ECE 535: Mobile Development & Ubiquitous Computing Systems

Instructor: Dr. Paul Watta

Name: Srinivas Sambaraju

Date: Dec 16, 2013

Final Project Report

Introduction

The main objective in designing this App is to understand the usage of Threads, Animation and Sensors in Android. The UI is handled in the main thread. The Animation and Sensors are handled in an auxiliary thread.

How to play the OnTarget Game

The Crux of this game is to guide a moving Arrow towards a moving target. The buttons on the screen help navigate to the start of the game. To move the arrow up or down, Tilt the screen Left-Up or Left-Down vice versa. When the Arrow hits the Target it explodes and a new Target is created in its place. A new Arrow is also created and it appears on the other side of the screen and the game continues for 60 seconds (1 minute).

There are two choices available to play the game. In the Beginner mode, 5 target hits are required to win. In the advanced mode, 10 are required. The difference between the modes is the speed of movement of the Arrow and Target. Naturally, the beginner mode has lower speeds to allow for more control.

The Tilt/Orientation information calculated from the Device's Sensor's is utilized to move the arrow up or down to position it in line with the target. When the time is up, there is a choice to retry the game with any of the Beginner or Advanced options.

Settings used

- Android.media.JetPlayer class is used in this App. It can also play background sound.
- Hardware acceleration = false setting is required in the Manifest to show animations like flying bird or an explosion in the Android phone.
- Landscape orientation is used.
- Game completes after 60 seconds, a timer is used on the game thread to control time.
- Sensors are continually broadcasting data but they are only processed when the thread gets around to the method that handles sensor feedback.

Android features used in game design

The App functions within the Android framework utilizing the rich source of classes and functionality available. A custom thread that extends thread is used instead of ASync-task due to the difficultly in handling both Graphics and Sensors at the same time. LogCat and the emulator are very useful in the initial stages of the game when sensors are not required much and also to check how that parameters are updated during thread execution.

Thread

A thread can be used in a situation where continuous execution is required for a task. In this case if animation is done on the main UI thread, it will become unresponsive. So the

repetitive and time consuming task can be done in the background using a Thread. Animation is a good scenario to use Thread's since the screen needs to be redrawn multiple times.

The custom thread class also uses a TimerTask to keep track of time, since the game only runs for a minute. The Start, Play and Retry button states are used to set the thread state after reading its current state. These are handled by the UI thread. The thread stops when the Timer tracks if 60 seconds are completed. When the App is closed, the SensorListeners are unregistered.

The custom thread class is created inside a View that extends SurfaceView class. The SurfaceView class determines the canvas width and height and passes them on to the thread after it instantiates the thread. The thread implements OnJetEventListener and a SensorEventListener. The OnJetEventListener functionality is not used much in this App except for the background sound.

Animation

The App sets Orientation to Landscape mode. When we start the App, we will see an Arrow coming from the right and moving to the left. The Target is rotating and also moving on the left side of the screen. The explosion is animated using multiple pictures with different graphics. The thread enables us to draw the Arrow and Target in increments along the canvas from one end of the phone to the other.

A custom class keeps track of the Arrow progress. We check if the Arrow scrolled off the screen, so that we can create a new Arrow. The background need not be painted repeatedly. Only the Arrow and Target is painted again and again as it moves on. The Hardware Acceleration property should be set to false in the Manifest to play the Target animation on the phone. The Custom class extends SensorEventListener which enables it to listen to OnSensorChanged events to calculate the tilt and control the animation of the Arrow.

Comparing the Arrow X and Y positions with those of the Target, we can determine if they are at collision distance. If they are, we set the explosion flag to true and create the explosion animation, and then create a new Target in its place.

Orientation (Tilt) Sensor

We have a SensorEventListener to keep track of the sensor changes. Using a simple calculation we can determine the Orientation or Tilt using the values of Acceleration and Magnetic field. Once we know the Tilt we need to apply it to the Arrow. In this App, when the Phone is Left side down – Arrow goes down, Right side down – Arrow goes up.

