

Programming II

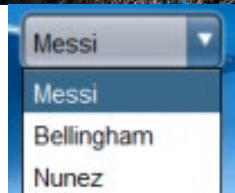
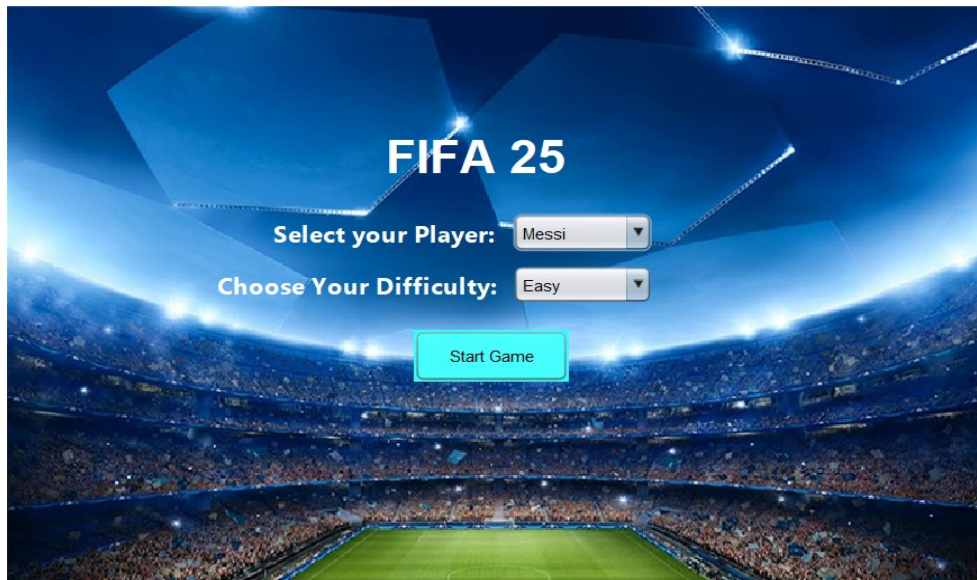
Final project report

Students(Group 4 – Section 1):

- **Karim Ashraf Mahmoud ElGamal (8233)**
- **Mahmoud Mohamed Badawy (8197)**
- **Abdullah Marwan ElSalmi (8150)**

User guide to the circus of plates(football edition):

-Main Menu screen:



the option to select a player from 3 players(messi,bellingham,nunez).

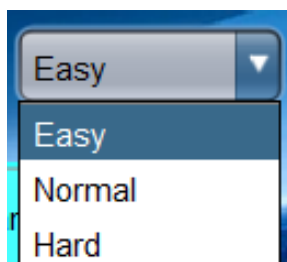
-The user have

-The user also have 3 difficulties to choose from(easy,normal,hard).

The Easy difficulty the objects fall in a straight line horizontally with a slow speed.

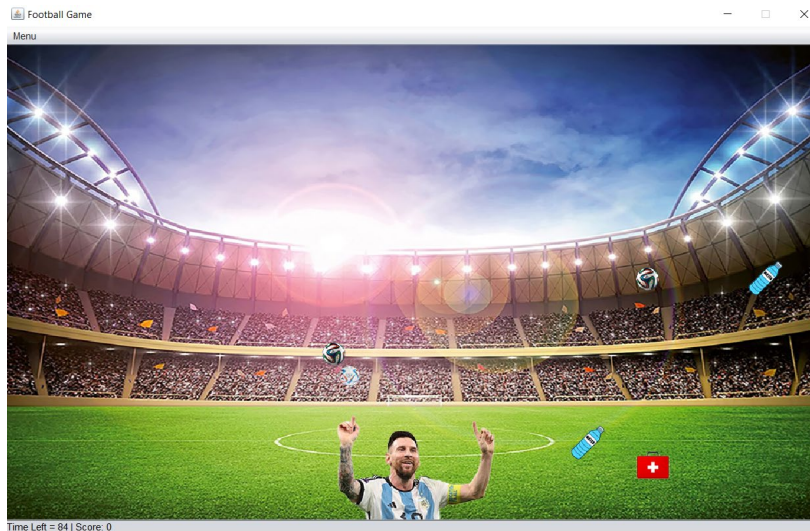
The Normal difficulty the objects fall horizontally while shaking vertically with a higher speed than the easy difficulty with the avoidables spawn rate increasing.

