Sheetal Jena, Johncarlo Cerna, Varun Iyer

Group 6

Dr. Shankar

Mobile Android and Development

1 July 2014

Group 6 Report

**Abstract**

The purpose of the project is to create an Android application that will guide users through the Museum of Discovery and Science and make their visit a more interactive experience. The app is intended for a broad audience of all ages. The goal of the app, MODS of Fun, is to create an entertaining visit at the museum for families and help them enjoy their time while utilizing technology in order to create a fun and educational experience. By featuring both informational and interactive aspects, MODS of Fun will aim to achieve this goal. Our game will hopefully interest many users, since we aimed to make it similar to the hit game Flappy Bird. The game also included random fun-facts which is aimed at providing an educational aspect towards the user. The app also consists of a useful survey which collects feedback, ultimately benefitting the museum and the developers. At the same time, the weekly survey will collect data and save it to a database called Parse.com. We can then send the data to the museum so they can improve certain aspects based on the feedback from our surveys.

**Background**

After touring the Museum of Discovery and Science, our group decided that the best idea was to create an interactive map of the grounds and to focus on one exhibit/attraction that needed elaboration. The reasoning behind this was that the museum can be hard to navigate through, and having a map for the museum that can guide users to where they want to go can save a lot of hassle for newcomers and veterans alike. Also notable was that the long lines and semi-far distances to the exhibits merited some sort of diversion to pass the time. The game was designed to combat boredom by having a high replay value and an easy interface.

**Methods**

For our app, we used both Processing and the Eclipse environment to create it. Processing is an environment that increases user enhancement and functionality by providing methods that allow more artistic creations to be developed, such as rectangles and ellipses. This program was primarily used to add an artistic touch to some of the activities in the app and make them more visually appealing as well. Eclipse was used as a linkage between the android application and Processing; exporting the project from Processing to Eclipse allowed us to run the Processing applications on our android devices and add the projects to other existing projects or new projects that we wanted to create. We used this feature to add all our Processing projects into one project generated in Eclipse so we could run each project simultaneously at once and not have to run each one separately.

The minimum version required to run our app is Android 2.3 (Api level 8). The latest version our app can run is the most current operating system Android has, Android 4.4 (Api level 19). We ran the project on both a Google Nexus 2013 and 2012 editions, and both of them were provided by FAU for development use.

In Processing, we created the game, the interactive map, and the survey. The game, originally, was designed so that it would mimic a well-known and popular game among all ages, Flappy Bird. We wanted to create a game that would be fun for both adults and kids. However, instead of recreating the game, we decided to add our own theme by making the game symbolize the Everglades. To create the game, we followed a tutorial1 online.

We believed that through the game, users would become more intrigued and interested about the problems associated with the Everglades and even start to organize community projects to help protect them as well. Instead of a bird, we decided to use a turtle, and we changed the background so it represented an oceanic scenery. We also changed the name from “Flappy Bird” to “Swimmy Turtle.”

Also, when the user died, we generated a random fun fact to make the game appear to have an interactive and learning experience. This way, users can learn small bits of information while enjoying playing a fun game. We chose to include small fun facts instead of pasting large copies of information because it would increase the sleekness and design of our app. We thought that simply pasting lots of information would discourage users from reading it since so much information would be presented to them in such a tiny screen. We want our readers to interact with our app and learn new information but not in a way that makes the user unhappy. By using fun facts, we eliminated the need for mass information and instead put it into a short 1-2 lines paragraph that is simple and easy to understand. Below is a snippet of the code from the game. Please read Appendix A for the full source code.

*Figure 1. In these few lines of code it shows how the image is added and also resized to fit based on the width and height of the android device, and then. It uses an if-else conditional statement and passes a random int value to a parameter called a. The images are then loaded based on the value of a.*

Our second feature created in Processing was our interactive map. Our methodology for the map was to provide something unique that was not something that was blatantly obvious but ingenious at the same time. Mr. Shankar, our instructor for this course, gave us a book called “Getting Started with Processing2,” which showed various features of android being used in Processing, such as Geolocation, gestures, and NFC communication between android devices. In the book, it showed a tutorial on how to use geolocation to make your android device become a navigator.

