Point Cloud Rendering

Christopher Jeffers, Eric Jensen, Joel Rausch

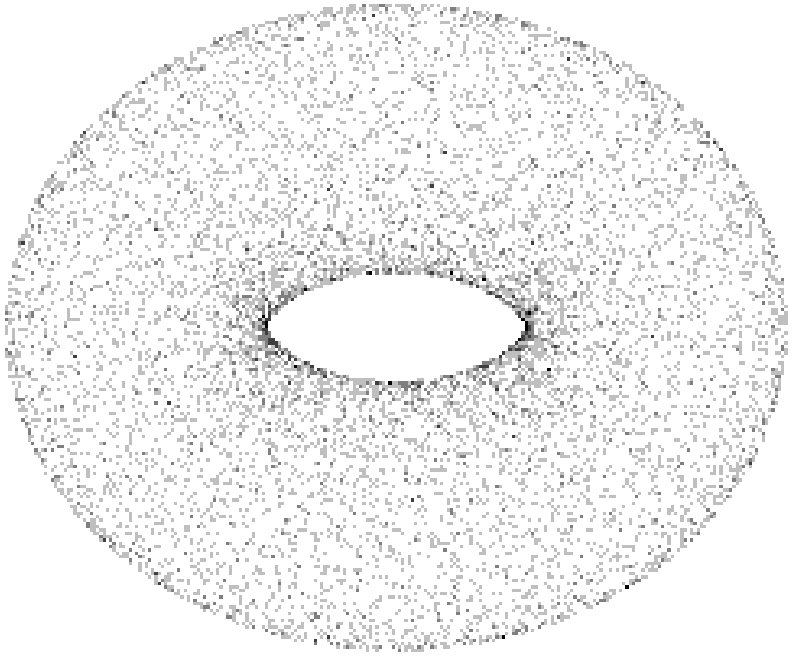
Purpose

- Siemens PLM creates software to view and compare 3D models.
- Some tools, such as 3D laser scanners, output these models as a large group of points in (x,y,z) space.
- This collection of points is known as a "point cloud".
- Siemens PLM desires a product that can render very large point clouds (100s of millions to a billion points).

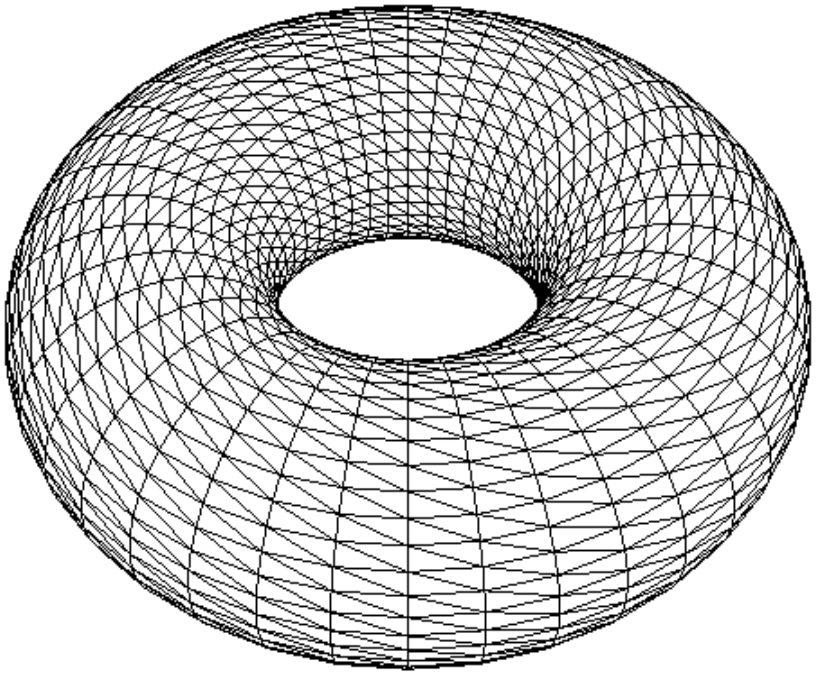
Problem

- Computer hardware is very good at drawing sets of triangles (meshes), not optimized for single points.
- Difficult and very demanding on hardware to draw every point, especially for very large (a billion point) point clouds.
  - Possible Solution: Use approximations of points to reduce number of points to be drawn.

