Jailbreaker: An Endless Runner Mobile Game Application

Cabili, Sherwin A.
Dept. of Computer Science,
Mindanao State University-Iligan Institute of Technology
Iligan City
vashX30a@gmail.com

Moso, Karen L.
Dept. of Computer Science
Mindanao State University-Iligan Institute of Technology
Iligan City
kmoso.iit@gmail.com

***Abstract*— JailBreaker is an endless runner mobile game application developed using Phonegap and Cordova. The objective of the game is to help the prisoner escape from the jail by avoiding the policemen. The application makes use of the accelerometer input sensors for moving the game character. It also features good music and sound effects suited for the game.**

***Keywords*— Mobile Game Application, Mobile Game Development, Android, Phonegap, Cordova, Endless runner game**

1. Introduction

As smartphones become a necessity nowadays, development for mobile application has also increased its popularity especially for software developers and programmers. In addition, it is easier to develop such applications because of various mobile development tools and frameworks/libraries that are made available on the web. One of the popular tools for developing cross platform mobile applications are Phonegap and Cordova that provide plugins to use and access hardware components in smartphones.

A popular category of mobile applications are games. One of the most successful mobile games for smartphones are the endless runner games that provide an exciting and fun gaming experience. They are popular also because these are games that can be fun to play at any age and does not require you to devote a lot of time into it so you can play it while you have only a little free time.

Jailbreaker is a simple endless runner game just like the popular game Temple Run, Subway Surfers and One Epic Knight. But instead of having 3D graphics, we use 2D graphics in order for the application to be easily implemented across multiple platforms and devices by using Phonegap and Cordova. The player must try to dodge the police men by moving left or right as long as he/she can to get the highest score. The game contains the prisoner character, the policemen, and the environment.

2. Related Work

Endless runner games already existed even before the age of smartphones. These games were far more simpler and were usually implemented as flash games that are playable using a computer's internet browser. An example of this is the famous Helicopter Game.

David McCandless created Helicopter Game in 2000. At the time, he was running the website seethru.co.uk, which was a tie-in to a BBC drama called Attachments; the show was about an Internet startup called seethru[1].

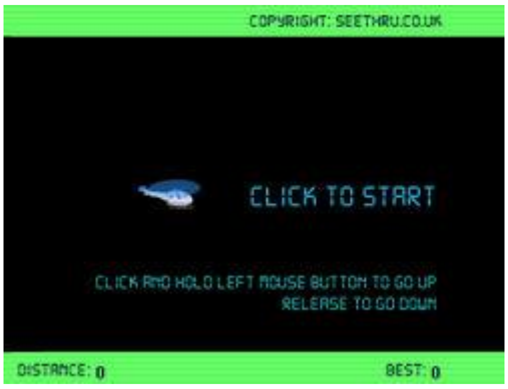


Figure 1. Helicopter Game start screen

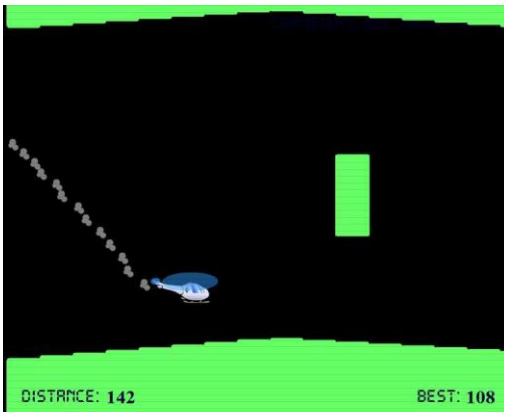


Figure 2. Helicopter Game gameplay screen shot

Another game that shares this concept is the famous smartphone mobile game Temple Run. Temple Run is a 2011 endless running video game developed and published by the Raleigh-based Imangi Studios. It is produced, designed and programmed by husband and wife team Keith Shepherd and Natalia Luckyanova, and with art by Kiril

Tchangov. The game was initially released for iOS devices, and later ported to Android system and Windows Phone 8.

A sequel to the original game was released on January 17, 2013 for iOS, and on January 24 for Android. As of June 2014, Temple Run and its sequel have been downloaded over 1 billion times[2].