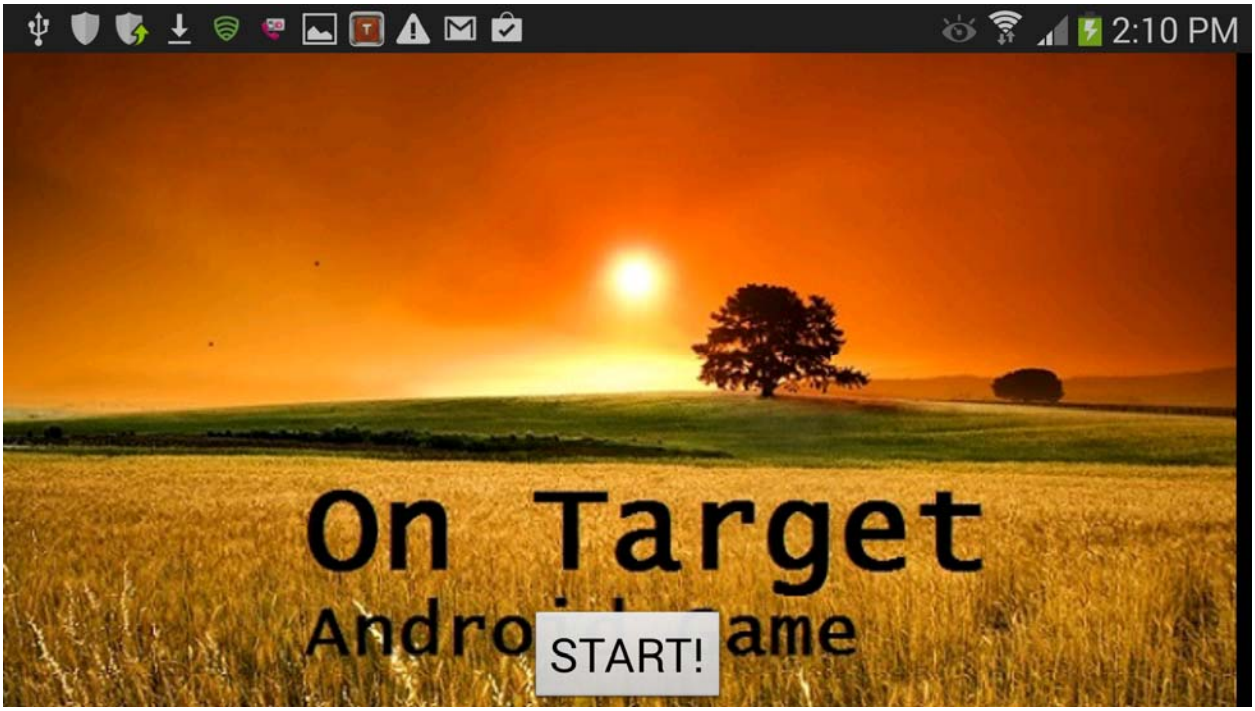
We get the value of Accelerometer and magnetic field from the sensors and based on these the rotation matrix is calculated. We can then calculate the Orientation using the Rotation matrix. For 90 degree rotation of the phone the y-axis orientation value goes from 0 – 1.4. We then scale this value to get an angle between to 0-90. Based on the angle, length of the

arrow we can determine how far to move the arrow from the center line. We perform this task repeatedly in the thread and control the Arrow position according to the Tilt of the phone.

