

Splatshop:

Efficiently Editing Large Gaussian Splat Models

Markus Schütz¹, Christoph Peters², Florian Hahlbohm³,
Elmar Eisemann², Marcus Magnor³, Michael Wimmer¹

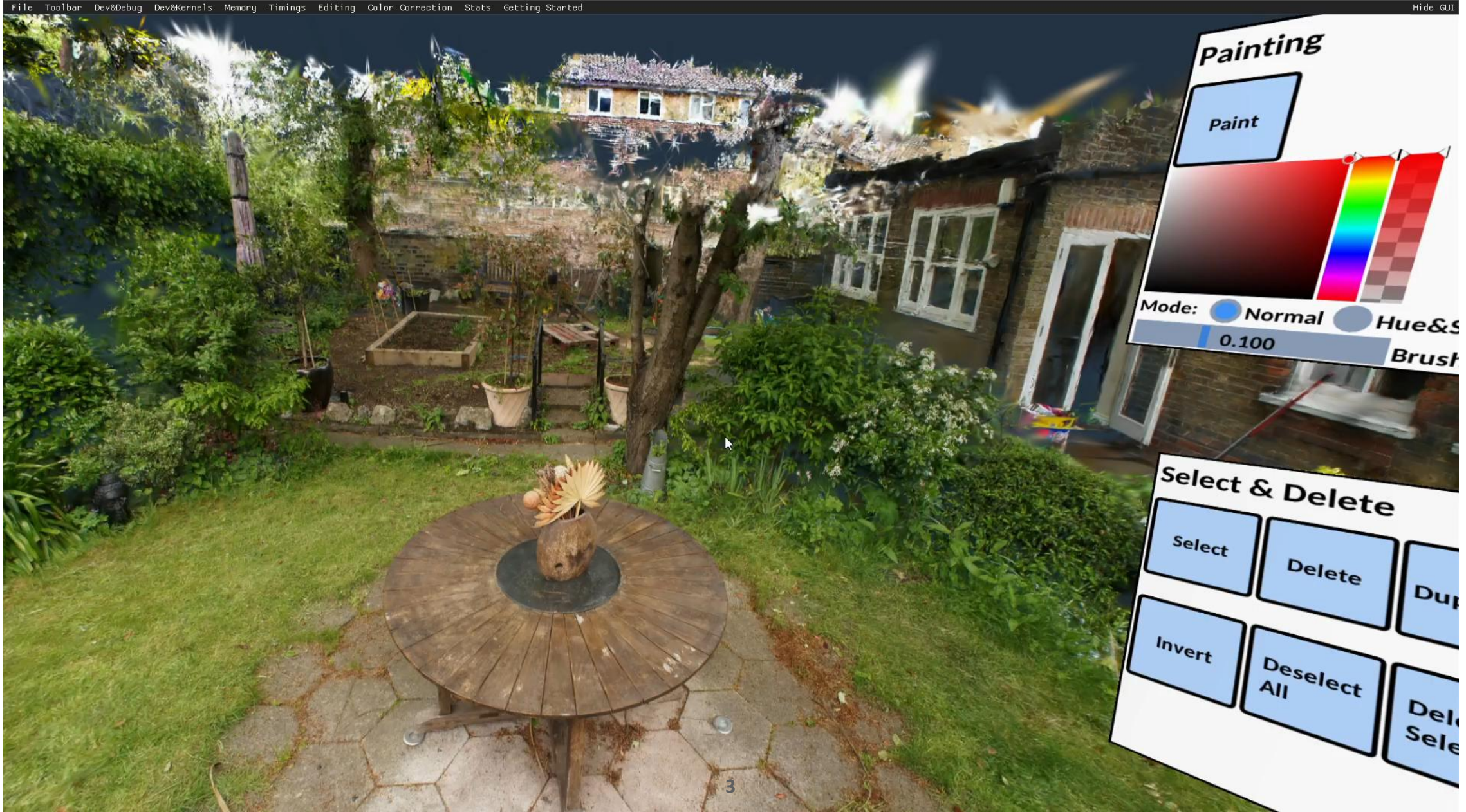
¹TU Wien, ²TU Delft, ³TU Braunschweig



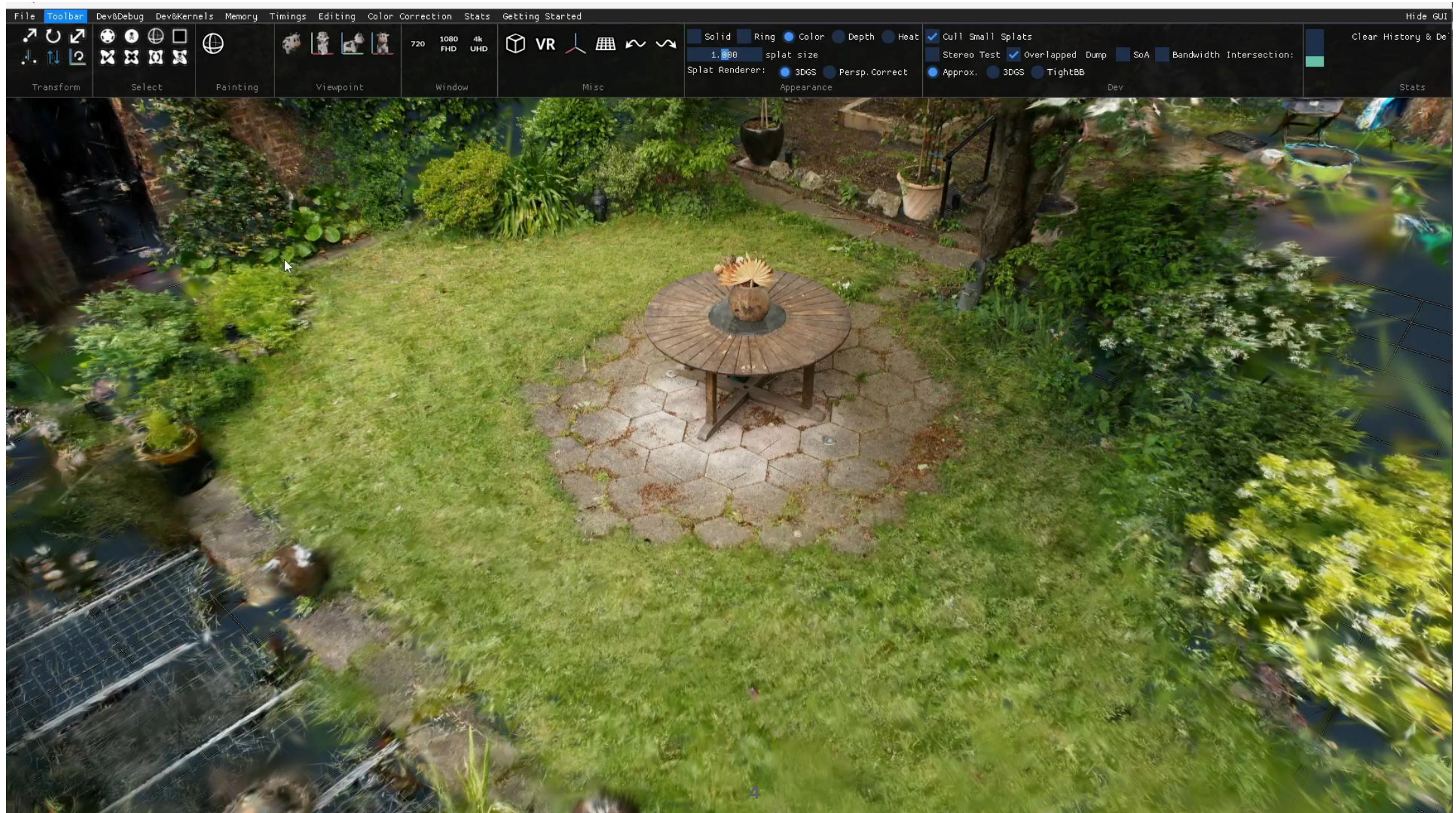
- Efficiently editing up to 100M Splats
- VR Support (90fps up to ~10M Splats)



