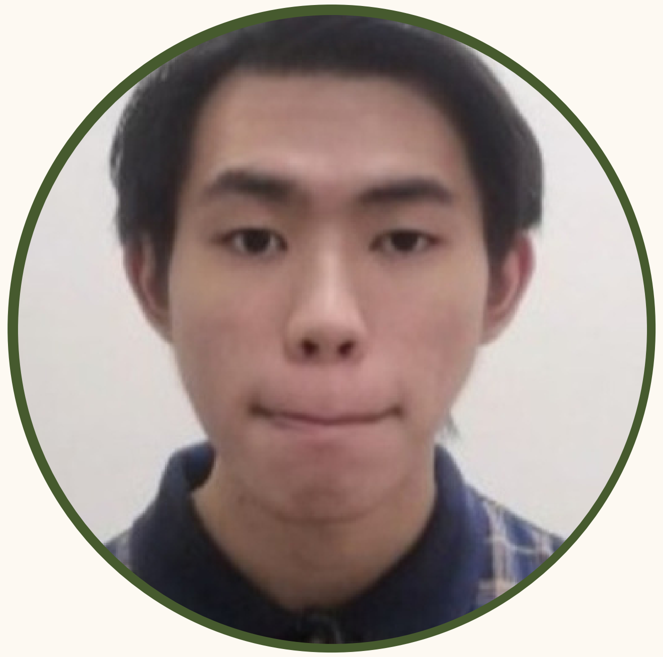
GROUP PROJECT

Presented By Group 5

HELLO SECTIONMATES WE CAME FROM GROUP 5!

MEET THE GROUP MEMBER



YEO CHUN TECK

Group Leader

NG KAI ZHENG

Researcher

JACK KEO LIN JIE

Resource Collector

SHI JUN HUI

Introducer

GAME NAME

INVASION

GAME MISSION

The mission is to control the spaceship to avoid the enemy's attack and defeat the enemy with bullets to get points. If your HP becomes 0, the game is over.

HOW TO PLAY

Use the keyboard WASD to control the movement of your spaceship

Click the left mouse button to shoot bullets to attack the enemy spaceship

GAMEPLAY

10/10

You can score 1, 2, and 3 points by shooting down enemies with 1, 2, and 3hp respectively.

