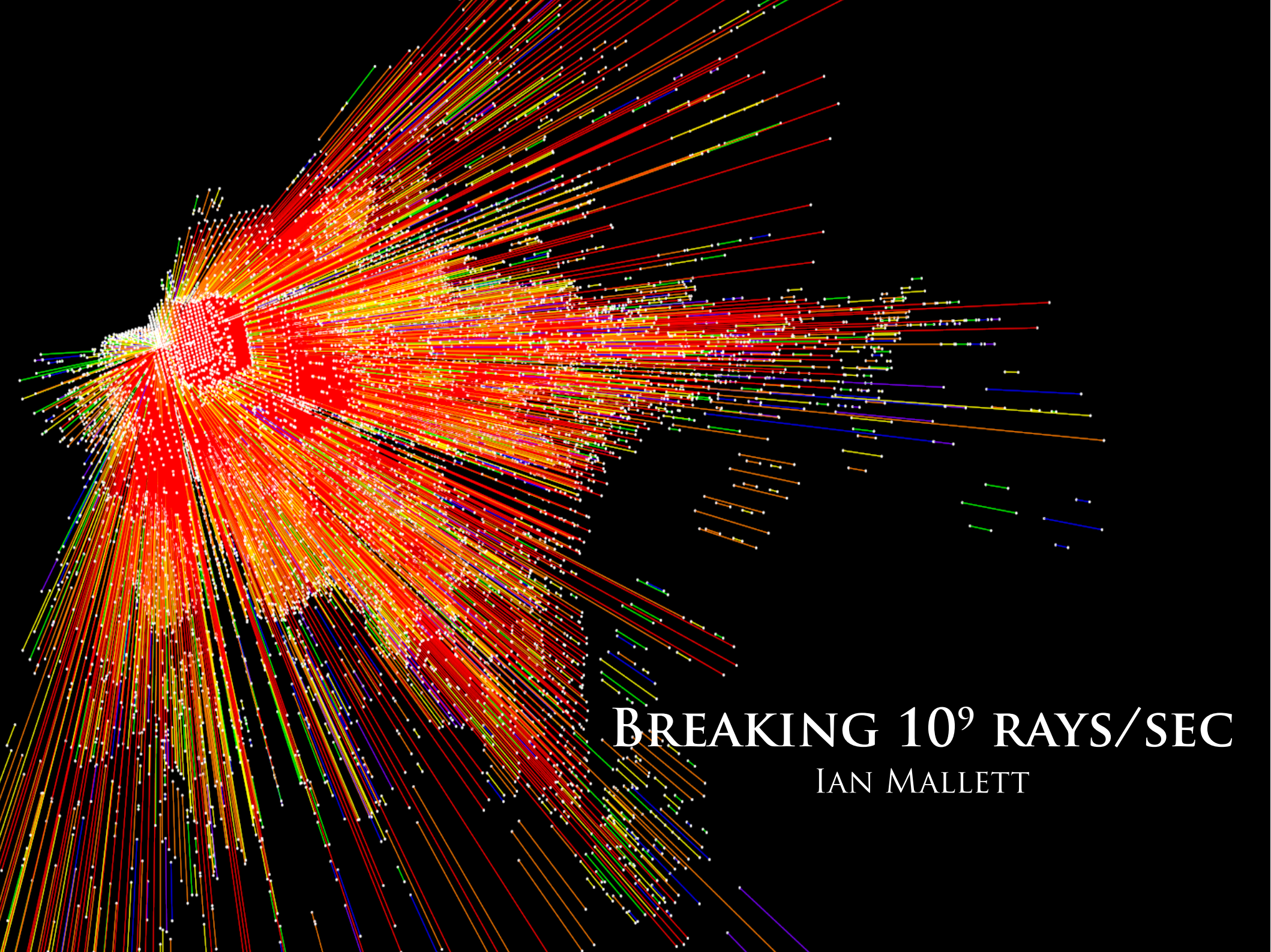


Breaking 10^9 rays/sec

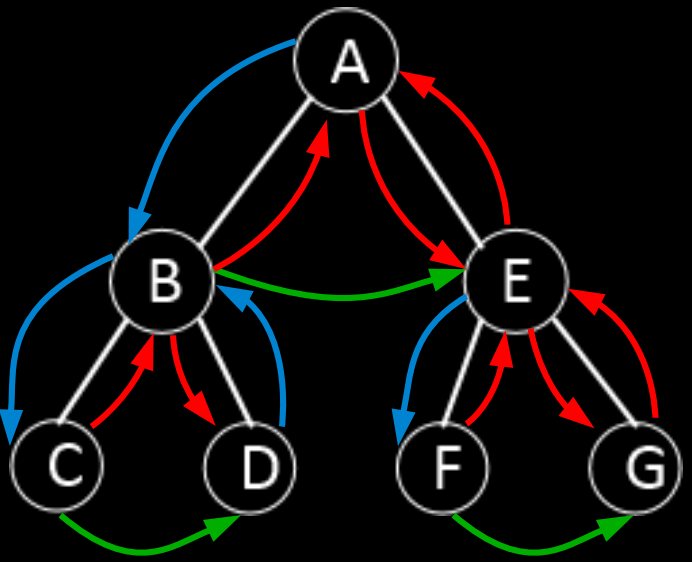
Ian Mallett

Overview

- Lots of fancy optimizations & lots of challenges
- Finished implementation of "idea 1" from original proposal: the stack-based scene-caching thing
- I *might* be getting ≥1,000,000,000 rays/second with reasonably sized chip (working on getting simulator working for all scenes before I claim success)

