Technical Concepts

Lecture 3

Computational vs Physical Space

- Computational and Physical dimensionality are two distinct properties of a dataset.
- Computational Space relates to the natural (abstract / mathematical) ordering between data points.
- Physical Space is the what we see: x,y,z.
- The relationship is not always 1:1.

Computational Space

- The dimensionality of an array storing the data is its computational dimensionality.
- A single point is 0D
- A line graph's data is computationally 1D.
- A digitized image forms a 2D array of values.
- A cube of data is computationally 3D.
- We are not limited to 3 dimensions…

Physical Space

- Physical Space may be natural or adopted.
- A set of X,Y,Z points are naturally "3-Space"
- An abstract 2D (computational) array of numbers may adopt a
  - 2D physical space "2-Space" as an image,
  - 3D physical space "3-Space" as a surface.

Terminology

- In many situations it is clear from the context whether "nD" refers to computational or physical space.
- If context is not clear computational and physical space may be the same.
- When unclear we will refer to "*n*D" as computational space and "*n*-Space" as physical space.

Dimensionality Examples

- A point in 3-Space is “0D 3-Space”
- A line/set of points in 3-Space is “1D 3-Space”
- A normal screen image is “2D 2-Space”
- An image mapped onto a cube face is “2D 3-Space”
- An MRI/CAT scan slice is “2D 3-Space”

Surfaces

- A surface is typically “2D 3-Space”
- Surfaces are a fundamental form of representing 2D or higher data.
- Many techniques exist to convert “nD” data into surfaces.

  Eg: 3D cube displayed using its 2D bounding surfaces.

- Surfaces may also be generated via equations.

Representing Surfaces

The need to represent surfaces comes from various sources:

- Design of physical objects
- Display mathematical equations
- Model/Digitize existing objects
- Display numerical data

Surface equations may exist for the first two.

This is rarely so in the last two cases.

Mathematical Surfaces

- Numerous equation forms are available for representing surfaces mathematically.
- Surfaces contain sets of curve segments.
- Examples include:
  - Parametric curves/surfaces
  - Quadric surfaces: $f(x,y,z) = 0$
- Benefits
  - Inherently smooth surface.
  - Surface height at any point can be calculated.

Parametric Surfaces

- A Parametric Polynomial Curve defines points on a 3-Space curve by using three functions of a parameter:
- Eg: Cubic polynomial:

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d$$

$$y(t) = a_y t^3 + b_y t^2 + c_y t + d$$

$$z(t) = a_z t^3 + b_z t^2 + c_z t + d$$

Cubic Polynomial Families

- Each curve segment P(t) is constrained by:
  - Endpoint position
  - Endpoint tangent vector
  - Curve continuity.
- Different constraints produce curve families
  - Hermite - 2 endpoints, 2 endpoint tangents
  - Bézier - 2 endpoints, 2 control points
  - Spline - 4 control points.

Approximating Surfaces

- Approximation is usually required at some point in an equation's passage though a computer.
- A displayed equation has to be evaluated at discrete points.
- Choosing a sampling interval at pixel resolution results in smooth surfaces, but takes a long time to calculate.
- A coarser resolution often provides the same information orders of magnitude faster.

Meshes

Mesh Surfaces

- A mesh is used to represent
  - raw numeric data
  - simple surfaces exactly
  - approximated surfaces
- Computer graphics hardware is optimized to draw meshes.
- A typical fast modern graphics computer may be able to sustain 50k-100k shaded triangles/second.

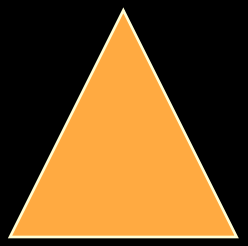
What is a Mesh ?

- A mesh is a spatial representation of data.
- A mesh is a set of vertices and connectivity information (generally based on a particular type of polygon).
- Key points:
  - Surface defined by discrete spatial values.
  - No data between points.

Mesh Approximations

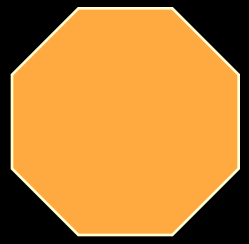
Issues when approximating surfaces:

A circle…how many sides ?

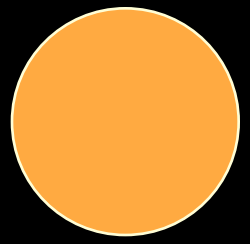


n = 3

n = 4



n = 8



n = ?