Point Cloud vs. Mesh



Point Cloud

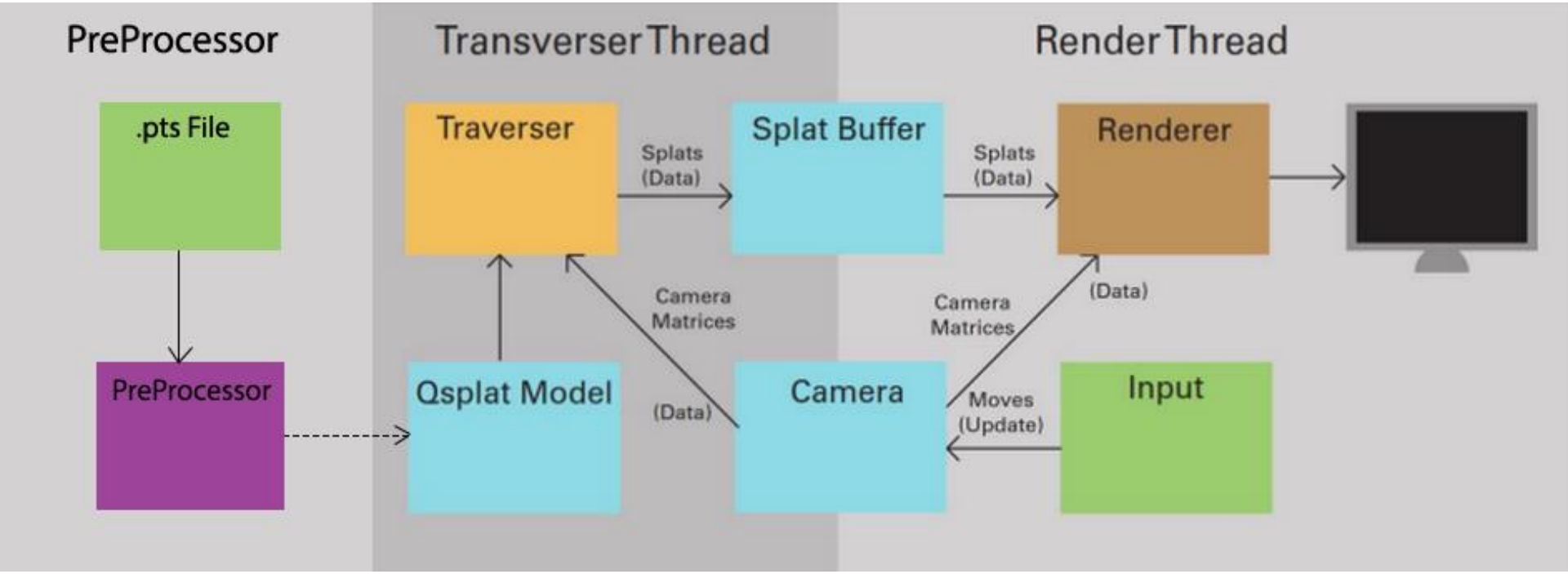


Mesh

Solution

- Our research led us to a system created at Stanford University in the year 2000, called QSplat.
- QSplat uses a very efficient and compressed tree structure to organize point cloud data.
- Traversing this tree allows us to find where a point should be rendered.
- Adapting QSplat to current technology allows us to reach our goal of rendering a billion points.

Program Structure



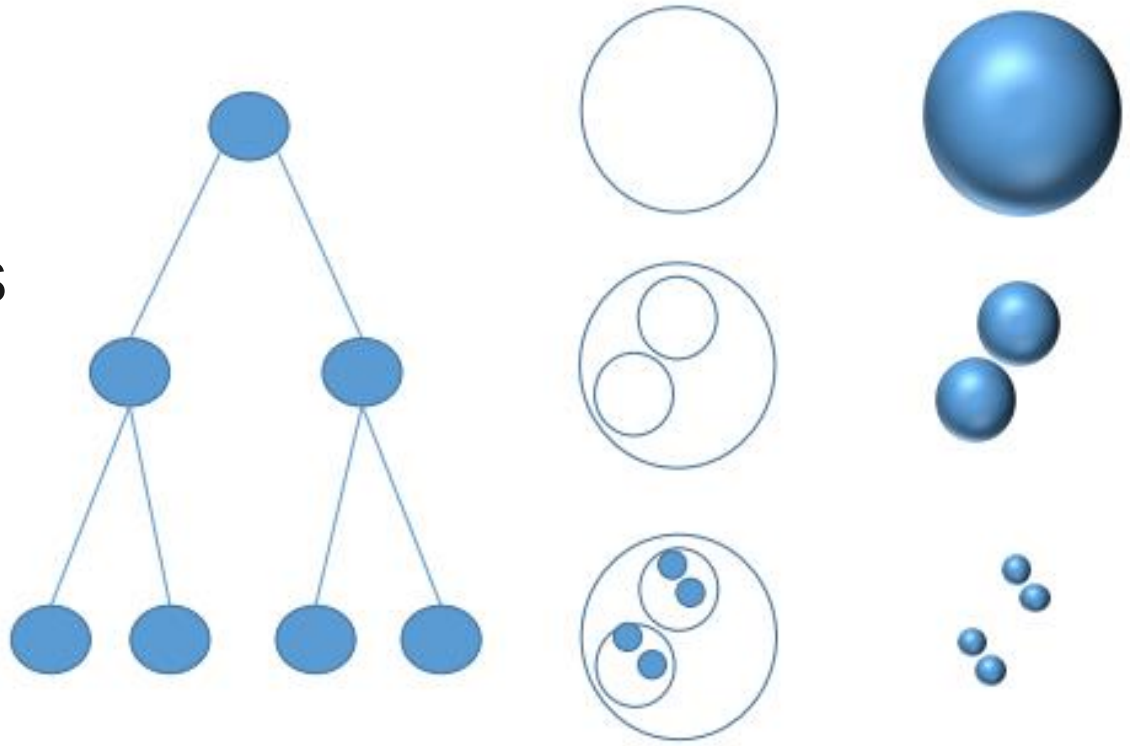
We perform two major tasks on separate, dedicated threads of execution. Traversal puts the points to render in a buffer, and the Renderer reads from this buffer and draws the points to the screen.

Traversal

- Choosing which points to render
- Discarding points that are not visible
  - Points not visible to the camera (frustum)
  - Points which are behind other points (occluded)
  - Points which are too small (sub-pixel size)
- Performance is affected by how points are organized.

