GDC

Technical Artist Summit: Real-Time Level-Surface Volumetric Simulation Reductions

Dr. Russell Campbell
Vancouver Island University
Computer Science Faculty

Eli Landa
Vancouver Island University

SimChop

Introduction

Dr. Russell Campbell

- Teaching computer science:
  - Vancouver Island University
  - University of the Fraser Valley

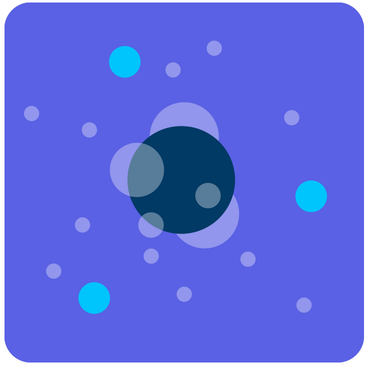
Eli Landa

- Studying computer science at VIU
- KCP Japanese Language Speech Contest winner (2017 Aug 5)

Open Source Project

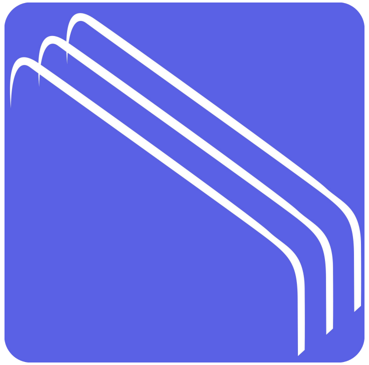
Description of Our Model

Parts of the System



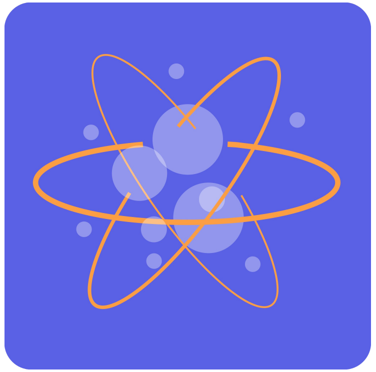
Colliders as Particles

No geometry for particles and only sphere colliders and vector positions. The particles are interactive with the game world.



Layers of Meshes as "Level Surfaces"

Visualization of particles via intersection with many planes sliced through the volume in front of the camera. The planes stay perpendicular to the camera view.

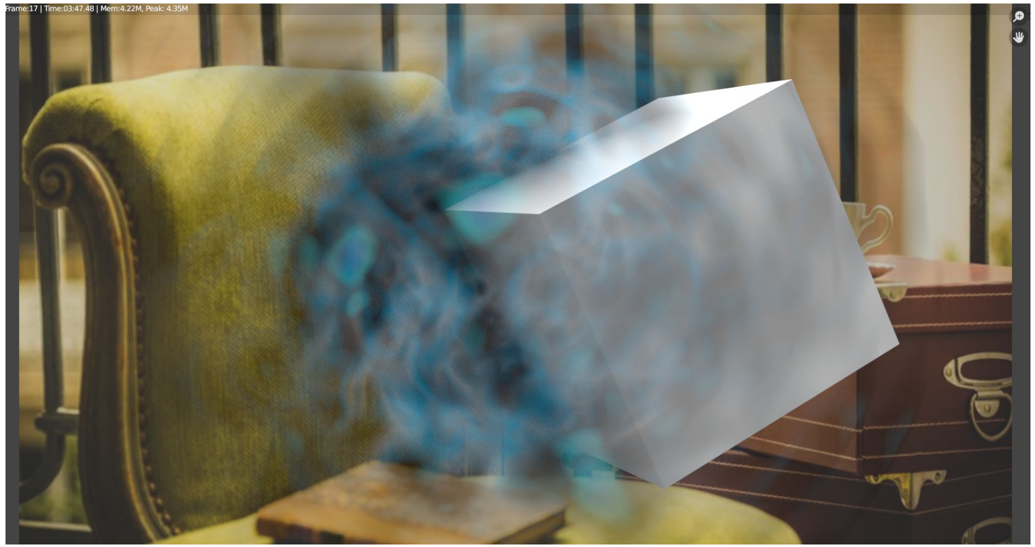


Data Structures and Shaders

Collisions are managed with commonly used graphics data structures, such as octrees or Z-order curve represented by interleaved position data.

