1

CS-184: Computer Graphics

Lecture #7: BSP and AABB Trees

Sean Arietta
University of California, Berkeley

V2014-S-07-1.0

Slides Adapted from: Prof. James O'Brien

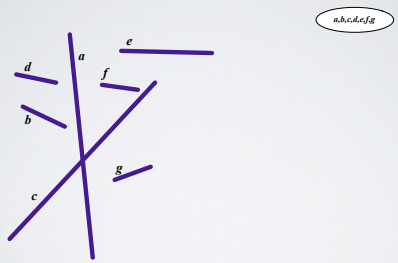
2

BSP-Trees

- Binary Space Partition Trees
  - Split space along planes
  - Allows fast queries of some spatial relations
- Simple construction algorithm
  - Select a plane as sub-tree root
  - Everything on one side to one child
  - Everything on the other side to other child
  - Use random polygon for splitting plane

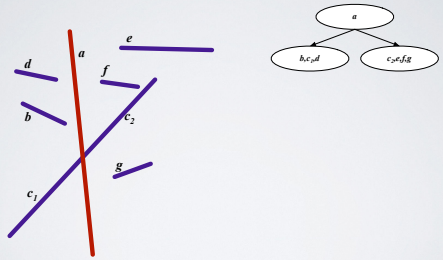
BSP-Trees

3



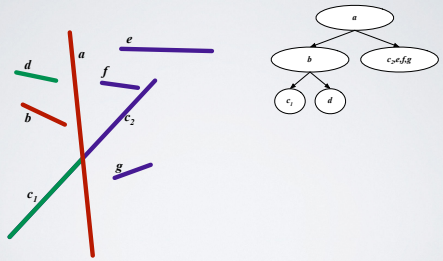
BSP-Trees

4



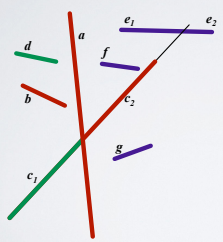
BSP-Trees

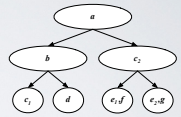
5



BSP-Trees

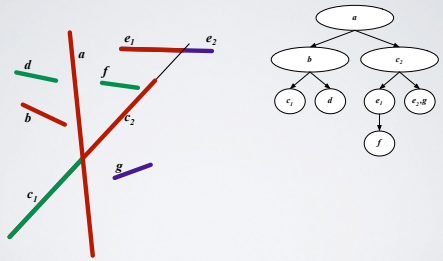
6





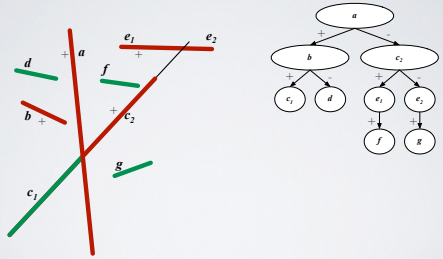
BSP-Trees

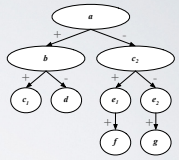
7



BSP-Trees

8





BSP-Trees

- Visibility Traversal
  - Variation of in-order-traversal
    - Child one
    - Sub-tree root
    - Child two
  - Select "child one" based on location of viewpoint
    - Child one on same side of sub-tree root as viewpoint

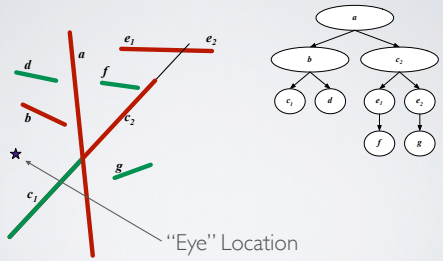
9

BSP-Trees

Let's consider an example…

10

BSP-Trees



11

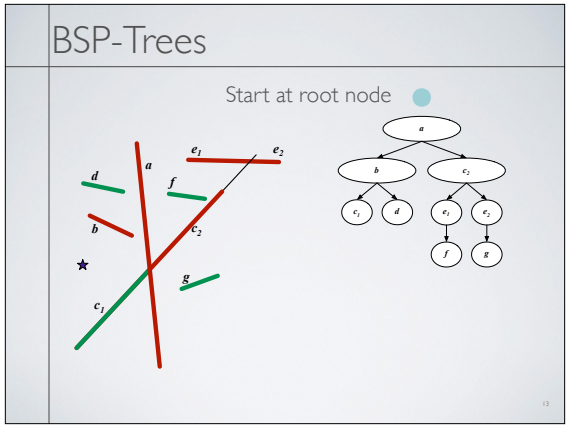
BSP-Trees

A Simple Mechanism:

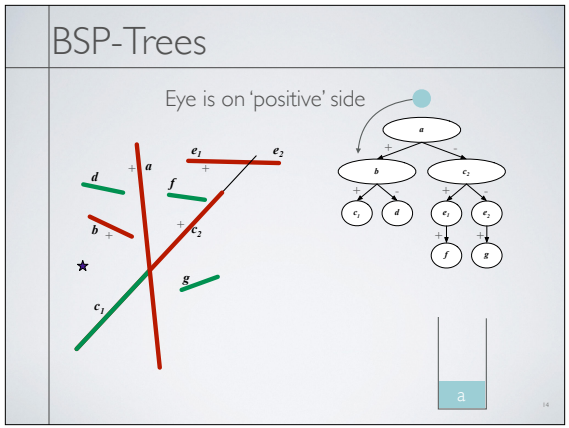
Perform depth-first search and save nodes when we pop them off the search stack

There are other ways to understand the traversal scheme, but I prefer this one due to its simplicity.