3/3 2/2 1/1

3 point

2 point

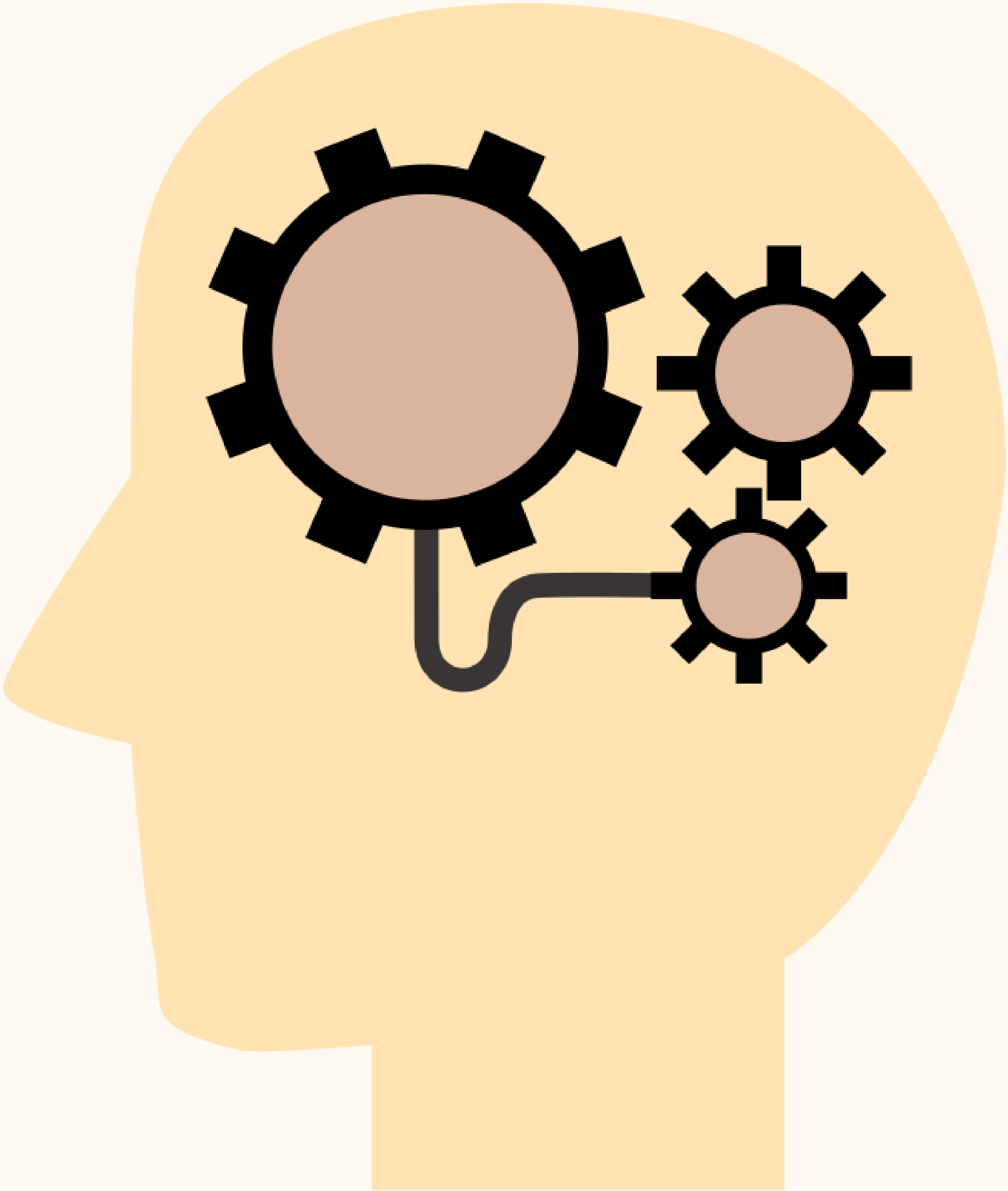
1 point

While every time you bump into an enemy spaceship, 1 HP will be deducted until the game ends when your HP went to 0.

Enemies HP are randomly spawning in between 1 to 3HP.

Difficulty increases as the score increases. Enemies spaceship's respawn speed and moving speed will increase.

PROJECT SPECIFICATION



ARRAY OF OBJECT

ENCAPSULATION

ASSOCIATION

INHERITANCE AND POLYMORPHISM

ARRAY OF OBJECT

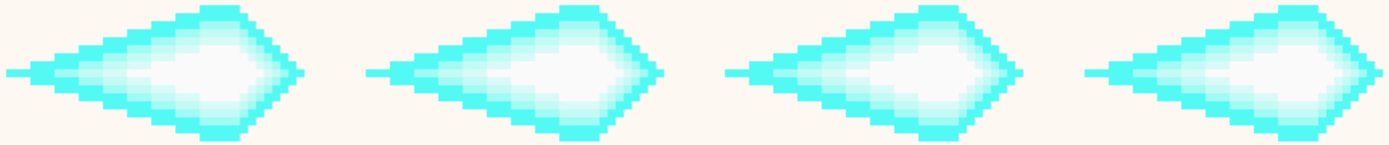
The array of objects represents storing multiple objects in a single name. In an array of objects, the data can be accessed randomly by using the index number. It could be used to reduce the time and memory by storing the data in a single variable.

ENEMY

Multiple Enemies will be generated randomly in the scene of the game. Hence, an array is built for storing the data of these enemies which are declared from the same class.

BULLET

The spaceship which is controlled by the player is designed to be able to fire unlimited bullets. Hence, an array is built for storing more than one Bullet data.



Encapsulation

Encapsulation describes the idea of bundling variables(private) and methods(public) that work on that data within one unit which is a class. This concept is often used to hide the internal representation or state of an object from the outside. The classes built in this project are categorized into three:

Entity

The classes declared as an entity are: Player, Enemy, Bullet and Window. This kind of class includes the specific variable or method of these entities for describing the internal state and characteristic of the object

- ✓ Describe the characteristic and the components of the entity
- ✓ Describe the actions which will be performed in different entity

Classes From External Graphic Library

The external graphic library is included for ease of development and to perform complex mechanisms. The library which is expected to be used in this project will be SFML. This library includes many classes which have their function in different aspects, especially in the field of graphics

WHAT CLASSES FROM EXTERNAL LIBRARY WILL BE USED?

- ✓ Sprint
- ✓ Texture
- ✓ Font
- ✓ Text
- ✓ RenderWindow

Feature & Component

The Feature & Component class is declared for performing a different kinds of actions or components on different entities. In order to perform a specific action, different kinds of classes from the external graphic library are required to be used together and these classes are stored in one class which is called Feature & Component class

