# Implementation of Kd-Trees on the GPU to Achieve Real Time Graphics Processing

Will W. Martin

December 3 2011

University Of Minnesota
Computer Science Senior Seminar

# Ray Tracing

Figure: Taken from [3]

## What Does Ray Tracing Do?

- Creates high quality graphics
- Renders reflections, refraction, shading
- Takes minutes to hours to render single frame on CPU

# Table of Contents

# What is Ray Tracing

## How Does Ray Tracing Work?

- Creating a 3-dimensional scene.
- Shooting "light" rays through the scene.

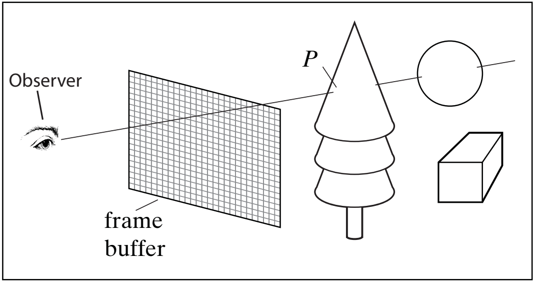



Figure: Taken from [1]

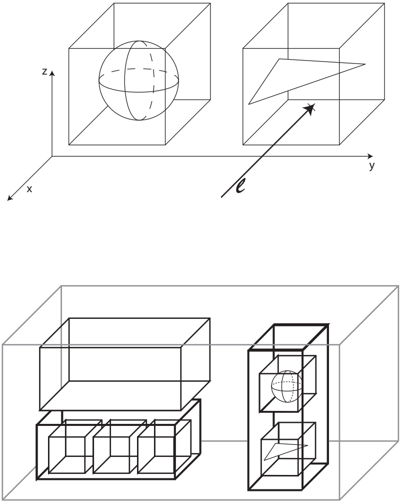



Figure: Based on [2]

## Axes Aligned Bounding Boxes (AABBs)

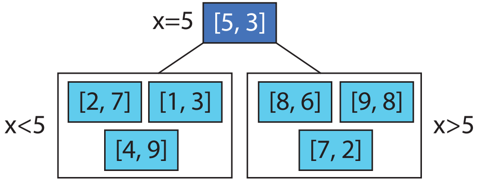
- A rectangular prism surrounding an object
- All faces are axis aligned
- Encloses primitives and other AABBs
- Used to simplify intersection calculations
- Look very similar to graphical representations of kd-trees

# 2-D Kd-Tree

## Tree Structure

- Binary tree sorted on k dimensions.
- Data sorted on x.

| $x$ | $y$ |
|---|---|
| 1 | 3 |
| 2 | 7 |
| 4 | 9 |
| 5 | 3 |
| 7 | 2 |
| 8 | 6 |
| 9 | 8 |

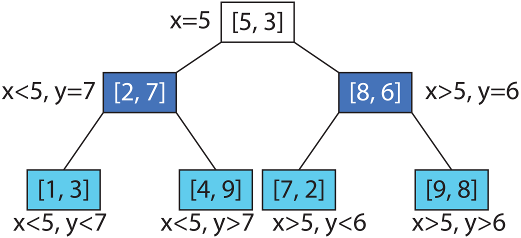


# 2-D Kd-Tree

## Tree Structure

- Binary tree sorted on k dimensions.
- Data sorted on x then y.

| *x* | *y* |
|---|---|
| 1 | 3 |
| 2 | 7 |
| 4 | 9 |
| 5 | 3 |
| 7 | 2 |
| 8 | 6 |
| 9 | 8 |

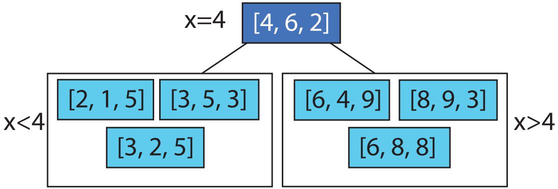


# 3-D Kd-Tree

## Tree Structure

- Binary tree sorted on k dimensions.
- Data sorted on x.