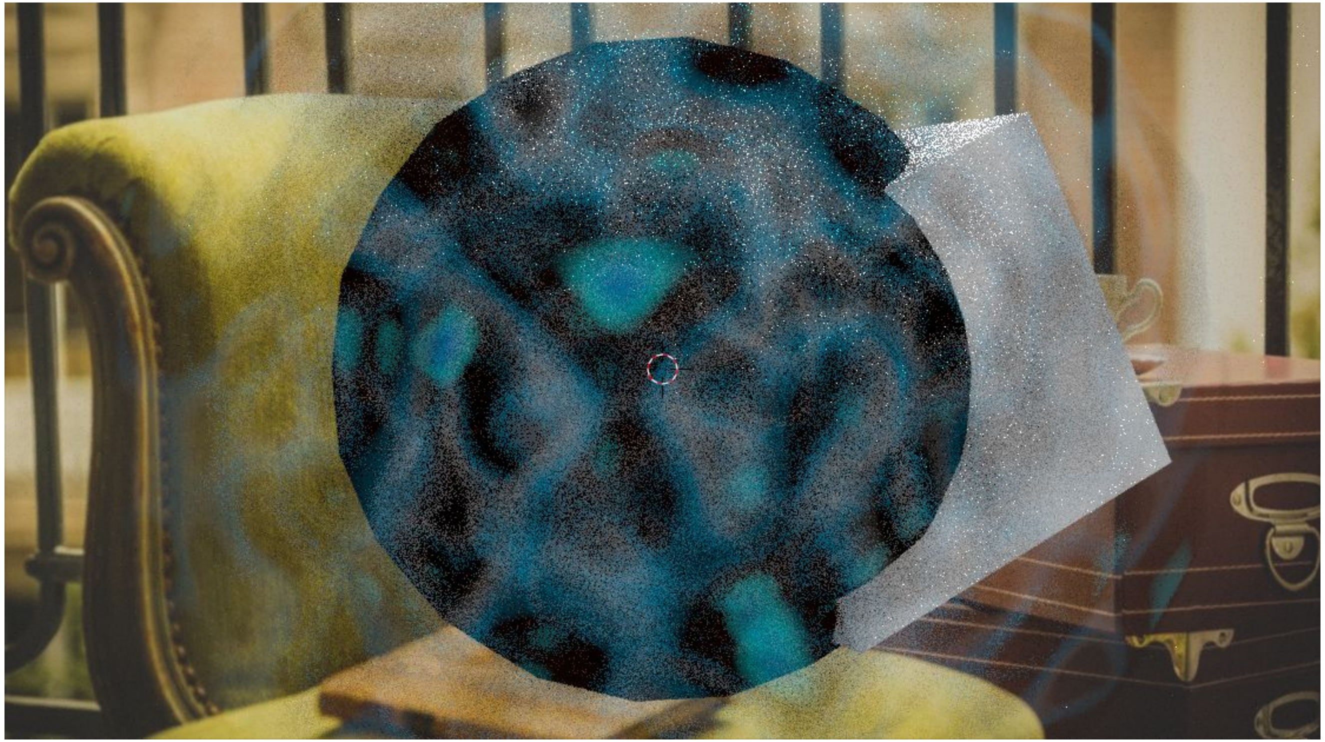
Mock-up Test

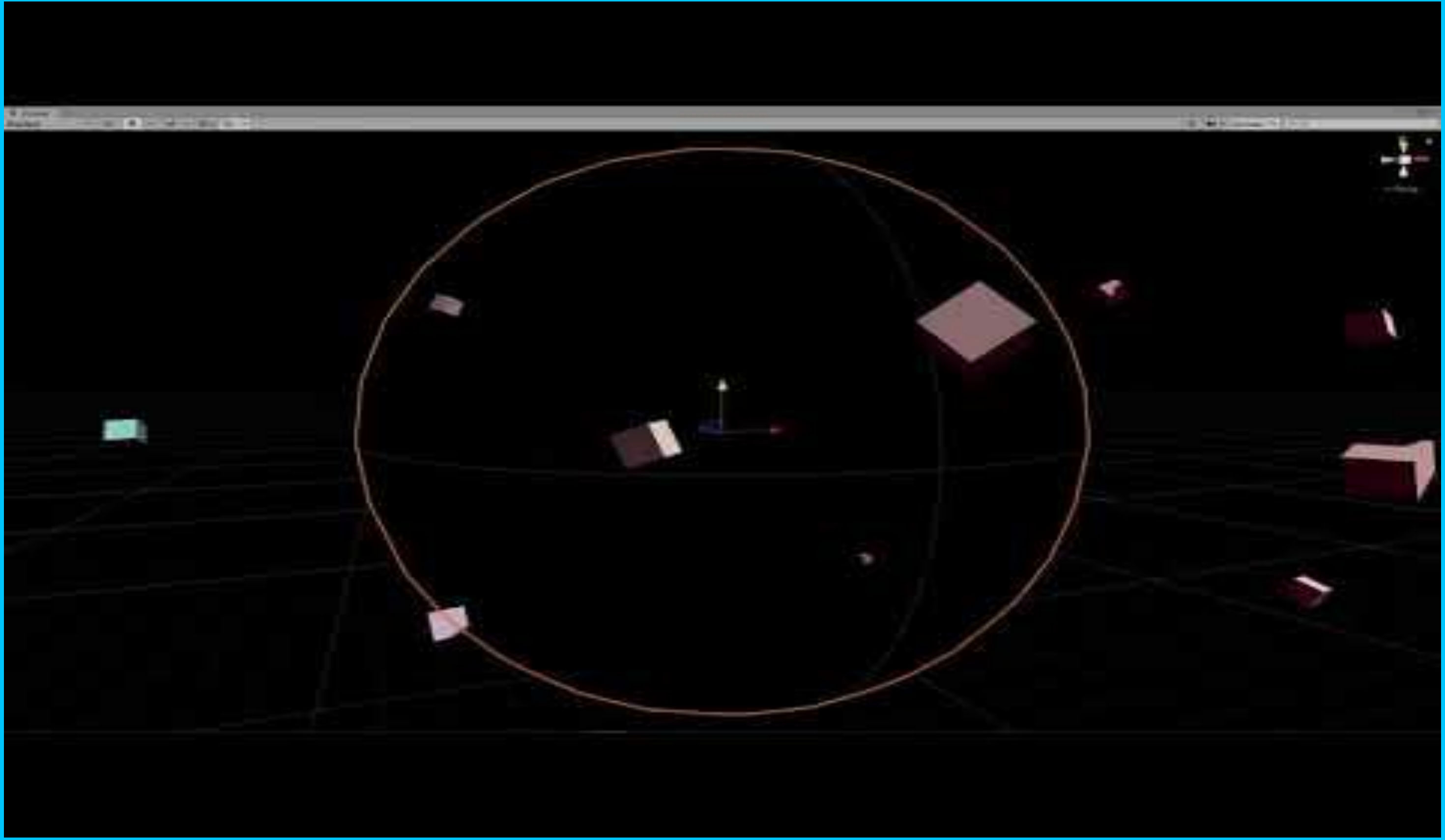
- Blender with Cycles renderer
- Sphere level surfaces with shader editor and nodes
- Tested interaction with other geometry
- Normals perpendicular to camera more transparent



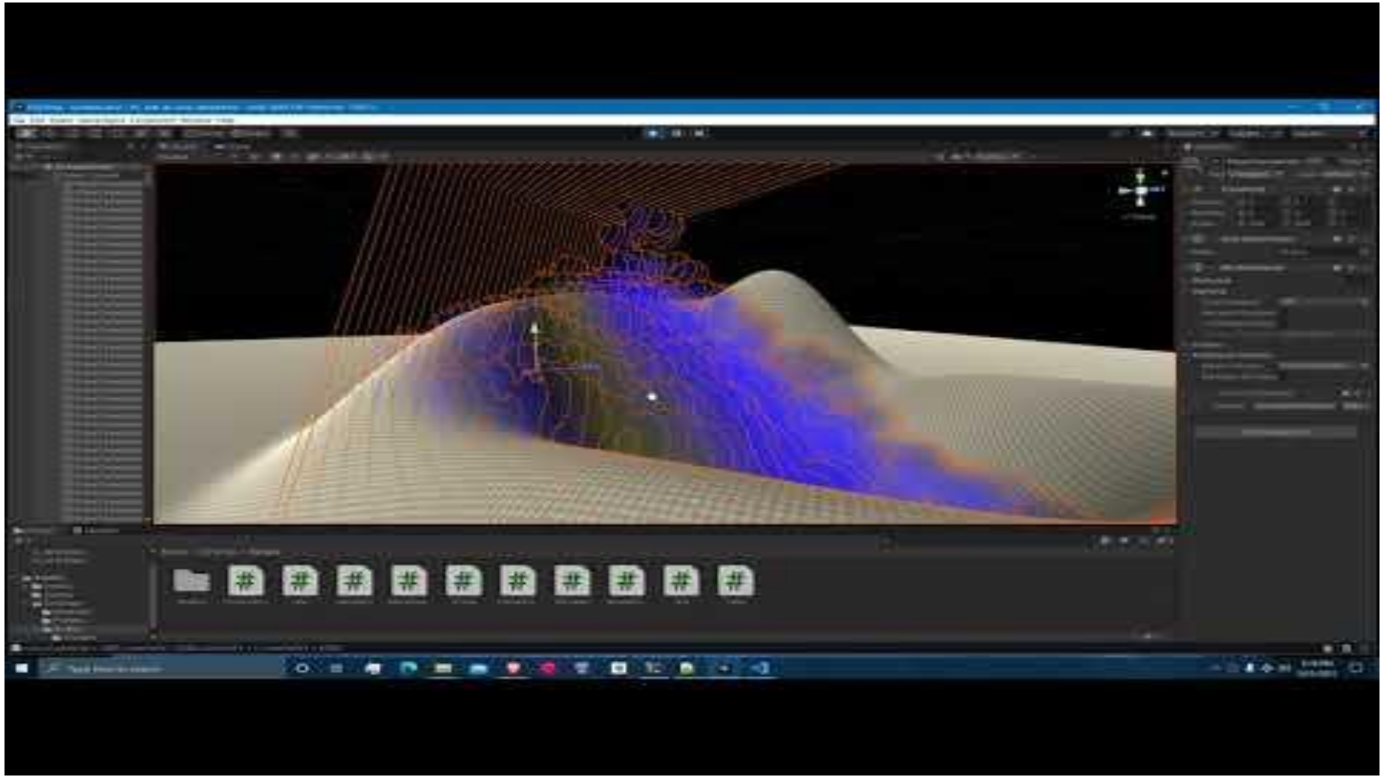
```
Frame:17 | Time:03:47.48 | Mem:4.22M, Peak: 4.35M
```