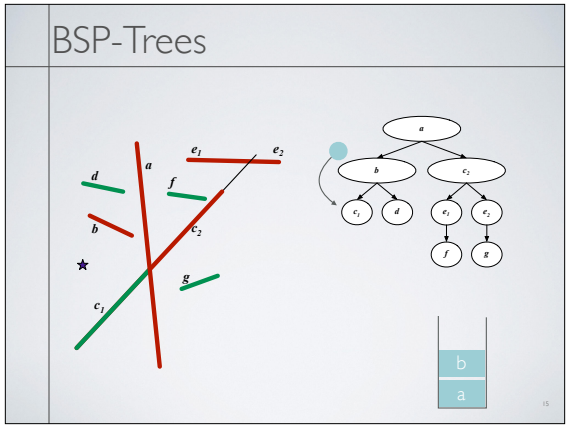
12


BSP-Trees
Start at root node
a
b
c2
c1
d
e1
e2
f
g

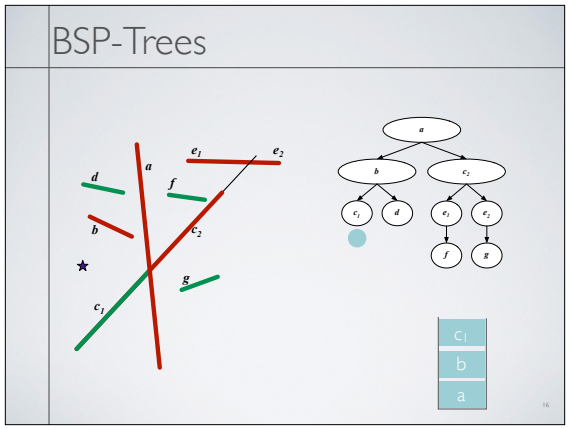
13


BSP-Trees
Eye is on 'positive' side
a
b
c2
c1
d
e1
e2
f
g
a

14


BSP-Trees
a
b
c2
c1
d
e1
e2
f
g
15

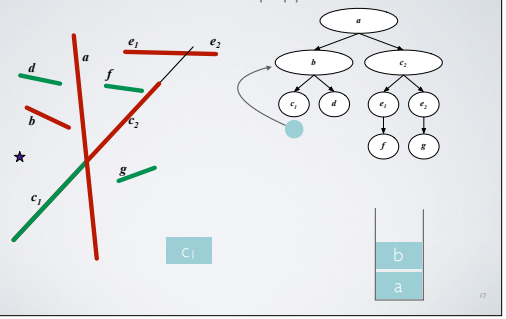
15


BSP-Trees
a
b
c2
c1
d
e1
e2
f
g
c1
b
a
16

16

BSP-Trees

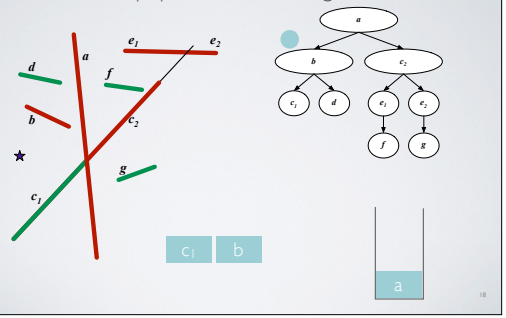
Save nodes in order when popped



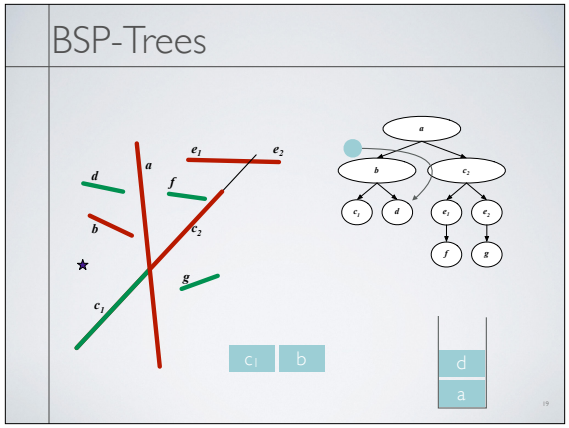
17

BSP-Trees

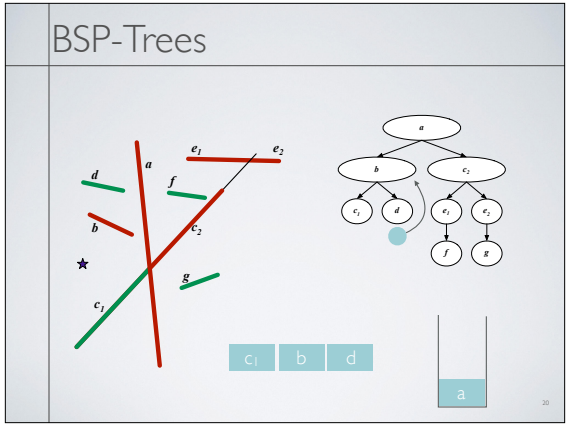
Remember to 'pop' before traversing children