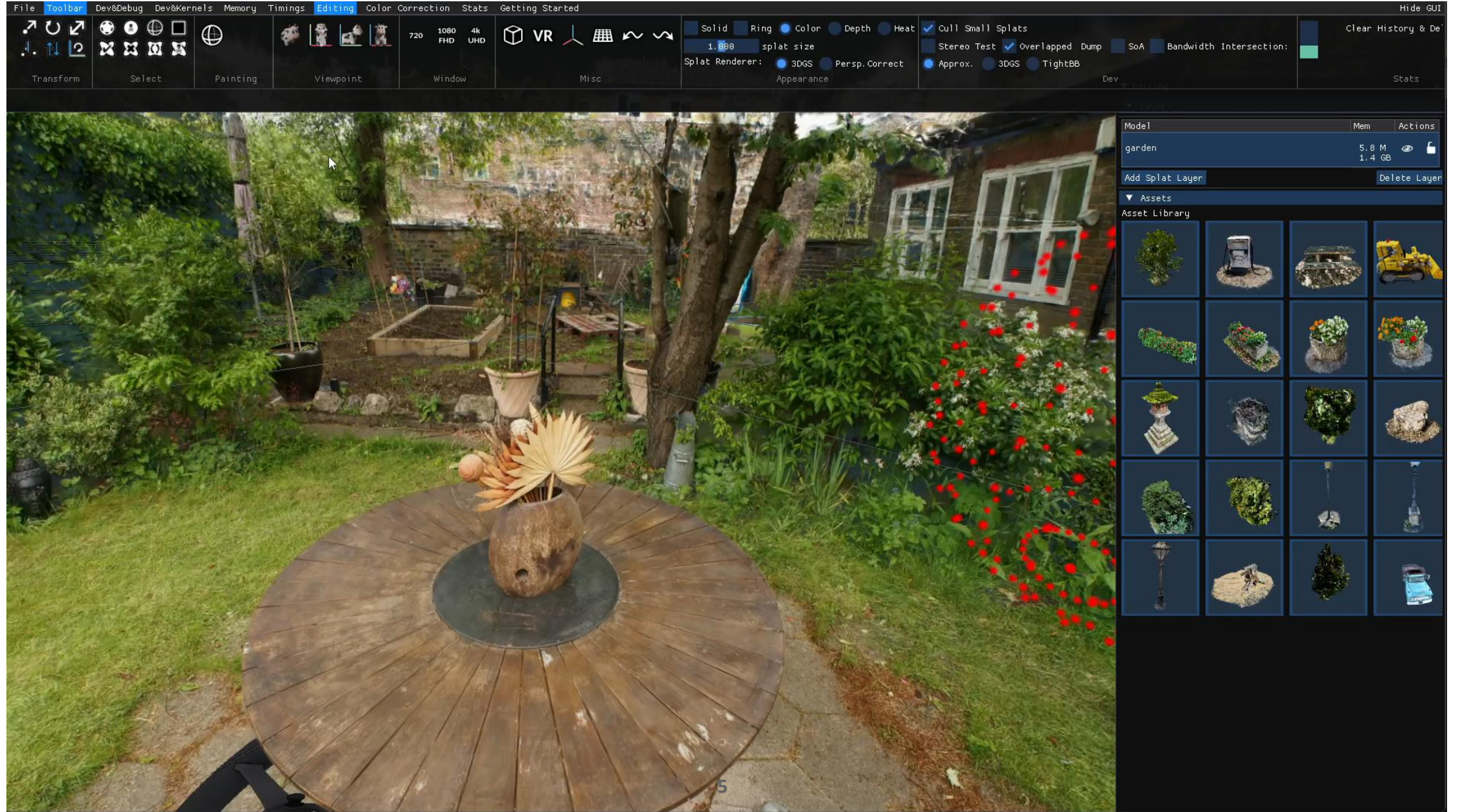
Splatshop



Splatshop



Splatshop: Transformation



Splatshop

- Painting



- Hue-Brushing



Undo / Redo



- The most important feature: Act without consequences
- Selection: 1 bit per splat to indicate change
- Painting: 12 byte per splat, storing index and original color of splats
- Transformation: Storing transformation matrix (lossy undo)



- At start of stroke: duplicate color buffer
- At end of stroke: Compare original to modified
- Create compact diff
 - 4 byte index + 8 byte original color value

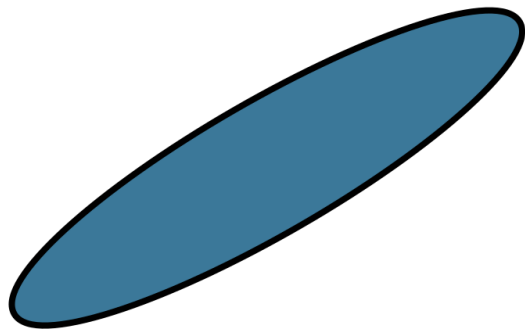
	Painting	compaction	Undo/Redo
Garden (5.6M)	0.1 – 0.3 ms	0.5 – 2.6 ms	0.3 ms
Campus (28.6M)	0.5 – 1.7 ms	1.6 – 9.2 ms	2.3 ms



