COMPUTER & INFORMATION SCIENCES (CIS)

FBH0013 OBJECT ORIENTED PROGRAMMING

No	Student Name	Student ID	Course
1.	Ahmad Akif Bin Mohamad Afifi	22010192	Information Technology
2.	Iman Zafran Bin Hasril Anuar	22010238	Information Technology
3.	Nur Anis Farhana Binti Amir	22010425	Information Technology
4.	Nur Sofianisa Binti Shahruddin	22010285	Information Technology
5.	Shachindran A/L Veerangan	22010295	Information Technology
6.	Nur Anis Hidayah Binti Saifulasnoor	22010267	Information Technology

Contents

1.0 Background of Study

For the course **TFB1033/TEB1043: Object-Oriented Programming**, we are tasked with developing a **swarm game**, where a large group of enemies or units (known as a "swarm") moves and acts in together. The game can be built using **Unity**, a popular game development engine. The objective is to apply the core principles of **Object-Oriented Programming (OOP)**, such as classes, objects, inheritance, and encapsulation, to structure the game logic and behaviour effectively. By doing so, we not only enhance our programming skills but also gain experience in both game and web development.

In the case of **Unity**, the game will involve scripting enemy behaviour, creating player interactions, and managing game mechanics such as health, movement, and attacks. We will use **C#** to write scripts that control the swarm's movement patterns, simulate realistic behaviours like pathfinding or coordinated attacks, and integrate user inputs.

This project provides practical exposure to the application of OOP principles in real-world scenarios. By designing a swarm game, we will understand the importance of modularity, reusability, and code maintenance. Additionally, the challenge of simulating swarm behaviours, such as the movement and interaction of enemy units, encourages creative problem-solving and a deeper understanding of algorithms like pathfinding or flocking. It also provides a unique opportunity to work with either Unity or Blazor, both of which are widely used in the software development and game development industries.

Ultimately, this project is not just a technical exercise but a chance to develop teamwork and problem-solving skills. The task fosters collaboration as we work together to design, implement, and test the game, while also gaining experience that can be applied to future projects in game development or web application design.

2.0 Project Description

Swarm Attack Multiplayer Game - Wan and Whiskers

Wan and Whiskers is an exciting multiplayer swarm assault game created in Unity in which players go on an unending adventure with Wan, a talented shot with lightning-fast reflexes, and his devoted cat, Whiskers, who shields him with furious claw strikes. They struggle for survival against swarms of adversaries and zombies across three difficulty levels: easy, medium, and hard. Enemies will only chase Wan as they are only interested in a human's brain. This game mechanic makes it so that it is possible to play with on player only. Players must not let Wan's health points reach zero.

***Dynamic Multiplayer Experience**:*

- Players can team up, with one controlling Wan and the other taking on the role of Whiskers. Both characters have unique abilities that require teamwork and coordination to succeed.
    - **Wan**: Equipped with precise shooting skills and the ability to dodge enemy attacks with his rapid reflexes.
    - **Whiskers**: A brave and protective cat who defends Wan by scratching and slashing enemies in close combat.

Single Player Possibility

- The game mechanic where enemies will only chase Wan enables the possibility to play this game alone (quite challenging)

***Three Difficulty Levels**:*

- Players can choose between Easy, Medium, and Hard modes, each offering an escalating challenge.
    - **Easy**: Perfect for beginners, with slower and weaker enemies.
    - **Medium**: Introduces faster enemy swarms.
    - **Hard**: A test of ultimate teamwork and skill, with relentless and highly aggressive enemies including an immune boss that will follow you around.

***Endless Adventure Mode**:*

- Players battle through swarms striving to survive and protect Wan as long as possible while facing progressively tougher challenges trying to achieve the highest possible time score they can.

Interactive Environments

- Include destructible objects like walls and rocks that players can use strategically during combat.
- An immune boss that acts as a moving object with high damage.

