

# Trax Memory Layout

CS 6965 Fall 2011

# Memory Hierarchy

- Instruction memory
  - Stored as `Vector<Instruction*>`
  - Branches jump to addresses
- Local memory
  - Everything on the stack
  - No dynamic memory support (yet)
- Global memory
  - Read only: light, camera, objects, bvh
  - Write only: frame buffer

# Instruction Memory

- Loaded at run time
- Assembler.cc
- Shared instruction caches
- word addressed
- --num-icaches 2
- --num-icache-banks 16
- --load-assembly <filename>

```
LOAD r7, r20, 1
SWI r7, r1, 512
LOAD r3, r20, 2
SWI r3, r1, 516
LOAD r4, r20, 4
SWI r4, r1, 380
LOAD r5, r20, 5
SWI r5, r1, 520
LOAD r6, r20, 7
SWI r6, r1, 524
LOAD r8, r20, 8
SWI r8, r1, 400
LOAD r9, r20, 9
SWI r9, r1, 408
LOAD r10, r20, 10
SWI r10, r1, 384
```

# Local Memory

- Stack space
- r1 is the stack pointer
- LocalStore.cc
- Byte addressed
- 1 cycle access time

```
LOAD r7, r20, 1
SWI r7, r1, 512
LOAD r3, r20, 2
SWI r3, r1, 516
LOAD r4, r20, 4
SWI r4, r1, 380
LOAD r5, r20, 5
SWI r5, r1, 520
LOAD r6, r20, 7
SWI r6, r1, 524
LOAD r8, r20, 8
SWI r8, r1, 400
LOAD r9, r20, 9
SWI r9, r1, 408
LOAD r10, r20, 10
SWI r10, r1, 384
```

# Global Memory

- Global read and write data
- LOAD, STORE
- Reads are cached
  - L1Cache.cc, L2Cache.cc
- Word addressed
- Variable access time

```
LOAD r7, r20, 1
SWI r7, r1, 512
LOAD r3, r20, 2
SWI r3, r1, 516
LOAD r4, r20, 4
SWI r4, r1, 380
LOAD r5, r20, 5
SWI r5, r1, 520
LOAD r6, r20, 7
SWI r6, r1, 524
LOAD r8, r20, 8
SWI r8, r1, 400
LOAD r9, r20, 9
SWI r9, r1, 408
LOAD r10, r20, 10
SWI r10, r1, 384
```

# What is in Global Memory?

- LoadMemory.cc
- Data structures
- Frame buffer
- Constants

```
// Setup memory for Assignment 1 ;)  
mem[0].uvalue = 0;  
// width, inv_width, fwidth  
mem[1].uvalue = image_width;  
mem[2].fvalue = 1.f/static_cast<float>(image_width);  
mem[3].fvalue = static_cast<float>(image_width);  
// height, inv_height, fheight  
mem[4].uvalue = image_height;  
mem[5].fvalue = 1.f/static_cast<float>(image_height);  
mem[6].fvalue = static_cast<float>(image_height);
```

# Global Memory cont.

```
// Pointers (actually written at the end
// mem[7].ivalue = start_fb
// mem[8].ivalue = start_scene;
// mem[9].uvalue = start_matls;
// mem[10].uvalue = start_camera;
// mem[11].uvalue = start_bg_color;
// mem[12].uvalue = start_light;
// mem[13].ivalue = start_permutation;
// mem[14].uvalue = end_memory;
// mem[15].uvalue = size-1;
  mem[16].ivalue = ray_depth;
  mem[17].ivalue = num_samples;
  mem[18].fvalue = epsilon;
// mem[19].ivalue = start_hammersley;
```

# Global Memory cont.

```
// mem[20] was used for atomic_inc address, but does not use memory anymore, in GlobalRegisterFile  
// keep this memory position free for backwards compatability
```

```
// mem[21].ivalue = num_nodes;  
// mem[22].ivalue = start_costs;  
    mem[23].ivalue = (int)log2(num_rotation_threads); // + 1 for finer granularity on work assignments  
// mem[24].ivalue = start_secondary_bvh;  
    mem[25].ivalue = num_rotation_threads;  
//mem[26].ivalue = start_subtree_sizes;  
//mem[27].ivalue = start_rotated_flags;  
//mem[28].ivalue = start_triangles;  
//mem[29].ivalue = num_triangles;
```