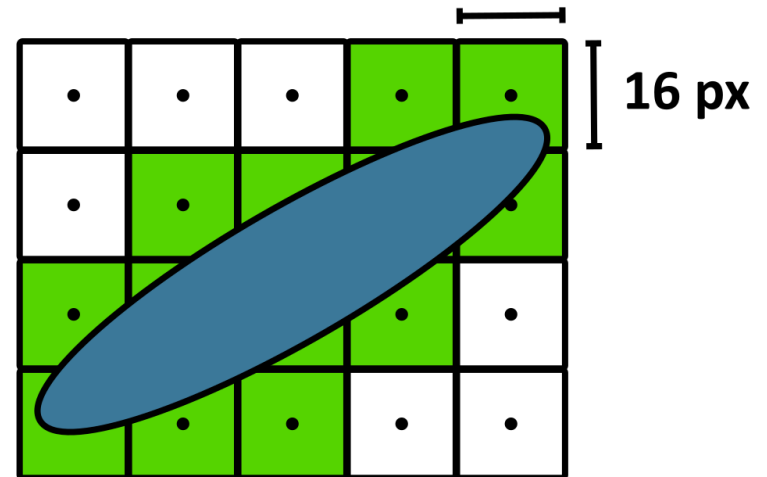
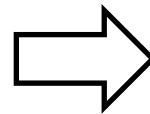
Improving Sorting



- Splats are sorted by depth and by TileID
- 48 bit sort:
 - 32 bit depth
 - 16 bit TileID