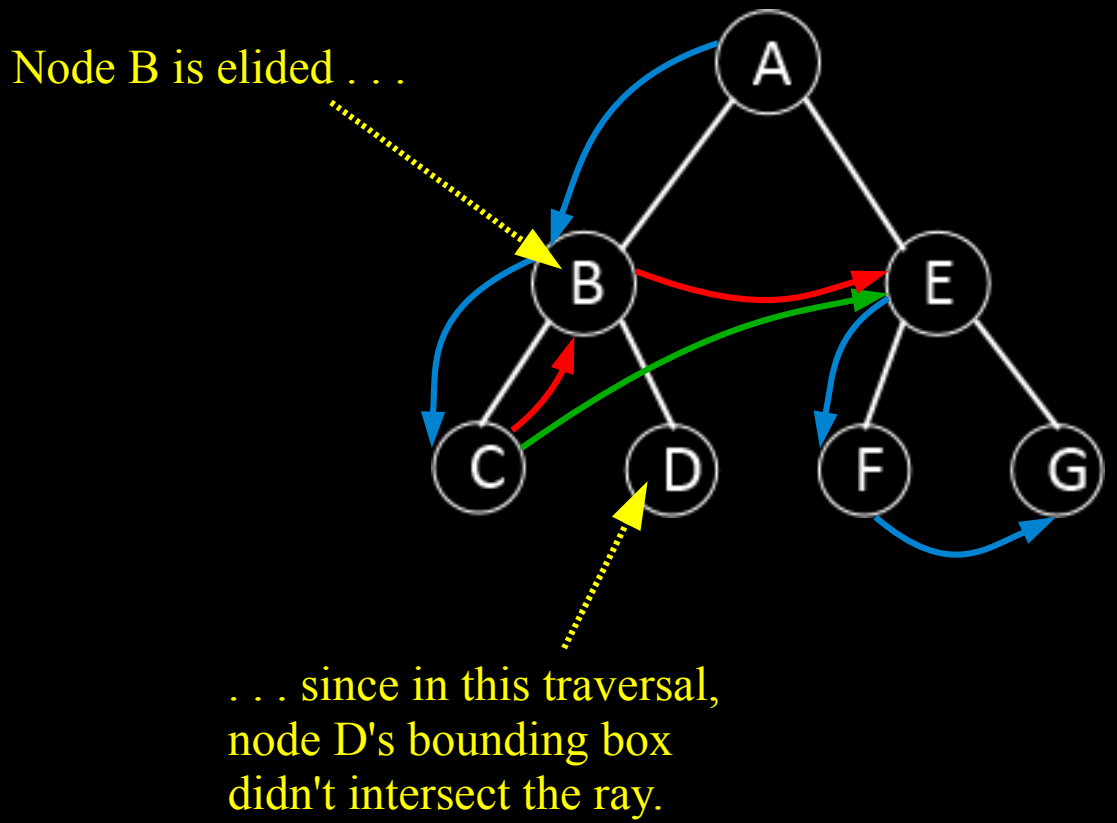
Important Optimizations

- Sidestep traversal (skip parent nodes)



Important Optimizations

- Elision of nodes where only one child hit in traversal (subtract one from depth of later nodes)



Important Optimizations

- Miscellaneous other for BVH:
  - Miscellaneous mentioned last time
    - Faster ray/box intersection
    - Nearest first & early termination
    - Non-recursive traversal
  - Miscellaneous fancy unmentioned stuff
    - Triangle edges, not verts
  - TODO:
    - Relative pointers (useful for hardware?)
    - Specialized shadow ray traversal

Important Optimizations

- Miscellaneous micro-optimization
  - Not sorry about this. Worth it.
- Sometimes didn't play nicely with compiler / simulator . . .
  - Placement new, std:: library stuff
  - Unimplemented instructions
  - Cycle counter
  - Danny is fixing/fixed all! Thanks!

Challenges

- Pointer arithmetic nightmare!
- Concurrent deletion within thread (i.e., node A's load triggers a delete, which requires node A to move within memory)
- Deletion in general (surprisingly difficult algorithm!)
- Implementation issues
- Slow compiles/runs (esp. larger scenes)
- EGSR paper deadline
- Ran out of coffee

Spot the bug

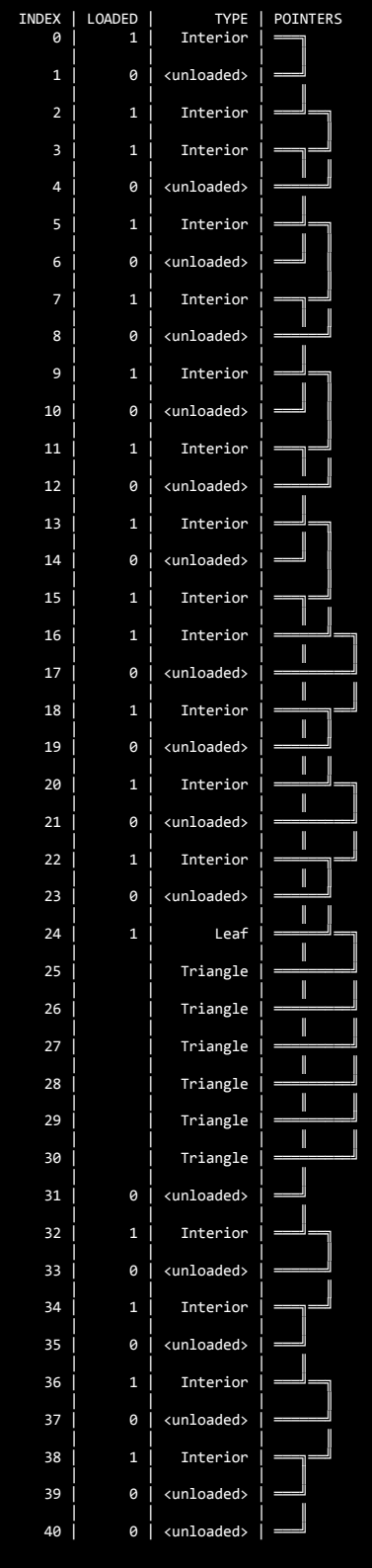
```
#ifdef BUILD_DEBUG
inline void print_stack(void) const {
    printf("Printing stack (%d/%d used):\n",_slots_used,static_cast<int>(MAX_SLOTS));
    printf("INDEX | LOADED |       TYPE | POINTERS\n");
    #define NUM_TARGETS 16
    #define CHECK_TARGETS(K,CMPVAL,COMBINE) (\
        (targets[K][1]CMPVAL) COMBINE (targets[K][2]CMPVAL) COMBINE (targets[K][3]CMPVAL) COMBINE (targets[K][4]CMPVAL) COMBINE\
        (targets[K][5]CMPVAL) COMBINE (targets[K][6]CMPVAL) COMBINE (targets[K][7]CMPVAL) COMBINE (targets[K][8]CMPVAL)\
    )
    int targets[NUM_TARGETS][1+8]; for (int i=0;i<NUM_TARGETS;++i) for (int j=0;j<8;++j) targets[i][1+j]=-1;
    for (int i=0;i<MAX_SLOTS;++i) {
        _Node const*const current = CNODE(i);
        if (i>0) { printf("      |        |            |  "); for (int k=0;k<NUM_TARGETS;++k) printf(CHECK_TARGETS(k,!=-1,||)?"╠":"═══"); printf("\n"); }
                        for (int k=0;k<31;++k) printf(" "); for (int k=0;k<NUM_TARGETS;++k) printf(CHECK_TARGETS(k,!=-1,||)?"╠":"═══"); printf("\r");
        printf("%5d | ",i);
        bool found_parent = false;
        int j=0; for (;j<NUM_TARGETS;++j) if (CHECK_TARGETS(j,==i,||)) { found_parent=true; break; }
        if (i==0 || found_parent) {
            if (i==0 || targets[j][0]==0) {
                printf("%6d | ",current->is_loaded);
                if (current->is_loaded) {
                    int k=0, m;
                    if (current->is_leaf_node) {
                        printf("      Leaf | =");
                        for (;k<NUM_TARGETS;++k) if (CHECK_TARGETS(k,==-1,&&)) break; assert(k<NUM_TARGETS,"Not enough pointers!");
                        targets[k][0] = 1;
                        for (m=0;m<IMIN(8,current->num_tris);++m) targets[k][1+m]=current->addr_tri0+m;
                    } else {
                        printf("  Interior | =");
                        for (;k<NUM_TARGETS;++k) if (CHECK_TARGETS(k,==-1,&&)) break; assert(k<NUM_TARGETS,"Not enough pointers!");
                        targets[k][0] = 0;
                        targets[k][1] = current->addr_child1;
                        targets[k][2] = current->addr_childr;
                        m = 2;
                    }
                    for (;m<8;++m) targets[k][1+m]=-1;
                    for (m=0;m<=IMAX(found_parent?j:k,k);++m) {
                        if (m==j) printf("═══╝");
                        else if (m==k) printf("═══╗");
                        else printf("═══");
                    }
                } else {
                    printf("<unloaded> | =");
                    for (int m=0;m<j;++m) printf("═══"); printf("═══╝");
                }
            } else {
                printf("       |   Triangle | =");
                for (int m=0;m<j;++m) printf("═══"); printf("═══╝");
            }
            for (int k=0;k<8;++k) if (targets[j][1+k]==i) targets[j][1+k]=-1;
        } else {
            printf("     - |    <empty> |");
        }
        printf("\n");
    }
    printf("Done!\n");
}
#endif
```