WHAT CLASSES ARE DESIGNED AS FEATURE CLASS

- ✓ Generate
- ✓ Movement
- ✓ Score
- ✓ gameOverText

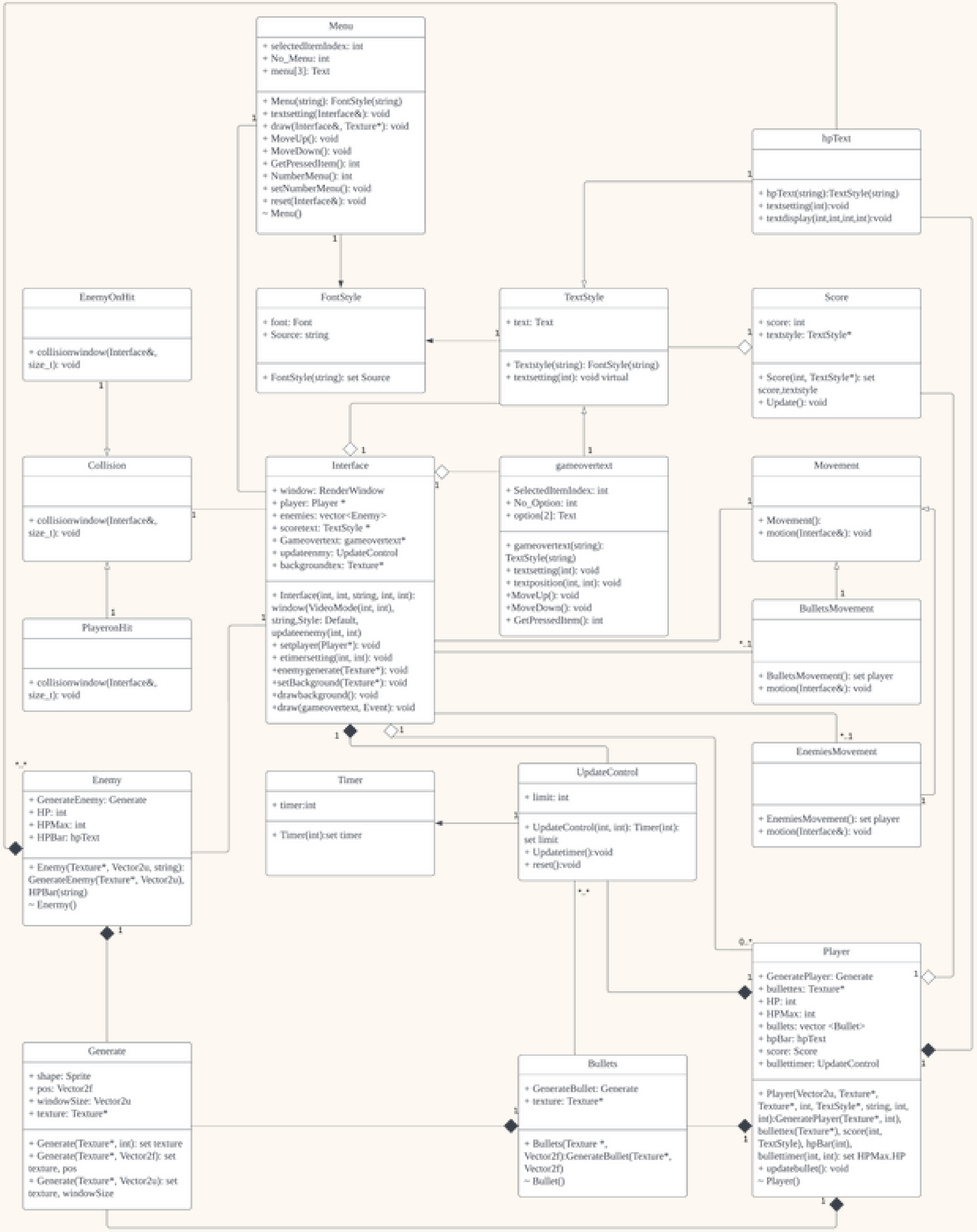
More Classes will be showed in the next segment

