Four Million Acres, Seriously

GPU-based Procedural Terrains in Serious Sam 4: Planet Badass

Alen Ladavac, CTO, Croteam

@AlenL *alen.ladavac@croteam.com*

GAME DEVELOPERS CONFERENCE
MARCH 18–22, 2019 | #GDC19

Motivations

- Serious Sam
  - “Bigger is better”
  - Unused space instead of invisible barriers
  - “Huge open levels” (in fact ~5 km so far)
- Real world vistas ~50-100 km distances
- Small scale detail (1 mm textures)
- “Scanned” vs generated
  - Is this in conflict?
- “Photorealistic” look
  - Photo-like environments contrast crazy enemy designs.

Carcassonne, France to Pyrenees (~80 km)

The Deceptive Scale

Hirschegg, Austria

Saalfelden, also Austria… just bigger

Goals

- 128×128 kim terrain
  - “Background” is not a special case.
- Texture detail: 1024 tex/m
  - Pixel size: 1 mim
  - About optimal for floors on 4k resolutions in first person
- Elevation detail: 64 vtx/m
  - Triangle size: ~1.5 cm
  - Actually model cobblestones/pavements

Disclaimer

- The following images and videos are all from the game, but...
- Material is not representative of the final game.
  - The game is still in development.
  - Many content elements are still WIP or placeholders.
  - Some features (usually far-distance colormap) are disabled in some screenshots to better show other features.
  - Bird's-eye views are used here to visualize some concepts - they may show artefacts.
    - (Actual game only takes place at ground level)

Video #1

How Much Data is That Again?

- Pre-made data is too big.
  - Elevation for 128×128 kim with 64 vtx/m => 8M x 8M vertices
    - That is 64 billions vertices.
    - About 100 TB data for elevation alone!
  - At 1 m per sample => still ~16-32 GB...
  - ...and still needs textures/materials info.
- Procedural generation on the fly
  - Slow
  - Looks artificial
  - ...but only if you generate everything!
- So...

Solution:

Hybrid Procedural Generation

+

Multiresolution Editing

Large-Scale Features

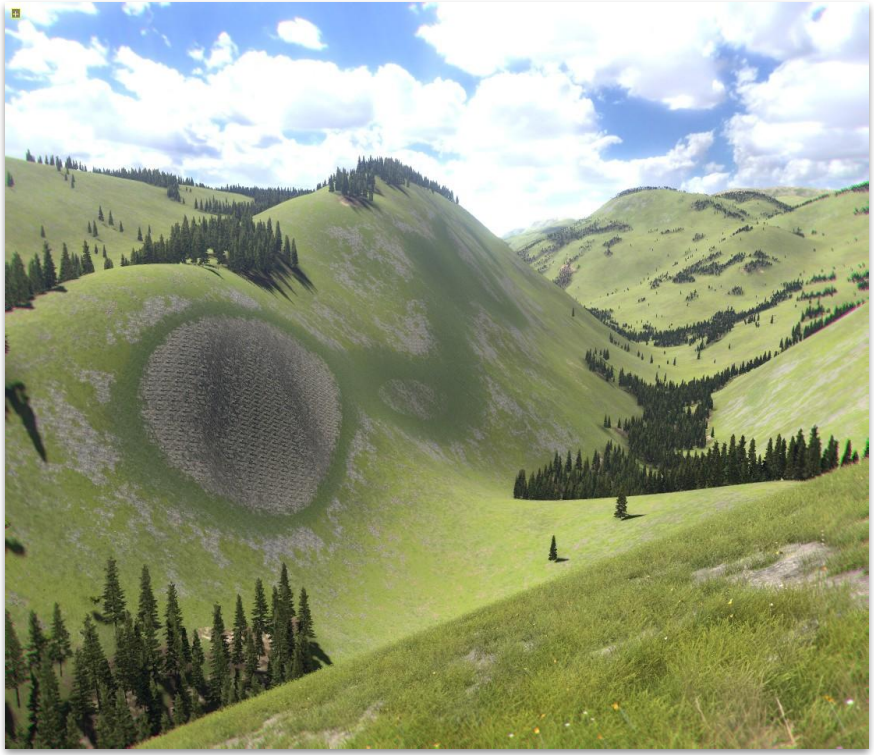
- Pre-created data
  - Roughest: 32 per kim (every 32 m)
    - => 4k×4k for whole terrain
  - Elevation
  - “Far texture” albedo
  - Vegetation density
- Total data for a 128×128 kim terrain:
  - ~128-200 MB
  - (On the order of size of lightmaps for our older levels)
  - We don’t even stream it (for now).

Fine Detail Features

- Photo-scanned ground textures:
  - Full material data:
    - Albedo
    - Normal
    - Gloss
    - Height
      - Mixed with elevation to generate actual geometry!
  - 1-4 m per texture
- What to do between 32 m and 1 m?

Mid-Level Terrain Features

- Elevation:
  - Cubic spline



Mid-Level Terrain Features

- Elevation:
  - Cubic spline+
  - Multi-band fractal noise
    - More noise at higher elevations
    - More if >"angle of repose" for soft ground

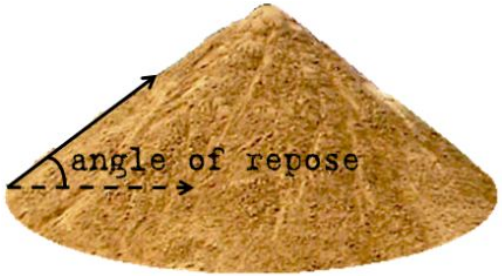


image: Wikipedia

Mid-Level Terrain Features

- Elevation:
  - Cubic spline+
  - Multi-band fractal noise
    - More noise at higher elevations
    - More if >"angle of repose" for soft ground

