Improving 3D Gaussian Splatting's Rasterization Performance

CSE Departmental Honors talk by

Kenneth J. Yang


Menu Views Capture
3D Gaussians
Camera Point view
Metrics
57.56 (17.37 ms)
VSync On

Goals of this talk

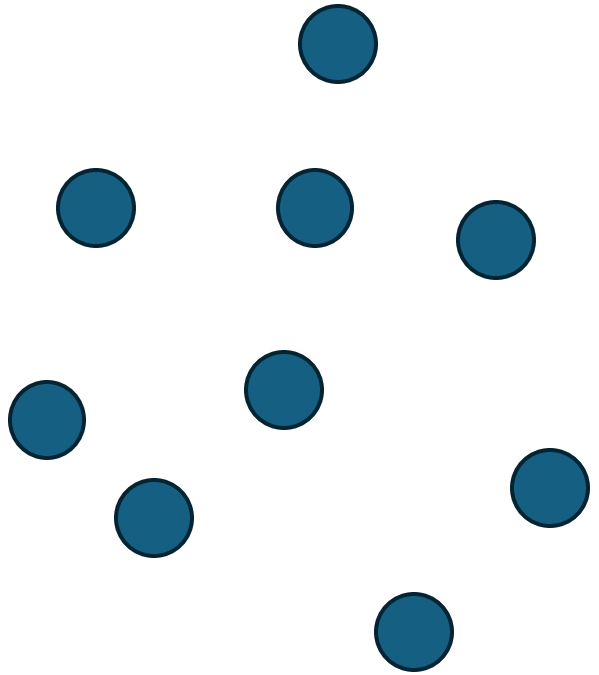
1. Review how 3DGS renders images
2. Appreciate the opportunity for improvement
3. Explain my solution

Outline

1. **How 3DGS renders an image**
2. Rendering transparencies
3. Empirical observations of scenes
4. Profiling the pipeline
5. Clustering
6. Streaming
7. Future

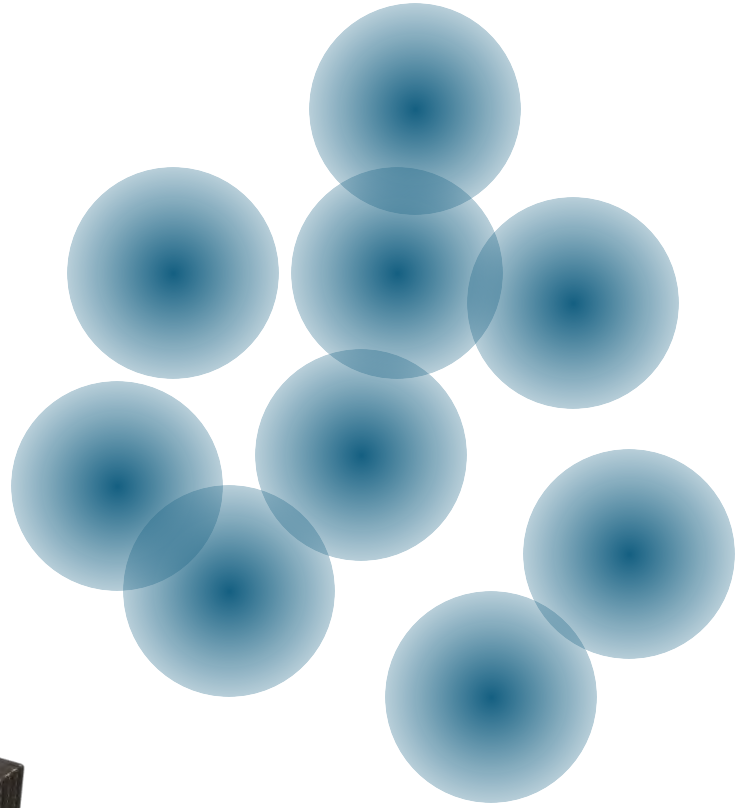
What’s Happening?

- Point cloud representation of the scene (SfM)