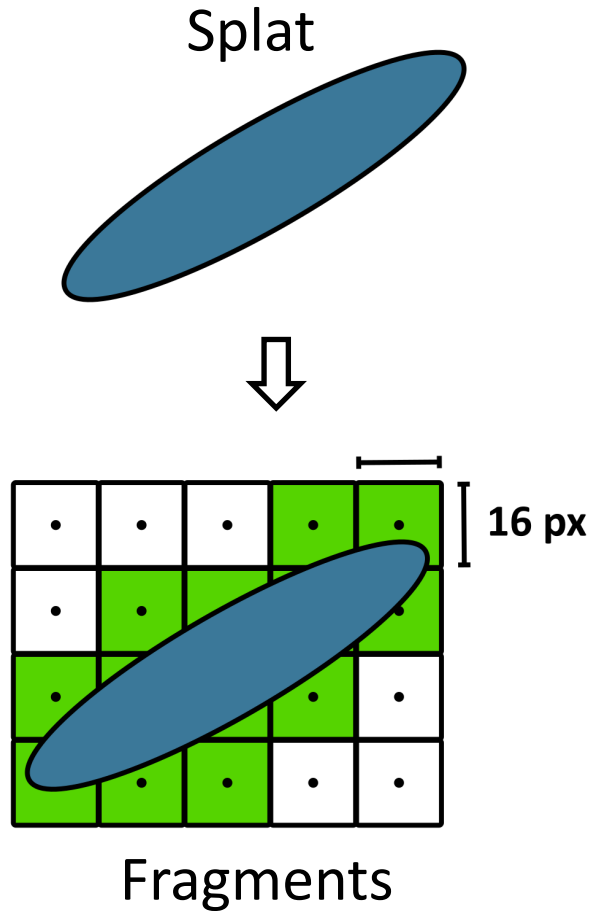
Splat



Fragments / Duplicates



Improving Sorting



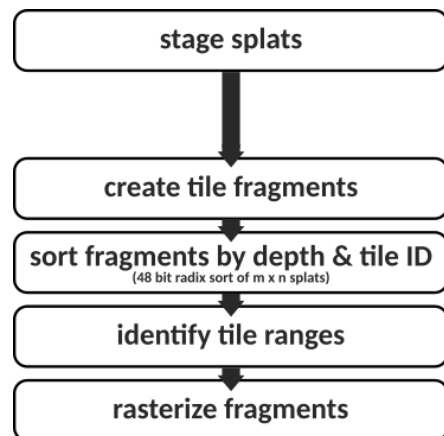
Visible splats: **1.7M**

Visible tile-wise fragments: **4.3M**

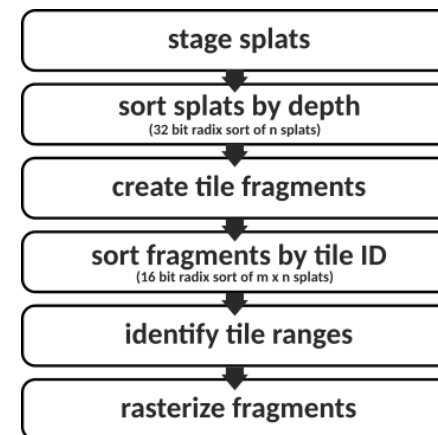


Improving Sorting

- Improvement: Split sorting
 - **Sort 1.7 million visible splats by depth**
 - Create depth-sorted duplicates / fragments
 - **Sort 4.3 million fragments by TileID**



3DGS Pipeline



Splatshop Pipeline



Improving Sorting

- Works because radix sort is typically stable
 - Fragments with equal TileID remain sorted by depth
- Pays off because depth-sort is more expensive (32 bit)
 - Do expensive sort on fewer elements
- „expensiveness“:
 - 4.3 M frags * 48 bit = **206.4**
 - 1.7 M splats * 32 bit + 4.3 frags * 16 bit = **123.2**



- Performance after splitting sort in two separate passes

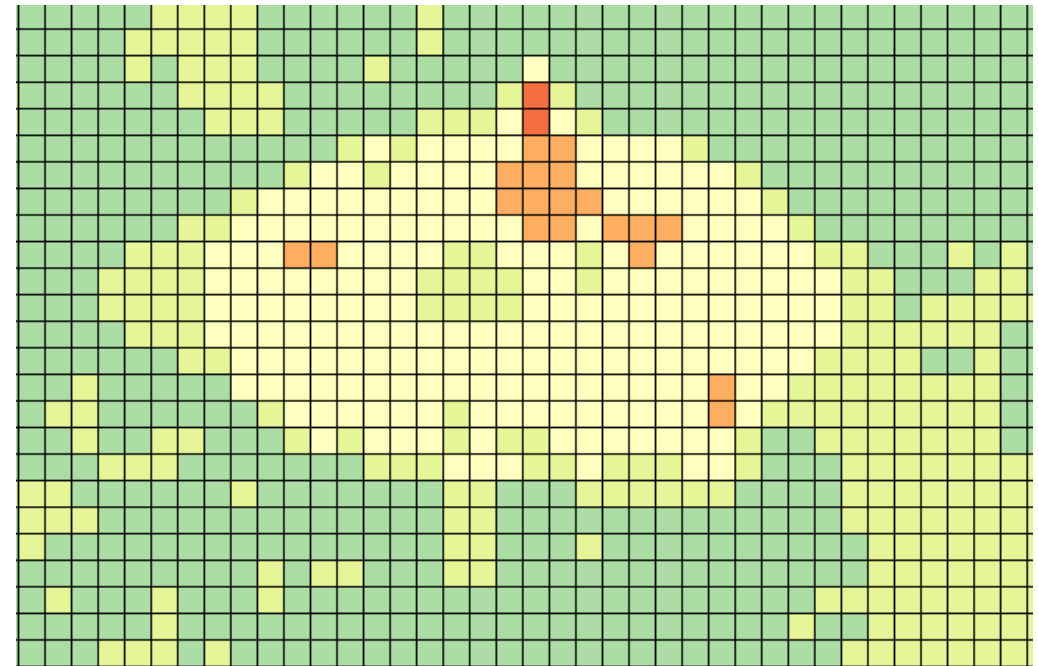
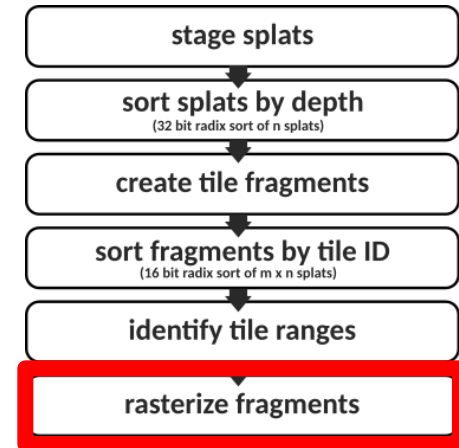
	48 bit	32 bit + 16 bit
Garden	0.48 ms	0.31 ms
Campus (close)	1.86 ms	1.00 ms
Campus (far)	1.23 ms	0.86 ms