RELATIONSHIP BETWEEN CLASSES

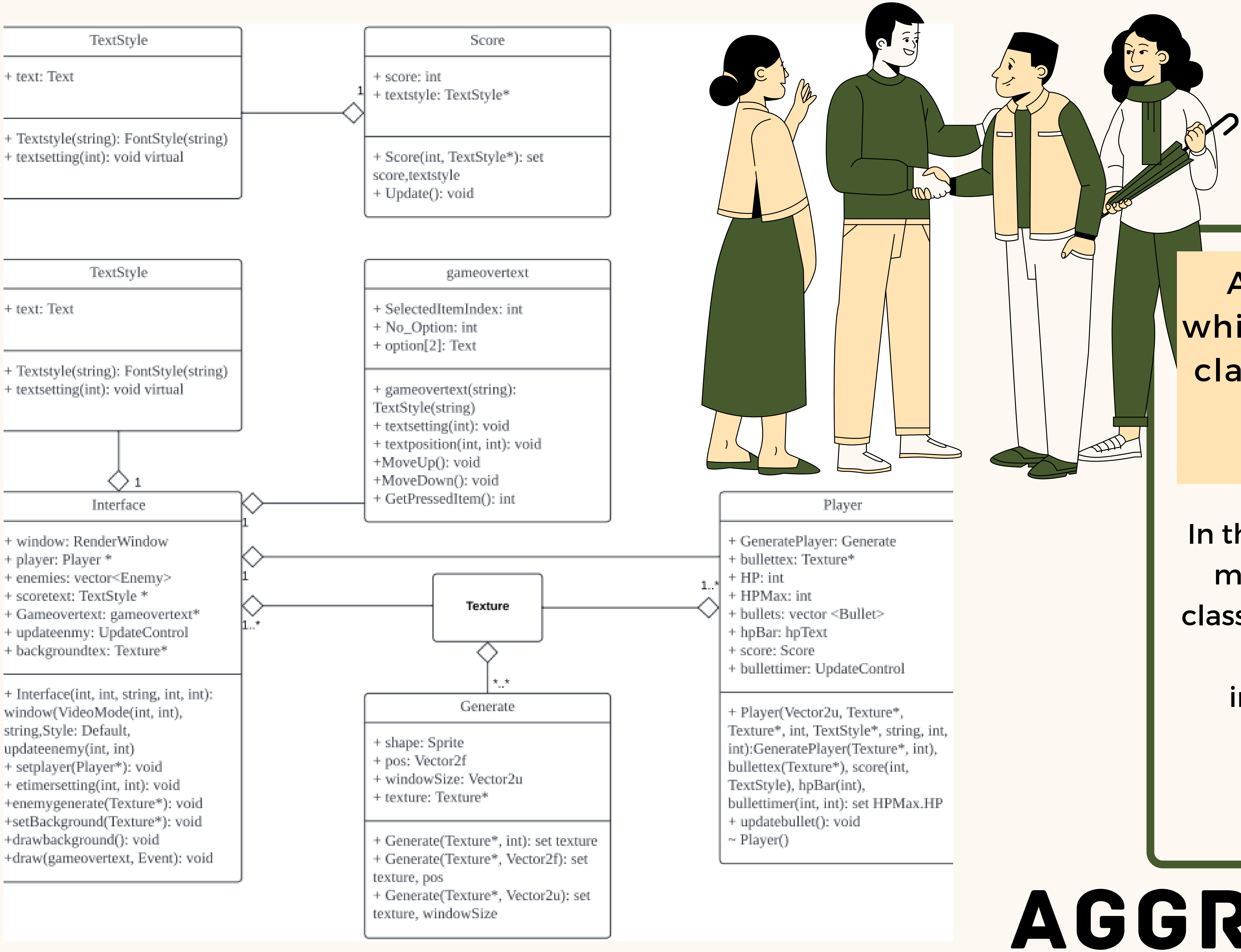
The right one is the UML diagram for this project. This diagram excludes the display of classes from the external library. In this session, we will focus more on these relationship:

☑ Aggregation and Composition

☑ Inheritance and Polymorphism


Menu
+ selectedItemIndex: int
+ No_Menu: int
+ menu[3]: Text
+ Menu(string): FontStyle(string)
+ textsetting(Interface&): void
+ draw(Interface&, Texture*): void
+ MoveUp(): void
+ MoveDown(): void
+ GetPressedItem(): int
+ NumberMenu(): int
+ setNumberMenu(): void
+ reset(Interface&): void
~ Menu()
hpText
+ hpText(string):TextStyle(string)
+ textsetting(int):void
+ textdisplay(int,int,int,int):void
EnemyOnHit
+ collisionwindow(Interface&, size_t): void
FontStyle
+ font: Font
+ Source: string
+ FontStyle(string): set Source
TextStyle
+ text: Text
+ Textstyle(string): FontStyle(string)
+ textsetting(int): void virtual
Score
+ score: int
+ textstyle: TextStyle*
+ Score(int, TextStyle*): set score,textstyle
+ Update(): void
Collision
+ collisionwindow(Interface&, size_t): void
Interface
+ window: RenderWindow
+ player: Player *
+ enemies: vector<Enemy>
+ scoretext: TextStyle *
+ Gameovertext: gameovertext*
+ updateenmy: UpdateControl
+ backgroundtex: Texture*
+ Interface(int, int, string, int, int): window(VideoMode(int, int), string,Style: Default, updateenemy(int, int)
+ setplayer(Player*): void
+ etimersetting(int, int): void
+enemygenerate(Texture*): void
+setBackground(Texture*): void
+drawbackground(): void
+draw(gameovertext, Event): void
gameovertext
+ SelectedItemIndex: int
+ No_Option: int
+ option[2]: Text
+ gameovertext(string): TextStyle(string)
+ textsetting(int): void
+ textposition(int, int): void
+MoveUp(): void
+MoveDown(): void
+ GetPressedItem(): int
Movement
+ Movement():
+ motion(Interface&): void
PlayeronHit
+ collisionwindow(Interface&, size_t): void
BulletsMovement
+ BulletsMovement(): set player
+ motion(Interface&): void
EnemiesMovement
+ EnemiesMovement(): set player
+ motion(Interface&): void
Enemy
+ GenerateEnemy: Generate
+ HP: int
+ HPMax: int
+ HPBar: hpText
+ Enemy(Texture*, Vector2u, string): GenerateEnemy(Texture*, Vector2u), HPBar(string)
~ Enermy()
Timer
+ timer:int
+ Timer(int):set timer
UpdateControl
+ limit: int
+ UpdateControl(int, int): Timer(int): set limit
+ Updatetimer():void
+ reset():void
Player
+ GeneratePlayer: Generate
+ bullettex: Texture*
+ HP: int
+ HPMax: int
+ bullets: vector <Bullet>
+ hpBar: hpText
+ score: Score
+ bullettimer: UpdateControl
+ Player(Vector2u, Texture*, Texture*, int, TextStyle*, string, int, int):GeneratePlayer(Texture*, int), bullettex(Texture*), score(int, TextStyle), hpBar(int), bullettimer(int, int): set HPMax.HP
+ updatebullet(): void
~ Player()
Generate
+ shape: Sprite
+ pos: Vector2f
+ windowSize: Vector2u
+ texture: Texture*
+ Generate(Texture*, int): set texture
+ Generate(Texture*, Vector2f): set texture, pos
+ Generate(Texture*, Vector2u): set texture, windowSize
Bullets
+ GenerateBullet: Generate
+ texture: Texture*
+ Bullets(Texture *, Vector2f):GenerateBullet(Texture*, Vector2f)
~ Bullet()
1
..
*..1
0..*