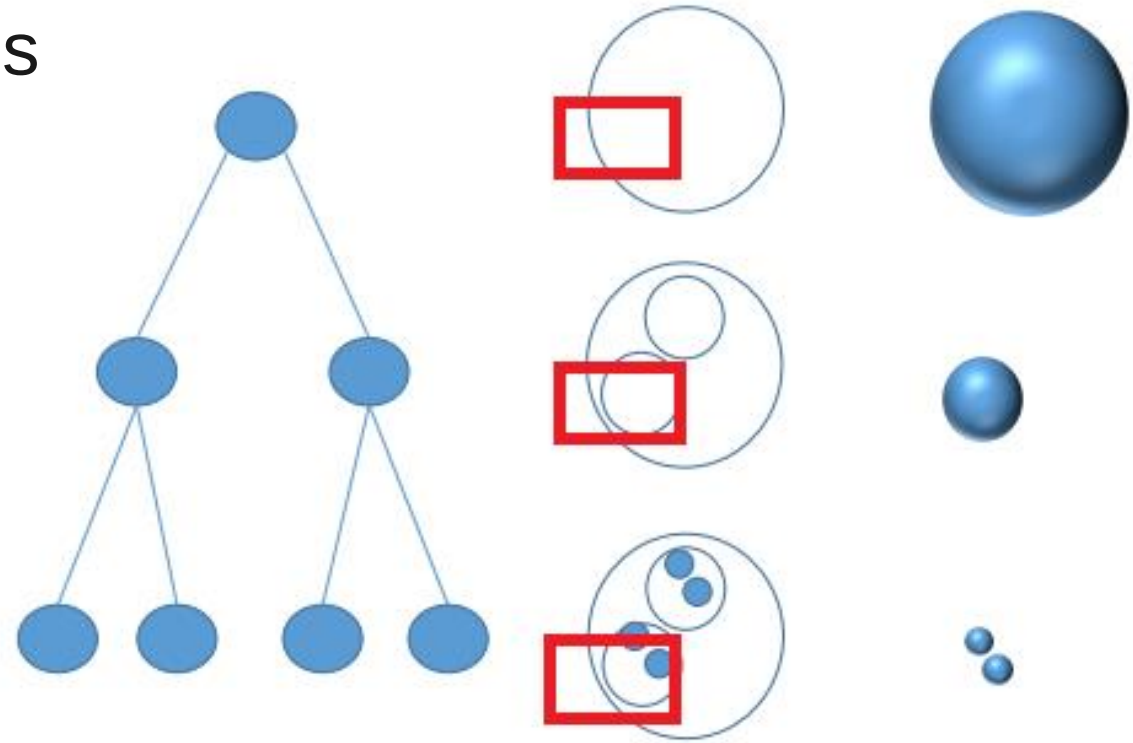
QSplat Tree Structure

- Points are leaves in the tree.
- Clusters of points are bounded by spheres, which form branches.



Traversing the QSplat Tree

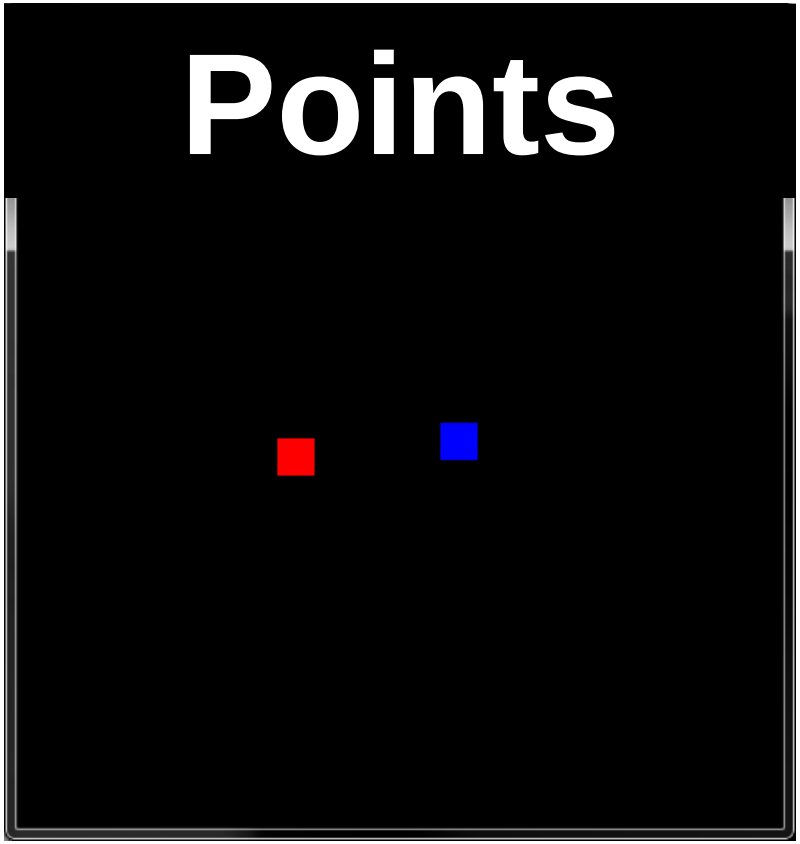
- Drawing branches approximates a cluster of points.
- Prune entire tree branches.
- Prioritize closer branches.