# Work Queue

```
// Build the work queue
start_wq = 30;
printf("Work queue starts at %d (0x%08x)\n", start_wq, start_wq);
// int start_queue_storage = start_wq + 2; // need cur_tile and num_tiles
// int num_tiles = 0;
// for (int y = 0; y < image_height; y += tile_height) {
//   for (int x = 0; x < image_width; x += tile_width) {
//     int start_x = x;
//     int stop_x = (x+tile_width < image_width) ? x+tile_width : image_width;
//     int start_y = y;
//     int stop_y = (y+tile_height < image_height) ? y+tile_height : image_height;
//     mem[start_queue_storage + num_tiles * 4 + 0].ivalue = start_x;
//     mem[start_queue_storage + num_tiles * 4 + 1].ivalue = stop_x;
//     mem[start_queue_storage + num_tiles * 4 + 2].ivalue = start_y;
//     mem[start_queue_storage + num_tiles * 4 + 3].ivalue = stop_y;
//     num_tiles++;
//   }
// }
// mem[start_wq + 0].ivalue = num_tiles;
// printf("Have %d tiles\n", num_tiles);
```

# Frame Buffer

- Space allocated
  - $\text{width} * \text{height} * 3$
  - Not initialized
- `--width 256`
- `--height 256`

```
// Pointers (actually written at the end
// mem[7].ivalue = start_fb
// mem[8].ivalue = start_scene;
// mem[9].uvalue = start_mats;
// mem[10].uvalue = start_camera;
// mem[11].uvalue = start_bg_color;
// mem[12].uvalue = start_light;
// mem[13].ivalue = start_permutation;
// mem[14].uvalue = end_memory;
// mem[15].uvalue = size-1;
mem[16].ivalue = ray_depth;
mem[17].ivalue = num_samples;
mem[18].fvalue = epsilon;
// mem[19].ivalue = start_hammersley;
```

# Scene

- Comprises a lot of stuff
- `--model models/cornell.obj`
- More in a little bit

```
// Pointers (actually written at the end
// mem[7].ivalue = start_fb
// mem[8].ivalue = start_scene;
// mem[9].uvalue = start_matris;
// mem[10].uvalue = start_camera;
// mem[11].uvalue = start_bg_color;
// mem[12].uvalue = start_light;
// mem[13].ivalue = start_permutation;
// mem[14].uvalue = end_memory;
// mem[15].uvalue = size-1;
mem[16].ivalue = ray_depth;
mem[17].ivalue = num_samples;
mem[18].fvalue = epsilon;
// mem[19].ivalue = start_hammersley;
```

# Materials

- Array of materials
- You get pointers from triangles
- .mtl file is included by .obj
- Material.cc

```
// Pointers (actually written at the end
// mem[7].ivalue = start_fb
// mem[8].ivalue = start_scene;
// mem[9].uvalue = start_matsl;
// mem[10].uvalue = start_camera;
// mem[11].uvalue = start_bg_color;
// mem[12].uvalue = start_light;
// mem[13].ivalue = start_permutation;
// mem[14].uvalue = end_memory;
// mem[15].uvalue = size-1;
mem[16].ivalue = ray_depth;
mem[17].ivalue = num_samples;
mem[18].fvalue = epsilon;
// mem[19].ivalue = start_hammersley;
```

# Materials

```
void Material::LoadIntoMemory(int& memory_position, int max_memory, FourByte* memory)
{
    memory[memory_position].ivalue = mat_type;
    memory_position++;
    for (int i = 0; i < 3; i++) {
        memory[memory_position].fvalue = c0[i];
        memory_position++;
    }
    for (int i = 0; i < 3; i++) {
        memory[memory_position].fvalue = c1[i];
        memory_position++;
    }
    for (int i = 0; i < 3; i++) {
        memory[memory_position].fvalue = c2[i];
        memory_position++;
    }
}
```

Example .mtl file:

```
newmtl shinyred
Ka 0.1986 0.0000 0.0000
Kd 0.5922 0.0166 0.0000
Ks 0.5974 0.2084 0.2084
illum 2
Ns 100.2237
```

