Jaume I University, Castellon

Degree on Design and Development
of Video games

Technical Report of the Final Degree Project

Development of a female team manager simulator for Android

Author: José Daroqui Plaza
Tutor: José Miguel Sanchiz Martí

Summary

The purpose of this project is to build a basketball simulator which includes games simulations and season simulations. All the system is integrated in an application for Android. The user will take over a basketball team, and will have the chance to train players, prepare game strategies, and visualize statistics and classifications during all the season.
In the game, the players will be women instead of men. The goal of this change is to introduce women basketball in this type of games in order to seek equality.

Key words

Application, management, statistics, basketball

Index

Summary 1
Key words 1

Chapter 1: Technical Proposal 7
1. Introduction 7
2. Related subjects 7
3. Objectives 8
4. Planification 8
5. Expected results 9
6. Tools 9
7. References 9

Chapter 2: Related work 10
1. References 10

Chapter 3: Design 13
1. Statement 13
2. Controls & Requirements 14
3. Game Interface 14
4. Teams and players 20
5. Art Design 21

Chapter 4: Development 28
1. Technical Proposal 28
    1.1 Technical Proposal classes 28
    1.2 Final delivery of the Technical Proposal 28
2. Concept Design 29
    2.1 Information Research 29
    2.2 Tools to develop the game 29
3. Programming 30
    3.1 Players 30
    3.2 Teams 34
    3.3 Training 37
    3.4 Team preferences 38
    3.5 Stats screen 38
    3.6 Games 42
    3.7 Team choosing 53
    3.8 Save and load manager 53
    3.9 Manager 55
    3.10 Tasks 56

4. Artistic Design 57
4.1 Design of the Teams' logos 57
4.2 Design of the buttons 57
4.3 Design of the backgrounds 58
4.4 Distribution on the different screens 60
5. Documentation 61
5.1 Game Design Document 61
5.2 Final Report of the project 61
5.3 Presentation of the project 61

Chapter 5: Results & Testing 62
1. Results 62
2. Testing 64

Chapter 6: Deviations from the project 65
1. Changes from the original idea 65
1.1 Extra-sports aspects 65
1.2 Database 65
1.3 Unity3D instead of Android 65
2. Changes in planification 66
2.1 Planification after the decided changes: 66

Chapter 7: Conclusions 68
7.1 Summary of the project 68

References 70

Figures Index

Figure 1. ESPN league menu 10
Figure 2. ESPN standing menu 10
Figure 3. Comunio 11
Figure 4. Comunio menu 11
Figure 5. NBA General Manager 11
Figure 6. NBA 2K17 MyGM 1 12
Figure 7. NBA 2K17 MyGM 2 12
Figure 8. Hoops Rivals 1 12
Figure 9. Hoops Rivals 2 12
Figure 10. Gameplay 1 14
Figure 11. Flowchart 14
Figure 12. Start screen 15
Figure 13. Team choosing screen 16
Figure 14. Season screen 16
Figure 15. Stats screen 17
Figure 16. Training screen 17
Figure 17. Preferences screen 18
Figure 18. Game screen 18
Figure 19. End screen 19
Figure 20. Teams' logos 21
Figure 21. New game button 22
Figure 22. Load game button 22
Figure 23. Start screen background 22
Figure 24. Teams' buttons 23
Figure 25. Stats button 23
Figure 26. Train button 23
Figure 27. Preferences button 23
Figure 28. Play button 23
Figure 29. Stats background 24
Figure 30. Training background 24
Figure 31. Training buttons 25
Figure 32. Position buttons 25
Figure 33. Position buttons 2 26
Figure 34. Preferences background 26
Figure 35. Game background 27

Figure 36. End screen 27
Figure 37. Logos development 57
Figure 38. Background development 58
Figure 39. Stats and games background development 58
Figure 40. Training background development 59
Figure 41. Unity distribution 1 59
Figure 42. Unity distribution 2 60
Figure 43. Unity distribution 3 60
Figure 44. Gameplay 2 62
Figure 45. Gameplay 3 62
Figure 46 .Gameplay 4 63
Figure 47. Gameplay 5 63

Tables Index

Table 1. Original planification 66
Table 2. Final planification 67

Code Index

Code 1. Player training 31
Code 2. Small forward generator 32
Code 3. Assign players to teams in creation 33
Code 4. Calculate averages 34
Code 5. Assign players to teams 35
Code 6. Teams' skills levels 35
Code 7. Wins and loses getters and setters 36
Code 8. Set preferences 36
Code 9. Team training 37
Code 10. Training 38
Code 11: Set preferences 2 38
Code 12. Calculate standings 39
Code 13. Calculate top 5 scorers 40
Code 14. Calculate Most Valuable Player candidates 41
Code 15. Select shooter 43
Code 16. Calculate shot distribution percentages 43

Code 17. Select shooter 2 44
Code 18. Return shooter and defender 44
Code 19. Make play 45
Code 20. Shoot outside two points 46
Code 21. Defense stat 46
Code 22. Rebound 47
Code 23. Assist 48
Code 24. AI timeout 49
Code 25. User timeout 49
Code 26. Tire players 50
Code 27. Change players 51
Code 28. Close game 52
Code 29. Team name swap 53
Code 30. Save and load stats functions 54
Code 31. Serializable Stats data class 54
Code 32. Save stats 55

Chapter 1: Technical Proposal

1. Introduction

Nowadays mobile application is experiencing a huge boom. This is a great chance show to the world contents that will reach many people, so it makes it so much easier for little companies or developers to announce their products, or to transmit an idea to the public.
Currently female sport does not have too much visibility, and either has the means to reach people and show its potential. The window that mobile apps open can be used to accomplish this goal.
The main purpose of this project is to design and develop a female teams basketball simulator. This application will allow the user manage the team in all its aspects, so it will give knowledge of how a sports entity works (basketball in this case) in an attractive and entertaining way, promoting women's sport.
The simulator will have three big blocks:

- First, the user will have to manage aspects outside of the sport itself. This is, giving a great image for the fans, manage the tickets price, marketing, improving facilities… This will be some of the tasks the user will have to achieve to make its team the biggest of all and the best.
- Second, sport aspects that affect the game indirectly such as training, trade players… to build a successful team.
- Last, inside the game simulation the user will have the chance to prepare the strategies, like style of play or minutes management, to increase the possibilities of winning.

2. Related subjects

- VJ1208: Programming II
- VJ1209: 2D Design
- VJ1215: Algorithms and data structure
- VJ1220: Database
- VJ1227: Game Engines
- VJ1229: Mobile Device Applications

3. Objectives

- Implement a team management system
- Implement a realistic game simulator and statistics generator
- Adapt the systems to Unity
- Promote female sport

4. Planification

- Technical proposal: 10 hours
    - Lectures about how to write a technical proposal: 8h
    - Write the technical proposal: 2h
- Concept Design: 20 hours
    - Research how basketball teams and clubs work in female basketball
    - Aspects of sport entity management
    - Aspects of team management
    - Sport aspects
- Artistic Design: 40 hours
    - Menus design: 10 h
    - Interfaces design: 15 h
    - Design of elements for the random character creation: 13h
    - Sounds of the video game: 2h
- Programming: 190 hours
    - Implementation of the game simulation system: 50h
    - Implementation of the game management system: 20h
    - Implementation of the entity management system: 50h
    - Implementation of a player generator system from a database: 30h
    - Implementation of an AI able to manage games and entities: 30h
    - Assemble every system in Android: 10h
- Documentation: 40 hours
    - Write the Game Design Document: 10h
    - Write the Final Report: 20h
    - Prepare the defence of the project: 10h

5. Expected results

It is expected to develop every of the necessary systems for the simulator and put them together in order to perform correctly in a mobile application for the Android system.

- Game simulation: generate a realistic results and statistics taking into account players skills.
- Game management: User decisions have a real impact in the final result of the game simulator.
- Team management: It is expected to have a realistic behavior within its limitations.

6. Tools

- Android Studio
- Adobe Photoshop CC
- Adobe Audition
- Unity 3D
- MonoDevelop

7. References

- Android Studio [1]
- Adobe Photoshop CC [2]
- Adobe Audition CC [3]
- Unity 3D [4]
- MonoDevelop [5]

Chapter 2: Related work

1. References

A lot of people likes sports nowadays. It is one of the most known and popular ways of entertaining people. Also, a lot of people who likes sports would like to be able to do what professional sportists do and play like them or take control over their teams. Sports games allow them to do that, they give the user the chance to become a professional player (even the best of all) or manage and lead one team to success. In 2016, sports game were the fourth most played genre of videogames, according to Statista [6].

Usually the main requirement for a sport game is to be realistic. Even if it is a simulator or a management game, the results have to be credible.

Fantasy games use the results and performances of real players in a particular game. Usually, based on statistics turn their performances into game points. This games can be played to get the highest score during the season, matchups against other players or any other kind of competition. Some examples of this games are *ESPN Fantasy Games* [7] which include basketball, football, baseball, hockey… Using real players statistics player can create their own leagues and compete during the year.

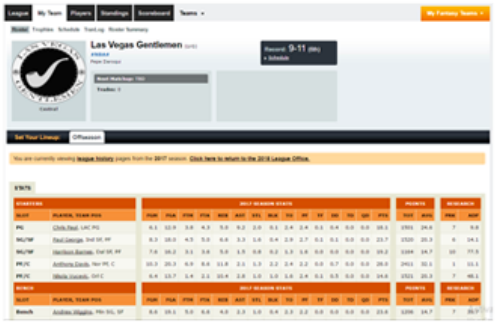

Figure 1. ESPN Basketball League Menu

Figure 2. ESPN Basketball Standings Menu

In Spain the most popular fantasy game is *Comunio* [8], a soccer manager based on La Liga. This game allows you to compete against other players, transfer players for money and manage your lineups.

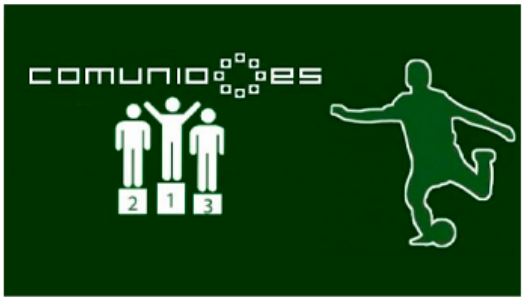

Figure 3. Comunio

Figure 4. Comunio menu

But manager games can be really more complete than that. If they are not based in real life, the possibilities are higher. Manager that use players and information for actual leagues can give the user a context to develop the team. One great example of that kind of games is *NBA General Manager* [9]. In this game, real life only as an impact in the level and skills of the different NBA players the game includes. The user can train players to affect the final score of the games, which are online against other users.

Figure 5. NBA General Manager

NBA 2K is the greatest basketball video game simulator, but also includes some management. In its latest game, *NBA 2K17*, users can manage almost every aspect of an NBA franchise in MyGM mode. In this mode, the user can do almost everything a general manager of a NBA franchise can do. In terms of franchise management, it is possible to improve facilities, set the price of the tickets and even the hot dogs at the stadium. Also the user has to manage contracts of the staff (coach, scout team…) and the players, even having the chance to have a conversation with them to negotiate.

In sport terms, the player can plan the train sessions by day, selecting between a lot of things, and plan the game strategy, from players minutes to systems they run and preferences in the way they play.

This might be one of the most complete managers in sports games, and also allowing the player to play the games.

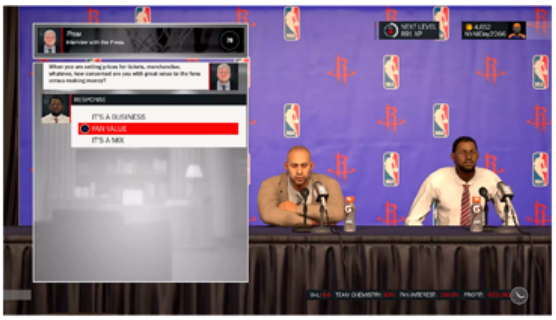

Figure 6. NBA 2K MyGM 1

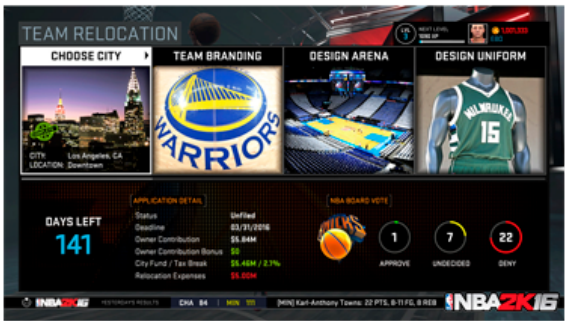

Figure 7. NBA 2k MyGM 2

Finally, this kind of games can perform really great in smartphones since usually they use a simple 2D menu with all the data necessary and does not need too much store and does not use too many memory and resources from the device. One example of an application that performs well in smartphones with 2D graphics is *Hoops Rivals* [10], which is a game with non real player and non real teams.

