Acceleration Algorithms and Scene Representation

Sandro Spina
Computer Graphics and Simulation Group

Computer Science Department
University of Malta

Scene Representation

- We've seen so far how rendering involves a process which loops through the geometry in your scene and renders it individually.
- It would make sense to organise all the geometry in a scene into a data structure in order to try and improve (optimise) certain aspects of the rendering process.
- Probably the simplest form of data structure would be an array (or linked list) which stores a reference to every piece of geometry in the scene.
- The problem with an array is that operations on it are linear and as the size (number of items) of our scene increases such a linear structure would not be ideal.
- Trees and sometimes more generic graphs are used to organise (and thus optimise) the objects in your scene.

Need for Performance …

- One can argue that GPUs (and to a certain extent CPUs) are becoming increasingly fast and that we can just adopt a brute force approach were we just push every piece of geometry down the rendering pipeline.

- In reality (especially in real-time rendering), a lot of time could be spent much better towards achieving four different performance goals:
  - More frames per second
  - Higher resolution and sampling rates
  - More realistic materials and lighting
  - Increased complexity of geometry

- Beyond these goals, when rendering a scene in real-time, one might also try to improve other aspects of the application such as physics and AI algorithms.

- Bottom line … there will always be a need to improve algorithms to increase performance.

Spatial Data Structures (i)

- A spatial data structure is one that organises geometry in some *n*-dimensional space.
- Since we are dealing with 3D CG our data structures will be organising our geometry in 3-dimensional space.
- We shall be using these data structures in order to accelerate queries (related to culling, intersection testing, ray tracing and collision detection) about whether geometries overlap.
- A spatial data structure usually exhibits a hierarchical form, such as a tree where the topmost level (somehow) encloses the one beneath it, with this level enclosing the one beneath it, and so on making the structure recursive in nature.

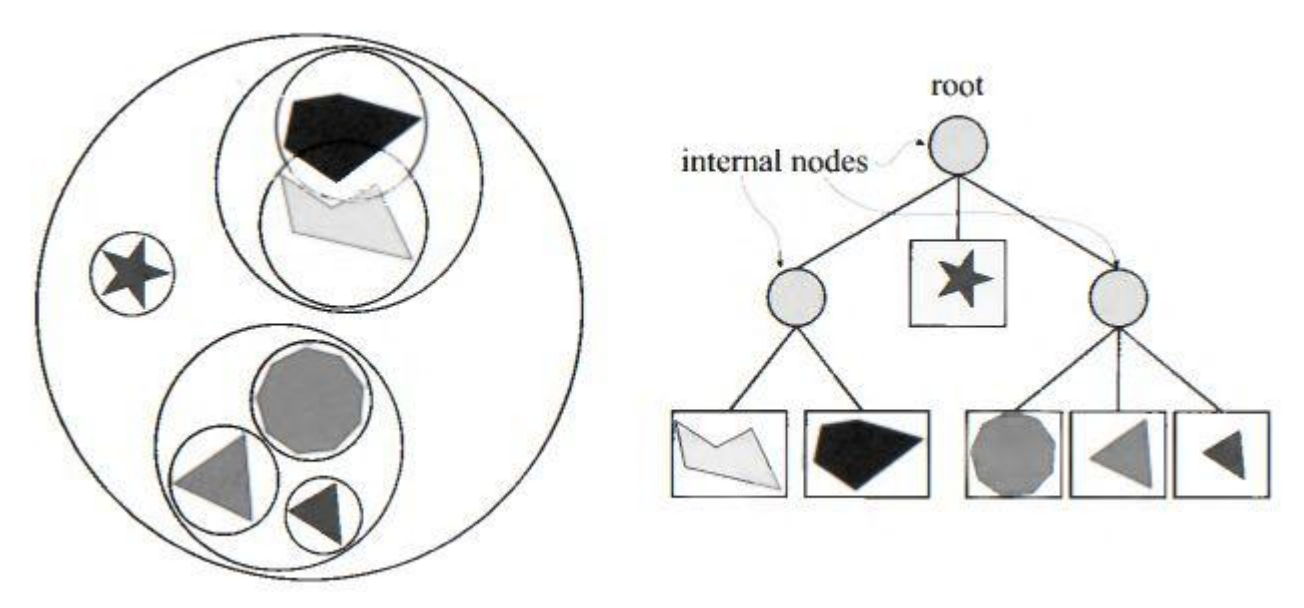
Spatial Data Structures (ii)

- Using trees (or other hierarchical structures) other than simple arrays usually leads to an improvement for queries from O(n) to O(log n).
- However one also has to factor in the time it takes to build the tree.
- We shall be looking at three common types of spatial data structures:
  - Bounding Volume Hierarchies
  - Binary Space Partition Trees
  - Quadtrees and Octrees
- BSP, Quadtrees and Octrees are data structures which capture all the space (volume) in the scene and are referred to as space subdivision structures. On the other hand BVH is not guaranteed to cover all space but will cover all the geometric entities in the scene.