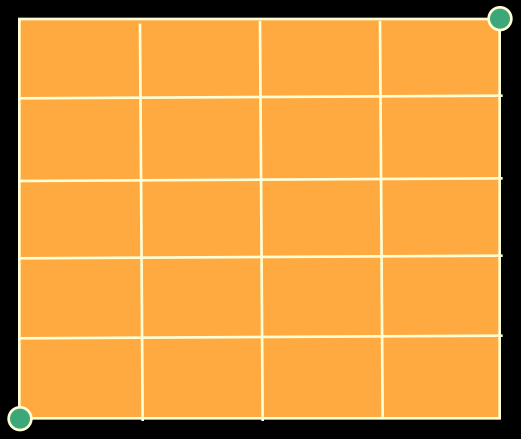
Mesh Types

- Structured
  - Uniform
  - Rectilinear
  - Irregular
  - *Compound*
    - Hierarchical (Quadtree / Octree)
    - Block Structured
- Unstructured

Structured Meshes - Uniform

- Every grid cell is the same size
- Definition:
  - 2 points + dims

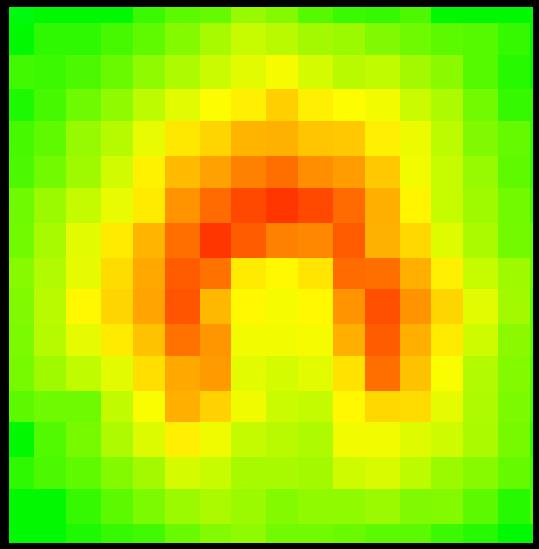
    or
  - 1 point + dims + grid size
- *Implied* Coordinates
- *Implied* Connectivity



2D Uniform Mesh

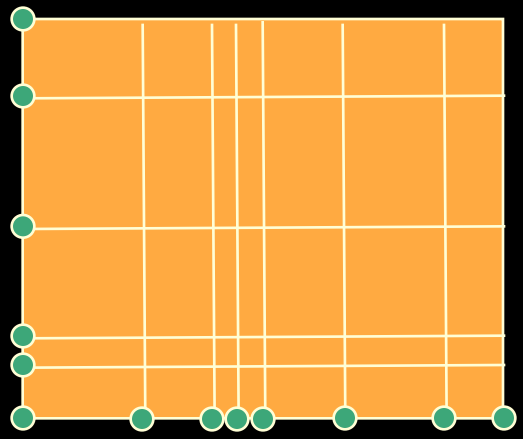
Uniform Mesh - Example

- Data in a uniform mesh
- Cells are clearly visible
- Each cell is the same size
- Surface defined by mesh is coloured by data



Structured Meshes - Rectilinear

- Defined by 1D arrays of axis intersection points
  - f(x), f(y),…
- *Implied* Coordinates
- *Implied* Connectivity