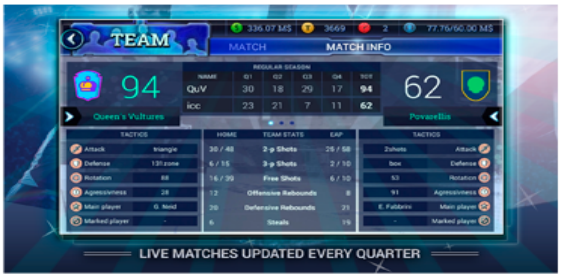

Figure 8. Hoops Rivals 1

Figure 9. Hoops Rivals 2

Chapter 3: Design

1. Statement

GM Baloncesto femenino is a 2D sports management, specifically a basketball manager, focused on female basketball. In this game, the user takes control over a basketball team and the goal of the game is to win as many leagues as possible, improving his players and planning strategies for the different games through the season. It is possible to have a little influence in the game calling time outs, but all the decision outside the game during the season will have an indirect influence on the results of the games. As all the teams are composed by female players, this might generate some interest in real female basket ball and can announced and show its potential to the world.

This game works for smartphones with the Android system. Every screen of the game is simple and intuitive, and displays all the necessary data in a friendly distribution, but there is no lack of information in them.

Who, what and how

- Who?: This game is for every basketball fan older than three years and capable of handling a smartphone or tablet. It is simple and intuitive, with no complicated aspects, so anyone who likes basketball can play it. As it has been mentioned before, it has the aim of generate some interest in female basketball.

- What?: It is easy to use and understand. The goal of the player is to make his team the best through all the seasons by training and making the right strategy and win as many championships as possible.

- How?: This game is an entertaining way to see how sport entities work, and by making the user have fun with female basketball will generate some curiosity in him.

2. Controls & Requirements

This game is played in a touchable screen by tapping the buttons. It has been implemented for android, and it is optimized for 16:9 screens. The minimum version of Android in the smartphone is 4.0 Lollipop.

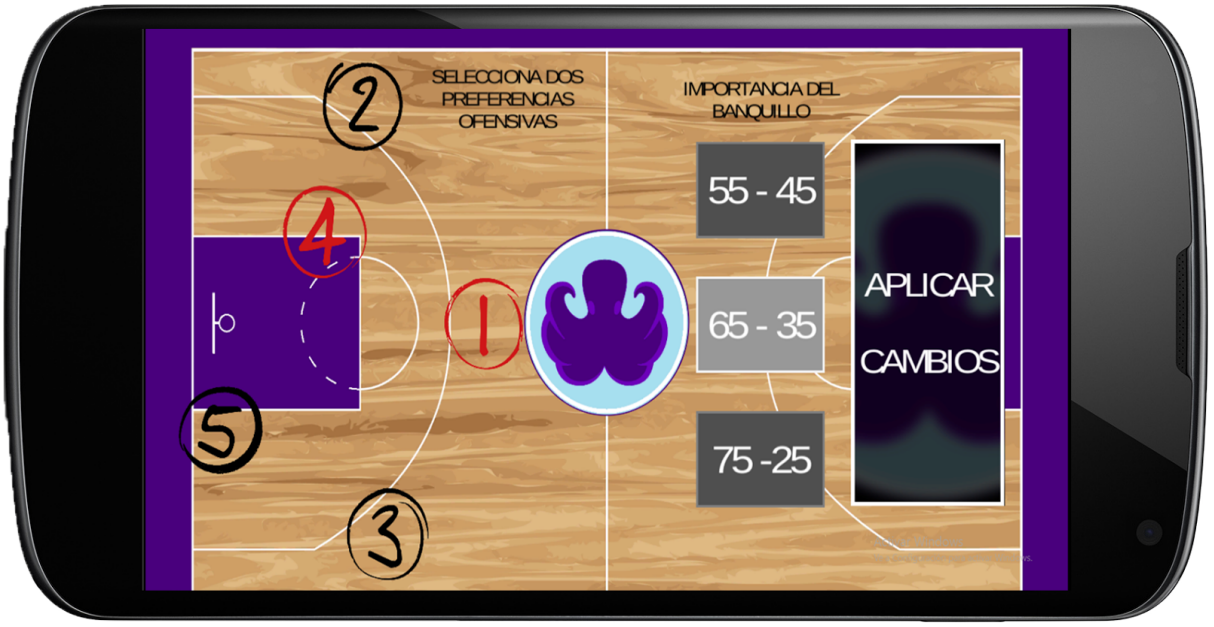

Figure 10. Gameplay 1

3. Game Interface

- **Flow chart**

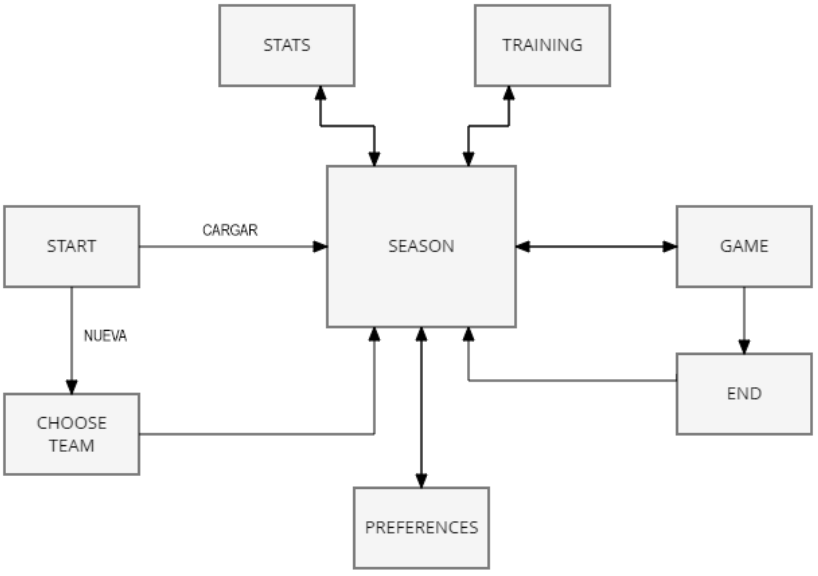

Figure 11. Flowchart

The first screen is the Start screen, where you can choose either New Game or Load Game. This screen appears only one time when the application starts running and never comes again. If New Game is chosen, there is another screen where the player can select the team he wants, and then takes him to the Season Screen. If Load Game is chosen, it takes the user directly to the Season Scene. In the Season Scene the user can go to the Training Scene, the Preferences Scene, and the Stats Scene. In this scenes there is only the possibility to go back to the Season Scene. If, in the Season Scene, the user plays a game, the screen shown will be Game Scene. When the game is done, unless the season is finished in which case will appear the Final Scene, it is always followed again by the Season Scene. The final Scene resets the season and starts a new one.

- **Start Screen**

First scene of the game, here the player can start a new game or load the one saved before. It shows the game title and the two buttons to start playing.

Figure 12. Start screen

- **Team Selection Screen**

This is scene where the player can choose the team he wants to manage. It displays all the teams, and when one is selected, the button to start the game shows the team's name, logo and average.

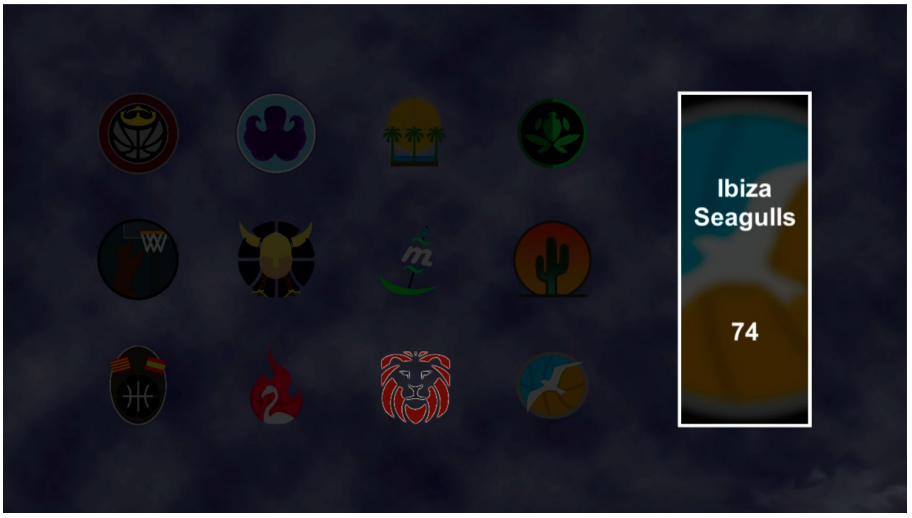

Figure 13. Team choosing screen

- **Season Screen**

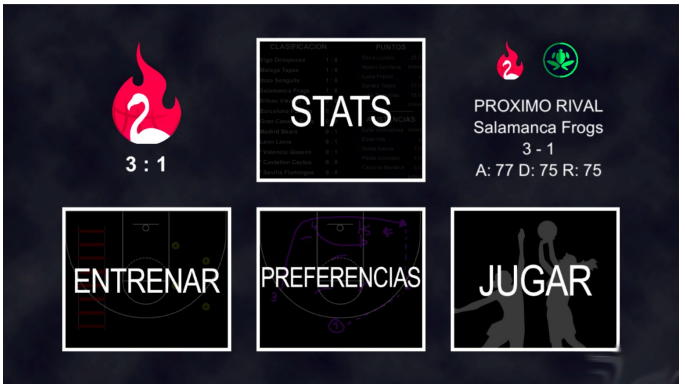

Figure 14. Season screen

It is the central screen of the game. Appart from Start Screen and Team Selection Screen, which only appear once in the game as it starts running, every other screen can only be accessed from the Season Screen, and from them there is only the possibility to go back.

It shows the general information of the user team (logo and record), the information of the next matchup (rival's logo, name, record, and general attributes like attack, defense and rebounding), and with the buttons allows the user to go to other screens:

- Stats Screen
- Training Screen
- Preferences Screen
- Game Screen

- **Stats Screen**

It is a display of the information of the course of the league. It shows

- Standings: How the teams are doing in the season, ordered by wins. It shows the position of the team, the name and the record.
- Statistics: It shows the best five players in the three basic aspects of the game (points, rebounds and assists), their names and the number of the specific statistic.
- Most Valuable Player race: It shows the three players that are doing the best in the season, displaying their names and team logos.

Figure 15. Stats screen

- **Training Screen**

In this screen the player can improve his player by training them. The user can select in which aspect of the game he wants to work and which position he wants to improve (point guard, shooting guard, small forward, power forward and center).

Figure 16. Training screen

The training improves both starter and bench player of the position chosen, and it can only be done three a week (one week between each game).

- **Preferences Screen**

In order to prepare the games, the user has to select the strategy, and this is the screen where he can do it. There are two different things he can choose:

- Preferences in shooting: It is possible to select none, one or two shooters. This selections are taken into account during the games. If there is no preference on shooting, the distribution of shots is equal for each position. If there is one or two preferences, those positions have a higher chance to get the shoot.
- Importance of the bench. Selecting the minutes distribution (55-45, 65-35, 75-25) the play that the players are on the court changes. The user has to see if the performance of his starters when they are tired affects them too much that they need to change.

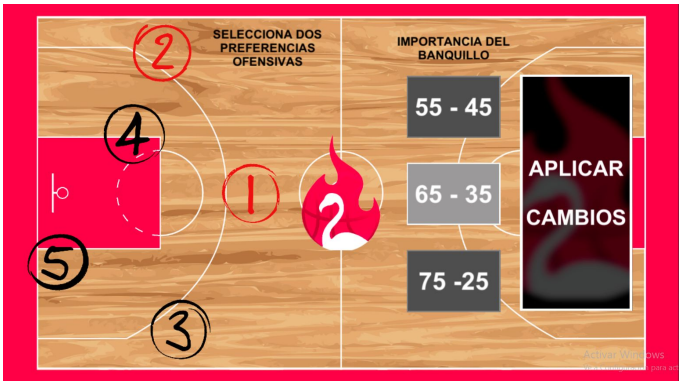

Figure 17. Preferences screen

- **Game Scene**

This is where the games happen. There is a display of both teams names and logos, record of each team, score, team stats and individual stats of each player on the court.
There is also the time out button, which changes its colour depending on the tendency of the game.

Figure 18. Game screen

- **Final Scene**

This scene shows the league champion's name and logo, and also who has won the mvp of the league. Pressing the button leads to a new season, where all stats and records are reseted to zero.

Figure 19. End Scene

4. Teams and players

- **Players**

  In this game all players are not real. Every time a new game starts, it generates random players that will be assigned for the teams according to their level and position.
  The attributes every player has are:
  - Attack:
    - Close two point shooting
    - Outside two point shooting
    - Three point shooting
  - Defense:
    - Interior defense
    - Outside defense
  - Rebounding
    - Offensive rebounding
    - Defensive rebounding

  All this attributes have a real impact in during the games and can be improved by training in big blocks.
  Also, each player has her own statistics that affect in the race for the MVP.