Bounding Volume Hierarchies (i)

- A Bounding Volume (as it's name implies) is a volume that encloses a set of objects.
- This means that all vertices of this set of objects lie within this BV.
- The most important characteristic of a BV is that it should be a simpler geometry (for eg a sphere), than it's enclosed objects.
- This BV geometry needs to be simpler, in order to act as a proxy in place of the bounded objects, so that when doing tests/queries with the BV, these tests/queries are done much faster.
- Examples of BVs include spheres, axis-aligned bounding boxes (AABB), oriented bounding boxes (OBB)

Bounding Volume Hierarchies (ii)



- The diagram above shows 6 objects, each with its own BV.
- BV are then grouped and enclosed in larger BV
- Finally the root BV encloses the whole scene
- On the right is the BVH representing this simple 2D scene, consisting of one root node, internal nodes and leaf nodes which store the actual geometry.

Bounding Volume Hierarchies (iii)

- In constructing a BVH, one should attempt to create a balanced tree, i.e. a tree in which all leaf nodes either are at height *h* or *h*-1.
- In general the height of a balanced tree is $\lfloor \log_k n \rfloor$, where n is the total number of nodes and k is the number of children of each internal node.
- Note that the higher the value of *k*, the lower the height of a balanced tree is.
- BVHs are excellent for performing intersection queries. For eg assume we want to check whether a ray interest our scene. Testing starts at the top (root node) and proceeds downwards to check the different bounding volumes at the next level. If no intersection is found with a certain BV than one can declare that the ray will not intersect any of the geometries within that volume. If an intersection is found then the next level is tested.

Bounding Volume Hierarchies (iv)

- In constructing a BVH, one should attempt to create a balanced tree, i.e. a tree in which all leaf nodes either are at height *h* or *h*-1.
- In general the height of a balanced tree is $|\log_k n|$, where n is the total number of nodes and k is the number of children of each internal node.
- Note that the higher the value of *k*, the lower the height of a balanced tree is.
- BVHs are excellent for performing intersection queries. For eg assume we want to check whether a ray interest our scene. Testing starts at the top (root node) and proceeds downwards to check the different bounding volumes at the next level. If no intersection is found with a certain BV than one can declare that the ray will not intersect any of the geometries within that volume. If an intersection is found then the next level is tested.

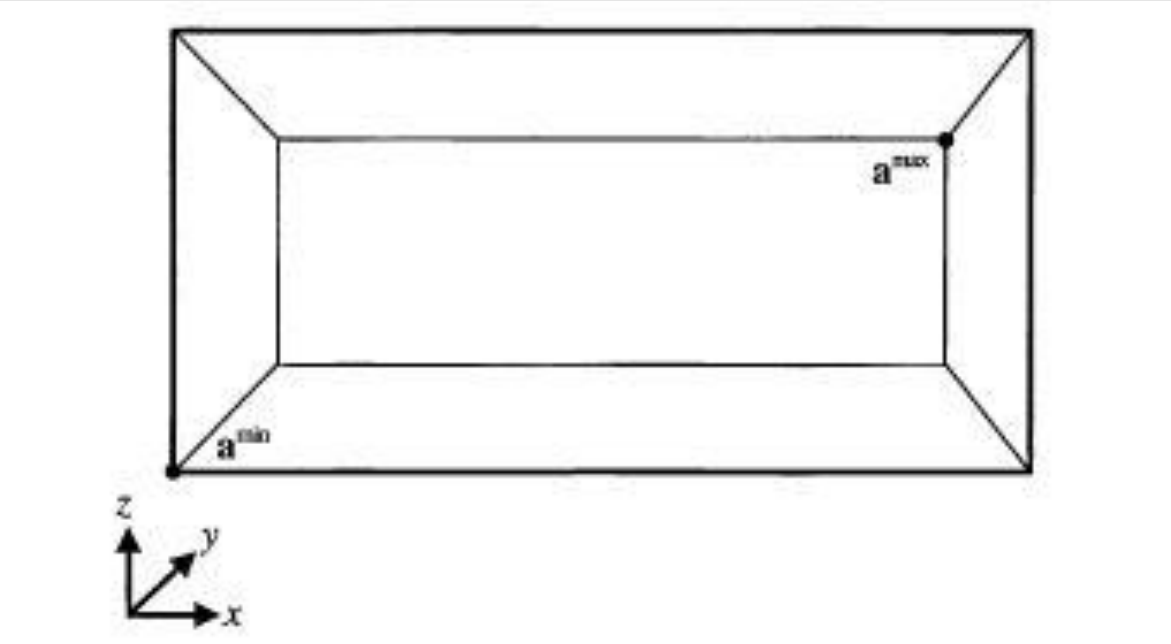
Bounding Volume Hierarchies (v)