Mock-up Test





Prototype Demo

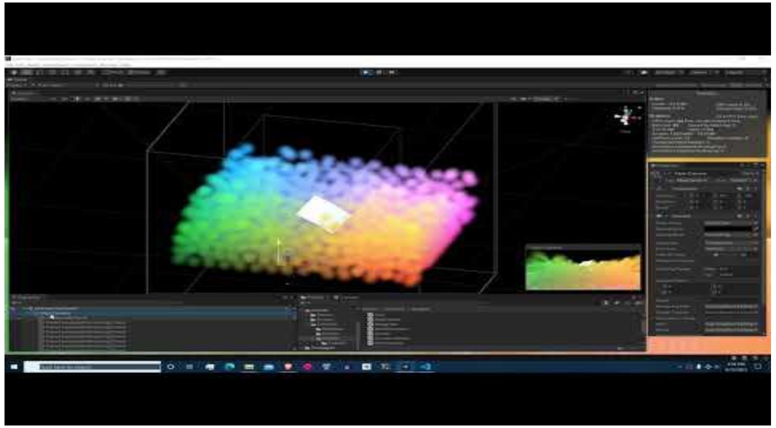


System Overview

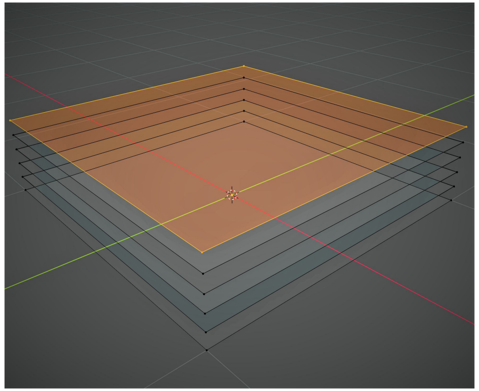
Summary of Development

- Different level surface geometry design
- Data structures:
  - Arrays
  - Octrees
  - Interleaving
- Passing data structures to shader code
- Volume mapping
- Exploring Use Cases

- Culling vertices of level-surface geometry
- Sorting using Unity's Job System
- Interactivity with Unity's Physics Engine
- Advantages of data structures in the shader code for design possibilities

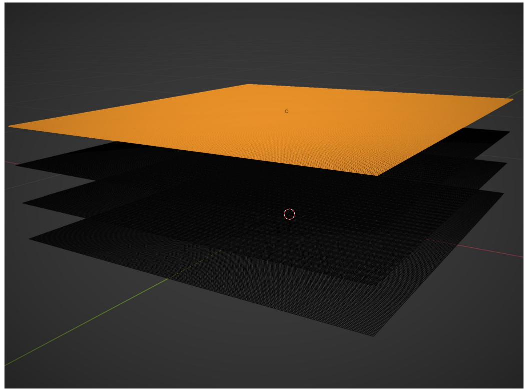


Level-Surface Geometries



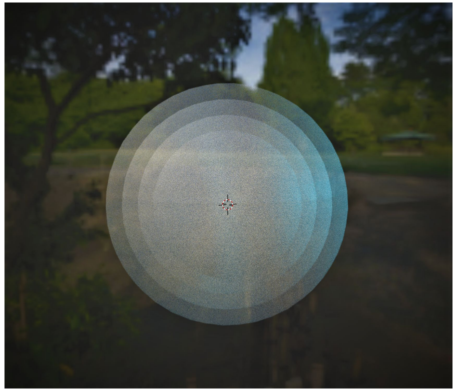
Planes

Four vertices for a plane means almost all the visualization is calculated in the fragment shader, which results in a slower system than the other geometries.



Tessellated Planes

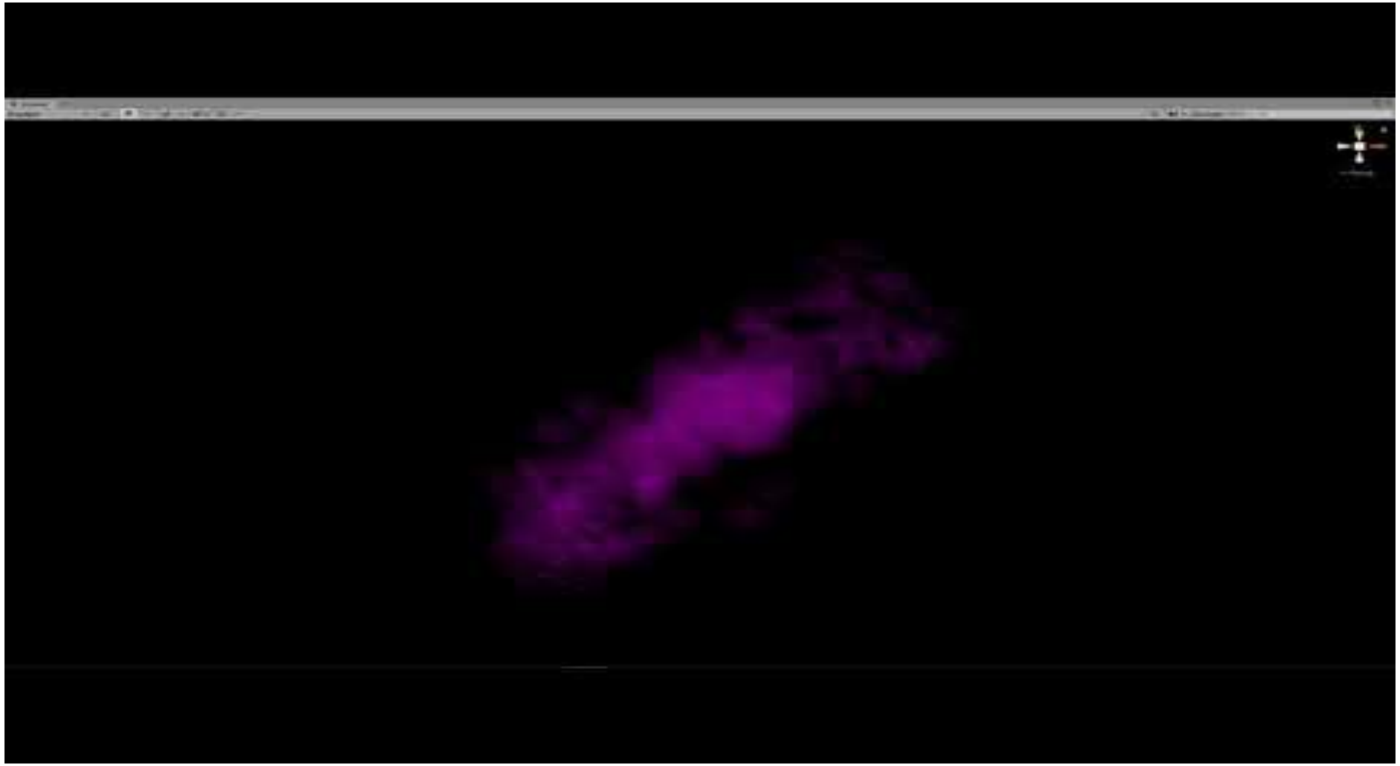
More vertices in each plane allows for culling any geometry that does not intersect nearby the particle positions. This saves the fragment shader from executing needlessly.