- **Teams**

  This games includes twelve teams from different cities of Spain, but they are not real either.
  The teams have name, a logo, and ten players distributed as:
  - Two superstars
  - Three good players
  - Two role players
  - Three bench players

  The user can improve his players and use personalized strategies against other teams to win the title. Depending on the team the user chooses, both training and preferences screen will display courts with the colours and logo of their team, so the atmosphere involves the player within the team.

5. Art Design

The style selected for the visual elements is simple, as its main goal is to show every important data and element in a friendly and intuitive way.

- **Teams**

  The style of the team logos is simple, almost minimal, and they have been designed from existing images and adapted to the style of the game:

Figure 20. Teams' logos

- **Start Screen**

  It only needs two buttons for start a new game or load game, and the game title.

Figure 21. New game button

Figure 22. Load game button

Figure 23. Start screen background

- **Team Selection Screen**

  It shows every team logo, and the button to start the scene it's personalized and shows the average skills of the team (via scripting, not in the current button)

Figure 24. Teams' buttons

- **Season Screen**

  There are four different buttons to navigate through the screens of the game: Statistics, training, preferences and games.

Figure 25. Stats button

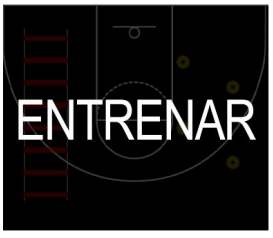

Figure 26. Train button

Figure 27. Preferences

Figure 28. Play button

- **Stats Scene**

  It only has a background where every information is organized and displayed properly. The space for each statistic part has to be big enough to show the five player but not too big in order to show all four parts.

Figure 29. Stats background

- **Training scene**

  For the background, every team has its own personalized court, with the zone and the stands coloured with the team standard colour.

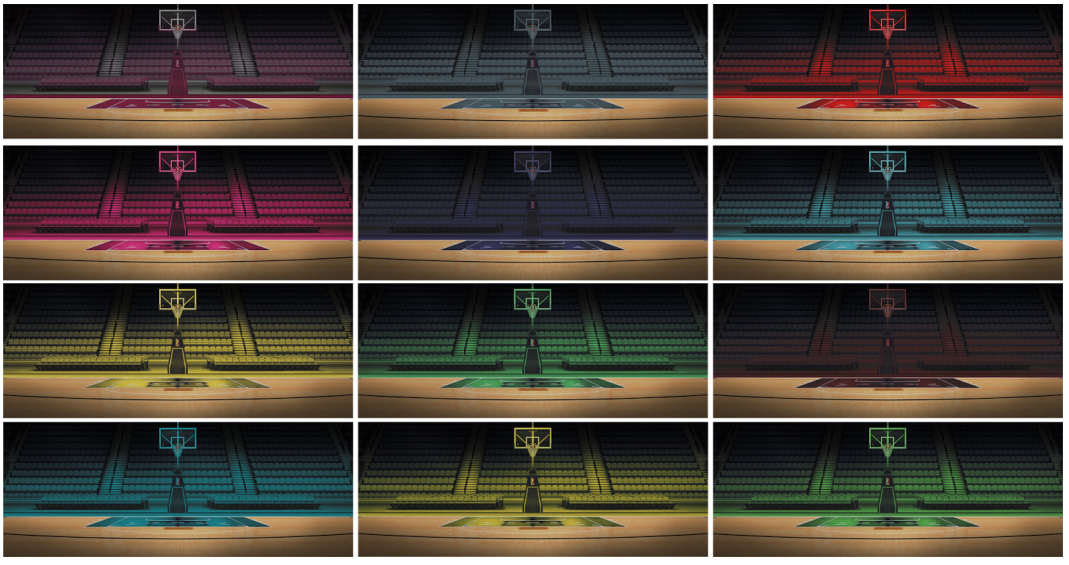

Figure 30. Training backgrounds

This screen has two different kind of buttons, five girl player silhouettes for select the position to train and five to select the attribute to train, apply the training and go back.

(The black part is transparent)

Figure 31. Training buttons

In order, point guard, shooting guard, small forward, power forward, center

Figure 32. Position buttons

- **Preferences Screens**

This screen has the design of a coach basketball board, as the preferences of the position try to imitate a handwritten numbers rounded out in the team's court.
The style of the minutes distribution is simple as the rest of the buttons in the game.
Finally, the button to apply every change is the same as the one to choose the team.

Figure 33. Position buttons 2

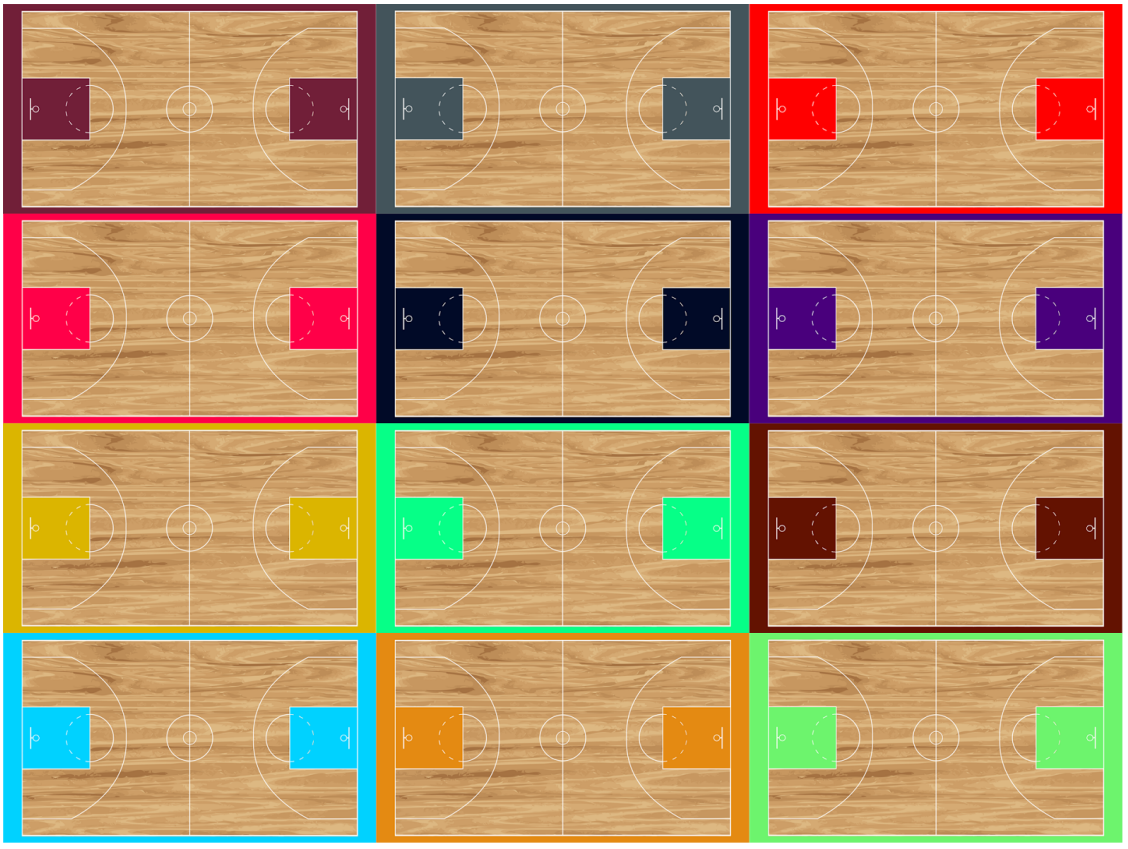

Figure 34. Preferences backgrounds

- **Game Screen**

  As the Stats screen, it is only a background to distribute every information displayed properly

Figure 35. Game background

- **Final Screen**

  In the final scene, there is only two logos, one for the winner team and another one for the MVP's team.

Figure 36. End Screen

Chapter 4: Development

1. Technical Proposal

In this section it described how the technical proposal of the project was redacted. This process includes the attendance to the four classes given to learn how to write properly a technical proposal and its final composition and delivery

1.1 Technical Proposal classes

From January 30th to February 2nd, there were four classes to help the students on their Technical proposal writing. This lectures were distributed during the four days on two hour classes per day.

In this classes, the lecturer gave tips for every part of the document. Every part of the class was divided in sections, one for each part of the proposal. Once the part was finished, it was uploaded to *Google Drive* to be corrected by any other student. There was also the chance to give tips and tell them how to improve the proposal.

When all the four classes were finished, the Technical proposal was almost finished and ready to be delivered

1.2 Final delivery of the Technical Proposal

Following the guidelines learned in the classes, it was not difficult to write the document. Every part of the document had its own difficulties, almost all of them related to the content and not the composition, since every possible doubt of that had been solved in the classes

The estimated programming and distribution of tasks for the project was the part where everything had to be well thought and planned, since the project calendar should be built from it.

Finally, for the final delivery of the whole project and the Final Report, it was revised, modified a little bit and translated to english, since it was written in spanish.

2. Concept Design

The concept design of this project consist getting information about how sport entities work, and how they work during the seasons. This information is crucial to make the game to perform as real as possible.

2.1 Information Research

Most of the information is based on personal experience as a female basketball team coach and manager. During a whole season (2016-2017) a lot of field work has been made.
At the beginning of the season a new team had to be build from nothing. Call players, design the team logos and marketing stuff, and, the most important part for the project, make plans for the season.
All the personal experience gained this year was really useful to know in first person how it feels to run a team, both inside and outside the court, and those experience led to some very important conclusions. Sport aspects are so much fun than extra-sport aspects. This was really took into account during the development of the project.

For the game development and statistics generator, more factor get into the equation.
First, twenty-five game reports have been collected, studied and analyzed to see how every player perform during the games. For each position, statistics were taken and compared by position and minutes played.

Then, the professional statistics of the WNBA were reviewed [11] in order to adapt the game to some professional performances.

2.2 Tools to develop the game

In first place, the game was going to be develop in Android Studio, but after making some research about how the tool works with graphics, it was changed to Unity3d since the tasks of the project barely change and it allows to do a better game with less problems and difficulties that Android Studio gives, and the result was going to be the same.

3. Programming

In this section it is going to be explained how every screen has been developed and integrated in the game

3.1 Players

The first step in this process was to create the PlayerClass.

Each player has to store the next basic information:

- Personal information:
    - Name and last name (it stores two indexes to a string list with fifty names and fifty last names, and it also stores those strings)
    - Number
    - Team
- Skills information:
    - Close two point shooting
    - Outside two point shooting
    - Three point shooting
    - Exterior defense
    - Interior defense
    - Defensive rebound
    - Offensive rebound
- Stats information: Each player stores its own stats performed in the games in an array. There are ten statistics collected.
    - Basic: Points, rebounds, assists, steals and blocks
    - Turnovers
    - Percentages: Field goals made, field goals attempted, three pointers made, three pointers attempted
- Each player has getters for all the skills data, personal data, and stats data. There are setters for team, name and lastname, and stats. Stats also can be reset with a function

- Finally, there are three functions modify (always improve) the skills, divided in three blocks: Attack, defense, and rebound, here is an example of one function. It improves all the skills a certain amount unless they are higher than 99, which is the limit for each attribute.

```
public void entrenarAta(int q) {
    if (pt3 + q <= 99) {
        pt3 += q;
    } else {
        pt3 = 99;
    }
    if (pt2Ext + q <= 99) {
        pt2Ext += q;
    } else {
        pt2Ext = 99;
    }
    if (pt2Int + q <= 99) {
        pt2Int += q;
    } else {
        pt2Int = 99;
    }
}
```

Code 1. Player training

With the player class created, the next step was to generate players randomly.

The process of creating each player consists on generate random name and attributes according to the position, and the level of the player.

- Point guards have great outside shooting (Three point shooting and Outside two point shooting) but at worse at shooting close to the rim (Close two point shooting). Also their defense is better outside, and their rebounding is not as high as the rest of positions, since usually point guards are the shortest players and play far away from the rim.
- Shooting guards shoot great from outside, even better than point guards, but their close shot is pretty similar. They defend better in the outside but can do it inside better than point guards, and rebounding is a little bit higher since those players are higher.
- Small forwards skills are good in every aspect, but they are not the best in anything. They can attack inside and outside, defend, and rebound, but they tend to play more outside than inside.

- Power forward is the first interior position. Outside shooting is lower than the positions below, but some of them can shoot well from outside the zone but inside the three point line. Their rebounding is so much higher since they are close to the rim, and the height gives them advantage when rebounding
- Center is the second interior position. As they play most of the time close to the rim, their interior shooting is the highest. Some of them can shoot from outside, but this case is something excepcional por this position. They have problems to defend little outside players because of its speed, but near the rim their height gives them some intimidation power.