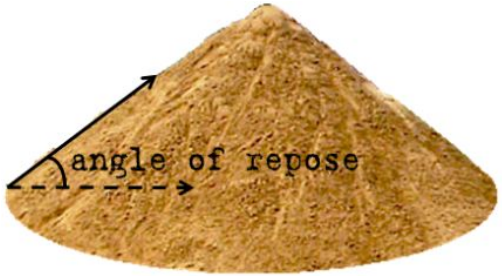
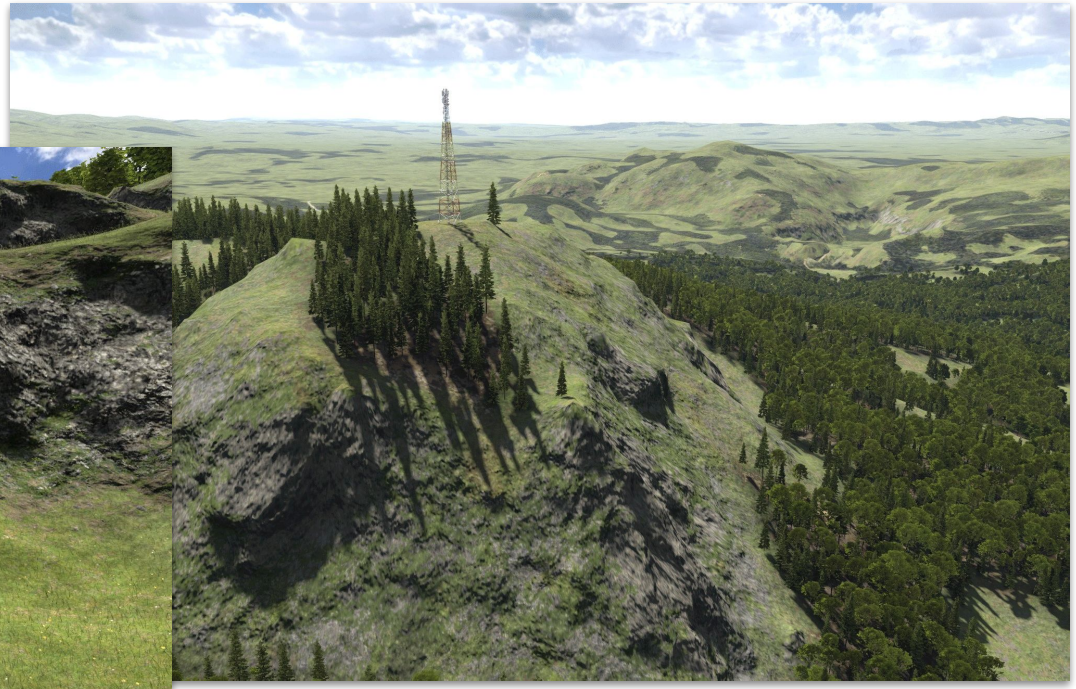


image: Wikipedia

  - Also - horizontal displacement...

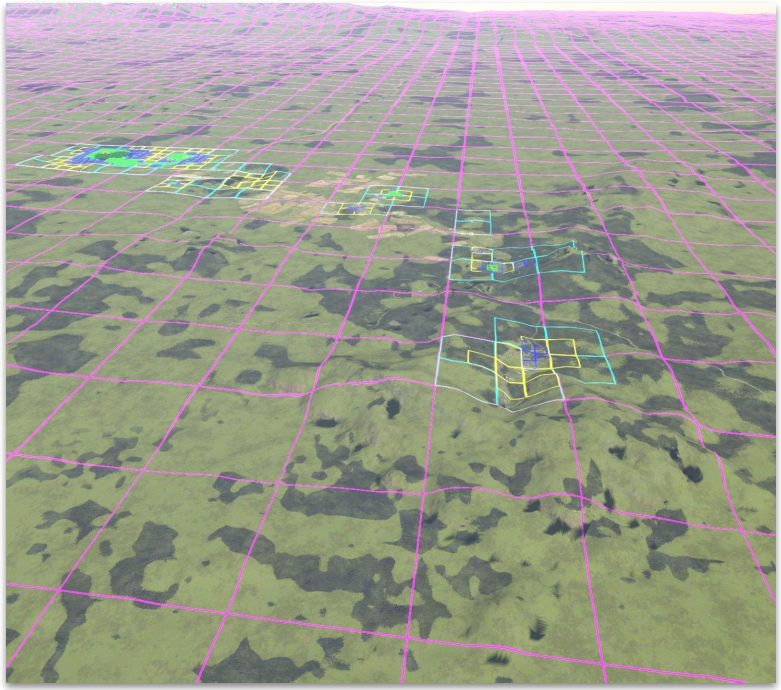
Horizontal Displacement

- Add x-z offset noise when caching elevation
- More natural look of steeper slopes
- Ref: *Brano Kemen, Outerra blog, 2009*



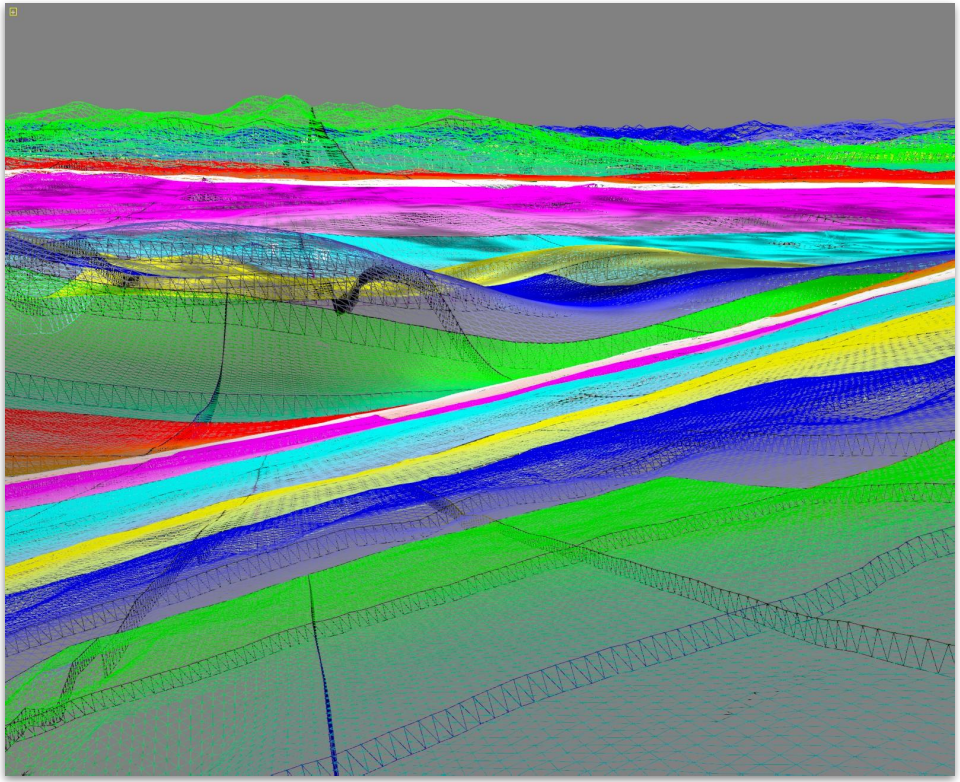
Multi-Resolution Editing

- All terrain data is stored in quad-trees
  - Elevation, vegetation, custom materials, clip masks
  - 32 m is roughest resolution always available
    - (1km nodes - purple)
  - Edit to finer precision in areas of interest
  - “Freeze” fractal data on edit
- High-precision limit: 25 cm
  - Just to prevent “multi-gigabyte-level” accidents
- Some data has defaults
  - Clip mask => visible
  - Vegetation => defaults to e.g. 0.28 (😊!?)