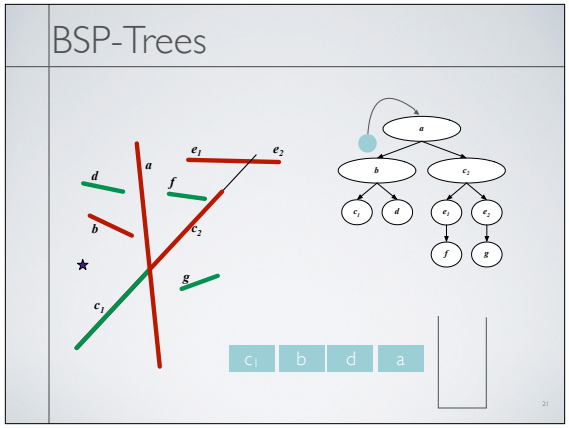
18


BSP-Trees
a
e1
e2
d
f
b
c2
g
c1
c1
b
d
a
19

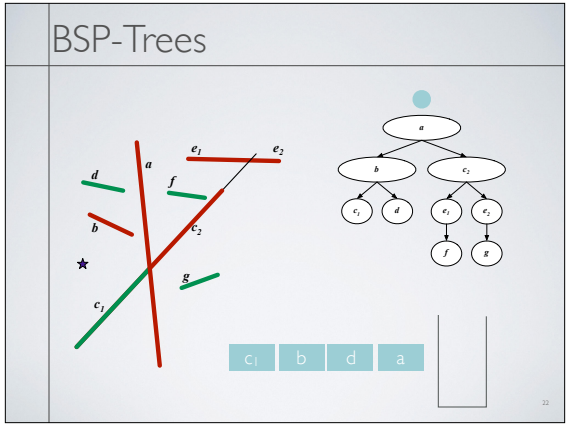
19


BSP-Trees
a
e1
e2
d
f
b
c2
g
c1
c1
b
d
a
20

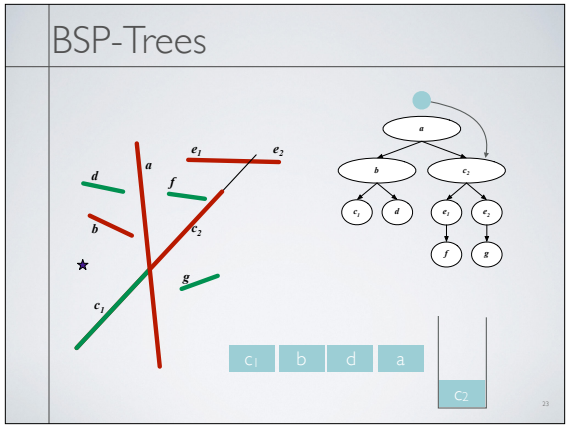
20


BSP-Trees
a
e1
e2
d
f
b
c2
g
c1
c1
b
d
a

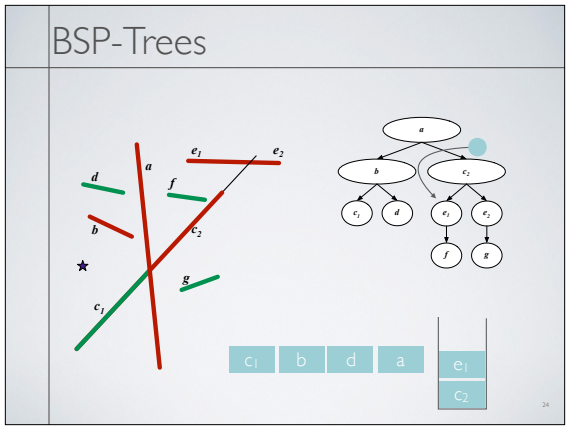
21


BSP-Trees
a
e1
e2
d
f
b
c2
g
c1
c1
b
d
a

22



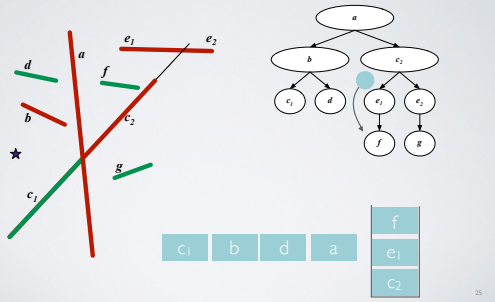
23



24

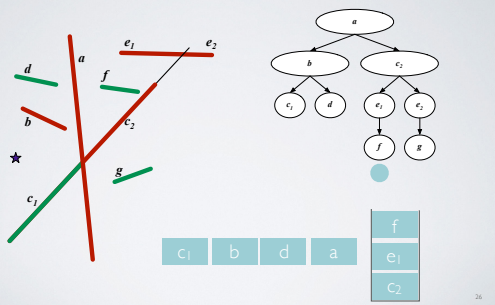
BSP-Trees

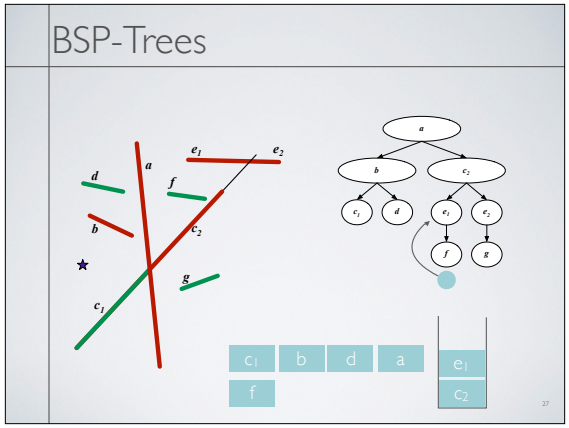
25



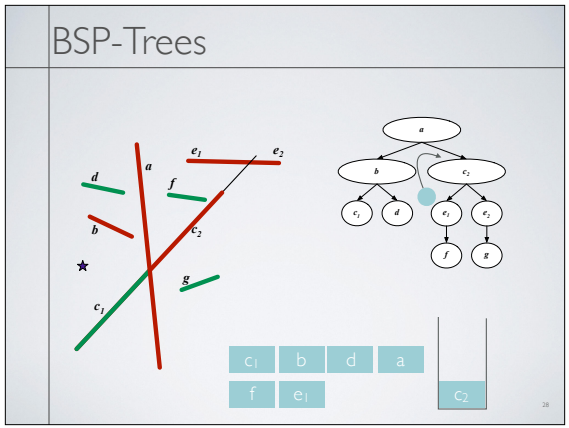
BSP-Trees

26

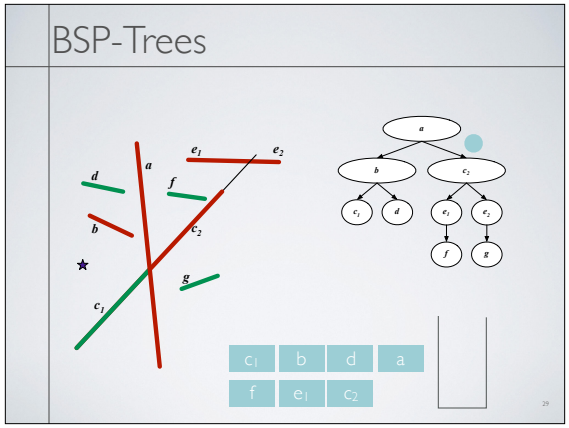




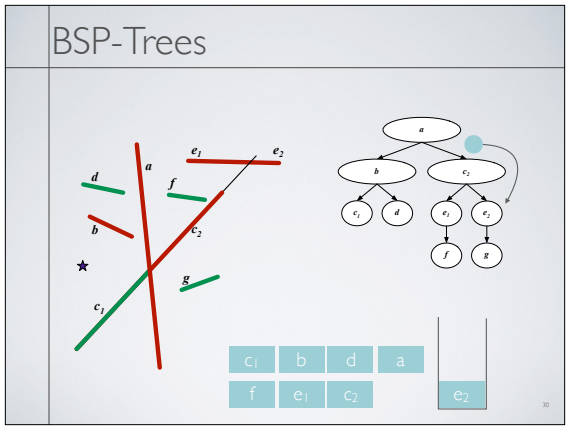
27



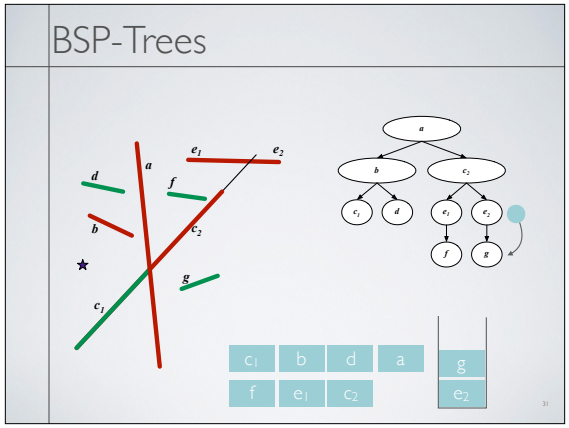
28



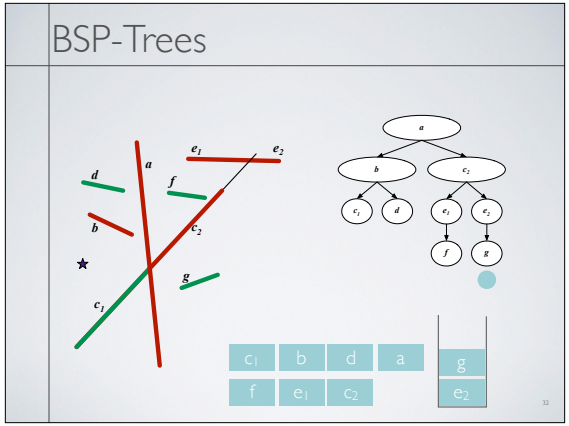