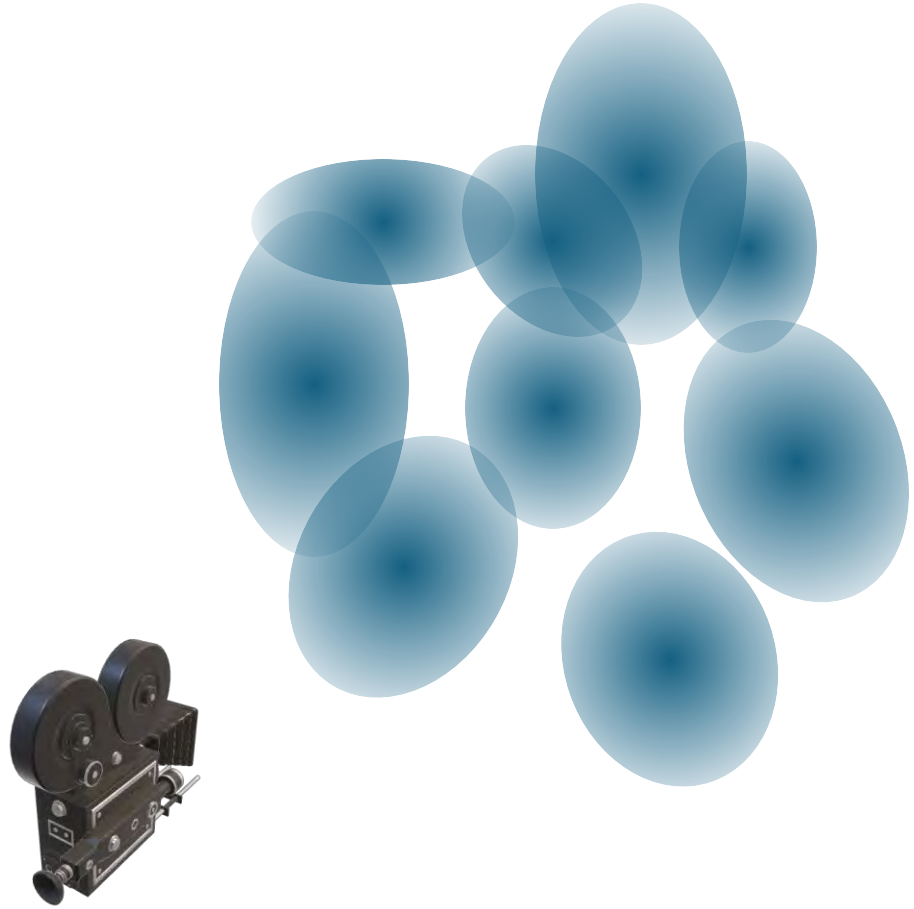
What’s Happening?

- Point cloud representation of the scene (SfM)
- Place 3D Gaussians where points are



What's Happening?

- Point cloud representation of the scene (SfM)
- Place 3D Gaussians where points are
- Morph Gaussians to better fit the scene



What's Happening?

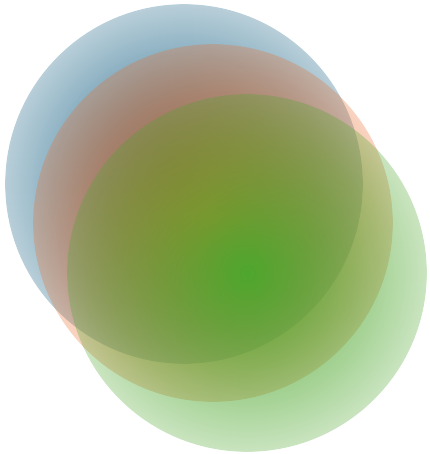
- Point cloud representation of the scene (SfM)
- Place 3D Gaussians where points are
- Morph Gaussians to better fit the scene
- 2D projection into "splats" and ***rasterize***

Outline

1. How 3DGS renders an image
2. **Rendering transparencies**
3. Empirical observations of scenes
4. Profiling the pipeline
5. Clustering
6. Streaming
7. Future

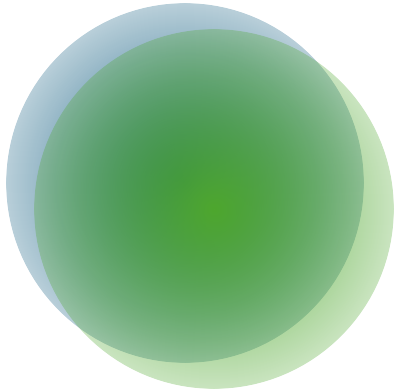
3 Methods

Alpha Over



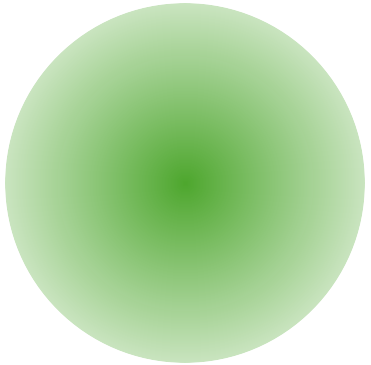
✅ Accurate
❌ Slow (**sorting**)

Stochastic



✅ Cheap
❌ Noisy (random)