3.0 Class

1. HealthBarUI

Attributes:

- healthBarForeGroundImage : image

Methods:

+ UpdateHealthBar() : void

2. HealthController

Attributes:

- currentHealth : float

- maximumHealth : float

Properties:

+ RemainingHealthPercentage : float

+ IsInvincible : bool

Methods:

+ TakeDamage() : void

+ AddHealth() : void

+ getInvincible() : bool

+ setInvincible() : bool

3. EnemyAttack

Attributes:

- damageAmount : float

Methods:

+ OnCollisionStay2D() : void

4. EnemySpawner

Attributes:

- minimumSpawnTime : float

- maximumSpawnTime : float

- timeUntilSpawn : float

5. EnemyMovement

Attributes:

- speed : float

- rotationSpeed : float

- screenBorder : float

Methods:

+ Awake() : void

+ Update() : void

+ SetTimeUntilSpawn() : void

+ UpdateTargetDirection() : void

+ HandleEnemyOffScreen() : void

6. BossMovement (inherits from EnemyMovement)

Methods:

+ FixedUpdate() : void

+ RotateTowardsTarget() : void

+ SetVelocity() : void

7. PlayerAwareness

Attributes:

+ AwareOfPlayer : bool

+ DirectionToPlayer : Vector2

- playerAwarenessDistance : float

- player : Transform

Methods:

+ getAwareOfPlayer() : bool

+ setAwareOfPlayer() : bool

+ getDirectionToPlayer() : Vector2

+ setDirectionToPlayer() : Vector2

+ Awake() : void

+ Start() : void

+ Update() : void

8. Bullet

Attributes:

- camera : Camera

9. Shoot

Attributes:

- bulletPrefab : GameObject

- bulletSpeed : float

- gunOffset : Transform

- timeBetweenShots : float

- fireContinuously : bool

- lastFireTime : float

Methods:

+ Awake() : void

+ Update() : void

+ OnTriggerEnter2D() : void

+ DestroyWhenOffScreen() : void

10. Timer

Attributes:

+ elapsedTime : float

Methods:

+ Update() : void

11. WanMovement

Attributes:

- MyRigidBody : Rigidbody2D

- movementInput : Vector2

- rotationSpeed : float

- speed : float

- smoothedMovementInput : Vector2

- movementInputSmoothVelocity : Vector2

- camera : Camera

+ mousePos : Vector2

Methods:

+ Awake() : void

+ Update() : void

+ FixedUpdate() : void

+ SetPlayerVelocity() : void

+ PreventPlayerGoingOffScreen() : void

+ RotateInDirectionOfInput() : void

+ OnMove() : void

12. WhiskerAttack

Attributes:

+ moveSpeed : float

+ attackRadius : float

+ animator : Animator

- movement : Vector2

Methods:

+ Update() : void

+ Attack() : void

+ OnDrawGizmosSelected() : void

13. PauseMenu

Attributes:

+PausePanel : GameObject
-isPaused : bool

.Methods:

+Start() : void
+Update() : void
+PauseGame() : void
+ResumeGame() : void

14.Game Manager

Attributes :

-timeToWaitBeforeExist : float
-GameOverMenu : GameObject

Methods :

+OnPlayerDied() : void
+EndGame() : void
+GoToMainPage() : void

15. WhiskerMovement

Attributes :

-MyRigidBody : Regidbody2D
-movementInput : Vector2
-rotationSpeed : float
-speed : float
-smoothedMovementInput : Vector2
-movementInputSmoothVelocity : Vector2
-camera : Camera

Methods :

-Awake() : void
-FixedUpdate() : void
-SetPlayerVelocity() : void
-PreventPlayerGoingOffScreen() : void
-RotateInDirectionOffInput() : void
-OnMove() : void

16.Main Menu

Methods:

+PlayForest()

+PlayCastle()

+PlayRealm()