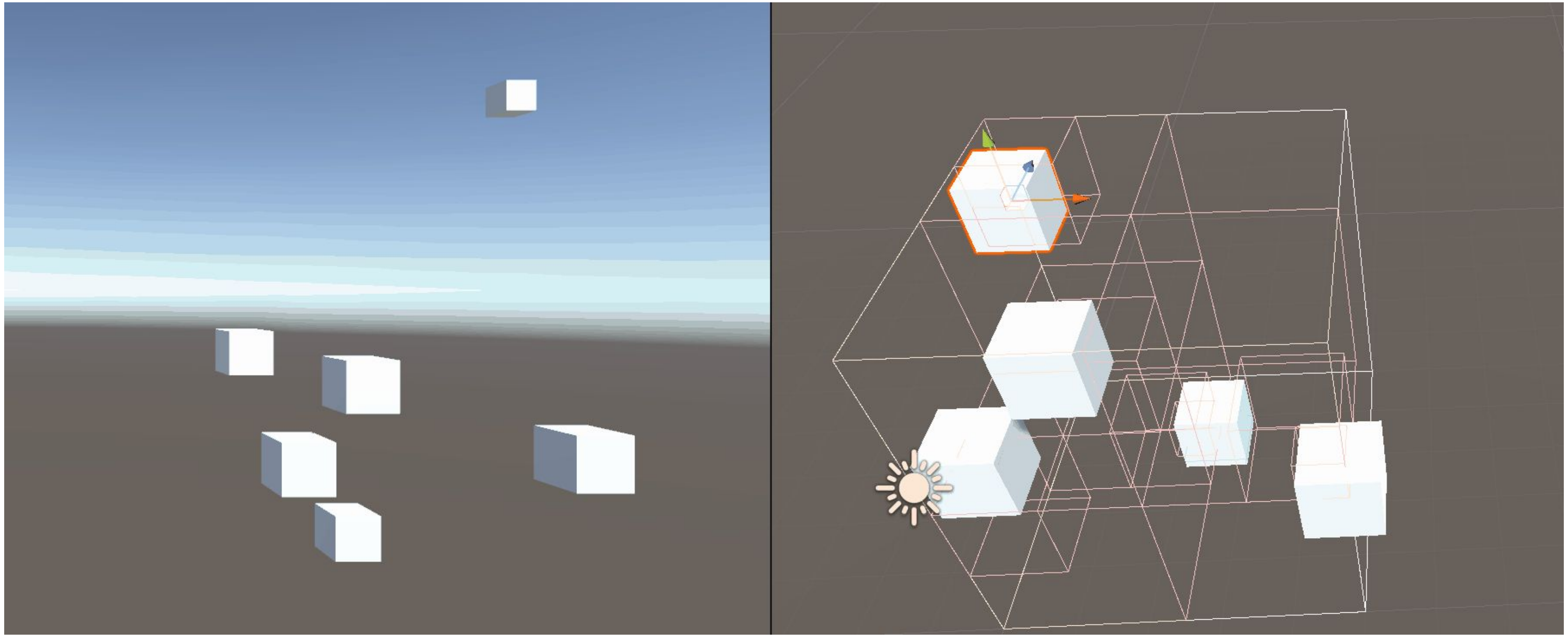
Concentric Spheres

Different geometries could be used for visualizing, depending on the desired application. In some uses, the geometry itself could remain more visible.

Data Structures: Arrays

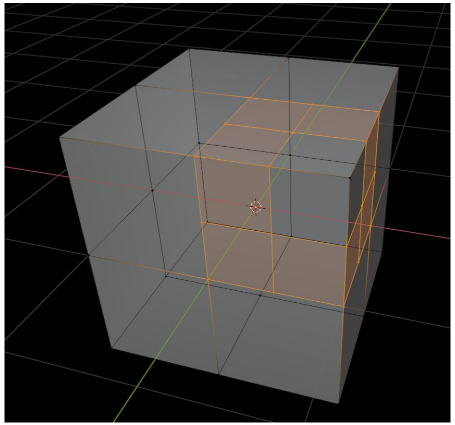


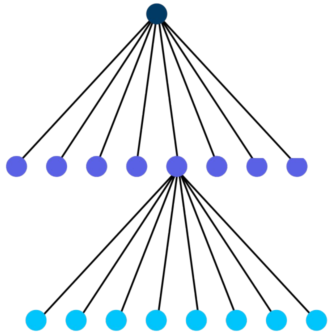
Data Structures: Octrees



Data Structures: Octrees

- GPU Gems 2, edited by Matt Pharr
  Chapter 37. Octree Textures on the GPU
- Passed in to shader via Texture3D
- Indirection uses texture lookup function `tex3D`
- Octree cell has eight octants, stored as 2x2x2 pixels
- Texture lookup takes a vector (x, y, z)
  - texture has normalized dimensions
  - (x, y, z) has $0 \leq x, y, z \leq 1$
- About $\log_8$ number of steps to traverse from root to leaf node



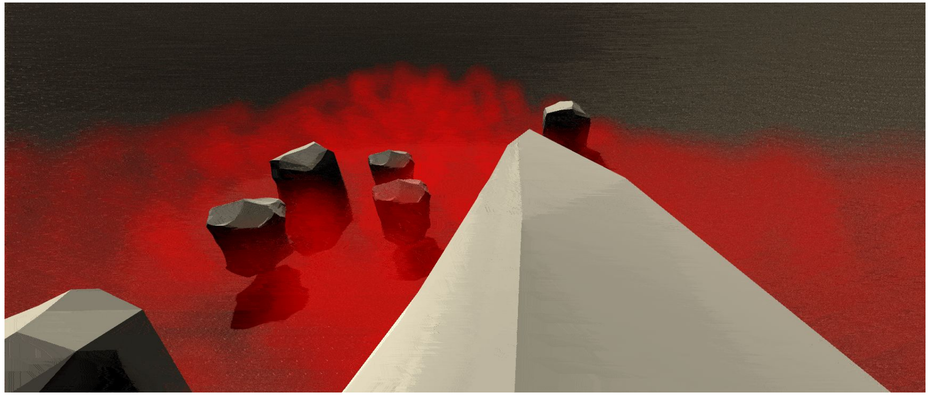


Data Structures: Octrees with Hash Indexing (1)

- GPU Octrees and Optimized Search
  D. Madeira and L. Thomas
  http://www.ic.uff.br/~esteban/files/papers/SBGames09_Madeira.pdf

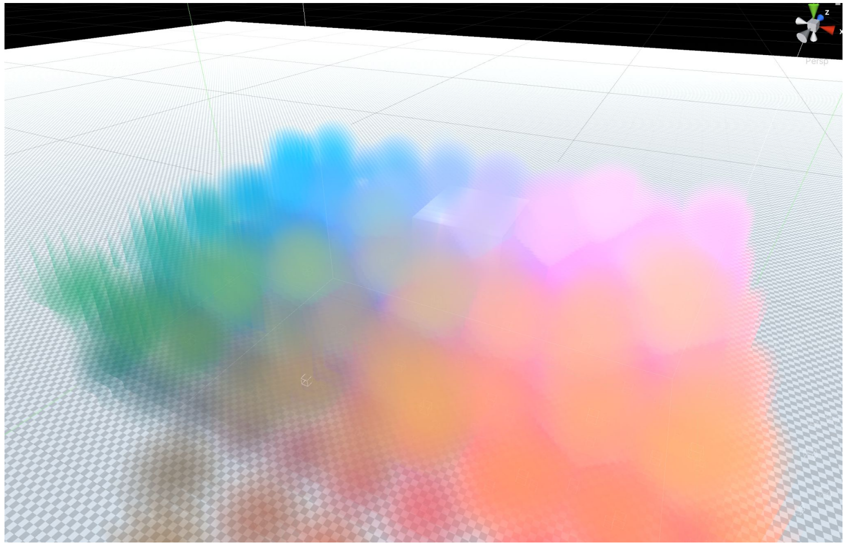
- Dealing with hash collisions using linear probing reduces performance, so...

  ...we do not use parent nodes!

