# Design Overview for <<TopDownGame: Orb Shooter>>

[Note: this is the final version for the game]

Name: SM Ragib Rezwan

Student ID: 103172423

# Summary of Program

Basically it’s a TopDownShooter/ survival type game. Player will move around the screen, listening to music and avoiding different types of enemies (each with different behaviour) that spawn while trying to shoot them down using different types of projectiles. Each enemy killed gives 1 point to player. Also when enemy collides with player, enemy dies and player loses 1 health. Player has 5 health in total and when all health has been depleted, its game over and number of points player got is stated.



# Required Roles

Roles that I am using:

1 DrawOnly

2 Entity

3 Player

4 ActiveGameLogic

5 Endscreen

6 MainMenu

7 Button

8 Program

9 Game1

10 SoundPlayer

11 Song1

12 SoundEffect1

13 IAudio

14 Enemy

15 EnemyNormal

16 EnemyHoming

17 EnemySpawner

18 Projectile

19 ProjectileManager

20 ProjectileNormal

21 ProjectileSlow

22 CollisionDetector

23 EnemyProjectileCollisionResponder

24 EnemyPlayerCollisionResponder

25 KeyboardController

26 ICommand

27 DownCommand

28 LeftCommand

29 NulCommand

30 QuitCommand

31 RightCommand

32 UpCommand

33 Global (to store all constant stuff)

[Note: There are several other places that can be made into singleton, but I didn’t do so as then it might be too much]

Table 1: <<DrawOnly>> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| DrawOnly | Constructor,  Takes in parameters  Vector2 position, Vector2 dimension, string address, Game1 game | It is used as a parent class that other objects can use as template to make themselves.  Sets position, dimension and also how to load texture from address using game.Content.load.texture2D>  Also sets isExpired as false and stores game into \_game |
| Rotation | Property, return float \_rotation and sets float \_rotation value | It is use to get and set the rotation of the object |
| IsExpired | Property, returns int \_IsExpired and sets int \_IsExpired | It is used to get and set the IsExpired value of the object |
| Pos | Property, returns Vector2\_pos and sets Vector2 \_pos | It is used to get and set the position of the objects created |
| Dim | Property, returns Vector2 \_dim and sets Vector2 \_dim | It is used to get and set the dimension of the object |
| Draw | Virtual method,  Returns void | This mainly sets the conditions to draw the object properly with all the conditions set in a single line (see the comment written to understand what each part does). When other object will call base.draw(), they will be able to utilize this method to draw themselves |

Table 2: <<Entity>> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| Entity | Constructor,  Takes in parameters  Vector2 position, Vector2 dimension, string address, Game1 game | It is used base to make itself.  Also it is going to pass game to \_game |
| Speed | Property, returns float \_speed and sets float \_speed | It is used to get and set the speed of the object |
| Health | Property, returns int \_health and sets int \_health | It is used to get and set the health of the object |
| SoundPlayer | Property, returns SoundPlayer from game’s soundplayer and sets SoundPlayer to game’s soundplayer | It is used to get and set the sounds of the object (if they need any) |
| Direction | Property, return vector2 \_direction and sets vector2 \_direction value | It is use to get and set the direction of the object |
| Update | Virtual method,  Takes in parameter GameTime gametime,  Returns void | Mainly will let the child classes override this to update themselves in their desired way |

Table 3: <<Player> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| Player | Constructor,  Takes in parameters  Vector2 position, Vector2 dimension, string address, Game1 game | It is used base to make itself.  Also it is going to set player speed, health, collideddamage, lastshot, direction and also projectile manager |
| LastShot | Property, returns double \_lastshot and sets double \_lastshot | It is used to get and set the lastshot value |
| CollideDamage | Property, returns int \_collideddamage and sets int \_collideddamage | It is used to get and set the collided damage value  (ie the damage player will do to enemy when collided with it) |
| ProjectileManager | <<readonly>>Property, return ProjectileManager \_projectileManager | It is used to get the Projectile Manager |
| Point | <<readonly>>Property, return int \_point and set int \_point | It is used to get the Point value |
| HitEnemy | Method, returns void | Reduce player \_health by 1, if health below zero, then play player killed sound by soundplayer and make player isExpired true. |
| GoRight | Method, returns void | Changes player position to right by using its speed |
| GoLeft | Method, returns void | Changes player position to left by using its speed |
| GoUp | Method, returns void | Changes player position to up by using its speed |
| GoDown | Method, returns void | Changes player position to down by using its speed |
| Update | override method,  Takes in parameter GameTime gametime,  Returns void | If player is not expired, it will tell \_projjectilemanager to update itself using gametime, and also will use base to update itself as well |
| Draw | override method,  Returns void | If player is not expired, it will tell \_projjectilemanager to draw itself, and also will use base to draw itself as well |
| GetInstance | Static method, Takes in Parameter Game1 game, returns Player | It will create an instance of the player class using singleton design pattern |