Spot the bug

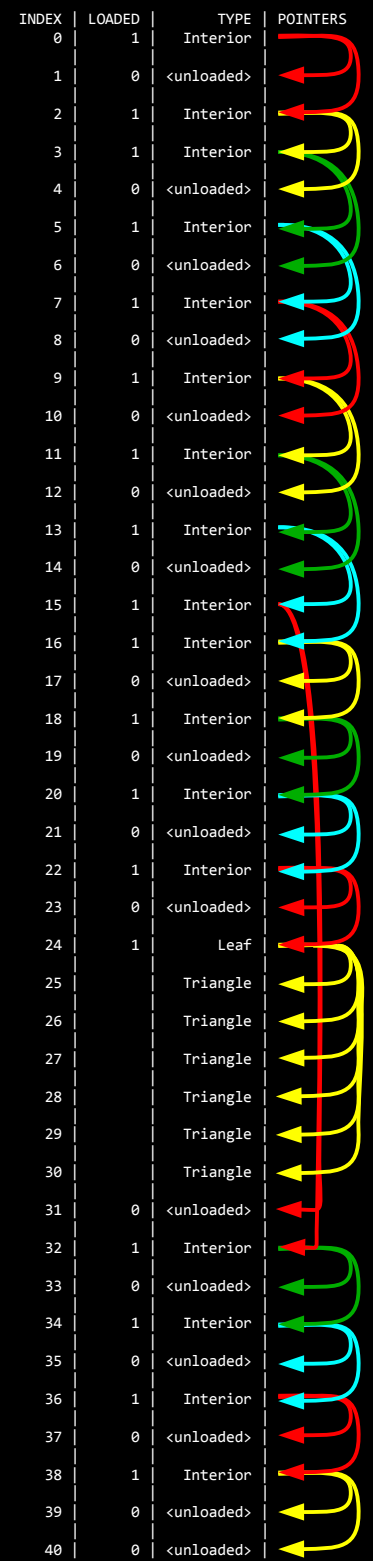
```
#ifdef BUILD_DEBUG
inline void print_stack(void) const {
    printf("Printing stack (%d/%d used):\n",_slots_used,static_cast<int>(MAX_SLOTS));
    printf("INDEX | LOADED |       TYPE | POINTERS\n");
    #define NUM_TARGETS 16
    #define CHECK_TARGETS(K,CMPVAL,COMBINE) (\
        (targets[K][1]CMPVAL) COMBINE (targets[K][2]CMPVAL) COMBINE (targets[K][3]CMPVAL) COMBINE (targets[K][4]CMPVAL) COMBINE\
        (targets[K][5]CMPVAL) COMBINE (targets[K][6]CMPVAL) COMBINE (targets[K][7]CMPVAL) COMBINE (targets[K][8]CMPVAL)\
    )
    int targets[NUM_TARGETS][1+8]; for (int i=0;i<NUM_TARGETS;++i) for (int j=0;j<8;++j) targets[i][1+j]=-1;
    for (int i=0;i<MAX_SLOTS;++i) {
        _Node const*const current = CNODE(i);
        if (i>0) { printf("      |        |            |  "); for (int k=0;k<NUM_TARGETS;++k) printf(CHECK_TARGETS(k,!=-1,||)?"╠":"═"); printf("\n"); }
                        for (int k=0;k<31;++k) printf(" "); for (int k=0;k<NUM_TARGETS;++k) printf(CHECK_TARGETS(k,!=-1,||)?"╠":"═"); printf("\r");
        printf("%5d | ",i);
        bool found_parent = false;
        int j=0; for (;j<NUM_TARGETS;++j) if (CHECK_TARGETS(j,==i,||)) { found_parent=true; break; }
        if (i==0 || found_parent) {
            if (i==0 || targets[j][0]==0) {
                printf("%6d | ",current->is_loaded);
                if (current->is_loaded) {
                    int k=0, m;
                    if (current->is_leaf_node) {
                        printf("      Leaf | =");
                        for (;k<NUM_TARGETS;++k) if (CHECK_TARGETS(k,==-1,&&)) break; assert(k<NUM_TARGETS,"Not enough pointers!");
                        targets[k][0] = 1;
                        for (m=0;m<IMIN(8,current->num_tris);++m) targets[k][1+m]=current->addr_tri0+m;
                    } else {
                        printf("  Interior | =");
                        for (;k<NUM_TARGETS;++k) if (CHECK_TARGETS(k,==-1,&&)) break; assert(k<NUM_TARGETS,"Not enough pointers!");
                        targets[k][0] = 0;
                        targets[k][1] = current->addr_child1;
                        targets[k][2] = current->addr_childr;
                        m = 2;
                    }
                    for (;m<8;++m) targets[k][1+m]=-1;
                    for (m=0;m<=IMAX(found_parent?j:k,k);++m) {
                        if (m==j) printf("═╝");
                        else if (m==k) printf("═╕");
                        else printf("══");
                    }
                } else {
                    printf("<unloaded> | =");
                    for (int m=0;m<j;++m) printf("══"); printf("═╝");
                }
            } else {
                printf("       |   Triangle | =");
                for (int m=0;m<j;++m) printf("══"); printf("═╝");
            }
            if (i!=0) for (int k=0;k<8;++k) if (targets[j][1+k]==i) targets[j][1+k]=-1;
        } else {
            printf("     - |    <empty> |");
        }
        printf("\n");
    }
    printf("Done!\n");
}
#endif
```

