PhyreEngine™ Terrain

Jason G Doig
SCEE R&D

Terrain

* Very common requirement
* Current solutions not so great
* There are better ways...

Agenda

* Choosing the technology
* Designing the art pipeline
* Implementation and performance

Goals

* Represent a variety of terrain types
  * Vertical surfaces and overhangs
  * Caves and tunnels
    * Surfaces with genus > 0

More Goals

* Scalable - terrains are big
  * Needs to be streamable
  * Need view dependent LOD
* Unique geometry and texture
* Must be easy to edit

Geometry

* Heightfields
  * Not good for complex topology, even with displacements
* Meshes
  * Too much data, unwieldy at a large scale

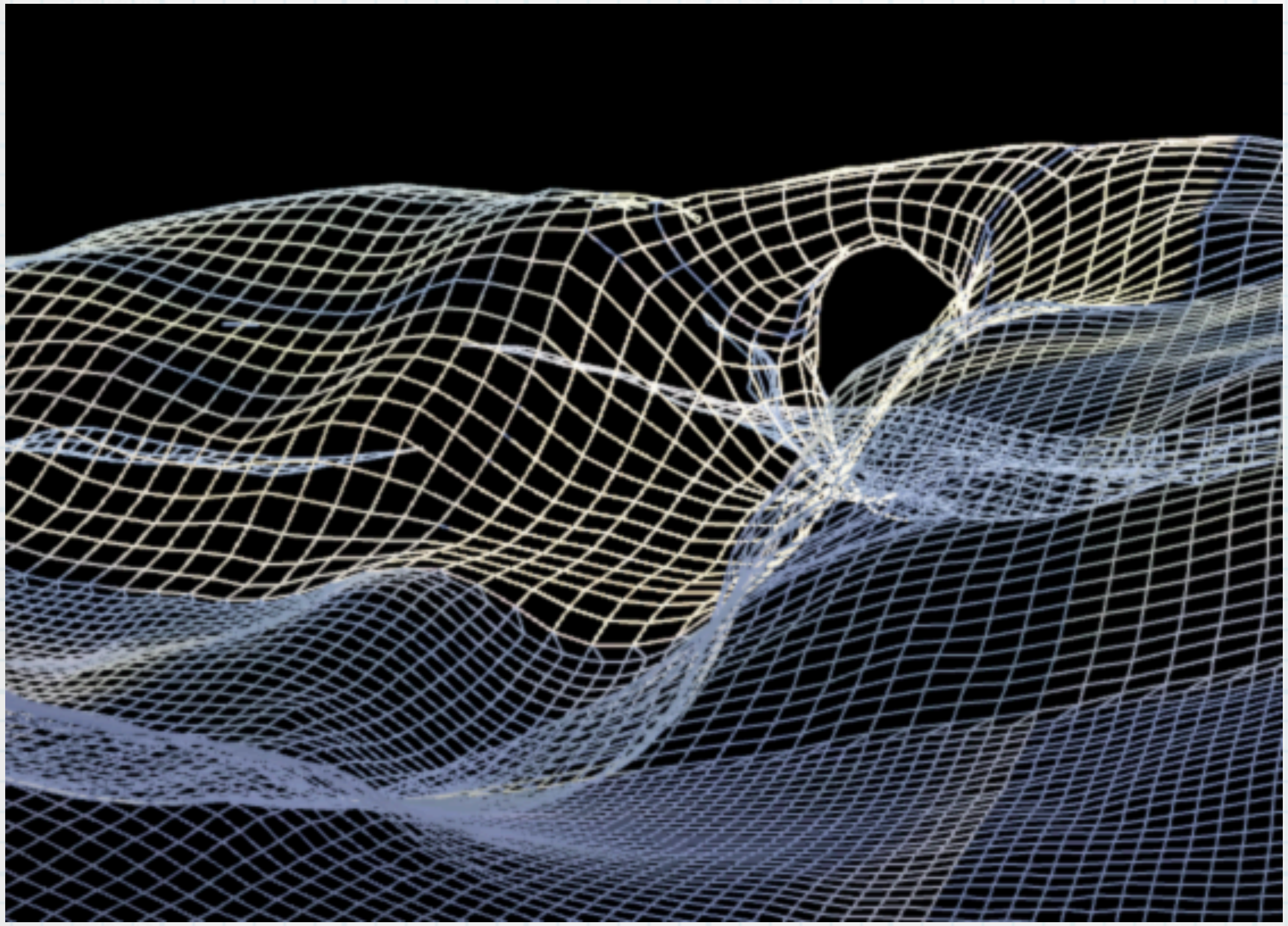
High order surfaces

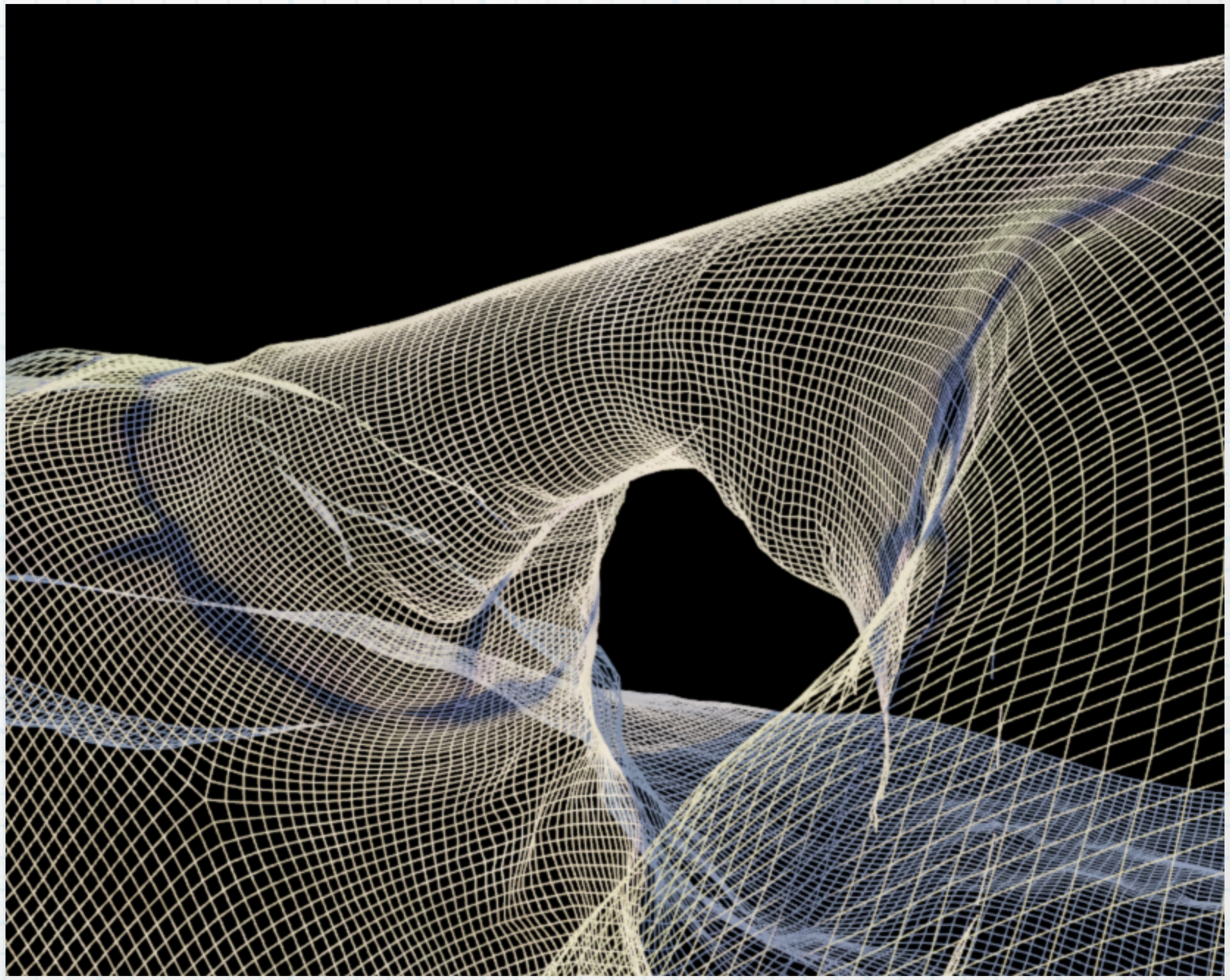
* Smooth surface is unsuitable
  * Terrain is not often smooth
  * Probably need a displacement
* However LOD is good - subdivision
* Still need good authoring tools