With all this information, players can be created. The next part of the code generates small forward good players.

```
//Aleros
for (int i = 0; i < 7; i++) {
    int nombre = Random.Range (0, nombres.Length);
    int apellido = Random.Range (0, nombres.Length);
    int posicion = 3;
    int dorsal = Random.Range (0, 99);
    int equipo = -1;

    int p3 = Random.Range (60, 70);
    int p2e = Random.Range (75, 85);
    int p2i = Random.Range (75, 80);

    int defe = Random.Range (70, 80);
    int defI = Random.Range (70, 80);

    int rebo = Random.Range (70, 80);
    int rebd = Random.Range (70, 80);

    aplicarDatos (nombre, apellido, posicion, dorsal, equipo,
        p3, p2e, p2i, defe, defI, rebo, rebd, contador);
    contador++;
}
```

Code 2. Small forward generator

The first part of the code generates the personal info: name and last name from the names and lastnames arrays of strings, position (small forward in this case), number and does not assign a team since that will be done later.

The second part gives the player values to its attributes according to the information and considerations described below. There is a top and below limit so there can be some exceptions in order to create unique players and teams, with different levels but within the limits of a normal team and well distributed and balanced.

This process of creation generates a 120x12 matrix (120 players, 10 per team, and 12 attributes per player). This is what will be stored and once they are assigned to a team the player class will be created and added as a component to the teams.

Team assignment works as it is going to be described: Player generator creates the players in the following order:

- Superstars first
    - 5 Point guards, 5 shooting guards, 5 small forwards, 5 power forwards and 4 centers
- Good players
    - 7 Point guards, 7 shooting guards, 7 small forwards, 7 power forwards and 8 centers
- Role players
    - 5 Point guards, 4 shooting guards, 5 small forwards, 5 power forwards and 4 centers
- Bench players
    - 7 Point guards, 8 shooting guards, 7 small forwards, 7 power forwards and 7 centers

Then, every team selects the first player available if they don't have one on this position or if they have already selected one for each position. This creates teams with similar lever but different properties. A team with point guard and shooting guard superstars is different than a team with power forward and center superstars. The next function makes the team assignments.

```
void draft(){
    bool sigue = true;

    for (int i = 0; i < 10; i++) { //10 rondas
        for (int j = 0; j < 12; j++) { //12 equipos
            sigue = true;
            for (int k = 0; k < 120 && sigue; k++) {
                if (Jugadoras[k,4] == -1) {
                    if (draftMatrix[j,Jugadoras[k,2] - 1] == 0) {
                        Jugadoras[k,4] = j;
                        draftMatrix [j, Jugadoras[k,2] - 1] = 1;
                        sigue = false;
                        bool primeraVuelta = true;
                        for (int l = 0; l < 5; l++) {
                            if (draftMatrix[j, l] == 0) {
                                primeraVuelta = false;
                                break;
                            }
                        }
                        if (primeraVuelta) { //Resetear Matriz
                            for (int l = 0; l < 5; l++) {
                                draftMatrix [j, l] = 0;
                            }
                        }
                    }
                }
            }
        }
    }
}
```

Code 3. Assign players to teams in creation

3.2 Teams

This game has twelve teams with ten players. Each team stores its players and some information necessary for the game:

- Team number
- Name (City and name)
- Logo
- Workout courts
- Victories and losses
- Team stats
- Team preferences
    - Two shooting preferences
    - Minutes distribution
- Team attributes (calculated as average of players attributes)
    - Attack
    - Defense
    - Rebound
    - Average

The next function calculates the team attributes, so they can be used during the game:

```
public void calcularMedias() {
    for (int i = 0; i < jugadoras.Length; i++) {
        int totalJ = jugadoras [i].devolver3Pt() + jugadoras [i].devolver2PtInt()
            + jugadoras [i].devolver2PtExt();
        float mediaJ = totalJ / 3;
        ataque += mediaJ;
    }
    ataque = ataque / jugadoras.Length;

    for (int i = 0; i < jugadoras.Length; i++) {
        int totalJ = jugadoras [i].devolverDefExt() + jugadoras [i].devolverDefInt();
        float mediaJ = totalJ / 2;
        defensa += mediaJ;
    }
    defensa = defensa / jugadoras.Length;

    for (int i = 0; i < jugadoras.Length; i++) {
        int totalJ = jugadoras [i].devolverRebDef() + jugadoras [i].devolverRebOfe();
        float mediaJ = totalJ / 2;
        rebote += mediaJ;
    }
    rebote = rebote / jugadoras.Length;

    media = (ataque + defensa + rebote) / 3;
}
```

Code 4. Calculate averages

Using the player matrix (with team data updated) every team selects its players. With the gameObject added as a component for the team, the attributes are taken from the player matrix to the player class, and this process is done ten times since there are only ten players per team, so when they are all assigned the loop can be stopped. The following function shows the process of doing that:

```
public void asignarJugadoras(int team) {

    int[,] jugs = new int[120, 12];
    jugs = mg.devolverJugadoras ();
    int k = 0;

    //Segun el equipo, sacar de la base de datos y crear jugadoras

    for (int i = 0; i < 120; i++) {
        if (jugs[i,4] == team) {
            jugadoras [k] = gameObject.AddComponent<PlayerClass> ();
            jugadoras [k].asignarVariables (jugs [i, 0], jugs [i, 1], jugs [i, 2],
                jugs [i, 3], jugs [i, 4], jugs [i, 5], jugs [i, 6], jugs [i, 7],
                jugs [i, 8], jugs [i, 9], jugs [i, 10], jugs [i, 11]);
            jugadoras [k].setNomYApe (mg.devolverNombre (jugs [i, 0]),
                mg.devolverApellido (jugs [i, 1]));
            k++;
        }
        if (k == 10) {
            break;
        }
    }
}
```

Code 5. Assign players to teams

As it is necessary during the game, each team has four getters for each team attribute (attack, defense, rebounding and average). Since this is only used as information, the numbers are rounded to show them without decimals. Only rebound information is used for more than just information display.

```
public int devolverAta () { return Mathf.RoundToInt(ataque); }
public int devolverDef () { return Mathf.RoundToInt(defensa); }
public int devolverReb () { return Mathf.RoundToInt(rebote); }
public int devolverMed () { return Mathf.RoundToInt(media); }
```

Code 6. Teams' skills levels

Getters and setters for victories and losses work a little bit different. Getter works as any other getter, returning the value of the victories or losses. But there are two different kind of setters. The standard setter only adds one to victories or losses. The reason of doing that is to ensure that when a game is finished it does not add more wins or losses so it works properly. The other setter sets the wins and losses as the value provided. This is only used for load purposes.

```
public int devolverV () { return victorias; }
public void setV() { victorias++; }
public int devolverL () { return derrotas; }
public void setL() { derrotas++; }
public void cargarV(int v) { victorias = v; }
public void cargarL(int l) { derrotas = l; }
```

Code 7. Wins and loses getters and setters

For team preferences, getters work as normal getters, but setters for shooting preferences have to do some extra functionality and take some conditions into account.

There can only be none, one or two shooting preferences, so if there is no preference selected and one is assigned, the other one must be zero. If the user adds another one it is updated. But, when there are two preferences, if another one is selected, the first one to be chosen is rejected and updated. So, the last preference to be selected will be the last preference to be discarded, if necessary. Also, if the user wants to change from two to one preferences, this is done by deleting the first preference selected and updating it to the new one, and the second one is assigned as none.

```
public void setEleccion (int ele) {
    if (eleccion1 == 0) {
        eleccion1 = ele;
    } else if (eleccion2 == 0 && ele != eleccion1){
        eleccion2 = ele;
    } else if (ele != eleccion1) {
        eleccion1 = eleccion2;
        eleccion2 = ele;
    }
    if (eleccion1 == eleccion2) {
        eleccion2 = 0;
    }
}

public void setReparto (int rep) {
    reparto = rep;
}
public int devolverReparto () { return reparto; }
public int devolverE1 () { return eleccion1; }
public int devolverE2 () { return eleccion2; }
```

Code 8. Set preferences

Every team has getters for every attribute of every player. A function with player and attribute parameters returns the information required. This is used during games and will be described where games are explained.

Also each team has a function to train its players. This function just receives as parameters the position to be improved, the attribute to improve and the amount of improvement, and calls the players train function to update attributes.

```
public void entrenar(int pos, int atr, int cant) {
    foreach (PlayerClass jug in jugadoras) {
        if (jug.devolverPosicion () == pos) {
            if (atr == 1) {
                jug.entrenarAta (cant);
            } else if (atr == 2) {
                jug.entrenarDefensa (cant);
            } else if (atr == 3) {
                jug.entrenarRebote (cant);
            }
        }
    }
}
```

Code 9. Team training

This is all what a team needs to work as expected. The players are stored, their attributes can be accessed and improved, and the team strategies can be setted and updated.

3.3 Training

The objective of this script is select the positions and attributes to be trained. Most of the script handles the button events to select the desired position and attribute. Once they are selected, the button *apply* makes the training, which works as described below.

There is a random between one and five which defines the amount of improvement of the workout. This is done because not all workouts have the same effect on players, so they may improve just a little bit or more, but one workout can not make a player improve a lot, they need a lot of workouts in a long period of time to get results.
So, the team is limited to three trainings between games, considering a training every time *apply* button is pressed. The player has to consider who wants to

improve and in which attribute to get the better result. Of course, training has a direct impact on game results.
The next function is the one that applies the improvements. It makes a call to the team training function which calls player training as described below.

```
void training() {
    if (mg.acciones < mg.accMax) {
        int mej = Random.Range (1, 5);
        mg.mejoras [pos-1, atr-1] += mej;
        mg.SaveTrain ();
        team.entrenar (pos, atr, mej);
        mg.acciones++;
    }
}
```

Code 10. Training

3.4 Team preferences

Here is where the user can select his preferences. As in training script, most of the code takes care of handling the buttons, but in this case, once the button is pressed the preference es updated in that moment. This is done because there is no limit to change preferences. The next code updates the preferences for the team:

```
void min1() { team.setReparto (1); }
void min2() { team.setReparto (2); }
void min3() { team.setReparto (3); }

void click1() { team.setEleccion (1); }
void click2() { team.setEleccion (2); }
void click3() { team.setEleccion (3); }
void click4() { team.setEleccion (4); }
void click5() { team.setEleccion (5); }
```

Code 11: Set preferences 2

3.5 Stats screen

The goal of this screen is to display the standings of the league, the leaders in points per game, rebounds per game, assists per game, and the three best player fighting for the MVP. To do that, it takes into account teams' wins and losses and statistics data of each player.

Once this screen is opened, it calculates everything that it needs and stores it in the corresponding array. There are five different arrays necessary for each of the information displayed:

- The standings array stores information of each team, and sorts them depending on the amount of wins. Since all the teams play the same games at the same time, every
- The arrays for points, rebounds, assists, and MVP race only needs to store the team and the player.

The next code shows how the standings are calculated and sorted by wins:

```
void chart() {
    for (int i = 0; i < 12; i++) {
        Vteam [i,0] = teams [i].devolverV ();
        Vteam [i,1] = teams [i].equipo;
    }

    for (int i = 0; i < 12; i++) {
        for (int j = 0; j < 12; j++) {
            if (Vteam[i,0] > Vteam [j,0]) {
                int aux = Vteam [j,0];
                int aux2 = Vteam [j, 1];
                Vteam [j, 0] = Vteam [i, 0];
                Vteam [j, 1] = Vteam [i, 1];
                Vteam [i, 0] = aux;
                Vteam [i, 1] = aux2;
            }
        }
    }
}
```

Code 12. Calculate standings

Wins and team number are stored in the standings matrix (Vteam) with no order. Then, they are sorted by wins exchanging positions. The final result is the standing in the correct order.

