Physics based animation

Lecture 12 - Collision detection
Part 5 - Spatial partitioning

Grégory Leplâtre

g.leplatre@napier.ac.uk, room D32
School of Computing
Edinburgh Napier University

Semester 1 - 2016/2017

Outline

Introduction

Goals:

- Restrict the number of pairwise collision tests

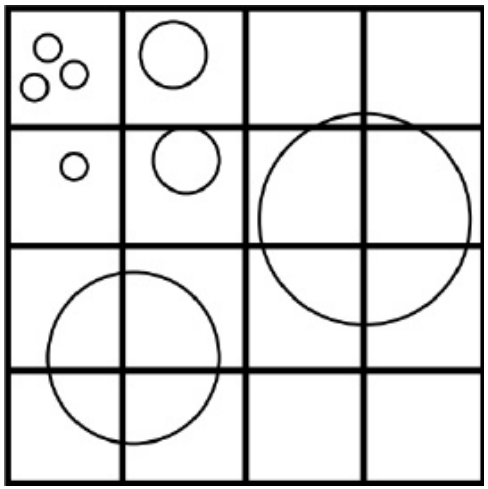
Introduction

Goals:

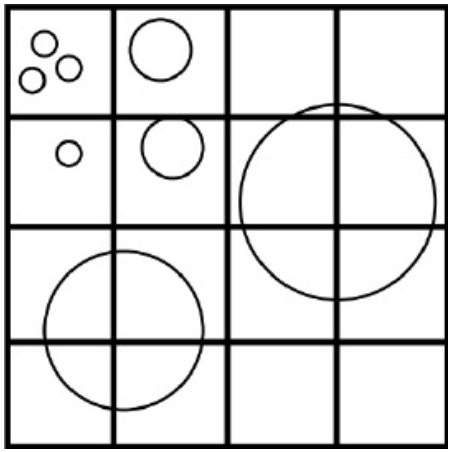
- Restrict the number of pairwise collision tests
- Two types of spatial partitioning:
  - Grids
  - trees

Outline

Uniform grid

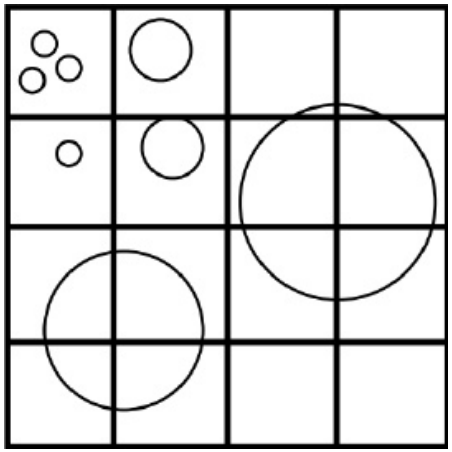


Uniform grids



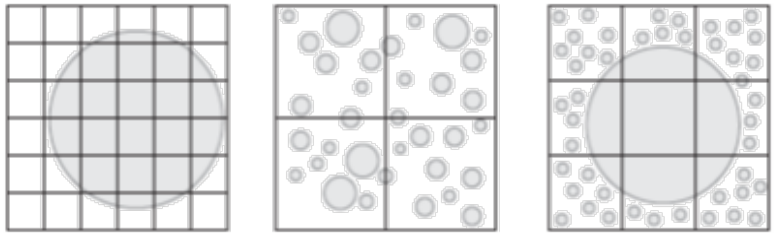
- A simple but effective spatial decomposition scheme is to simply overlay space with a uniform grid (i.e. comprising a number of **equal sized regions** (or cells)).

Uniform grids

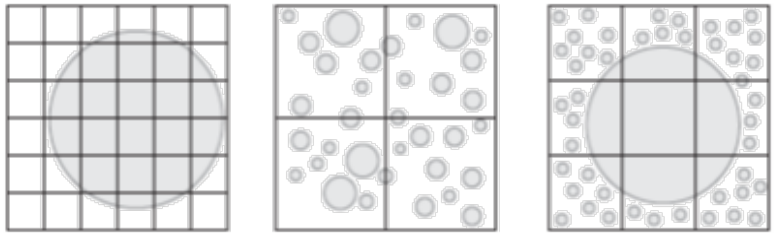


- A simple but effective spatial decomposition scheme is to simply overlay space with a uniform grid (i.e. comprising a number of **equal sized regions** (or cells)).
- Only those objects which overlap a common cell(s) can be in contact $\Rightarrow$ intersection tests are only performed against objects which share cells.

