S7698:
CanvoX: High-Resolution VR Painting for Large Volumetric Canvas

Yeojin Kim¹, Byungmoon Kim², Jiyang Kim¹ and Young J. Kim¹
Ewha Womans University¹, Adobe Research²

http://graphics.ewha.ac.kr/canvox/
VR Painting?

Quill

Tilt Brush by Google

PaintLab VR
Fundamental Questions in VR Painting

• Can we recolor, erase, and mix the color?

• Can we draw and mix transparent object?

• How much can we extend canvas?

• How detail can we draw?

• How can we navigate 3D canvas?
Voxels?
Challenges

• Large Canvas with high detail
  • Deep Level Octree
    • Expensive refinement and coarsening
  • Dynamic Tree on GPU
    • Random access → need complex data structure
    • CPU-GPU transfer cost

• Rendering
  • Real-time ray casting (resolution $1680 \times 1512 \times 2$, 90fps~)
    • Tree traversal time
  • Accumulated error along the ray
CanvoX Model

HMD
- View
- Position

Controller /Haptic Device
- Brush
- Position

CPU

Octree
- Parent ID
- Child ID
- flag
- RGBA
- Temp0

Stroke Data
- Stroke ID
- Segment ID

Main Thread
- Initialize Octree
- Update view matrix & controller pos.
- Render
- Store stroke data
- Update texture block to GPU

Paint Thread
- Color and mark cell
- One-level refine/coarsen marked cell

GPU
- Update 3-neighbor texture
- Render heat map
- Render Scene (Ray Casting)
- Update Scene Quad Tree
- Interpolate Scene Quad Tree

Update Block Ordered Set
- block ID
CPU Side Octree

- Strong 2-to-1 Balanced Tree [Kim15]
  - Root array (Uniform grid) + Tree
  - Simple primal-only tree
  - Maximum depth level: 26
  - Physical Unit: \(0.3 \text{mm}^3\sim 40 \text{km}^3\)

- Each cell has
  - Parent ID
  - Child #0 ID
  - Flags – depth, refine, coarsen, etc.
  - RGBA

[Kim15] Byungmoon Kim, Panagiotis Tsiotras, Jeong-Mo Hong, and Oh-young Song, Interpolation and parallel adjustment of center-sampled trees with new balancing constraints
GPU Side Octree

- GPU has shadow octree of CPU octree
  - Memory management benefits from CPU
  - Convert 1D Array Fields → 2D Array Texture
  - Size of texture image: 30MB

- Only updates blocks of texture
  - Block: $M \times N$ Texels
  - Brush causes only local changes with tree
  - Tree Index is located on same texel regardless of cell
At each frame, do only one-level refinement/coarsening
- Refinement/Coarsening will finished less than #Max Depth frames
- While tree traversal, color the cells and find cells to be refined simultaneously
- “Outside” cell helps to reduce tree traversal cost
... and Update Tree on GPU

Frame $t_0$

Cell to be updated

Push block ID: 0

Update Block
Ordered Set

block ID 0 1 2 7 3 11

Main Thread:
Update one block in every frame

Frame $t_1$

Push block ID: 1

Frame $t_n$

Pop block ID
Ray Casting in Large Canvas
Ray Casting in Large Canvas

Problem 1
Tree traversal from root to leaf at every sample points

Problem 2
Sampling at empty space is wasting time

Problem 3
Error increases along the ray
Octree: 3-Neighbors

- Tree traversal from root to leaf at every sample points
  → Tree traversal using neighbor cells with ray direction

- Thanks to strong 2-to-1 balance tree,
  - A cell always has 6 neighbors
  - 3 neighbors share the parent
    (Their ID can be computed by using offset)
  - 3 neighbors have different parent

→ If we precompute only 3 Neighbors, we can move to next neighbor directly
It’s not so far........

Using World Coordinate System

Using Local Coordinate System
vec3 rayCastWithLocalCoord(in vec3 rayDir,
in float rayMaxLength,
in uint initCellID,
in vec3 initRelativeVec)
{
    posCellID = initCellID;
posRelVec = initRelativeVec;

    while(rayLength <= rayMaxLength)
    {
        //find neighbor cell
        (nbDir, minDist) = rayBoxIntersection(rayDir, posRelVec)
        neighborID = getNextNeighbor(posCellID, neighborDir)
        rayLength += minDist*cellWidth;

        if(cellHasColor)
            colorSum = blendColor(colorVolume);

        //update current cell and relative vector
        posCellID = neighborID;
posRelVec = computeRelativeVectorInNextFrame(posRelVec);
    }
}
Foveated Rendering

With Screen resolution $W \times H$,

**Volume Ray Casting**

- Convert Screen coordinates to Tree Texture coordinates
- Make Orphan Nodes according to level

**Screen Quad Tree**

**Screen Quad Tree Samples**

**Heat Map**

**Final Image**

$\frac{W}{4} \times \frac{H}{4}$

$\frac{W}{2} \times \frac{H}{2}$

$W \times H$
Summary

• Dynamic and Simple Octree both on CPU and GPU
  • Shadow octree on GPU and local updates
  • One-level refine/coarsen strategy

• Ray Casting in Large Canvas
  • 3-neighbor and ray casting with local coordinates
  • Foveated Rendering

• Future work
  • Performance optimization
  • Improve assistive tools
  • Isosurface Rendering
Floating Island
Max Voxel Resolution: $2^{26} \times 2^{26} \times 2^{26}$
Thank you😊

Ack. :

Project Webpage: [http://graphics.ewha.ac.kr/canvox/](http://graphics.ewha.ac.kr/canvox/)

Yeojin Kim, yeojinkim@ewhain.net

Byungmoon Kim, bmkim@adobe.com

Jiyang Kim, soarmin11@ewhain.net

Young J. Kim, kimy@ewha.ac.kr