Clarification of Idea 1 (by Request)

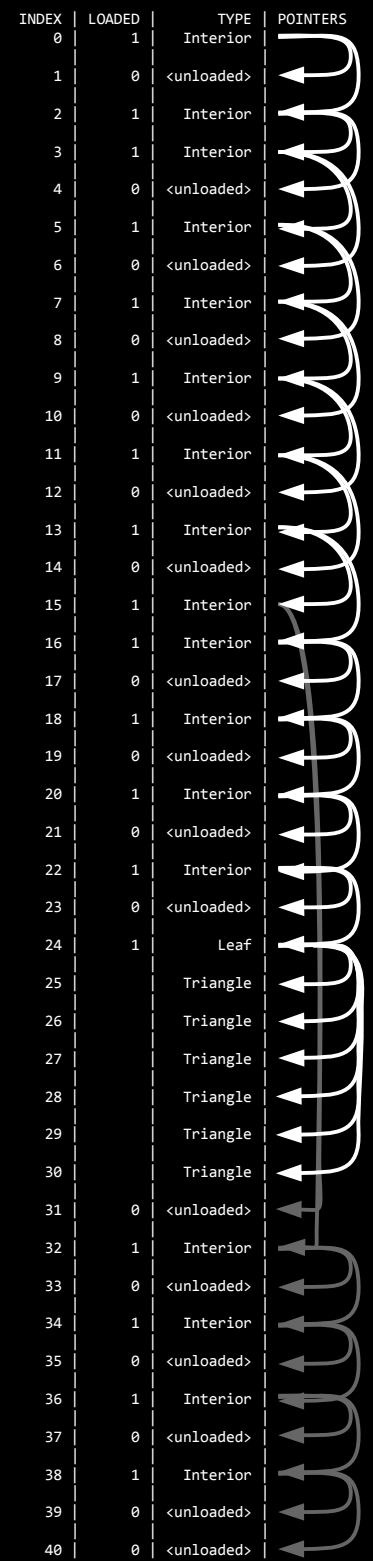
- Simple example . . .


INDEX | LOADED | TYPE | POINTERS
0 | 1 | Interior
1 | 0 | <unloaded>
2 | 1 | Interior
3 | 1 | Interior
4 | 0 | <unloaded>
5 | 1 | Interior
6 | 0 | <unloaded>
7 | 1 | Interior
8 | 0 | <unloaded>
9 | 1 | Interior
10 | 0 | <unloaded>
11 | 1 | Interior
12 | 0 | <unloaded>
13 | 1 | Interior
14 | 0 | <unloaded>
15 | 1 | Interior
16 | 1 | Interior
17 | 0 | <unloaded>
18 | 1 | Interior
19 | 0 | <unloaded>
20 | 1 | Interior
21 | 0 | <unloaded>
22 | 1 | Interior
23 | 0 | <unloaded>
24 | 1 | Leaf
25 | | Triangle
26 | | Triangle
27 | | Triangle
28 | | Triangle
29 | | Triangle
30 | | Triangle
31 | 0 | <unloaded>
32 | 1 | Interior
33 | 0 | <unloaded>
34 | 1 | Interior
35 | 0 | <unloaded>
36 | 1 | Interior
37 | 0 | <unloaded>
38 | 1 | Interior
39 | 0 | <unloaded>
40 | 0 | <unloaded>

