**Game Design Document**

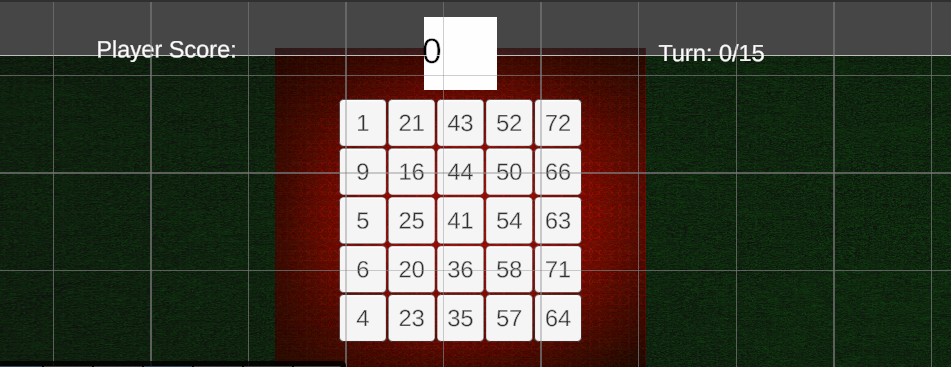
By: Carlos, Andres and Johan

Executive Summary:

Our game is called Bingo Star, it is a bingo game But unlike a normal bingo in this one you have limited turns to be able to do bingo and also you must get a number of points to be able to pass to the next level that will increase as you go from round to round and in turn you will obtain different powers and improvements to get an advantage to achieve your goal in each round and you win when you reach a number of round to be determined and if at the end of each round you do not get the goal, you lose and you must start from 0. The world is simply a bingo card on a table and the character is you trying to achieve the objective in each round.

User Interfaces





The menu is quite simple with two buttons to start the game or close the game with a typical casino and gambling music.

For now the main interface of our game is the board with the bingo card, a scoreboard with your score and the number of turns you have to reach your target number of points to pass the round.

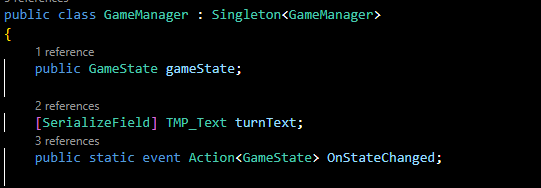
Controls

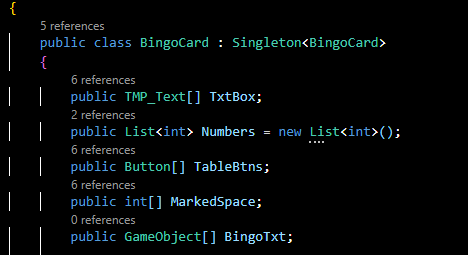
The controls are also quite simple as you only have to use the space bar for the random numbers and if that number you selected is on your bingo card it will be selected automatically and if that box has a power it will also be activated automatically.

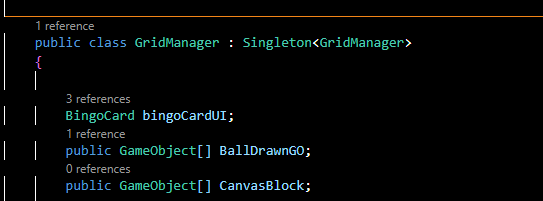
Design patterns:

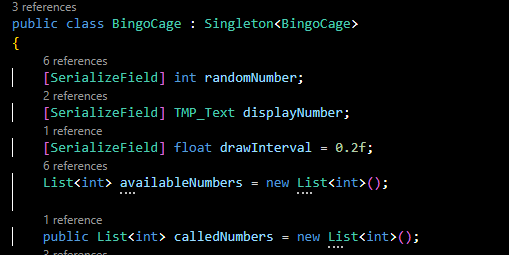
Singleton pattern: It is the pattern we use the most in our game and in several of our codes as it provides for multiple instances of important objects to be created more than once.