29



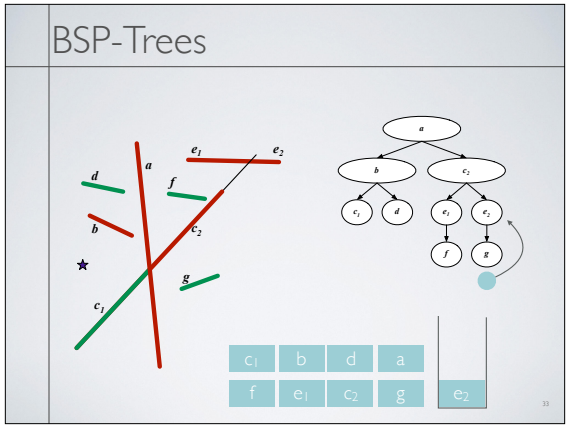
30


BSP-Trees
a
e1
e2
d
f
b
c2
g
c1
c1 b d a
f e1 c2
g
e2

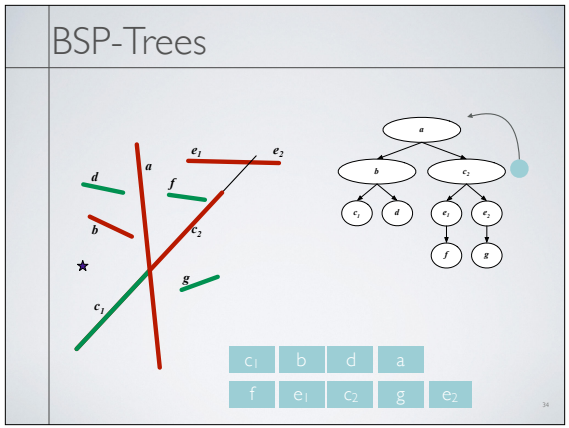
31


BSP-Trees
a
e1
e2
d
f
b
c2
g
c1
c1 b d a
f e1 c2
g
e2

32


BSP-Trees
a
e1
e2
d
f
b
c2
g
c1
c1 b d a
f e1 c2 g
e2
33

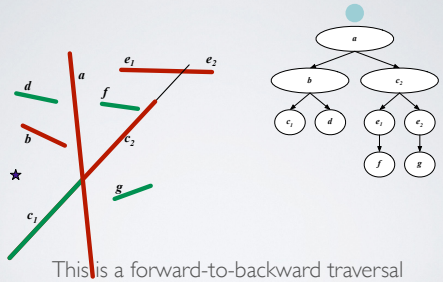
33


BSP-Trees
a
e1
e2
d
f
b
c2
g
c1
c1 b d a
f e1 c2 g e2
34

34

BSP-Trees

35



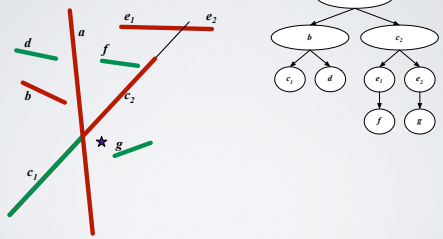
This is a forward-to-backward traversal

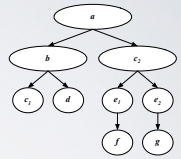
c1 b d a f e1 c2 g e2

35

BSP-Trees

36





g e2 c2 f e1 a c1 b d

36

37

On to another useful space partition…

38

Your Ray Tracer

```
RayTrace(image)
   For ray in camera
     image[pixel] = Trace(ray)
Trace(ray)
     t_hit = infinity
     For object in scene
       t_hit = min(object.intersect(ray), t_hit)
     shade at t_hit
     possible calls to Trace(new_ray)
```

Your Ray Tracer

39

```
RayTrace(image)
   For ray in camera
     image[pixel] = Trace(ray)
Trace(ray)
     t_hit = infinity
     For object in scene
       t_hit = min(object.intersect(ray), t_hit)
     shade at t_hit
     possible calls to Trace(new_ray)
```

Your Ray Tracer

40

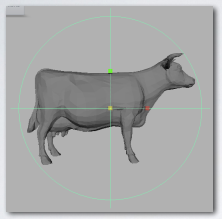
```
RayTrace(image)
   For ray in camera
     image[pixel] = Trace(ray)
Trace(ray)
     t_hit = infinity
     For object in scene
       t_hit = min(object.intersect(ray), t_hit)
     shade at t_hit
     possible calls to Trace(new_ray)
```