+Exit

4.0 Screenshots

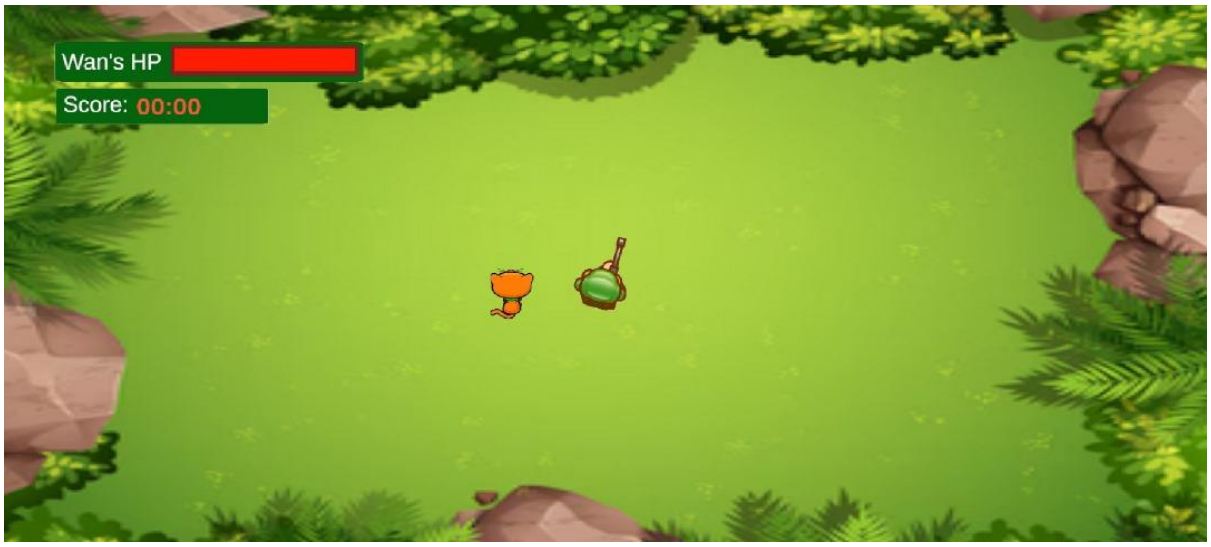
This is the main page. There are three worlds, forest, castle and realm. Users must choose one to start the game.

This is the instruction for users on how to play the game.


CHOOSE A MAP
FOREST
CASTLE
REALM
CONTROLS
Whisker's movement
• Up = W
• Left = A
• Back = S
• Right = D
Wan Shoots = Left Click
Wan's Aim = Mouse
Whisker's Attack = Space Bar
Wan's Movement = Arrow Keys


CHOOSE A MAP
FOREST
CASTLE
REALM
ABOUT
Wan & Whiskers is a two-player game where you play as Wan, a soldier with a high accuracy gun. And Whiskers, an orange cat that is able to do a scratch attack.
In this game, monsters will chase only Wan because the monsters are only interested in a human brain. Invite a friend to play as Whiskers to protect you and help get a higher score!
Different Maps have different difficulties:
Forest = Easy
Castle = Medium
Realm (Contains Boss) = Hard

This is the Forest map. Wan and Whiskers will be swarmed by enemies such as Zombies and Brown Bugs



This gameplay interface shows the player's HP bar, the current time score, and the swarm of zombies and bugs in a forest environment (easy level).

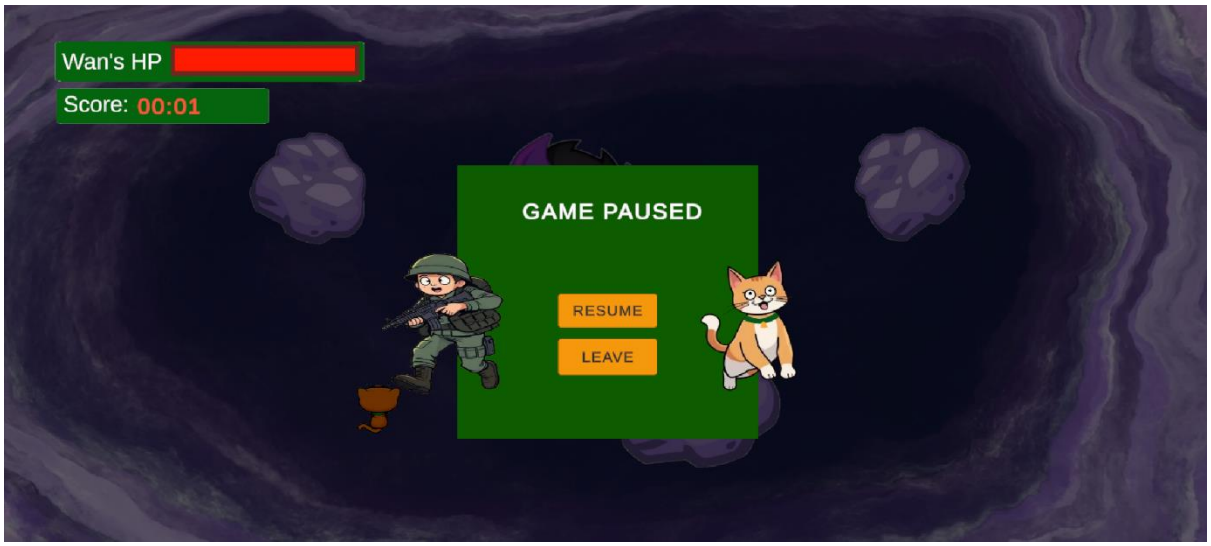
This is the Castle map. The enemies are faster this time than in the Forest map. Enemies include Mummies and Black Bugs