# Camera

- Camera loaded from file
- `--view-file views/cornell.view`
- `Camera.cc`

```
// Pointers (actually written at the end
// mem[7].ivalue = start_fb
// mem[8].ivalue = start_scene;
// mem[9].uvalue = start_mats;
// mem[10].uvalue = start_camera;
// mem[11].uvalue = start_bg_color;
// mem[12].uvalue = start_light;
// mem[13].ivalue = start_permutation;
// mem[14].uvalue = end_memory;
// mem[15].uvalue = size-1;
mem[16].ivalue = ray_depth;
mem[17].ivalue = num_samples;
mem[18].fvalue = epsilon;
// mem[19].ivalue = start_hammersley;
```

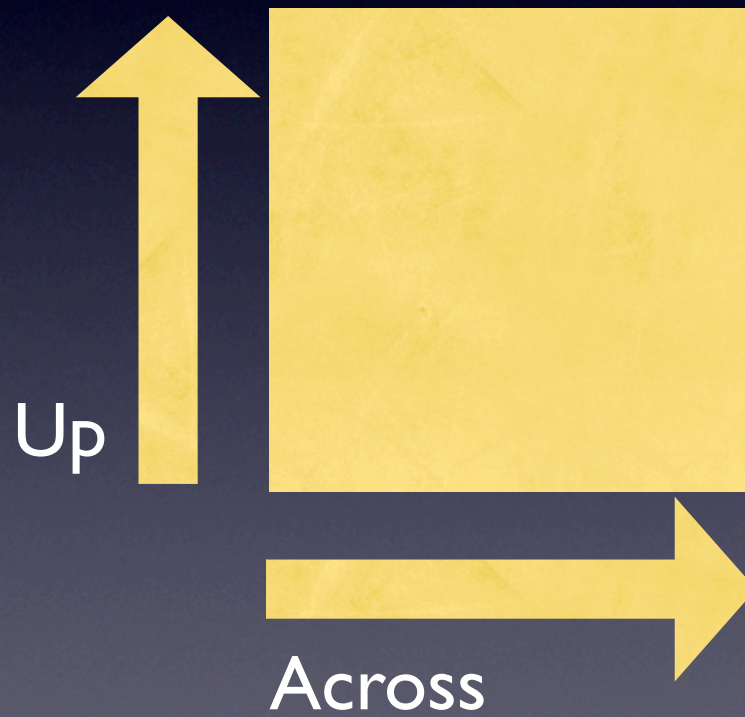
# Camera Loading

```
LoadVector(memory_position, max_memory, memory, center); //eye  
LoadVector(memory_position, max_memory, memory, corner);  
LoadVector(memory_position, max_memory, memory, across);  
LoadVector(memory_position, max_memory, memory, up);
```

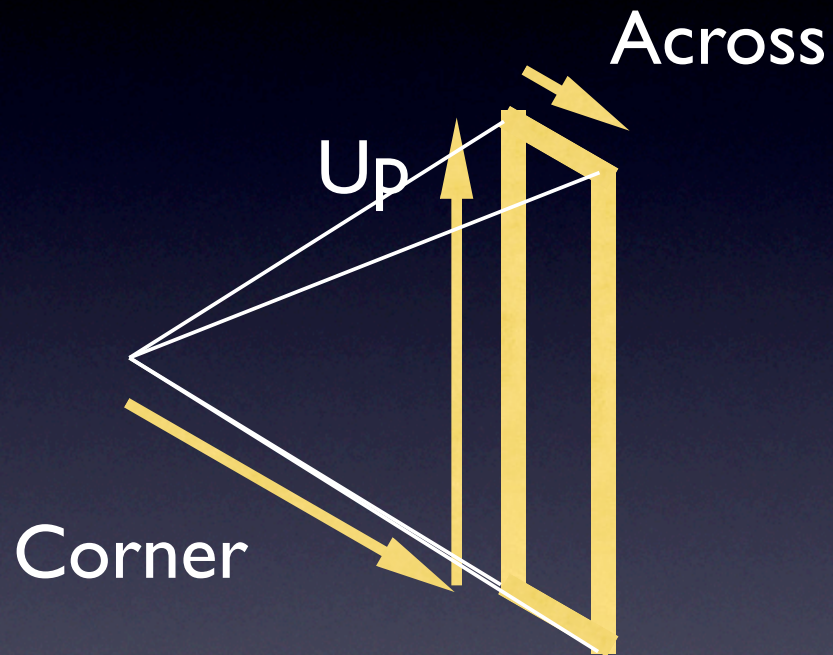
```
LoadVector(memory_position, max_memory, memory, unitgaze);  
LoadVector(memory_position, max_memory, memory, u);  
LoadVector(memory_position, max_memory, memory, v);  
memory[memory_position].fvalue = radius;  
memory_position++;
```

```
return ( corner - eye + across * ( i * inv_width ) + up * ( j * inv_height ) ).unit();
```