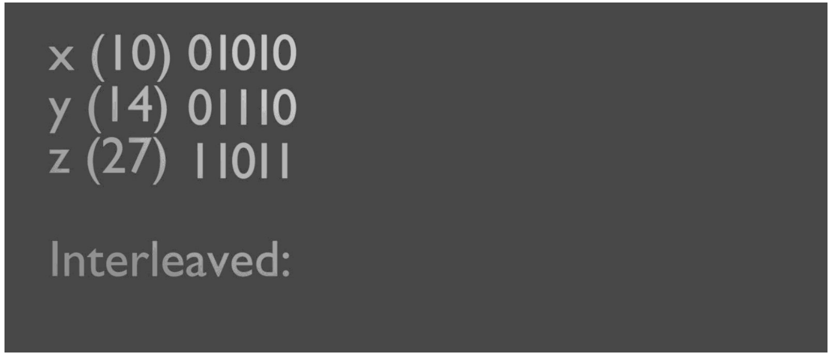


Data Structures: Octrees with Hash Indexing (2)

- Morton codes (describes Z-order curve) are used as hash table keys
- Faster access in shader than indirection pool tree traversal
- Another speed-up:
  - more vertices in planes
  - vertex shader sends distance calculation for the closest particle to the fragment shader



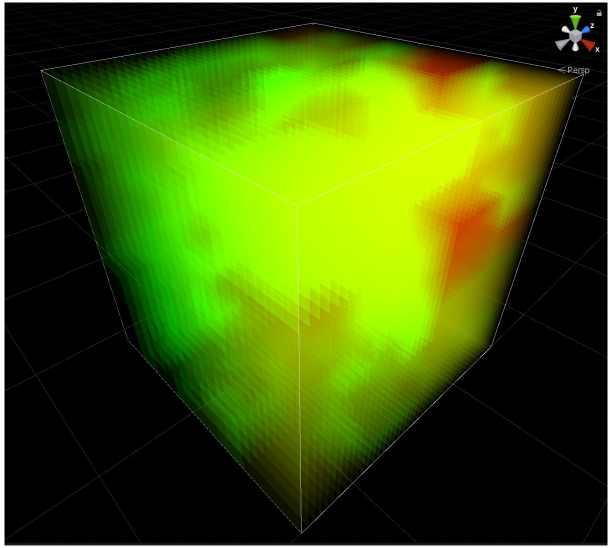
Data Structures: Interleaving (1)


x (10) 01010
y (14) 01110
z (27) 11011

Interleaved:

- Z-order curve of particles based on interleaving binary digits for x, y, and z values
  https://redis.io/topics/indexes#multi-dimensional-indexes
  (accessed: 2021 May 12)
- Queries through a subvolume are restricted to similar octants as octrees
  e.g.: set the last 3 digits to 0 and iterate / increment until the last 3 digits are all 1
- Using binary means we can split a volume, each dimension in half recursively, similar to octants

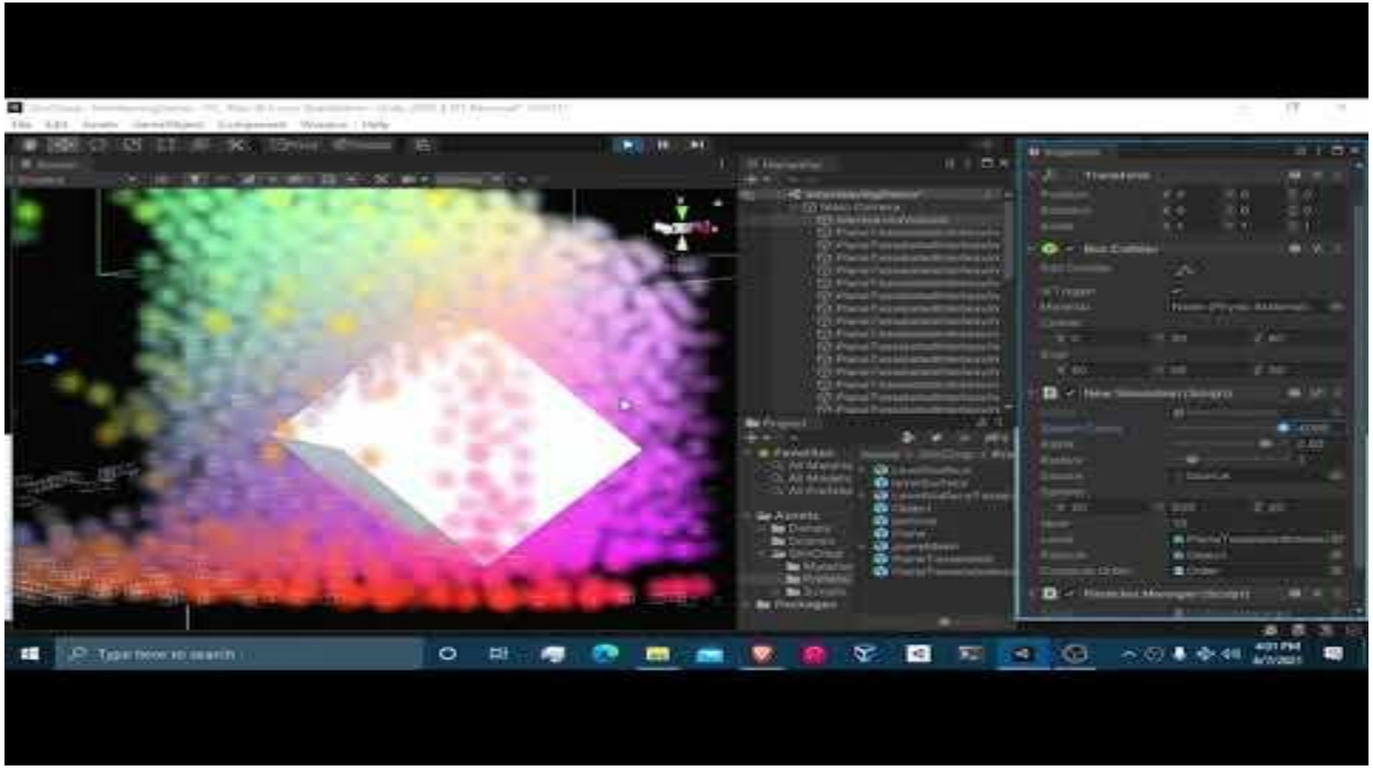
Data Structures: Interleaving (2)



- Morton codes give indices (describes Z-order curve)
- Quicksort with Unity Jobs / binary search in shader
- Left figure visualizes binary search for 250 particles
  - More than 32 bits means splitting across colours
  - Volume side-length divided by 2^k for cell size
    - Morton code would then have 3k bits
- Level surfaces were quite useful for debugging shader!