Table 4: <<ActiveGameLogic> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| ActiveGame1Logic | Constructor,  Takes in parameters  Game1 game | It is going to get a player instance, create enemyspawner, store game into \_game, create keyboardinput, create collisiondetector, create gamescreen (ie background pic) |
| Update | method  Takes in GameTime gameTime,  Returns void | As long as player Isexpired is false, It is used to do all of these:   * tell enemyspawner to update itself * Loads PlayerFaceMouse method by passing the parameter \_player * Logic to determine when to let player shoot and what (ie mouse click, shootdelay and type of projectile shot), * Tell collision detector to detect whether enemy collided with player or projectile or not * Update player * Update keyboard input * Store player’s point into current player point and player health into current player health   Else  Set \_gameScreen.IsExpired to true |
| Player | <<readonly>>Property, returns Player | It is used to get player |
| PlayerFaceMouse | Method  Takes in parameter Player \_player | It is used to make player face the mouse and rotate towards it by using mouse’s X and Y position as a vector, subtracting it from player position to find difference, then using mathAtan to get the angle of rotation by using X and Y position of the difference (casted as float) and using cos and sin parts of rotation to set direction player will face (again casting as float) |
| Draw | virtual method,  Returns void | Tells player to draw itself, tells enemyspawner to draw itself  And when gameScreen is not Expired: draw gamescreen, and text for player health and player point counter |

Table 5: <<EndScreen> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| EndScreen | Constructor,  Takes in parameter Game1 game, Player player | It takes game and passes to \_game, takes player and sets it to \_player, loads gameScreen (background picture for Endscreen), loads quit button |
| Update | method | Constantly checks if quit button pressed or not  If quit button has been selected, then exit the game |
| Draw | method | Draws gameScreen, text saying points player has now, and the quit button |

Table 6: <<MainMenu> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| MainMenu | Constructor,  Takes parameters Game1 game | It takes game and passes to \_game, plays song Mainmenu song, loads gameScreen (background picture for MainMenu), loads start and quit button, sets game trigger as false |
| Update | Method  Takes parameter GameTime gametime | Constantly checks if start or quit has been pressed or not  If newGameLogic is not null,update newGameLogic, make start,quit, gameMenuScreen and game trigger as true. Now if player is not null, but is expired and endscreen is null, then plays music for endscreen and loads endscreen  If endscreen is not null, update it  If game trigger is false, but start is true, then load activegame and play active game song  If game trigger is false, but quit game is true, exit from game |
| Draw | method, | Draw start and quit  If gameMenuScreen is not expired, draw it  If newGameLogic is not expired , then draw it  If endScreen is not expired, draw it |

Table 7: <<button> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| Button | Constructor,  Takes in parameters  Vector2 position, Vector2 dimension, string address, Game1 game | It is used base to make itself.  Also it is going to set \_buttonSelect as false |
| ButtonSelect | <<readonly>>Property, returns bool \_buttonSelect | It is used to get buttonSelect for the button |
| Draw | Override method | It button is not expired, draw it using base |
| ButtonClick | method | Stores mouse’s x and y position as cursor position  Then it checks whether cursor is in button’s box or not when left button was selected, if so makes \_buttonSelect as true |

Table 8: <<Program> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| Main | Constructor, | Gets a game1 instance and tells game to run |