AGGREGATION



TextStyle
- + text: Text
- + Textstyle(string): FontStyle(string)
- + textsetting(int): void virtual

Score
- + score: int
- + textstyle: TextStyle*
- + Score(int, TextStyle*): set score,textstyle
- + Update(): void

TextStyle
- + text: Text
- + Textstyle(string): FontStyle(string)
- + textsetting(int): void virtual

gameovertext
- + SelectedItemIndex: int
- + No_Option: int
- + option[2]: Text
- + gameovertext(string): TextStyle(string)
- + textsetting(int): void
- + textposition(int, int): void
- +MoveUp(): void
- +MoveDown(): void
- + GetPressedItem(): int

Interface
- + window: RenderWindow
- + player: Player *
- + enemies: vector<Enemy>
- + scoretext: TextStyle *
- + Gameovertext: gameovertext*
- + updateenmy: UpdateControl
- + backgroundtex: Texture*
- + Interface(int, int, string, int, int): window(VideoMode(int, int), string,Style: Default, updateenemy(int, int)
- + setplayer(Player*): void
- + etimersetting(int, int): void
- +enemygenerate(Texture*): void
- +setBackground(Texture*): void
- +drawbackground(): void
- +draw(gameovertext, Event): void

Texture

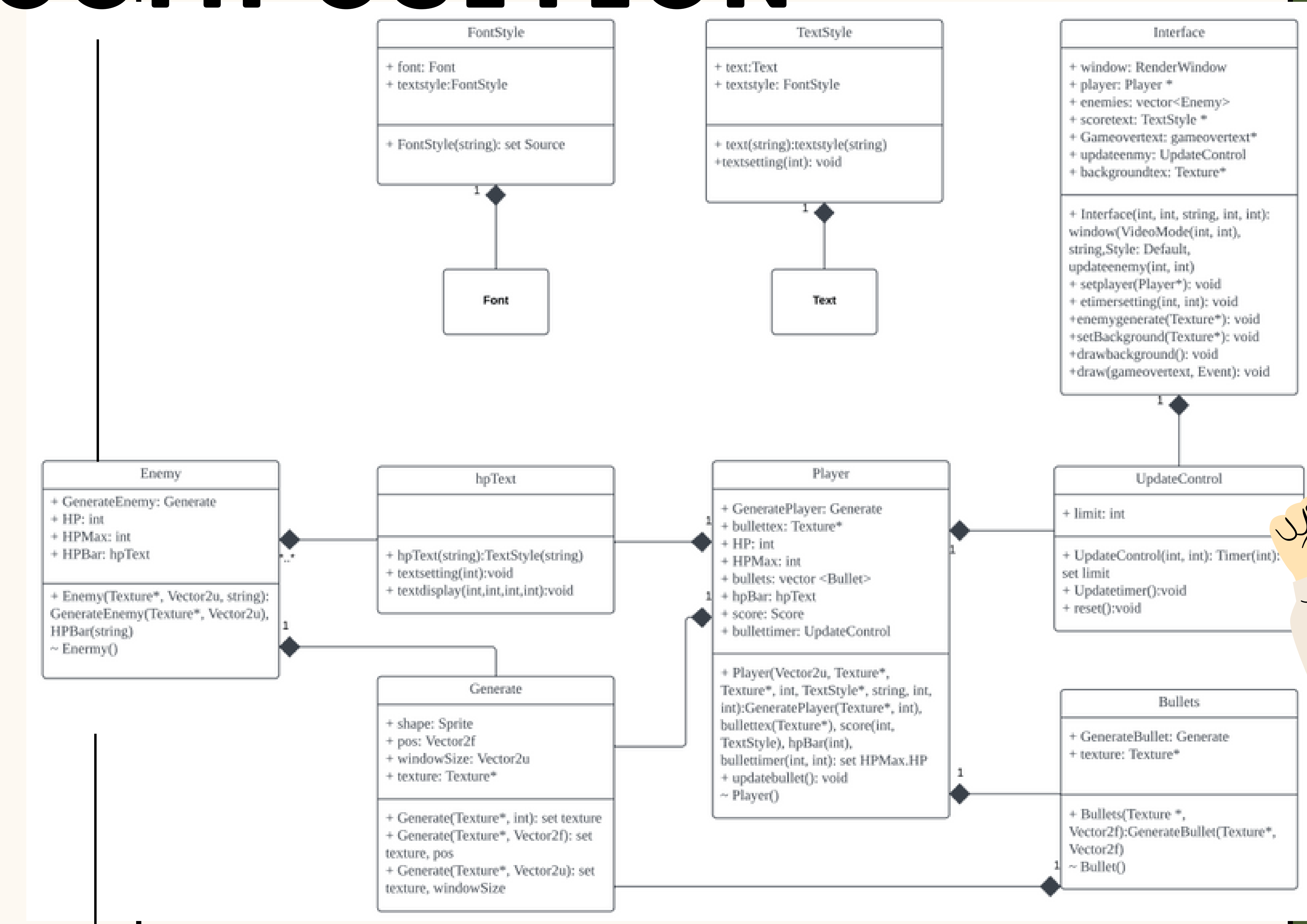
Generate
- + shape: Sprite
- + pos: Vector2f
- + windowSize: Vector2u
- + texture: Texture*
- + Generate(Texture*, int): set texture
- + Generate(Texture*, Vector2f): set texture, pos
- + Generate(Texture*, Vector2u): set texture, windowSize