Subdivision Surfaces

* Mesh based authoring
* No restrictions for topology
* Requires displacements for detail
* LOD fairly easy
  * Especially if we subdivide the terrain

Example





Terrain Patches

* Split terrain into patches
* Each patch store:
  * Low res mesh - keep in memory
    * Full topology, UV coords, etc.
  * High res positions only - stream
    * No topology or UVs

Authoring

* Build a terrain out of low-res meshes
* Subdivide into a smooth hi-res mesh
* Sculpt the hi-res detail
  * No topology changes to hi-res!
* Apply textures - including normal map

Unanswered Questions

* How to put meshes together
* Need to handle subdivision across meshes
* How flexible will mesh layout be?
* Runtime implementation
* Texturing and sculpting

Mesh Layout

* On a grid!
  * Terrain tends to be 2D on large scale
  * Simplifies LOD greatly
* Downsides
  * May restrict alignment of features

Level Editor

* Custom tool for laying down patches
* Handle palette of low-res shapes
* Place low-res mesh instances into level
* Create smooth subdivisions
* Export/import sections for sculpting and texturing

Sculpting

* Many packages can sculpt polygons
  * We just need to import/export
* Want to ignore mesh boundaries
  * Sculpt continuously across edge
  * No geometric restrictions
* Allow arbitrary selection of polys

Texturing

* Want unique texturing
* Need to stream
  * Could use one giant texture
    * i.e. virtual-texture techniques
  * Discrete meshes on a grid allows a simpler solution - texture per mesh.

Editing texture

* Need to edit discrete textures seamlessly - treat as continuous
* Need to paint multiple channels
  * colour, bump, material parameters...
* Tools exist, but didn't fit our needs