Example of Singleton pattern in the game:









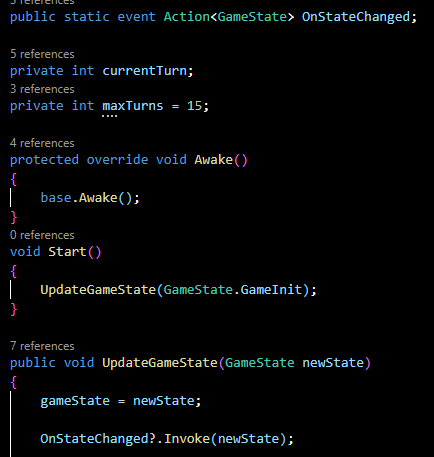
State pattern: It is implemented on the “GameManager” because it handle different game state using enumeration and switch statement inside the “UpdateGameState”

Example of State pattern:

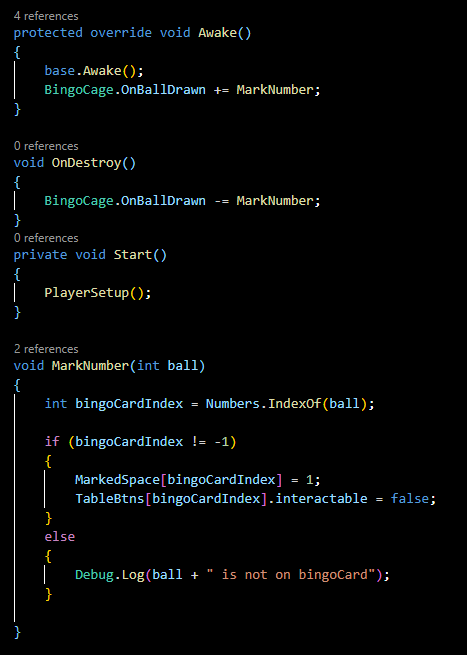


Observer pattern: Another pattern that we use in several of our codes because in order for some codes to work they have to subscribe to other elements of the code in order to work.

Example of observer pattern in the code:

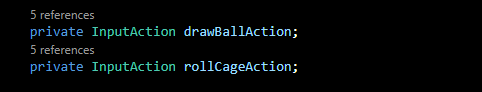


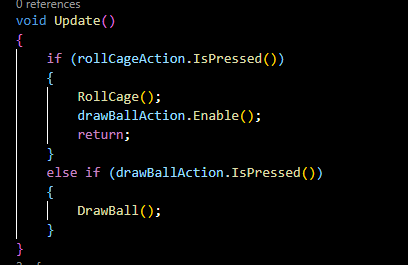




Command pattern: It is use on the “bingoCage” code because it handle player input actions like: “DrawBall” and “RollCage”

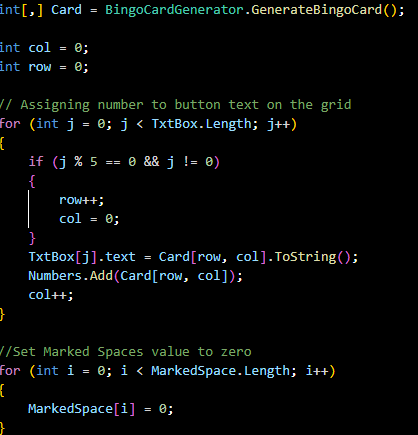
Example of Command pattern in the code:





Factory Pattern: This pattern is use in the “BingoCard” to generate different bingo card a create a different card every time you play.

Example of factory pattern in the code:



Tasks and schedule breakdown:

The most important things to implement in the game are the powers, the advantages that the player will have in each game he plays, the score that the player will have to obtain to pass the round and to implement the player's score every time he gets a number in his bingo.

First we want to do the scores since it is the most important part of the game and then we want to concentrate on the powers and the advantages for the player.

UML Diagram: 