Design and performance issues

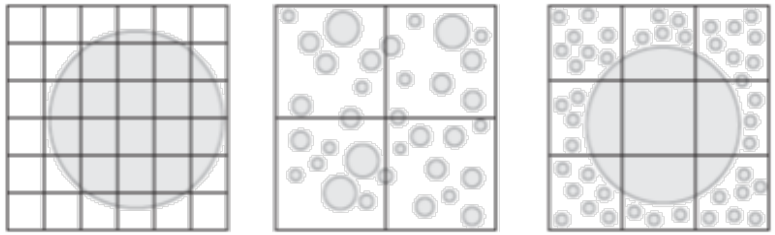


Design and performance issues



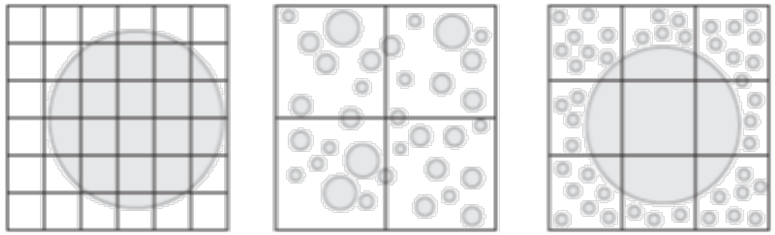
- **Too fine**: Large number of cells need to be updated when a large object moves

Design and performance issues



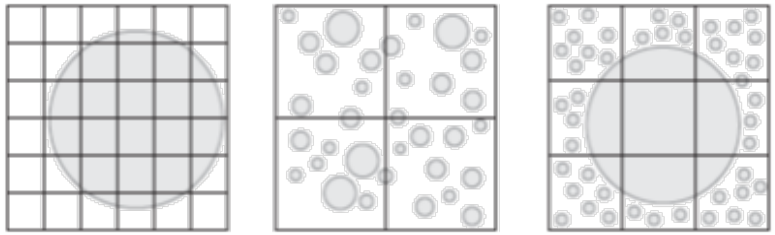
- **Too fine**: Large number of cells need to be updated when a large object moves
- **Too coarse**: Large amount of collision tests likely to be required

Design and performance issues



- **Too fine**: Large number of cells need to be updated when a large object moves
- **Too coarse**: Large amount of collision tests likely to be required
- **Both?**: Linked to the relative size of the partition and of the objects

Design and performance issues



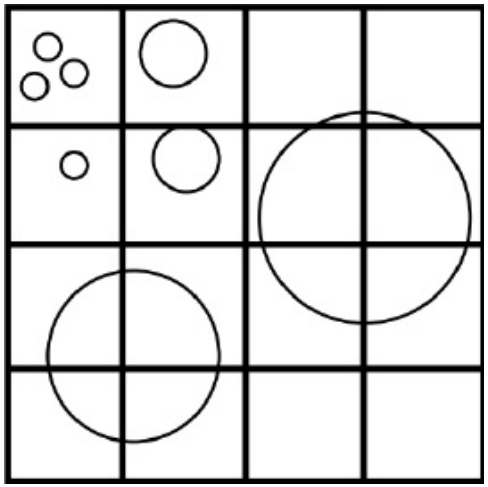
- For uniform grids, the cell size is normally adjusted to be **large enough to accommodate the largest object object** at any rotation.
- $\Rightarrow$ an object cannot overlap more than 4 cells (in 2D) or 8 cells (in 3D).

Summary

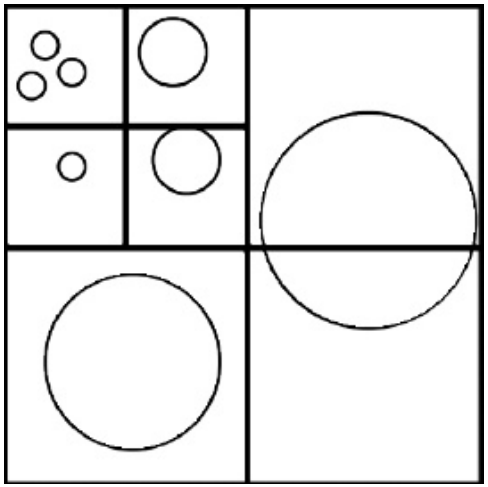
- **Pros**
  - Simple implementation
  - Fast for **small dynamic** objects and **large static** ones (environments).
- **Cons**
  - finding optimal grid size
  - Memory (large 3d grid)
  - Accuracy depends on grid resolution

