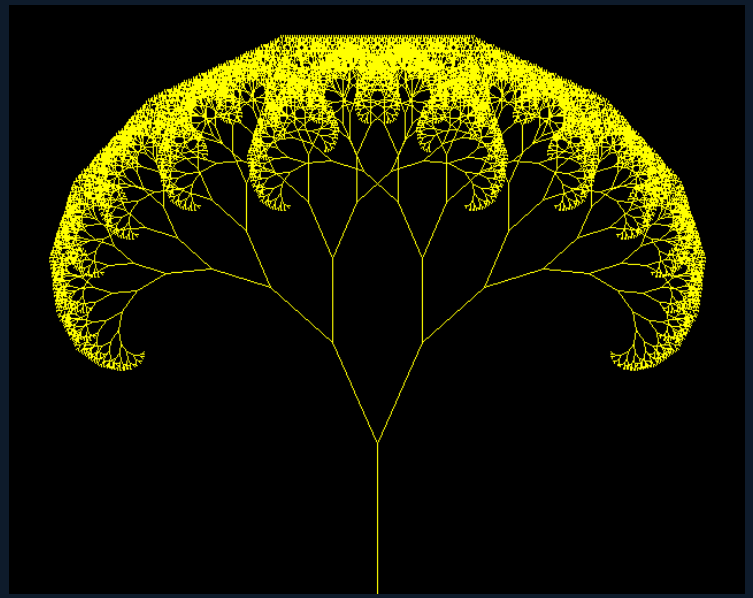
Specialty Trees

Red-Black Trees and Spatial Trees

Contents

- Types of Trees
- Red Black Trees
- Spatial Trees
  - Quadtrees
  - BSP Trees
  - K-D Trees
  - Octrees
- Summary
- References



Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

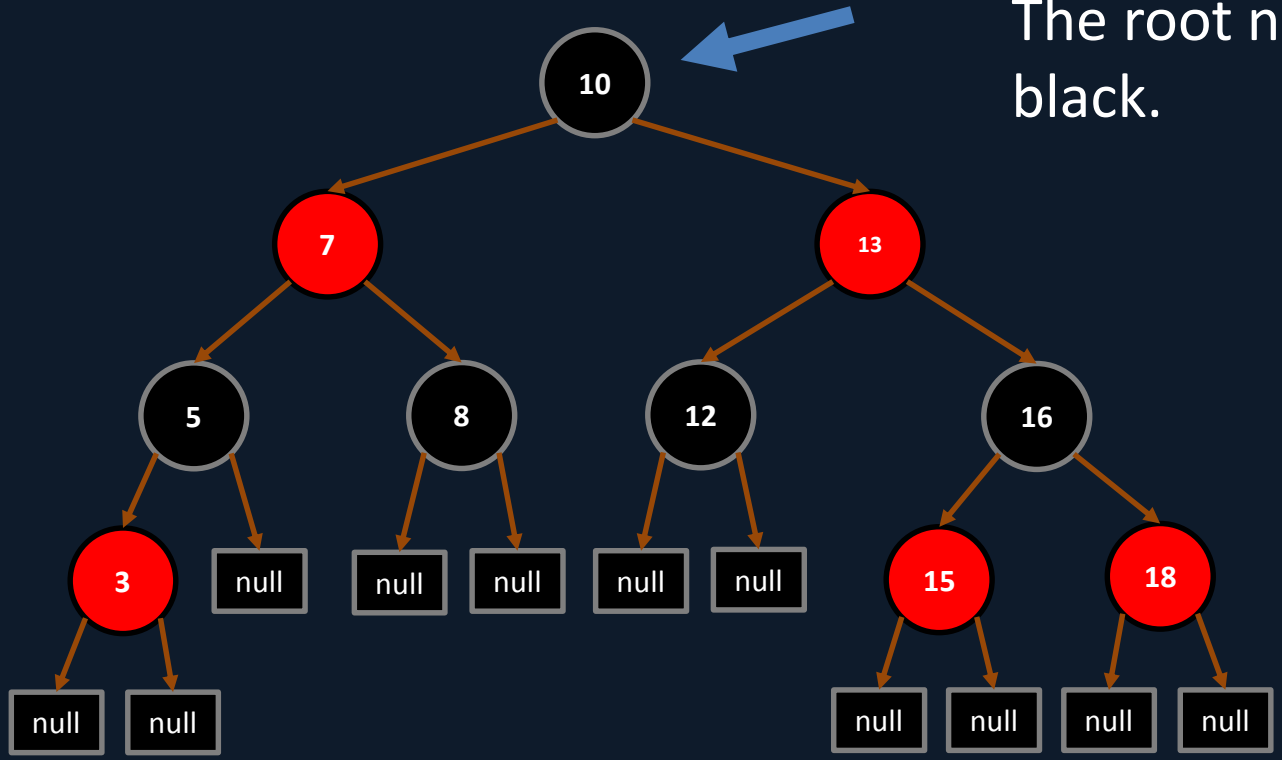
Types of Trees

- There are a lot of different types of trees and they are all suited to different tasks.
  - 2-3 Trees – Each node has either 1 value and 2 branches or 2 values and 3 branches.
  - (a,b) Trees – Where all leaves are at the same depth.
  - B Trees – Each node has 2 or more children, often used in databases and filesystems.
  - Dancing Trees – A form of self-balancing tree.
  - Interval Trees – Each node contains a range of numbers. (e.g. 3 to 7)
  - Cartesian Trees – The root contains the lowest value and nodes are arranged in the order they are added.
  - Splay Trees – A binary tree where the most recently accessed nodes are the fastest to access again.
  - …and many more

Red-Black Trees

- Red-Black trees are a self-balancing form of binary tree.
- The aim is to minimise how unbalanced the tree can become and so make it more efficient to search.
- They are constructed like normal binary trees, except that each node contains an additional piece of data: its ‘colour’, usually described as ‘red’ or ‘black’.
- We use the node’s colour to help rebalance the tree.
- Red-Black trees are not perfectly balanced but usually considered “good enough”.