Terrain Mesh Size

- Tile:
  - 33×33 grid of vertices
  - 512×512 texture
  - Min size: 0.5 m
- 2x bigger terrain -> add 9 new tiles
- 1 level ≅ 9 tiles -> 0.5m
- 8 levels ≅ 70 tiles -> 128 m
- 18 levels ≅ 160 tiles -> 128 kim
- ~350k tris total for a typical scene

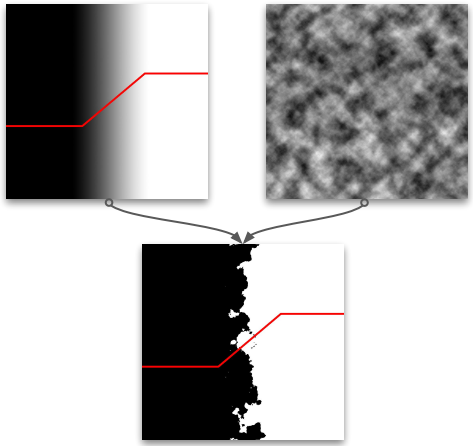


Per-Pixel Elevation

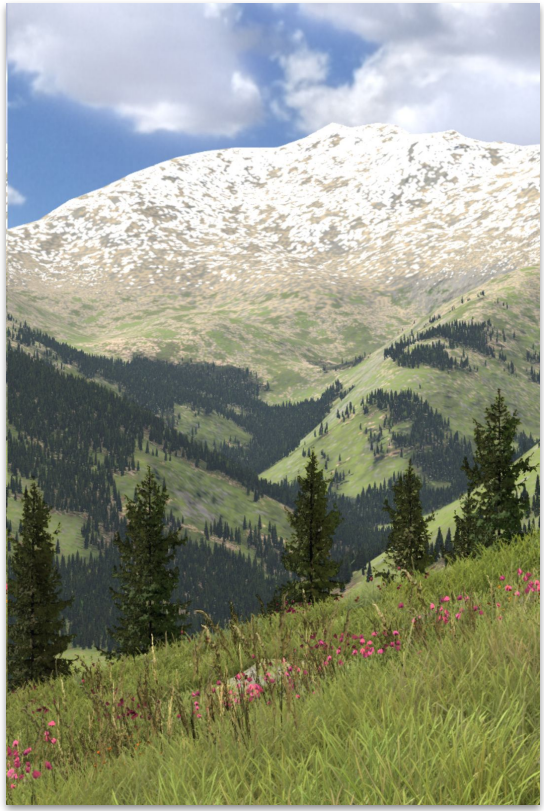
- Adds detail in the distance
  - Normal maps, materials and other details are based on elevation!
- Generation
  - Temporarily generate elevation to per-texel level
  - Generate per-pixel normals, materials, etc.
  - Throw the fine elevation away

Material Rules - Elevation Regions

- Regions (usually):
  - Seaside/sandy (optional)
  - Grasslands
  - Grass & ground
  - Bare rocks
  - Snow
- Uses “simple noisy ramp”
  - Prevents sharp cuts



```
if elevation < lerp(transition_start, transition_end, noise)
  region = lower
else
  region = upper
```

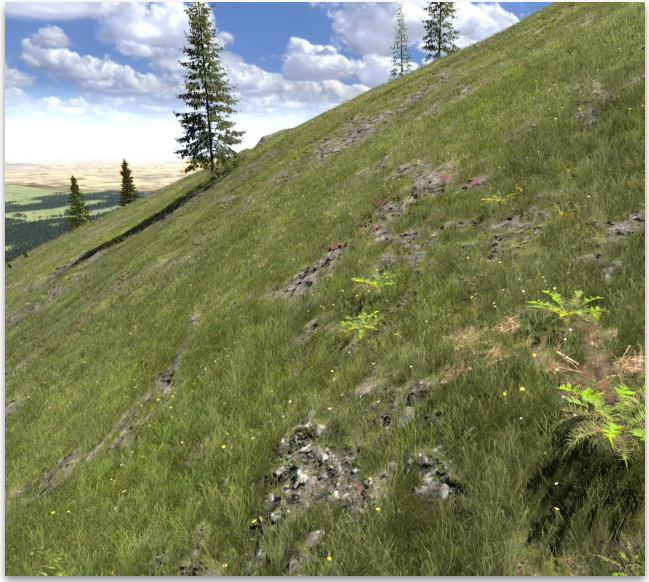


Material Rules - Inside One Region

- Multiple materials per region (2-8 usually)
  - Random pick based on:
    - Slope ramp
    - Vegetation ramp (arid, grass, forest)

