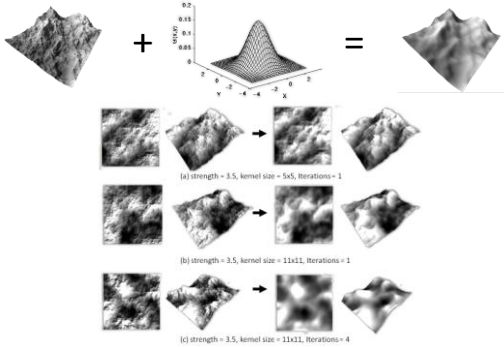
$$\text{roughness} = \text{roughness} / 2;$$



Three gradual values for the 'roughness' parameter and a mesh size of 192x192

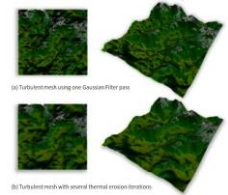


### Terrain Filtering



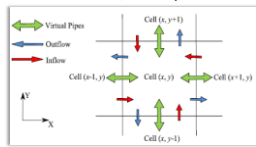
### Thermal Erosion

- Thermal erosion effects:
  - Appearance of accentuated slopes
    - Present natural-looking ridges
  - Flat plateaus
    - Where sediment tends to deposit
- Implementation based on (Musgrave et al., 1989), (Jako and Toth, 2011)



### Hydraulic Erosion

- Hydraulic erosion refers to the natural process in which the motion of fluids (specifically water) produces mechanical weathering over the soil
  - (Mei et al, 2007), (Anh et al., 2007), (Štava et al., 2008), (Jákó and Tóth, 2011)

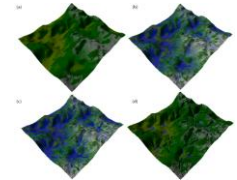


Virtual pipes representation (Mei et al, 2007)

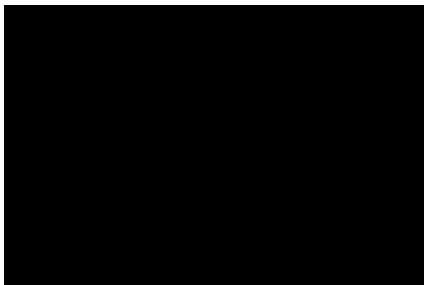


### Rain Simulation & Evaporation

- The rainfall causes the water levels to rise up at random positions
- Water movement by allowing water to escape
  - Approximation of the sediment amount that is to be eroded
- Water evaporation based on environmental heat



### Video



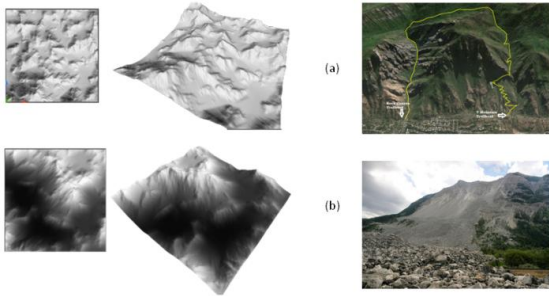
### Evaluation

- Qualitative evaluation with 12 expert users
  - 6 academics with computer graphics/virtual reality background
  - 6 industry professionals working in game companies
- Three stage evaluation:
  - GUI exploration
  - Simple flight simulator game
  - Comparison with real landscapes





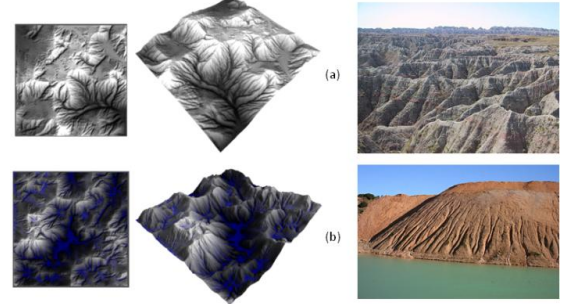
## Thermal erosion evaluation



(a) Medium temperature values comparison (b) High temperature values comparison



## Hydraulic erosion evaluation



(a) Low density + heavy erosion comparison (b) Low density + high viscosity comparison



## Academics Feedback

- Improve the visual realism of the terrain
  - More accurate algorithms for hydraulic erosion
  - Making the software open-source
  - Compare these methods with Perlin noise generator
  - Port the implementation (or part of it) on GPUs
  - Implement fluid simulation for the representation of sea, rivers and lakes
  - Stochastic smoothing algorithms to make the landscapes look more realistic



## Industry Feedback

- The level of realism quite satisfying and appropriate for the generation of interactive video games
  - Strong similarities with real life landscapes
- All of them stated that the generated terrains cannot be used without manual editing into modern games
  - Larger studios mentioned that they are using similar procedural approaches but then they perform a lot of editing
  - Independent studios stated that they could use the generated terrain as they are
- More procedural elements such as different types of trees and plants
- Better texturing techniques



## Conclusions

- Real terrain data still used in games
  - Very expensive
  - Need a lot of manual editing
- Procedural are getting more and more attention
  - More work is required to make them more realistic



## References

- <https://www.google.co.uk/#q=ordnance+survey>
- <http://ned.usgs.gov/>
- [http://en.wikipedia.org/wiki/Digital\\_elevation\\_model](http://en.wikipedia.org/wiki/Digital_elevation_model)
- [http://wiki.gis.com/wiki/index.php/Contour\\_line](http://wiki.gis.com/wiki/index.php/Contour_line)
- [http://wiki.gis.com/wiki/index.php/Digital\\_Elevation\\_Model](http://wiki.gis.com/wiki/index.php/Digital_Elevation_Model)
- <http://www.gdcvault.com/play/12777/HALO-WARS-The-Terrain-of>
- <http://davis.wpi.edu/~matt/courses/fractals/brownian.html>
- [http://code.google.com/p/fractalterraingeneration/wiki/Fractional\\_Brownian\\_Motion](http://code.google.com/p/fractalterraingeneration/wiki/Fractional_Brownian_Motion)
- <http://www8.cs.umu.se/kurser/TDBD12/HT01/papers/MusgraveTerrain00.pdf>
- <http://www.lighthouse3d.com/opengl/terrain/index.php3?fault>
- [http://www.decarpentier.nl/downloads/InteractivelySynthesizingAndEditingVirtualOutdoorTerrain\\_report.pdf](http://www.decarpentier.nl/downloads/InteractivelySynthesizingAndEditingVirtualOutdoorTerrain_report.pdf)
- <http://www.gameprogrammer.com/fractal.html#midpoint>
- Musgrave, et al. "The Synthesis and Rendering of Eroded Fractal Terrains", Siggraph 1989
- Ebert, David S., Musgrave, F. Kenton, Peachey, Darwyn, Perlin, Ken and Worley, Steve. Texturing and Modeling: A Procedural Approach, 3rd edition. USA. Morgan Kaufman Publishers, 2003



## Bibliography

- DeLoura, Mark. Game Programming Gems. Charles River Media, 2002.
- Martz, Paul. "Generating Random Fractal Terrain." Game Programmer. Publisher Robert C.



## Questions