| $x$ | $y$ | $z$ |
|---|---|---|
| 2 | 1 | 5 |
| 3 | 2 | 5 |
| 3 | 5 | 3 |
| 4 | 6 | 2 |
| 6 | 4 | 9 |
| 6 | 8 | 8 |
| 8 | 9 | 3 |

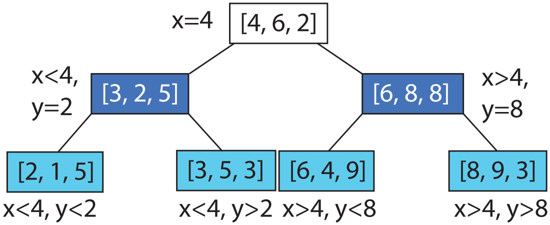


# 3-D Kd-Tree

## Tree Structure

- Binary tree sorted on k dimensions.
- Data sorted on y.

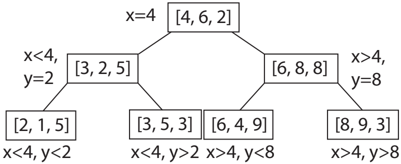
| *x* | *y* | *z* |
|---|---|---|
| 2 | 1 | 5 |
| 3 | 2 | 5 |
| 3 | 5 | 3 |
| 4 | 6 | 2 |
| 6 | 4 | 9 |
| 6 | 8 | 8 |
| 8 | 9 | 3 |

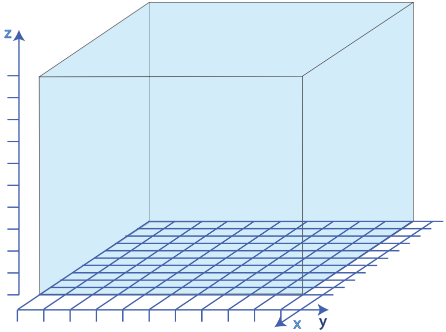


# 3-D Tree (Graphical)

## Split Planes

- Assume this kd-tree fits in a $10 \times 10 \times 10$ volume.
- Each non-leaf node crates a *split plane*

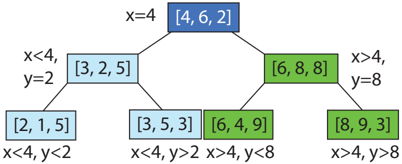


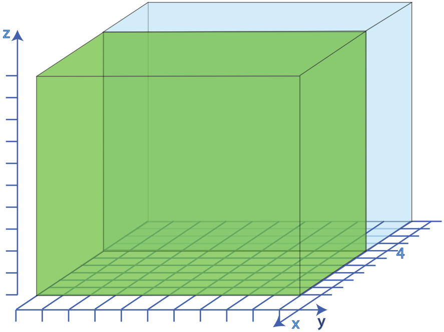



# 3-D Tree (Graphical)

## Split Planes

- Assume this kd-tree fits in a $10 \times 10 \times 10$ area.
- The root node splits on x=4

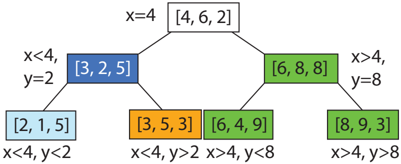


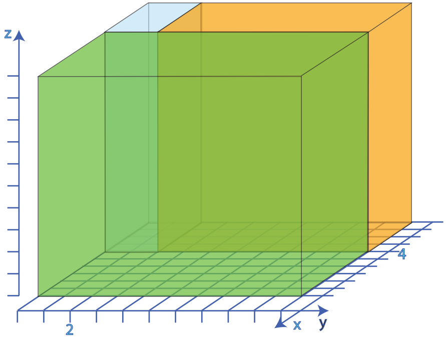



# 3-D Tree (Graphical)

## Split Planes

- Assume this kd-tree fits in a $10 \times 10 \times 10$ area.
- The right child splits on y=2