# Camera Model



# Camera Model



# View File

Radius Distance

278 -550 273 0 | 0 0 0 | 0 -1 | -1 | 2

Eye, Gaze, Up

Left, Right, Bottom, Top

# Background

- Location of background color
- LoadMemory.cc

```
float bg_color[3] = {1.0f, 1.0f, 1.0f};  
for (int i = 0; i < 3; i++) {  
    mem[start_bg_color + i].fvalue = bg_color[i];  
}
```

```
// Pointers (actually written at the end  
// mem[7].ivalue = start_fb  
// mem[8].ivalue = start_scene;  
// mem[9].uvalue = start_matls;  
// mem[10].uvalue = start_camera;  
// mem[11].uvalue = start_bg_color;  
// mem[12].uvalue = start_light;  
// mem[13].ivalue = start_permutation;  
// mem[14].uvalue = end_memory;  
// mem[15].uvalue = size-1;  
    mem[16].ivalue = ray_depth;  
    mem[17].ivalue = num_samples;  
    mem[18].fvalue = epsilon;  
// mem[19].ivalue = start_hammersley;
```

# Lights

- Only one loaded
- Comes from a file
- Loaded in LoadMemory.cc

```
for (int i = 0; i < 3; i++) {  
    mem[start_light + i].fvalue = light_pos[i];  
}
```

- --light-file lights/cornell.light

```
// Pointers (actually written at the end  
// mem[7].ivalue = start_fb  
// mem[8].ivalue = start_scene;  
// mem[9].uvalue = start_matls;  
// mem[10].uvalue = start_camera;  
// mem[11].uvalue = start_bg_color;  
// mem[12].uvalue = start_light;  
// mem[13].ivalue = start_permutation;  
// mem[14].uvalue = end_memory;  
// mem[15].uvalue = size-1;  
mem[16].ivalue = ray_depth;  
mem[17].ivalue = num_samples;  
mem[18].fvalue = epsilon;  
// mem[19].ivalue = start_hammersley;
```

# Light File

278.0 279.5 540.0

# Tables

- We won't use them
- More info in LoadMemory.cc

```
// Pointers (actually written at the end
// mem[7].ivalue = start_fb
// mem[8].ivalue = start_scene;
// mem[9].uvalue = start_matls;
// mem[10].uvalue = start_camera;
// mem[11].uvalue = start_bg_color;
// mem[12].uvalue = start_light;
// mem[13].ivalue = start_permutation;
// mem[14].uvalue = end_memory,
// mem[15].uvalue = size-1;
mem[16].ivalue = ray_depth;
mem[17].ivalue = num_samples;
mem[18].fvalue = epsilon;
// mem[19].ivalue = start_hammersley;
```

# End Memory

- Where the “used” memory ends
- Free memory
  - Starts at `end_memory`
  - Ends at `size - 1`
- size is set in config

```
// Pointers (actually written at the end
// mem[7].ivalue = start_fb
// mem[8].ivalue = start_scene;
// mem[9].uvalue = start_matls;
// mem[10].uvalue = start_camera;
// mem[11].uvalue = start_bg_color;
// mem[12].uvalue = start_light;
// mem[13].ivalue = start_permutation;
// mem[14].uvalue = end_memory;
// mem[15].uvalue = size-1;
mem[16].ivalue = ray_depth;
mem[17].ivalue = num_samples;
mem[18].fvalue = epsilon;
// mem[19].ivalue = start_hammersley;
```

# Program Constants

- Loaded by the simulator at runtime

- ray\_depth

- --ray-depth l

- num\_samples

- --num-samples l

- epsilon

- --epsilon l e-4

```
// Pointers (actually written at the end
// mem[7].ivalue = start_fb
// mem[8].ivalue = start_scene;
// mem[9].uvalue = start_matls;
// mem[10].uvalue = start_camera;
// mem[11].uvalue = start_bg_color;
// mem[12].uvalue = start_light;
// mem[13].ivalue = start_permutation;
// mem[14].uvalue = end_memory;
// mem[15].uvalue = size-1;
mem[16].ivalue = ray_depth;
mem[17].ivalue = num_samples;
mem[18].fvalue = epsilon;
// mem[19].ivalue = start_hammersley;
```

