# Design Overview for <<Multi Tank >>

Name: Nghia Hieu Pham

Student ID: 103533868

# Summary of Program

**Describe what you want the program to do… one or two paragraphs.**

**Include a sketch of sample output to illustrate your idea.**

My program is a 2-player tank shooting game called “Multi Tank”. Each player will have to defeat each other by shooting bullets or using items. Each player can fire a maximum of 5 bullets appearing on the screen at the same time. After each match, the score is shown to the players.

There are 4 items, and each item has a duration of effect:

+ Speed Boost: Boost the player’s speed

+ Fast Bullet: Increase the bullets’ speed

+ Mini Gun: Increase the player’s maximum bullet to unlimited

+ Mystery Box: Get a random item from the 3 items above

There are a total of 4 screens: start screen, how-to-play screen, game screen, and score screen.

# Required Roles

Describe each of the classes, interfaces, and any enumerations you will create. Use a different table to describe each role you will have, using the following table templates.

Table 1: <<Player(abstract)>> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| \_x, \_y | protected, double | Position |
| \_angle: double | protected, double |  |
| \_forwardspeed, \_backwardspeed | protected, double |  |
| \_rotation speed | private, double |  |
| \_maxammo | protected, int |  |
| \_score | protected, int |  |
| \_tankImage | protected, Bitmap |  |
| \_ammo | protected, List<Bullet> |  |
| \_items | protected, List<Item> |  |
| \_dead | protected, bool | Check status of player |
| Player() | public | Constructor |
| RotateRight() | public, void |  |
| RotateLeft() | public, void |  |
| UseItem() | public, void |  |
| Shoot() | public, void |  |
| Control() | public, void |  |
| Hit() | public, void |  |
| Draw() | public, void, abstract |  |
| Ammo : List<Bullet> | public, readonly property |  |
| Items: List<Item> | public, readonly property |  |
| getImage: Bitmap | public, readonly property |  |
| Deafeat: bool | public, property |  |
| Score: int | public, property |  |
| X: double | public, property |  |
| Y: double | public, property |  |
| MaxAmmo: int | public, property |  |
| ForwardSpeed: double | public, property |  |
| BackwardSpeed: double | public, property |  |

Table 2: <<Player1>> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| Player1() | public | Constructor |
| Draw() | public, void |  |
| Control() | public, void |  |

Table 3: <<Player2>> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| Player2() | public | Constructor |
| Draw() | public, void |  |
| Control() | public, void |  |

Table 4: <<Bullet>> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| \_x, \_y | private, double | Position |
| \_speed | private, double |  |
| \_angle | private, double |  |
| \_bulletImage | private, Bitmap |  |
| Bullet(double x, double y, double angle) | public | Constructor |
| Move() | public, void |  |
| Draw() | public, void |  |
| getImg: Bitmap | public, readonly property |  |
| Speed: double | public, property |  |
| X: double | public, property |  |
| Y: double | public, property |  |

Table 5: <<Item(abstract)>> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| \_x, \_y | private, double | Position |
| \_ItemImg | protected, Bitmap |  |
| \_duration | protected, int |  |
| \_bulletImage | private, Bitmap |  |
| Item(double x, double y) | public | Constructor |
| DrawOutline() | public, void |  |
| Draw() | public, void |  |
| Effect(Player p) | public, void, abstract |  |
| ReverseEffect(Player p) | public, void, abstract |  |
| getImg: Bitmap | public, readonly property |  |
| Duration: int | public, property |  |
| X: double | public, property |  |
| Y: double | public, property |  |

Table 6: <<CollisionHandler>> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| + CollisionHandler() | public | Constructor |
| CheckCollision(Player p1, Player p2, List<Item> items) | public, void |  |
| CheckBulletHit(Player p1, Player p2) | public, void |  |
| CheckItemCollision(Player p, List<Item> items> | public, void |  |
| CheckBulletOutOfScreen(Player p) | public, void |  |
| CheckItemOutOfScreen(Player p) | public, void |  |

Table 7: <<Page>> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| \_background | protected, Bitmap |  |
| \_duration | protected, int |  |
| + Page() | public | Constructor |
| Draw() | public, void, virtual |  |
| ButtonClicked(double x, double y) | public, void, virtual |  |
| Duration: int | public, property |  |

Table 8: <<GameScreenManager>> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| \_currentscreen | private, Page | The current screen of the game |
| + GameScreenManager(Page screen) | public | Constructor |
| ChangeScreen(double x, double y, Player p1, Player p2) : | public, void |  |
| Draw() | public, void |  |
| getPage : Page | public, readonly property |  |

Table 9: <ExtensionMethod(static)>> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| ToRadian(double number) | public, double, static | This functions converts degree to radian |

Table 10: <ScreenSize(sealed)>> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| size | private, ScreenSize, readonly, static | The only instance of the class |
| ScreenSize() | internal, static |  |
| ScreenSize() | private | Constructor |
| Size: ScreenSize | public, readonly, static, property |  |
| WIDTH | public, const int |  |
| HEIGHT | public, const int |  |

Table 11: <MultiTank>> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| \_player1 | private, player |  |
| \_player2 | private, player |  |
| \_gamescreenmanager | private, GameScreenManager |  |
| \_collision | private, CollisionHandler |  |
| \_items | private, List<Item> |  |
| + MultiTank() | public | Constructor |
| Draw() | public, void |  |
| Update() | public, void |  |
| SpawnItem() | public, void |  |
| UpdateScore() | public, void |  |
| NewRound() | public, void |  |

Table 12: <Program(static)>> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| \_game | MultiTank, static |  |
| window | Window, static |  |
| LoadResource() | public, void, static | Load game images |
| Main() | public, void, static | Execute game |

# Class Diagram

Diagram, engineering drawing

Description automatically generatedProvide an initial design for your program in the form of a class diagram.

# Sequence Diagram

Provide a sequence diagram showing how your proposed classes will interact to achieve a specific piece of functionality in your program.

Diagram

Description automatically generated