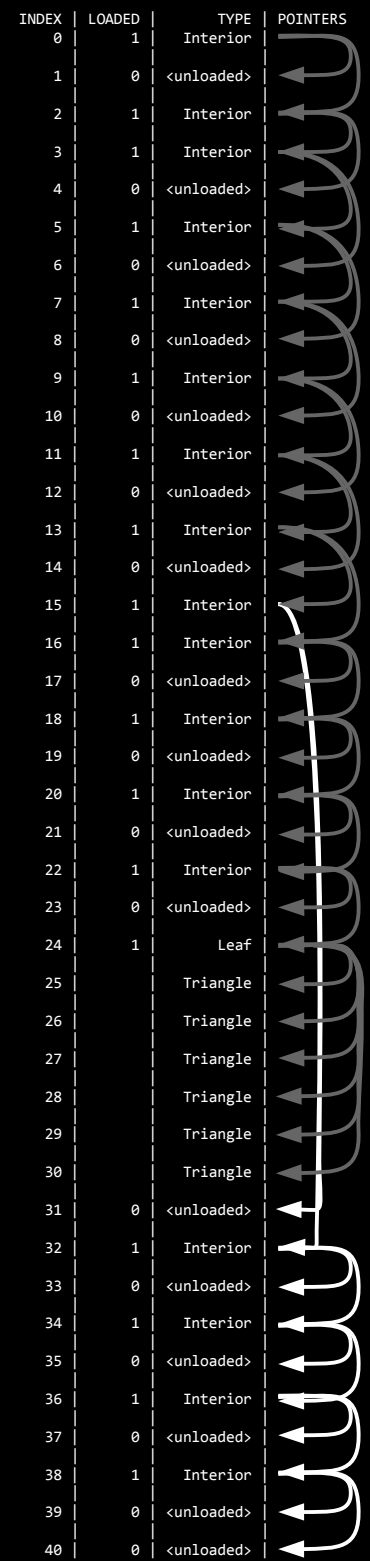
- A full stack


INDEX | LOADED | TYPE | POINTERS
0 | 1 | Interior |
1 | 0 | <unloaded> |
2 | 1 | Interior |
3 | 1 | Interior |
4 | 0 | <unloaded> |
5 | 1 | Interior |
6 | 0 | <unloaded> |
7 | 1 | Interior |
8 | 0 | <unloaded> |
9 | 1 | Interior |
10 | 0 | <unloaded> |
11 | 1 | Interior |
12 | 0 | <unloaded> |
13 | 1 | Interior |
14 | 0 | <unloaded> |
15 | 1 | Interior |
16 | 1 | Interior |
17 | 0 | <unloaded> |
18 | 1 | Interior |
19 | 0 | <unloaded> |
20 | 1 | Interior |
21 | 0 | <unloaded> |
22 | 1 | Interior |
23 | 0 | <unloaded> |
24 | 1 | Leaf |
25 | | Triangle |
26 | | Triangle |
27 | | Triangle |
28 | | Triangle |
29 | | Triangle |
30 | | Triangle |
31 | 0 | <unloaded> |
32 | 1 | Interior |
33 | 0 | <unloaded> |
34 | 1 | Interior |
35 | 0 | <unloaded> |
36 | 1 | Interior |
37 | 0 | <unloaded> |
38 | 1 | Interior |
39 | 0 | <unloaded> |
40 | 0 | <unloaded>

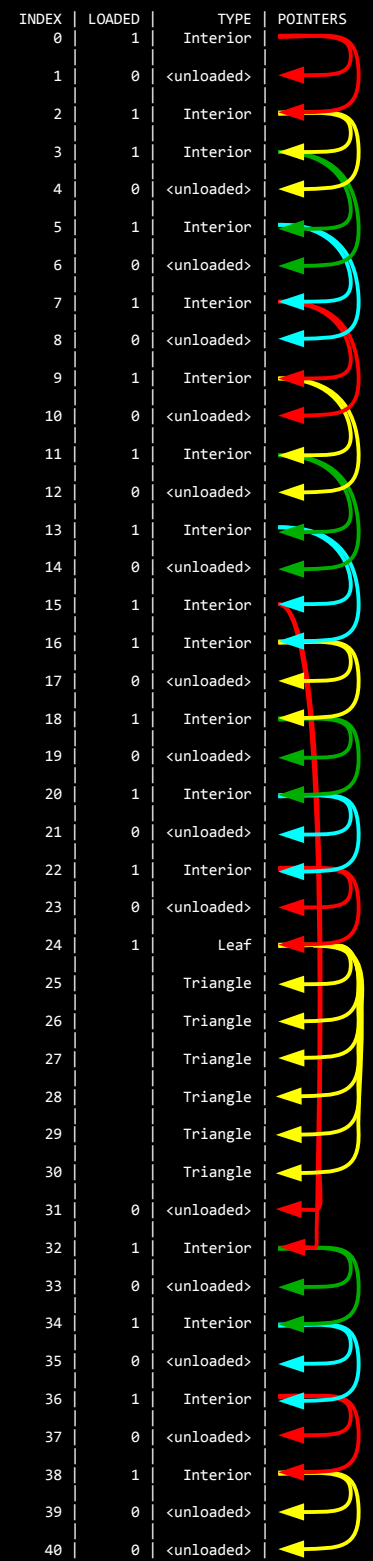
- A full stack



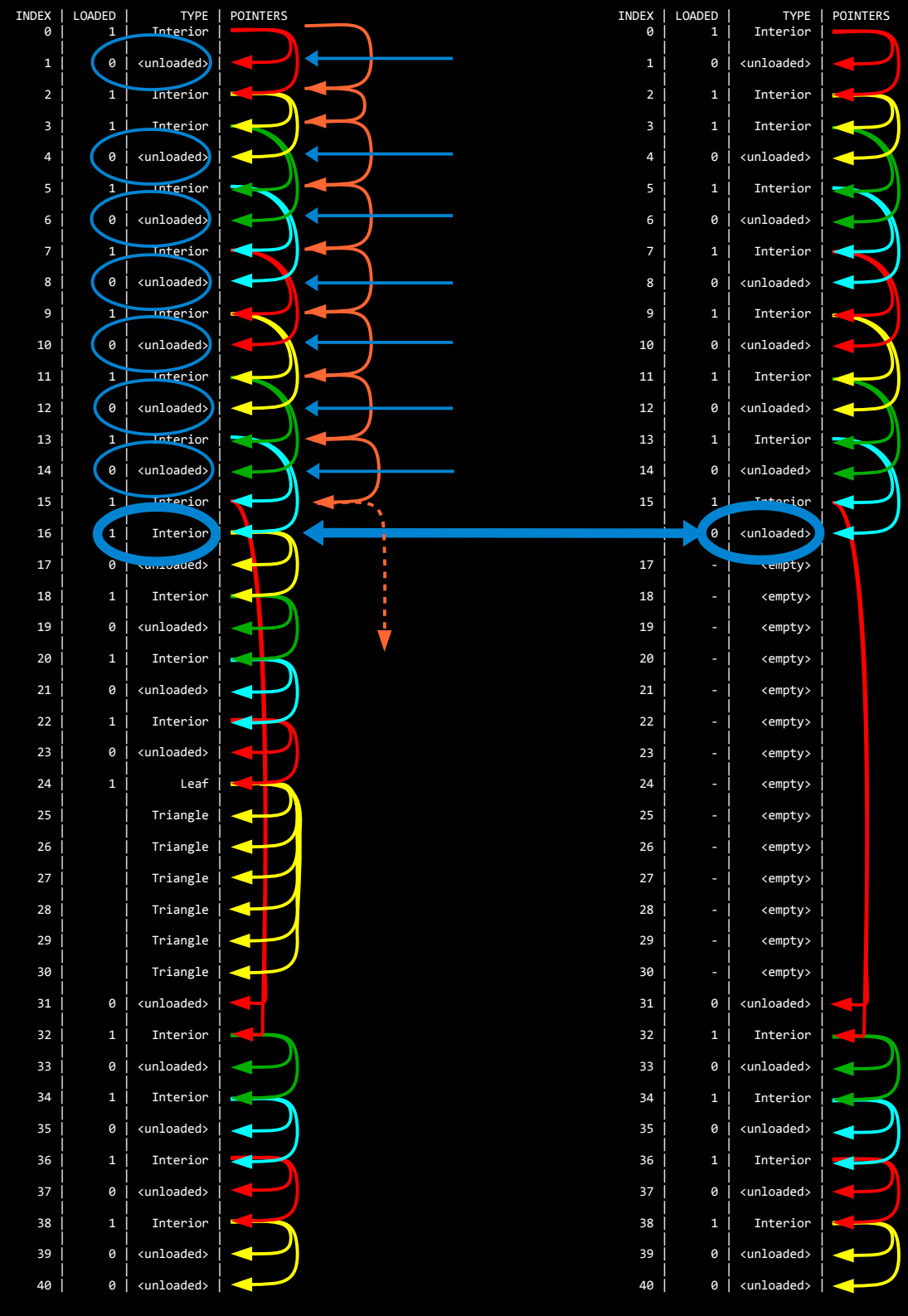
- Traversal first triggered these loads . . .