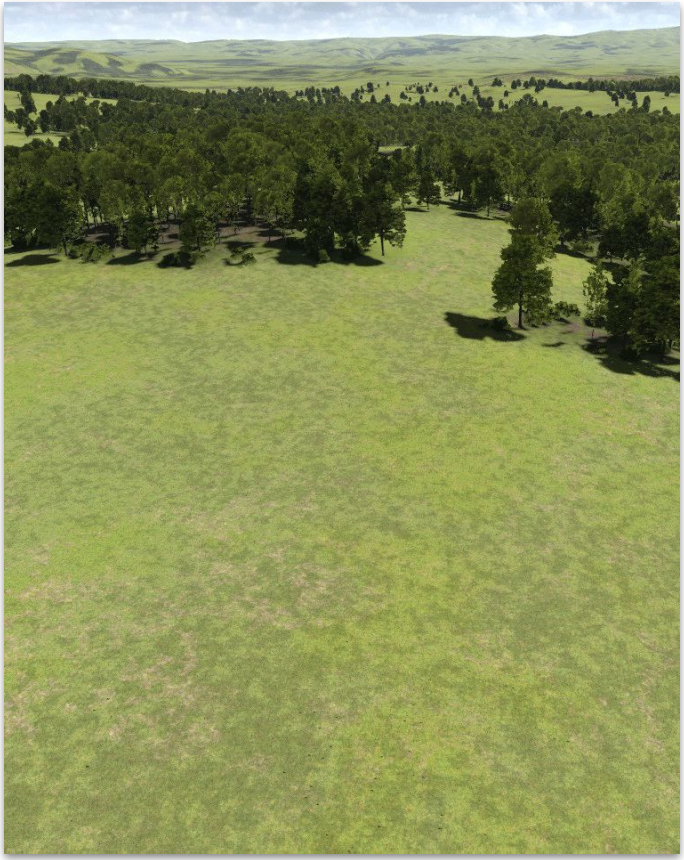
```
chance[i] =
    noise(material[i].noise_params)*
    ramp(slope, material[i].slope_params)*
    ramp(vegetation_density, material[i].vegetation_params)
// ...pick material with highest chance in each pixel
```

  - Roughly match elevation angle of repose to rocky material slope rule
  - Also generates, grass, flowers...



Texture Cascades

- Near textures
  - Per material: 1-4 meters tiles
- Mid textures
  - Per material: 100+ meters tile
- Far texture
  - One for entire terrain



Forest Materials

- Special cases:
  - Forest floor (red circle):
    - Each region has separate material sets for inside the forest.
  - Far forest (blue circle):
    - Splatted into terrain at distance beyond forest imposters.

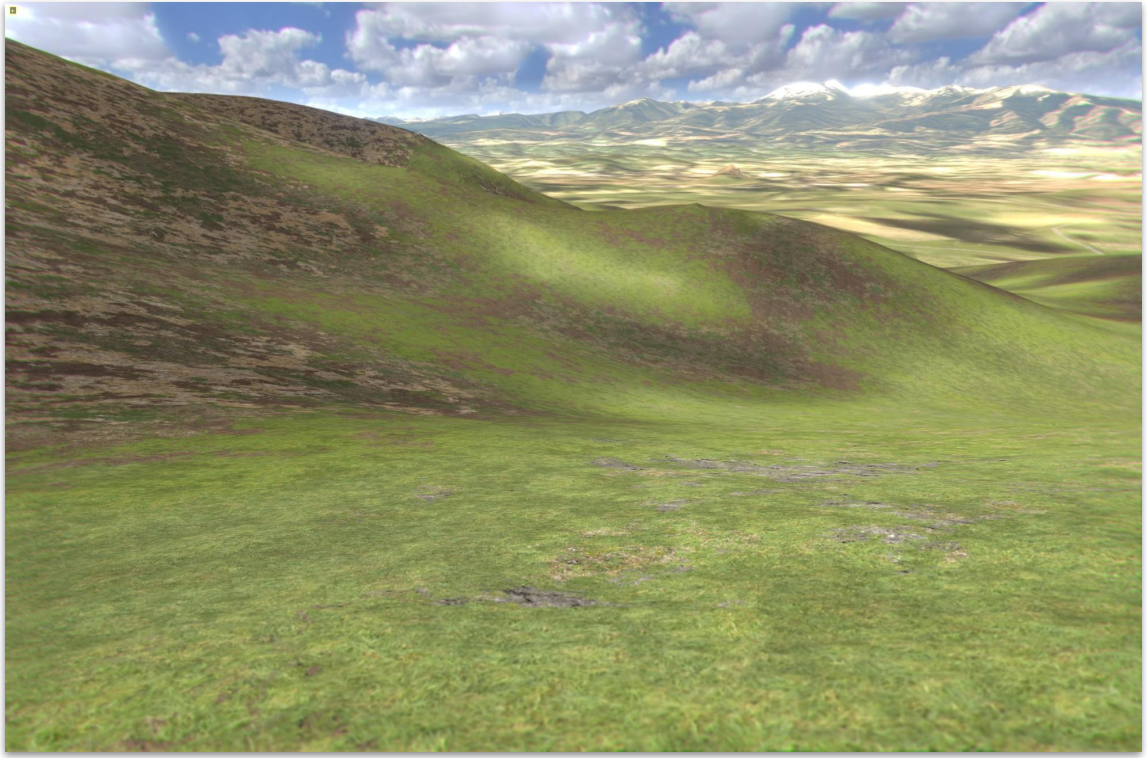
Vegetation Density and Materials

- Density, with elevation/slope/convexity*, controls:
  - Materials (and indirectly grass/flower placement)
  - Distributions for misc plants
  - Distributions for forest (tree probability)
    - E.g trees not spawned on steep slopes, or in drylands.
- Workflow accelerator:
  - Define huge swaths of diverse forests by "spray-painting" vegetation density very roughly.

*Convexity - second derivative of elevation
- i.e. whether to spawn on hilltops (>0) and/or in valleys (<0).