Users must defeat the faster enemies.

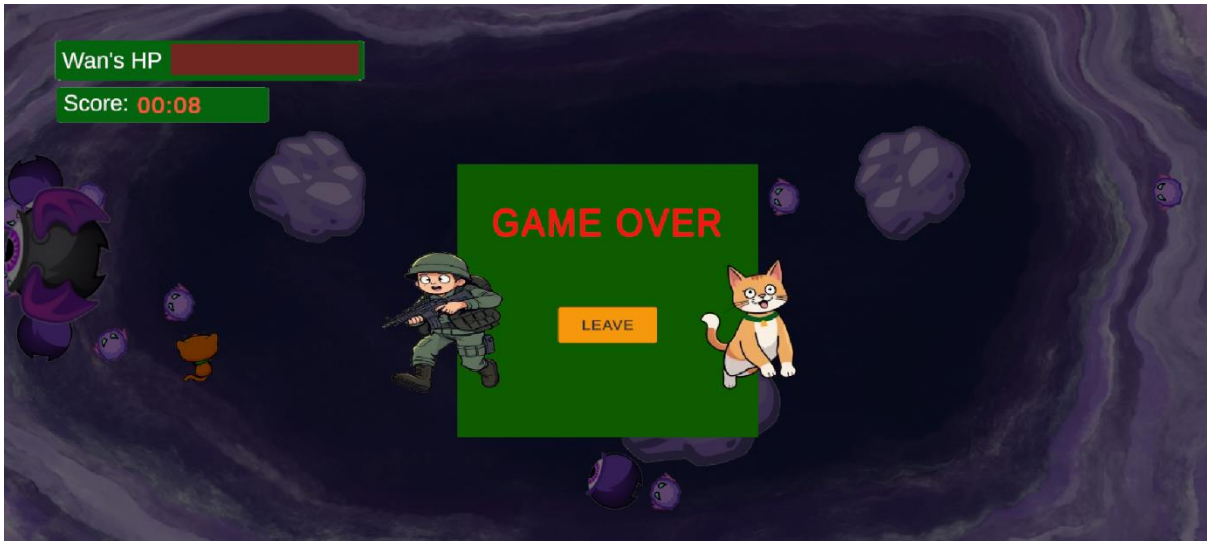
This is the Realms map. Wan and Whiskers must run away from a gigantic boss (Black Eyeball) monster that is immortal.

At the same time, the players must defeat a bunch of his minions including Purple Mists and Purple Eyeballs