INDEX | LOADED | TYPE | POINTERS
0 | 1 | Interior |
1 | 0 | <unloaded> |
2 | 1 | Interior |
3 | 1 | Interior |
4 | 0 | <unloaded> |
5 | 1 | Interior |
6 | 0 | <unloaded> |
7 | 1 | Interior |
8 | 0 | <unloaded> |
9 | 1 | Interior |
10 | 0 | <unloaded> |
11 | 1 | Interior |
12 | 0 | <unloaded> |
13 | 1 | Interior |
14 | 0 | <unloaded> |
15 | 1 | Interior |
16 | 1 | Interior |
17 | 0 | <unloaded> |
18 | 1 | Interior |
19 | 0 | <unloaded> |
20 | 1 | Interior |
21 | 0 | <unloaded> |
22 | 1 | Interior |
23 | 0 | <unloaded> |
24 | 1 | Leaf |
25 | | Triangle |
26 | | Triangle |
27 | | Triangle |
28 | | Triangle |
29 | | Triangle |
30 | | Triangle |
31 | 0 | <unloaded> |
32 | 1 | Interior |
33 | 0 | <unloaded> |
34 | 1 | Interior |
35 | 0 | <unloaded> |
36 | 1 | Interior |
37 | 0 | <unloaded> |
38 | 1 | Interior |
39 | 0 | <unloaded> |
40 | 0 | <unloaded> |

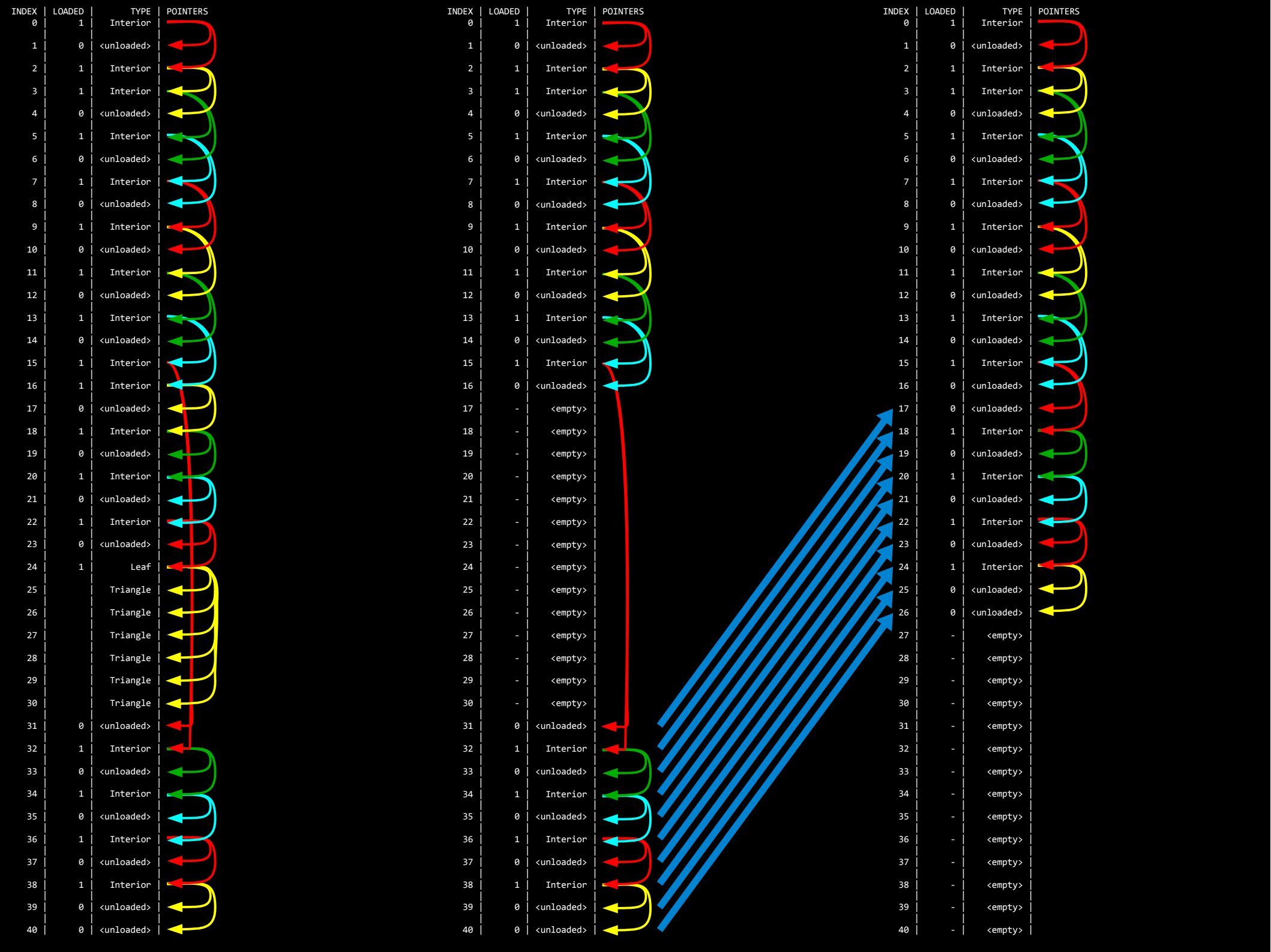
- . . . then these.


