**Physics Simulation**

The physics simulation is a way of accurately demonstrating how collisions between objects such as circles, polygons and planes would resolve, as well as the effects of wind resistance and gravity on the aforementioned objects.

The shapes involved include as follows:

* **Plane**

An infinitely stretching, and infinitely thin line the keeps any objects from passing through. It is defined by a direction vector, and a distance float.

* **Sphere**

A perfectly round object. It is defined by its position and radius.

* **AABB**

An Axis-Aligned Bounding Box is a rectangle that is always aligned with the global axes, in other words the rectangle’s up is the same as the global up, and the rectangle right is the same as the global right.

It is defined by a position vector, as well as the bottom left and top right position vectors (relative to the center position). It is the fastest for collision checks.

* **Polygon**

A shape with any number of sides, and is not necessarily axis aligned. It uses SAT (Separating Axis Theorem) to detect collision, which involves projecting the two involved shapes onto the axis of each side of both shapes. It is defined by a position vector, and a list of all of its vertices.

For every shape, there needs to be a function for testing collision with every shape (including another of itself).

There also exists functionality for “kinematic objects” that aren’t affected by the physics engine, but can affect the physics engine. These objects typically have a separately controlled movement.

**Improvements**

In future development for the engine, improvements could include:

* **Imposter collision**

A polygon wrapped up by a more primitive collider (e.g. Sphere) which is used for detection before moving on to the more complex SAT. Without this, having multiple Polygons slows down the program quite a lot.

* **Friction**

Friction between objects should cause a more realistic outcome, as objects ideally should slow down while sliding along the ground or when colliding with other objects.

* **Rotation**

Angular velocity should affect Spheres and Polygons. Spinning objects create a different - and more entertaining - outcome than just regular collisions. AABBs and Planes should never rotate.

**Third Party Libraries**

* GLM
* vector
* iostream

**Reference**

* aie.instructure.com
* dyn4j.org
* cppreference.com
* draw.io