Table 9: <<Game1> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| Game1 | Constructor, | It is used to make a new instance of graphics manager, set height and width of game, set the root directory for image and audio to content folder, make mouse visibility true |
| Initialize | Override method,  Return void | It is used to initialize game1 using parent class game (prebuilt in library) |
| LoadContent | Override method,  Return void | Loads in new instance of spritebatch, soundplayer, spritefont and also makes new instance of mainMenu |
| Update | Override method,  Takes in parameter Game1Time gameTime  Returns void | It makes mainMenu update itself using gametime parameter and also it updates itself using the same parameter with help of parent class game |
| Draw | Override method,  Takes in parameter Game1Time gameTime  Returns void | Clears device screen and loads Black colour, Draws all sprites stored in global spritebatch, tells mainMenu to draw itself, ends spritebatch drawing, draws itself using gametime parameter |
| SpriteBatch | <<readonly>> property, returns \_spriteBatch | Used to get the spritebatch |
| SoundPlayer | <<readonly>> property, returns \_soundplayer | Used to get the soundplayer |
| SpriteFont | <<readonly>> property, returns \_spriteFont | Used to get the spriteFont |
| GetInstance | Static method,  returns Game1 | It will create an instance of the game1 class using singleton design pattern |

Table 10: <<SoundPlayer> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| SoundPlayer | Constructor,  Takes in parameter Game1 game | Creates dictionaries for song1 and soundeffect1, creates all needed song1 and soundeffect1 using helper class and stores them in dictionaries for central management and easy use. |
| PlaySong | Method,  Takes parameter string smth  Returns void | It is used to search though songlibrary dictionary to play a certain song using helper class |
| PlaySoundEffect | Method,  Takes parameter string smth  Returns void | It is used to search though soundEffectlibrary dictionary to play a certain soundEffect using helper class |
| StopMusic | Static method,  Returns void | Stops whatever background song is being played |

Table 11: <<Song1>> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| Song1 | Constructor,  Takes in parameter string address, Game1 game | Creates a song instance by using game.Content.Load <song> and passing in where song is being stored |
| Play | Method,  Returns void | It is used to lower volume of mediaplayer, using it to play a song, make it repeat over when finished |

Table 22: <<SoundEffect1>> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| SoundEffect1 | Constructor,  Takes in parameter string address, Game1 game | Creates a soundEffect instance by using game.Content.Load <soundeffect> and passing in where soundeffect is being stored |
| Play | Method,  Returns void | It is used to play the sound effect |

Table 13: <<IAudio> interface details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| Play | Interface method that returns void | Mainly used it to ensure song1 and soundeffect always have Play method that returns void.  (This can be used to put them both in a single dictionary instead of putting in two different dictionary, but I am not doing that as I prefer to keep song and soundeffects separate) |

Table 14: <<Enemy> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| Enemy | Constructor,  Takes in parameters  Vector2 position, Vector2 dimension, string address, Game1 game | It is used base to make itself. |
| WasHit | Method,  takes in parameter int  returns void | Decreases health with an int value that is passed to the method.  If health is less or equal to 0, turn \_isExpired into true and tell soundplayer to play enemykilled sound |
| Update | override method,  Takes in parameter Game1Time gametime,  Returns void | Will update itself using its base if \_isExpired is set to false. |
| Draw | override method,  Returns void | Will draw itself using its base if \_isExpired is set to false. |

Table 15: <<EnemyNormal> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| EnemyNormal | Constructor,  Takes in parameters  Vector2 position, Vector2 dimension, string address, Game1 game | It will follow template in base to make itself.  Also it is going to set its own speed and health values and set \_oldMoveTime as 0 |
| CalculateNewMove | Method  Returns void | It is used to create a random x and y value between -5 to 5, to give enemy randomness in movement |
| EnemyHitScreen | Method,  Returns Bool | Condition used to see whether it hits wall or not from any side  (modified it so that it will match with walls given in picture instead of screen) |
| Update | override method,  Takes in parameter GameTime gametime,  Returns void | Will update itself using its parent class.  Then it will check if certain time has been passed since it last made correction to its movement (using a move timelimit set in global).  If so, it will CalculateNewMove (changing its x and y parts of vector added to its position) and store the currentgametime into oldmovetime to recalculate later on.  Gives enemy’s general movement  Then it adds vector of x and y part multiplied with its speed to position to change its position.  Also, if it hits screen (enemyHitScreen) then alter x and y velocity to make it hit wall and return back |