Red-Black Trees - Rules

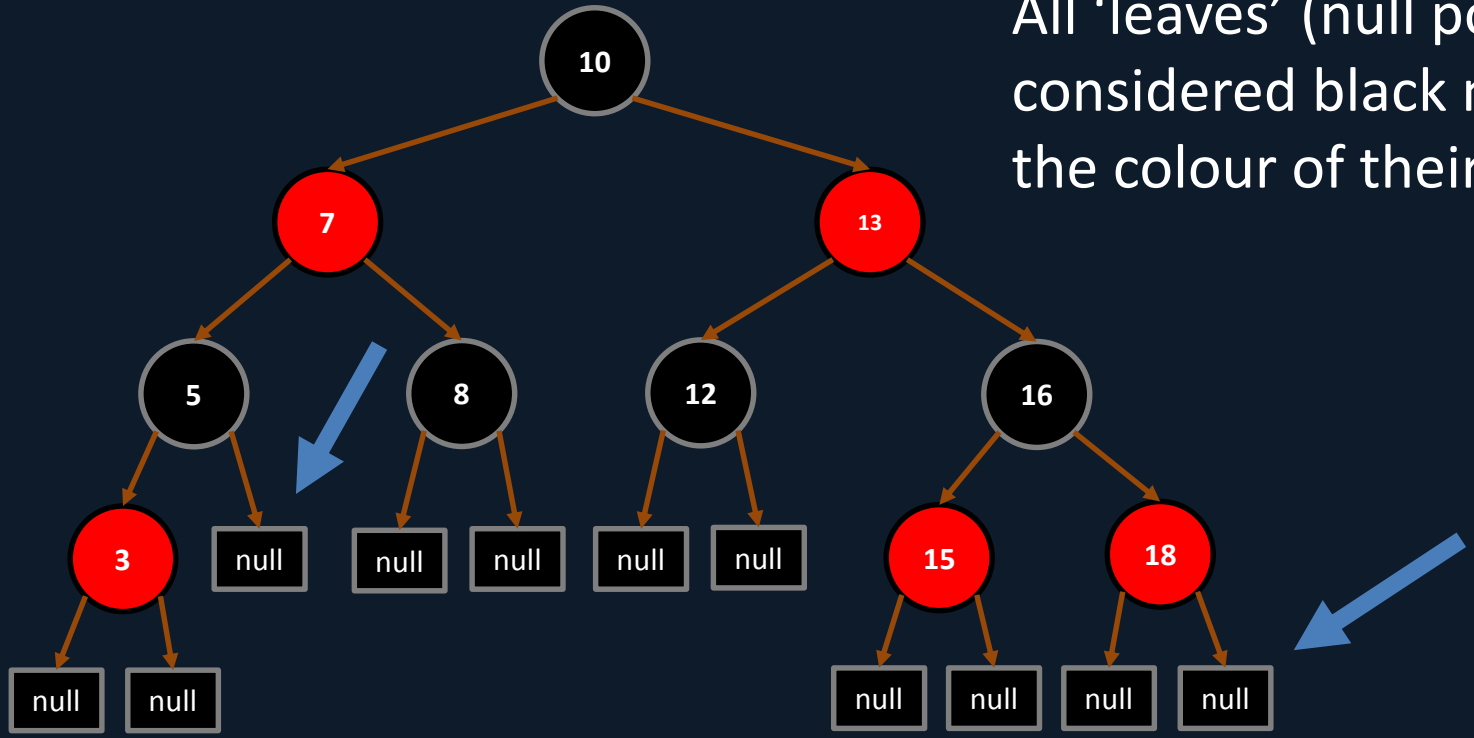


The root node is always black.

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Red-Black Trees - Rules

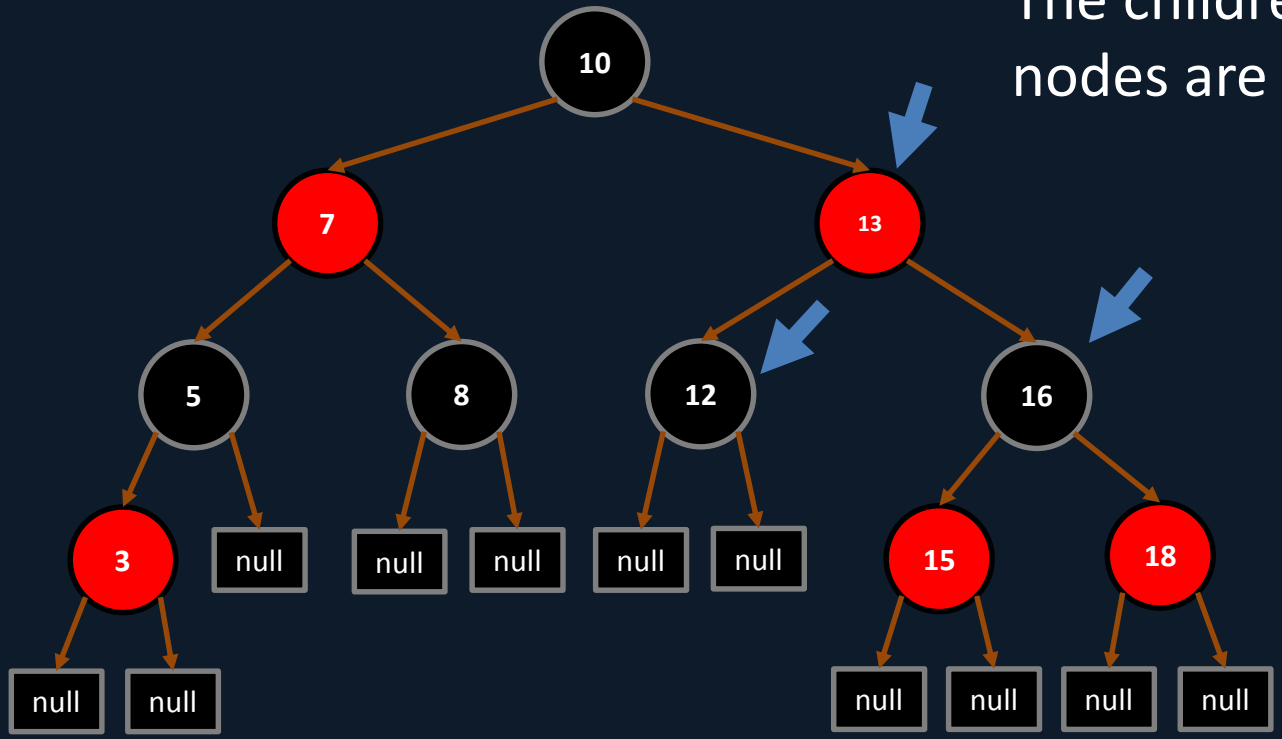


All ‘leaves’ (null pointers) are considered black no matter the colour of their parent.

