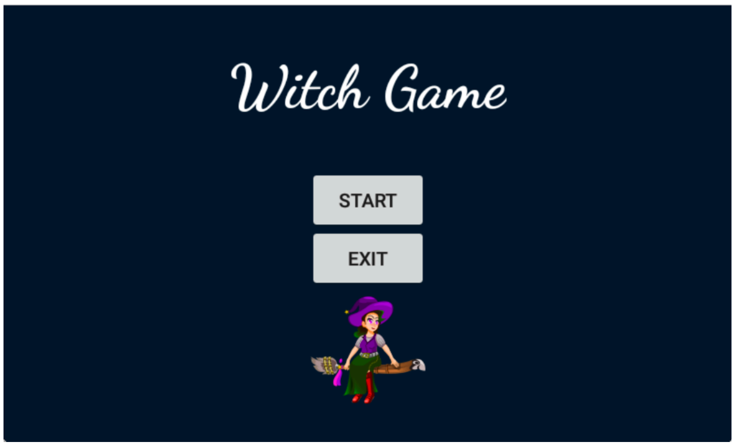
Major Coursework 2: Android Game

Abstract

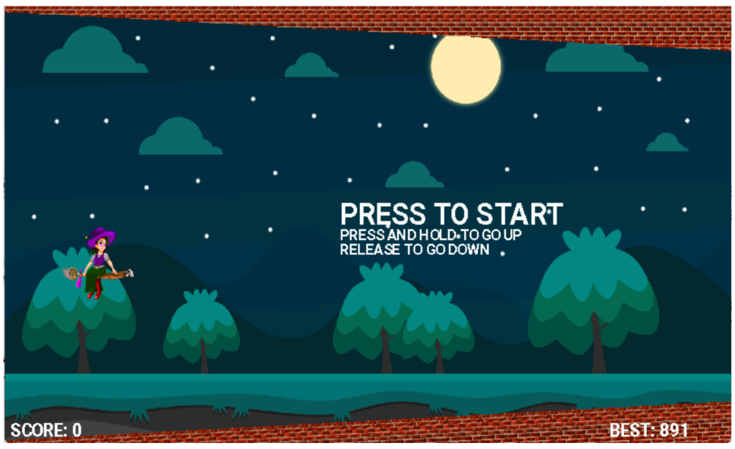
In this coursework, it was required to implement an Android Game and produce an APK from it, that could be installed on an Android phone. The application that was created for this project was developed in Android Studio using Object Oriented Programming (OOP) design. The application consists of a menu screen that gives the options to either start the game or exit the application and a screen in which the user is able to play the game. The game includes a character that is controlled by the user by touching the screen, more than one opponents that play the game autonomously, walls at the top and bottom of the screen that limit the space of the player, a current score that starts counting the moment the user starts to play the game, a best score that keeps record of the user's highest score and three levels of difficulty for each of which the number and speed of the opponents increase. The goal of the game is for the player to avoid touching the walls as well as the opponents in order to pass all three levels. This report aims to describe the design of the game and to justify the code that was developed.

Introduction and Showcase

The theme of the game is fairytale inspired, therefore, the background is set as a forrest, the player's character is a witch seated broom and the opponents are bats.



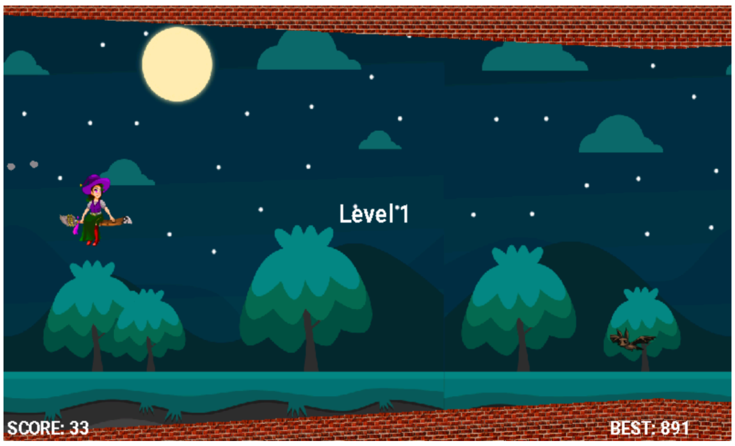
When the application launches, the menu screen appears first. The menu screen presents the title of the game at the top, an image of the main character at the bottom and two buttons in between. By selecting the START button the user continues to the game, while pressing the EXIT button simply exits the application.