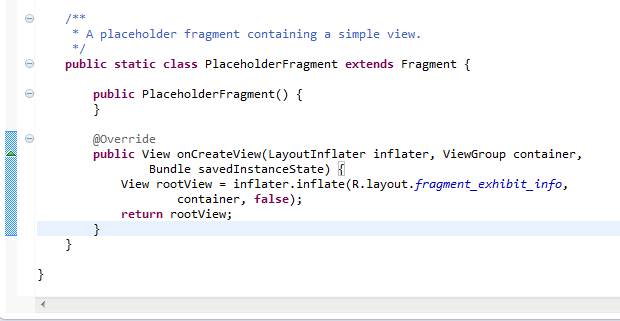
After reading it, we thought that it would a very useful feature to implement into an interactive map. We utilized this feature so that users could type in the location they wanted to go to, and then press submit. After that, they long press the screen and the device runs the GPS navigation to that specific location. The reason why we chose to use this unique idea is because we want to readdress the concept of interaction. Our app is not simply designed to present information cleanly and in a neat and presentable manner; we want to make users interact with it to help them learn. Our interactive map does not simply just present a floor map that when clicked on, opens a new view that presents more information about each exhibit. It was be considered too dull, it would not seem to interest users. We wanted to improve user interaction and make users type in their location and be navigated towards it as well. The full source code is available in Appendix B.

Our last feature added in Processing was the survey. This idea was initially developed because we wanted to find a method in order to relay data back to museum and inform them what people think of their museum so they can improve it based on the opinions given. We then decided that a survey would best fit this purpose.

What we did was we created a screen title survey, and then it had a text box that would feature the weekly survey question for users to answer. In order to generate the questions, we were going to use a site called Parse.com. Parse.com is a database that allows you to either grab data from the server into your project or release the data from your project to the server. We are using both methods.

First, we are using the database to grab the questions from it and send it to our app so that way whenever we change the question it gets updated automatically in the app. This way we also do not have to update our app every week to accommodate for the weekly survey question. Then the user will be able to tap the number of stars based on their answer towards the question. Also, users can double tap the screen to reset the stars and make a choice again. 5 is the maximum, and 1 would be the minimum. Once users have selected their preference, they can press the submit button at the bottom and the data will be sent into a spreadsheet containing the ratings that users have submitted. We will then send these ratings to the museum and they can use the data to change whatever is necessary to their museum. By creating this survey, we are understanding each user’s feelings about the museum and their opinions about as well and the museum can benefit by understand what visitors like and dislike about the museum. To see full source code, please see Appendix C.

The other environment used to program our app was Eclipse. We used this environment because we didn’t need to use utilize the artistic creations made in Processing. We used Photoshop to handle the background image and color scheme. It also was easier to position the text and create widgets.

Using Eclipse, we created our last two screens, “Exhibit Info” and “About Us.” In the “Exhibit Info” screen, we created a page with a small paragraph that lists facts about the Everglades. This page had the intent of allowing users to understand what the everglades are understand the dire necessity to save them as well. Using photoshop, we made the background image visually appealing using primary color schemes. This allows the user to view the information in a setting that makes it looks neat. It also counteracts the dull and mass information presented at the user. Here is a snippet of code, you can view the full source code in Appendix D. 

*Figure 2. In these few lines of code it shows how a new view is created and the fragment “exhibit\_info” is added to the main activity.*

The last feature of our app, also created in Eclipse, is an “About Us” screen, This screen is basically designed as a basic UI to give credit to all to everyone who has helped this app, including the professor and his TAs. It also cites the credit of all the images gathered from the internet. You can view the full source code at Appendix E.

**Results**

When the user opens the app, a menu screen comes up showing the title of our app, MODs of Fun, and 5 buttons below it, “Game,” “Map,” “Exhibit Info,” “Survey,” and “About Us.” If the users presses either of these buttons, it sends them to a new activity/screen. The android OS automatically generates a back button, so clicking on it will allow the user to go back to the menu screen.

Sometimes, when running the app, the user may experience some difficulties and the app may unexpectedly crash (most of the time it occurs when the interactive map is run). We do not know the reason for these bugs, but we are trying our best to locate the source of those problems and will update the app accordingly as soon as we find it. Reloading the app again will usually fix the problem. Also, on older devices such as the Samsung galaxy S3 or the Google Nexus tablet (2012 version), our game may not run because the memory allocation for the game is too low. It is possible to run the game as a separate project, which we have also included in the link to our final project. We are trying to figure out to increase the memory allocation and will update as soon as we find a working solution for that as well.

To view screenshots of each screen in the app, please see Appendix F.

**Discussion**

Even though our app is very good, there is still work that could be improved. One thing is to change the UI and layout so that it looks for cleaner and presentable. Our text is placed in such a way that its not uniform and centered, so it doesn’t look nice. Along with that, another alternative we could have seeked was using tabs and a sidebar instead of creating multiple buttons. In this the layout is improved as well. Another feature we did not add was instructions. We feel that our app isn’t known well enough to be able to perform the anticipated actions without guiding the user first. In the future, we are going to create a separate screen that houses the instructions for the entire app.