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Red-Black Trees - Rules

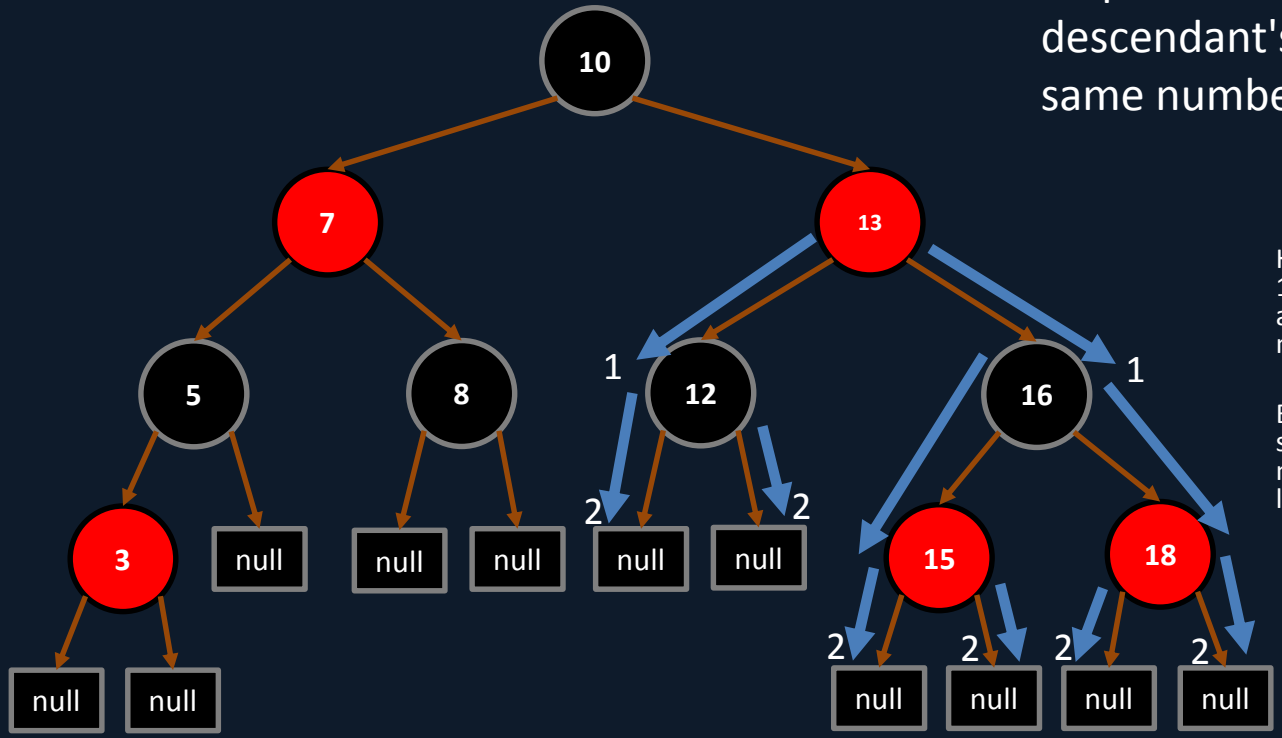


The children of red nodes are always black.

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Red-Black Trees - Rules



All paths from a node to its descendant's leaves contain the same number of black nodes.

Here all the paths from the 13 node are shown in blue and the number of black nodes counted.

Each path has exactly the same number of black nodes: 2 (counting the null leaf as a node).

Canberra Institute of Technology

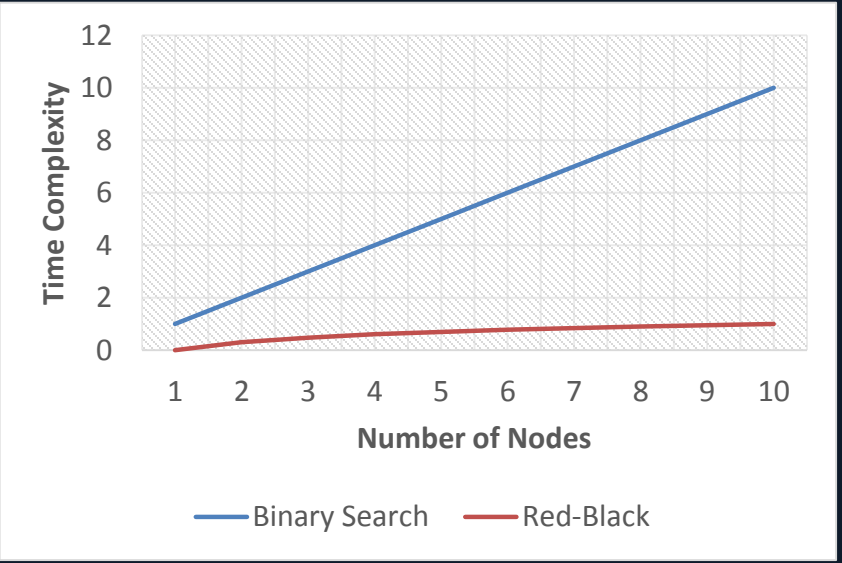
aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Balance

- These properties keep the Red-Black tree roughly balanced since:
  - No path can ever have two red nodes in a row.
  - No path can have more black nodes than any other path.
  - Which means at most a path can only be twice as long as another path
    - e.g. a path of 2 black nodes compared to a path of 2 black nodes and 2 red nodes.

Performance

- Significantly more efficient than a normal binary search tree:
  - Binary Tree Search: O(n)
  - Red-Black Tree Search: O(log(n))



Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Insertion

- New nodes are inserted just like in a binary tree except that the node starts red.

- Once the node has been inserted, nodes can be repositioned or recoloured to keep the rules and the balance of the tree intact.
  - This be beyond the scope of this lesson.
  - We can watch it here:
    - https://www.cs.usfca.edu/~galles/visualization/RedBlack.html

Canberra Institute of Technology

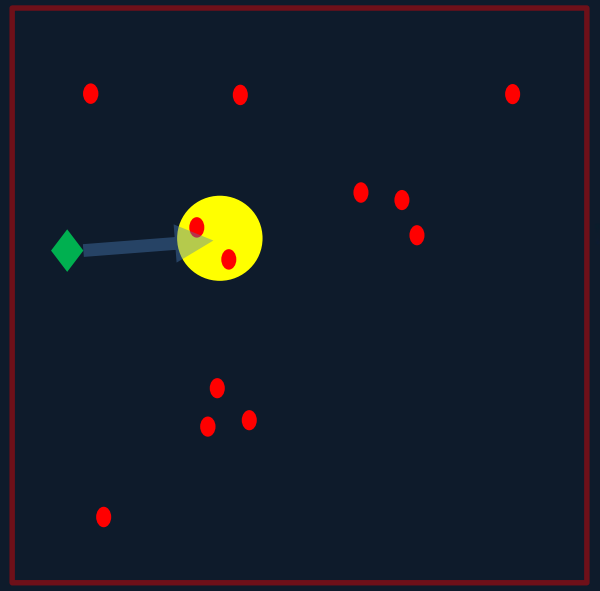
aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Spatial Trees

- Spatial Trees are used to describe physical space in a tree structure.
- Used for physics calculations, AI, rendering, or even just to work out how close two objects are to one another.
- Very useful in games: Imagine the player has just thrown a grenade and we need to damage all enemies caught within its blast.
  - We could test against every enemy in the level to see how far away they are but this could be very expensive if there are a lot of enemies.
  - A spatial tree allows us to narrow down our search first to just those enemies that are roughly in the same area as the explosion, and then perform more precise checks against them.