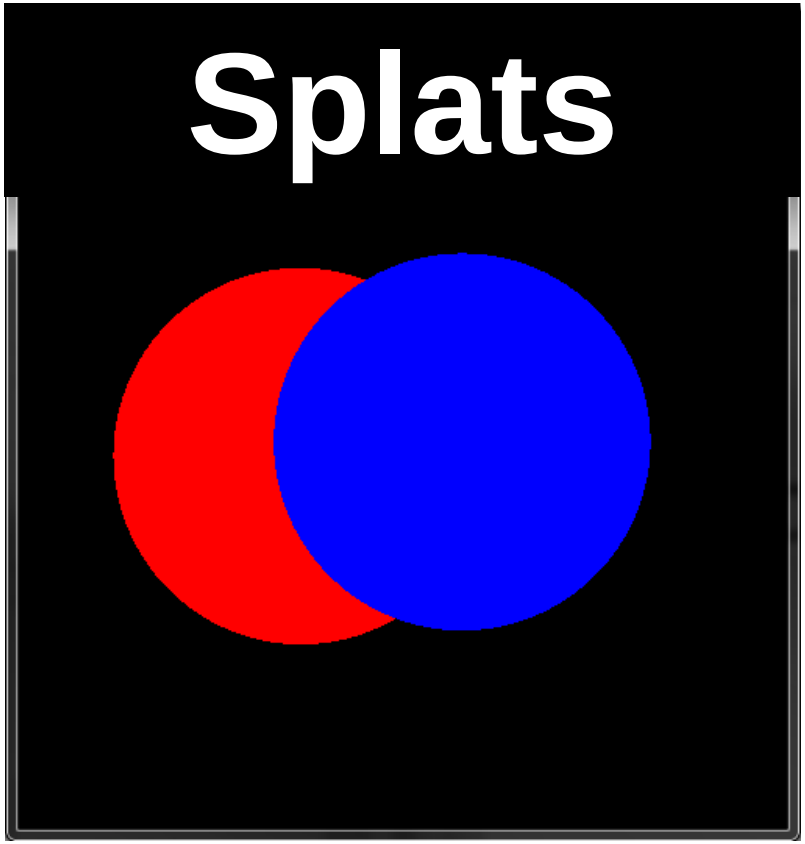


Rendering

- Take the points currently in the buffer and render them to screen at an interactive frame rate.

What is a Splat?

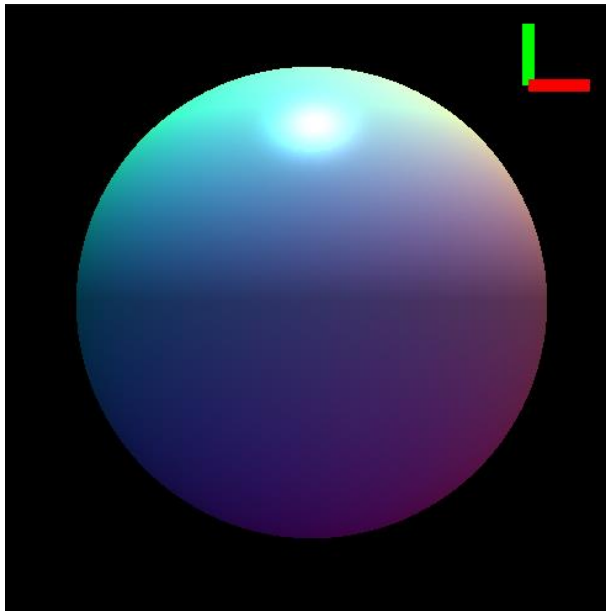




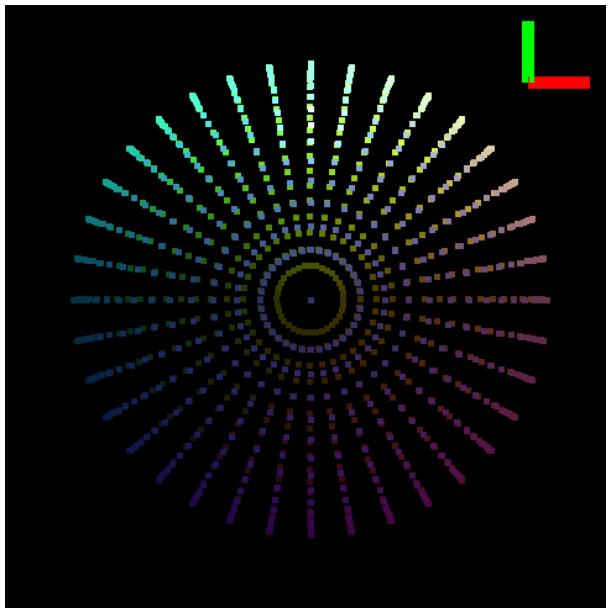
Why Use Splats?

- Convert a point back to a surface.

