**Collider**

Contents

[1. Introduction 2](#_Toc93933721)

[2. Collision Checks 2](#_Toc93933722)

[2.1. Point-Point 2](#_Toc93933723)

# Introduction

|  |  |  |
| --- | --- | --- |
| Type | Data structure | Image |
| Point | float x  float y  float z (3D) |  |
| Circle (2D) Sphere (3D) | float center x  float center y  float center z (3D)  float radius |  |
| AABB | float min x  float min y  float min z  float max x  float max y  float max z |  |
| Ray | float x  float y  float z (3D)  float vector x  float vector y  float vector z (3D) |  |
| Line |  |  |
|  |  |  |
|  |  |  |

# Collision Checks

Checking whether each type of object overlap each other

## Point-Point

## Point-Circle

### Steps



## Circle-Circle

* + - * 1st Get the distance between the 2 circles, which is also the length of the vector between the 2 centres of the 2 circles
      * Lastly, compare the distance by the sum of the radius of the 2 circles
      * If is the distance is less than the value, 2 circles have collided else not

Optimizing

* + - * In practise, finding the length/distance of a vector requires square root, which is computation expensive.
      * We learn that dot product of itself yield the square length of the vector, using this facts, we can optimize the equation