INDEX | LOADED | TYPE | POINTERS
0 | 1 | Interior |
1 | 0 | <unloaded> |
2 | 1 | Interior |
3 | 1 | Interior |
4 | 0 | <unloaded> |
5 | 1 | Interior |
6 | 0 | <unloaded> |
7 | 1 | Interior |
8 | 0 | <unloaded> |
9 | 1 | Interior |
10 | 0 | <unloaded> |
11 | 1 | Interior |
12 | 0 | <unloaded> |
13 | 1 | Interior |
14 | 0 | <unloaded> |
15 | 1 | Interior |
16 | 1 | Interior |
17 | 0 | <unloaded> |
18 | 1 | Interior |
19 | 0 | <unloaded> |
20 | 1 | Interior |
21 | 0 | <unloaded> |
22 | 1 | Interior |
23 | 0 | <unloaded> |
24 | 1 | Leaf |
25 | | Triangle |
26 | | Triangle |
27 | | Triangle |
28 | | Triangle |
29 | | Triangle |
30 | | Triangle |
31 | 0 | <unloaded> |
32 | 1 | Interior |
33 | 0 | <unloaded> |
34 | 1 | Interior |
35 | 0 | <unloaded> |
36 | 1 | Interior |
37 | 0 | <unloaded> |
38 | 1 | Interior |
39 | 0 | <unloaded> |
40 | 0 | <unloaded>

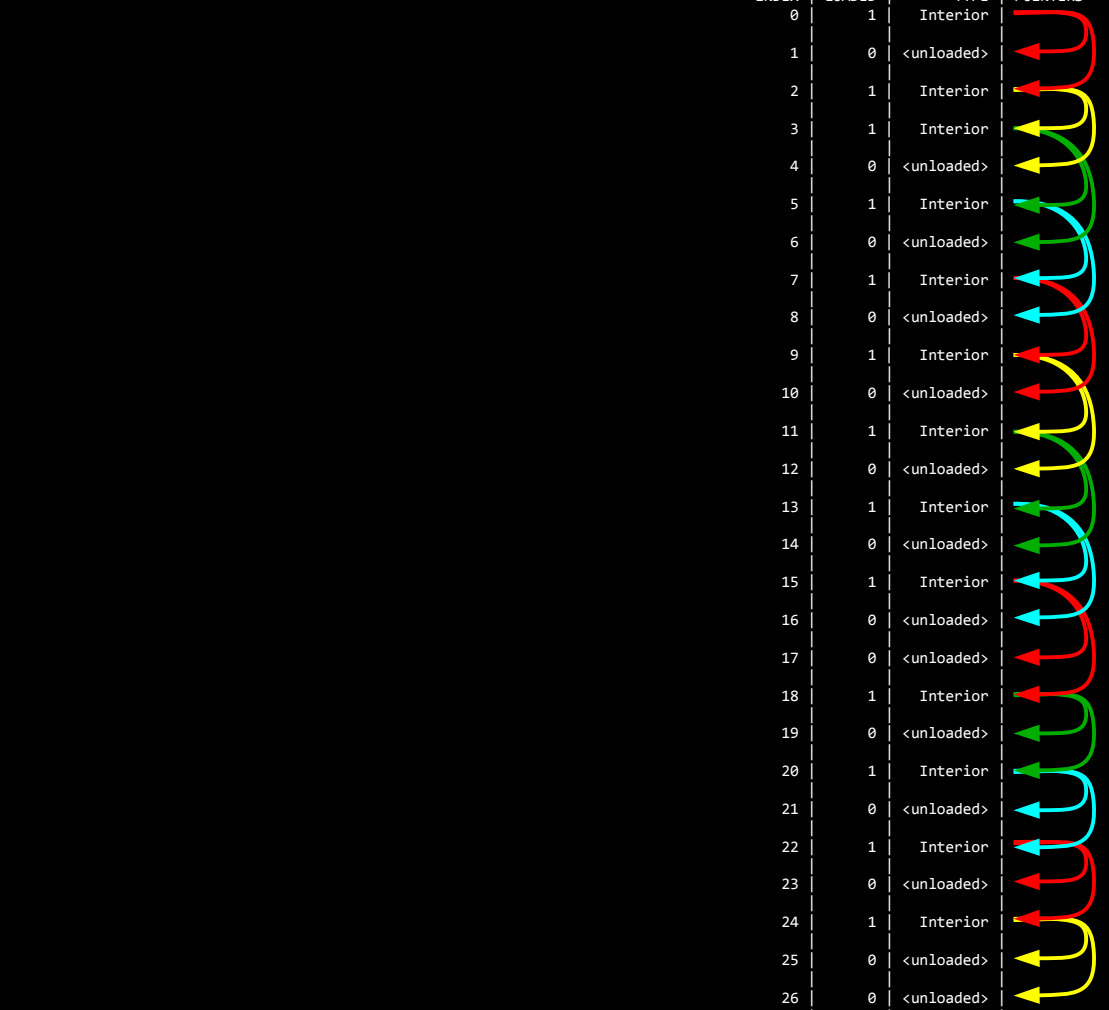
- A full stack


INDEX | LOADED | TYPE | POINTERS
0 | 1 | Interior
1 | 0 | <unloaded>
2 | 1 | Interior
3 | 1 | Interior
4 | 0 | <unloaded>
5 | 1 | Interior
6 | 0 | <unloaded>
7 | 1 | Interior
8 | 0 | <unloaded>
9 | 1 | Interior
10 | 0 | <unloaded>
11 | 1 | Interior
12 | 0 | <unloaded>
13 | 1 | Interior
14 | 0 | <unloaded>
15 | 1 | Interior
16 | 1 | Interior
17 | 0 | <unloaded>
18 | 1 | Interior
19 | 0 | <unloaded>
20 | 1 | Interior
21 | 0 | <unloaded>
22 | 1 | Interior
23 | 0 | <unloaded>
24 | 1 | Leaf
25 | | Triangle
26 | | Triangle
27 | | Triangle
28 | | Triangle
29 | | Triangle
30 | | Triangle
31 | 0 | <unloaded>
32 | 1 | Interior
33 | 0 | <unloaded>
34 | 1 | Interior
35 | 0 | <unloaded>
36 | 1 | Interior
37 | 0 | <unloaded>
38 | 1 | Interior
39 | 0 | <unloaded>
40 | 0 | <unloaded>

INDEX | LOADED | TYPE | POINTERS
0 | 1 | Interior
1 | 0 | <unloaded>
2 | 1 | Interior
3 | 1 | Interior
4 | 0 | <unloaded>
5 | 1 | Interior
6 | 0 | <unloaded>
7 | 1 | Interior
8 | 0 | <unloaded>
9 | 1 | Interior
10 | 0 | <unloaded>
11 | 1 | Interior
12 | 0 | <unloaded>
13 | 1 | Interior
14 | 0 | <unloaded>
15 | 1 | Interior
16 | 0 | <unloaded>
17 | - | <empty>
18 | - | <empty>
19 | - | <empty>
20 | - | <empty>
21 | - | <empty>
22 | - | <empty>
23 | - | <empty>
24 | - | <empty>
25 | - | <empty>
26 | - | <empty>
27 | - | <empty>
28 | - | <empty>
29 | - | <empty>
30 | - | <empty>
31 | 0 | <unloaded>
32 | 1 | Interior
33 | 0 | <unloaded>
34 | 1 | Interior
35 | 0 | <unloaded>
36 | 1 | Interior
37 | 0 | <unloaded>
38 | 1 | Interior
39 | 0 | <unloaded>
40 | 0 | <unloaded>