Outline

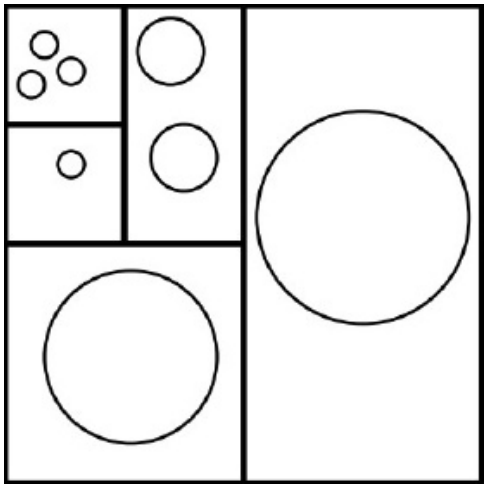
Introduction



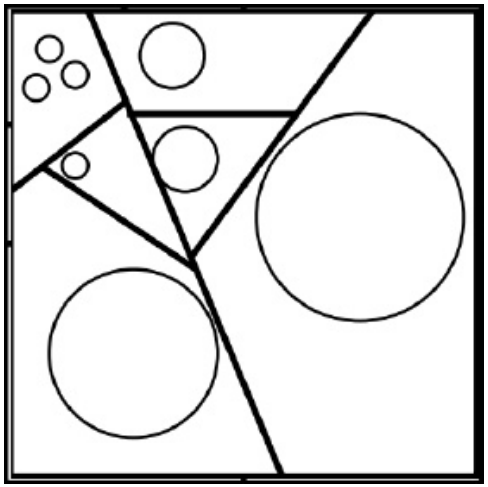
Introduction



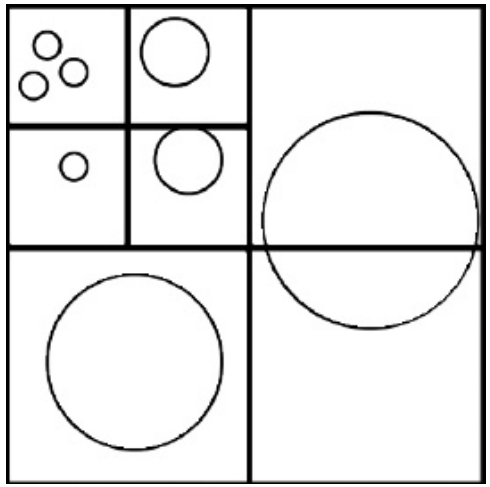
Introduction



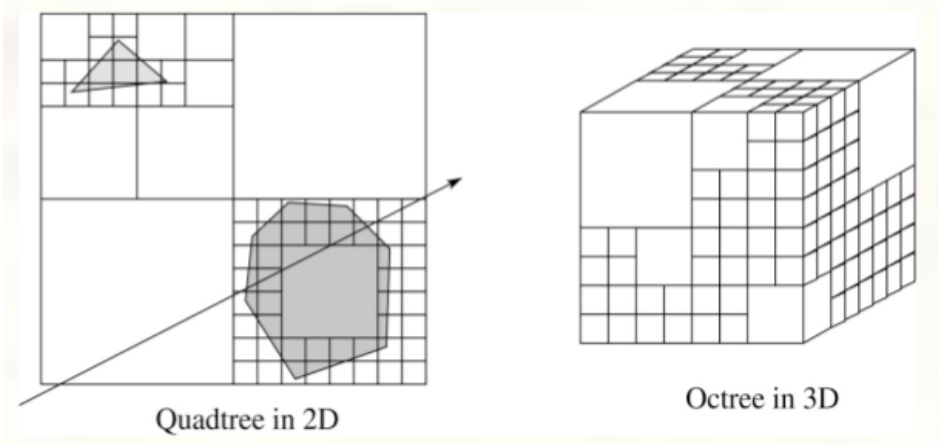
Introduction



Octrees

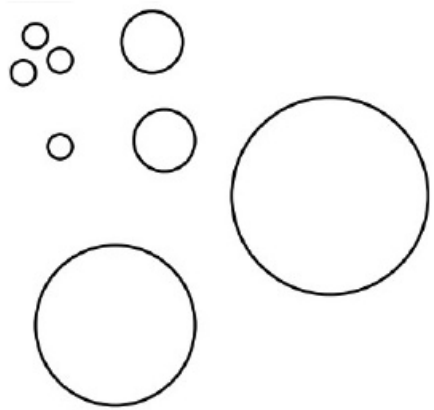


Octrees



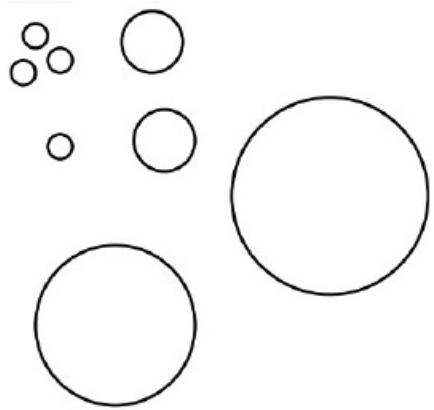
- Axis-aligned tree-based hierarchical partitioning of space.
  - The root node is typically the smallest AABB which fully encloses the world.
  - Each tree node can be divided into eight smaller regions of space
  - Typically the root node is recursively subdivided until either some maximum tree depth or minimum cube size limit is reached

Octree construction



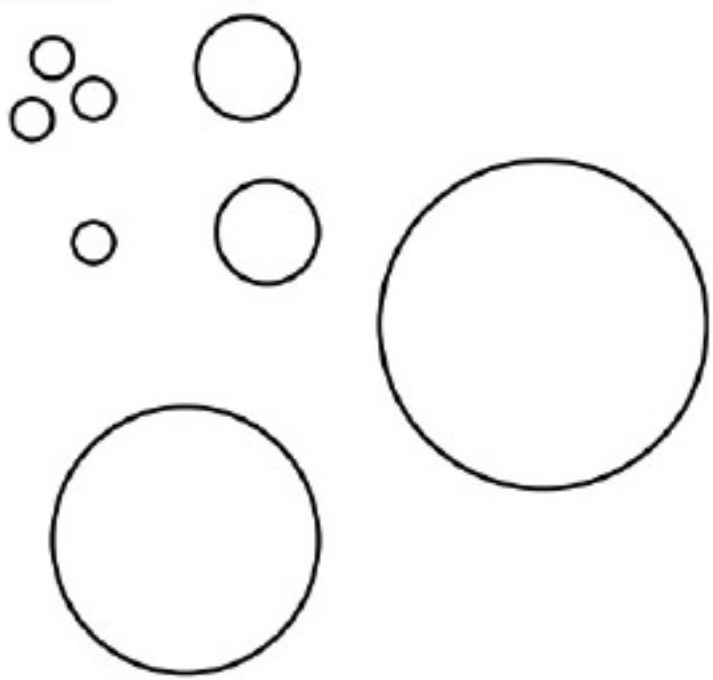
- **static data** Octree formed using a top-down approach.