Figure 3. Temple Run 2 main screen(left) and gameplay(right)

3. DESIGN AND IMPLEMENTATION

3.1 IDENTIFYING GAME FEATURES

Identifying the features to be implemented in the game is important. The features for this game are as follows.

3.1.1 Play Game

This will direct the player to the game environment of Jailbreak and allows him/her to play the game.

3.1.2. Pause Game

This will pause the game the player is currently playing.

3.1.3. Resume Game

This will allow the player to resume the paused game.

3.1.4. Change Options

This feature will redirect the player to the settings page where he/she can change the settings. The options page includes the music, sound effects and tilt bias settings.

3.1.5. Help

This will let player read the instructions for the game.

3.1.6. Back to Main Menu

This will let the player return to the main menu of the game.

3.1.7. Quit Game

This will allow the player to quit the currently played game.

3.1.8. Exit

This will allow the player to quit the currently played game.

3.1.9. Score Display

The score is displayed in the game environment as well as the highest score attained by a player of the game.

3.1.10. Character Movement

To control the game character, accelerometer input controls are used. The player tilts his/her phone to move the character left or right.

3.1.11. Restart Game

This will allow the player to start a new game after a game has ended.

3.2. Methods of Implementation

3.2.1. Sprint Backlog

Backlog Items	Priority	Iteration Path (Sprint Number)	Estimated Time(Hours)
Learn about tools	High	1	30
Setup tools	High	2	30
Create Graphics for game	High	3	60
Open App with homepage	High	4	10
Access help page	Low	4	10
Start game	High	5	30
Resume paused game	Medium	5	10
Pause game	Medium	5	10
Quit and restart game	Medium	5	10
Integrate Hardware Sensors	High	6	50
Change the settings of the game	Medium	7	10
Game display (integrating the graphics for game environment, character etc), Music and Sound effects	High	7	40

Table 1. Jailbreaker Product Backlog

Table 1 shows the product backlog planning for the development of our Jailbreaker mobile game application. For every sprint, there is an estimated number of hours allotted to finish it.

3.2.2. Sprint 1 (Learn about tools)

For this sprint, we read about how to use the tools that we would be using such as Phonegap and Cordova. We studied the Phonegap API[3].

3.2.3. Sprint 2 (Setup Tools)

For this sprint, we setup all the tools that we need. To install Phonegap and Cordova, we first needed to have NodeJS installed. After installing NodeJS, we used the command "npm install –g phonegap" to install Phonegap, Cordova is already included when installing Phonegap. We also installed Genymotion for emulating an Android device using a personal computer.

3.2.4. Sprint 3 (Create Game Graphics)

The game graphics are created in this sprint such as the game character, the background for the game environment, the police character and other graphics needed.

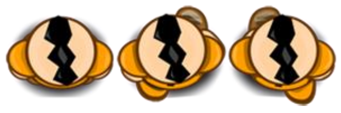


Figure 4. Game Character

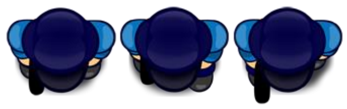


Figure 5. Police Character

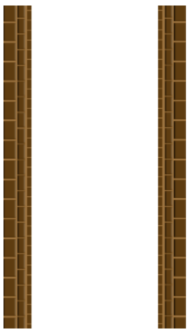


Figure 6. Walls

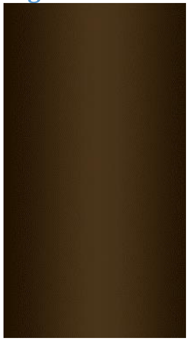


Figure 7. Floor

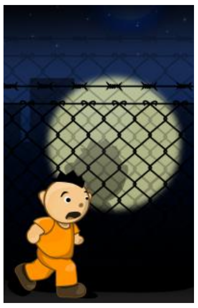


Figure 8. Homepage Background

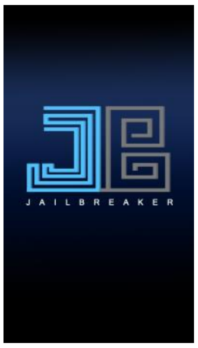


Figure 9. Jailbreaker Splashscreen

3.2.5. Sprint 4 (Create Homepage and Help Page)

In this sprint, the homepage which includes the main menu and the help page is created. In the main menu, play game, options, help, and exit buttons are displayed.