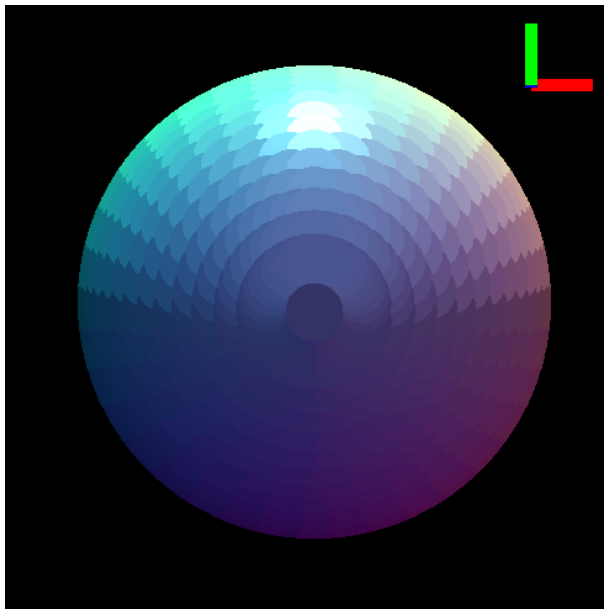
Real Object



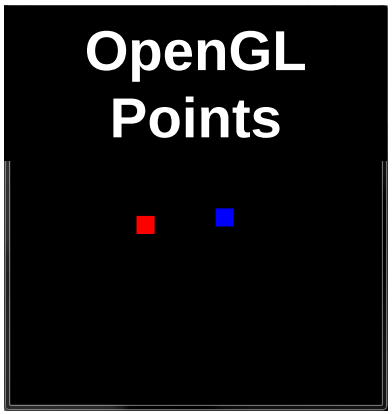
Sampled Points

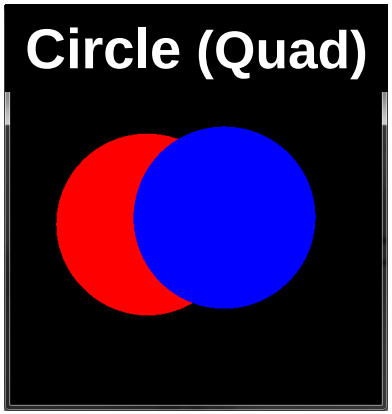


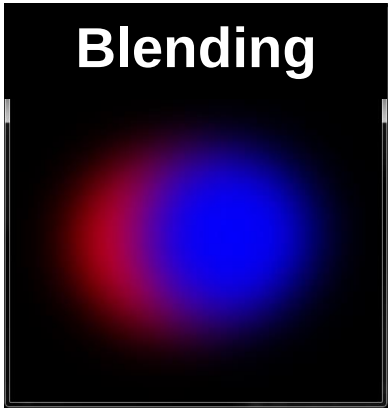
Approximated Surface

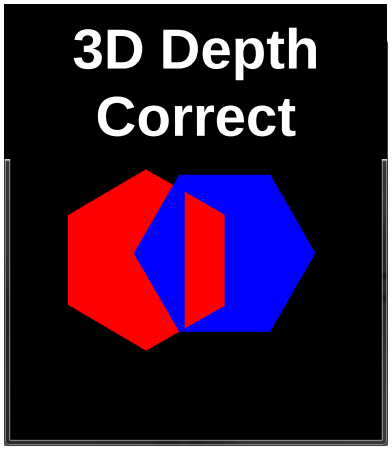


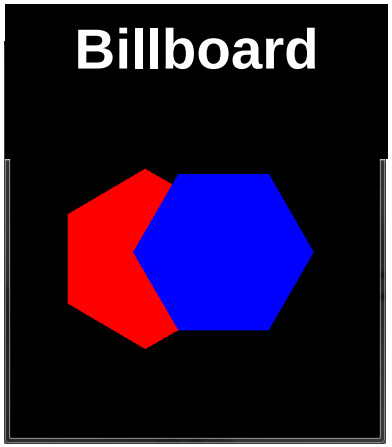
Types of Splats

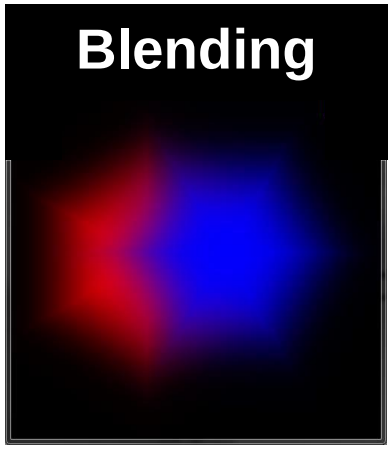




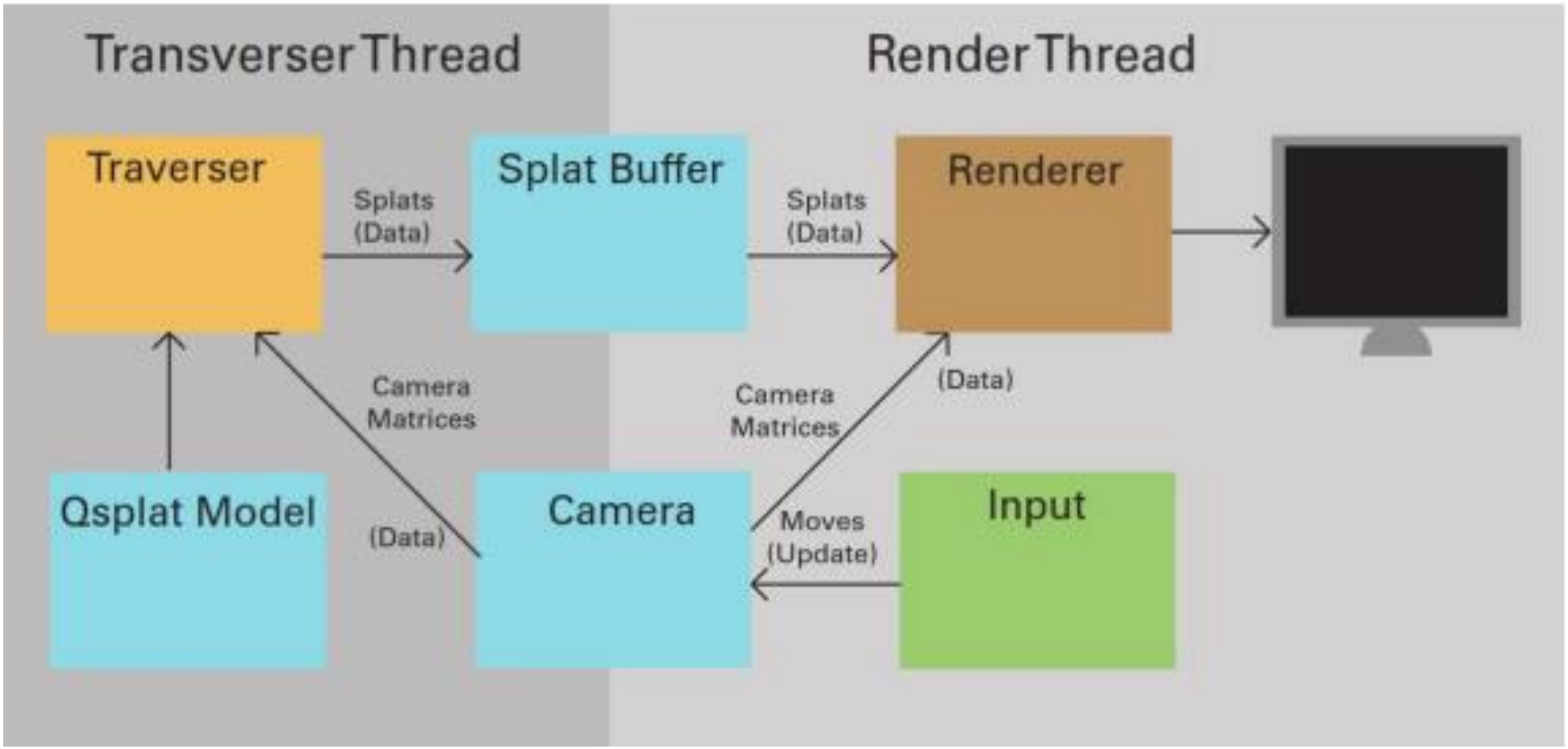




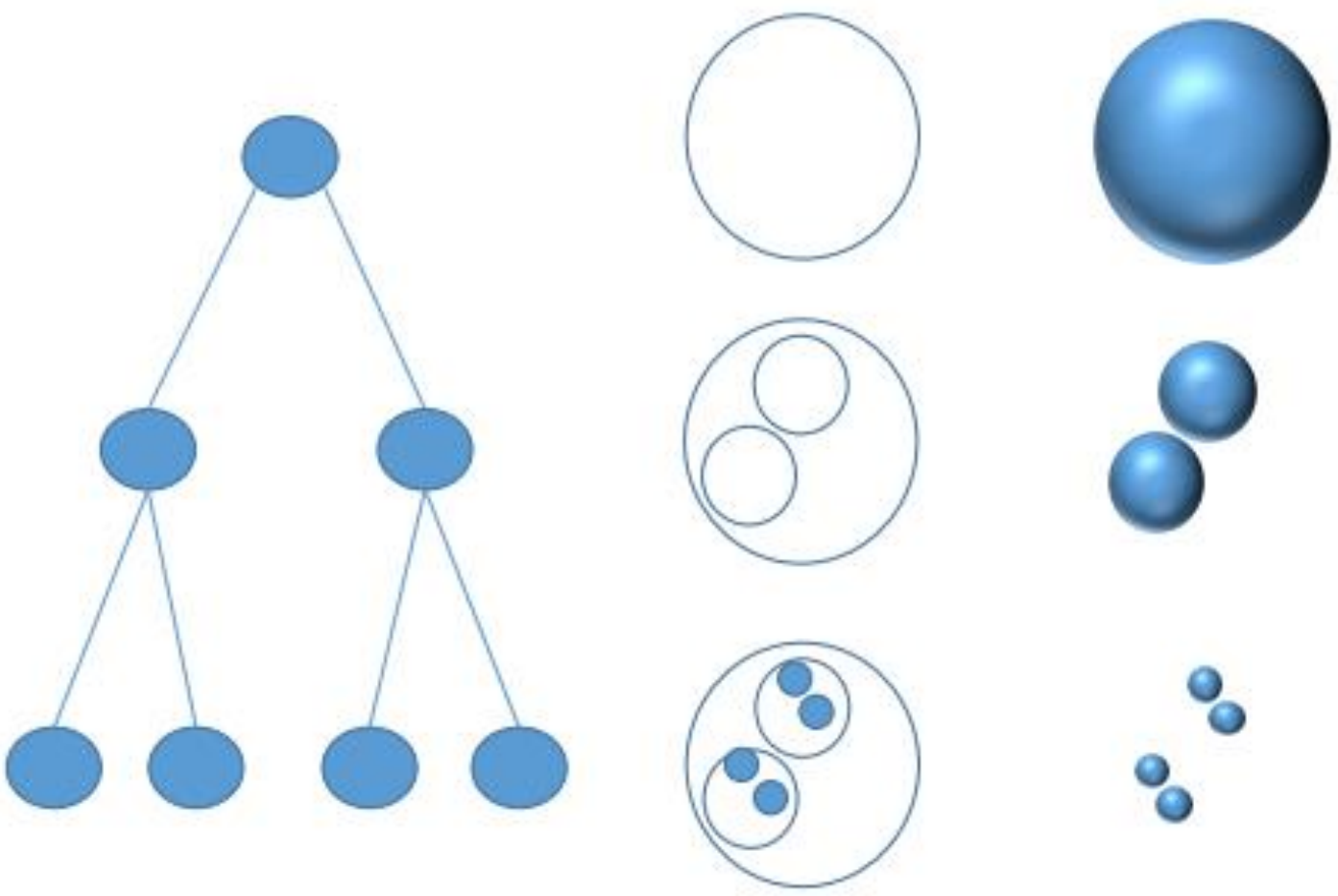