  - All objects initially associated with the root node. As the root node is split, objects are assigned to all the child nodes it overlaps.
  - The process is recursively repeated until some stopping criteria is reached (e.g. max depth, min objects per cell, etc.).

Octree construction

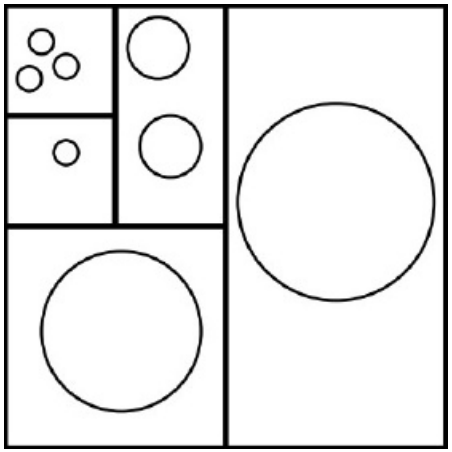


- **static data** Octree formed using a top-down approach.
  - All objects initially associated with the root node. As the root node is split, objects are assigned to all the child nodes it overlaps.
  - The process is recursively repeated until some stopping criteria is reached (e.g. max depth, min objects per cell, etc.).
- **Dynamic data** Octree formed by restricting objects to the **lowest octree node that fully contains the object**

