**Spike:** Spike No.24

**Title:** Task 24 – Collisions

**Author:** Khang Trinh - 102118468

**Goals / Deliverables:**

The goal of this spike is to teach the developer how to handle sprite collisions.

**Technologies, Tools, and Resources used:**

* Visual Studio 2017

**Useful Links:**

1. Basic examples of 2D collision detection

<https://developer.mozilla.org/en-US/docs/Games/Techniques/2D_collision_detection>

1. More examples of 2D collision detection

<http://www.jeffreythompson.org/collision-detection/>

# Tasks undertaken:

## Step 1: Define the properties of the shape

Defining a collision is basically saying, “this shape is overlapping another shape”, or in another word, “this shape is in a *location* such that its *bounding area* would be overlapping another shape’s bounding area.” There are many ways to define a shape’s location and/or bounds using its properties. Some examples of this are:

* Points have their coordinates (x, y or x, y, z etc.)
* Boxes have edges that can be used to determine both their bounds and location
* Circles have their origin and radius for their location and distance from other shapes

## Step 2: Compare the differences

Once you’ve defined the properties, the next step is to compare the differences to see if they are overlapping each other. Some examples to look for are:

* Check for intersection between 2 lines
* Check for distance between 2 points
* Check for distance between 2 parallel edges

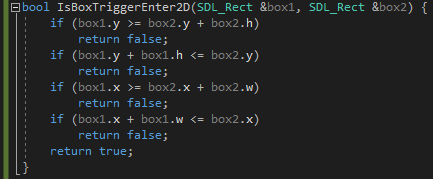


Fig . How an AABB (Axis-Aligned Bound Box) collision detection works

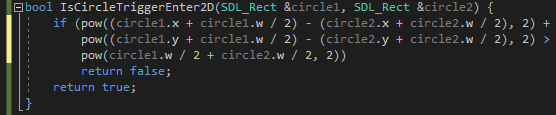


Fig . How a circle - circle collision detection works

# What we found

## Opt for performance

Some collision detection operations cost more than others. This means if you can afford to use a quicker, more simple operation before resorting to a more expensive one, you should use it. An example of this would be testing a collision between a circle and a star shape. A quicker and most likely easier method would be doing a circle – circle test instead of testing each side of the star against the circle.

## Test quickly

It’s recommended that your collision tests should finish (break, return) as soon as it proves that your shapes aren’t colliding. Since they can take a lot of performance depending on the complexity, the earlier the test can finish, the less time and resources it’ll take.

## Debug often

To help you understand the concept/logic quicker, you should do debug messages showing the stats of the shapes, or making visual changes when a collision happens. It also makes it easier to know when an error occurs.