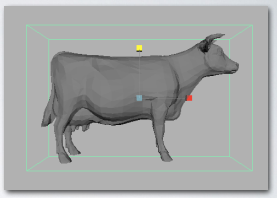
41

That's a lot of monkeys to test

42

Bounding Shapes

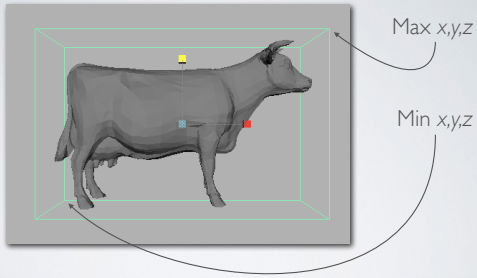




- Bounding shape completely encloses associated object
- Rays cannot hit object w/o intersecting bounding shape
- Two objects cannot collide if shapes don't overlap
- Tradeoff: Simplicity -vs- tightness

Axis-Aligned Bounding Boxes

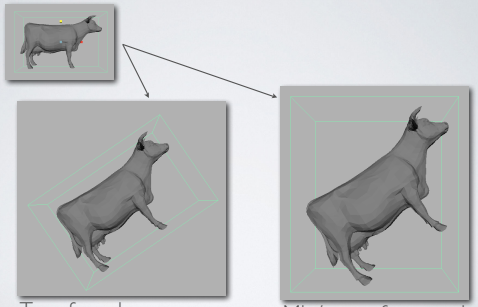
43



- Axis-aligned bounding box defined by min and max *x,y,z*

Axis-Aligned Bounding Boxes

44

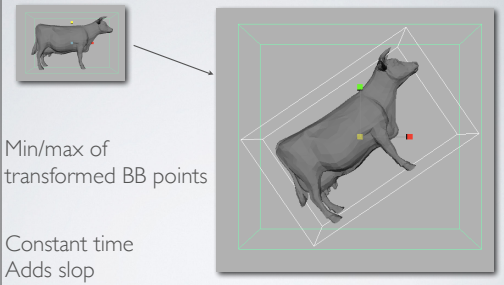


Transform box
Not axis-aligned

Min/max of new points
Linear cost to compute

Axis-Aligned Bounding Boxes

45



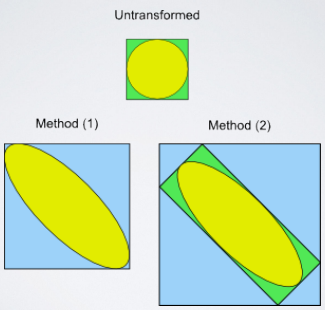
Min/max of
transformed BB points

Constant time
Adds slop
Cumulative slop if multiple transforms occur sequentially

Why would we do this?