The following code calculates the top 5 scorers of the league:

```
void ppg(){
    int max1 = 0, max2 = 0, max3 = 0, max4 = 0, max5 = 0;

    for (int i = 0; i < 12; i++) {
        for (int j = 0; j < 10; j++) {
            if (teams[i].devolverJugadora(j).devolverStats()[0] > max1) {
                max5 = max4;
                max4 = max3;
                max3 = max2;
                max2 = max1;

                pointsPG [4, 0] = pointsPG [3, 0];
                pointsPG [4, 1] = pointsPG [3, 1];
                pointsPG [3, 0] = pointsPG [2, 0];
                pointsPG [3, 1] = pointsPG [2, 1];
                pointsPG [2, 0] = pointsPG [1, 0];
                pointsPG [2, 1] = pointsPG [1, 1];
                pointsPG [1, 0] = pointsPG [0, 0];
                pointsPG [1, 1] = pointsPG [0, 1];
                pointsPG [0, 0] = i;
                pointsPG [0, 1] = j;

                max1 = teams [i].devolverJugadora (j).devolverStats () [0];
            } else if (teams[i].devolverJugadora(j).devolverStats()[0] > max2) {
                max5 = max4;
                max4 = max3;
                max3 = max2;

                pointsPG [4, 0] = pointsPG [3, 0];
                pointsPG [4, 1] = pointsPG [3, 1];
                pointsPG [3, 0] = pointsPG [2, 0];
                pointsPG [3, 1] = pointsPG [2, 1];
                pointsPG [2, 0] = pointsPG [1, 0];
                pointsPG [2, 1] = pointsPG [1, 1];
                pointsPG [1, 0] = i;
                pointsPG [1, 1] = j;

                max2 = teams [i].devolverJugadora (j).devolverStats () [0];
            } else if (teams[i].devolverJugadora(j).devolverStats()[0] > max3) {
                max5 = max4;
                max4 = max3;

                pointsPG [4, 0] = pointsPG [3, 0];
                pointsPG [4, 1] = pointsPG [3, 1];
                pointsPG [3, 0] = pointsPG [2, 0];
                pointsPG [3, 1] = pointsPG [2, 1];
                pointsPG [2, 0] = i;
                pointsPG [2, 1] = j;

                max3 = teams [i].devolverJugadora (j).devolverStats () [0];
            }
            else if (teams[i].devolverJugadora(j).devolverStats()[0] > max4) {
                max5 = max4;

                pointsPG [4, 0] = pointsPG [3, 0];
                pointsPG [4, 1] = pointsPG [3, 1];
                pointsPG [3, 0] = i;
                pointsPG [3, 1] = j;

                max4 = teams [i].devolverJugadora (j).devolverStats () [0];
            }
            else if (teams[i].devolverJugadora(j).devolverStats()[0] > max5) {
                pointsPG [4, 0] = i;
                pointsPG [4, 1] = j;
                max5 = teams [i].devolverJugadora (j).devolverStats () [0];
            }
        }
    }
}
```

Code 13. Calculate top 5 scorers

The work of the function is simple. It compares the points scored by all players with the top 5 scorers. If it is greater than one of them, they are all updated. When the process is finished the matrix stores the teams and players of the top scorers.
The same goes for rebounds and assists, the process is exactly the same, but changing the stat to rebounds and assists.

For calculating the MVP, it goes a little bit different. The race for the MVP takes into account all five basic statics attributes, but it gives them a different importance:

- Points are multiplied per six
- Rebounds per two
- Assists per three
- Steals and blocks remain the same

This is done because scoring is the most important thing in basketball, so it has to have a higher weight in calculating the MVP.

```
void mvp(){
    int max = 0, max2 = 0, max3 = 0;

    for (int i = 0; i < 12; i++) {
        for (int j = 0; j < 10; j++) {
            int total = ((teams [i].devolverJugadora (j).devolverStats () [0] * 6) +
                (teams [i].devolverJugadora (j).devolverStats () [2] * 2) +
                (teams [i].devolverJugadora (j).devolverStats () [1] * 3) +
                teams [i].devolverJugadora (j).devolverStats () [3] +
                teams [i].devolverJugadora (j).devolverStats () [4]);
            if (total > max) {
                max3 = max2;
                max2 = max;

                mvpG [2, 0] = mvpG [1, 0];
                mvpG [2, 1] = mvpG [1, 1];
                mvpG [1, 0] = mvpG [0, 0];
                mvpG [1, 1] = mvpG [0, 1];
                mvpG [0, 0] = i;
                mvpG [0, 1] = j;
                max = total;
            } else if (total > max2) {
                max3 = max2;

                mvpG [2, 0] = mvpG [1, 0];
                mvpG [2, 1] = mvpG [1, 1];
                mvpG [1, 0] = i;
                mvpG [1, 1] = j;
                max2 = total;
            }   else if (total > max3) {
                mvpG [2, 0] = i;
                mvpG [2, 1] = j;
                max3 = total;
            }
        }
    }
    man.provisionalMVP [0] = mvpG [0, 0];
    man.provisionalMVP [1] = mvpG [0, 1];
}
```

Code 14. Calculate Most Valuable Player candidates

3.6 Games

Games are the most complete element of the game. It uses every attribute of teams and players and generates realistic results and statistics.

Games as real basketball games, face two teams against each other, in four quarters of ten minutes plus any necessary extra time of five minutes if the final result is a tie.
In this simulator, the process of the game works as it follows:

- Possessions last between 4 and 24 seconds. As real basketball, fast breaks last a few seconds and long possessions are limited to 24.
- In each possession, the team with the ball selects the shooter, which can decide between shooting for three, outside two, or close two, based on studied percentages and players skills.
- Then, before the shot is made, there is a chance for the other team for stealing the ball or it is possible for the team with the ball to lose it.
- If the shot is made, real percentages and skills are taking into account to see if the shot is converted or not. If it is done, the possession changes and updates the statistics (points, field goals, assists…). If it does not go in, it generates a rebound, which can be offensive or defensive.
- Players can get tired, on fire (when they make some shots in a row they are more likely to make the next shots), and are replaced according to their energy and their role on the team (minutes preferences).
- The user and AI can call timeouts to change the tendency of the game.
- The possession game speed is 0.25 seconds per update, so the user can see what is happening and has time to react when needed.

3.6.1 Select shooter

The first thing to do when the shooter is selected is to see what role the player have and how the shot distribution is determined. For AI teams the distribution is the same for each position. For user teams the distribution change according to it has been selected in the team preferences.

```
int[] selectShooter (int t) {
    int shooter;
    int defender;
    int[] porcentajes = new int[5];
    if (t == 1) {
        if (team1.devolverE1() == 0) {
            for (int i = 0; i < 5; i++) {
                porcentajes [i] = 20;
            }
        } else if (team1.devolverE2() == 0){
            for (int i = 0; i < 5; i++) {
                if (team1.devolverE1() == i+1) {
                    porcentajes [i] = 28;
                } else {
                    porcentajes[i] = 18;
                }
            }
        } else {
            for (int i = 0; i < 5; i++) {
                if (team1.devolverE1() == i+1 || team1.devolverE1() == i+1) {
                    porcentajes [i] = 26;
                } else {
                    porcentajes[i] = 16;
                }
            }
        }
    }
```

Code 15. Select shooter

With the percentages calculated, the next step is to assign each one to the correspondent position. The shooter will be selected with a random number between 0 and 100, so the percentages are added instead of assigned.

```
int rango = Random.Range (0, 100);
int p1, p2, p3, p4, p5;
p1 = porcentajes [0];
p2 = porcentajes [0] + porcentajes [1];
p3 = porcentajes [0] + porcentajes [1] + porcentajes [2];
p4 = porcentajes [0] + porcentajes [1] + porcentajes [2] + porcentajes [3];
p5 = porcentajes [0] + porcentajes [1] + porcentajes [2] + porcentajes [3] + porcentajes [4];
```

Code 16. Calculate shot distribution percentages

The variable *rango* is the one used to select the shooter. Once the the shooter is selected, the function returns a pair of players, the shooter and the defender of the shooter, since attributes of both players will be used to get the success of the shot.

```
if (t == 1) {
    if (rango < p1) {
        shooter = base1;
        defender = base2;
    } else if (rango < p2) {
        shooter = escolta1;
        defender = escolta2;
    } else if (rango < p3) {
        shooter = ala1;
        defender = ala2;
    } else if (rango < p4) {
        shooter = alapivot1;
        defender = alapivot2;
    } else {
        shooter = pivot1;
        defender = pivot2;
    }
}
```

Code 17. Select shooter 2

```
int[] dupla = new int[2];
dupla [0] = shooter;
dupla [1] = defender;
return dupla;
```

Code 18. Return shooter and defender

3.6.2 Select shot

Once the shooter is selected, the next step is to know if there is a shot or not, and if it is where it shooted.
The next code generates either a shot or an unforced turnover (no steal). The probability of taking a shot has been changed constantly during the project until getting a realistic result and statistics.

The first decision the shooter takes is to shoot from three points or from two points. Even the best three point shooter in real life shot barely one third of their shots from three point. So the probability of taking a three point shot is proportional to the three point shoot attribute of the shooter, so players with this skill in a high level shoot more from there than players with bad level. This probability is adapted to real life shooting multiplying it by 0.33.

Then, if the selected shot is a three pointer its function will be called. Instead, if the shot is a two pointer, there is one more decision to make.

Players who shoot better close to the basket will shoot more from there, but players who shoot similar from inside and outside the zone will take whatever shot they can. So this decision is taken comparing the attributes *close two point shooting* and *outside two point shooting*. When the decision is done, the shot is taken.

Finally, according to the result of the shot (if it goes in or not) the state *on fire* is updated to be used in the next possessions.

The next code shows the explanation above:

```
void makePlay (int pos) {
    int[] tira = new int[2];
    tira = selectShooter (pos);
    int rango, eleccion1, eleccion2;

    if (pos == 1) {
        //Modificar valores segun atributos de equipo
        rango = Random.Range(0,100);
        if (rango < 85) {
            eleccion1 = Random.Range (0,100);
            if (eleccion1 < team1.devolverStat(tira[0], pt3) * 0.33f) {
                shoot3p (tira[0], tira[1], possession);
            } else {
                eleccion2 = Random.Range(0, team1.devolverStat(tira[0], p2e) + team1.devolverStat(tira[0], p2i));
                if (eleccion2 < team1.devolverStat(tira[0], p2e)) {
                    shoot2pe (tira[0], tira[1], possession);
                } else {
                    shoot2pi (tira[0], tira[1], possession);
                }
            }
            updateFire (tira [0], possession);
        } else {
            turnoverStat (possession);
        }
    }
```

Code 19. Make play

3.6.3 Shooting

There is a function for each kind of shoot, but they only changed in certain numbers that modify the chances of making the shoot and they work almost the same. The process of a shot is the next.

The first test a shot has to pass is against the defender. Three pointers and outside two pointers are compared with the outside defense of the defender, and close two pointers with the inside defense. There is a little bonus for the shooter since basketball is a game where attacking is easier than defending, and the result of the shot depends more on the attacker skills, but of course, defense has an influence.

Once this tests is passed, there is a second test to adapt percentages to the real ones, and the explanation of this is simple. Even the best shooter fail some open shots, and the probability of failing gets higher proportionally to the distance of the

shoot. So this modifier changes according to the kind of shoot that has been taken. If the shoot goes in, the points are added and the stats updated. If not, it generates a rebound manage on the *defenseStat* function, that will be explained later.

The next code shows how this process works for outside two point shooting, but it works the same for three point shooting and close two point shooting. In the code, it can be seen how the offense and defense attributes are compared to get the first test. In the second test, the comparison takes into account if the player is on a good run (on fire), and also uses the energy of the player. If the player is tired, the chances of making the shot are less than normal, as in real life.

```
void shoot2pe (int t, int d, int p) {
    int exito1, exito2;
    if (p == 1) {
        exito1 = Random.Range (0, team1.devolverStat (t, p2e) + team2.devolverStat (d, defe));
        if (exito1 < team1.devolverStat (t, p2e) + 10) {
            exito2 = Random.Range (0, 100);
            if (exito2 < 80 + (tendenciasJ1[t, fire] * 5) - ((int) (tendenciasJ1[t, energia]) * 0.5f)) {
                score2p (p, quarter, t);
                assist (p, t);
                t1Stats [t, p2m]++;
                tendenciasJ1 [t, consec]++;
            } else {
                defenseStat ();
                tendenciasJ1 [t, consec] -= 2;
            }
        } else {
            defenseStat ();
            tendenciasJ1 [t, consec] -= 2;
        }
        t1Stats [t, p2a]++;
    }
```

Code 20. Shoot outside two points

3.6.4 DefenseStat function

There are two reasons why a shot does not go in. The first one is because the shooter just has failed the shot, and after that, there is a rebound. But some shots (nearly 20-25%) are failed because one defender has blocked the shot. *DefenseStat* takes this into consideration. In both cases, there is a rebound, but if there is a block it has to be updated. The next code shows the function

```
void defenseStat() {
    cambio = 1;
    statDefensiva = Random.Range (0, 100);
    if (statDefensiva < 75) {
        cambio = rebound (possession);
    } else {
        block (possession);
        cambio = rebound (possession);
    }
}
```

Code 21. Defense stat

3.6.5 Rebounding

One shot failed generates a rebound. This rebound can be defensive if the team defending gets the ball, or offensive if the team that shooted gets the ball again. So when a rebound occurs the possession might not change. This has to taken into consideration to generate more realistic results and games. About a third of the rebounds of a game are offensive rebounds, but it may change according to the teams' rebounding.
Also, interior players have a high possibility of getting rebounds, since most of them fall close to the basket and that is where they tend to be located in the court. Also, they take more rebounds because of their height.