- Slight speed-up with double interleaving: pass in two Texture3D, one shifted half a cell

Data Structures: Interleaving (3)

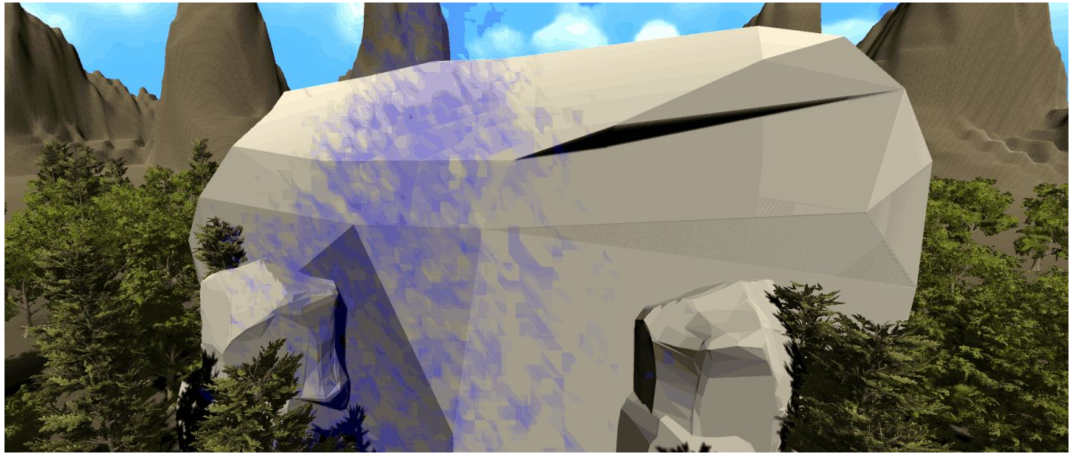


System Configurations

- In isolation, octree with indirection pool:
  - 900 collidable particles and 20 level surfaces, ~30 fps (NVIDIA RTX 2060)
- In isolation, octree with hashed indexing:
  - 2500 collidable particles and 50 level surfaces, ~40 fps (NVIDIA RTX 3070)
- In isolation, double interleaving:
  - 3000 collidable particles and 30 level surfaces, ~50 fps (NVIDIA RTX 3070)
- Estimates above heavily depend on configuration of radius, resolution of data structures, etc.

Limitations

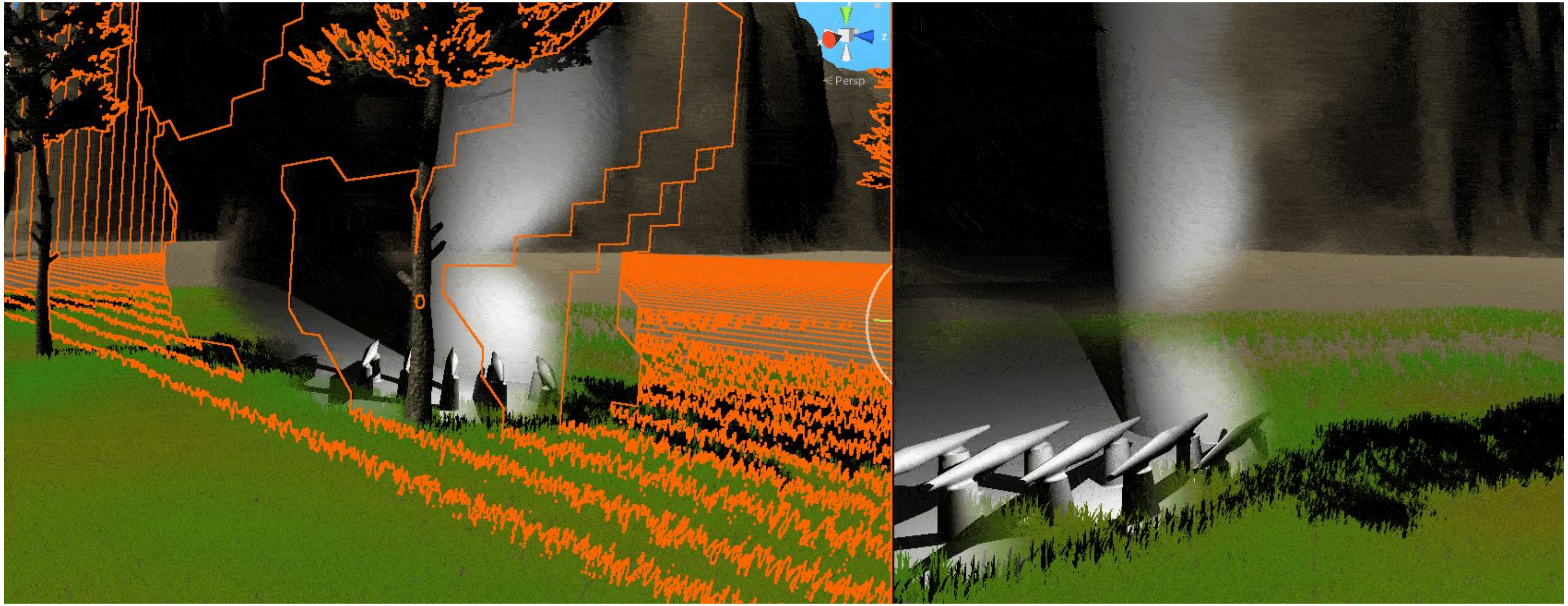
- The radius of particles is limited by the octant sizes (and currently only one size particle)
- Colliders interact with Unity Physics, but these calculations add up for large numbers of particles
- Global arrays with a maximum number of elements, or forced values across colour channels
- Intersections with other flat geometry show sharp edges
- Camera zoom-in and -out causes change in FPS



Demo Examples

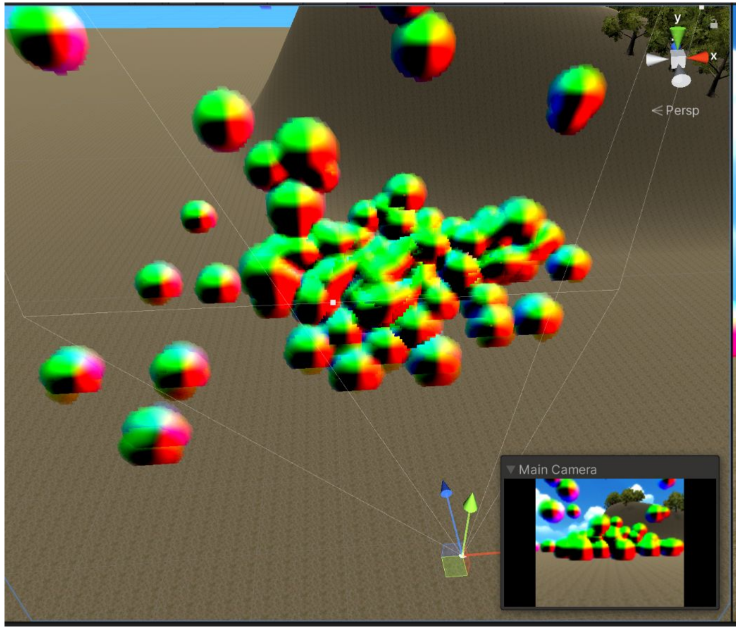
Fuzzy and Interactive Portals

- all particles rendering the same raymarched scene with Voronoi-cell transparency

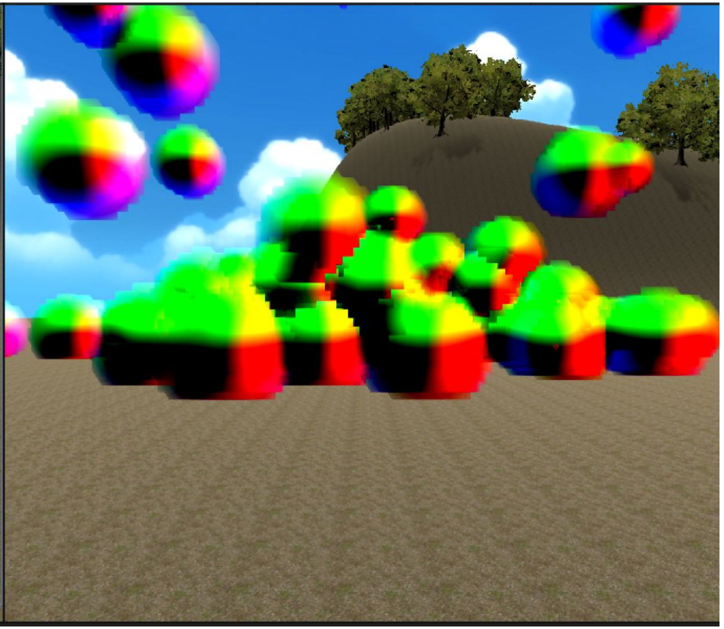


Raymarched Particles (1)