- In the case of dynamic scenes … BVH need to be checked for consistency.
- If an object in a BV has moved, we need to calculate first a new bounding volume and check whether this new volume is still within it's parent node in the BVH.
- If it is then fine the BVH remains consistent, but if not the object node is removed and the parent BV is re-computed.
- The removed object is inserted in the BVH starting from the root of the BVH. Optimizations in this algorithm can be carried out by moving the object to bottom to top in the BVH rather than top (root) to bottom.
- Optionally one can also grow the parent BV to include the movement of the object.

AABB Bounding Volume

- Definition: An axis-aligned bounding box (AABB) is a box whose faces have normals that coincide with the standard basis axis.

- An AABB is specified by two diagonally opposite points, a^{min} and a^{max}, where $a_i^{min} \leq a_i^{max}$, for all *i* member of {*x*, *y*, *z*} .

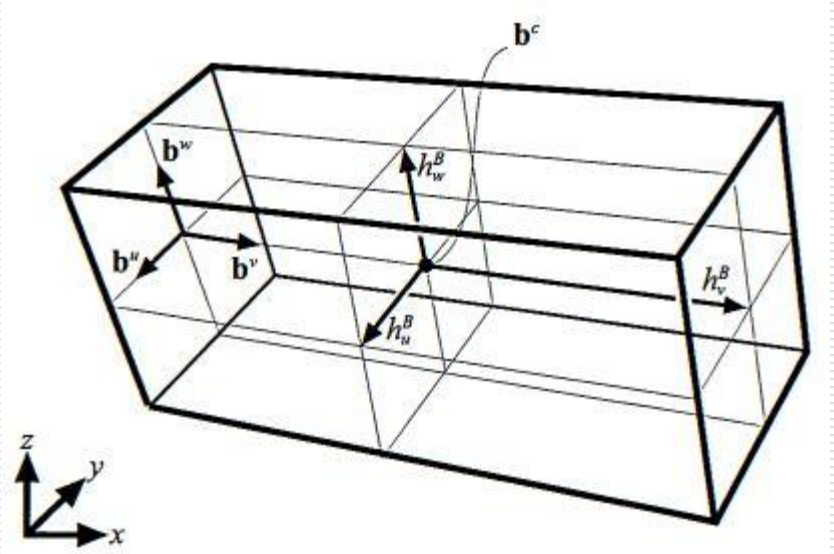


OBB Bounding Volume

- Definition: An Oriented Bounding Box (OBB), is a box whose faces have normals that are all pairwise orthogonal – i.e. it is an AABB that is arbitrarily rotated.

- An OBB can be described by the centre point, $\mathbf{b}^c$, and three normalised vectors $\mathbf{b}^u$, $\mathbf{b}^v$ and $\mathbf{b}^w$, that describe the side directions of the box.

- The half lengths also need to be specified.



Sphere Bounding Volume

- The sphere bounding volume is defined by specifying the centre of the sphere and a value for the radius.
- A sphere as a bounding volume is a sensible choice only when your x:y:z ratio is similar …
- If for example x ranges from 1 .. 2, y ranges from 1 .. 3, and z ranges from 1 .. 30, using a sphere as a bounding volume is not a good move.
- For eg. arms and legs would be better off using AABB or OBB.
- XNA provides a method for creating a sphere bounding volume.
-

Creation of Bounding Volumes

Intersections tests ...

AABB with Plane ...

Sphere with Plane ...

View Frustum Bounding Volume

AABB / View Frustum Volume Intersection

Sphere / View Frustum Volume Intersection

BSP Trees

- In the case of dynamic scenes … BVH need to be checked for consistency.
- If an object in a BV has moved, we need to calculate first a new bounding volume and check whether this new volume is still within it's parent node in the BVH.
- If it is then fine the BVH remains consistent, but if not the object node is removed and the parent BV is re-computed.
- The removed object is inserted in the BVH starting from the root of the BVH. Optimizations in this algorithm can be carried out by moving the object to bottom to top in the BVH rather than top (root) to bottom.
- Optionally one can also grow the parent BV to include the movement of the object.

Bounding Volumes in XNA

-