Approximation



✅ Cheap
❌ Not Accurate

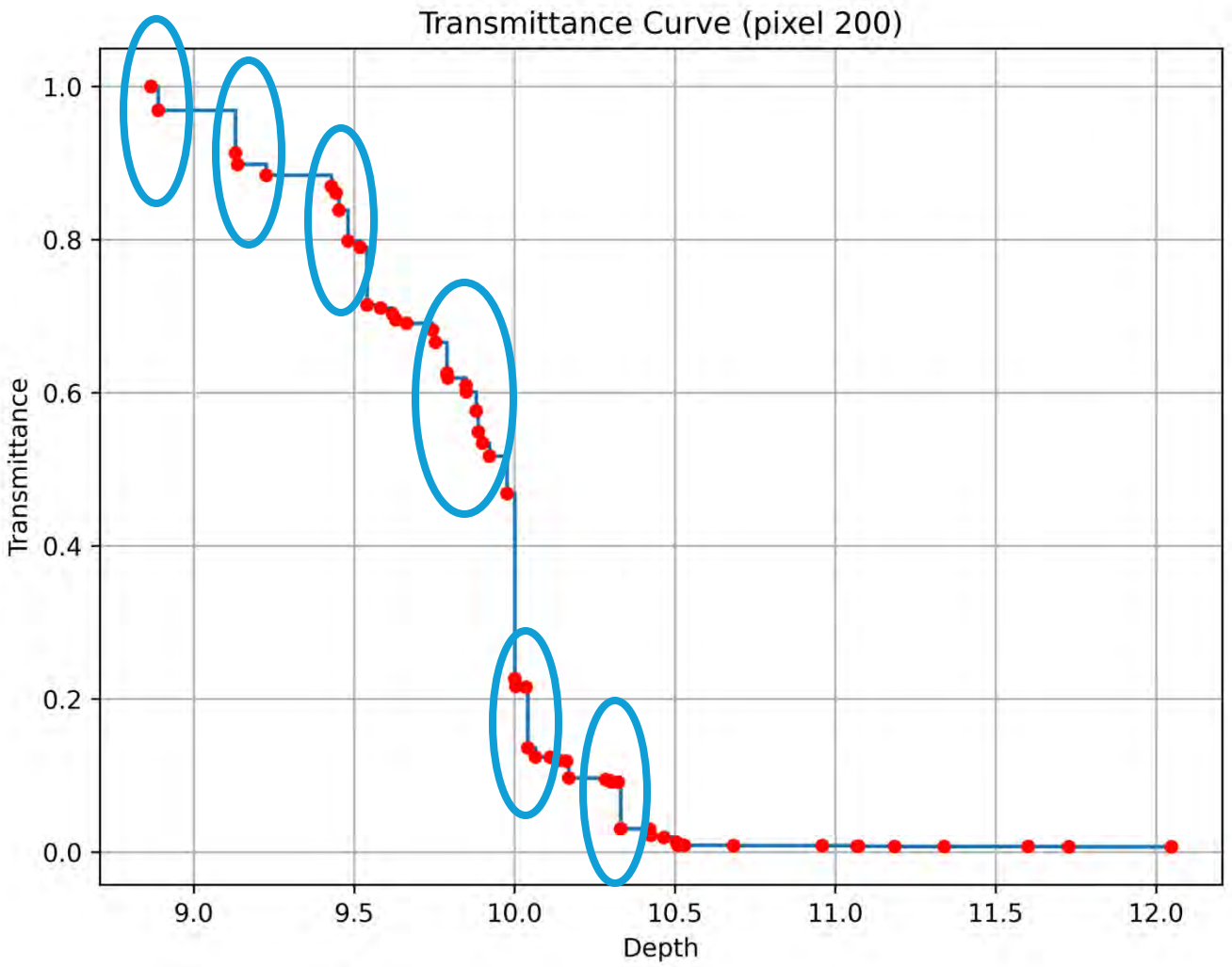
Sorting **thousands** of splats
is **slow**

