Design for Joint Project 1

(Programming & Graphics)

## **Class Structure**

### **Player Class**

### // the player in the game

### **Data members:**

sf::Texture texture; // Texture for player

sf::Sprite sprite; // sprite used to represent player

int health; // the health of the player

int speed; // the speed of the player

int score; // the player score

int direction; // the direction the player is facing

bool isAlive; //To see if the player is still alive

### **Function members:**

Player() //constructor

draw() //used to draw the player

update() //updates every 60 frames

shoot() //function to cause bullet to shoot

dies() // the player dies

decreaseHealth() // reduce player health

moveLeft() // Used to move the player left

moveRight()

moveUp()

moveDown()

boundaryCheck( ) //make sure the player does not go outside the screen boundary.

displayPlayerStats( ) // display the player health & score on the screen

increaseScore() // the player’s score increases

decreaseScore() // the player’s score decreases

### **EnemyFollow Class:**

### // this enemy follows the player

### **Data members:**

sf::Texture texture; // Texture for enemy

sf::Sprite sprite; // sprite used to represent enemy

int speed; // the speed of the enemy

int timesHit; // the number of times hit

int direction; // the direction the enemy is facing

bool isAlive; //To see if the enemy is still alive

### **Function members:**

EnemyFollow() //constructor

draw() //used to draw the enemy

update() //updates every 60 frames

hit( ) // enemy is hit by a bullet

dies() // enemy dies

reSpawn // the enemy respawns somewhere randomly on the screen

move () // the enemy follows the player

boundaryCheck( ) //make sure the enemy does not go outside the screen boundary.

### **EnemyLeftRight Class:**

### // this enemy moves left and right

### **Data members:**

sf::Texture texture; // Texture for enemy

sf::Sprite sprite; // sprite used to represent enemy

int speed; // the speed of the enemy

int direction; // the direction the enemy is facing

bool isAlive; //To see if the enemy is still alive

### **Function members:**

EnemyLeftRight() //constructor

draw() //used to draw the enemy

update() //updates every 60 frames

hit( ) // enemy is hit by a bullet

dies() // enemy dies

reSpawn // the enemy respawns to a fixed position on the screen

move () // the enemy moves left to right on the screen

boundaryCheck( ) //make sure the enemy does not go outside the screen boundary.

### **Bullet Class:**

### // the bullet used by the player

### **Data members:**

sf::Texture texture; // Texture for bullet

sf::Sprite sprite; // sprite used to represent bullet

int speed; // the speed of the bullet

int direction; // the direction of the bullet

bool isAlive; //To see if the bullet is still alive

### **Function members:**

Bullet() //constructor

draw() //used to draw the enemy

update() //updates every 60 frames

hit( ) // the bullet has hit something

fired() // the bullet is fired

move() // the bullet moves

boundaryCheck( ) //make sure the bullet does not go outside the screen boundary.

### **Game Class:**

### // The game class controls the game.

### // Note how the collision detection is done in the Game class.

### **Data members:**

Player thePlayer; // the player in the game

Bullet playerBullet; // the player’s bullet in the game

int maxEnemies = 10; // the maximum number of EnemyFollow objects allowed in the game

int noEnemies; // the number of current EnemyFollow objects in the game

EnemyFollow enemyFollowArray[maxEnemies]; // an array of enemies of type follow

EnemyLeftRight enemyOne; // 3 enemies of type leftRight

EnemyLeftRight enemyTwo;

EnemyLeftRight enemyThree;

Bool gameOver; // true if the game is over

sf::RenderWindow window; //game window

sf::Font m\_font; // font for writing text

sf::Text m\_message; // text to write on the screen

### **Function members:**

setUpGame() //sets up player and enemies etc for a new game

loadContent() //load the game font etc

run() // contains the main game loop

update() //updates every 60 frames

draw() //draws the game world

collisionDetection( ) // checks for collisions between the player and enemies

bulletCollisionDetection( ) // checks for collisions between the bullet and enemies

isGameOver () // checks if the game is over

displayGameStats() //display player health and score to the screen