The problem…

- The green diamond throws a grenade (yellow circle).
- How many of the red enemies are inside the blast?
- We could check all of them but that's expensive.



Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Spatial Trees

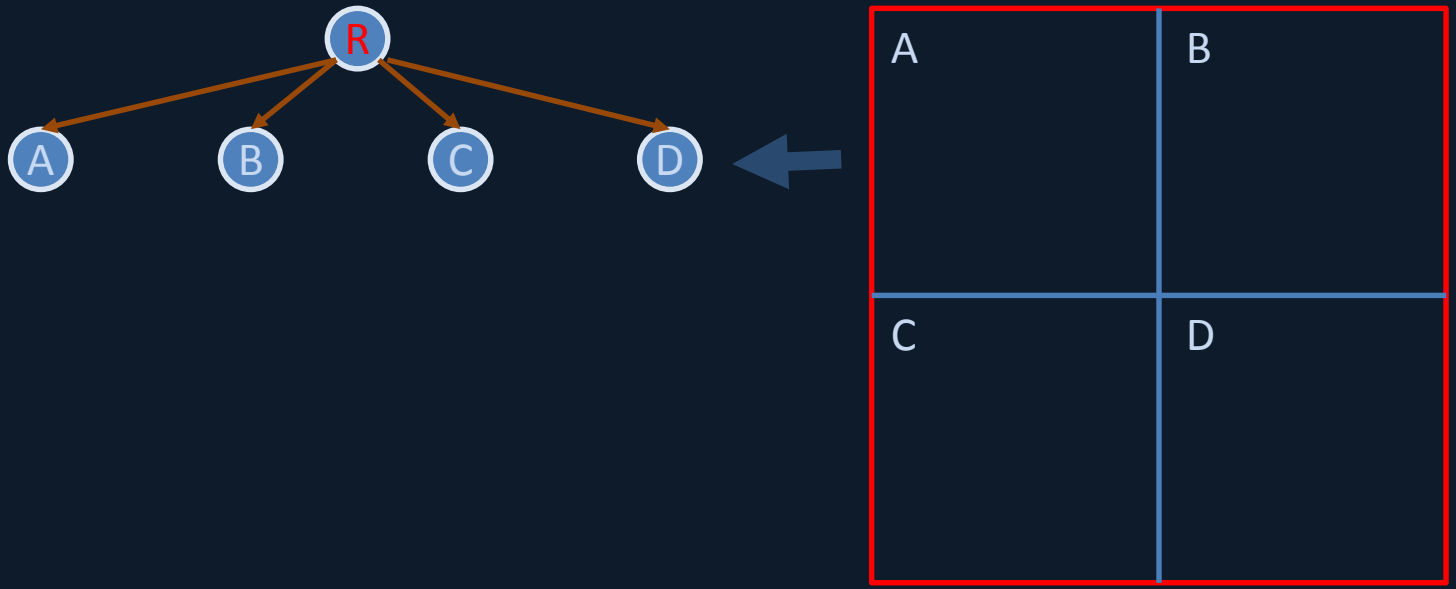
- The most common types of spatial trees are:
  - Quadtrees
    - Fairly easy to implement and good for top-down or 2D games. (e.g. RTS games, retro RPGs, etc)
  - BSP (Binary Space Partition) Trees
    - Used in a lot of old famous FPS games (Unreal, Quake, Half-Life, Portal, etc). Not used as much now.
  - Octrees
    - Good for when objects are clustered in 3D space (e.g. space games, games where height is as important as width and length)
  - K-D Trees
    - A hybrid of BSP and Quad trees.

Quadtrees

- First we treat the level as a giant square and add that to our tree as the root node

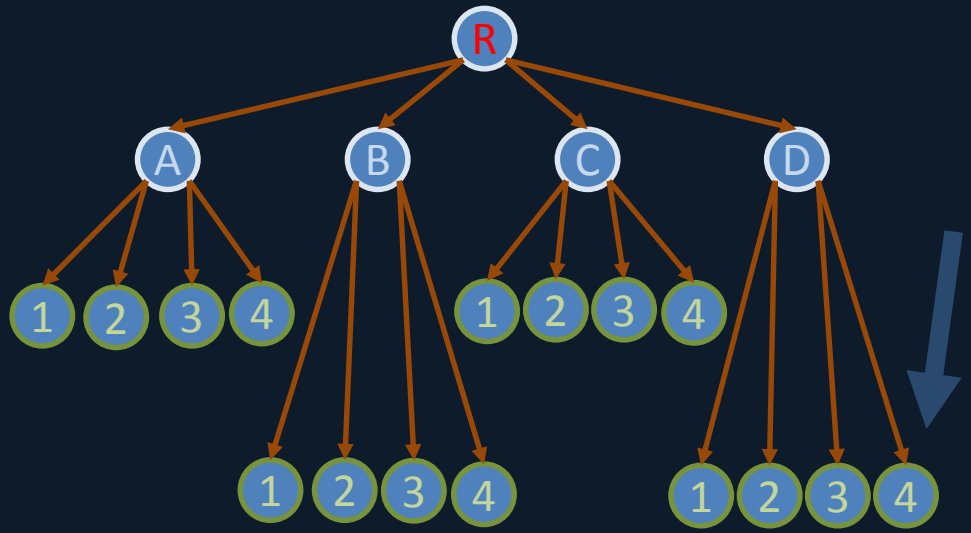
Quadtrees

- Then we divide the level into four large squares (quads), and each of these is added as a child of the root.



Quadtrees

- Then we divide each quad into four more squares, and each of these is added as a child of its parent quad.