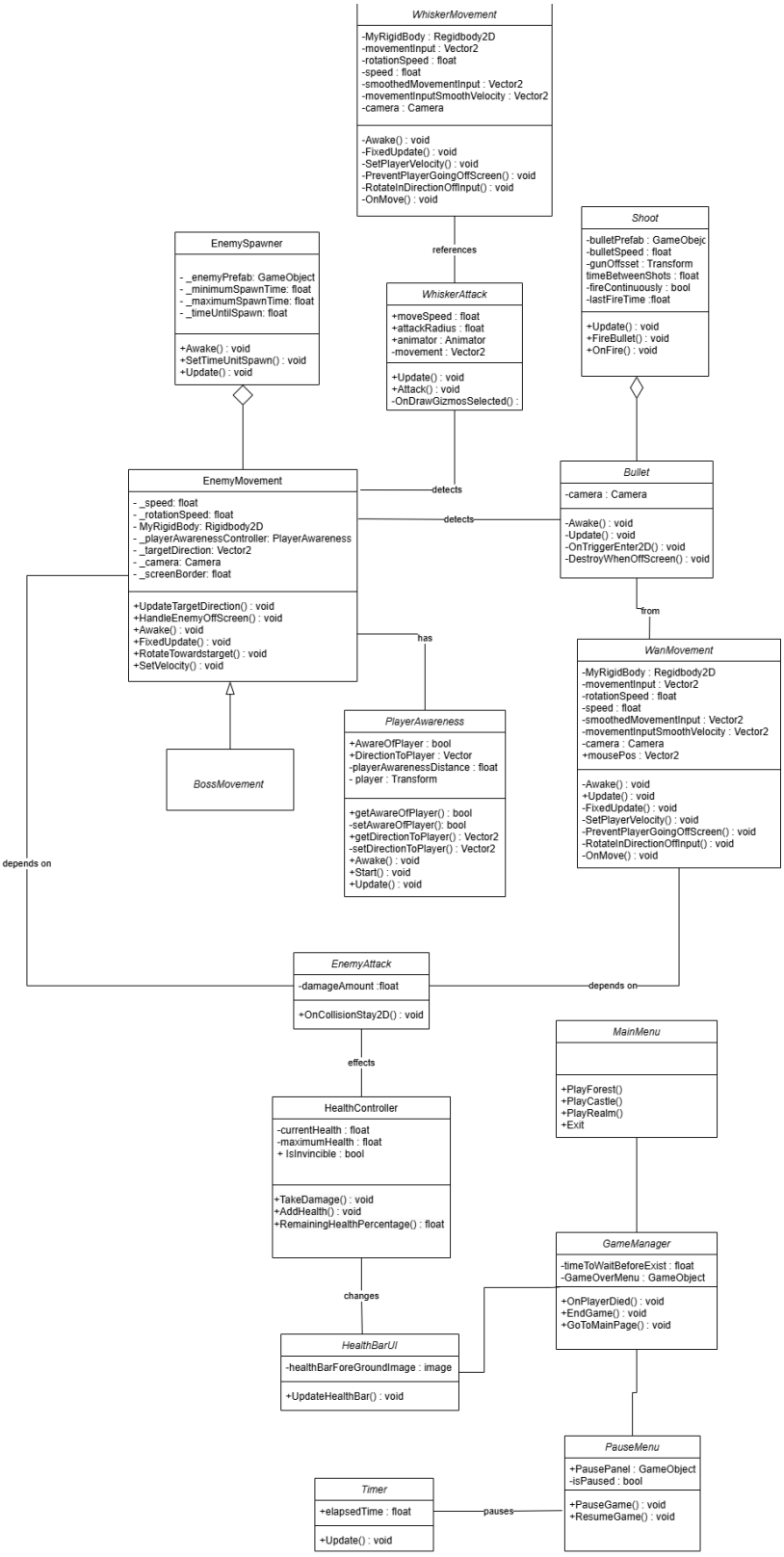
This is the interface while pausing the game.



This is the interface when the Wan is defeated.