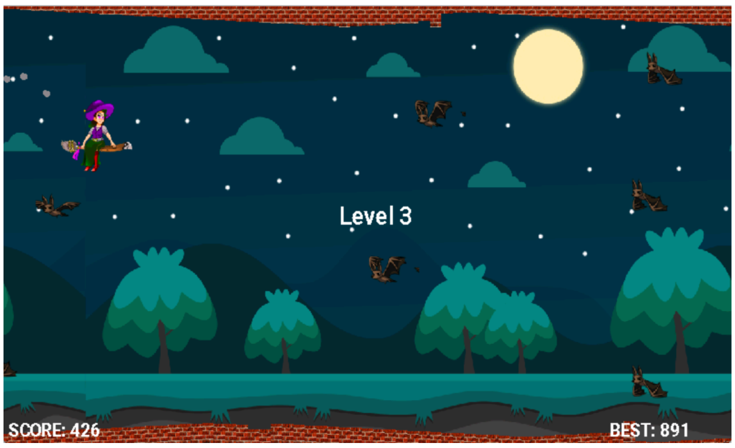
After pressing the START button, the game screen is presented. The background is set to the selected image, the walls appear at the top and bottom of the screen and player's character is set in the middle of the left side. The instructions of the game are displayed to the screen along with the score counter at the bottom left which is set to zero and the best score achieved at the bottom right. The game does not start unless the user touches the screen.

Instructions:

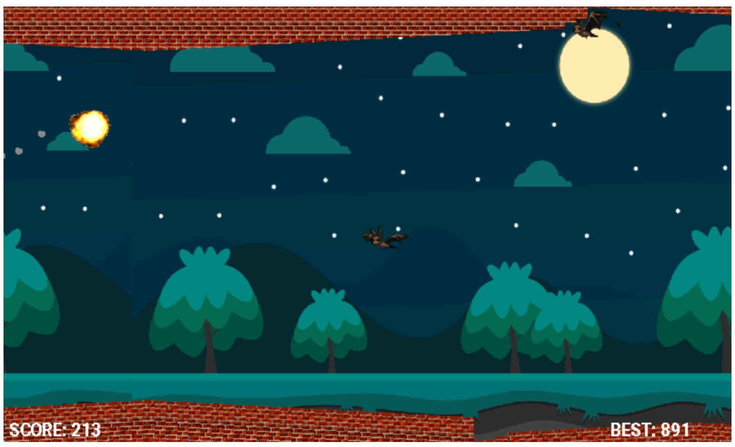
The functionality of the game is simple, and is explained to the user before the game begins. The character is fully controlled by the user and is set to go up while they press on the screen or go down the moment they release it.



When the game begins, the player's character starts moving to the right followed by smoke puffs, while bats start appearing from the right side of the screen moving at the opposite direction. The score starts counting immediately and the text "Level 1" appears briefly on the screen. The background moves along with the main character as well as the two walls which also change dimensions randomly as the game progresses.



200 points after the game was initiated, the player's character enters the second level with a corresponding message and after 400 points from the start, a third message appears on the screen informing the user that Level 3 has been reached. As demonstrated in the picture provided, the number of opponents shown on the screen on the third level is remarkably higher compared to the one on the first level.