Player
- + GeneratePlayer: Generate
- + bullettex: Texture*
- + HP: int
- + HPMax: int
- + bullets: vector <Bullet>
- + hpBar: hpText
- + score: Score
- + bullettimer: UpdateControl
- + Player(Vector2u, Texture*, Texture*, int, TextStyle*, string, int, int):GeneratePlayer(Texture*, int), bullettex(Texture*), score(int, TextStyle), hpBar(int), bullettimer(int, int): set HPMax.HP
- + updatebullet(): void
- ~ Player()

Aggregation is a process in which one class defines another class as any reference or usage for performing specific functions

In the project, aggregation will be used mostly for making the object of the classes from the external graphic library as a reference so that we could implement the methods in these classes.

COMPOSITION



FontStyle
+ font: Font
+ textstyle:FontStyle
+ FontStyle(string): set Source

Font

TextStyle
+ text:Text
+ textstyle: FontStyle
+ text(string):textstyle(string)
+textsetting(int): void

Text

Interface
+ window: RenderWindow
+ player: Player *
+ enemies: vector<Enemy>
+ scoretext: TextStyle *
+ Gameovertext: gameovertext*
+ updateenmy: UpdateControl
+ backgroundtex: Texture*
+ Interface(int, int, string, int, int): window(VideoMode(int, int), string,Style: Default, updateenemy(int, int)
+ setplayer(Player*): void
+ etimersetting(int, int): void
+enemygenerate(Texture*): void
+setBackground(Texture*): void
+drawbackground(): void
+draw(gameovertext, Event): void

Enemy
+ GenerateEnemy: Generate
+ HP: int
+ HPMax: int
+ HPBar: hpText
+ Enemy(Texture*, Vector2u, string): GenerateEnemy(Texture*, Vector2u), HPBar(string)
~ Enermy()

hpText
+ hpText(string):TextStyle(string)
+ textsetting(int):void
+ textdisplay(int,int,int,int):void

Player
+ GeneratePlayer: Generate
+ bullettex: Texture*
+ HP: int
+ HPMax: int
+ bullets: vector <Bullet>
+ hpBar: hpText
+ score: Score
+ bullettimer: UpdateControl
+ Player(Vector2u, Texture*, Texture*, int, TextStyle*, string, int, int):GeneratePlayer(Texture*, int), bullettex(Texture*), score(int, TextStyle), hpBar(int), bullettimer(int, int): set HPMax.HP
+ updatebullet(): void
~ Player()