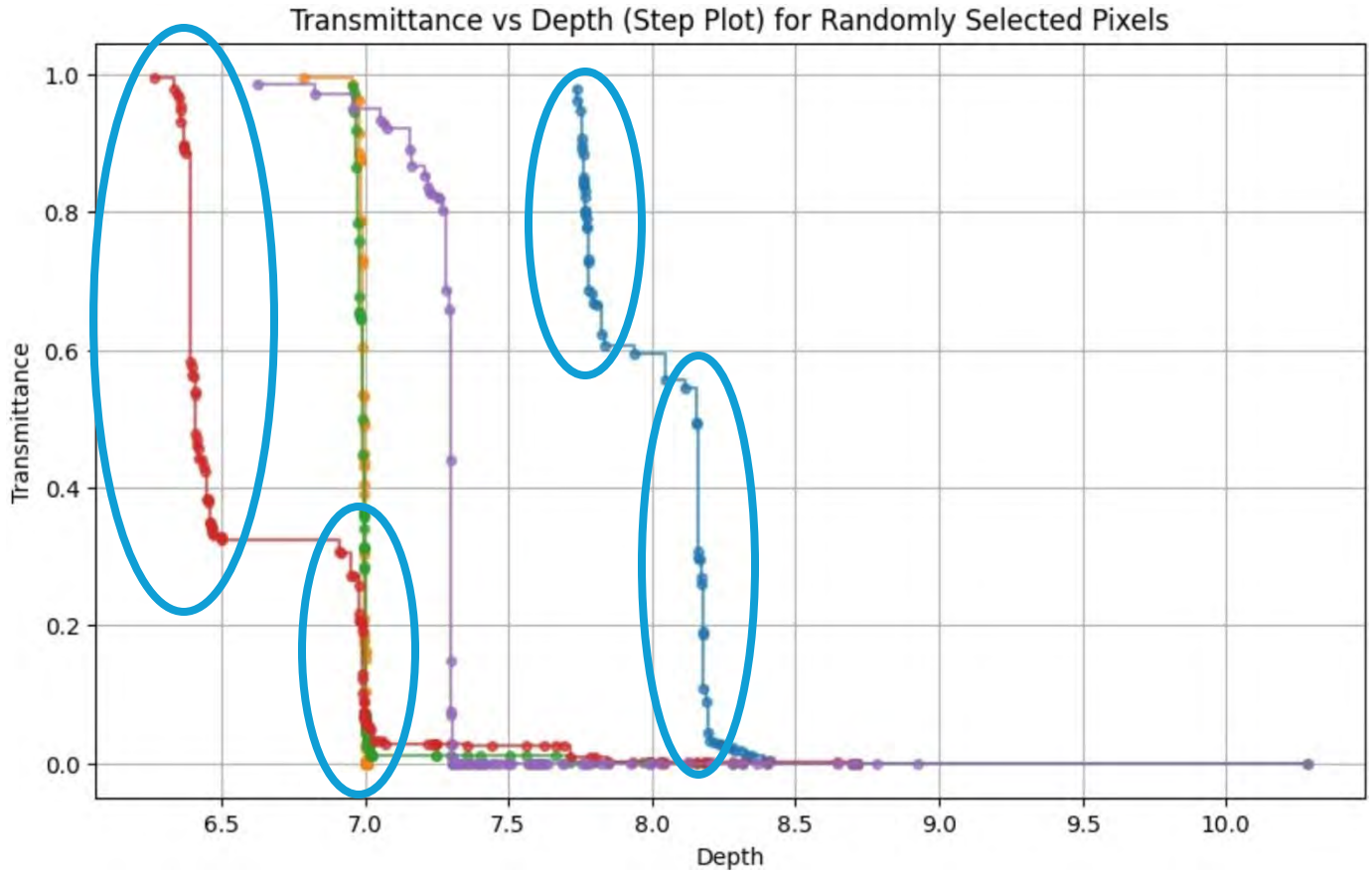
Outline

1. How 3DGS renders an image
2. Rendering transparencies
3. **Empirical observations of scenes**
4. Profiling the pipeline
5. Clustering
6. Streaming
7. Future

Alpha Over:

$$\alpha A + (1 - \alpha)B$$



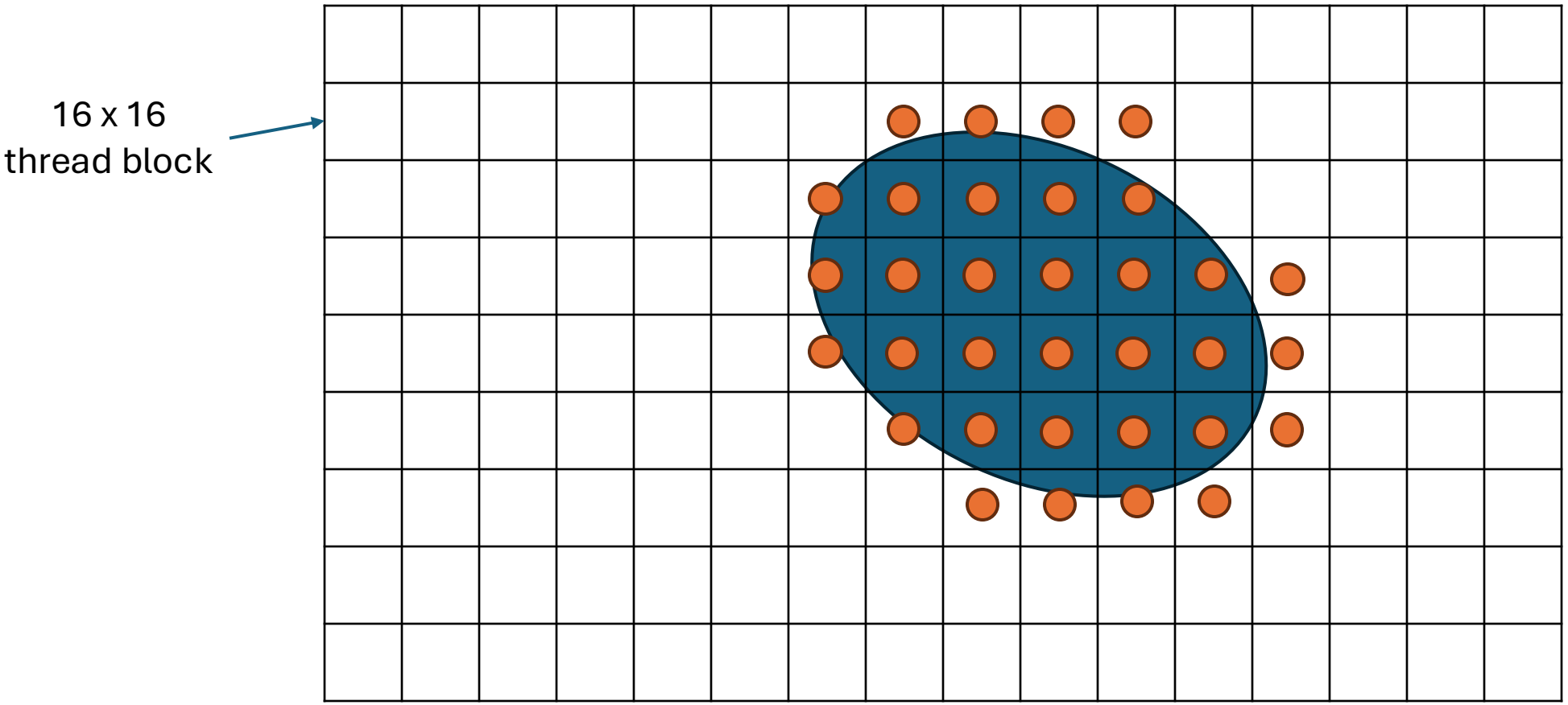

Transmittance vs Depth (Step Plot) for Randomly Selected Pixels
Transmittance
Depth
1.0
0.8
0.6
0.4
0.2
0.0
6.5
7.0
7.5
8.0
8.5
9.0
9.5
10.0

What if
we **cluster** splats
to
sort less?

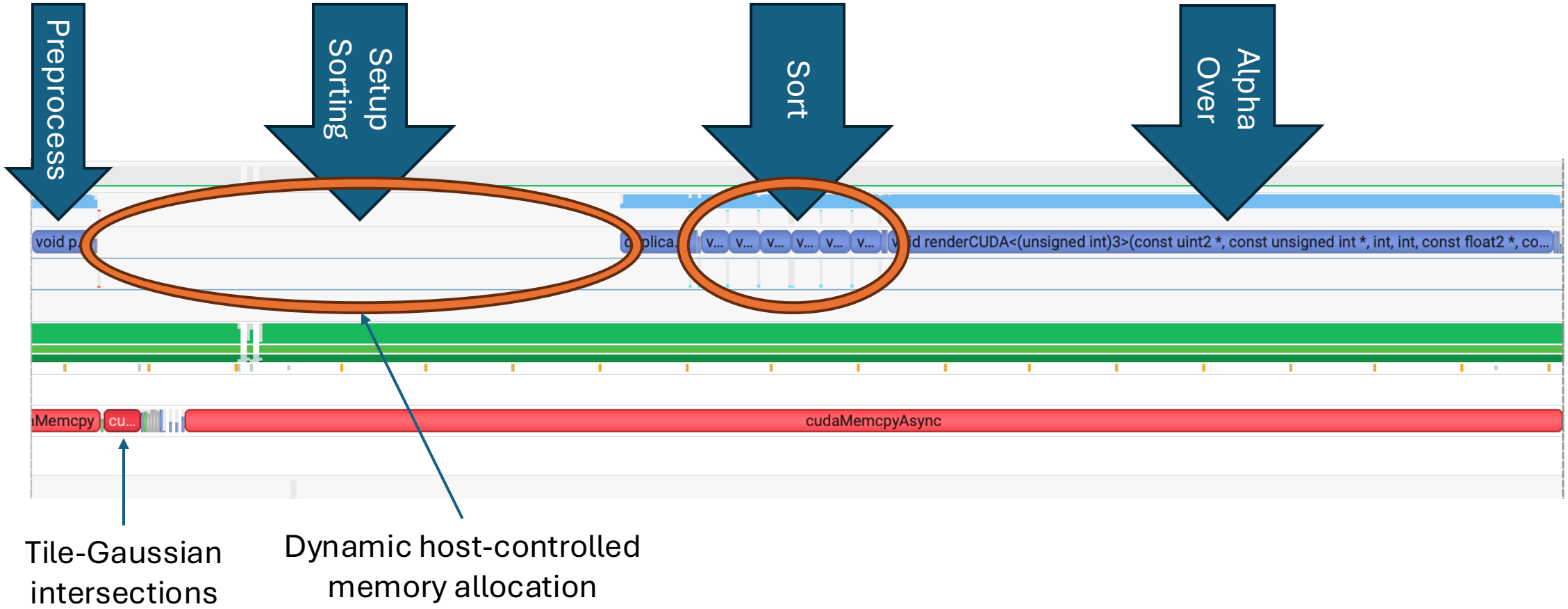
Outline

1. How 3DGS renders an image
2. Rendering transparencies
3. Empirical observations of scenes
4. **Profiling the pipeline**
5. Clustering
6. Streaming
7. Future

How is an Image Split on the GPU?