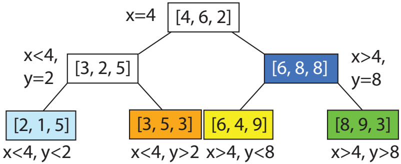


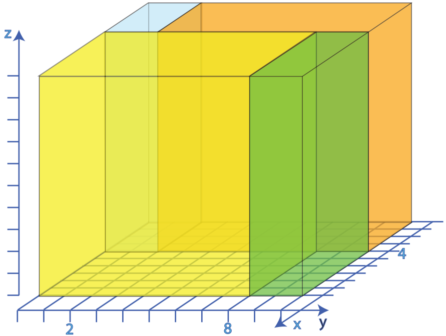



# 3-D Tree (Graphical)

## Split Planes

- Assume this kd-tree fits in a 10 × 10 × 10 area.
- The left child splits on y=8



# Kd-trees constructed for ray tracing

## Specialized kd-trees

- All non-leaf nodes are used for sorting.
- Kd-trees for ray tracing store all graphics data in leaf nodes.
- Heuristics are used to maximize the efficiency of kd-trees for ray tracing

# Heuristics

## Heuristics

- Experienced based technique for problem solving
- Used to narrow search spaces when exhaustive searches are impractical

## Goals

- Minimize surface area
- A node split into 2 nodes of minimal surface area will be balanced
- Split nodes into cubes

# Surface Area vs Intersections

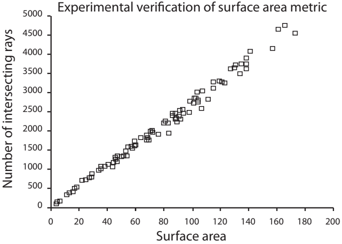



Figure: Taken from [4]

## Surface Area vs. Ray Hits

- Left: (Taken from [4]) Shows number of intersections to surface area of bounding box
- Strong linear relationship

# SAH Approximation

$$SAH[x] = C_{ts} + \frac{C_L[x]SA_L[x]}{SA_{\text{parent}}} + \frac{C_R[x]SA_R[x]}{SA_{\text{parent}}} \quad (1)$$

## Terms

- Used in [7]
- $C_{ts}$ - Cost of traversing a node
- $C_L$ and $C_R$ - Cost of left and right child
- $SA_L$ and $SA_R$ - Surface area of left and right child
- $SA_{\text{parent}}$ - Total surface area of node being split

# Empty Space Minimizing

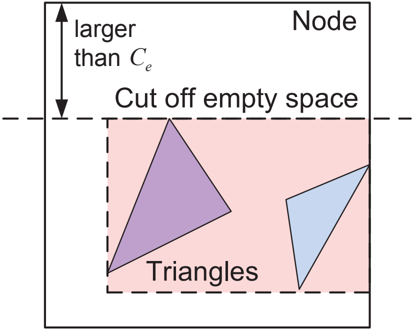



Figure: Taken from [8]

## Surface Area vs. Ray Hits

- Left: (Taken from [8]) shows empty space being cut off a node
- Splits the node to cut off empty space
- Requires a piece of the node larger than $C_e$ to be empty

# Median Split

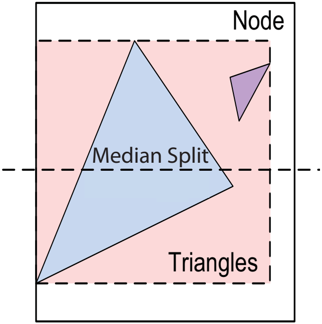



Figure: Taken from [8]

## Surface Area vs. Ray Hits

- Left: (Taken from [8]) shows a node being split along its longest axis
- Splits the node arbitrarily along its longest axis

# Median Split

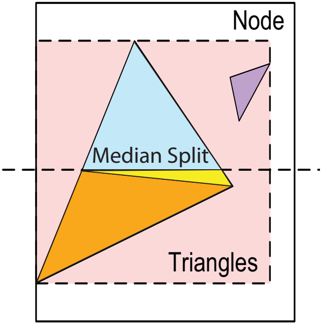



Figure: Taken from [8]