- For a copy of geometry at each particle, we use a local coordinate system for each particle



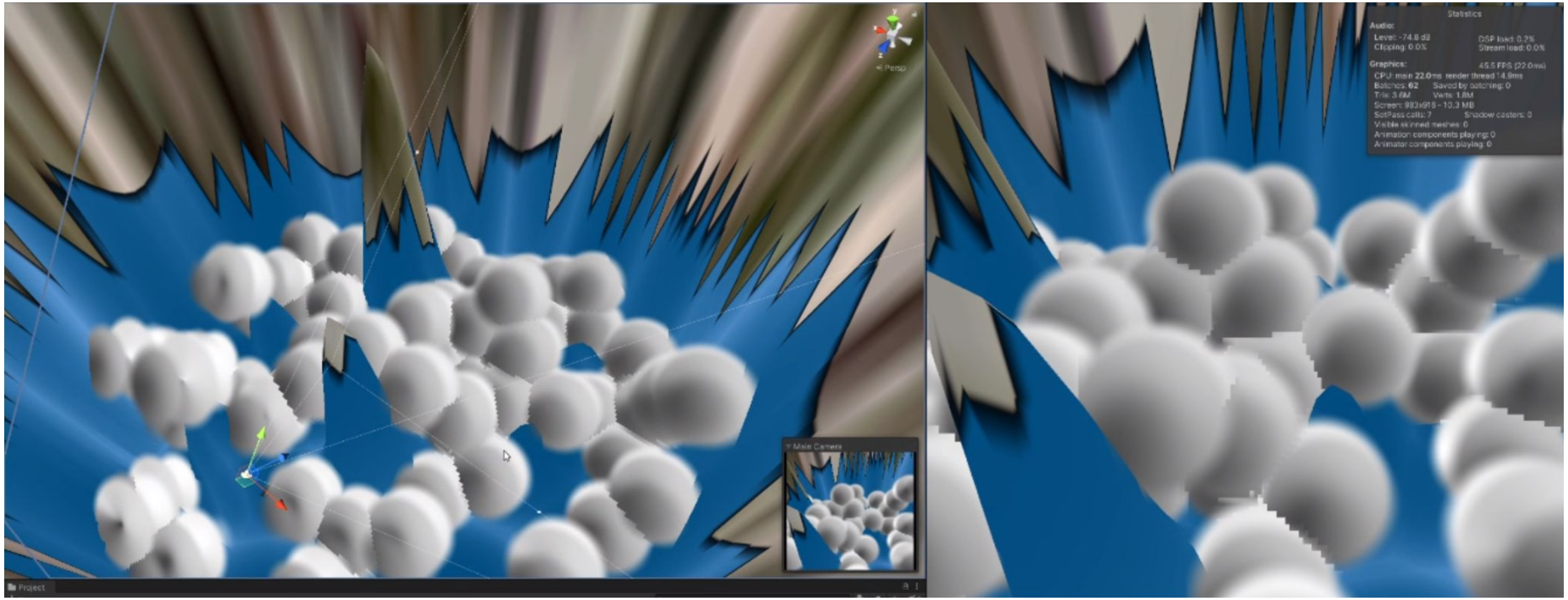
Scene view



Camera/Game view

Raymarched Particles (2)

- An example of raymarching the same SDF local to each particle

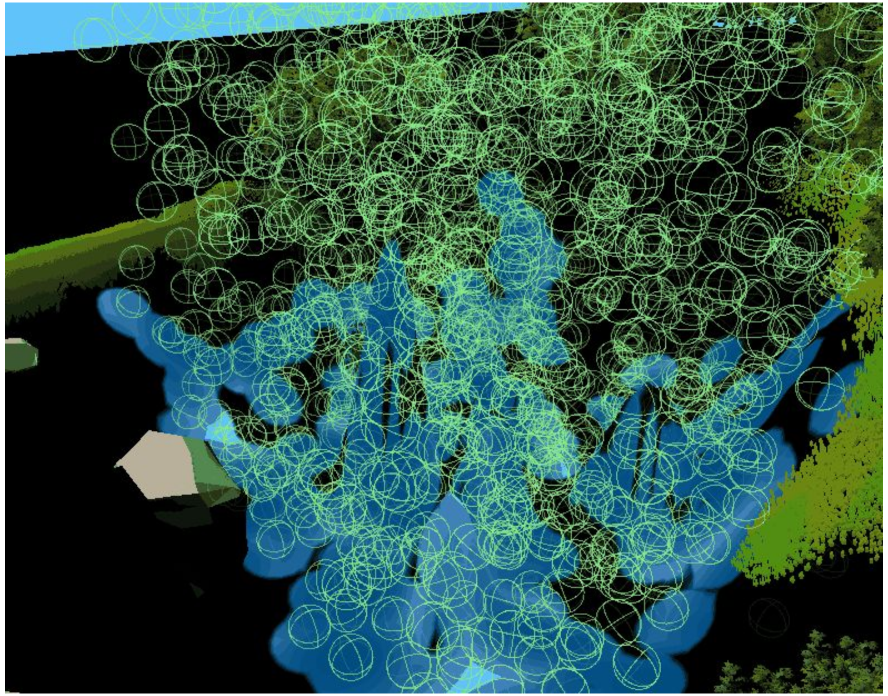


```
Statistics
Audio:
  Level: -74.8 dB                    DSP load: 0.2%
  Clipping: 0.0%                     Stream load: 0.0%

Graphics:                            45.5 FPS (22.0ms)
  CPU: main 22.0ms  render thread 14.9ms
  Batches: 62      Saved by batching: 0
  Tris: 3.6M       Verts: 1.8M
  Screen: 983x918 - 10.3 MB
  SetPass calls: 7              Shadow casters: 0
  Visible skinned meshes: 0
  Animation components playing: 0
  Animator components playing: 0
```