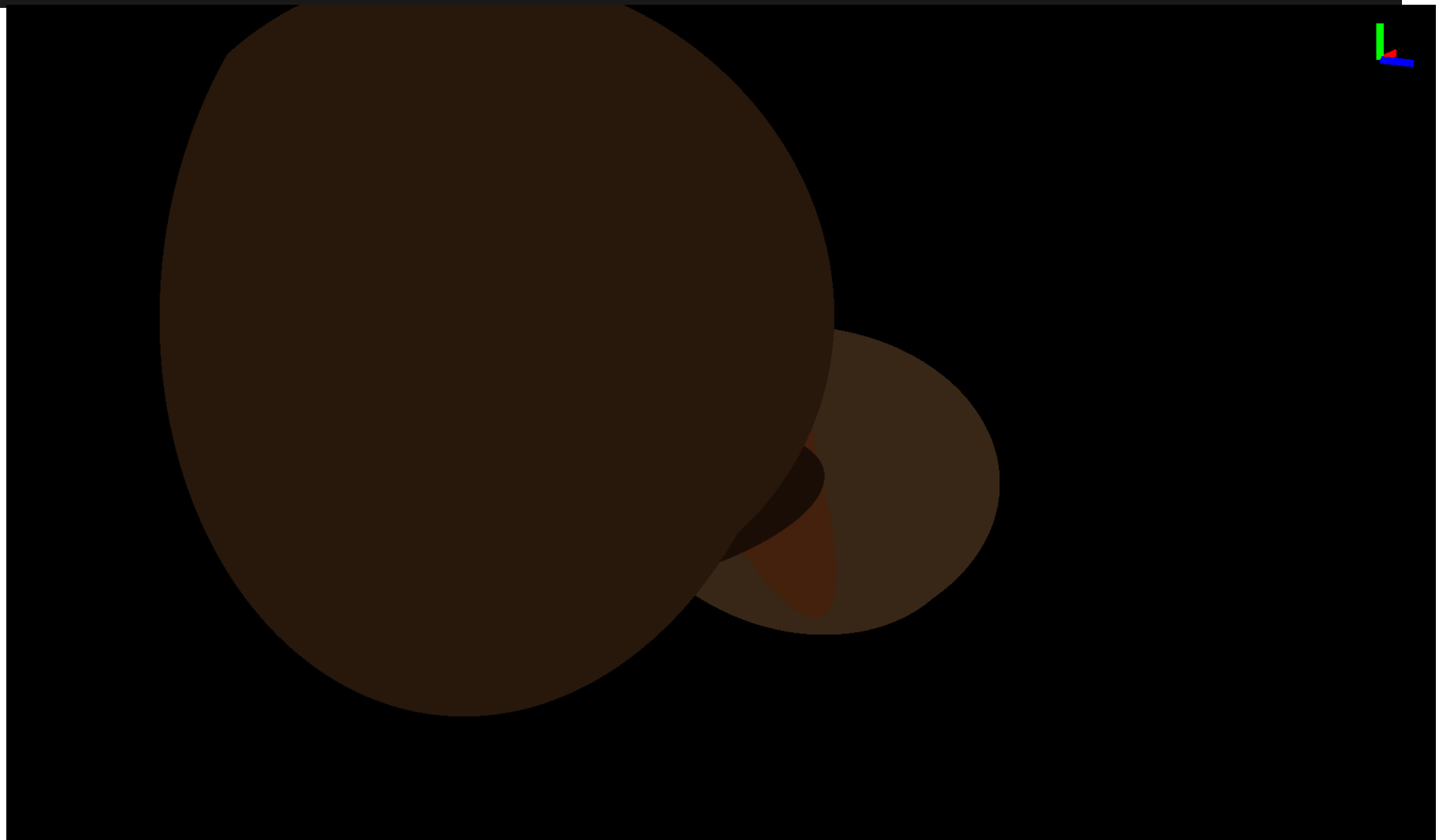
Program Structure



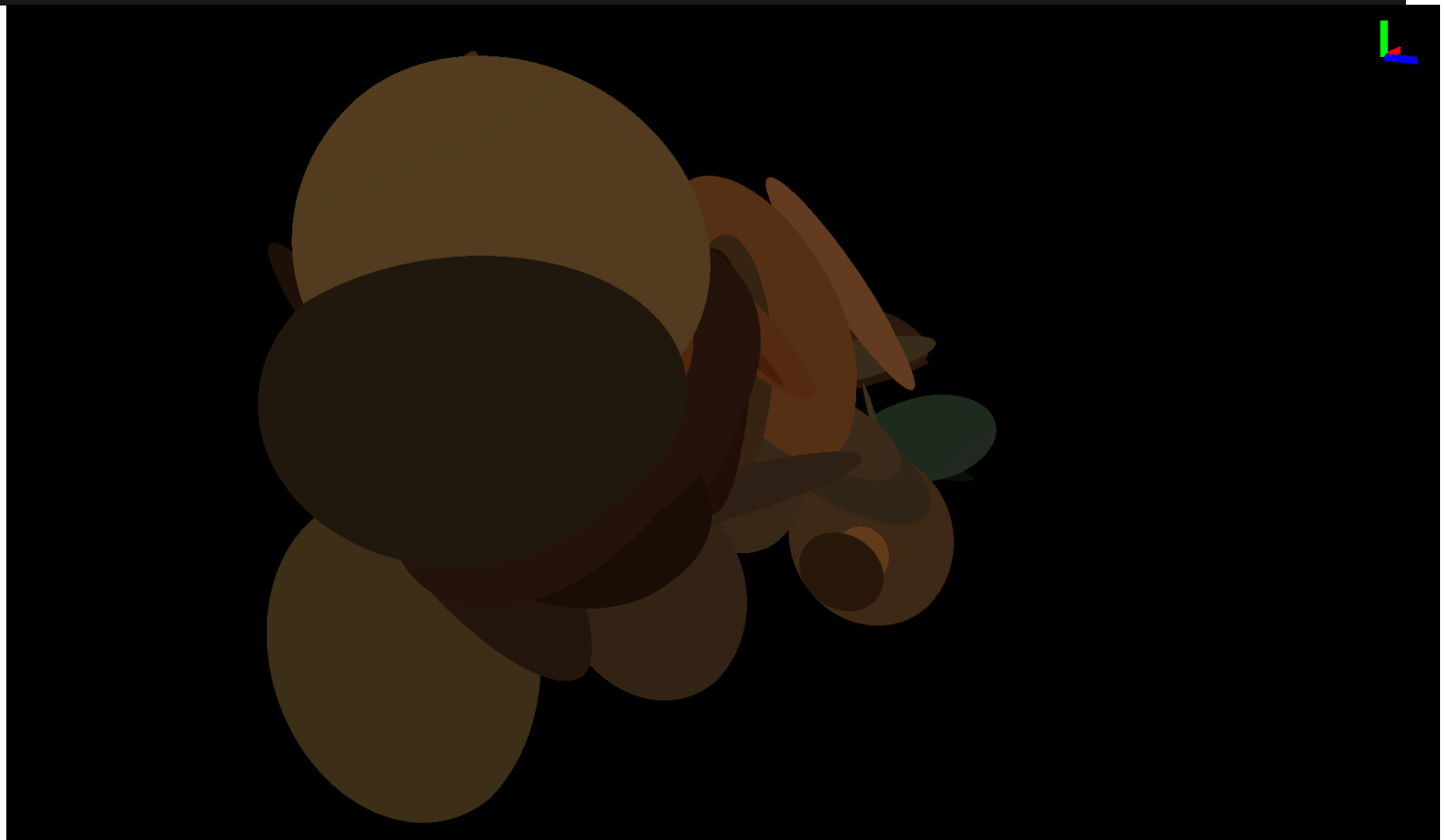
QSplat Tree Structure



Depth 0



Depth 1



Depth 2

Depth 4

Depth 8

Depth Max

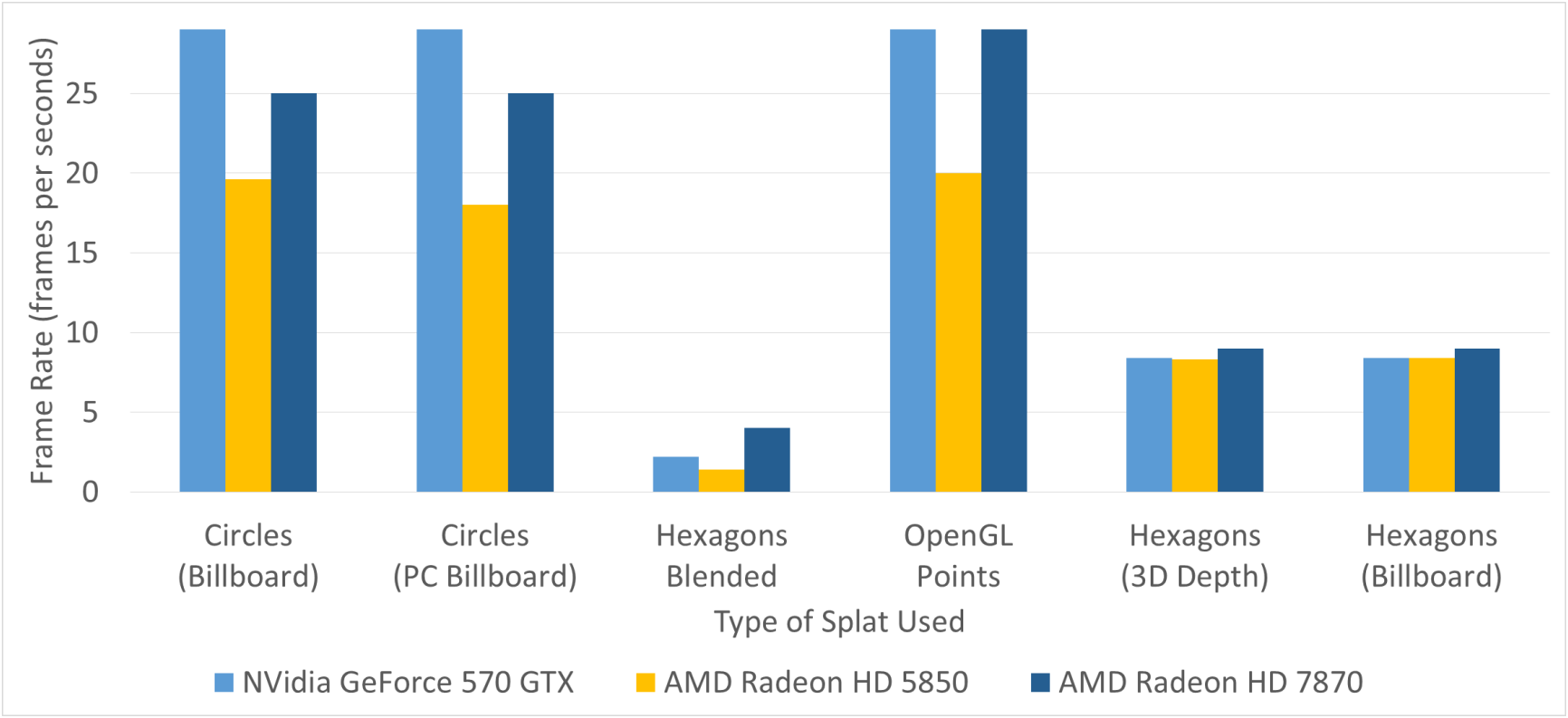
Moving Results

Depth Correct

Billboard Quad

Blended Hexagon

Frame Rate Comparison



Responsibilities

Member	Implementation	Research
Chris	Preprocessor	Octree
Eric	Rendering System	Ray Casting
Joel	Traversal	QSplat

Challenges

- Working with 13 year-old code
  - Problems
    - Uses more complex methods to do simple things because of hardware limitations that no longer exist.
    - Uses some functionality that have been replaced.
    - Incompatible number format.
  - Solutions
    - Wrapped their interface to fix inconsistency in data.

Challenges

- Working with a billion points
  - Problem
    - Large model files
  - Solution
    - Compress the file to a tree-based hierarchy, then use memory mapping and a top-down traversal to minimize thrashing.

Challenges

- Designing rendering system to work on varying hardware.
  - Problem
    - Not all hardware supports OpenGL functionality.
    - Supported Platform:
      - NVidia Quadro
      - NVidia Geforce
      - AMD Radeon
      - Intel Integrated GPUs
  - Solutions
    - Check compatibility of hardware.
    - Limit features used from newer OpenGL versions.

Questions?