Terrain Paint

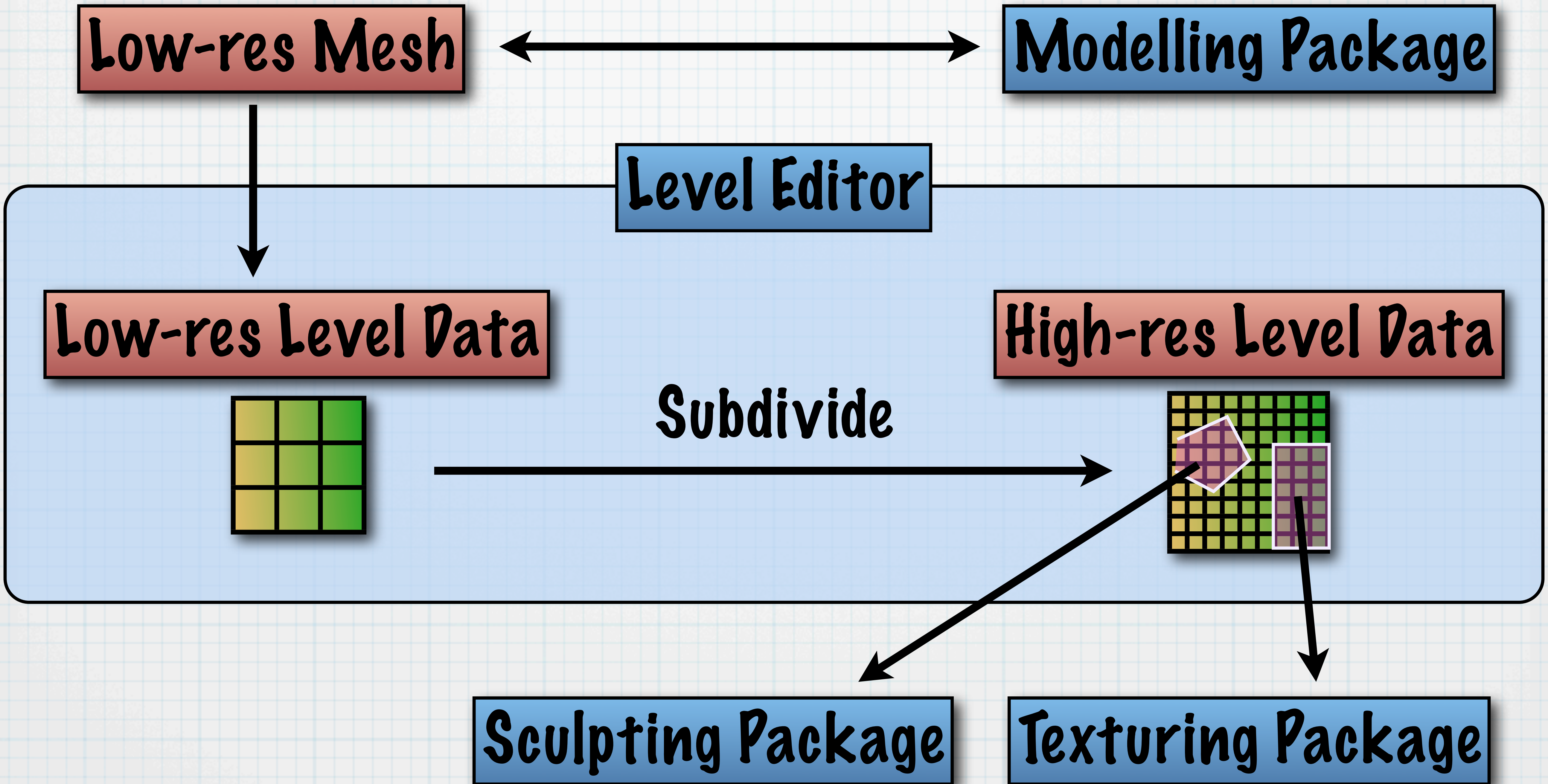
* Allows painting onto a 3D mesh
* Allows painting across multiple meshes
* Allows painting of multiple channels
* Directly shows displacement maps

Terrain Paint Demo

Level Editor

* Exports whole meshes for texture editing
    * Doesn't make sense to only partially edit a texture

Diagram of workflow



Texture Compression

* Lots of texture data
    * 72 bits per texel
    * 9MB per patch.
* DXT gives a fixed reduction
    * But isn't always appropriate
* Flexible compression would be good

Compression Schemes

* DXT is not ideal
  * But good for runtime!
* Tricky to further compress DXT data
* Good ratios require lossy compression

Compression

* Compress single channels
  * Optional colour space conversation
  * Can downsample
* Compress in blocks
  * Exploit small-scale structure
  * Good for re-compressing to DXT

Block Compression

* DCT based (like JPEG)
* Transform block into cosine terms
* Quantize terms
  * This is the lossy step!
  * Adjustable for different channels / ratios

Stream Compression

* Order block into 1D stream
  * Zig-zag
* Run-length encode any zeros
  * Quantization should discard a lot
  * Terminate if no non-zero values
* Standard data compression on result

Runtime

* Decode blocks of three channels
* Upsample pixels and CSC if necessary
* Optionally re-encode as DXT

DXT Compression

* Find a palette for each block
  * Not completely trivial
    * Some entries are interpolated
    * Choice of interpolation
* Match texels to palette entries
  * Simple distance - no dithering...