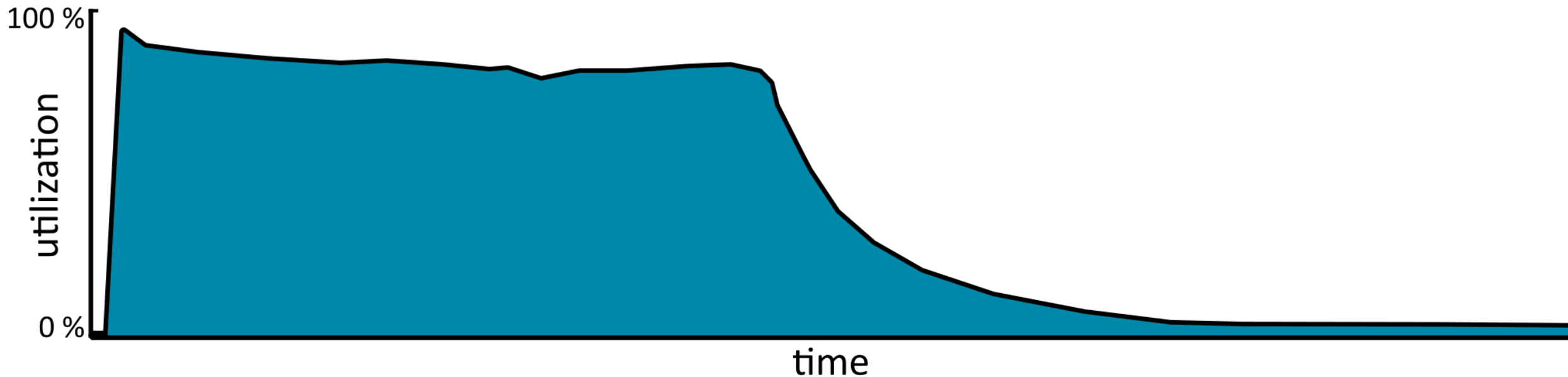
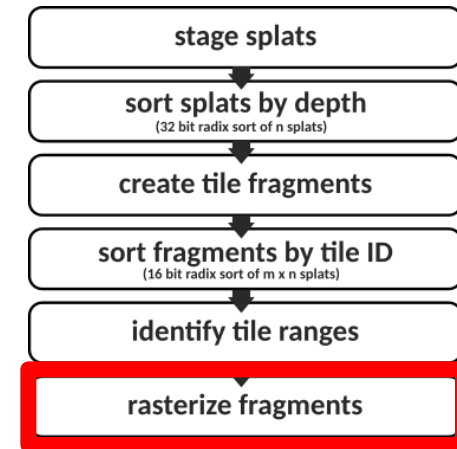
Imbalanced Workloads