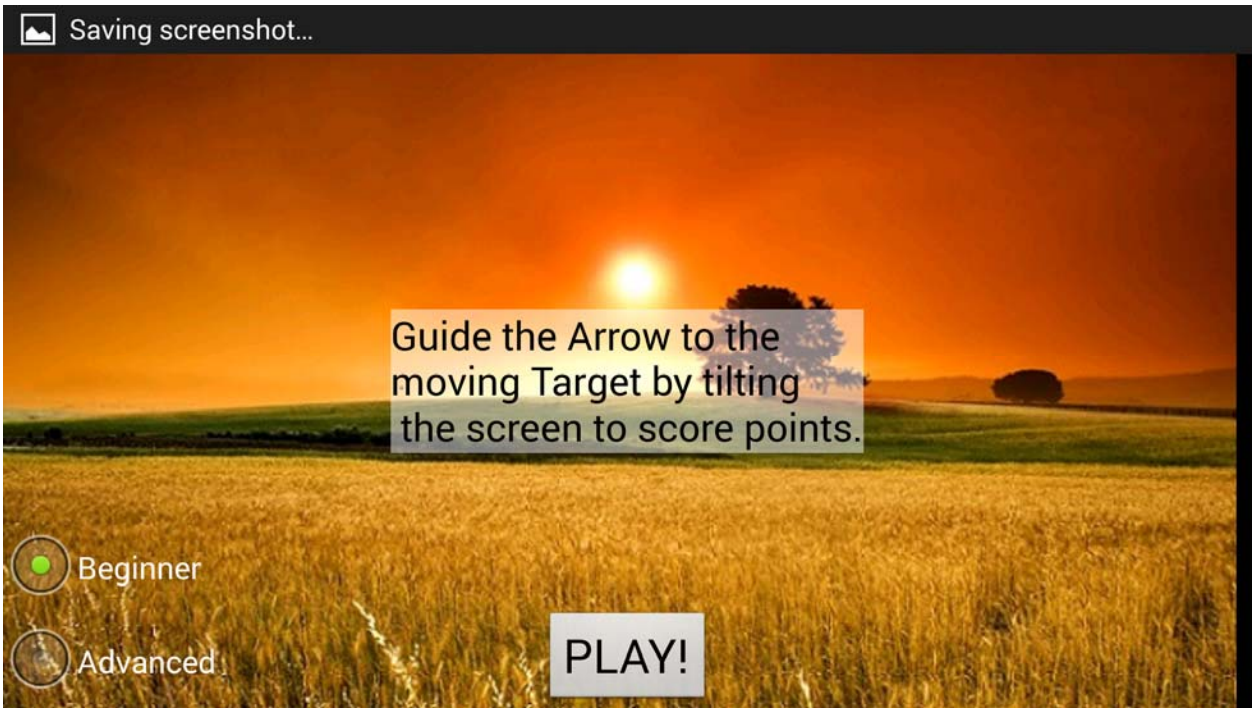
Since the Arrow can scroll off the phone, there are max and min limits to the Tilt beyond which the position of the Arrow does not change. This App has been tested on a Samsung Galaxy S4. The Sensors are not available on the Emulator, so testing is possible only on a phone.