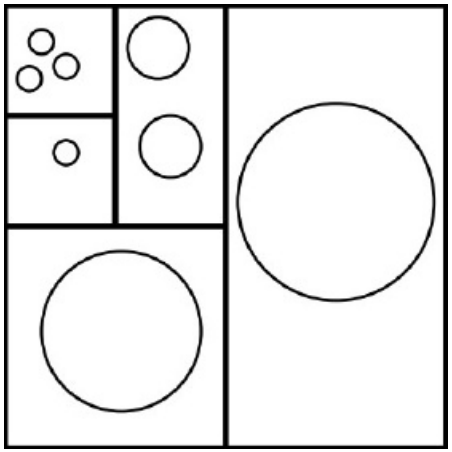
Octree construction



K-d trees

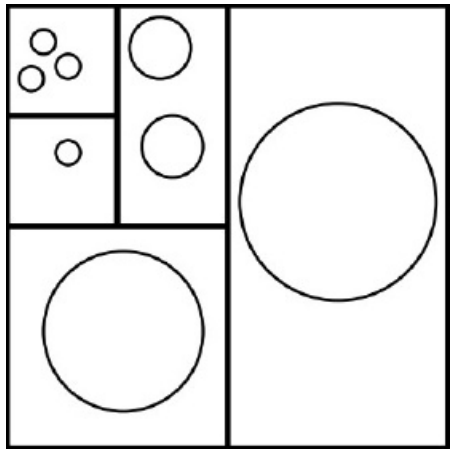


K-d tree



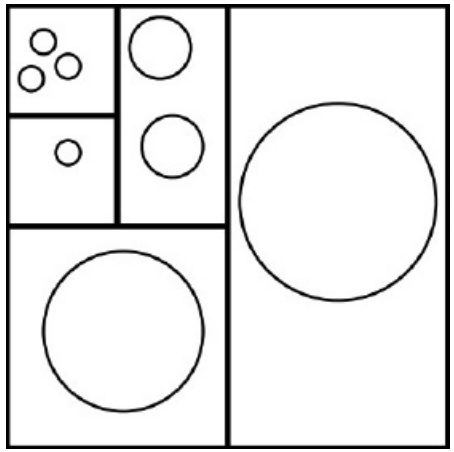
- The k-dimensional tree (or k-d tree) is a generalisation of octrees and quadtrees, where **k** represents the **number of dimensions subdivided**.

K-d tree



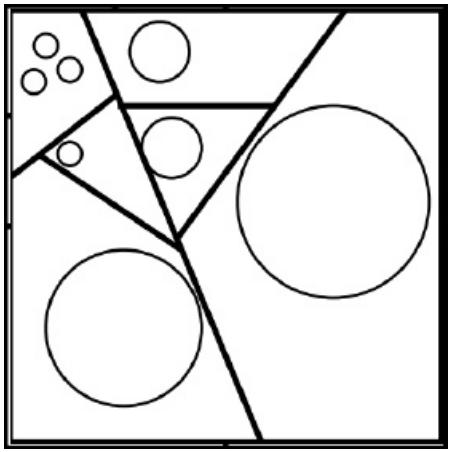
- The k-dimensional tree (or k-d tree) is a generalisation of octrees and quadtrees, where **k** represents the **number of dimensions subdivided**.
  - Instead of simultaneously dividing space in two (quadtree) or three (octree) dimensions, the k-d tree divides space along one dimension at a time.

K-d tree

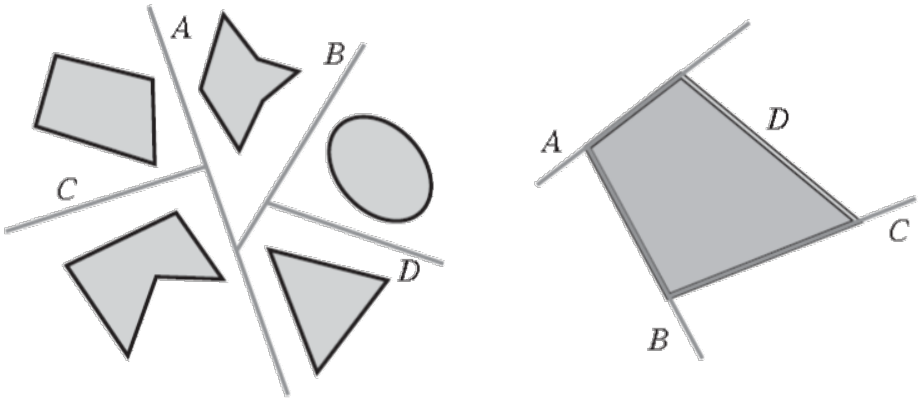


- The k-dimensional tree (or k-d tree) is a generalisation of octrees and quadtrees, where **k** represents the **number of dimensions subdivided**.
  - Instead of simultaneously dividing space in two (quadtree) or three (octree) dimensions, the k-d tree divides space along one dimension at a time.
  - **The splitting axis can be freely selected**