Rebound function uses all this information to generate realistic rebounding. It updates the stats and return "1" if the possession has to change (defensive rebound) or "2" if the possession does not change (offensive rebound). Here is part of the code for team 1 shots.

```
int rebound (int p) {
    int rango;
    int reboteador = Random.Range (0, 100);

    if (p == 1) {
        rango = Random.Range (0, team1.devolverReb() + (team2.devolverReb() * 3));
        if (rango < team2.devolverReb() * 3)  { //Rebote defensivo

            if (reboteador < 10)
                t2Stats [base2, reb]++;
            else if (reboteador < 20)
                t2Stats [escolta2, reb]++;
            else if (reboteador < 40)
                t2Stats [ala2, reb]++;
            else if (reboteador < 70)
                t2Stats [alapivot2, reb]++;
            else t2Stats [pivot2, reb]++;

            return 1;

        } else { //Rebote ofensivo
            if (reboteador < 10)
                t1Stats [base1, reb]++;
            else if (reboteador < 20)
                t1Stats [escolta1, reb]++;
            else if (reboteador < 40)
                t1Stats [ala1, reb]++;
            else if (reboteador < 70)
                t1Stats [alapivot1, reb]++;
            else t1Stats [pivot1, reb]++;

            return 2;
        }
    }
```

Code 22. Rebound

3.6.6 Assists

As a sports team, baskets can come from a good pass that generates the good situation for a shot. This is called assist. Usually, the outside players make more assists since they have the ball in their hand more time than interior players. Point guards, the ones who organized the team and run the plays are the ones with more assists. But not all the baskets come from a good pass, they may come from an individual play, so, in the function *assist* that is taken into consideration.

```
void assist (int p, int t) {
    int exito = Random.Range (0, 100);
    int asistente = 0;

    if (exito < 85) {
        asistente = Random.Range (0, 100);
        if (p == 1) {
            if (asistente < 30) {
                asistente = base1;
            } else if (asistente < 50) {
                asistente = escolta1;
            } else if (asistente < 70) {
                asistente = ala1;
            } else if (asistente < 85) {
                asistente = alapivot1;
            } else {
                asistente = pivot1;
            }
        } else {
            if (asistente < 30) {
                asistente = base2;
            } else if (asistente < 50) {
                asistente = escolta2;
            } else if (asistente < 70) {
                asistente = ala2;
            } else if (asistente < 85) {
                asistente = alapivot2;
            } else {
                asistente = pivot2;
            }
        }
        if (p == 1 && asistente != t) {
            t1Stats [asistente, asi] += 1;
        }
        if (p == 2 && asistente != t) {
            t2Stats [asistente, asi] += 1;
        }
    }
}
```

Code 23. Assist

3.6.7 Steals and blocks

This are defensive stats that are generated after a fail or a turnover. They work as the *assist* function, the chances of getting the steal/block depend on the position.

- Steals usually are made by defender of outside players
- Blocks usually are made by inside defenders.

3.6.8 Tendencies

In basketball, the tendencies of the game are really important. It is so strange to see a continuous exchange of baskets. Usually the teams go through positive or negative runs, which affect the confidence of the players and, indirectly, the results. As it can be seen in the shooting functions, when the soot is made, the tendencies change. A team who makes a basket will be more likely to make the next one, and a team in a bad run will have less possibilities of making a shoot. This is how this part of basketball is reflected on the game and the simulation, giving pretty good results.

3.6.9 Timeouts

The way the coaches have to stop a bad run and modify the tendency of the game is call a timeout. In this game, timeouts reset the tendencies so once a team enters a bad run it does not stay there for the rest of the game. Of course, teams can leave that bad run by playing, but as in real life, a timeout helps.

AI calls timeout when the tendency is more or less 20. There is a random value because not all teams and coaches react at the same time and it is not something purely mathematical.

The next two functions show what timeouts do for user and AI teams:

```
void timeOutIA () {
    if (minutos2 < maxMinutos) {
        tendencia1 = 0;
        tendencia2 = 0;
        corriendo = false;

        minuto = true;
    }
}
```

Code 24. AI timeout

```
void timeOut () {
    if (corriendo) {
        if (minutos1 < maxMinutos) {
            tendencia1 = 0;
            tendencia2 = 0;
            minutos1++;
            corriendo = false;

            minuto = true;
        }
    } else {
        timeOutS = duracionTO;
    }
```

Code 25. User timeout

3.6.10 Players states

There are two basic states that have been included in the game. Good runs (on fire) and energy:

- Good runs: When a player hits some shoots in a row (three in the game), the confidence gets higher, so this makes the player more likely to make the next shoots. But if he fails one or two, this state disappears since the confidence comes back to its normal state.
- Energy. Players are people, and are doing an active activity, so if they do it continuously without stopping, their performance will be affected. This has been included as a functionality in the game. Players get tired using a random value (in real life, in one possession player may make a sprint and the next possession can rest a little bit, depending on a random situation). And if the players are resting in the bench, their energy resets so they can play again.
  The function that gets the players tired is the next one:

```
void cansar(){
    //Cansancio
    for (int i = 0; i < tendenciasJ1.GetLength(0); i++) {
        if (i == base1 || i == escolta1 || i == ala1 || i == alapivot1 || i == pivot1 ) {
            tendenciasJ1 [i, energia] += Random.Range (1, 2);
        } else {
            if ((tendenciasJ1 [i, energia] - 10) > 0) {
                tendenciasJ1 [i, energia] -= 10;
            }
        }
    }

    for (int i = 0; i < tendenciasJ2.GetLength(0); i++) {
        if (i == base2 || i == escolta2 || i == ala2 || i == alapivot2 || i == pivot2 ) {
            tendenciasJ2 [i, energia] += Random.Range (1, 2);
        } else {
            if (tendenciasJ2 [i, energia] - 10 > 0) {
                tendenciasJ2 [i, energia] -= 10;
            }
        }
    }
}
```

Code 26. Tire players

The first condition in the loop is to see which of the both players by position is on the court. The one on the court will get tired, and the one in the bench will be resting and the energy will increase (The amount of energy is subtracted when used, so max energy is 0).

3.6.11 Substitutions

In order to get the best performance of the players, they have to be changed properly, so good players play more than bad players, but ensure that their efficiency is not affected because they are tired. So to do that a system of substitutions have been implemented. For AI, it gives the starter players a high quantity of minutes. For user team, it depends on the distribution of minutes selected. By default, starters will play more, but it can be modified by the user prior the game, not during the game.

The next code shows how changes work for user team. For AI teams it is the same code, but without the minute distribution variable:

```
void cambiar() {
    for (int i = 0; i < tendenciasJ1.GetLength(0); i++) {
        if ((i == base1 || i == escolta1 || i == ala1 || i == alapivot1 || i == pivot1) && tendenciasJ1[i, energia] > Random.Range (15, 25)) {
            switch (i) {
            case (0):
                if (tendenciasJ1[i, energia] > (Random.Range (30, 50) * team1.devolverReparto()) && (quarter != 4 && timeSeconds < 400)) {
                    base1 = 5;
                }
                break;
            case (1):
                if (tendenciasJ1 [i, energia] > (Random.Range (30, 50) * team1.devolverReparto ()) && (quarter != 4 && timeSeconds < 400)) {
                    ala1 = 6;
                }
                break;
            case (2):
                if (tendenciasJ1 [i, energia] > (Random.Range (30, 50) * team1.devolverReparto ()) && (quarter != 4 && timeSeconds < 400)) {
                    escolta1 = 7;
                }
                break;
            case (3):
                if (tendenciasJ1 [i, energia] > (Random.Range (30, 50) * team1.devolverReparto ()) && (quarter != 4 && timeSeconds < 400)) {
                    alapivot1 = 8;
                }
                break;
            case (4):
                if (tendenciasJ1 [i, energia] > (Random.Range (30, 50) * team1.devolverReparto ()) && (quarter != 4 && timeSeconds < 400)) {
                    pivot1 = 9;
                }
                break;
            case (5):
                base1 = 0;
                break;
            case (6):
                ala1 = 1;
                break;
            case (7):
                escolta1 = 2;
                break;
            case (8):
                alapivot1 = 3;
                break;
            case (9):
                pivot1 = 4;
```

Code 27. Change players

Random values are useful to avoid mathematical changes depending on the energy and give the system more freedom to make changes.

3.6.12 Course and closure of the game

All the functionalities described above are taken into account when running every play, and the stats generated are stored in a matrix.

Once the game is finished, a function called *closeGame* saves all the stats in their corresponding team and player, sets the wins and loses, and waits until the player presses the *Final* button to go to the Season screen.

```
void closeGame() {
    for (int i = 0; i < 10; i++) { //Jugadoras
        for (int j = 0; j < 12; j++) { //Estadisticas
            team1.setStats (i, j, t1Stats [i, j]);
            team2.setStats (i, j, t2Stats [i, j]);
            mg.SaveStats ();
            mg.SaveSeason ();
        }
    }
    if (score1 > score2) {
        team1.setV ();
        team2.setL ();
    } else {
        team2.setV ();
        team1.setL ();
    }
    base1 = 0;
    base2 = 0;
    escolta1 = 1;
    escolta2 = 1;
    ala1 = 2;
    ala2 = 2;
    alapivot1 = 3;
    alapivot2 = 3;
    pivot1 = 4;
    pivot2 = 4;

    mg.balance [12, 0] = mg.seleccion;
    mg.SaveSeason ();
}
```

Code 28. Close game

3.7 Team choosing

This is the screen where the user can select the team he wants to manage. As the game has been build, *Team1* has the be the team he is managing, so the name of the different game objects and the team number changes as he chooses a team. The next code how this change is done. It happens when the user presses the button to start the game.

```
GameObject.Find ("Team" + (seleccion+1)).name = "Cambio";
GameObject.Find ("Team1").name = "Team" + (seleccion+1).ToString();
GameObject.Find ("Cambio").name = "Team1";
GameObject.Find ("Team1").GetComponent<Team> ().equipo = 0;
GameObject.Find ("Team" + (seleccion+1)).GetComponent<Team> ().equipo = seleccion;
break;
```

Code 29. Team name swap

3.8 Save and load manager

Data in this game is stored in binary files. The reasons to use this type of files are that it works properly, it can store some amount of numbers in arrays and they are easy to create, store and load.

In this game, four binary files are created and stored, and they are used for:

- Store players
- Store season data
- Store statistics
- Store improvements

The process for the creation and load of the binary files is the same for all of them. There are two functions per file, one for saving and one for loading. In the same script, there is also a serializable class (not monobehaviour) for each file which takes the data from the Manager.

- Class: It is serializable, so the data that it stores can be used by the save and load functions to create and load the binary files. This class has access to the manager to get the information (which is public), but the manager does not have access to it, the data is sent in the load function
- Save: The binary file is created, and the data taken by the corresponding class is serialized, and stored in the file. Finally, the file is closed.
- Load: Only if the file exists, the function opens it, extract the data deserializing it, and after the file is closed the data is returned. If the file does not exist, it creates an empty array of the corresponding size.

The next code is an example of how this has been implemented. This is the functions and class for storing the statistics data:

- Save and load

```
public static void SaveStats(Manager m) {

    BinaryFormatter bf = new BinaryFormatter ();
    FileStream stream = new FileStream (Application.persistentDataPath
        + "/stats.sav", FileMode.Create);

    StatsData data = new StatsData (m);

    bf.Serialize (stream, data);
    stream.Close ();

}

public static int[,] LoadStats() {
    if (File.Exists(Application.persistentDataPath+ "/stats.sav")) {

        BinaryFormatter bf = new BinaryFormatter();
        FileStream stream = new FileStream(Application.persistentDataPath
            + "/stats.sav", FileMode.Open);
        StatsData data = bf.Deserialize(stream) as StatsData;

        stream.Close();
        return data.stats;

    } else {
        Debug.LogError ("NOP");
        int[,] vacio = new int[10, 12];
        return vacio;
    }
}
```

Code 30. Save and load stats functions

- Stats data class

```
[Serializable]
public class StatsData {
    public int[,] stats;

    public StatsData(Manager m) {
        stats = new int[120, 12];

        stats = m.devolverStats ();
    }
}
```

Code 31. Serializable Stats data class

3.9 Manager

This is where everything is managed in the game. This script stores data from the next elements of the game:

- Players
    - Array of created players, with the teams assigned
    - Arrays of names and lastnames
- Season course
    - Week
    - Calendar (matchups per week)
    - Statistics
    - Improvements (obtained by training)
    - Leader of the league
    - Most valuable player
    - Team selected
- Teams: Every team is stored as a children of this object.

The reason for the existence of this manager is to keep all the important data in one script. All the data that will be saved and loaded, will pass through here, so this is one of the most important scripts of the game.

The process of accessing the Save and load manager and saving and loading data consists on calling the correspondent function passing the right parameters.

- Save players: A function calls the corresponding save function in the save and load manager. This function gives access to the players matrix and stores it in the correspondent binary file.
- Load players: The matrix returned by the load functions is stored in a temporary variable. After that, the players matrix is updated with the loaded one.