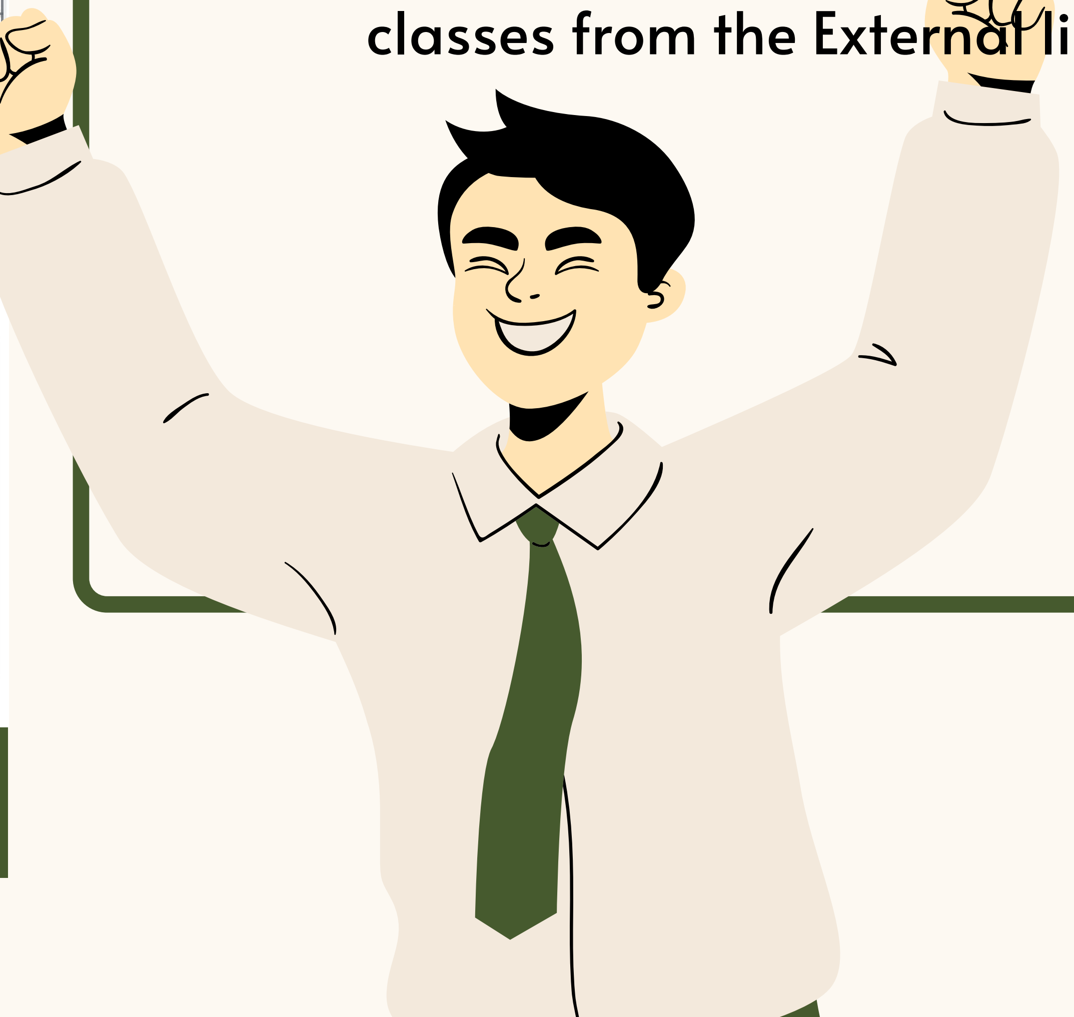
UpdateControl
+ limit: int
+ UpdateControl(int, int): Timer(int): set limit
+ Updatetimer():void
+ reset():void

Generate
+ shape: Sprite
+ pos: Vector2f
+ windowSize: Vector2u
+ texture: Texture*
+ Generate(Texture*, int): set texture
+ Generate(Texture*, Vector2f): set texture, pos
+ Generate(Texture*, Vector2u): set texture, windowSize

Bullets
+ GenerateBullet: Generate
+ texture: Texture*
+ Bullets(Texture *, Vector2f):GenerateBullet(Texture*, Vector2f)
~ Bullet()

Composition is a process of builiding complex objects from simpler ones.

In the project, the composition will be established when the object of some classes is built inside another class as the data member for this class so that these objects could be used as components. The lifespan of these objects is tied to that of the class. This kind of relationship could be seen between Entity classes, and Feature & Component classes. Between Feature & Component classes and classes from the External library.