DXT Implementation

* Attempt 1 - port of Squish
  * Lots of loops and conditionals
  * Didn't work well on SPU
* Attempt 2 - naive min/max algorithm
  * Looked very poor

DXT Implementation

* Attempt 3 - principal axis
  * More or less what Squish does
  * Fairly maths heavy
* Optimisations
  * AOS -> SOA
  * Move from float to integer (uchar16)

Performance

* Naive min/max
  * 120 cycles/block (7.5 cycles/pixel)
* Full float implementation
  * 530 cycles/block (33 cycles/pixel)
* Optimised principal axis code
  * 325 cycles/block (20 cycles/pixel)

Compression Ratio

* Currently we're aiming at 10:1
* Slightly better than DXT for colour
  * But better quality if we don't re-DXT
* Much better than uncompressed
  * Good for normals and material info

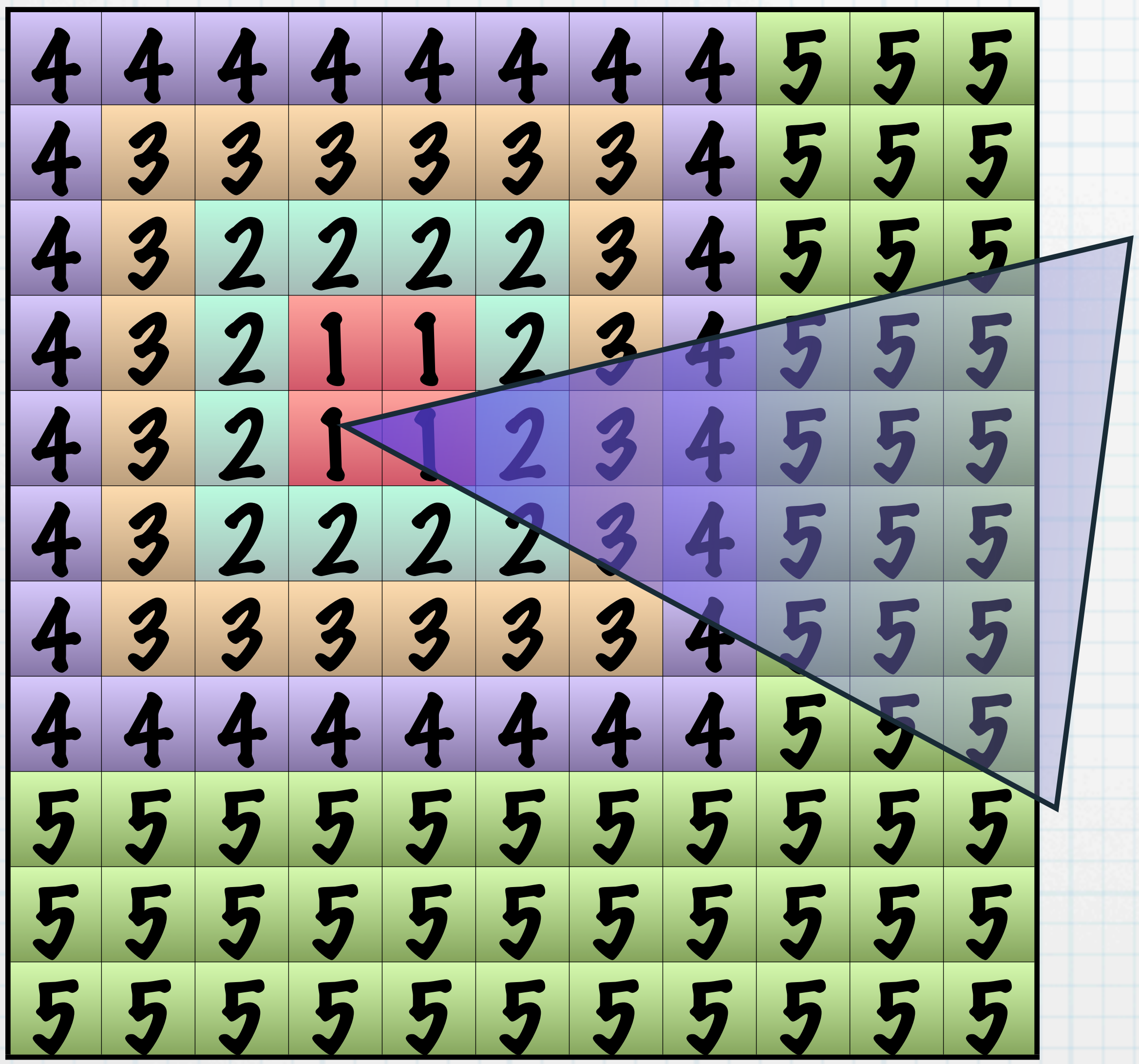
Runtime

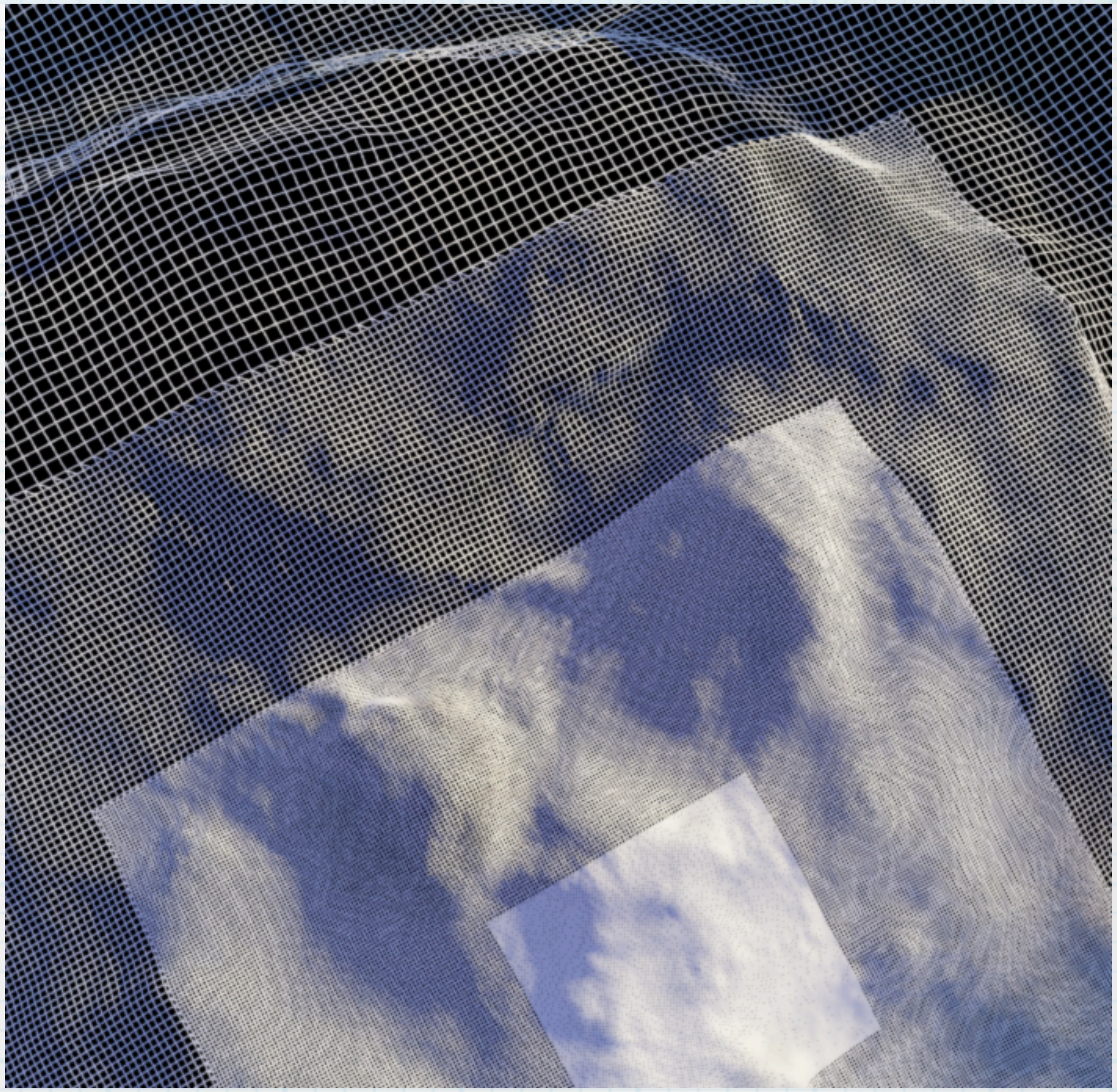
* Low-res meshes always loaded
  * No current need for reduced LOD
* Hi-res content streamed
  * LOD chosen and streamed per-block
  * Geometry can morph between levels