- Save Season: As save players, it only calls the correspondent function of the save and load manager. The matrix only stores the record of the team (wins and loses) and the team selected, there is no more information necessary since the rest can be calculated afterwards.
- Load Season: First, the matrix is loaded and stored in a temporary variable. After that, the record of each team is assigned, saving both wins and loses. The week can be calculated adding the wins and loses of any team. Then, the team selected is taken from the matrix, and it is updated. Finally, the stats of each player are loaded.

- Save Stats: Since each player of each team stores her own statistics, they have to be loaded in a new matrix. Teams have a function that creates a matrix of stats organized by player, the size of this matrix is 10x12 (ten players, 12 statistics). Then, this matrices are added to a new one of size 120x12 (12 teams) and passed to the save load manager to store it in a new file. The code for this operation is the following one:

```
public void SaveStats() {
    for (int i = 0; i < 12; i++) {
        Team t = GameObject.Find ("Team" + (i + 1)).GetComponent<Team> ();
        t.crearMatrizStats ();
        for (int j = 0; j < 10; j++) {
            for (int k = 0; k < 12; k++) {
                stats [j + (10 * t.equipo), k] = t.devolverStats () [j, k];
            }
        }
    }

    SaveLoadManager.SaveStats (this);
}
```

Code 32. Save stats

- Load stats: The returned stats matrix is stored in the manager stats matrix. The the teams will update its players stats.

- Save train: There is a call for the save train function in the save and load manager, which stores it as it has to do.
- Load train: The returned matrix is stored in the train matrix of the manager, and each team will update its players stats

3.10 Tasks

- Implementation of the game simulation system: The result of programming the games.
- Implementation of the game management system: The combination of team preferences and games, how they work together exchanging information
- Implementation of the entity management system: The integration of games, preferences, trainings and statistics in the season screen.
- Implementation of an AI able to manage games and entities: The system works as expected, and only the user decisions can affect in his own team, the rest work independently.

4. Artistic Design

The style of the games tries to be simple, but elegant and visually attractive, with different colours and contrasts between the backgrounds and the elements of the game, trying to implicate the user in the game and the team he has selected.

4.1 Design of the Teams' logos

Team logos are based on real images researched on the Internet. The style pretends to be minimal, with a clear base color for the team and easily recognizable.

Process:

- Almost all the teams' names are related with the city they were assigned, but there are three exceptions (Valencia Queens, Barcelona Patriots and Castellon Cactus), which were chosen due to personal reasons.
  The first step of the creation of each logo was to look for visual references according to the name.
- Once one good reference was found, it was modified to get the minimal style searched and with the colors of the team
- Finally, in some cases a geometrical form was added to make the logos more colourful and recognizable

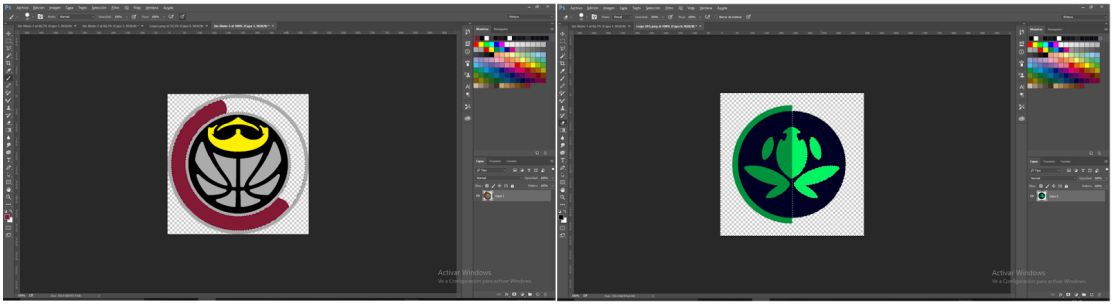

Figure 37. Logos development

4.2 Design of the buttons

The buttons were designed to be visually attractive, but also to indicate clearly what they do or which screen are they going load. The style is simple and works well with the backgrounds

4.3 Design of the backgrounds

The backgrounds of the game are divided in three: the standard background, and the distribution backgrounds, and the theme backgrounds.

- Standard background: It is the standard image that is at the background of some screens (Start, Team selection, Season and End). Its design is dark, with black, grey, and blue tones to avoid draw attention.

  It was design with the Clouds generator in *Adobe Photoshop CC 2016*

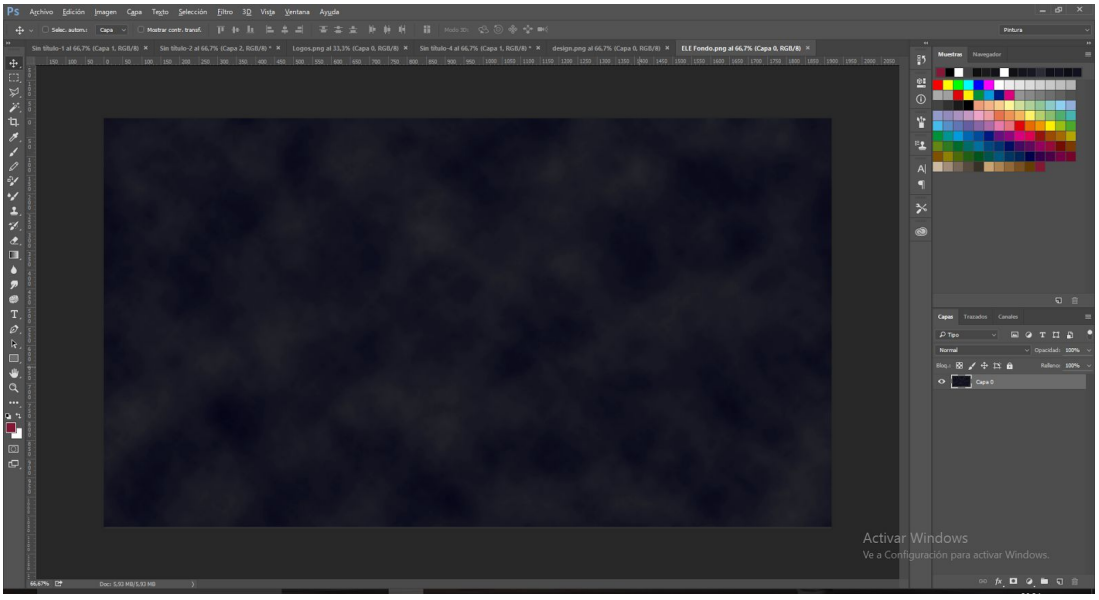

Figure 38. Background development

- Distribution backgrounds: Used in the Game screen and Stats screen, the objective of those designs is to allow a correct display of information, where everything is clear and has its own space, where it can be easily found and understood

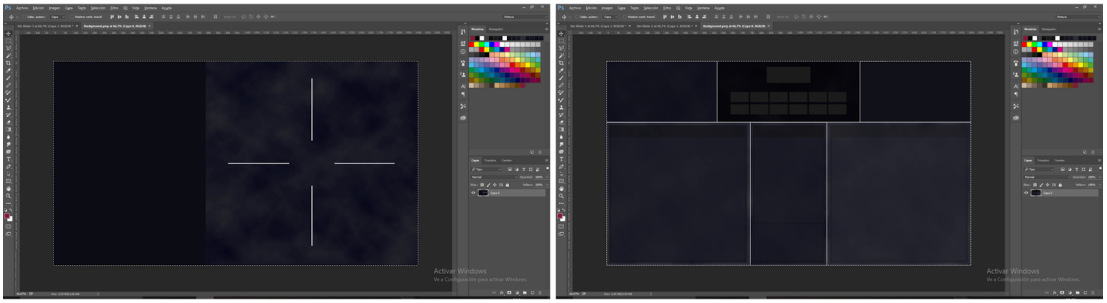

Figure 39. Stats and games background development

- Theme backgrounds: Used in the Train screen and the Preference screen

The first one tries to involve the player in the ambient of the court, making visible the players playing in their own court. With one image found on the internet, its colors were modified with every one of the teams' colours.

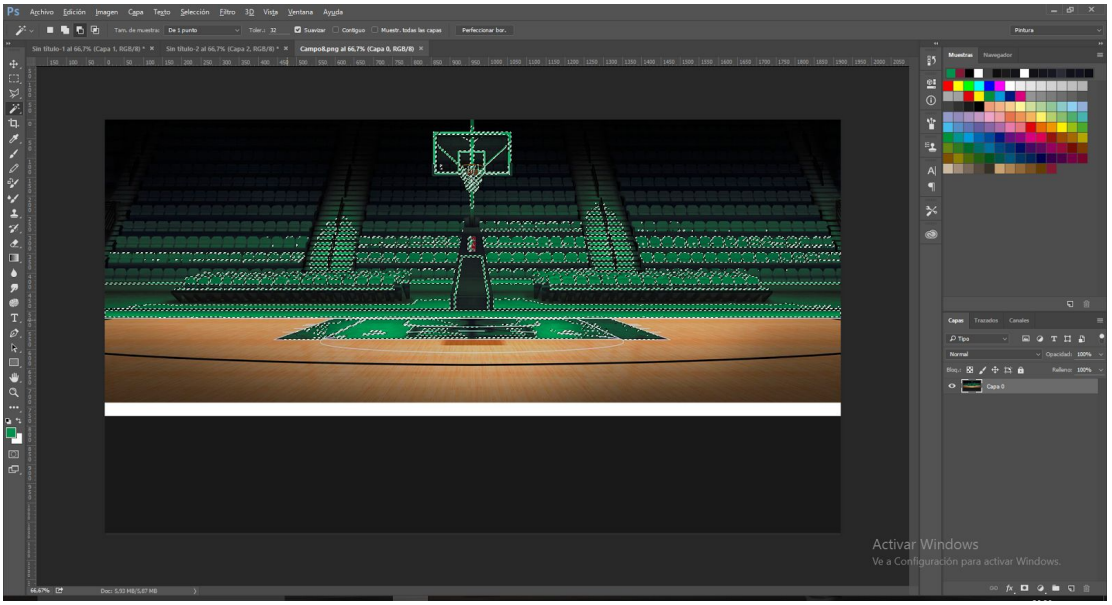

Figure 40. Training background development

The second ones try to emulate a basketball coach board. As the train background, the goal of this background implicate the user in the game and the situation of thinking the strategies. The logo on the center of the court is added in unity via scripting.

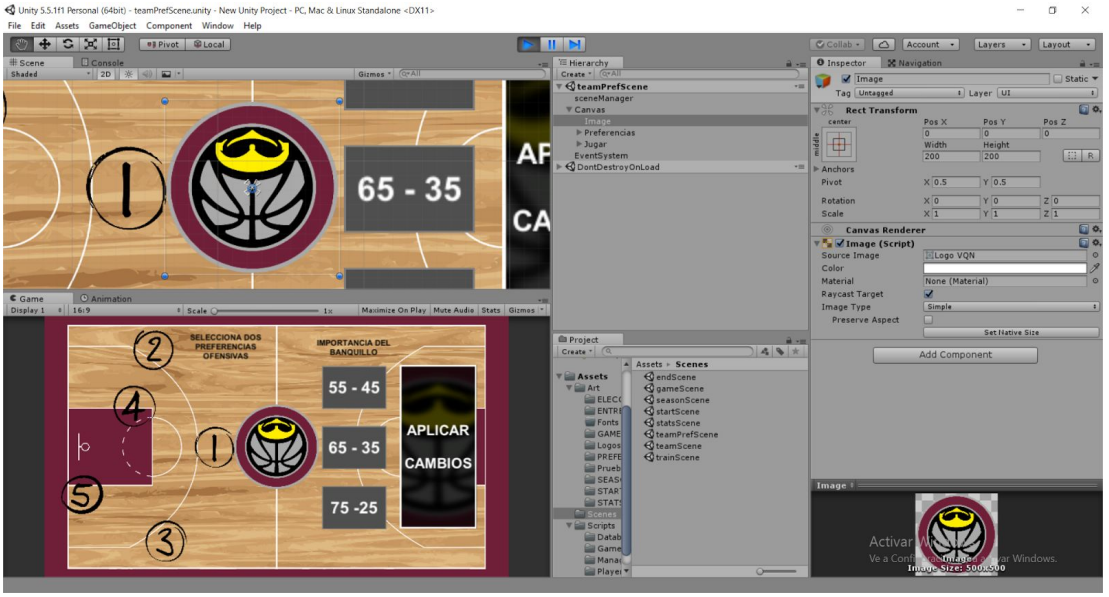

Figure 41. Unity distribution 1

4.4 Distribution on the different screens

With all the elements designed and finished, the last step of the art design was to include every single one in unity and the game.