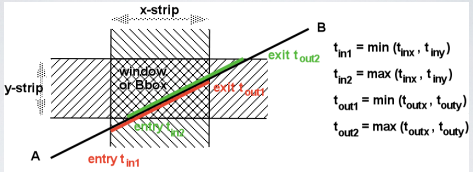
Tightness

46



Box Intersection

47



$t_{in1} = \min(t_{inx}, t_{iny})$

$t_{in2} = \max(t_{inx}, t_{iny})$

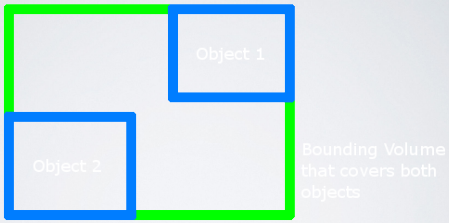
$t_{out1} = \min(t_{outx}, t_{outy})$

$t_{out2} = \max(t_{outx}, t_{outy})$

Line goes through window/Bbox if intervals overlap

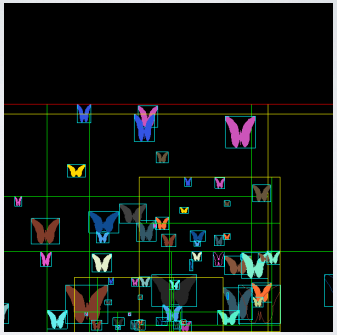
Bounding Box of Bounding Boxes

48



AABB Trees

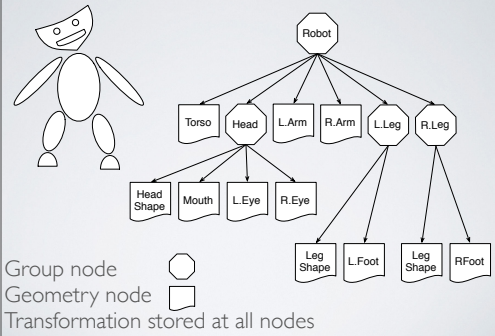
49



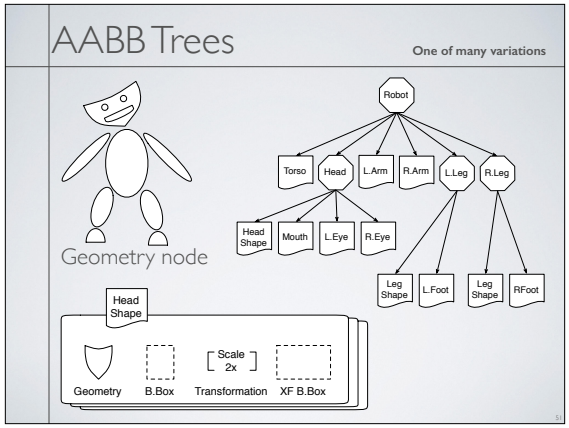
AABB Trees

One of many variations

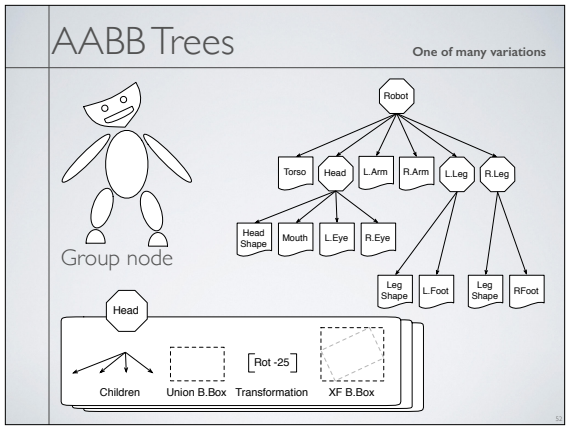
50



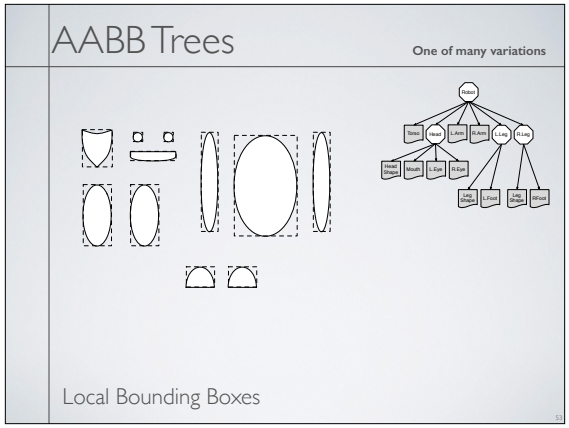
Group node
Geometry node
Transformation stored at all nodes


AABB Trees
One of many variations
Robot
Torso
Head
L.Arm
R.Arm
L.Leg
R.Leg
Head Shape
Mouth
L.Eye
R.Eye
Leg Shape
L.Foot
Leg Shape
RFoot
Geometry node
Head Shape
Scale 2x
Geometry
B.Box
Transformation
XF B.Box

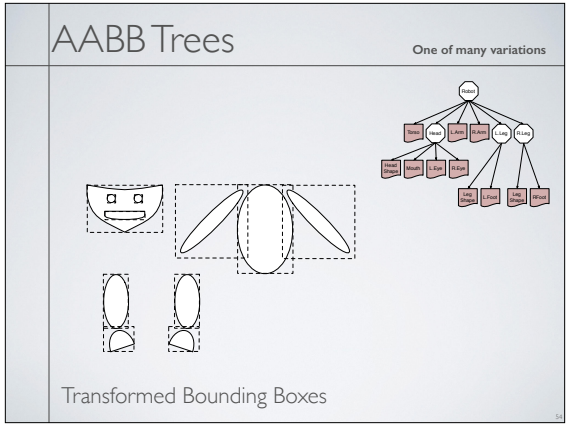
51


AABB Trees
One of many variations
Robot
Torso
Head
L.Arm
R.Arm
L.Leg
R.Leg
Head Shape
Mouth
L.Eye
R.Eye
Leg Shape
L.Foot
Leg Shape
RFoot
Group node
Head
Rot -25
Children
Union B.Box
Transformation
XF B.Box

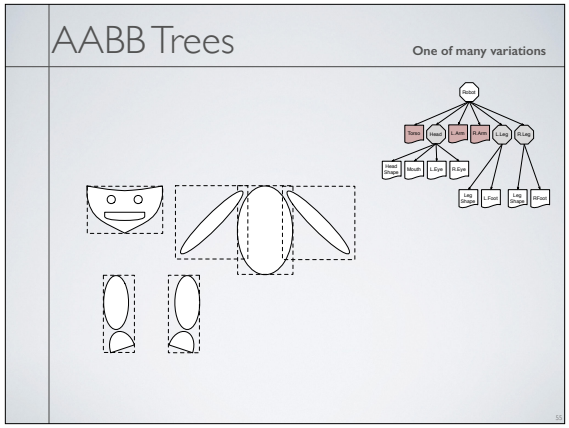
52


AABB Trees
One of many variations
Robot
Torso
Head
L Arm
R Arm
L Leg
R Leg
Head Shape
Mouth
L Eye
R Eye
Leg Shape
L Foot
Leg Shape
R Foot
Local Bounding Boxes
53