BSP trees



BSP Trees

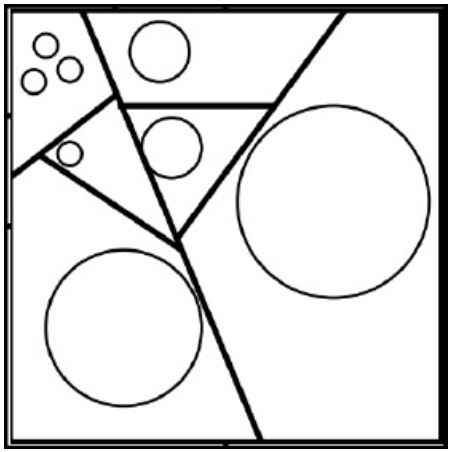


- Can serve two functions:
  - **Spatial partitioning**
  - **Volume representation** for polygons/polyhedra

Partitioning approaches

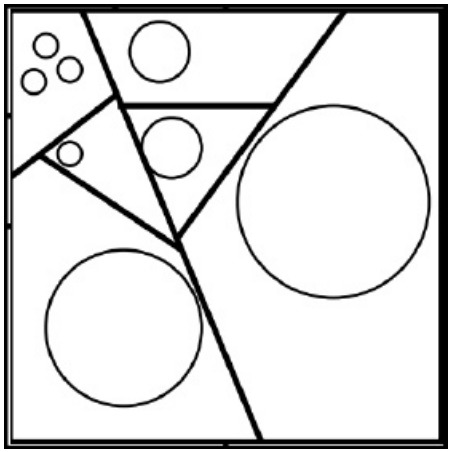
- **Object aligned** partitioning
- **Axis aligned** partitioning $\Leftrightarrow$ k-d tree
- **Arbitrary** partitioning

Building a BSP tree



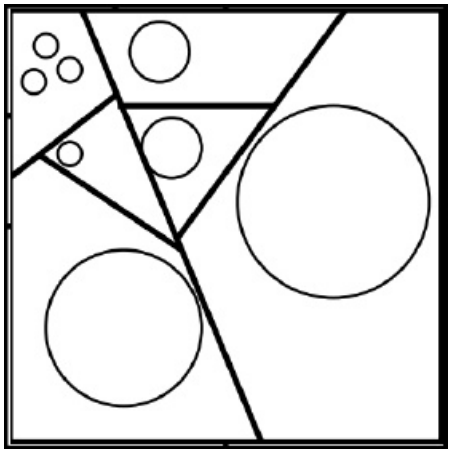
- Building a BSP tree involves three steps.
  - Selection of a partitioning plane

Building a BSP tree



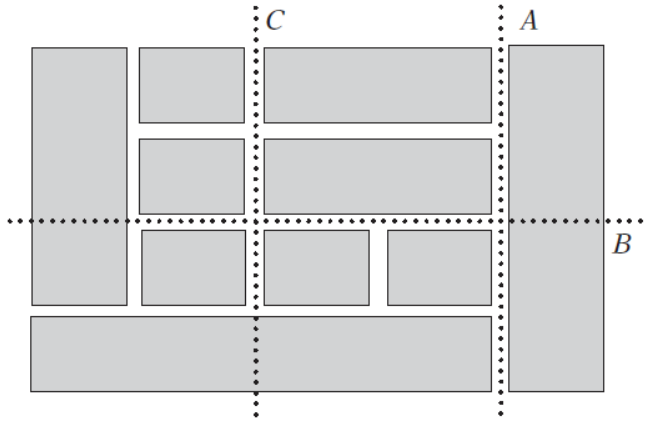
- Building a BSP tree involves three steps.
  - Selection of a partitioning plane
  - Partitioning input geometry into the positive and negative halfspaces of the dividing plane. Geometry that straddles the plane is split to the plane before partitioning.