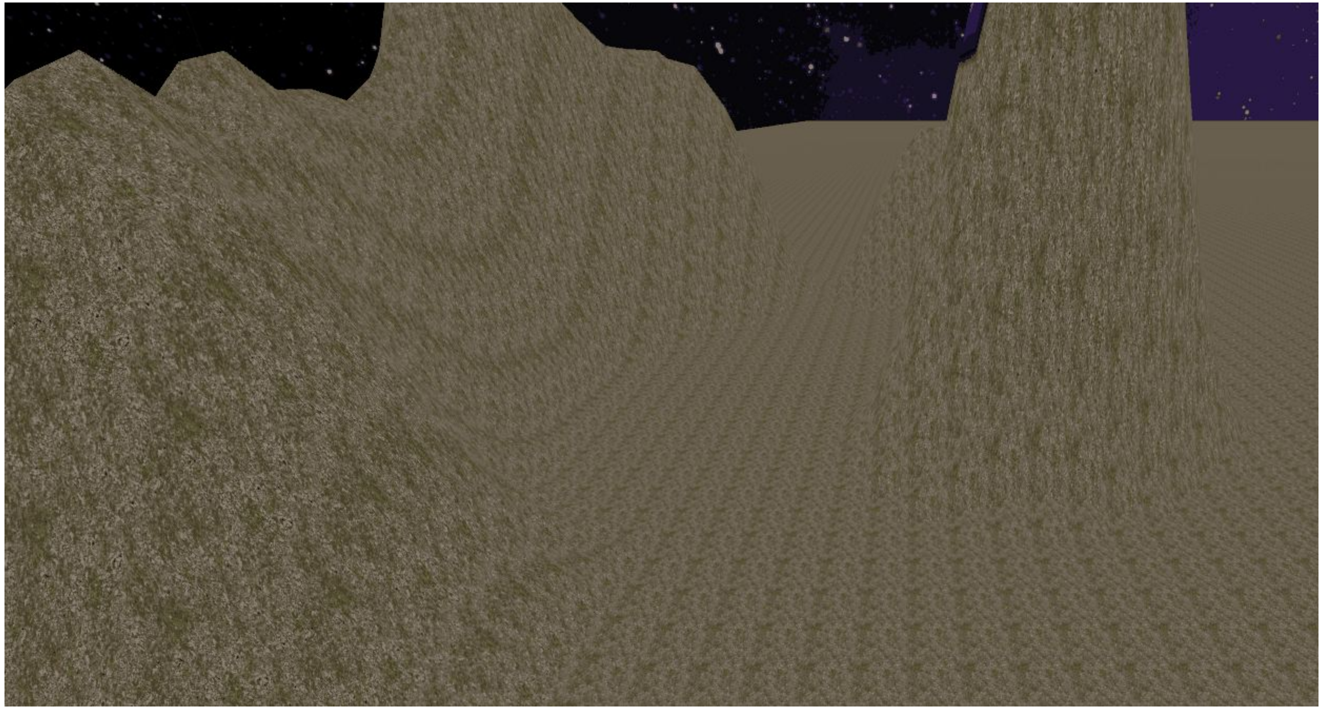
Caustics

- Textures on surfaces underneath can be shaded with the particle positions, for a refractive effect



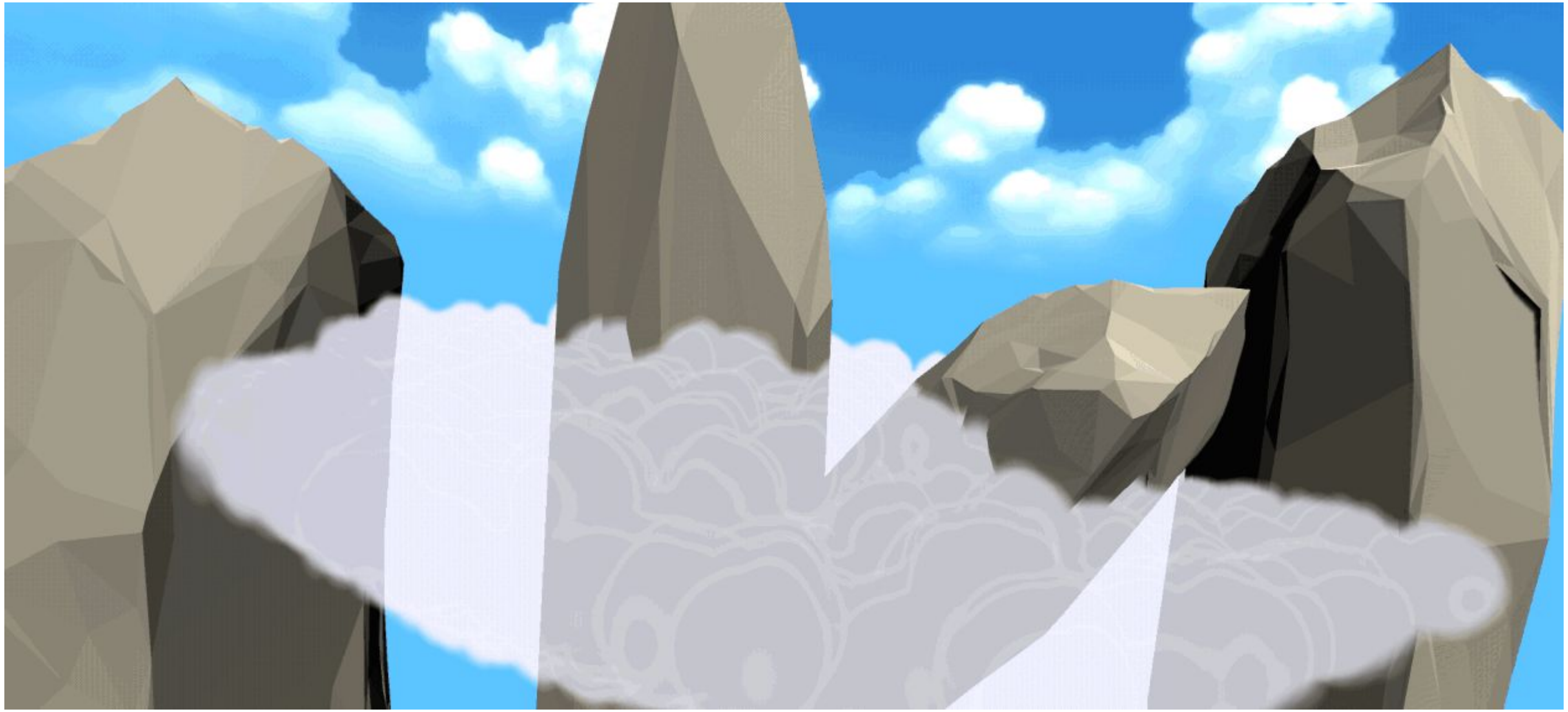
Water

- An example of caustics together with shading particles for a more water-like result



Variations

Variations

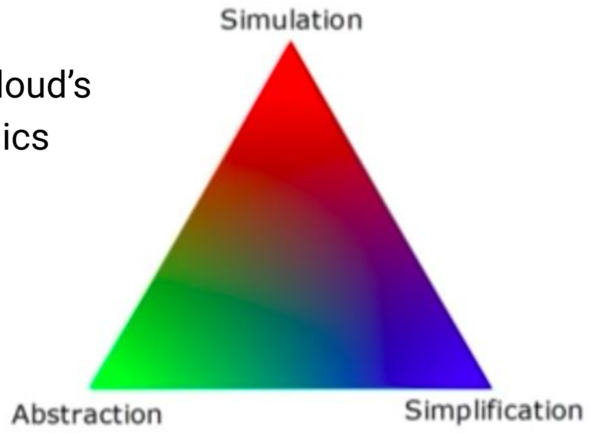


¿ Thoughts on Design ?

Continuum of Design

- Tyriq Plummer’s previous GDC presentation adapts Scott McCloud’s triangle continuum from the graphic novel Understanding Comics (GDC 2016, Made Out Of Meat: Health Systems In Video Games)

- Consider a further abstraction of this continuum that places our system in a broader context of design.



- We could place our system closer to the “simplification” corner.

- Simplification is useful for educational purposes, or perhaps as a quick test as part of a larger more complex system, or building more detail on top for structural support, etc.

Future Work

Future Work

- Optimizations with more culling
- More configurations through UI Editing
- Occlusion mask as a transparency to blend with intersecting geometry better
- A version of our system for DOTS when Unity releases its packages out of preview
    - early testing seems to allow ~15000 collidable particles in a subscene
- Combining with other systems
- Applications, such as atomic, chemical visualizations without the need for large file sizes
- More demo packages

```
SCORE:  20
DIFFICULTY:   4
```

```
Press ESC to exit the game.
```

Contact

screenshot: Yael Oosthuizen
(VIU CSCI 161 student)

Dr. Russell Campbell

✉ Email: Russell.Campbell@viu.ca
🐤Twitter: @RussCampell
▸ YouTube channel: Russell Campbell
🖌️ https://www.artstation.com/russellcampbell

Eli Landa

✉ eliplanda@gmail.com
🐤@eli_landa_
🖌️ ShaderToy: https://www.shadertoy.com/user/intrakits