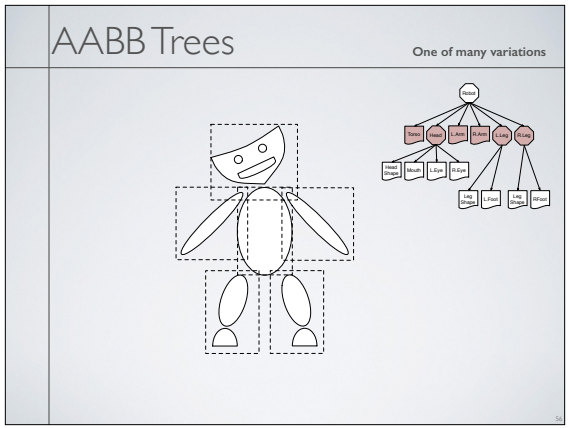
53


AABB Trees
One of many variations
Robot
Torso
Head
L Arm
R Arm
L Leg
R Leg
Head Shape
Mouth
L Eye
R Eye
Leg Shape
L Foot
Leg Shape
R Foot
Transformed Bounding Boxes
54

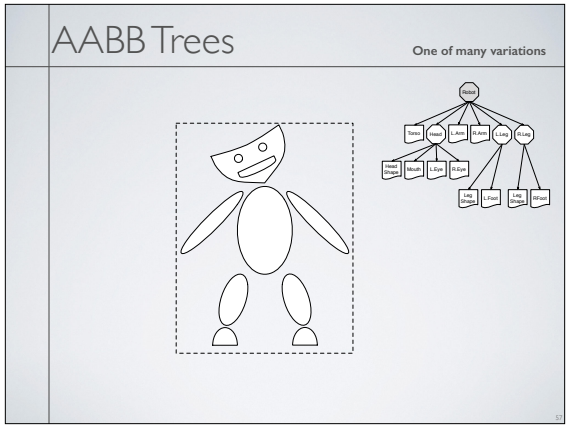
54


AABB Trees
One of many variations
Robot
Torso
Head
L-Arm
R-Arm
L-Leg
R-Leg
Head Shape
Mouth
L-Eye
R-Eye
Leg Shape
L-Foot
Leg Shape
R-Foot
55

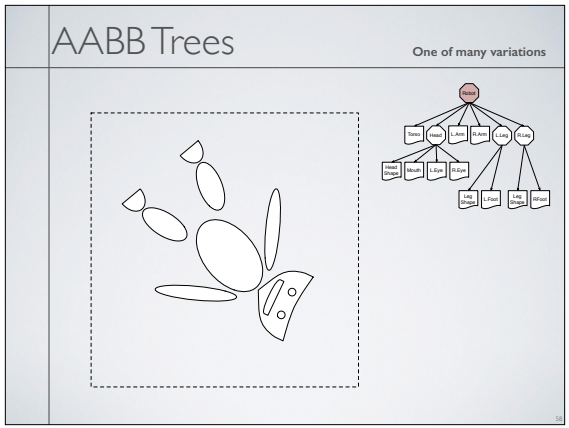
55


AABB Trees
One of many variations
Robot
Torso
Head
L-Arm
R-Arm
L-Leg
R-Leg
Head Shape
Mouth
L-Eye
R-Eye
Leg Shape
L-Foot
Leg Shape
R-Foot
56

56


AABB Trees
One of many variations
Robot
Torso
Head
L-Arm
R-Arm
L-Leg
R-Leg
Head Shape
Mouth
L-Eye
R-Eye
Leg Shape
L-Foot
Leg Shape
RFoot
57

57


AABB Trees
One of many variations
Robot
Torso
Head
L-Arm
R-Arm
L-Leg
R-Leg
Head Shape
Mouth
L-Eye
R-Eye
Leg Shape
L-Foot
Leg Shape
RFoot
58

58

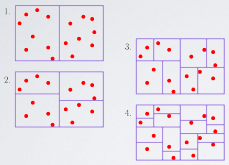
Ray Test Against Bound Tree

59

- `RayHitSubTree(&ray,node)`
  - `If RayHitsBB(ray,node.xfBB)`
    - `ixfRay = Inverse(node.xf)*ray`
    - `If RayHitsBB(ixfRay,node.BB)`
      - `If node is group`
        - `Foreach child in node.children`
          - `RayHitSubTree(ixfRay,child)`
      - `else // node not group`
        - `RayHitGeometry(ixfRay,node.geom)`
    - `ray.collisionInfo.update(ixfRay)`

Building the tree

60



- Sort (or QuickSelect) and split on one axis
- Repeat for the other axis
  - x,y,z

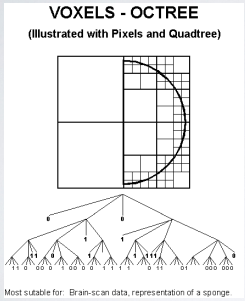
Other Schemes

61

- Uniform Grid/Octrees
- Spatial Hierarchies
- Etc

Voxels/Octree

62



- Fun Links
- http://www.youtube.com/watch?v=6x8fYNFZsDQ
- http://www.youtube.com/watch?v=JI54WZtm0QE