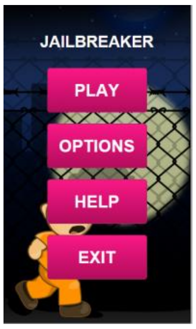
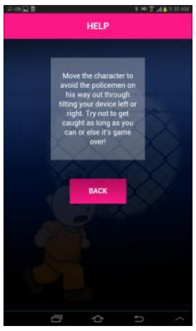


Figure 10. Jailbreaker Homepage



Move the character to avoid the policemen on his way out through tilting your device left or right. Try not to get caught as long as you can or else it's game over!

Figure 11. Jailbreaker Help Page

3.2.6. Sprint 5 (Start/Pause/Resume/Quit/Restart Game)

The functionality for the play button is added to start the game. Pause, Resume, and Quit features are also added. The game environment with the game character is loaded when starting the game. Quit button quits the game and redirects to the main menu. Pressing the Restart button after a game has ended will start a new game.

3.2.7. Sprint 6 (Integrate Hardware Sensor)

This sprint focuses in accessing the accelerometer input sensors to move the character left or right. We used the Phonegap plugin for the accelerometer to gain access to the sensor. The police character is also displayed and spawned. Prisoner and police collision is also implemented.

```
function startAccel(){
var options = { frequency: 300 };
watchMove=navigator.accelerometer.watchAcceleration(on
Success, onError, options);
}
```

We used this code to access the accelerometer after adding the plugin. We also set the frequency in which it will fetch the reading from the sensor to 300ms.

```
player.draw = function() {
    player.pos = player.x;
    if(retAcceleration.x > 0 ){
      player.x -= player.speed;
    }else if(retAcceleration.x < 0){
      player.x += player.speed;
    }
    if(player.x < 70 || player.x > 310){
      player.x = player.pos;
    }
    player.anim.draw(player.x, player.y);
  };
```

This is the code we used to draw the game character and use the data from the accelerometer to determine its position.

```
function spawnPoliceSprites() {
  score++;
  var valX = rand(80,305);
  police.x = valX;
  if(score/20 > 50 && score/20 % 20 == 0){
    if(psize >= 5){
      psize = 3;
    }else{
      psize+=1;
    }
  }
  police.y = rand(-80,-500);
  if(score > 10 && Math.random() < 0.96 && pmen.length <
psize && (pmen.length ? H - pmen[pmen.length-1].y >=
score/20 && police.speed > 3.5 && pmen[pmen.length-1].y
> pdistance: true)){
    pmen.push(new PoliceMen(police));
  }
}
```

This code is for the spawning of Police sprites.

```
if((distance.x > -90 && distance.x < 0) | (distance.x > 0&&
distance.x < 95)){
      if(distance.y > -80 && distance.y < 58 && player.y +
player.height > pmen[i].y){
        console.log(player.y + " " + pmen[i].y);
        gameOver();
      }
}
```

This code is for the collision of the game character and police. If it collides, it will cause a Game Over by calling the gameOver() function.

3.2.8. Sprint 7 (Settings, Music, Sound Effects, Scores)

Sound effects and music are added to the game. Also, the score and the highest score is displayed in the game environment. The settings in the options page can be changed and saved. Bugs found are also fixed.

4. Results and Discussion

It took more than two months to finish the Jailbreaker mobile game application. We started the project on January 5, 2015 and finished it on March 8, 2015. During that time, we had a total of 7 sprints.

For every sprint, a burn down chart is created in order to monitor the progress of our game development. We used a line graph to represent the ideal (blue line) and actual remaining working hours (orange line) with respect to the projected delivery dates.

Below are the burndown charts for all the sprints.

4.1. Sprint 1

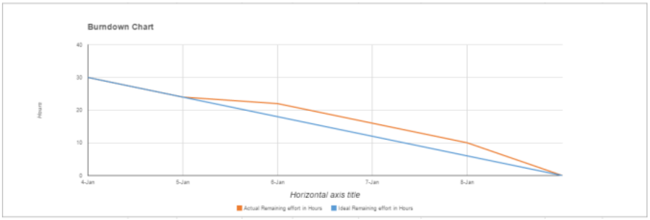


Figure 12. Sprint 1 Burndown Chart

During this sprint, we started on track with the ideal hours but were a bit behind in between but we managed to finish studying about the tools in time.