For the game, we are also facing some bugs because sometimes the score isn’t incremented when the turtle passes through a pipe, or the sound is not played either. Also, some of the fun facts have the text squeezed and very small, so it makes it hard for readers to read it. In the future, we are thinking about changing the font style and size so that users would be able to read the fun facts more easily. Another feature we are thinking about implements is grabbing the fun facts from a database (Parse.com) so we don’t need to hardcode it in constantly when we add more fun-facts.

**Conclusion**

Overall, the application followed our design. It has 5 menu options. Our first one is our game, and users will be able to play a fun, addictive, and interactive game. When users die as well, they get randomly generated fun fact to read. Then the user can play again or go back to menu screen by pressing the back button. The second button is the map, where again, users can interact with it by typing in the destination they would like to go to. Once they finish typing, they press the submit button and an ellipse appears on the center of the screen. Then the user long presses on the screen, and the nexus becomes a navigator, showing a giant triangle pointing to the user’s destination. It also shows the distance to that destination both in feet and miles.

The third tab is our exhibit info, which simply displays some information about the Everglades. The fourth tab is a our survey, where we ask weekly survey questions and the users responds by picking between 1 to 5 stars, symbolizing their opinions about the question. The data is then stored in an online database, and we will release the information monthly to the museum so they can understand the views of their visitors and make changes as necessary. Our last button is the “About Us” screen, where we simply show the credit, acknowledge the people who have helped with the app, and cite sources that gave us either information or pictures.

**References**

1. http://iamjbn.blogspot.com/2014/04/how-to-create-flappy-bird-in-processing.html

2. Sauter, Daniel. *Rapid Android Development: Build Rich, Sensor-based Applications with Processing*. Dallas, Tx.: Pragmatic shelf, 2013. Print

**Appendices**

**Appendix A - FlappyBird.java**

package processing.test.flappybird;

import processing.core.\*;

import processing.data.\*;

import processing.event.\*;

import processing.opengl.\*;

import ketai.sensors.\*;

import apwidgets.\*;

import android.os.Environment;

import java.util.HashMap;

import java.util.ArrayList;

import java.io.File;

import java.io.BufferedReader;

import java.io.PrintWriter;

import java.io.InputStream;

import java.io.OutputStream;

import java.io.IOException;

public class flappyBird extends PApplet {

//import ddf.minim.\*;

APMediaPlayer wingSound, hitSound, scoreSound, dieSound;

//Minim minim;

Table tsv;

ArrayList<Stump> stumps;

float stumpMIN=0,stumpMAX=0,stumpDiffY=150,stumpDiffX=250;

int stumpCount=0;

int birdX,score=0,nextStump=0,i,TopScore;

PImage[] imgNumBig,imgNumSmall;

Boolean hit=false,hitSnd=false,stumpHit=false,gameOver=false;

PVector birdPosition,velocity,gravity,up;

PImage[] bird;

PImage background,base,stump,stumpi;

int brd=0,deg=0,baseInc=0;

float BY,BGY;

int scene=0;

PImage imgTitle,imgGetReady;

PImage imageGameOver,imageScoreCard,imageClick,goldScoreCard, funFact1, funFact2, funFact3, funFact4, funFact5, funFact6, funFact7, funFact8, funFact9, funFact10, funFact11, funFact12, funFact13, funFact14, funFact15, funFact16, funFact17, funFact18, funFact19, funFact20, funFact21, funFact22, funFact23, funFact24, funFact25;

int gameOverPosY,scoreCardPosY;

int a;

char ch;

public void setup(){

frameRate(120);

//topScoreFileLoader();

orientation(LANDSCAPE);

a = (int)random(25);

wingSound = new APMediaPlayer(this); //create new APMediaPlayer

wingSound.setMediaFile("wing.mp3"); //set the file (files are in data folder)

//wingSound.start(); //start play back

//wingSound.setLooping(true); //restart playback end reached

wingSound.setVolume(1.0f, 1.0f); //Set left and right volumes. Range is from 0.0 to 1.0

hitSound = new APMediaPlayer(this); //create new APMediaPlayer

hitSound.setMediaFile("hit.mp3"); //set the file (files are in data folder)

//hitSound.start(); //start play back

//hitSound.setLooping(true); //restart playback end reached

hitSound.setVolume(1.0f, 1.0f); //Set left and right volumes. Range is from 0.0 to 1.0

scoreSound = new APMediaPlayer(this); //create new APMediaPlayer

scoreSound.setMediaFile("point.mp3"); //set the file (files are in data folder)