Table 16: <<EnemyHoming> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| EnemyHoming | Constructor,  Takes in parameters  Vector2 position, Vector2 dimension, string address, Game1 game | It will follow template in parent class to make itself.  Also it is going to set its own speed and health values  And also make an instance of activelogic by passing in game |
| Update | override method,  Takes in parameter GameTime gametime,  Returns void | Will update itself using its parent class.  Then it update its position by using position of player in activelogic and its own speed  (beneficial for player as now if enemy is far, it will run to player but if its close it will slowly creep towards the player making it easier to player to shoot it) |

Table 17: <<EnemySpawner> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| EnemySpawner | Constructor,  Takes in parameters  Game1 game | It makes a new enemy list, sets oldspawntime to 0 create a list for removeEnemy, pass game to its own \_game field |
| EnemyList | <<readonly>> Property, returns List<Enemy>\_enemyList | It is used to get Enemylist |
| SpawnEnemyNormal | Method  Takes in parameters int x, int y, Player \_player  Returns void | If x and y passed is not same as player position then make normal enemy and add it to enemylist |
| SpawnEnemyHoming | Method  Takes in parameters int x, int y, Player \_player  Returns void | If x and y passed is not same as player position then make Homing enemy and add it to enemylist |
| Update | override method,  Takes in parameter GameTime gametime,  Returns void | Uses similar technique to that done for movement in enemynormal. Here it just adds random aspect when choosing what type of enemy to spawn.  Also if isexpired is true for any enemy, then move it to removeEnemyList, add 1 point to player’s point counter and them remove it from \_enemyList |
| Draw | Method,  Returns Bool | Tells all enemies present in draw to draw themselves |

Table 18: <<Projectile> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| Projectile | Constructor,  Takes in parameters  Vector2 position, Vector2 dimension, string address, Game1 game | It is uses base to make itself. |
| HitEnemy | Method  Return void | It is used to make \_isExpired into true |
| Damage | Property, returns int \_damage and sets int \_damage | It is used to get and set the damage value of projectile |
| Update | override method,  Takes in parameter Game1Time gametime,  Returns void | Checks if projectile’s position is outside the walls given in picture, if so makes projectile isExpired into true.  If projectile is not expired, it will update itself using its parent class and also set its position using speed and direction |
| Draw | override method,  Returns void | Will draw itself using its parent class if \_isExpired is set to false. |

Table 19: <<ProjectileManager> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| ProjectileManager | Constructor,  Takes in parameters  Game1 game | It is used to make 2 lists:\_projectileList, \_remoprojectile. And also stores game into \_game |
| ProjectileList | <<readonly>> Property, returns List<Projectile> \_projectileList | It is used to get projectilelist |
| NormalBullet | Method,  Takes in parameter Player player  Returns void | It is used make a projectilenormal using helper class, set its directions using player direction and add it to list of projectile |
| SlowBullet | Method,  Takes in parameter Player player  Returns void | It is used make a projectileSlow using helper class, set its directions using player direction and add it to list of projectile |
| Update | method,  Takes in parameter Game1Time gametime,  Returns void | Will tell each projectile its list to update itself, but if a projectile has isExpired set to true, it will add it to removeProjectileList.  Also it will remove all projectiles present in removeProjectileList |
| Draw | method,  Returns void | Will tell all projectiles in projectilelist to draw themselves |

Table 4: <<ProjectileNormal> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| ProjectileNormal | Constructor,  Takes in parameters  Vector2 position, Vector2 dimension, string address, Game1 game | It is used to make projectileNormal. Also its sets damage and speed of it using values preset for it in global |

Table 21: <<ProjectileSlow> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| ProjectileSlow | Constructor,  Takes in parameters  Vector2 position, Vector2 dimension, string address, Game1 game | It is used to make projectileSlow. Also its sets damage and speed of it using values preset for it in global |

Table 22: <<CollisionDetector> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| CollisionDetector | Constructor, | It is used to create two responders: \_enemyPLayerCollideResponder,  \_enemyProjectileCollideResponder |
| Detect | Method,  Takes in parameters Player player, List<Enemy> enemyList, List<Projectile> projectileList | It is used to detect whether minimum collision distance (set in global) has been reached between either enemy and player or enemy and projectile and if so calls their respective responder |