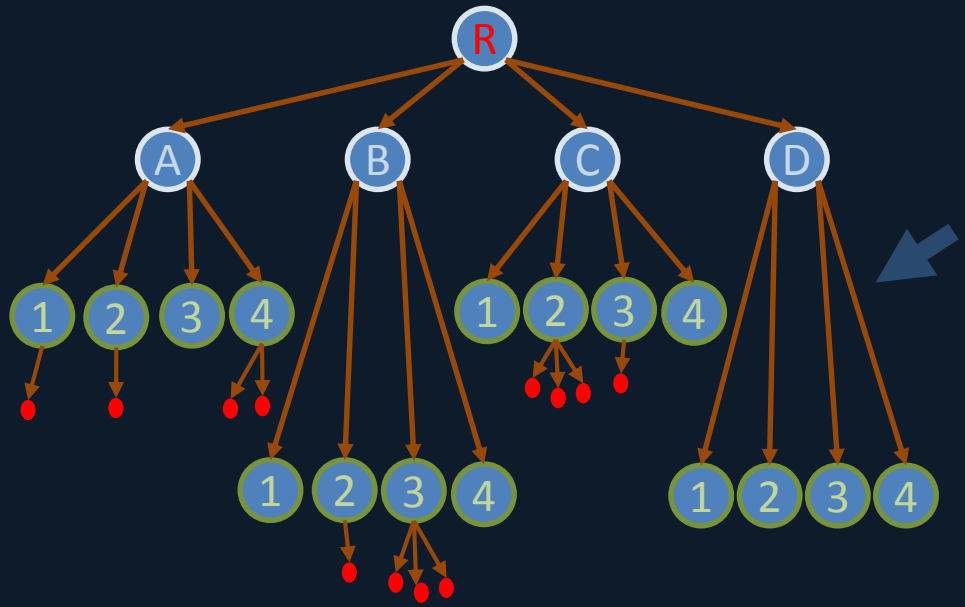
A 1 2 3 4	B 1 2 3 4
C 1 2 3 4	D 1 2 3 4

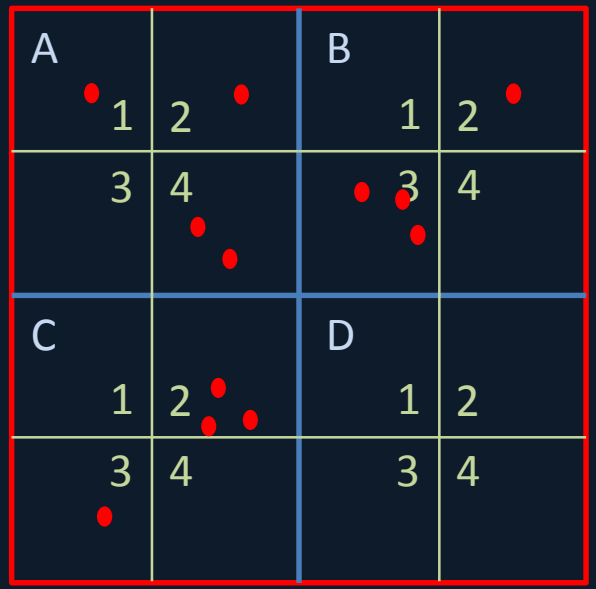
Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Quadtrees

- Then we add our enemies as nodes below the square they are in.



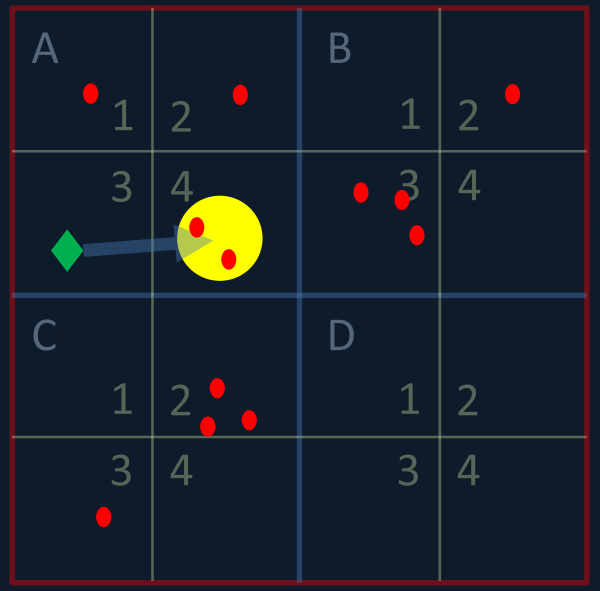


Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

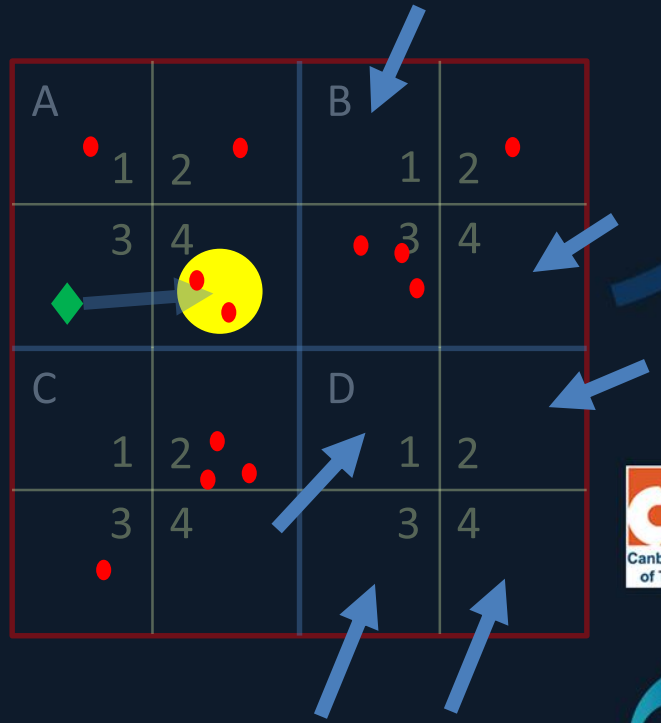
Quadtrees

- Now when the player throws a grenade…
- We traverse the tree:
  - We check the position of the explosion against the four largest quads (A, B, C, D) and find it's in A.
  - Then we check against the four child squares and find it's in square 4.
- Now we only need to check the position of enemies in quad A4.
- Much better than checking against all enemies in the level!



Quadtrees

- But we can do better!
- Notice how some of the squares don’t contain enemies, those are wasting space!