Vegetation Rendering Subsystems

- Bare terrain

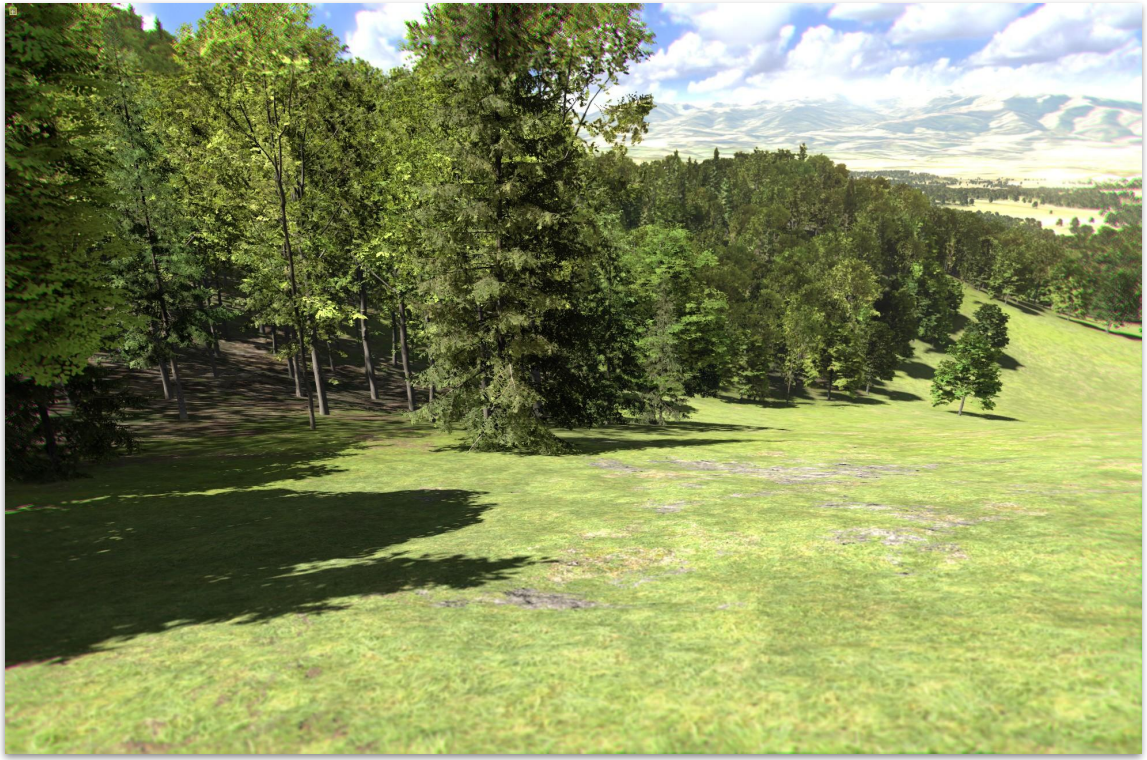


Vegetation Rendering Subsystems

- Bare terrain
- Forest imposters

Vegetation Rendering Subsystems

- Bare terrain
- Forest imposters
- Near trees

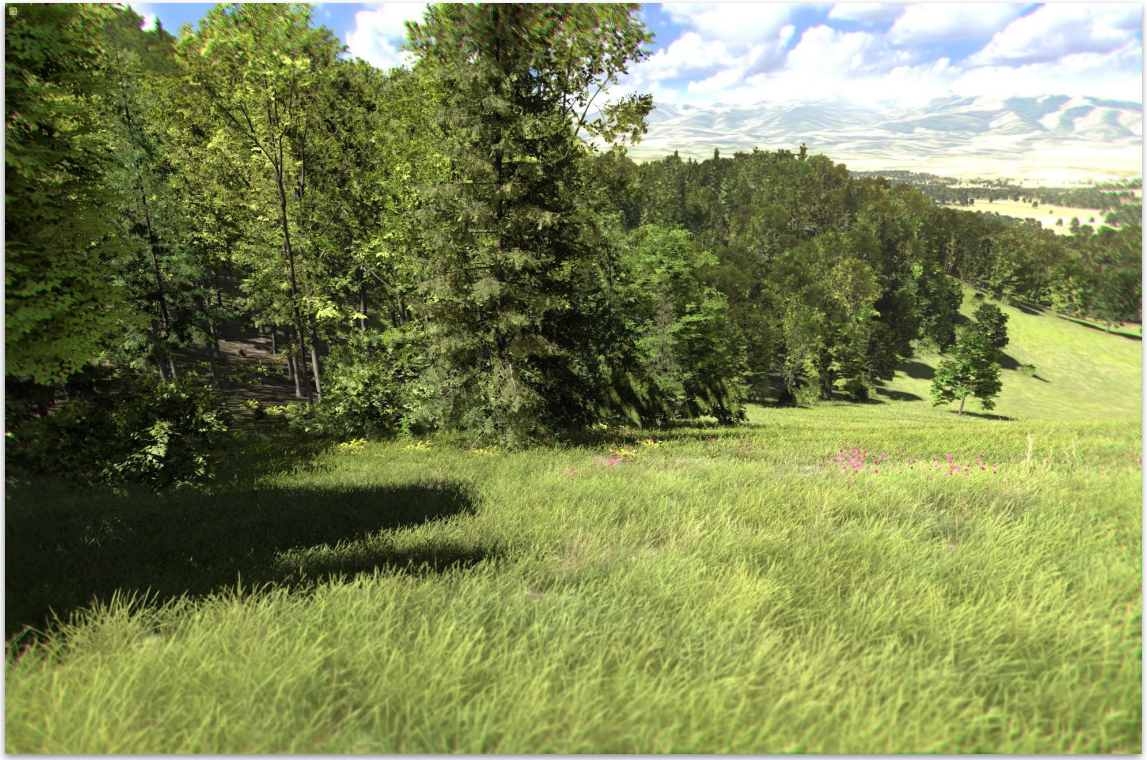


Vegetation Rendering Subsystems

- Bare terrain
- Forest imposters
- Near trees
- Misc plants and crops

Vegetation Rendering Subsystems

- Bare terrain
- Forest imposters
- Near trees
- Misc plants and crops
- Grass blades (geometry!)