LOD levels

* LOD level calculated as Chebyshev distance to camera
* Highest level in nearest 4 blocks
* Falls off to lowest level
* No more than 1 level difference to adjacent blocks
  * Required for stitching and morphing

LOD Diagram


4 4 4 4 4 4 4 4 5 5 5
4 3 3 3 3 3 3 4 5 5 5
4 3 2 2 2 2 3 4 5 5 5
4 3 2 1 1 2 3 4 5 5 5
4 3 2 1 1 2 3 4 5 5 5
4 3 2 2 2 2 3 4 5 5 5
4 3 3 3 3 3 3 4 5 5 5
4 4 4 4 4 4 4 4 5 5 5
5 5 5 5 5 5 5 5 5 5 5
5 5 5 5 5 5 5 5 5 5 5
5 5 5 5 5 5 5 5 5 5 5



Geometry Processing

* Limited memory
* Fast CPU
* Subdivided dynamically every frame
* No geometry caching
* Continuous LOD (geo-morphing)

Stitching and Morphing

* Patches can morph between levels
* Adjacent patches can be 1 base-level different
* Edges clamped to LOD of adjacent patch
* Clamping done as a fix-up pass after subdivision

Geometry Processing

* Geometry processed in parallel
* Terrain patch geometry split into smaller chunks
  * Need to fit into a buffer on SPU
* Custom routines for each LOD
  * Faster than a generic routine

Basic Performance

* General Scene complexity
  * Usually around 20 visible patches
  * Generates about 70,000 triangles
  * More if re-rendering for shadows
* Terrain tessellation is 50% of 1 SPU
* Decompression scalable on remainder

Geometry Performance

* Single SPU
  * Lowest LOD around 11 M verts/s
  * Highest LOD around 35 M verts/s
* Note - these are *after* culling
  * Vertices generated actually higher

Streaming

* Typically needs 1.2MB/s
  * Peaks at 4MB/s in short bursts
  * Quite comfortable for HDD
* Texture Decompression Speed
  * 16 - 70 million texels/s non-DXT
  * 35 - 40 million texels/s DXT

More Performance

* Scales linearly with more cores
  * Until we run of out of bandwidth
  * Or the GPU bottlenecks
  * Typically these don't happen for us

Summary

* Displaced subdivision surfaces are cool
  * Flexible geometry
  * Good LOD
* Consider alternative compression
  * Re-compress to runtime format
* Performs well on modern platforms

Questions?