INDEX | LOADED | TYPE | POINTERS
0 | 1 | Interior |
1 | 0 | <unloaded> |
2 | 1 | Interior |
3 | 1 | Interior |
4 | 0 | <unloaded> |
5 | 1 | Interior |
6 | 0 | <unloaded> |
7 | 1 | Interior |
8 | 0 | <unloaded> |
9 | 1 | Interior |
10 | 0 | <unloaded> |
11 | 1 | Interior |
12 | 0 | <unloaded> |
13 | 1 | Interior |
14 | 0 | <unloaded> |
15 | 1 | Interior |
16 | 1 | Interior |
17 | 0 | <unloaded> |
18 | 1 | Interior |
19 | 0 | <unloaded> |
20 | 1 | Interior |
21 | 0 | <unloaded> |
22 | 1 | Interior |
23 | 0 | <unloaded> |
24 | 1 | Leaf |
25 | | Triangle |
26 | | Triangle |
27 | | Triangle |
28 | | Triangle |
29 | | Triangle |
30 | | Triangle |
31 | 0 | <unloaded> |
32 | 1 | Interior |
33 | 0 | <unloaded> |
34 | 1 | Interior |
35 | 0 | <unloaded> |
36 | 1 | Interior |
37 | 0 | <unloaded> |
38 | 1 | Interior |
39 | 0 | <unloaded> |
40 | 0 | <unloaded> |
INDEX | LOADED | TYPE | POINTERS
0 | 1 | Interior |
1 | 0 | <unloaded> |
2 | 1 | Interior |
3 | 1 | Interior |
4 | 0 | <unloaded> |
5 | 1 | Interior |
6 | 0 | <unloaded> |
7 | 1 | Interior |
8 | 0 | <unloaded> |
9 | 1 | Interior |
10 | 0 | <unloaded> |
11 | 1 | Interior |
12 | 0 | <unloaded> |
13 | 1 | Interior |
14 | 0 | <unloaded> |
15 | 1 | Interior |
16 | 0 | <unloaded> |
17 | - | <empty> |
18 | - | <empty> |
19 | - | <empty> |
20 | - | <empty> |
21 | - | <empty> |
22 | - | <empty> |
23 | - | <empty> |
24 | - | <empty> |
25 | - | <empty> |
26 | - | <empty> |
27 | - | <empty> |
28 | - | <empty> |
29 | - | <empty> |
30 | - | <empty> |
31 | 0 | <unloaded> |
32 | 1 | Interior |
33 | 0 | <unloaded> |
34 | 1 | Interior |
35 | 0 | <unloaded> |
36 | 1 | Interior |
37 | 0 | <unloaded> |
38 | 1 | Interior |
39 | 0 | <unloaded> |
40 | 0 | <unloaded> |
INDEX | LOADED | TYPE | POINTERS
0 | 1 | Interior |
1 | 0 | <unloaded> |
2 | 1 | Interior |
3 | 1 | Interior |
4 | 0 | <unloaded> |
5 | 1 | Interior |
6 | 0 | <unloaded> |
7 | 1 | Interior |
8 | 0 | <unloaded> |
9 | 1 | Interior |
10 | 0 | <unloaded> |
11 | 1 | Interior |
12 | 0 | <unloaded> |
13 | 1 | Interior |
14 | 0 | <unloaded> |
15 | 1 | Interior |
16 | 0 | <unloaded> |
17 | 0 | <unloaded> |
18 | 1 | Interior |
19 | 0 | <unloaded> |
20 | 1 | Interior |
21 | 0 | <unloaded> |
22 | 1 | Interior |
23 | 0 | <unloaded> |
24 | 1 | Interior |
25 | 0 | <unloaded> |
26 | 0 | <unloaded> |
27 | - | <empty> |
28 | - | <empty> |
29 | - | <empty> |
30 | - | <empty> |
31 | - | <empty> |
32 | - | <empty> |
33 | - | <empty> |
34 | - | <empty> |
35 | - | <empty> |
36 | - | <empty> |
37 | - | <empty> |
38 | - | <empty> |
39 | - | <empty> |
40 | - | <empty> |

INDEX	LOADED	TYPE	POINTERS
0	1	Interior	
1	0	<unloaded>	
2	1	Interior	
3	1	Interior	
4	0	<unloaded>	
5	1	Interior	
6	0	<unloaded>	
7	1	Interior	
8	0	<unloaded>	
9	1	Interior	
10	0	<unloaded>	
11	1	Interior	
12	0	<unloaded>	
13	1	Interior	
14	0	<unloaded>	
15	1	Interior	
16	0	<unloaded>	
17	0	<unloaded>	
18	1	Interior	
19	0	<unloaded>	
20	1	Interior	
21	0	<unloaded>	
22	1	Interior	
23	0	<unloaded>	
24	1	Interior	
25	0	<unloaded>	
26	0	<unloaded>	
27	-	<empty>	
28	-	<empty>	
29	-	<empty>	
30	-	<empty>	
31	-	<empty>	
32	-	<empty>	
33	-	<empty>	
34	-	<empty>	
35	-	<empty>	
36	-	<empty>	
37	-	<empty>	
38	-	<empty>	
39	-	<empty>	
40	-	<empty>	



Preliminary Results

- Cornell scene (simulator):
  - 2.508 gigarays / sec.
  - 446mm^2
  - 537 seconds to run!
  - 99.6% of overhead is waiting for last rays to finish . . .
- Conference scene (run_rt):
  - only 1 ray out of 50 to 100 requires load/delete
    → hopefully similar numbers, when run on simulator

Next Week

- Hopefully simulator will work for all scenes.
  → Performance vs. Area graph

- More optimizations! Never too many!

Further Ahead

- Remaining (original) two ideas:
  - 2: Bitmap coarse-traversal
    - Traversal is already ludicrously complicated. Well, *very*, anyway.
    - Requires hacking BVH building. Thoughts?
  - 3: Ray rescheduling (req. 2) (was “stretch goal”)
- Still have room for optimization
  - Ray packets
  - Other datastructure (I changed node size)
  - &c. mentioned earlier
- New ideas?

Questions?

Image Credits

- http://bryanwagstaff.com/wp-content/uploads/2013/10/d
- me
- other?