# Other Stuff

- Old stuff
- Used for tree rotations
- num\_nodes might be used

```
// mem[20] was used for atomic_inc  
  
// mem[21].ivalue = num_nodes;  
// mem[22].ivalue = start_costs;  
mem[23].ivalue = (int)log2(num_rotation_threads);  
// + 1 for finer granularity on work assignments  
// mem[24].ivalue = start_secondary_bvh;  
mem[25].ivalue = num_rotation_threads;  
//mem[26].ivalue = start_subtree_sizes;  
//mem[27].ivalue = start_rotated_flags;  
//mem[28].ivalue = start_triangles;  
//mem[29].ivalue = num_triangles;
```

# Scene stuff

# Triangles

- Address of first triangle
- Number of triangles
- Offset is 11
- `--model models/cornell.obj`

```
// mem[20] was used for atomic_inc  
  
// mem[21].ivalue = num_nodes;  
// mem[22].ivalue = start_costs;  
mem[23].ivalue = (int)log2(num_rotation_threads);  
// + 1 for finer granularity on work assignments  
// mem[24].ivalue = start_secondary_bvh;  
mem[25].ivalue = num_rotation_threads;  
//mem[26].ivalue = start_subtree_sizes;  
//mem[27].ivalue = start_rotated_flags;  
//mem[28].ivalue = start_triangles;  
//mem[29].ivalue = num_triangles;
```

# Triangles

- Triangle.cc

```
for (int i = 0; i < 3; i++) {  
    memory[memory_position].fvalue = p0[i];  
    memory_position++;  
}  
for (int i = 0; i < 3; i++) {  
    memory[memory_position].fvalue = p1[i];  
    memory_position++;  
}  
for (int i = 0; i < 3; i++) {  
    memory[memory_position].fvalue = p2[i];  
    memory_position++;  
}  
memory[memory_position].ivalue = object_id;  
memory_position++;  
memory[memory_position].ivalue = shader_id;  
memory_position++;
```

# BVH

- Loaded by default
- More info in BVH.cc

```
// Pointers (actually written at the end
// mem[7].ivalue = start_fb;
// mem[8].ivalue = start_scene;
// mem[9].uvalue = start_matrices;
// mem[10].uvalue = start_camera;
// mem[11].uvalue = start_bg_color;
// mem[12].uvalue = start_light;
// mem[13].ivalue = start_permutation;
// mem[14].uvalue = end_memory;
// mem[15].uvalue = size-1;
mem[16].ivalue = ray_depth;
mem[17].ivalue = num_samples;
mem[18].fvalue = epsilon;
// mem[19].ivalue = start_hammersley;
```

# BVH

- Loads boxes as triples
- Axis aligned
- num\_children
  - number of triangles
  - -1 for interior
- start\_child
  - pointer to first triangle
  - node ID

```
for (int i = 0; i < 3; i++) {  
    memory[memory_position].fvalue = box_min[i];  
    memory_position++;  
}  
for (int i = 0; i < 3; i++) {  
    box_max[i] += 0.001; // don't allow for 2d boxes  
    memory[memory_position].fvalue = box_max[i];  
    memory_position++;  
}  
memory[memory_position].ivalue = num_children;  
memory_position++;  
  
memory[memory_position].ivalue = start_child;  
memory_position++;
```

# BVH

- Siblings are next to each other
- Node size is 8
- Root node is at start\_scene

# Grid

- Loaded if chosen
  - `--usegrid <grid dimensions>`
  - number of cells in each dimension
- More info in `Grid.cc`

```
// Pointers (actually written at the end
// mem[7].ivalue = start_fb;
// mem[8].ivalue = start_scene;
// mem[9].uvalue = start_matrices;
// mem[10].uvalue = start_camera;
// mem[11].uvalue = start_bg_color;
// mem[12].uvalue = start_light;
// mem[13].ivalue = start_permutation;
// mem[14].uvalue = end_memory;
// mem[15].uvalue = size-1;
mem[16].ivalue = ray_depth;
mem[17].ivalue = num_samples;
mem[18].fvalue = epsilon;
// mem[19].ivalue = start_hammersley;
```