Building a BSP tree



- Building a BSP tree involves three steps.
  - Selection of a partitioning plane
  - Partitioning input geometry into the positive and negative halfspaces of the dividing plane. Geometry that straddles the plane is split to the plane before partitioning.
  - Repeat recursively until termination condition is met

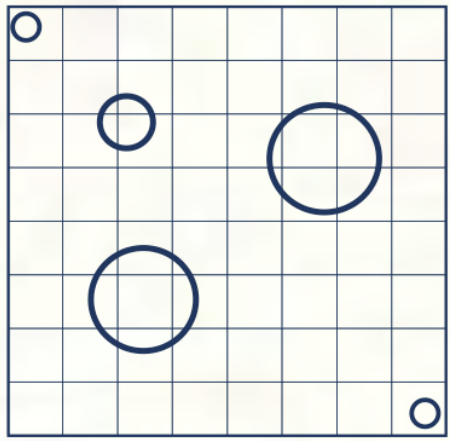
Selecting a dividing plane



- Two conflicting criteria:
  - minimise splitting of geometry
  - Balance geometry equally on each side of the plane
- Using a weighted combination is a good compromise.

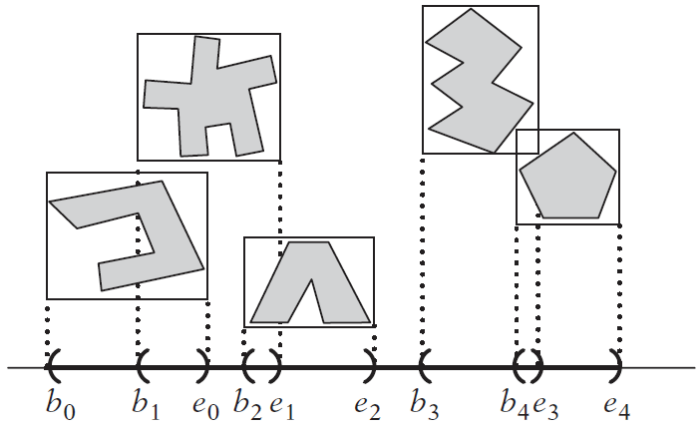
Outline

Sort and Sweep methods



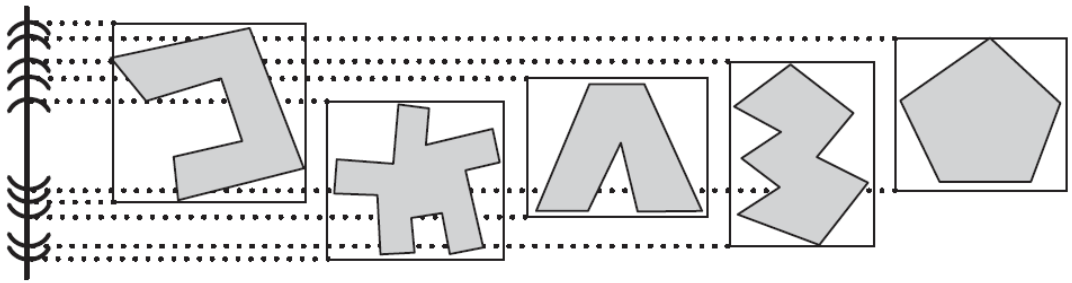
- Problem: One drawback of inserting objects into fixed spatial subdivisions (grids, octrees, etc.) is having to handle **objects straddling multiple partitions**.

Sort and Sweep methods



- Problem: One drawback of inserting objects into fixed spatial subdivisions (grids, octrees, etc.) is having to handle **objects straddling multiple partitions**.
- Solution:
  - maintain a **sorted spatial ordering** of objects.

Sort and Sweep methods



- Problem: One drawback of inserting objects into fixed spatial subdivisions (grids, octrees, etc.) is having to handle **objects straddling multiple partitions**.
- Solution:
  - maintain a **sorted spatial ordering** of objects.
- Limitations: clustering of objects, but also sorting cost

Outline

Summary

- Overview of common spatial division methods:
  - unifrom grid
  - Tree-based representations
  - Sort and sweep

Summary

- Overview of common spatial division methods:
  - unifrom grid
  - Tree-based representations
  - Sort and sweep
- Other topics (see textbook)
  - optimisation
  - Ideal topic for further studies

References

- Ericson, C. (2004). Real-time collision detection. CRC Press.