5.0 UML


WhiskerMovement
-MyRigidBody : Regidbody2D
-movementInput : Vector2
-rotationSpeed : float
-speed : float
-smoothedMovementInput : Vector2
-movementInputSmoothVelocity : Vector2
-camera : Camera
-Awake() : void
-FixedUpdate() : void
-SetPlayerVelocity() : void
-PreventPlayerGoingOffScreen() : void
-RotateInDirectionOfInput() : void
-OnMove() : void
references
Shoot
-bulletPrefab : GameObejc
-bulletSpeed : float
-gunOffsset : Transform
timeBetweenShots : float
-fireContinuously : bool
-lastFireTime :float
+Update() : void
+FireBullet() : void
+OnFire() : void
EnemySpawner
- _enemyPrefab: GameObject
- _minimumSpawnTime: float
- _maximumSpawnTime: float
- _timeUntilSpawn: float
+Awake() : void
+SetTimeUnitSpawn() : void
+Update() : void
WhiskerAttack
+moveSpeed : float
+attackRadius : float
+animator : Animator
-movement : Vector2
+Update() : void
+Attack() : void
-OnDrawGizmosSelected() :
EnemyMovement
- _speed: float
- _rotationSpeed: float
- MyRigidBody: Rigidbody2D
- _playerAwarenessController: PlayerAwareness
- _targetDirection: Vector2
- _camera: Camera
- _screenBorder: float
+UpdateTargetDirection() : void
+HandleEnemyOffScreen() : void
+Awake() : void
+FixedUpdate() : void
+RotateTowardstarget() : void
+SetVelocity() : void
detects
detects
Bullet
-camera : Camera
-Awake() : void
-Update() : void
-OnTriggerEnter2D() : void
-DestroyWhenOffScreen() : void
from
has
BossMovement
PlayerAwareness
+AwareOfPlayer : bool
+DirectionToPlayer : Vector
-playerAwarenessDistance : float
- player : Transform
+getAwareOfPlayer() : bool
-setAwareOfPlayer(): bool
+getDirectionToPlayer() : Vector2
-setDirectionToPlayer() : Vector2
+Awake() : void
+Start() : void
+Update() : void
WanMovement
-MyRigidBody : Regidbody2D
-movementInput : Vector2
-rotationSpeed : float
-speed : float
-smoothedMovementInput : Vector2
-movementInputSmoothVelocity : Vector2
-camera : Camera
+mousePos : Vector2
-Awake() : void
+Update() : void
-FixedUpdate() : void
-SetPlayerVelocity() : void
-PreventPlayerGoingOffScreen() : void
-RotateInDirectionOfInput() : void
-OnMove() : void
depends on
EnemyAttack
-damageAmount :float
+OnCollisionStay2D() : void
depends on
effects
MainMenu
+PlayForest()
+PlayCastle()
+PlayRealm()
+Exit
HealthController
-currentHealth : float
-maximumHealth : float
+ IsInvincible : bool
+TakeDamage() : void
+AddHealth() : void
+RemainingHealthPercentage() : float
changes
GameManager
-timeToWaitBeforeExist : float
-GameOverMenu : GameObject
+OnPlayerDied() : void
+EndGame() : void
+GoToMainPage() : void
HealthBarUI
-healthBarForeGroundImage : image
+UpdateHealthBar() : void
PauseMenu
+PausePanel : GameObject
-isPaused : bool
+PauseGame() : void
+ResumeGame() : void
Timer
+elapsedTime : float
+Update() : void
pauses

6.0 Evaluation of Platform

Unity is a game development platform that has transformed digital content creation since its inception in 2004. It provides developers with a toolset to create 2D and 3D interactive experiences across multiple platforms. This versatile engine has become a cornerstone in both indie and professional game development.

The core strength of Unity lies in its cross-platform development capabilities. It supports over 25 different platforms including, gaming consoles, desktop systems, and mobile devices such as iOS and Android. This allows developers to deploy their games across multiple platforms without rewriting code which automatically reducing the development time and resources.

Unity offers an interface that makes it approachable for beginners while still providing advanced features for professional developers. The drag-and-drop functionality and visual editing tools enable rapid prototyping and development. Furthermore, developers can use C# for scripts which is considered user-friendly and modern compared to other game development languages. It also supports both 2D and 3D game development with a wide range of toolsets, including physics engines and animation systems. The Unity Asset Store enhances development efficiency by providing access to pre-made assets, scripts, and tools. This marketplace helps developers save time and resources, particularly beneficial for smaller teams or freelancers.

However, Unity faces several weaknesses too. One of it is performance overhead, particularly for complex games. Unity can be less performing compared to custom-built engines which require careful optimization for demanding projects. While Unity offers a free version, advanced features still require paid subscriptions. The price can be expensive for complex projects.

Graphics quality remains a debated aspect. Some developers argue that Unity’s default graphics are not as polished as competitors like Unreal Engine. While recent updates have improved rendering capabilities, achieving high-end graphics quality requires additional effort and expertise.

Despite its limitations, Unity remains a powerful and versatile game development platform. Its strengths in cross-platform development, accessibility, and comprehensive toolset make it an excellent choice for many developers. While it may require optimization for complex projects and involve certain costs, the platform’s overall capabilities and improvements maintain its position as a leading game development solution.