- Standard backgrounds were simply added to the canvas, and the buttons in those scene where properly distributed.
- Distribution backgrounds were placed in the canvas too, and the next step was to put every data objects in their space and adapt them to look well.
- Theme backgrounds are changed via scripting according to the team of the user. The different buttons were also distributed in the design style

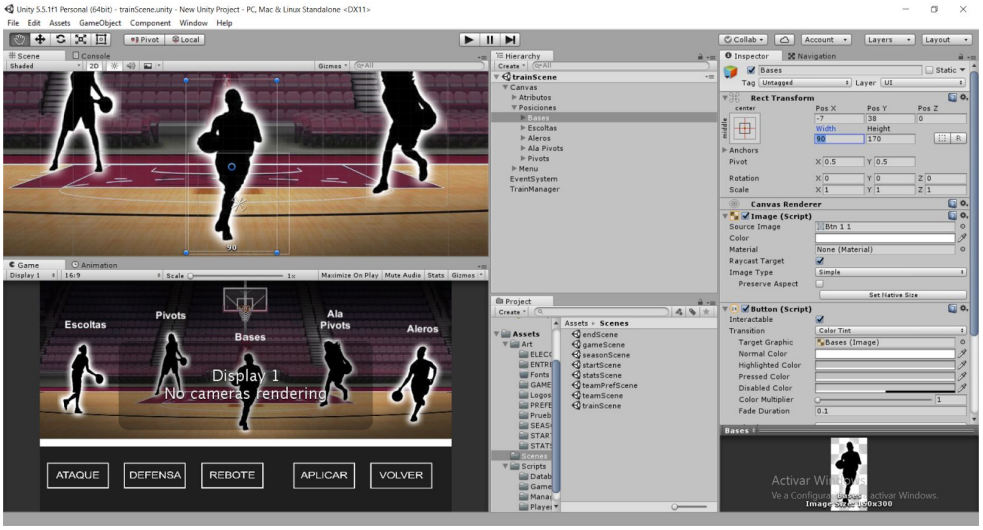

Figure 42. Unity distribution 2

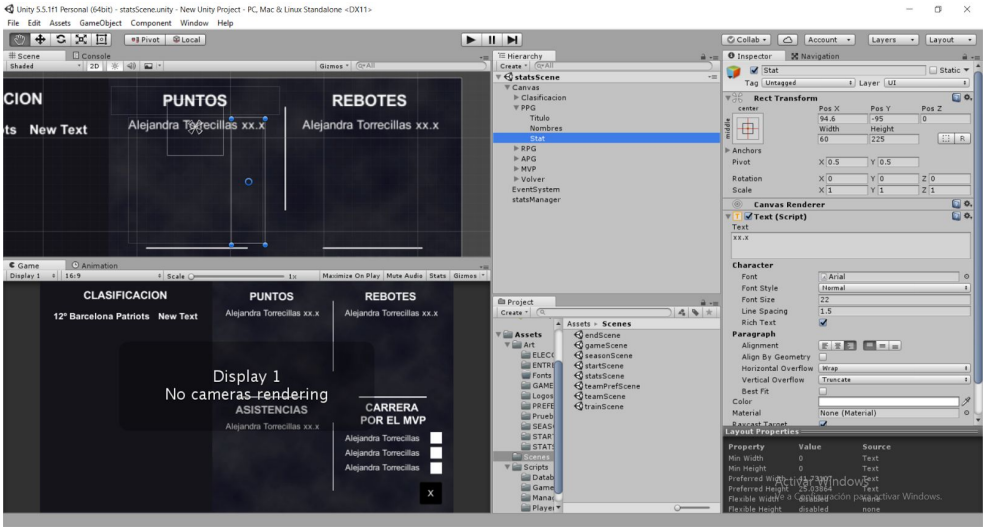

Figure 43. Unity distribution 3

5. Documentation

The last task of the project was to write all the necessary documentation and prepare the defence of the project.

5.1 Game Design Document

It has been redacted as it was explained in the subject VJ1222 Conceptual Design of Video Games, following all the steps and completing every necessary aspect.

5.2 Final Report of the project

It has been written following the guide provided for the subject VJ1240 Final Degree Project, modifying every necessary aspect to include all the work done during the project, and all the important information for its understanding.

5.3 Presentation of the project

The defence of the project will be done using a presentation made for it and a little video showing the game and its gameplay.

Chapter 5: Results & Testing

1. Results

The results expected at the beginning of the project were:

1. Game simulation: generate a realistic results and statistics taking into account players skills.
2. Game management: User decisions have a real impact in the final result of the game simulator.
3. Team management: It is expected to have a realistic behavior within its limitations.
4. Make the systems work on an Android device

1. As it was expected, the game simulator generates realistic results and statistics. As its behavior has been explained in the chapter above, it uses players skills to take decisions and the probabilities are modified and adapted to get better performance.
   As a result of that, this part of the project works as it should do. The best team or the team with better players is the team who has more chances of winning, but as in real life, it does not always happen. Also, the average statistics at the end of the season are so close to the WNBA and other female leagues real numbers.

Figure 44. Gameplay 2

Figure 45. Gameplay 3

2. The user has three ways to have an influence in the simulation of the matchups.

    a. Training: As it has been explained, the user can make three workouts during each week between games, so the players of his team improve their attributes. If the players are better, the chances of winning are higher. So there is a direct influence of the user in the game simulation and his decisions affect the probabilities of winning.
    b. Preferences: Planning the strategies for the game and making the right choices can influence also the final results of the games. Selecting the right shooters, or the right minutes distribution according to the performance of the players makes a real impact on the chances of winning.
    c. Timeouts: In games, the user can call a timeout to change a bad tendency in the game. Doing that at the right time can avoid the team to get in a bad run that overrides the possibilities of winning.

    So, the user really has an influence in the game, it is not only a simulator that cannot be altered. The decisions made during the game have a real impact, so this objective has been successfully completed.

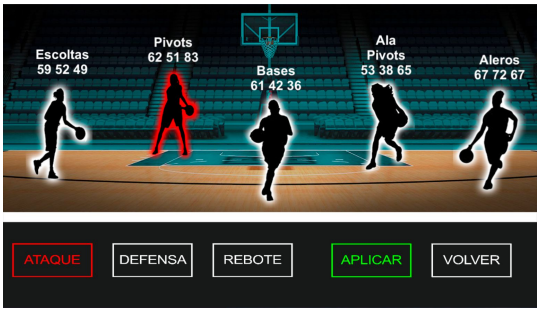

Figure 46. Gameplay 4

Figure 47. Gameplay 5

3. This game is very limited in team management. All the extra-sports aspects were discarded because they were not entertaining and made the game so much difficult to play and less attractive.

4. The game works as expected in Android devices, having the optimal display on 16:9 screens.

2. Testing

The results of the matchups, statistics, and influence of the player actions has been test continuously during the project in order to make modifications in the process of building the project and ensure that it has not to be done afterwards.

The comparison of the results with WNBA and more female competitions confirms that the game is a great simulator and team manager and has accomplished the expected behaviors.

Also, the game has been tried in real people to see if they liked the aesthetic, work, and results of the game. All the opinions were positive about the game:

- The aesthetic of the game gets good results, it is colourful, easy to understand, and involves the player inside the game
- The performance of the game also works properly. The users that tested the game usually tried to play more than one season to win, and constantly remember to train and select the correct preferences. This behavior confirms that the game hooks people. Even they got enthusiastic when they won titles or one of their player got the Most Valuable Player.
  Of course, the game has a lot of limitations, but it works as expected and generates interest within its limitations.

Chapter 6: Deviations from the project

1. Changes from the original idea

1.1 Extra-sports aspects

The decision to discard this part of the project was taken because it is not entertaining. Usually people want to win, manage the sport aspects of the team and tend to forget or directly avoid the financial and marketing parts. The games of this type that has more success don't deepen on economical aspects and are focused on give the user a great sport experience.
So, a decision was made in order to avoid boring parts in the game and give more time to develop a great simulator, which has been the final result

1.2 Database

At first the idea was to build a database with names and lastnames and generate the players database taking information from there and generating skills via scripting. But, as the project was being developed, it was found out that it was not necessary. Two arrays for the names and one matrix for all the players was enough to store every necessary data.
This system also was easier and better for loading and saving games, so finally the decision was to do everything (player storage, saving and loading) via scripting

1.3 Unity3D instead of Android

On one hand, Android Studio is a really powerful program for application developers, but it can be really messy and complicated when handling with graphics. Also, it requires to have some knowledge of XML language and it is limited to the Android system.
On the other hand, Unity3D allows developers to generate applications of the games developed just downloading the right SDK and building the *.apk* in an easy way. Also, it can generate an Xcode project to generate the same application for IOS system. Also is a more powerful tool to handle both 2D and 3D graphics, and the functionalities of the project can also be develop with this program.
So, the decision was to change the basic program of the project, since the results that can be obtained are the same, but it allows also to build the game for iPhone and iPad.

2. Changes in planification

2.1 Planification after the decided changes:

Task	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11
1. Technical Proposal	10										
2. Concept Design		20									
3.1 Players			10	5							
3.2 Teams			5	5							
3.3 Training					10	10	8				2
3.4 Preferences							10	8	5		2
3.5 Stats Scene									10	8	2
3.6 Games				20	20	20		10	8		2
3.7 Team choosing							5				
3.8 Save and Load							5	10		8	2
4. Artistic Design		10	10						10	10	
5. Documentation										10	10

Table 1. Original planification

- In the original planification, games had not the same importance as it had in the next planification.
- Database management and Android integration disappear from the planification since it had not to be done. Instead, the players and teams creation appears and also the save and load task.
- Since the system could work perfectly without the user, and all the teams are managed correctly, AI behavior is included in the work of the game.

Once the project started the difficulties and problems also showed up. Game development took more time that estimated to work properly, but most task were made at the same time as the application was getting build and the games needed some data from other parts of the project, such as preferences.
Players and team creation was as expected, creating the players and assigning them to teams so the league was balanced but with some differences.
Save and load task begun earlier, because if it would have given problems, it could be late to solve them and this functionality is key to the game since seasons are too long to be played in a little time. Fortunately, it worked as expected and it did not give too much trouble.

The final development of the game ended like this:

Task	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11
1. Technical Proposal	10										
2. Concept Design		20									
3.1 Players			10	5							
3.2 Teams			5	5							
3.3 Training					10	10		10			
3.4 Preferences							10	5	5		
3.5 Stats Scene									10	8	2
3.6 Games			10	10	20	20		15	8		2
3.7 Team choosing							5				
3.8 Save and Load				10	5	5	5				
4. Artistic Design		10	10						10	10	
5. Documentation										5	15

Table 2. Final planification

The final game does not have sounds since all the development took all the time of the project and they are not necessary in the game.

Chapter 7: Conclusions

7.1 Summary of the project

Everything in this chapter are personal opinions and valuations, so first person will be used along all the paragraphs.

In first place, I want to say that from the beginning of the project I was really enthusiastic with it, because the game idea was something that I really like. I have always liked basketball, but also game and statistics simulators. Also, as a basketball coach, i find attractive to have an application to manage my own team with my own rules, so the project got me pretty involved since the first moment.

During the project I have learned a lot. In my years as a student of this degree I had never faced a project of this dimensions, so I had to work hard, search for a lot of information and guides to achieve every objective planned in the first place.
I enjoyed making researches about basketball statistics, and combined with my personal experienced during the year I had a lot of good data to work on the project and generate a realistic simulator and statistics generator. Also the work done to build the team manager, with trainings and preferences have been fun, but also hard and sometimes humdrum, because every number had to be checked to get the best results.
Also I am proud of the players generator and teams creator. Those thing combined with the graphics design gives the game twelve different entities with their own personality and style.

With the project, I have learned a lot in programming, application development, art design and project management. Without this knowledge acquired, the project results would had barely been accomplished. For example, I had never had to save and load games, so I had to learn how to do it, and the result is pretty convincing.

So, I am glad to say that all the objectives have been successfully achieved. The game finished is fun to play and visually attractive.

One of the goals of the project, and the most important for me is to announce and give visibility to female sport. Nowadays male sports have all the attention, and women have no chance to play the sports they love and do it for a living. While men have this chance, and get highly paid for it, women are discriminated.
This game is one idea to give them visibility. If people play the game, may get involved in the world of female basketball and generate interest, what starts giving some fans, and that is one step to end this injustice.

References

[1] Android Studio
https://developer.android.com/studio/intro/index.html?hl=es-419

[2] Adobe Photoshop CC
http://www.photoshop.com/

[3] Adobe Audition CC
http://www.adobe.com/es/products/audition.html

[4] Unity 3D
https://unity3d.com/es

https://docs.unity3d.com/Manual/index.html?_ga=2.155374651.1031260490.1498911628-987678300.1493200674

[5] MonoDevelop
http://www.monodevelop.com

[6] Statista
https://www.statista.com/statistics/189592/breakdown-of-us-video-game-sales-2009-by-genre/

[7] ESPN Fantasy Games
http://www.espn.com/fantasy/

http://www.espn.com/fantasy/basketball/

[8] Comunio
http://www.comunio.es/

[9] NBA General Manager
https://www.nbageneralmanagerthegame.com/

[10] Hoops Rivals
http://hoopsrivalsgame.com/

[11] WNBA
http://www.wnba.com/
http://www.wnba.com/stats/