Graphics and Computational Programming Assignment 1

Graham Rigler  
i7465070

# Part 1

## Brute-Force Ray-Triangle Intersection

### Theory

* Ray-Plane testing
* Plane contact point – Triangle testing

### Implementation

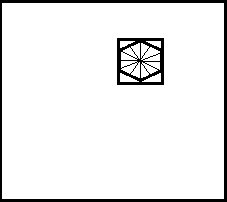
* Reason for ordering the way it is
* Pre-computation

## Bounding Volumes

### General Theory

A bounding volume (BV) is a representation of an object in world-space using some primitive shape, such as a cube or a sphere. This allows for more complicated geometry to be represented in a simplistic, if not completely accurate, form. A popular implementation of BVs is the Axis-Aligned Bounding Box (AABB), this encompasses an object in a box that is relative to the X, Y and Z axis as opposed to being oriented to the object (although this implementation exists as the Oriented Bounding Box). The purpose of this in the context of this assignment is that it allows for each ray to be tested against the bounding volume before being tested against triangle, this is a computationally cheaper test and has been explained further below.

Figure - Simple diagram of an arbitrary object, surrounded with a tight-fitting AABB, displayed in a on a screen.

As can be seen in Figure 1, the arbitrary object in this example is taking a relatively small portion of the screen. This means if each ray, which in the context of this example is equivalent to a pixel on the screen, was to test against each triangle on the object then there would be a lot of expensive calculation being done needlessly as there it is clear there is no way in which those rays will be colliding with the object. Therefore, a check must be done to ensure that the ray’s origin is within the BV in the relevant axis, for this instance the only relevant axis are Y and Z as the ray’s direction is always constant in the X axis as 1.

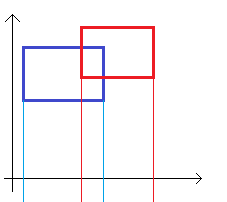


Figure - Diagram showing the projection of the minimum and maximum values of two AABBs onto an arbitrary axis for collision testing.

In order to do this, the minimum and maximum values of the BV must be projected onto the axis, as is shown in Figure 2 for an arbitrary axis. In this figure the minimum and maximum values have been displayed against the axis using lines. When AABBs are used for collision detection between objects, each object has a BV in 3D and is said to pass the collision test if the minimum value of one is less than the maximum value of another (or vice-versa) on all axis, however in this context the ray does not need to have a bounding volume per-say as it does not exist as a physical object. Instead the Y and Z bounding values for the object’s AABB are projected and used to test against the origin of the ray. The equations for this simple test are shown below.

()

Where is the origin of the ray, and and define the minimum and maximum bounds of the AABB of an object.

()

As can be seen in (1), the ray’s origin is defined in the Y and Z axis in this example and as it’s direction is constant in the X axis, the only axis of the object that need to be tested of the object are the Y and Z axis as is shown in (2).

If this test has not passed then this ray will not test against any triangles at all whereas the ray would have tested against every single triangle using brute-force, wasting processing time. If the test has passed, then it can be seen that there is a chance for this ray to collide with a triangle within the object (although it is important to note that this is not guaranteed, due to the inaccuracy in AABB and it not exactly representing the object) therefore a brute-force test will continue for this ray.

### Per-Triangle

#### Differences

While an AABB can be calculated to encompass the entire object to initially test for a possible collision and then carrying on with a brute-force pass if this passes, it is also possible to create an AABB for each triangle in order to reduce the overhead in this brute-force testing pass.

#### Memory Overhead vs. Performance

### Bounding Volume Hierarchy

* What it is, the benefits and why I’ve not done it (laziness in truth, but easy marks in the report)

## Algorithm Analysis

### Brute-Force

### Per-Object AABB

### Per-Triangle AABB

# Part 2