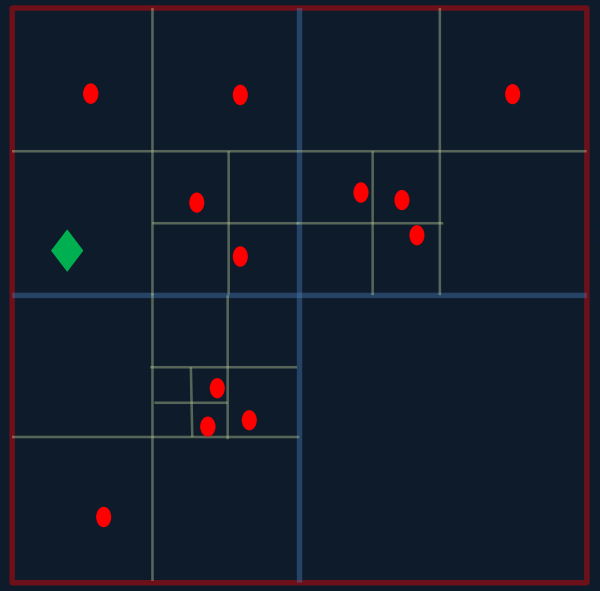


Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

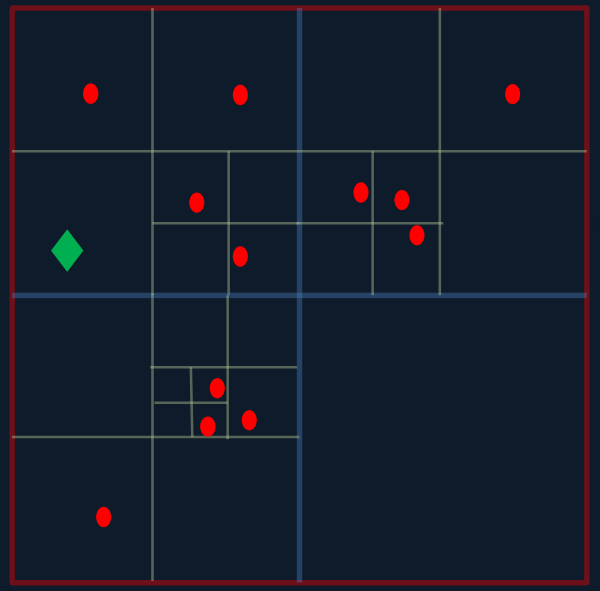
Quadtrees

- Instead of dividing the quads up regularly, we can divide them based on the number of objects inside them.
- As more objects are added, we divide each square more and more



Quadtrees

- This is called an adaptive quadtree.
- In this example there is no more than one enemy per quad but you can design it to support more.
- Very efficient to search.



Quadtrees - Conclusion

- Advantages:
  - Very efficient representation of spaces containing unevenly distributed objects.
  - Relatively simple algorithm.
  - Used in many games for scene graphs.
  - Theoretical maximum search time is O(log n).
- Disadvantages
  - Finding a neighbouring cell can be time consuming.
  - Implementation can lead to fragmentation of memory (cache coherency problems).
  - Creating balanced trees is tricky so theoretical speeds are rarely reached in practice.

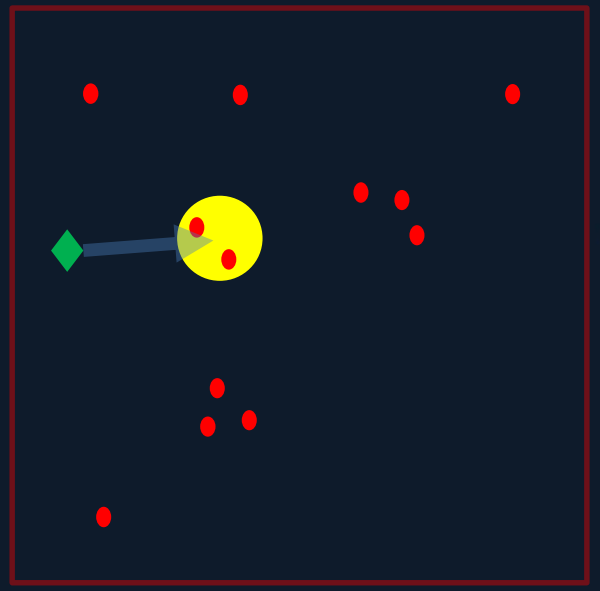
BSP (Binary Space Partitioning) Trees

- Splits space in two, usually along walls or other physical planes in the game.

- Uses a binary tree to position all objects on one side of the split in one branch and those on the other side in the other. Repeats this process over and over to build the tree.

BSP (Binary Space Partitioning) Trees

- Consider again this problem…

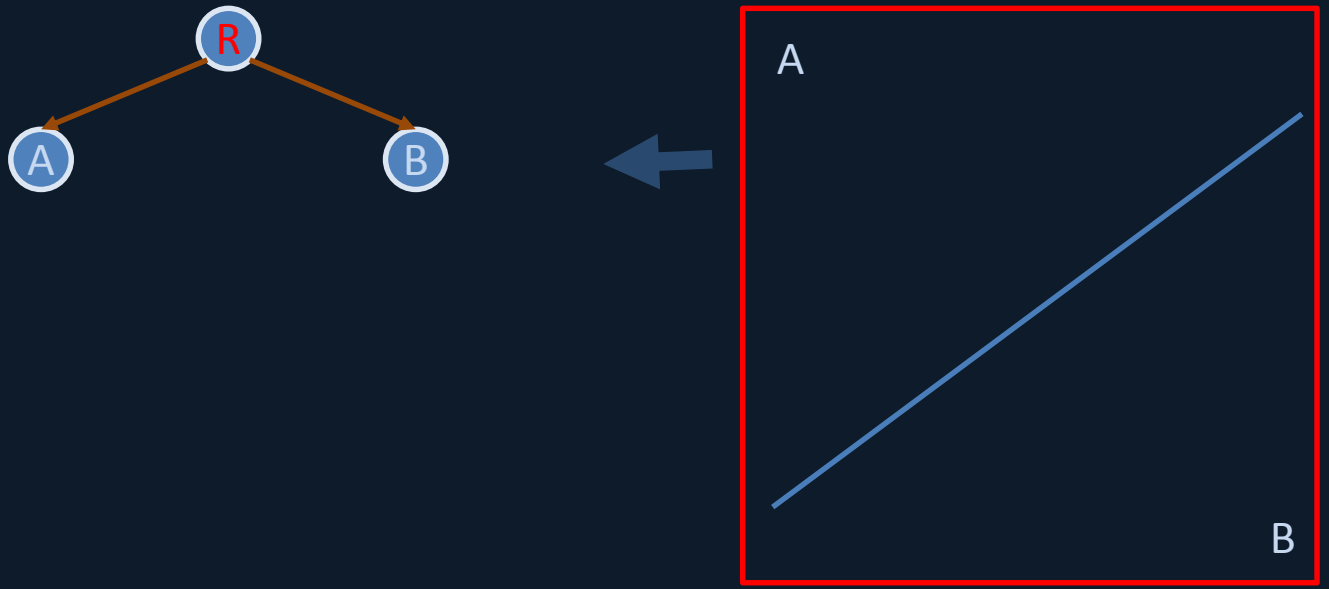


BSP (Binary Space Partitioning) Trees

- Again we treat the level as the root node

BSP (Binary Space Partitioning) Trees

- Then we split the space in two and make each area a branch in our binary tree.
- Normally we would split along physical geometry, like a wall or other obstruction the player can't see through.