## Surface Area vs. Ray Hits

- Left: (Taken from [8]) shows a node being split along its longest axis
- Splits the node arbitrarily along its longest axis

# Using heuristics effectively

## Construction

- Generate AABBs for all primitives
- Put all graphics primitives in root node
- Classify all nodes with over 64 primitives as large
- Classify all nodes with 64 or less primitives as small

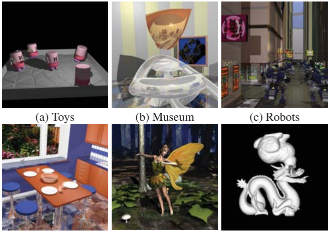
## Small Nodes

- Split candidates at all primitive's AABBs
- Run SAH approximation on all split candidates
- Split on lowest cost candidate
- Filter graphics primitives down to new children
- Overlapping primitives are brought down to both new children

## Large Nodes

- If applicable use empty space minimizing
- Else use median split
- Filter graphics primitives down to new children
- Overlapping primitives are clipped

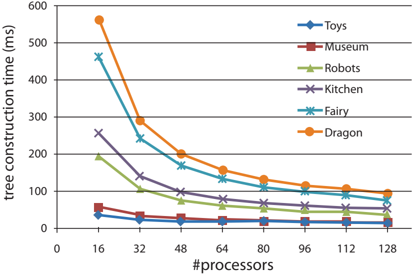
# Results



(a) Toys (b) Museum (c) Robots (d) Kitchen (e) Fairy Forest (f) Dragon

## Test Scenes From [8]

(a) 11K triangles, 1 light
(b) 27K triangles, 2 lights
(c) 71K triangles, 3 lights
(d) 111K triangles, 6 lights
(e) 178K triangles, 2 lights
(f) 252K triangles, 1 light




## Scaling From [8]

- Scales well in the beginning
- Tends to taper off toward 80 processors

# Results

## Performance

| Scene | [6] | [5] | GPU builder |
|---|---|---|---|
| (a) | 10.5fps | 23.5fps | 32.0fps |
| (b) | n/a | n/a | 8.00fps |
| (c) | n/a | n/a | 4.96fps |
| (d) | n/a | n/a | 4.84fps |
| (e) | 2.30fps | 5.84fps | 6.40fps |
| (f) | n/a | n/a | 8.85fps |

- [6] used AMD Opteron 2.6GHz CPU
- [5] used Dual Intel Core2 Duo 3.0GHz CPU

# Conclusion

## Conclusion

- GPU builders are faster than CPU builders
- GPU builders still need to get faster
- GPU builders show promise

F. S. Hill, Jr.
*Computer Graphics Using Open GL*.
Macmillan Publishing Company, second edition.

J. Goldsmith and J. Salmon.
Automatic creation of object hierarchies for ray tracing.
*Computer Graphics and Applications, IEEE*, 7(5):14 –20, May 1987.

B. Grass.
X3D class examples of ray tracing.
https://www.movesinstitute.org/pipermail/x3d-courses/attachments/20110428/0943b920/attachment-0004.jpg, Apr. 2011.

J. D. MacDonald and K. S. Booth.
Heuristics for ray tracing using space subdivision.
*The Visual Computer*, 6:153–166, 1990.
10.1007/BF01911006.

- M. Shevtsov, A. Soupikov, and A. Kapustin.
  Highly parallel fast kd-tree construction for interactive ray tracing of dynamic scenes.
  *Computer Graphics Forum*, 26(3):395–404, 2007.

- I. Wald, S. Boulos, and P. Shirley.
  Ray tracing deformable scenes using dynamic bounding volume hierarchies.
  *ACM Trans. Graph.*, 26, January 2007.

- I. Wald and V. Havran.
  On building fast kd-trees for ray tracing, and on doing that in O(n log(n)).
  In *Interactive Ray Tracing 2006, IEEE Symposium on*, pages 61 –69, Sept. 2006.

- K. Zhou, Q. Hou, R. Wang, and B. Guo.
  Real-time kd-tree construction on graphics hardware.
  *ACM Trans. Graph.*, 27:126:1–126:11, December 2008.