What's the Pipeline Doing? (Nsight System)



Outline

1. How 3DGS renders an image
2. Rendering transparencies
3. Empirical observations of scenes
4. Profiling the pipeline
5. **Clustering**
6. Streaming
7. Future

Ground Truth

1/4: Equal-width binning

1/4: Equal-width binning

Divide the depth range into equal bins

- Pros
  - Easy to compute
- Cons
  - 🤮
  - Requires knowing depth range (can’t stream)

1/4: Equal-width binning

2/4: Epsilon-delta

2/4: Epsilon-delta

Cluster anything epsilon distance from a common point

- Pros
  - Better than Equal-width
  - Easy to compute and can stream
- Cons
  - Sensitive to what the “common point” is

2/4: Epsilon-delta

3/4: k-Means

3/4: k-Means

Classical k-Means

- Pros
  - Polynomial
  - Consistent between runs
- Cons
  - Requires searching through all data (can't stream)
  - Computationally complex (compared to prior examples)

3/4: k-Means

4/4: Sequential k-Means

4/4: Sequential k-Means

k-Means but with data coming in sequentially

- Pros
  - 1-pass k-Means
  - Consistent between runs
- Cons
  - Still computationally complex

Sequential k-Means

let's us **cluster** splats

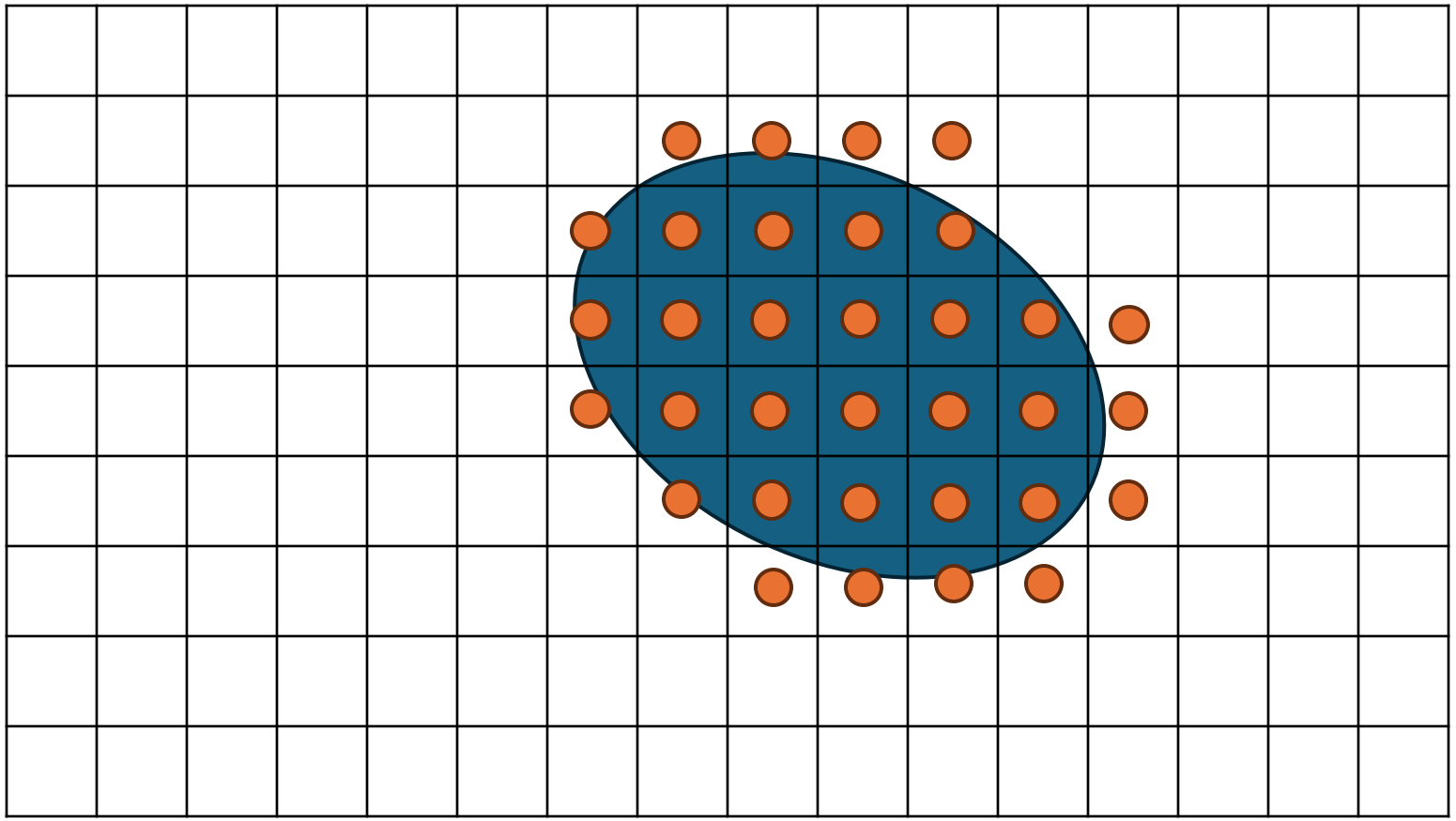
and only

sort k fragments

Outline

1. How 3DGS renders an image
2. Rendering transparencies
3. Empirical observations of scenes
4. Profiling the pipeline
5. Clustering
6. **Streaming**
7. Future