# Grid

- Loads bounding box
- Diagonal vector
- Size of a cell in each dimension

```
// load bounds, diagonal, cellsize
for(i=0; i<3; i++)
{
    memory[memory_position+i].fvalue = bounds.box_min[i];
    memory[memory_position+i+3].fvalue = bounds.box_max[i];
    memory[memory_position+i+6].fvalue = diagonal[i];
    memory[memory_position+i+9].fvalue = cellSize[i];
}
```

# Grid

- Next value is the number of cells in each dimension
- Then triangles are loaded

```
// load dimensions
memory[memory_position].ivalue = dimensions;
memory_position++;
int start_cells = memory_position;
memory_position++;
// load the triangles
int start_triangles = memory_position;
for(j=0; j<triangles->size(); j++)
    triangles->at(j)->LoadIntoMemory(memory_position, max_memory, memory);
```

# Grid

```
// pointer to begining of grid cells
memory[start_cells].ivalue = memory_position;
// load grid cells (triangle count and pointers to triangles)
int grid_base_address = memory_position;
memory_position += dimensions*dimensions*dimensions*2;
for(i=0; i<dimensions*dimensions*dimensions; i++)
{
    memory[grid_base_address].ivalue = cells[i].size();
    memory[grid_base_address+1].ivalue = memory_position;

    // load triangle pointers for this grid cell
    for(j=0; j<cells[i].size(); j++)
    {
        memory[memory_position].ivalue = start_triangles + (cells[i].at(j) * 11);
        memory_position++;
    }
    grid_base_address+=2;
}
```

# Standard Trax

- 32 threads per TM
  - `--num-thread-procs 32`
- 20 TMs per L2
  - `--num-cores 20`
- 4 L2s
  - `--num-l2s 4`
- `default.config`
  - `--config-file configs/default.config`

# Config File

```
FPADD 2 8
FPMIN 1 32
FPCMP 1 32
INTADD 1 32
FPMUL 2 8
INTMUL 1 2
FPINV 20 1
CONV 1 32
BLT 1 32
BITWISE 1 32
SPHERE 40 4
DEBUG 1 100
LI 1 4096 4 5
MEMORY 100 134217728
L2 3 131072 32 5
```

# Putting it all together

```
./simhwrt --num-regs 256 --num-thread-procs 1  
--threads-per-proc 1 --num-cores 1 --simd-width 1  
--width 512 --height 512 --output-prefix out  
--config-file configs/32_wide.config --num-icaches 2  
--num-icache-banks 16 --num-l2s 1  
--view-file views/cornell_obj.view  
--model test_models/cornell/CornellBox.obj  
--light-file lights/cornell.light  
--load-assembly ../LLVM_test/raster/rt-llvm.s
```

# Makefile Scene

```
# Cornell Scene
VIEWFILE="\${SIMDIR}/views/cornell_obj.view\"
MODELFILE="\${SIMDIR}/test_models/cornell/CornellBox.obj\"
LIGHTFILE="\${SIMDIR}/lights/cornell.light\"
# Conference Scene
#VIEWFILE="\${SIMDIR}/views/conference.view\"
# MODELFILE="\${SIMDIR}/test_models/conference.iw\"
# LIGHTFILE="\${SIMDIR}/lights/conference.light\"
```

# Runtime Output

Work queue starts at 30 (0x0000001e)

FB starts at 32 (0x00000020)

FB ends at 786463 (0x000c001f)

loading model ../../sim/test\_models/cornell/

CornellBox.obj

MTL file: "../../sim/test\_models/cornell/

CornellBox.mtl"

loading material file ../../sim/test\_models/cornell/

CornellBox.mtl

Found 4 total materials

Found 32 total triangles

vertex min/max = x: (0.000000, 556.000000) y:  
(0.000000, 559.200012) z: (0.000000, 548.799988)

Materials start at 786464 (0x000c0020)

Materials end at 786589 (0x000c009d)

Starting BVH build.

BVH build complete with 63 nodes.

Scene starts at 786590 (0x000c009e)

BVH bounds [0.000000 0.000000 0.000000]

[556.000000 559.200012 548.799988]

Triangles start at 787096 (0x000c0298)

Scene ends at 787637 (0x000c04b5)

Starting camera at 787638 (0x000c04b6)

Camera ended at 787657 (0x000c04c9)

Background Color 0x000c04ca to 0x000c04cc

Light at 0x000c04cd to 0x000c04cf

Permutation table from 0x000c04d0 to 0x000c06cf

Hammersley table from 0x000c06d0 to 0x000c08cf