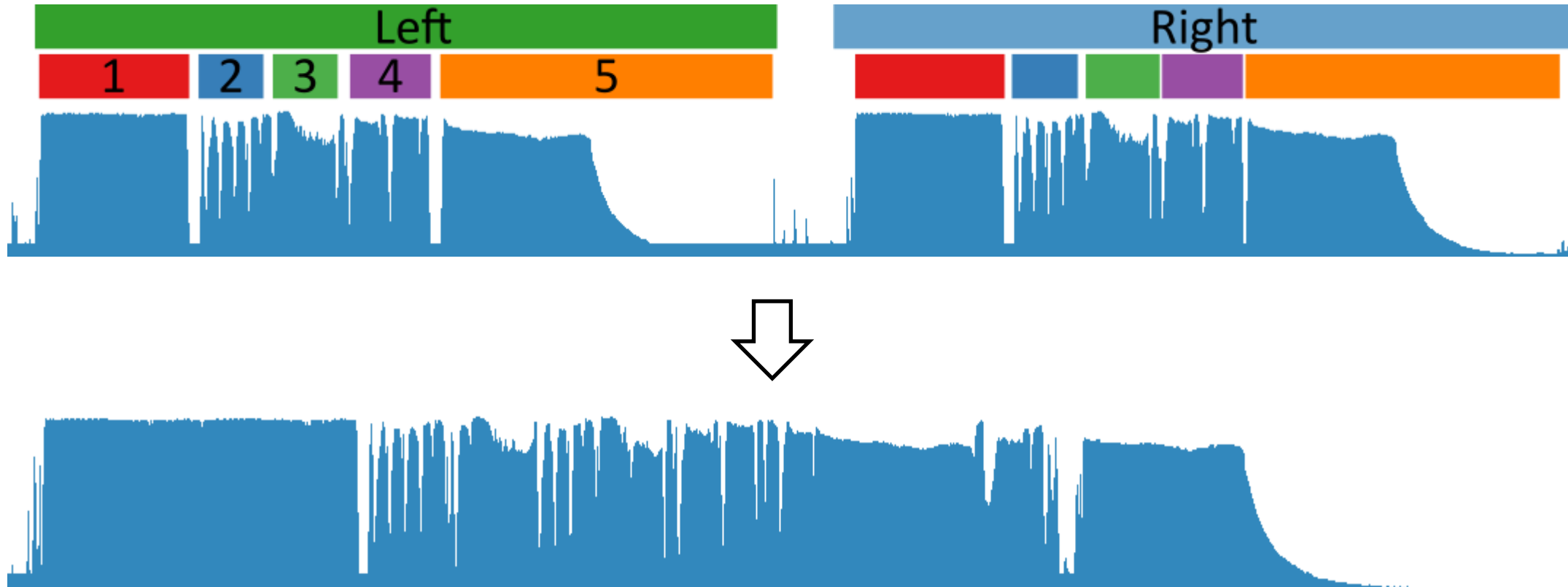
- Last stage in splat rasterization is imbalanced
 - Tiles with exceptionally many splats
 - One thread-block renders one tile
 - GPU waits until kernel's slowest thread-block finishes



- Last stage in splat rasterization is imbalanced
 - Tiles with exceptionally many splats
 - One thread-block renders one tile
 - GPU waits until kernel's slowest thread-block finishes



- In VR: Concurrent streams to improve utilization

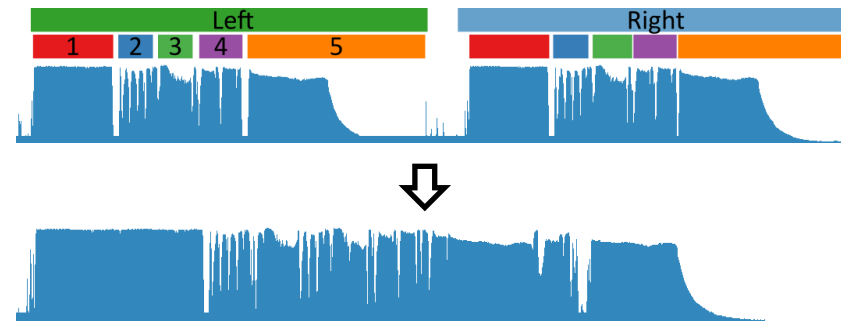


- Concurrent streams may require duplicate memory
 - Sorting buffers
 - Staging buffers
 - ...
- Hence we still do sorting for left and right eye one after the other
 - Utilization for sorting is good anyway



■ Concurrent Stream Performance in VR

	Single Stream	Concurrent Streams
Garden	8.1 ms	6.5 ms
Campus	16.2 ms	13.8 ms



Progressive Spherical Harmonics



- Evaluating SHs is expensive
- Progressive SHs
 - Evaluate $1/10^{\text{th}}$ of SHs each frame
 - Bake results into uint8 colors buffer

■ Performance

	All SHs each frame	$1/10^{\text{th}}$ per frame	$1/30^{\text{th}}$ per frame
Garden	2.6 ms	1.9 ms	1.8 ms
Campus	4.4 ms	1.5 ms	1.1 ms