Table23: <EnemyProjectileCollisionResponder> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| EnemyProjectileCollisionResponder | Constructor, | Does nothing  Only kept it for good practise and ensure it doesn’t go wrong in any way |
| EnemyProjectileCollide | Method,  Takes in parameters Projectile p, Enemy e | It is used to make projectile call hitenemy method and enemy to call washit method while passing in projectile’s damage as parameter to the washit method |

Table 6: <EnemyPlayerCollisionResponder> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| EnemyPlayerCollisionResponder | Constructor, | Does nothing  Only kept it for good practise and ensure it doesn’t go wrong in any way |
| EnemyPlayerCollide | Method,  Takes in parameters Player p, Enemy e | It is used to make player call hitEnemy method and enemy to call washit method whole passing in projectile’s damage as parameter to the washit method.  (currently set player collide damage very high to kill all enemy with 1 collide at the expense of 1 health) |

Table25: <<KeyboardController> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| KeyboardController | Constructor,  Takes in parameter Player player | It is used to set \_player as player, make a new dictionary called \_commandLibrary and add keys and commands to the dictionary (after making the commands)  It follows the strategy pattern and I am using it here to bind the keys to the commands |
| Update | method,  Returns void | Will set \_currentCommand as nullCommand, then it will get keyboardstate and store it as keyboard state. Then it will use the key to look though its command library and find the command and set it as \_current command and execute it |

Table 7: <<ICommands> <<interface>> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| Execute | Method  Returns void | It will be used to ensure all the classes that use it will have an Execute method that takes no parameter and returns void.  (here is used it as part of strategy pattern and added all of the commands into a single dictionary) |

Table 8: <<DownCommand> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| DownCommand | Constructor  Takes parameter Player player | It is used to set \_player as player |
| Execute | Method  Returns void | If player is not touching bottom of screen then tell player to GoDown  (calibrated it with respect to image in activelogic) |

Table 9: <<LeftCommand> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| LeftCommand | Constructor  Takes parameter Player player | It is used to set \_player as player |
| Execute | Method  Returns void | If player is not touching left of screen then tell player to GoLeft  (calibrated it with respect to image in activelogic) |

Table 10: <<NullCommand> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| NullCommand | Constructor | Does nothing. Only used it as placeholder for current command in keyboardController |
| Execute | Method | Does nothing. Only used it as placeholder for current command in keyboardController |

Table 11: <<QuitCommand> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| QuitCommand | Constructor | Does nothing. Only kept it for good practise and to ensure it doesn’t go wrong in any way |
| Execute | Method  Returns void | Exits code |

Table 12: <<RightCommand> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| RightCommand | Constructor  Takes parameter Player player | It is used to set \_player as player |
| Execute | Method  Returns void | If player is not touching right of screen then tell player to GoRight  (calibrated it with respect to image in activelogic) |

Table 13: <<UpCommand> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| UpCommand | Constructor  Takes parameter Player player | It is used to set \_player as player |
| Execute | Method  Returns void | If player is not touching top of screen then tell player to GoUp  (calibrated it with respect to image in activelogic) |

For Global, it is used to basically keep track of all the constants I have used in my code:

(although both player and enemy height and width are current kept same, I am using different variables to store the current constants as I may edits the values later on to make the enemy bigger or smaller than the player)

**[Note: it currently does not have the new constants that have been added due to background picture added in activegame logic!!!]**

public const int screenHeight = 800;

public const int screenWidth = 800;

public const int projectileHeight = 20;

public const int projectileWidth = 20;

public const int projectileRotationVelocity = 3;

public const int projectileLinearVelocity = 4;

public const double shootDelay = 0.15;

public const int projectileSlowDamage = 2;

public const int projectileSlowSpeed = 5;

public const int projectileNormalDamage = 1;

public const int projectileNormalSpeed = 10;

public const int enemyHeight = 50;

public const int enemyWidth = 50;

public const double moveTimeLimit = 2;

public const double spawnTimeLimit = 1;

public const float enemyNormalSpeed = 1;

public const int enemyNormalHealth = 1;

public const float enemyHomingSpeed = 0.005f;

public const int enemyHomingHealth = 5;

public const int playerHeight = 50;

public const int playerWidth = 50;

public const int playerSpeed = 2;

public const int playerHealth = 5;

public const int playerCollideDamage = 10;

public const int objectTouchDistance = 50;

I didn’t use any enums in my code. Instead I used lists and dictionaries which I have already mentioned in their specific classes.