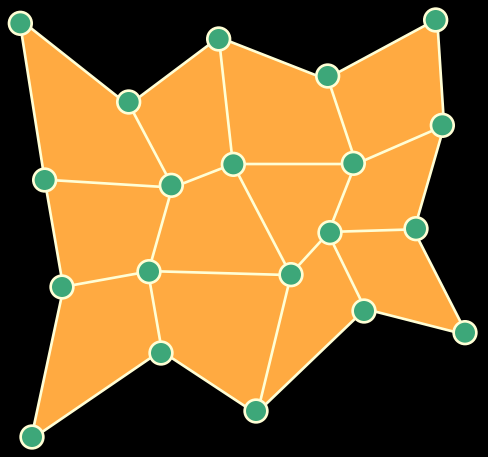


2D Rectilinear Mesh

Structured Meshes - Irregular

- *Explicit*
  Coordinates
  - Every point explicitly defined

    f(x,y)
- *Implied*
  Connectivity

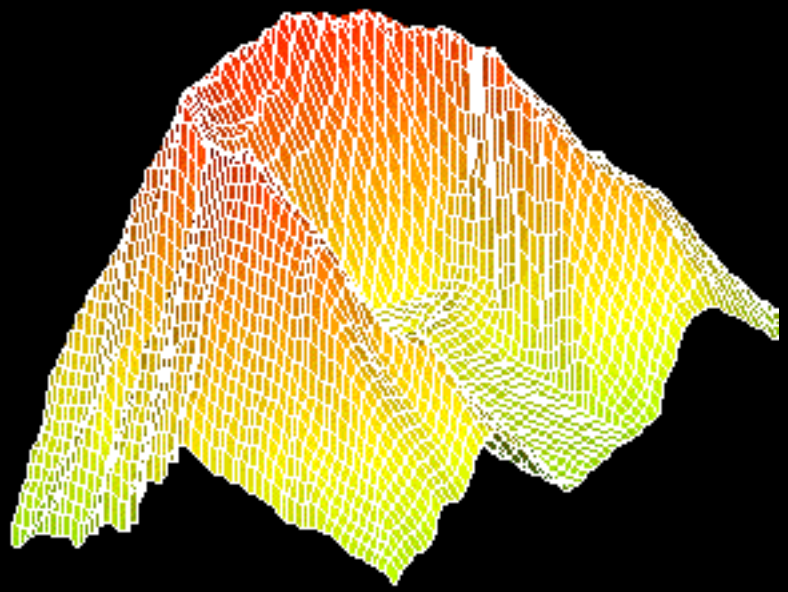


2D Irregular Quad

Structured Mesh Example

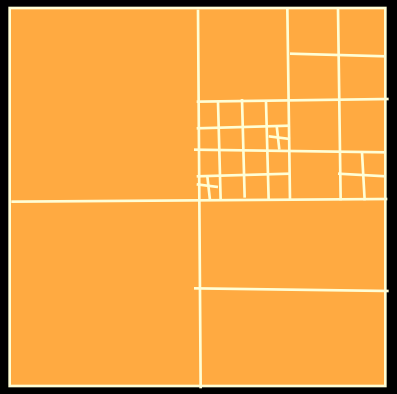
- Land surface

  Digital Terrain Model (DTM)
- Quad Mesh
- Mesh lines are shown.



Structured Meshes - Hierarchical

- Adaptive Resolution
- 2D - Quadtree
- 3D - Octree

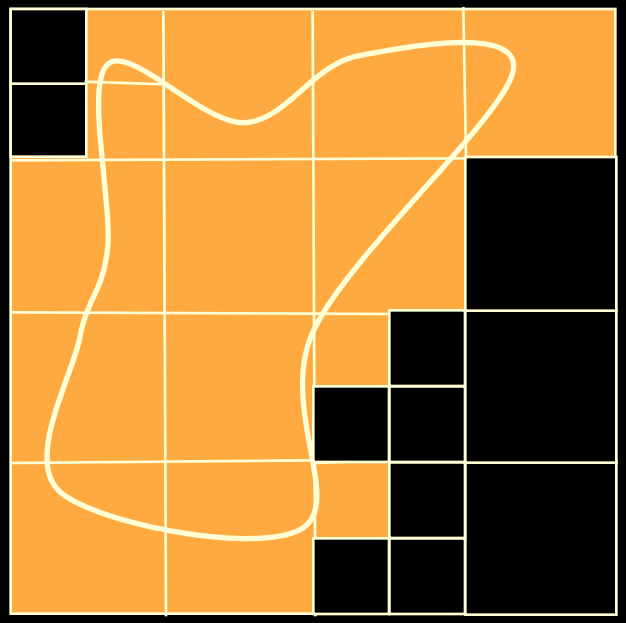


2D Binary Quadtree Mesh

Quadtree Example

Partially complete quadtree

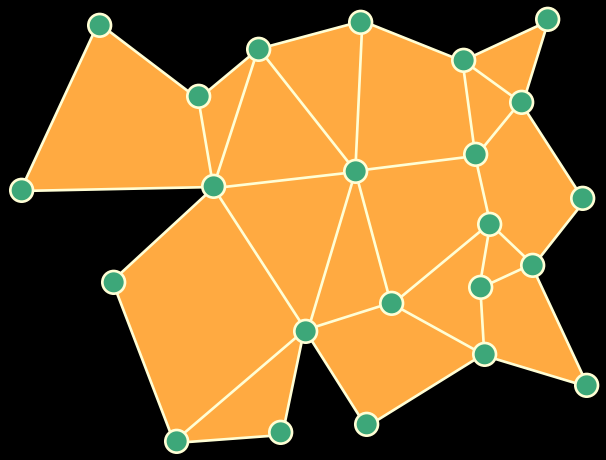
Quadrants are recursively sub-divided until homogeneity or depth limit is reached.