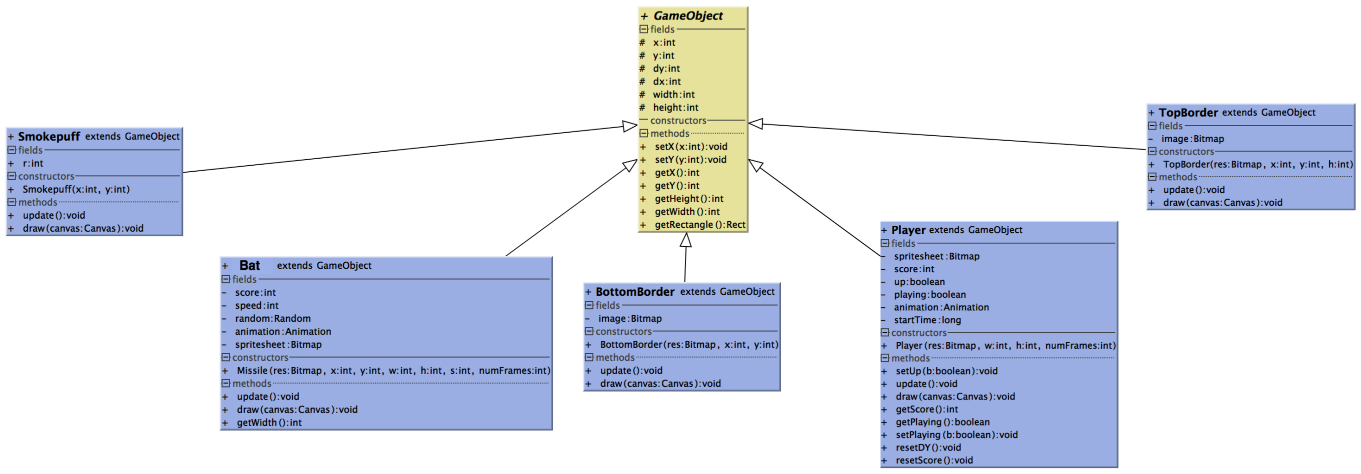


In the event of failure to avoid the opponents or the walls, the background movement stops and the character disappears, turning into an explosion. Shortly after the explosion is over, the game ends and the menu screen appears, giving the options to start again or exit the application.

OOP Design

Overall, the design used for the development of this application was strongly object oriented, utilising concepts such as inheritance, encapsulation and abstraction. Therefore, the abstract class GameObject was created, from which five other classes were able to inherit methods and variables. The GameObject class provided the basic variables x, y, dx, dy, width, height which were used a number of times in the classes that were created for the elements that move on the screen such as the Player, the Bats, the Smoke puffs, the Top and Bottom Border. Since the variables were declared as protected in the abstract class, other classes were able to inherit them with the assistance of encapsulation through the use of getters and setters. Encapsulation made it possible to hide private data variables and gave them the ability to be accessed through specific functions.

The following diagram presents the abstract class and the classes that inherit from it, along with their variables and methods.



The first class that was created was the Game class which sets the screen to full screen, turns the title off and sets the content view to a new instance of GamePanel. The GamePanel class extends SurfaceView and is where most of the game's functionalities are. For example, it includes overridden methods of surfaceChanged, surfaceDestroyed, and surfaceCreated, as well as onTouchEvent. Following, a thread was needed so the MainThread class was created to include the game loop. The Animation class contains methods that affect the images that appear on the screen such as going through the frames of an image once or continuously. Additionally, the Background class sets the background to move, as well as making it scroll so that it appears to be continuous by using an update() and a draw() method and the Explosion class handles the explosions.

To summarise, having used some of the fundamental OOP concepts has contributed greatly to the development of this program. Through encapsulation, the variables of a class were hidden from other classes, and could be accessed only through the methods of the encapsulated class. This was achieved by declaring the variable of the class as private and providing public setter and getter methods to modify and view the variables values. Furthermore, through abstraction it was feasible to hide the implementation details from the user and only provide them with their functionality. By creating an abstract class, the classes that inherit from it have to provide implementations for all its abstract methods.

Special Requirements:

Due to the fact that this application had to be designed for an Android mobile phone, there were some special requirements such as making the application adjust to the size of every screen. Each mobile screen has different dimensions, therefore the size of the application needs to be scaleable. This was achieved by creating the scale factors in the GamePanel class as *final float scaleFactorX = getWidth() / (WIDTH * 1.f)* and *final float scaleFactorY = getHeight() / (HEIGHT * 1.f)* for the x and y dimensions respectively. These methods give the width and height of the entire screen depending on the device, so the size of the application is not the size of the game but the entire surface view over the width and height of the game.

Memory Usage and Speed Improvements

One of the steps that were taken in order to improve the speed and memory usage of the game and therefore, the entire application, was using methods that delete images when they go off the screen. The randomly generated bats, top and bottom border were stored in array lists as well as the smoke puffs. Since they move to one direction only, they eventually go off the screen and new ones are generated, increasing the number of elements in each array list. The methods that were used, removed the images after they they left the screen, therefore the number of elements in each array list constantly remains low, having as a result the application to run faster. This improved the speed of the application, since large numbers in array lists usually cause applications to become slow. In addition, this step also improved the memory usage of the application, since it will not have to store all the elements that were ever generated in the array lists.

Conclusion

All in all, the main purpose of this project was to create an application that would implement a game for an Android mobile phone developed in Android Studio, something that was successfully achieved. All of the specifications were met including the menu screen with the option to start individual games, displaying the current score while playing, providing three levels of difficulty and having a Character controlled by touch on the screen and opponents with random movement as well as using object-orientation. To conclude, Object-Oriented design aided greatly in the design of this application, since the OOP concepts such as abstraction, inheritance and encapsulation played a major role in the development of the code.