A Better Algorithm



A Better Algorithm

For each tile (computed by a thread block)

1. Go through all splats and check if it intersects with this tile
2. If it intersects a tile, cluster it per pixel using SKM
3. After all splats have been checked, alpha-over render the clusters.

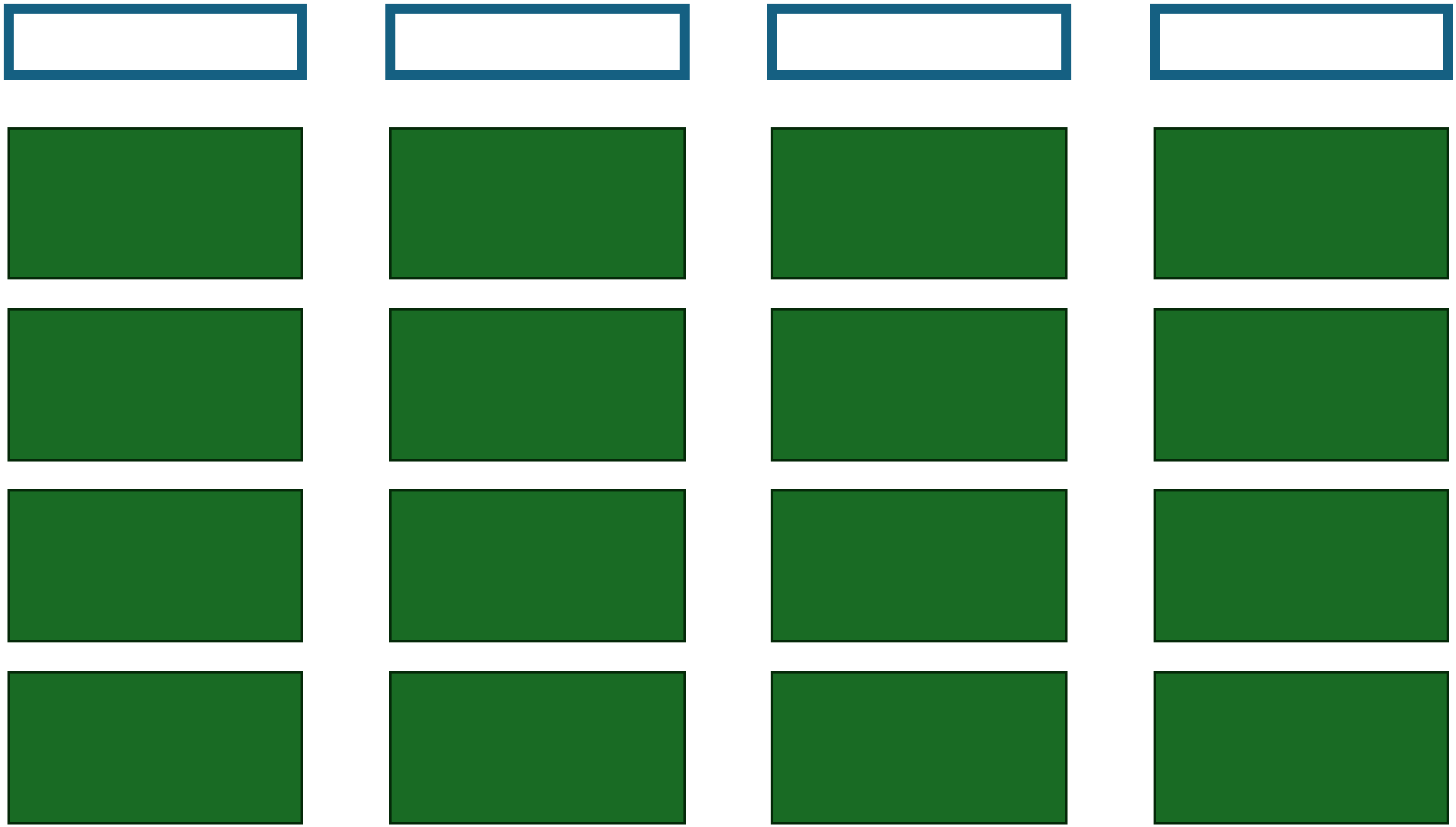
A Better Algorithm

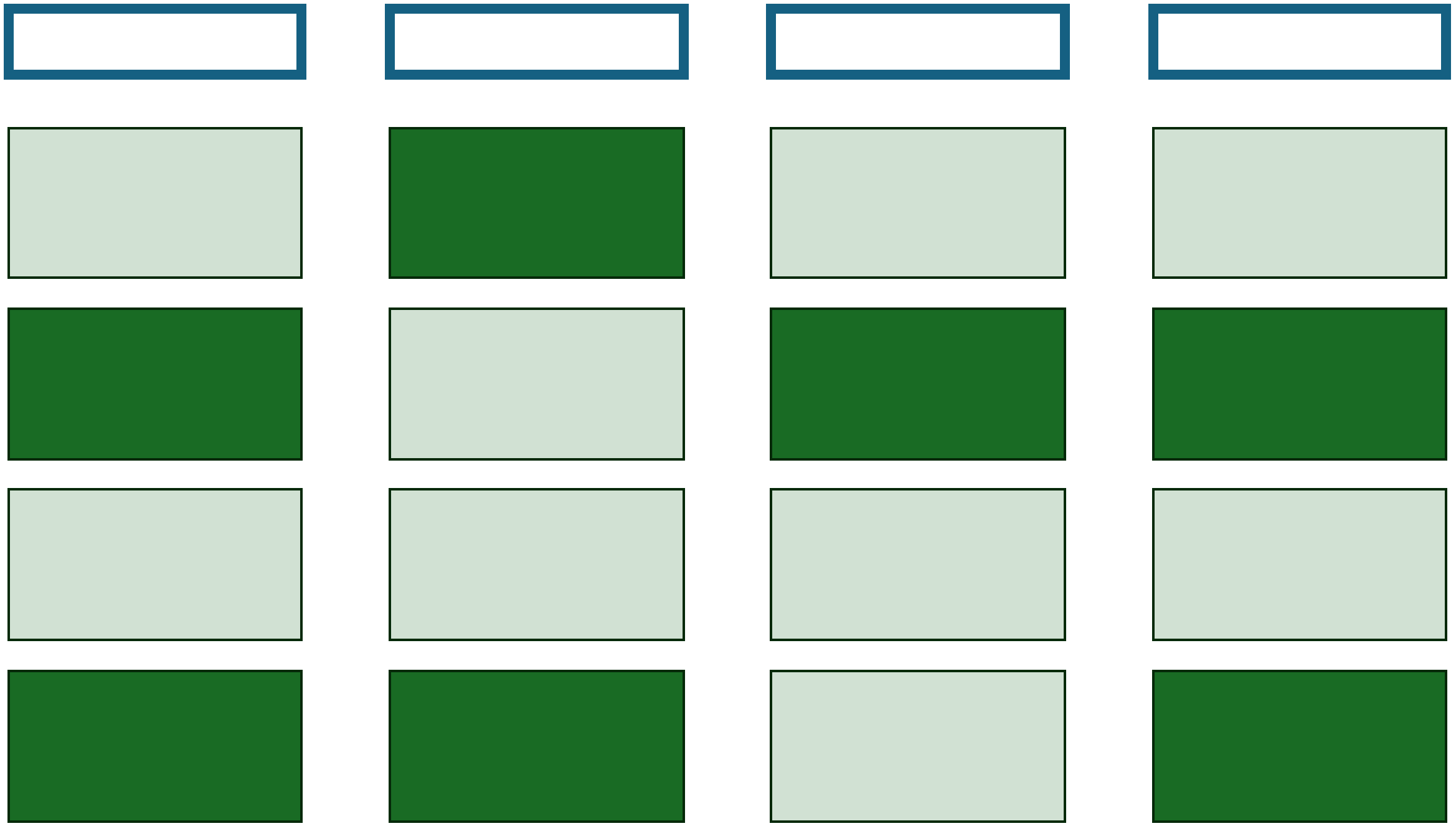
Pros

- No CPU-GPU round-trip dynamic allocation
- No sorting
- No data duplication and extra storage needed
- It’s just one kernel launch

Cons

- Does not take advantage of tile-intersection culling
- Current implementation has potentially poor memory cache utilization
- More computationally complex (thread divergence) than basic alpha-over






Preprocess
Sadness
+700ms
+750ms
+800ms
+850ms
+900ms
+950ms
+50ms
+100ms
+150ms
+200ms
+250ms
void skm_renderCUDA<(unsigned int)3>(int, int, int, dim3, int, int *, const float2 *, const float4 *, const float *, float *, const float *, float *, unsigned int *, const float *, float *)
cudaMemcpyAsync

Outline

1. How 3DGS renders an image
2. Rendering transparencies
3. Empirical observations of scenes
4. Profiling the pipeline
5. Clustering
6. Streaming
7. **Future**

Next steps

1. Better kernel engineering (caching and divergence)
2. Update training to tune for clustering
3. Alignment with hardware rasterization
4. Apply techniques on newer iterations of 3DGS

Thanks!

Questions?