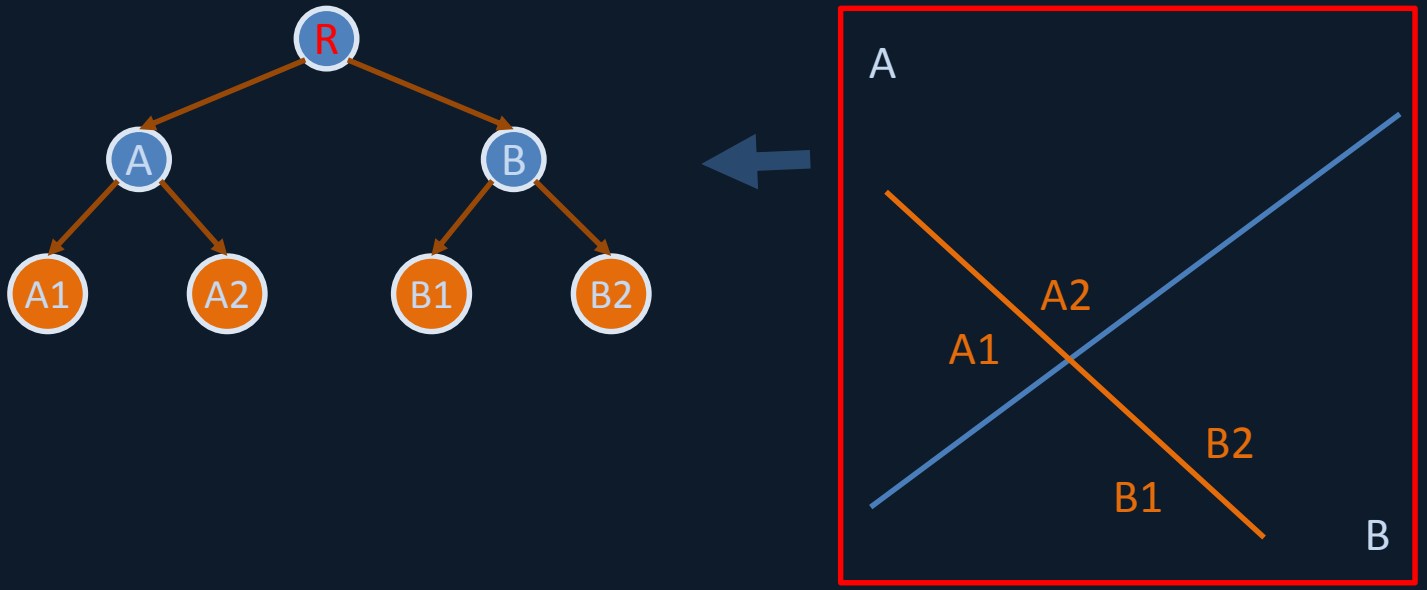


Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

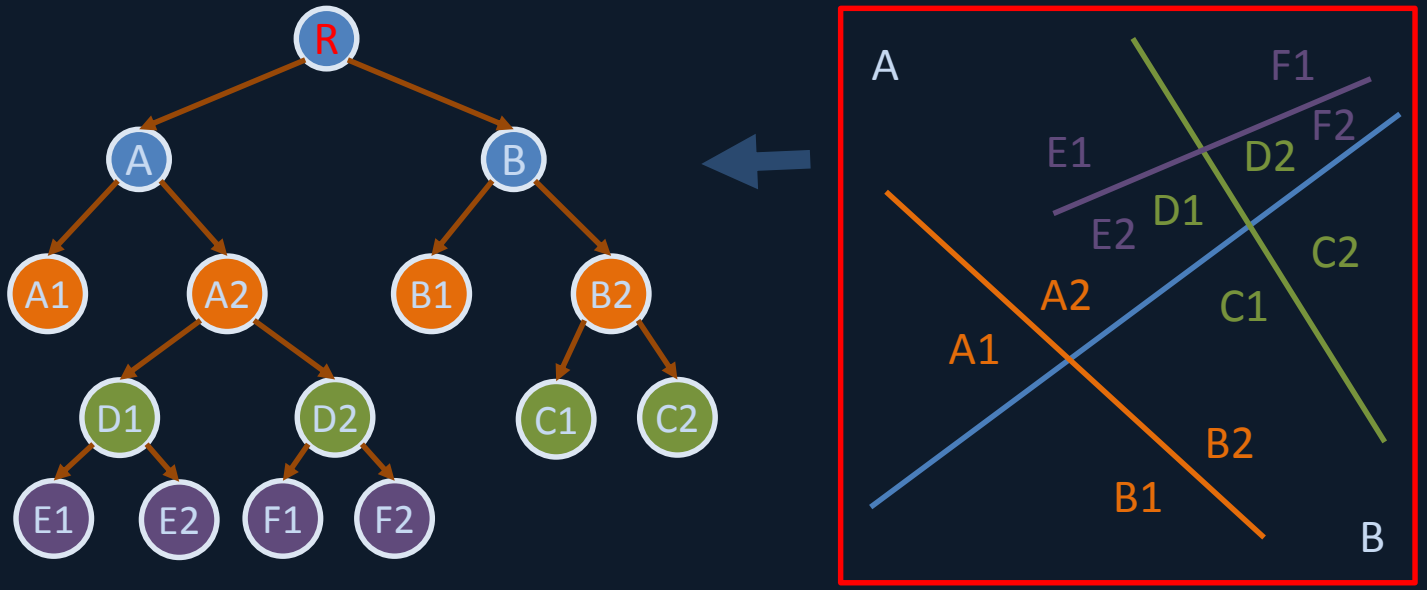
BSP (Binary Space Partitioning) Trees

- Then we split along another plane and add the split spaces to our tree.