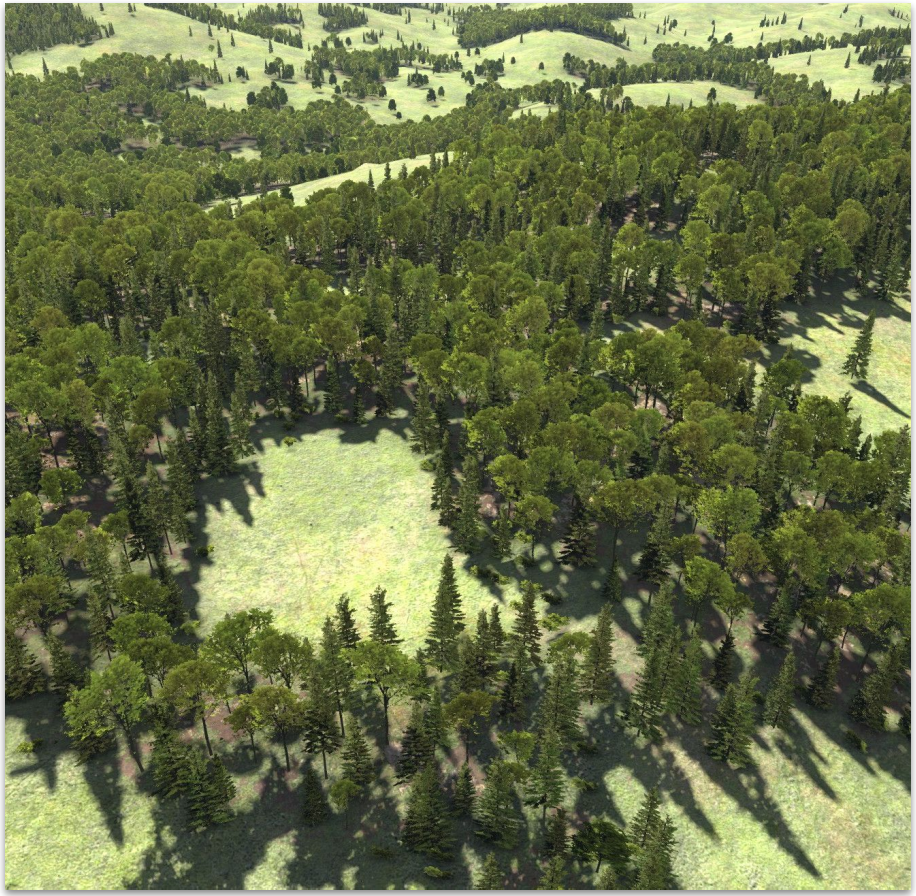


Vegetation Rendering Subsystems

- Bare terrain
- Forest imposters
- Near trees
- Misc plants and crops
- Grass blades (geometry!)
- Flowers (alpha-keyed)

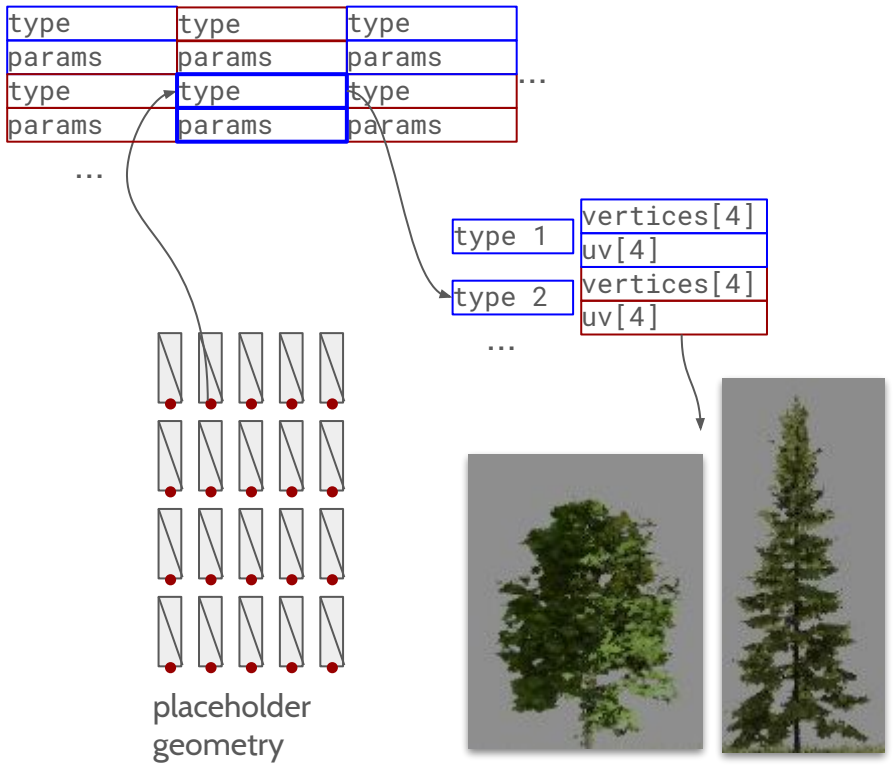
Forest

- Jittered grid: ~8 m between trees
- 1 kim blocks - ~ 16 000 trees per block
  - One tile is ~128×128 tex
  - Each sample stores one tree
    - Offset, rotation, tree type, hue



Forest Imposters Blocks

- Pregenerated geometry
  - Grid of 128×128 standalone quads
  - Each tree = 2 tris

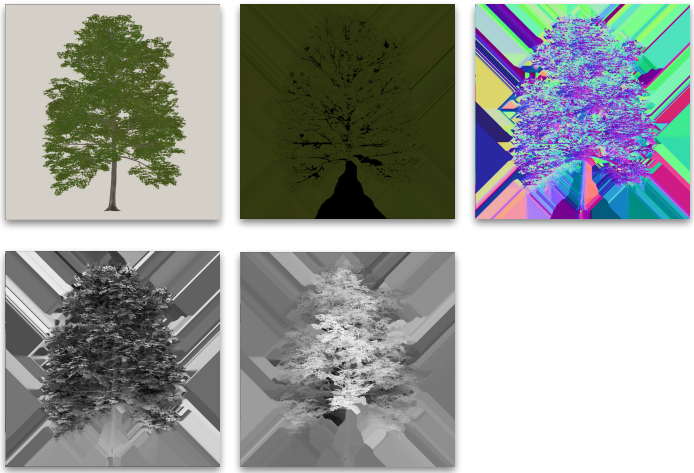


Forest Imposters

- 8-direction crossed geometry or viewer facing?
  - Crossed geometry is slower.
  - Viewer facing lacks parallax.
- So...
  - Use 8-direction textures.
  - Face quad to viewer.
  - Compensate for perspective in pixel shader.
    - See: *"Interior Mapping" by Joost van Dongen*
  - Blend 2 nearest directions.

Imposter Textures

- Store in texture array for different trees
- Albedo, subsurface, normal, AO,depth



Depth texture for proper shadows:

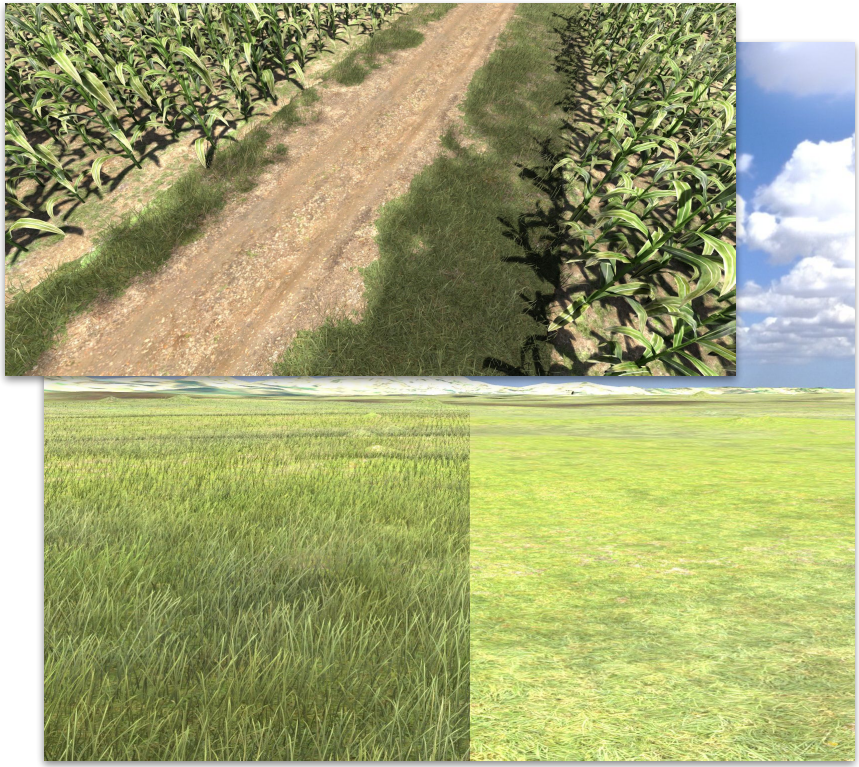
without

with

Video #2

Grass Blades

- True geometry (not alpha-keyed), ~7 tris per blade
- Pre-batched geometry in grids
  - Different density grids overlapped
- Sample from a splatted mask/color texture in vtx shader
- Perfectly match grass material position on the floor
- Albedo colored from the floor
- Per-blade ambient shadow on the floor



Flowers

- Similar to grass, much lower density, alpha-keyed
- Pre-batched placeholder geometry
  - E.g. 32 tris/model max (tri soup for now)
  - Real geometry stored in separate buffers
- Flower distribution from materials
  - Splatted to a cached texture
- Vertex shader reshapes placeholder geometry into a real flower
  - Flower index from splatted distribution
  - Read geometry data for that index

Misc Plants

- Procedurally instanced meshes
  - Bigger flowers
  - Bushes in the plains and forests
  - Forest overgrowth (and logs, roots, ...)
  - Crops
  - Etc.

Roads...

- Splines (roads, rivers, trails, streets...)
- Transition borders - fade-out vegetation map
- Force own material in those parts
  - Gravel/dirt/asphalt
  - Add lines, skid marks, etc. - custom decals
- Level out elevation, with transition
  - Add custom height texture on decals (wheel ruts...)
- "Doodads" - plants and rocks by riverside..., roadside bollards, lamp posts...

... and Patches

- Fields, lakes, yards...
- Transition borders - fade-out vegetation map
- Elevation
  - Unlike roads, smooth it out
  - Add custom height texture for crop rows
- “Doodads” - plants and trees at the edges of crop fields

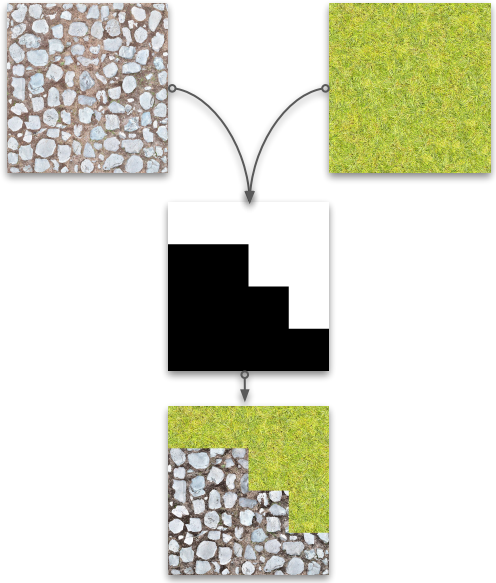
Retaining Walls

- Once you have horizontal displacement...
- ... you can handle retaining walls.
- Custom lines that force vertical cuts
- Move neighbor vertices to be aligned vertically
- Keeps good shape at lower LODs
- Memory-efficient (Doesn't need high-resolution elevation.)

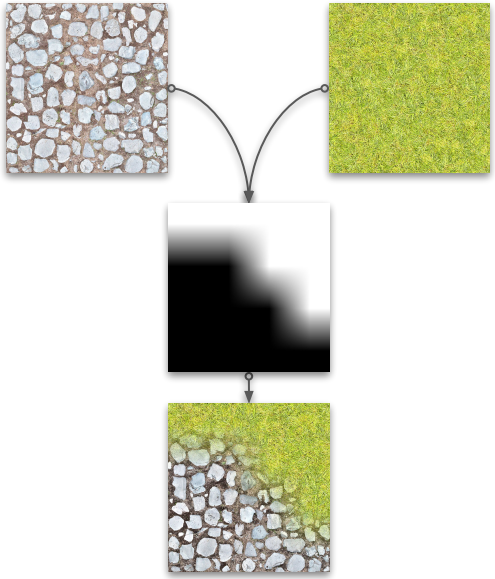
Materials - Custom

- Painted manually over rule-based materials.
- Max resolution: 4 samples/meter (but can be lower to save memory - using multires).
- Actual resolution - same as albedo - 1024 samples/meter
  - **Don't "blend" grass with rocks!**
- How to upsample integer data?

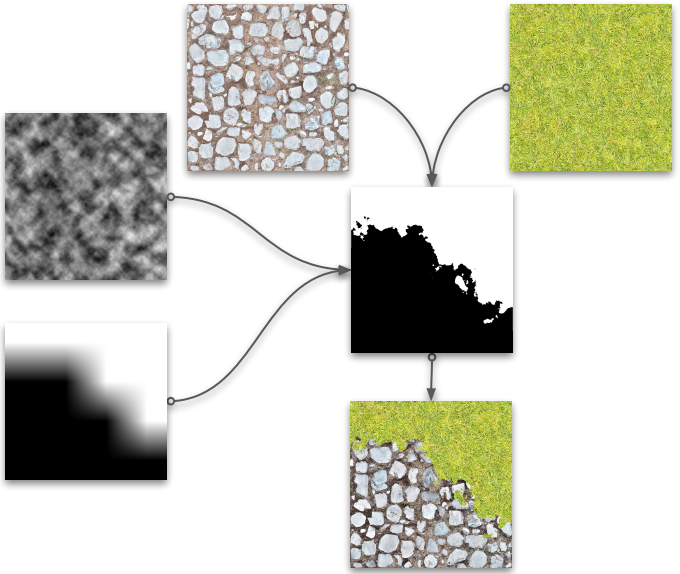
How to Mix Coarsely Defined Materials?



Erm... nope?



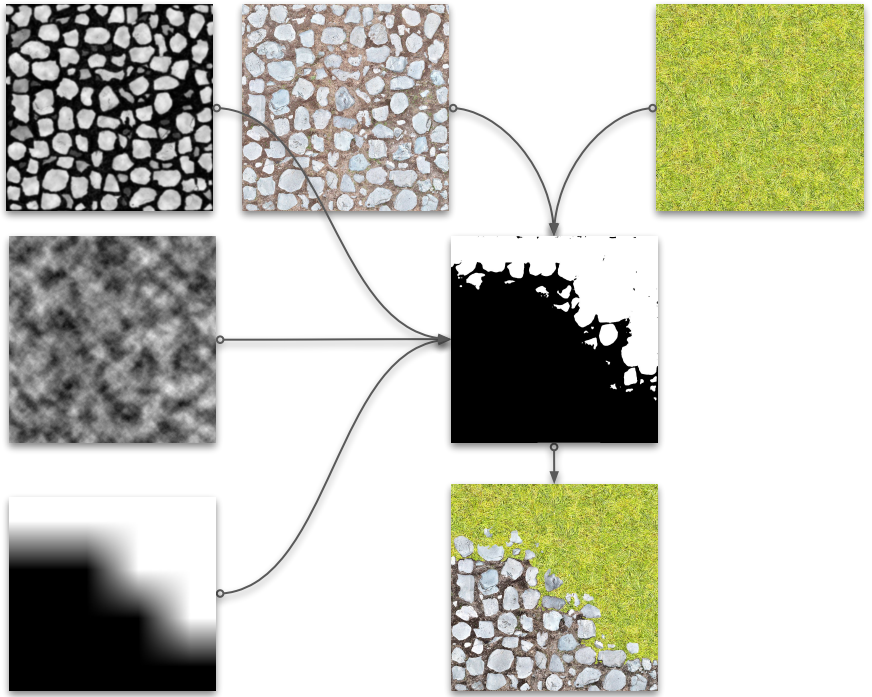
Blurry - still nope?



"Noisy Choose" (better)

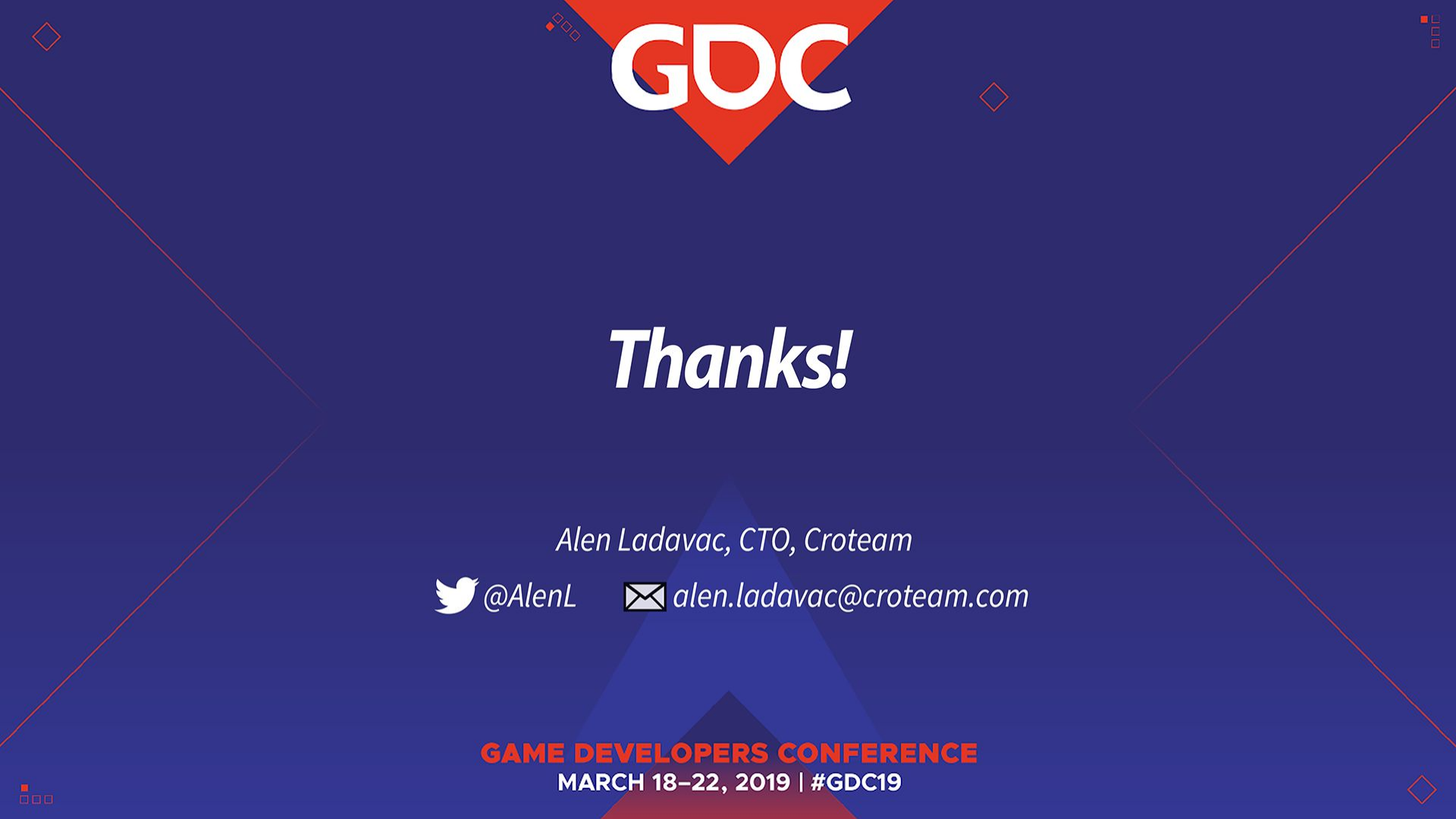
Height-Based Transition (Best)

- α × noise × height
- For multiple materials, pick the one with the highest value.



Oh, and one more thing...

Video #3


GDC
Thanks!
Alen Ladavac, CTO, Croteam
@AlenL
alen.ladavac@croteam.com
GAME DEVELOPERS CONFERENCE
MARCH 18–22, 2019 | #GDC19