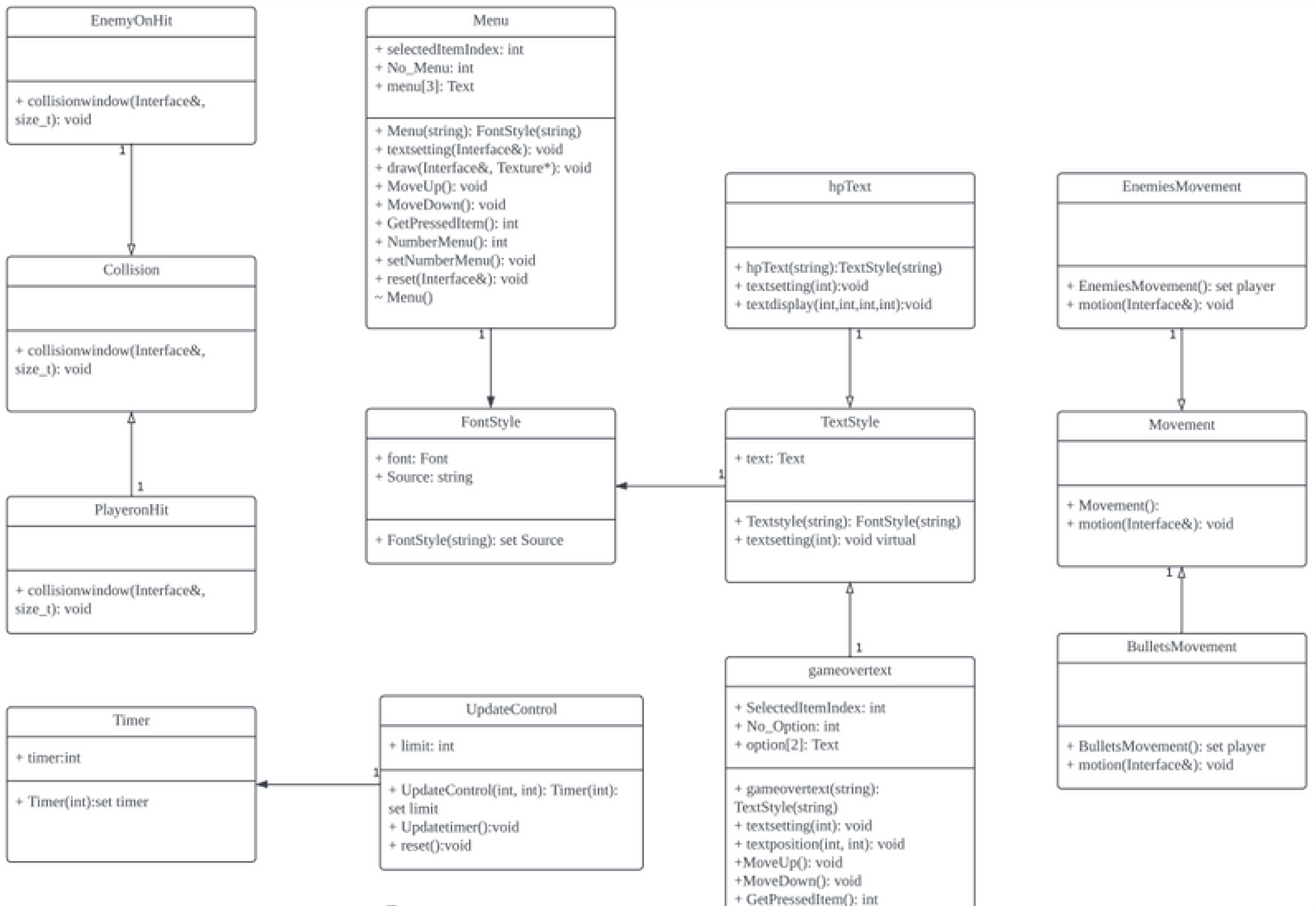
INHERITANCE AND POLYMORPHISM

Inheritance...

Inheritance allow us to create a new class (derived class) from an existing class (base class). The derived class inherits the features from the base class and can have additional features of its own. In this project, the base class is considered as a class for general setting for derived class.

Polymorphism...

Polymorphism means that a call to a member function will cause a different function to be executed depending on the type of object that invokes the function. In this project, this relationship is established for the same actions with some modification according to the class where they are tied to.



EnemyOnHit

+ collisionwindow(Interface&, size_t): void

Collision

+ collisionwindow(Interface&, size_t): void

PlayeronHit

+ collisionwindow(Interface&, size_t): void

Menu

+ selectedItemIndex: int
+ No_Menu: int
+ menu[3]: Text

+ Menu(string): FontStyle(string)
+ textsetting(Interface&): void
+ draw(Interface&, Texture*): void
+ MoveUp(): void
+ MoveDown(): void
+ GetPressedItem(): int
+ NumberMenu(): int
+ setNumberMenu(): void
+ reset(Interface&): void
~ Menu()

FontStyle

+ font: Font
+ Source: string

+ FontStyle(string): set Source

hpText

+ hpText(string):TextStyle(string)
+ textsetting(int):void
+ textdisplay(int,int,int,int):void

TextStyle

+ text: Text

+ Textstyle(string): FontStyle(string)
+ textsetting(int): void virtual

gameovertext

+ SelectedItemIndex: int
+ No_Option: int
+ option[2]: Text

+ gameovertext(string): TextStyle(string)
+ textsetting(int): void
+ textposition(int, int): void
+MoveUp(): void
+MoveDown(): void
+ GetPressedItem(): int

EnemiesMovement

+ EnemiesMovement(): set player
+ motion(Interface&): void

Movement

+ Movement():
+ motion(Interface&): void

BulletsMovement

+ BulletsMovement(): set player
+ motion(Interface&): void

Timer

+ timer:int

+ Timer(int):set timer

UpdateControl

+ limit: int

+ UpdateControl(int, int): Timer(int): set limit
+ Updatetimer():void
+ reset():void

◄ ❚❚ ►

GAME DEMO

THANK YOU

Do you have any questions of us?