4.2. Sprint 2

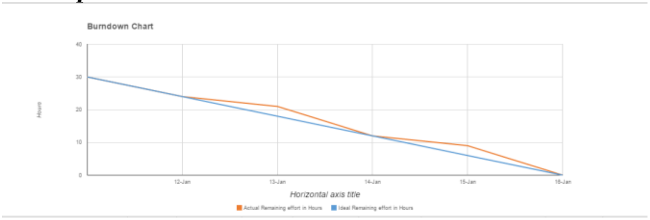


Figure 13. Sprint 2 Burndown Chart

In this graph, we were able to stay on track with our ideal work hours that was needed and finished it on time.

4.3. Sprint 3

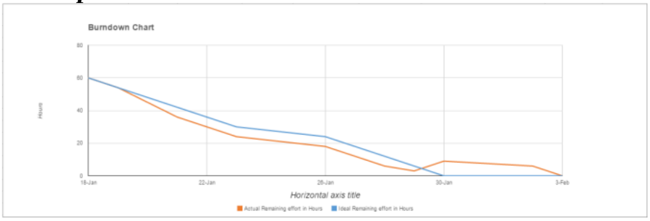


Figure 14. Sprint 3 Burndown Chart

In this graph, we were ahead in the beginning but we extended quite a bit because we decided to make changes on the designs and it took us a few hours of work. We extended for 3 days in this sprint.

4.4. *Sprint 4*

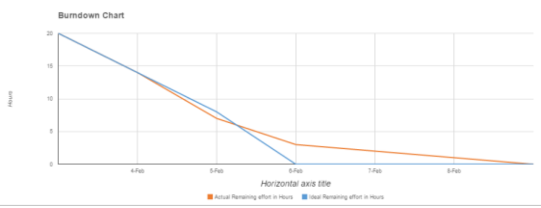


Figure 2. Sprint 4 Burndown Chart

In this graph, we were delayed by 3 days of the ideal deadline for this sprint. 2 days were weekends and we don't work weekends so that's why it was delayed by 3 days.

4.5. *Sprint 5*

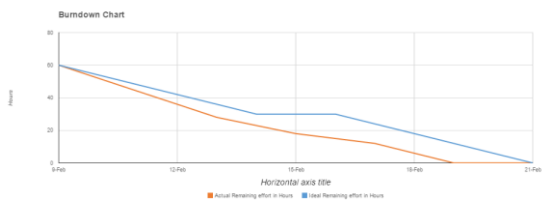


Figure 16. Sprint 5 Burndown Chart

In this graph, we were managed to finish it ahead of time by 3 days.

4.6. *Sprint 6*

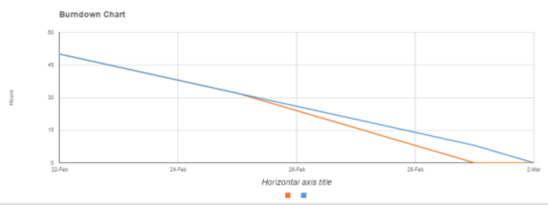


Figure 17. Sprint 6 Burndown Chart

In this graph, we were finished it ahead of schedule by 2 days.

4.7. *Sprint 7*

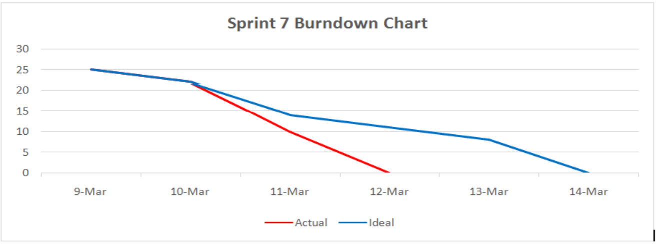


Figure 18. Sprint 7 Burndown Chart

This is our final sprint and we managed to finish it ahead of time by 5 days.

5. References:

[1] http://mashable.com/2014/02/09/flappy-bird-helicopter-game

[2] http://en.wikipedia.org/wiki/Temple_Run

[3] http://docs.phonegap.com/en/4.0.0/index.html