The Hard difficulty the objects fall horizontally with a larger vertical shake than the normal and the avoidable spawn rate increase.



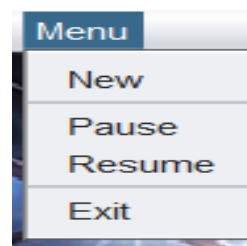
- The Main menu screen also has a theme, the champions league theme that loops.
- When the user is done with choosing his player and difficulty they can select start game to start playing.

-Game Screen:



The game works by having the player move left or right using the arrow keys to collect either collectable items or gets hit by an avoidable. The score corner in the left bottom corner, if 3 balls are stacked the score increases by 1, if the player stacks 3 ballon d'ors the score increases by 3, else if the player gets hit by an avoidable item the score -10 of the same collectable type on one hand, and a timer that starts at 90 seconds and counts down to 0, if the timer reaches 0 it will end the game with a game over text overlayed on the screen. The player has 3 lives. The screen has a background sound of crowd support


In the top left corner is a drop down menu called “Menu” that has the options of New, pause, resume and exit.





The game has 4 collectable objects that are generated to fall from the top of the screen and the player has to catch on either hand.

The collectables are:

➤ Ballon'dor: 

➤ Rihala ball: 


➤ Brazuca ball: 

➤ Jibulani ball: 

The game also has 2 avoidables, if one hit the player the score is reset to 0 and the player lose 1 live.

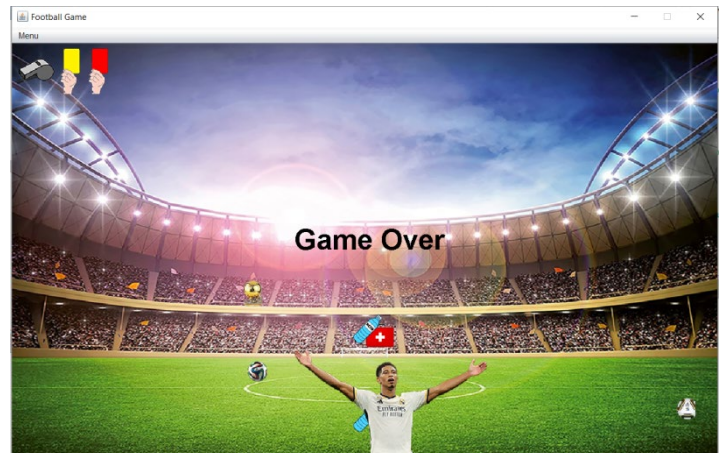
The Avoidables:

➤ Injury: 

➤ Water Bottle: 

The player lives at the start of the game are 3 lives and for 2 and 1 and 0 lives remaining its shown by having icons of a whistle a yellow card and a red card display on the top left of the screen. If the lives are 0 it will end the game

The Game also includes sound effects for scoring a point, hitting an avoidable and game over.



The score point sound effects varies according to the player used as each one has a unique sound effect.

The hitting an avoidable object sound effect is a whistle and the game over is a full time whistle.

Design patterns used:

- **Singleton**
- **Observer**
- **State**
- **Strategy**
- **Factory**
- **Flyweight**

Singleton:

Singleton is implemented in the project to ensure that a class has only one instance, while providing a global access point to this instance

Observer:

Observer is a behavioral design pattern that lets you define a subscription

mechanism to notify multiple objects about any events that happen to the object they're observing. It is used in the project to notify users when an avoidable hit the player or when live counter decrease or the timer of the game or when scoring a point when stacking 3 collectables.

State:

State is a behavioral design pattern that lets an object alter its behavior when its internal state changes. It appears as if the object changed its class. It is used in the project to show the state of the game, is it running or is it finished.

Factory:

Factory Method is a creational design pattern that provides an interface for creating objects in a superclass but allows subclasses to alter the type of objects that will be created. It is implemented to create the generated items to be dropped in the game, it will generate two types of objects: avoidable and collectables.

Strategy:

Strategy is a behavioral design pattern that lets you define a family of algorithms, put each of them into a separate class, and make their objects interchangeable.

Strategy design pattern is used in the project to load suitable game and have the player choose the difficulty of the game easy, medium and hard.

Flyweight:

Flyweight is used across the project in order to recycle already created objects and reuse them therefore decreasing memory consumption and increasing performance of the program