Unstructured Meshes

- *Explicit*
  Coordinates
  - Every point explicitly defined

    f(x,y)
- *Explicit*
  Connectivity



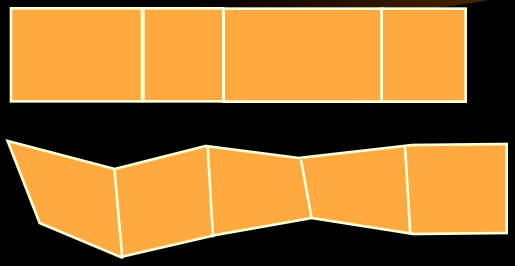
Unstructured 2D Mesh

Computer Graphics Meshes

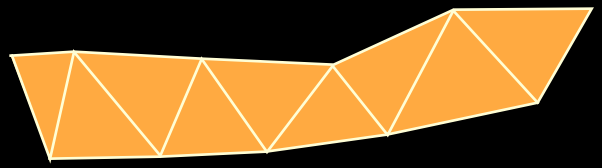
- All the mesh types are resolved into simpler meshes for computer graphics rendering.
- Computer graphics libraries implement various **structured** mesh types for **efficient** surface representation.
  - Points
  - Lines, Triangles, Quadrilaterals (individual)
  - PolyLines, Tri-Strips, Quad-Strips
  - Triangle Fans

Computer Graphics Meshes…

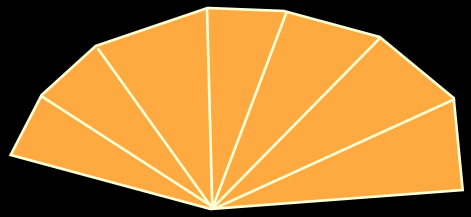
- Quad Strips
  - Planar vertices!



- Tri-Strips



- Triangle Fans



Why the concentration on strips

- Surely it is harder to get data intro strips ?
- Yes it is, but look at the downside to not doing this:
- Compare data size / efficiency of an arbitrary triangular mesh surface with that of a similar tri-strip.

Creating Meshes

So, now you're convinced meshes are on the whole good…how do you create one?

From Scattered Data to a Mesh

Two common methods

- Delauney Triangulation
- Regridding

Delauney Triangulation

Turn points directly into a mesh

- Pros
  - same data points
  - efficient surface
- Cons
  - long thin triangles
  - lack of consistency between similar data sets
  - problem of concave data sets
  - slow

Regridding

Forcing data onto a grid

Functions calculate value at new grid node

- Pros
  - more algorithms work with gridded data
  - relatively fast
- Cons
  - lose original data points
  - quality depends on grid size

Regridding Issues

- Problems with varying density data sets
  - Bad choice of grid size or “difficult” datasets with widely varying point spacing cause problems.
- Concave and hole problem.
- NULL data problem
  - Eg: no coordinates within search radius
- Storage efficiency of sparse arrays
  - may have 1000’s grid points with 0 data points

General Mesh Issues

- Surface sections at different resolutions.

  Eg: low resolution earth plus high resolution Australia.

  – Problems:
    - interference between meshes without cutout
    - cutout edge stitching

General Mesh Issues

- **Gridded** Mesh Closure
  - (Not a problem with arbitrary meshes)
  - Seam lines
  - Repeat vertex coordinates or share ?
  - Sphere - unique pole points or 1 point ?
  - Affects texture mapping and shading.

Beyond 2D

- The above meshes are computationally 2D (3-Space). What about 3D 3-Space ?
- There are 3D parallels to the 2D meshes:
  - hexahedron
  - triangle prism
  - tetrahedron (triangular base),
  - pyramid (rectangular base)

3D Computational Meshes

- These meshes are useful in CFD and FEA for example, but computer graphics sub-systems cannot display them directly.
- Instead resort to external faces, exploded views and other techniques for display.

End

Lecture 3