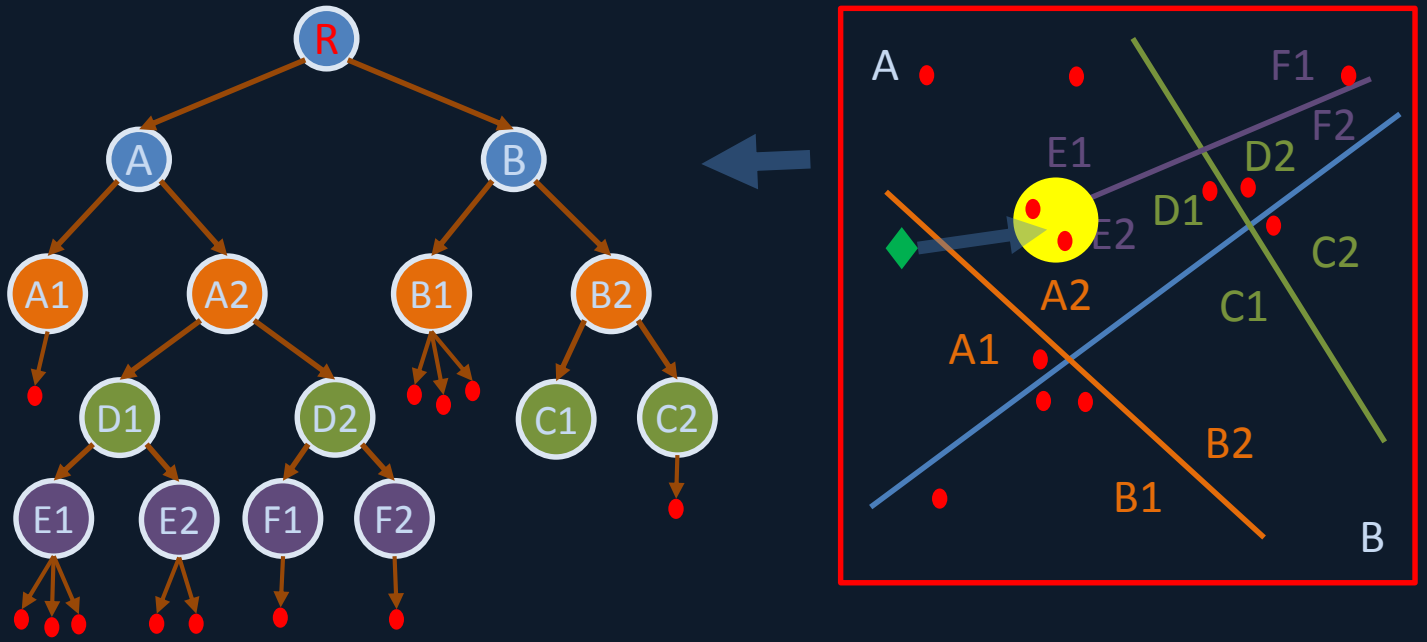
BSP (Binary Space Partitioning) Trees

- And so on… splitting and adding to the tree…



BSP (Binary Space Partitioning) Trees

- Then we add our enemies as leaves on the tree based on which side of each line they fall on.



Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

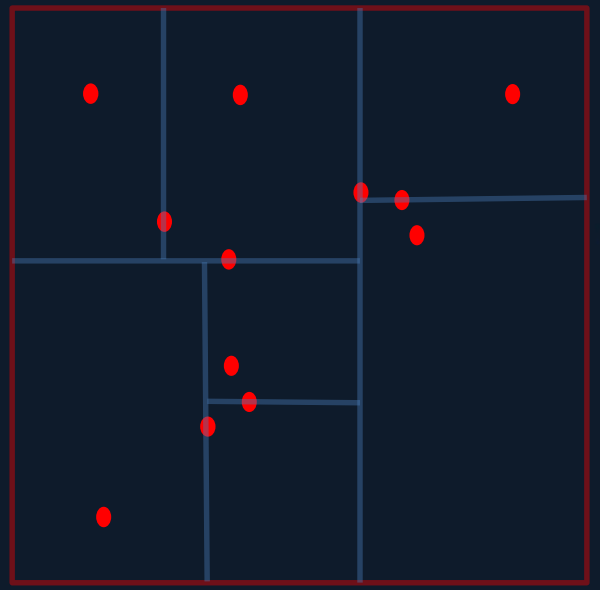
BSP Tree - Conclusion

- Advantages:
  - Works well in enclosed spaces where the area is naturally split by level geometry.
  - The plains can be oriented in any direction and with any origin so spaces can be efficiently divided up.
  - Algorithm is fairly simple.

- Disadvantages:
  - Quite slow to compute.
  - There is no right or wrong way to choose splitting planes so more sophisticated algorithms spend a lot of time trying to create balanced trees.
  - Doesn’t work well for large open spaces.
  - Better methods have been found so not used very often these days.

K-D Trees

- Best thought of as a hybrid of BSP and Quad Tree.
- Space is repeatedly split into two parts (like BSP tree).
- But it is split along one of the axis.
- A different axis is used on each iteration (x, y, x, y, x, y... etc.).
- Usually the median object in the list of objects is picked for the splitting plane.



Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

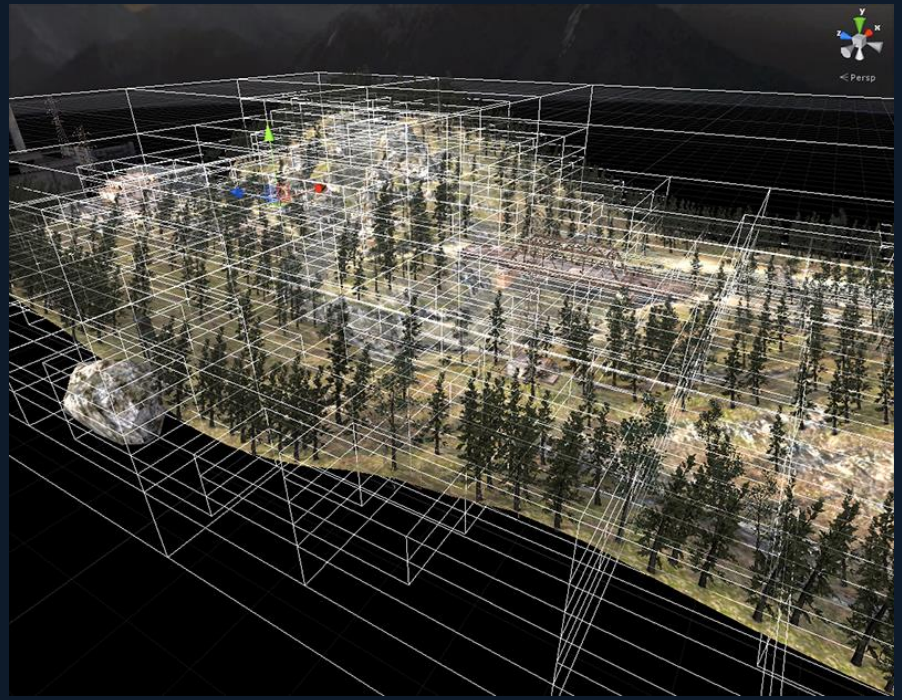
K-D Tree - Conclusion

- Advantages:
  - Straightforward to generate.
  - Easily creates a balanced tree.
  - Very efficient.

- Disadvantages:
  - Finding neighbouring cells is complex (approximate search algorithms are often used to improve performance).
  - Does not work well in high dimensional spaces.
  - Quad and Oct trees have largely replaced it.

Octree

- Works just like a quadtree but in 3D.
- Not much harder to implement.
- Very efficient for partitioning 3D space.



Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Summary

- There are a lot of trees that were designed to solve a very specific problem.
  - Red-Black trees are very efficient for searches.
  - Quadtrees and Octress are very efficient for spatial partitioning.

- Use the tree that fits the situation.

References

- Sedgewick, R, Left-leaning Red-Black Trees, Department of Computer Sciece, Princeton University
  - https://www.cs.princeton.edu/~rs/talks/LLRB/LLRB.pdf
- Samet, H & Webber, R, Storing a Collection of Polygons Using Quadtrees, University of Maryland & Rutgers Univeristy
  - http://www.cs.umd.edu/~meesh/420/Notes/PMQuadtree/pm_quadtree_samet.pdf
- Shimer, C 1997, Binary Space Partition Trees, Advanced Topics in Computer Graphics
  - http://web.cs.wpi.edu/~matt/courses/cs563/talks/bsp/bsp.html

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX