CGP600 AE1 – Group Project Report – James Moran

# Task Breakdown

## Game Object

This class is for all objects within a level (such as any hover-tanks, static obstacles etc.). Encapsulating similar elements for other classes to build on.

This class is to be used as a base (abstract) class, for all entities within a scene. This class will hence, have basic properties that can be applied to all sub-class instances (such as their current location, rotation and scale), as well as manipulation of these properties (e.g. by moving, rotating or transforming the game object). The scene class (as detailed by John), will have a collection of all game objects within it, so that one can find any particular game object, no matter what sub-class it belongs to.

## Hover-Tank

The Hover-tank class inherits from the GameObject class and is in of itself, the super class for all controllable entities within the game (whether they are controlled by the Player via external input, or by an AI system, with internal input from the game).

Any instance of this class can ‘float’ around the level, causing interactable obstacles to move or disperse, whilst static obstacles, will stop the tank and block movement into the area of such an object.

## Player Controlled Tank

The Player controlled tank is controlled by the Player, via external input (from a keyboard and mouse, or other control method such as a gamepad). This has the same functionality as the Hover-Tank base class, with the addition of being able to collect Energy Capsules (Collectibles), within the level area. If the Player collects these before time runs out, they will have completed the primary (and sole) objective for the level. The Player will have to attempt to avoid any static obstacles that get in their way, as well as any AI Controlled Tanks, that will actively try to harass the Player (causing them to lose time, that could be spent picking up the Energy Capsules in the level).

## AI Controlled Tank

The AI Controlled Tank is controlled by an AI System, granting input to this type of tank, internally. This sub-class also has the same functionality as the Hover-Tank base class, but as per the AI System this sub-class has, instances of this class will attempt to collide with the Player’s Hover-tank, interrupting them and wasting their time. The AI System of this class will have instances that simply move to the Player in a straight line, attempting to collide with them so as to waste the Player’s time.

## Static Obstacles

These are obstacles in the level, which are not movable by both the Player and any AI Controlled Tanks, blocking movement into their bounds, for both types of tanks. Both types of tanks will also have no interaction with these obstacles, other than colliding with them, to have their movement blocked by them.

## Moveable Obstacles

These are also obstacles in the level, which are moveable by both types of tanks (receiving translation to their position, or by getting scattered). These types of obstacles will also hinder movement to an extent (not by much if they are scatter-able, otherwise, they will impede movement appropriately), depending on the Hover-tank’s velocity.

## Collectibles

These are not obstacles (as they will not impede movement) and only interact with the Player Controlled Tank. These are known as Energy Capsules, with the primary objective for the Player, being that of collecting them. AI Controlled Tanks will have no interaction with them

## Heads-Up-Display (HUD)

This will not be a 3D-HUD (based within objects in the game world), but a 2D-HUD superimposed upon the Player’s viewport. This will show the Player how much time they have left, as well as how many Energy Capsules they have collected and how many Energy Capsules remain in the level, to be collected. This will be reset for every level that the Player is loaded into, as per the number of Energy Capsules in that particular level.

## Lighting

Lighting in the game will either exist on a per-object basis, for each GameObject. Or as a global ‘sky-light’ granting lighting to all GameObjects in a scene.

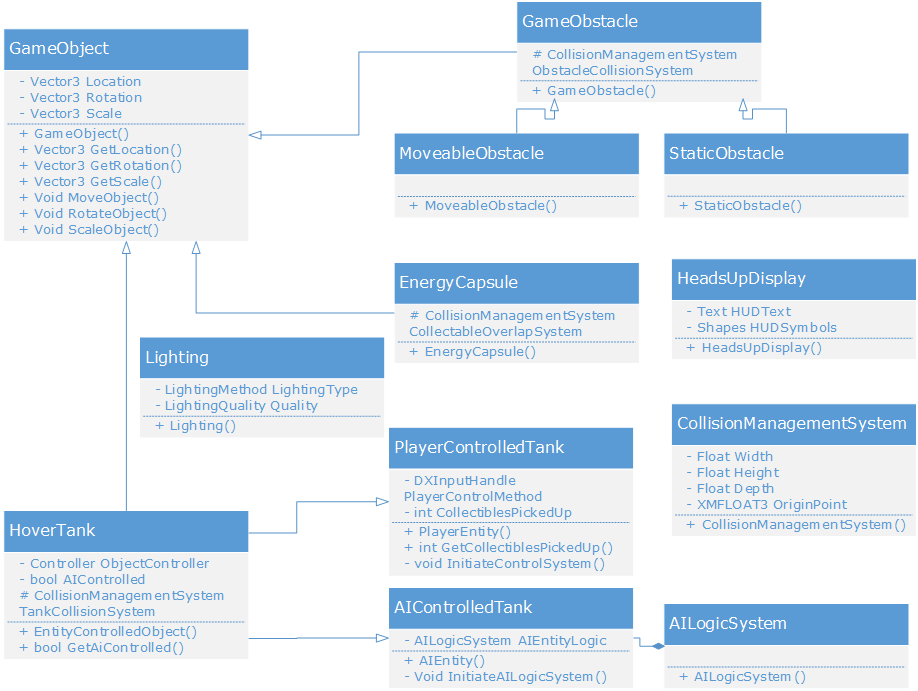
## Collision Management System

This is for handling collisions for each GameObject in the scene, that is to collide with or receive collision from other appropriate GameObjects (Any Hover-tanks, Static Obstacles and Moveable Obstacles), will be handled by Axis-Aligned Bounding Boxes (AABB).

# Critical Reflection and Discussion of Group Work

# (Sub) Appendix A: Class Diagrams

This contains all the classes that have been identified in the Task Breakdown section:

Figure 1: High Level Class Diagram 1.0.5

# (Sub) Appendix B: Class Method Pseudocode

This appendix is for the methods/functions of each class noted above:

## GameObject

**Method** GameObject() **is** Class Constructor

**Input:** *None.*

**Output:** *None.*

*Initialise a particular instance of this class*

**End Method**

**function** GetLocation() **is** a ‘getter’ function

**Input:** *None.*

**Output:** *Vector3 Location: This GameObject’s current location.*

*Return this GameObject’s Location member property*

**Return** Location

**End function**

**function** GetRotation() **is** a ‘getter’ function

**Input:** *None.*

**Output:** *Vector3 Rotation: This GameObject’s current rotation.*

*Return this GameObject’s Rotation member property*

**Return** Rotation

**End function**

**function** GetScale() **is** a ‘getter’ function

**Input:** *None.*

**Output:** *Vector3 Scale: This GameObject’s current scale.*

*Return this GameObject’s Scale member property*

**Return** Scale

**End function**

**Method** MoveObject() **is**

**Input:** *None.*

**Output:** *None*.

*Handle translation of this object*

**End Method**

**Method** RotateObject() **is**

**Input:** *None.*

**Output:** *None*.

*Handle rotation of this object*

**End Method**

**Method** ScaleObject() **is**

**Input:** *None.*

**Output:** *None*.

*Handle scaling of this object*

**End Method**

## MoveableObject

**Method** MoveableObject() **is** Class Constructor

**Input:** *None.*

**Output:** *None.*

*Initialise a particular instance of this class*

**End Method**

**Method** OnCollisionBegin() **is** a class event handler

**Input:** *None.*

**Output:** *None*.

*Handle the event of this object beginning collision with another object*

**End Method**

**Method** OnCollisionEnd() **is** a class event handler

**Input:** *None.*

**Output:** *None*.

*Handle the event of this object Ending collision with another object*

**End Method**

## EntityControlledObject

**Method** EntityControlledObject() **is** Class Constructor

**Input:** *None.*

**Output:** *None.*

I*nitialise a particular instance of this class*

**End Method**

**function** GetAIControlled **is** a ‘getter’ function

**Input:** *None.*

**Output:** *bool AIControlled: This flag indicates whether this object is controlled by the Player, or by an AI system.*

*Return this object’s AIControlled member property*

**Return** AIControlled

**End function**

### PlayerEntity

**Method** PlayerEntity() **is** Class Constructor

**Input:** *None.*

**Output:** *None.*

I*nitialise a particular instance of this class*

**End Method**

**Function** GetCollectiblesPickedUp() **is** a ‘getter’ function

**Input:** *None.*

**Output:** *int CollectiblesPickedUp: The number of collectibles that the Player has picked-up on this current level.*

**Return:** CollectiblesPickedUp

**End function**

**Method** InitiateControlSystem() **is**

**Input:** *None.*

**Output:** *None.*

*Initialise the control system of the input Method, given the control system the Player is using*

**End Method**

## AIEntity

**Method** AIEntity() **is** Class Constructor

**Input:** *None.*

**Output:** *None.*

I*nitialise a particular instance of this class*

**End method**

**Method** InitiateAILogicSystem **is**

**Input:** *None.*

**Output:** *None.*

*Initialise the logic system (FSM, Behaviour Tree etc.) that this AIEntity utilises*

**End method**

## AILogicSystem

**Method** AILogicSystem() **is** Class Constructor

**Input:** *None.*

**Output:** *None.*

I*nitialise a particular instance of this class*

**End method**

## PlayerHUD

**Method** PlayerHUD() **is** Class Constructor

**Input:** *None.*

**Output:** *None.*

I*nitialise a particular instance of this class*

**End method**

## CollisionManager

**Method** CollisionManager() **is** Class Constructor

**Input:** *None.*

**Output:** *None.*

I*nitialise a particular instance of this class*

**End method**

## AILogicSystem

**Method** AILogicSystem() **is** Class Constructor

**Input:** *None.*

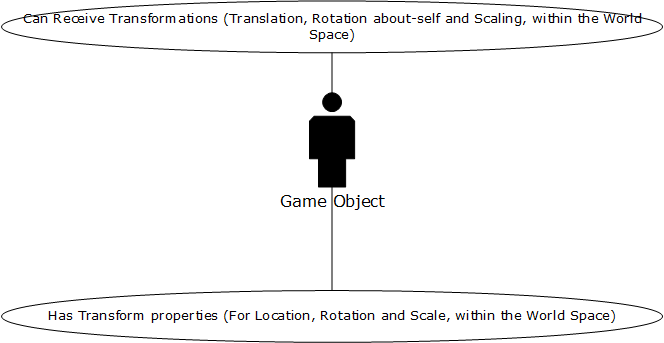
**Output:** *None.*

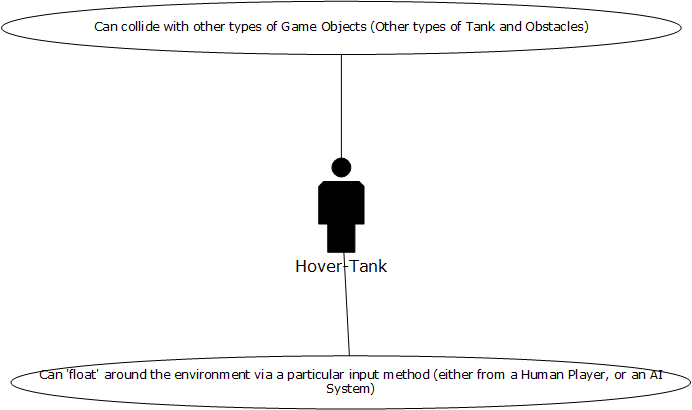
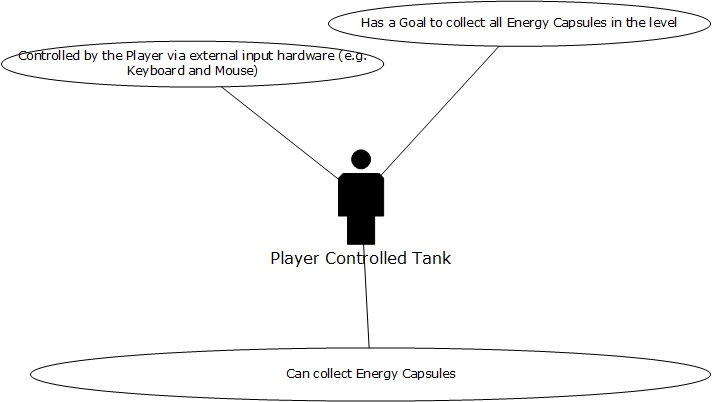
I*nitialise a particular instance of this class*

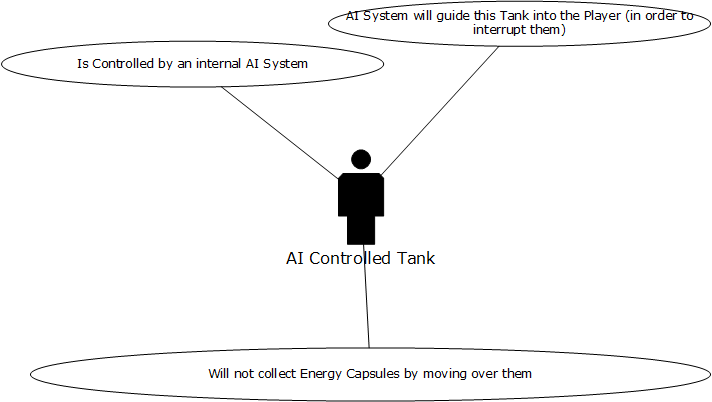
**End method**

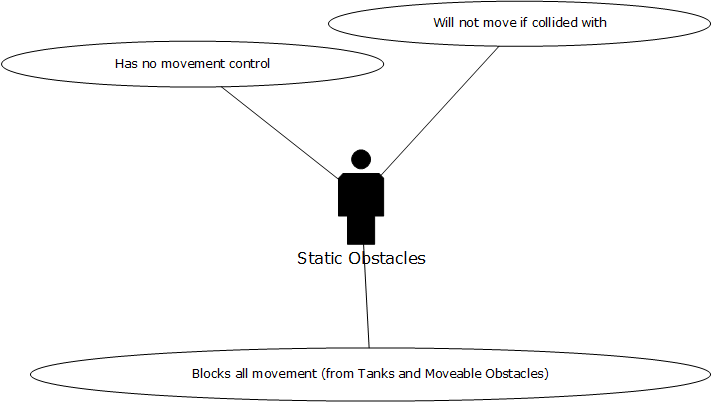
# (Sub) Appendix C: Use-Case Diagrams

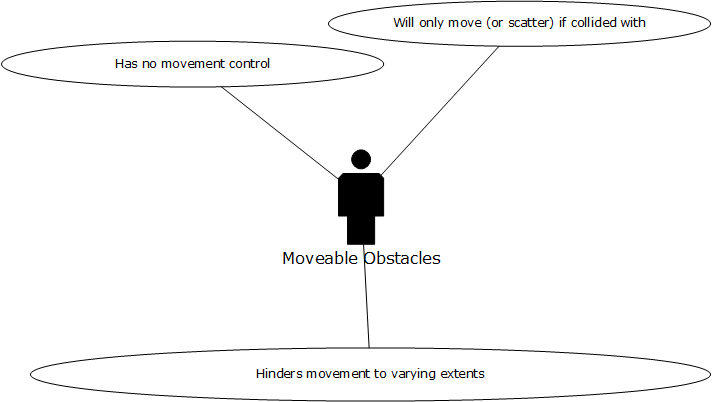
This appendix shows the Use-Case Diagrams for all the classes identified in the descriptions for each feature (User-Story):

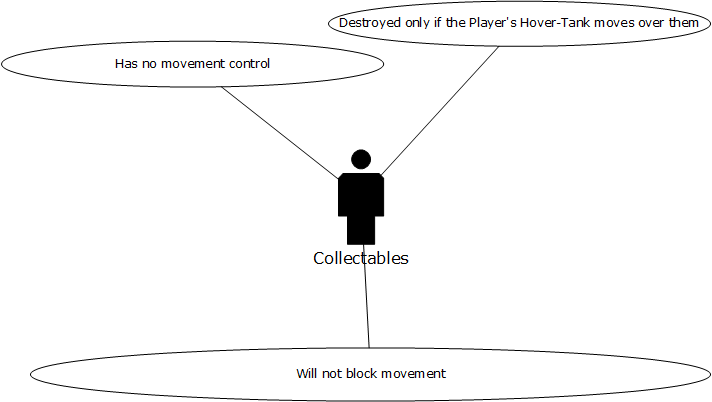
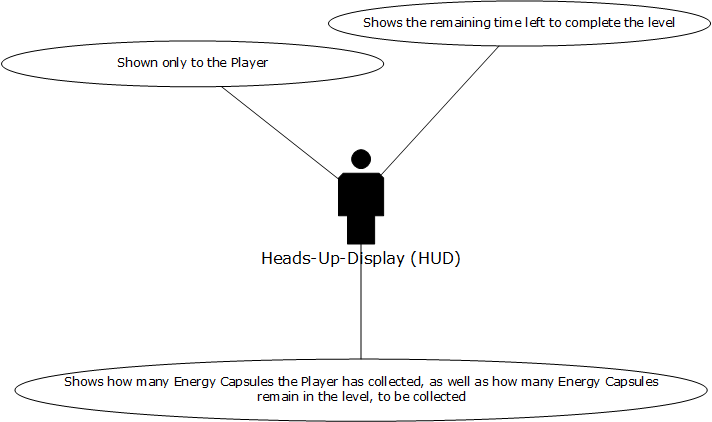


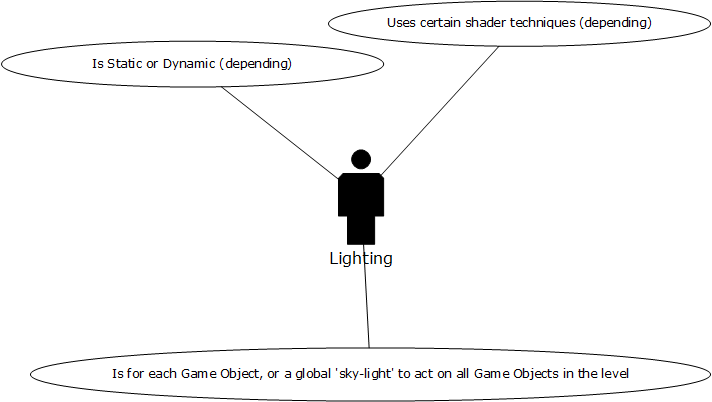
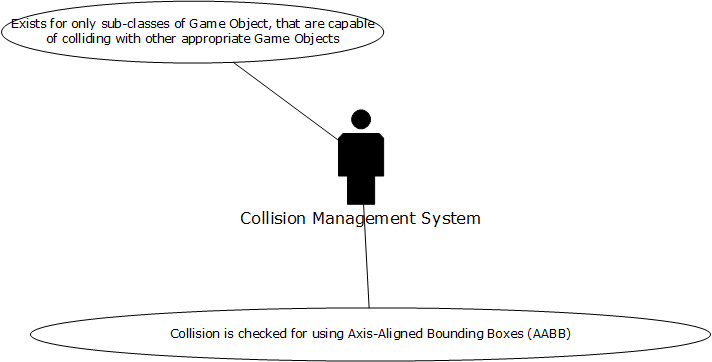












# References

1. James Moran, 2017. *Screenshot of the nodes in a High-Level Class Diagram 1.0.4* [Digital Image] [Viewed on the 18/10/2017] Available from: <https://drive.google.com/open?id=0B1bkA1bSHsaXWUppSDJyeExUMUE>
2. Wikipedia, 2017, Wikipedia: WikiProject Computer science/Manual of Style [Viewed on the 14/10/2017]. Available from: <https://en.wikipedia.org/wiki/Wikipedia:WikiProject_Computer_science/Manual_of_style#General_guidelines_for_writing_pseudocode>