Screen Shots of the OnTarget App

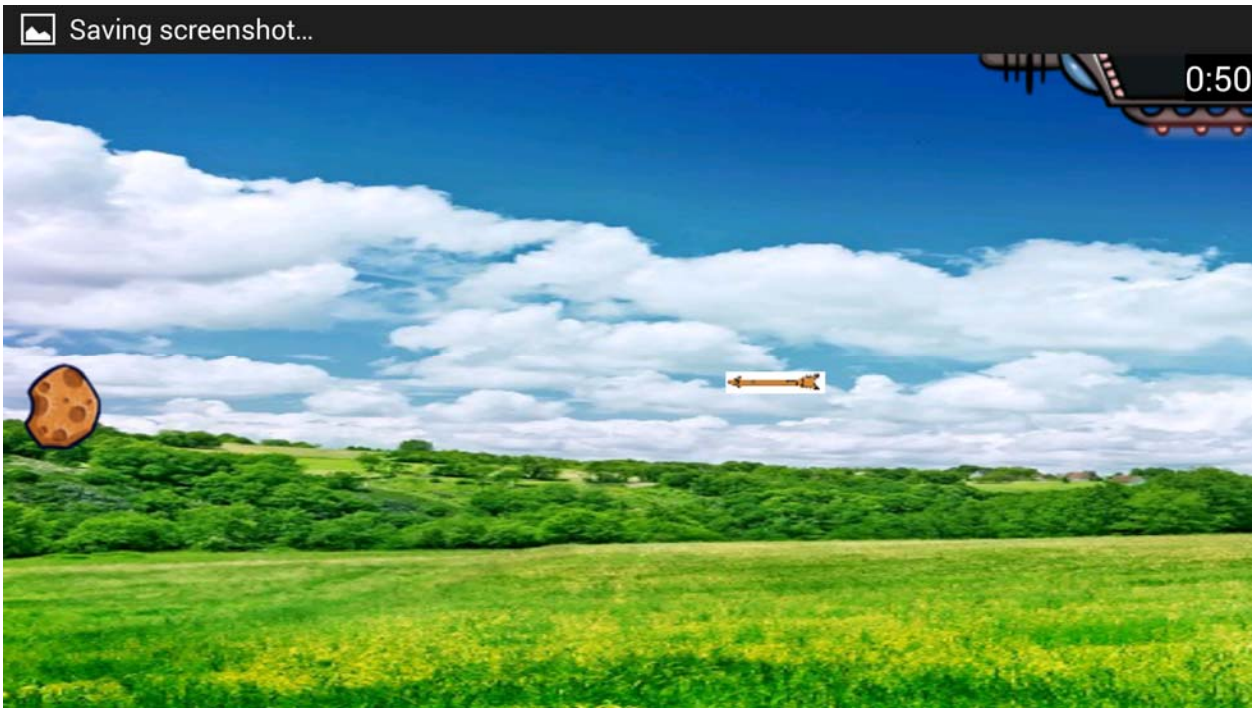
Six Screen shots are presented here. First is the home screen of the App. If the START button is clicked or selected, the next screen gives instructions to play and also a choice of 'Beginner' and 'Advanced'. When the PLAY button is clicked, the game starts. The next three screens show the position of the Arrow as it moves on the screen. The last screenshots shows the screen when the game is completed. A message is displayed to show if the game is won or lost. The Retry button takes back to screen 2.