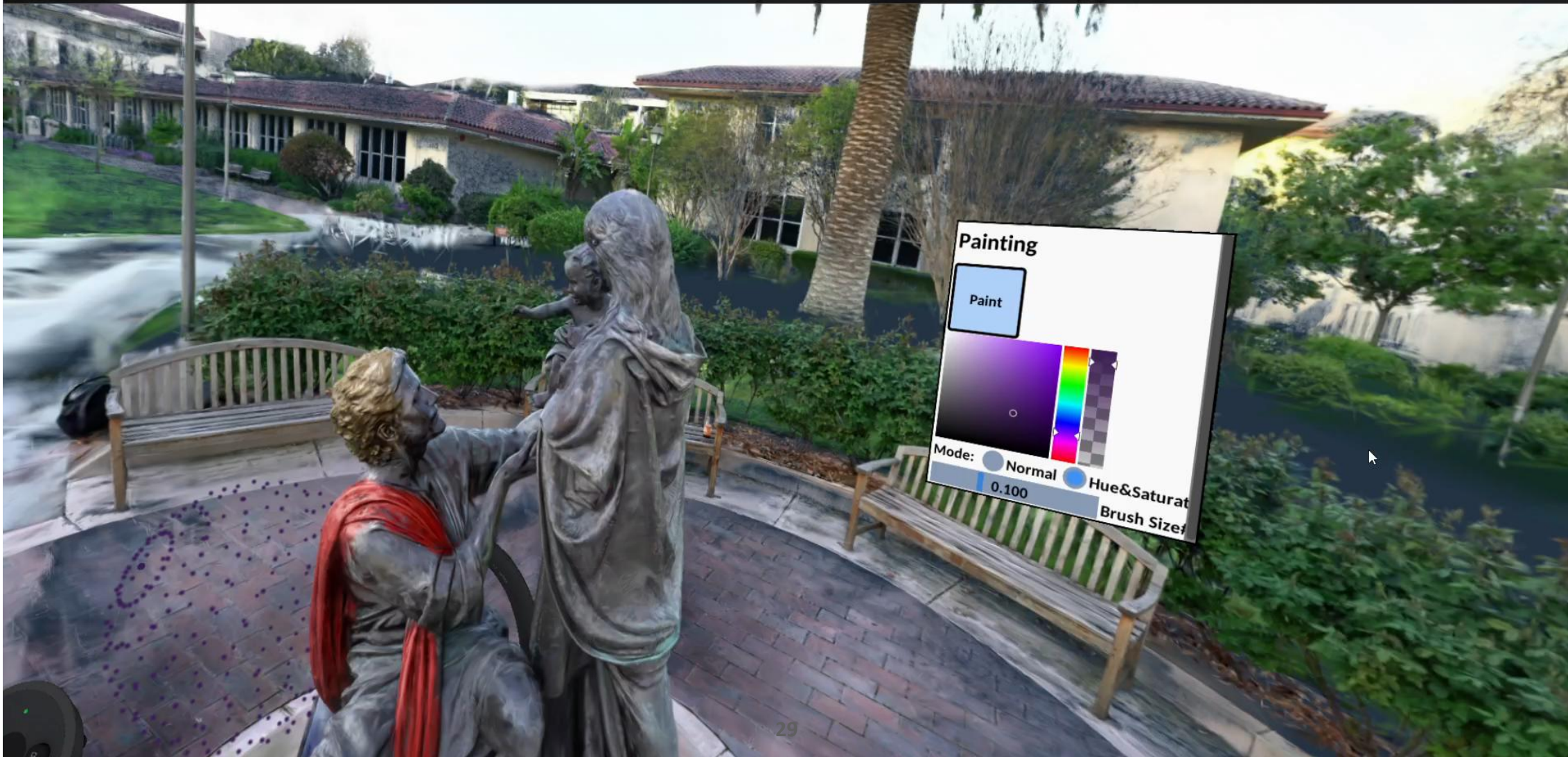
- Currently all SHs updated during transform → expensive
 - Progressive SHs during transformation
- Undo/Redo of transform currently lossy
 - Compress original values?
 - Inverse + delta coding?
- Painting sparse geometry → Aliasing



File **Toolbar** Dev&Debug Dev&Kernels Memory Timings Editing Color Correction Stats Getting Started Hide GUI

Solid Ring Color Depth Heat Cull Small Splats
 1,000 splat size Stereo Test Overlapped Dump SoA Bandwidth Intersection: Clear History & De
 Splat Renderer: 3DGS Persp. Correct Approx. 3DGS TightBB Dev Stats

Transform Select Painting Viewpoint Window Misc Appearance



Thank you for your attention

<https://github.com/m-schuetz/Splatshop>