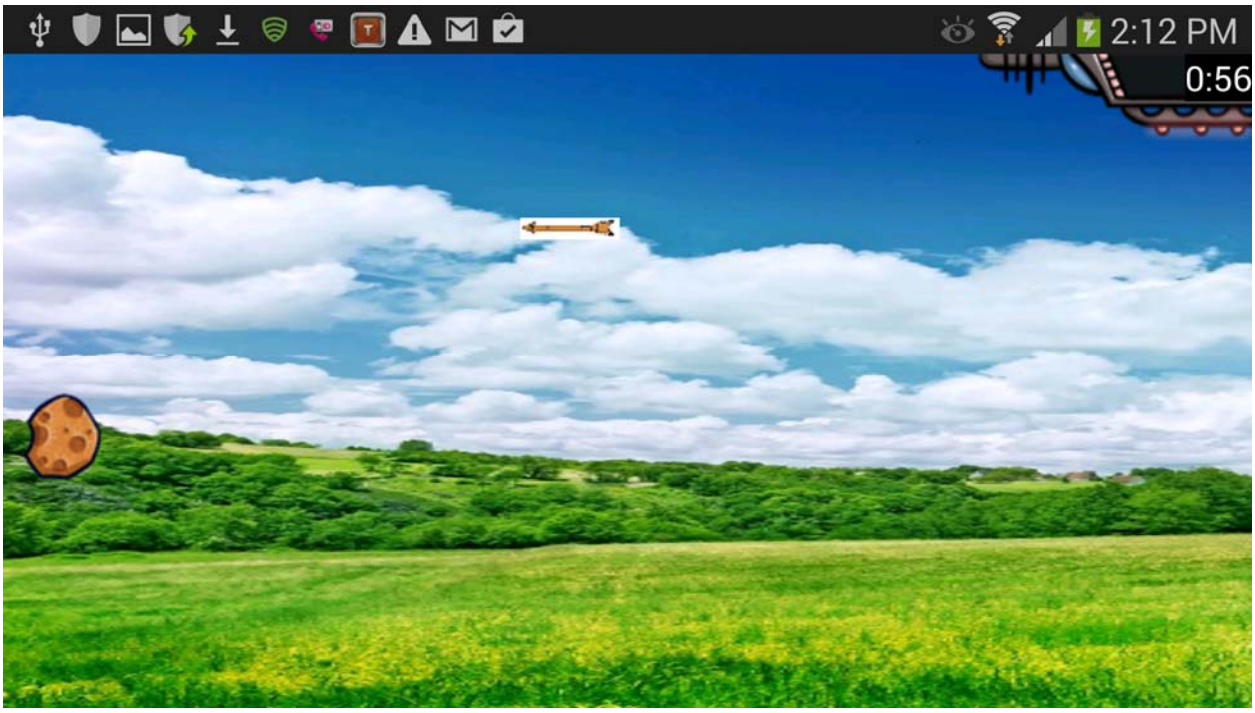
Home Screen of the App



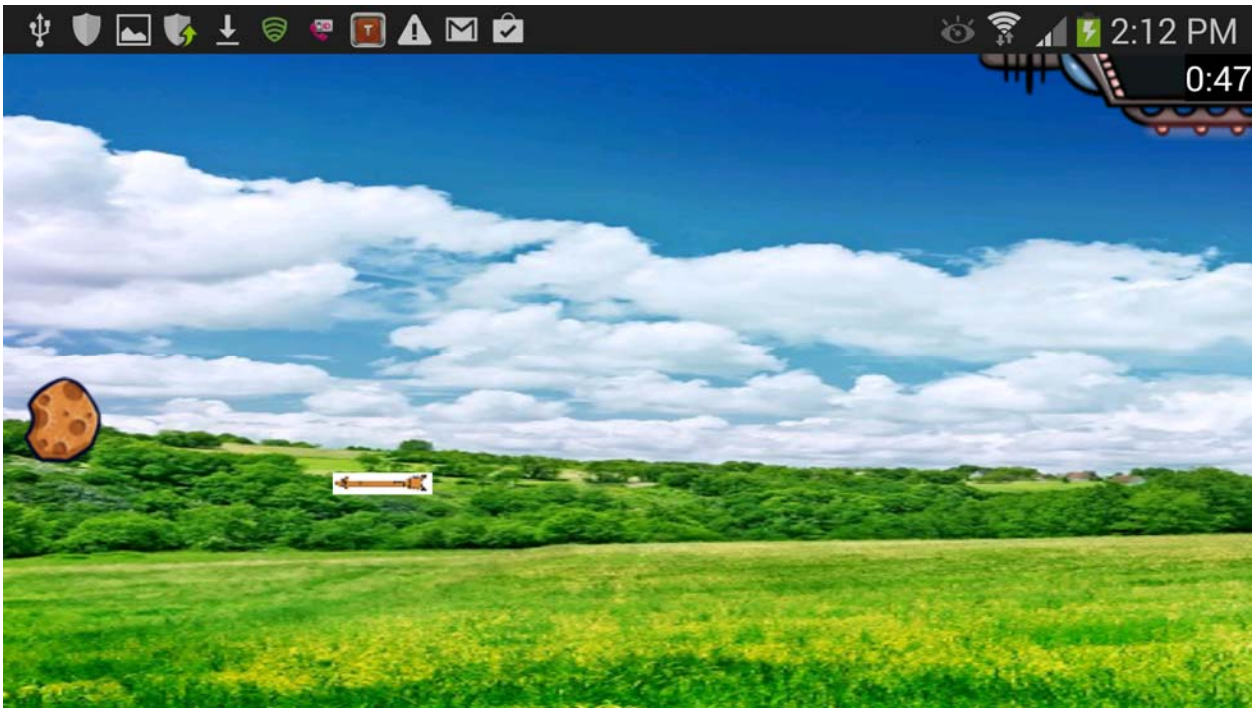
Second screen – make a selection here



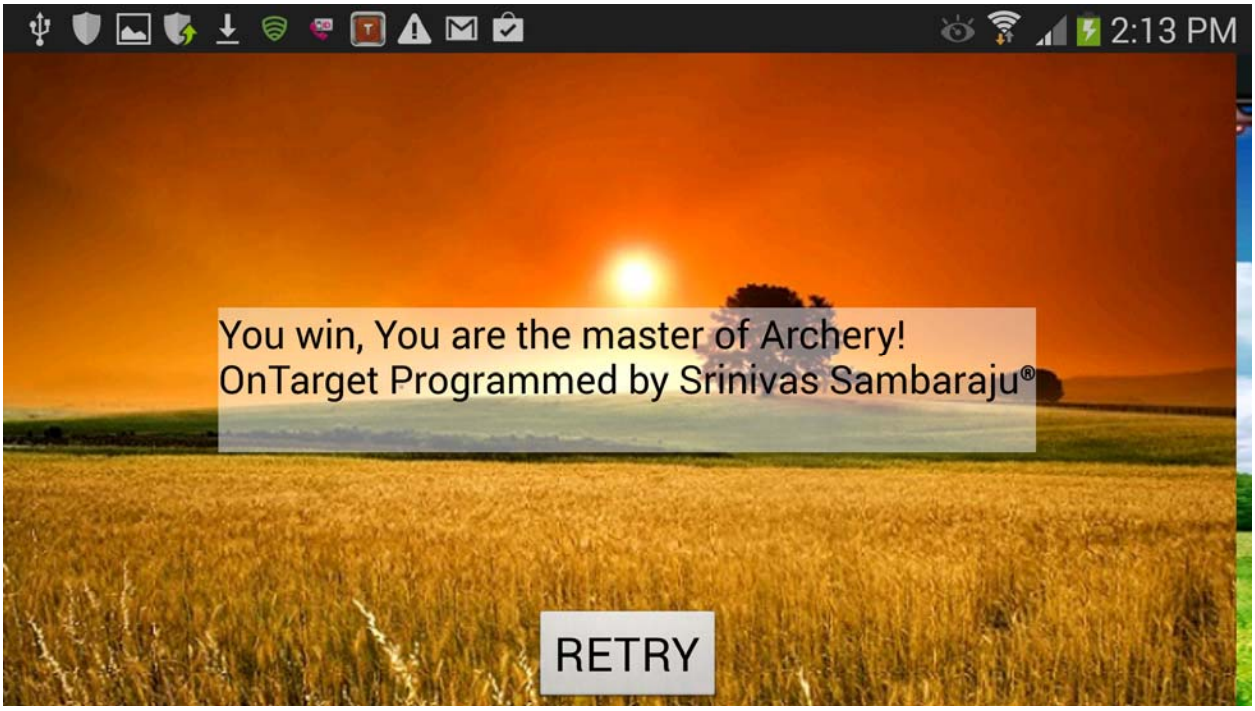
Arrow Animation - 1



Arrow Animation - 2



Arrow Animation - 3



Final Screen – Choose Retry to replay the game

References

http://developer.android.com/guide/topics/sensors/sensors_overview.html
http://developer.android.com/guide/topics/media/jet/jetcreator_manual.html
http://developer.android.com/reference/android/media/JetPlayer.OnJetEventListener.html
http://www.codingforandroid.com/2011/01/using-orientation-sensors-simple.html
http://www.techotopia.com/index.php/A_Basic_Overview_of_Android_Threads_and_Thread_handlers
http://wallpaperswide.com/summer_scenery-wallpapers.html
http://74211.com/free-scenery-wallpaper-includes-a-sunny-day-the-best-time/