//scoreSound.start(); //start play back

//scoreSound.setLooping(true); //restart playback end reached

scoreSound.setVolume(1.0f, 1.0f); //Set left and right volumes. Range is from 0.0 to 1.0

dieSound = new APMediaPlayer(this); //create new APMediaPlayer

dieSound.setMediaFile("die.mp3"); //set the file (files are in data folder)

//dieSound.start(); //start play back

//dieSound.setLooping(true); //restart playback end reached

dieSound.setVolume(1.0f, 1.0f); //Set left and right volumes. Range is from 0.0 to 1.0

/\*

wingSound = minim.loadSample("wing.mp3",512);

hitSound = minim.loadSample("hit.mp3",512);

scoreSound = minim.loadSample("point.mp3",512);

dieSound = minim.loadSample("die.mp3",512);

\*/

imgNumBig = new PImage[10];

imgNumSmall = new PImage[10];

for(i=0;i<10;i++){

imgNumBig[i]=loadImage(i+".png");

imgNumBig[i].resize(((26\*height)/700),((36\*height)/700));

imgNumBig[i].loadPixels();

imgNumSmall[i]=loadImage(i+".png");

imgNumSmall[i].resize(((13\*height)/700),((18\*height)/700));

imgNumSmall[i].loadPixels();

}

tsv = new Table();

// make the header columns for the table

tsv.addColumn("High Score");

bird = new PImage[3];

bird[0]=loadImage("turtle.png");

bird[1]=loadImage("turtle.png");

bird[2]=loadImage("turtle.png");

bird[0].resize(((34\*height)/700), ((24\*height)/700));

bird[0].loadPixels();

bird[1].resize(((34\*height)/700), ((24\*height)/700));

bird[1].loadPixels();

bird[2].resize(((34\*height)/700), ((24\*height)/700));

bird[2].loadPixels();

base=loadImage("turtle\_base.png");

stump=loadImage("stump.png");

stumpi=loadImage("stumpi.png");

base.resize(width+100, ((111\*height)/700));

base.loadPixels();

stump.resize(((52\*height)/700),((489\*height)/700));

stump.loadPixels();

stumpi.resize(((52\*height)/700),((489\*height)/700));

stumpi.loadPixels();

stumps = new ArrayList<Stump>();

stumpCount = (int) (width/(stumpDiffX+stump.width));

stumpCount++;

stumps.add(new Stump(random(stumpMIN,stumpMAX),0));

for(int k=1;k<=stumpCount;k++){

stumps.add(new Stump(random(stumpMIN,stumpMAX),stumpDiffX\*k));

}

imgTitle=loadImage("mods\_title.png");

imgTitle.resize(((178\*height)/700), ((48\*height)/700));

imgTitle.loadPixels();

imgGetReady=loadImage("getReady.png");

imgGetReady.resize(((184\*height)/700), ((42\*height)/700));

imgGetReady.loadPixels();

imageGameOver=loadImage("gameOver.png");

imageGameOver.resize(((192\*height)/700), ((42\*height)/700));

imageGameOver.loadPixels();

imageScoreCard=loadImage("scoreCard.png");

imageScoreCard.resize(((226\*height)/700), ((114\*height)/700));

imageScoreCard.loadPixels();

imageClick=loadImage("click.png");

imageClick.resize(((114\*height)/700), ((98\*height)/700));

imageClick.loadPixels();

goldScoreCard=loadImage("goldScoreCard.png");

goldScoreCard.resize(((226\*height)/700), ((114\*height)/700));

goldScoreCard.loadPixels();

funFact1 = loadImage("fun fact #1.png");

funFact2 = loadImage("fun fact #2.png");

funFact3 = loadImage("fun fact #3.png");

funFact4 = loadImage("fun fact #4.png");

funFact5 = loadImage("fun fact #5.png");

funFact6 = loadImage("fun fact #6.png");

funFact7 = loadImage("fun fact #7.png");

funFact8 = loadImage("fun fact #8.png");

funFact9 = loadImage("fun fact #9.png");

funFact10 = loadImage("fun fact #10.png");

funFact11 = loadImage("fun fact #11.png");

funFact12 = loadImage("fun fact #12.png");

funFact13 = loadImage("fun fact #13.png");

funFact14 = loadImage("fun fact #14.png");

funFact15 = loadImage("fun fact #15.png");

funFact16 = loadImage("fun fact #16.png");

funFact17 = loadImage("fun fact #17.png");

funFact18 = loadImage("fun fact #18.png");

funFact19 = loadImage("fun fact #19.png");

funFact20 = loadImage("fun fact #20.png");

funFact21 = loadImage("fun fact #21.png");

funFact22 = loadImage("fun fact #22.png");

funFact23 = loadImage("fun fact #23.png");

funFact24 = loadImage("fun fact #24.png");

funFact25 = loadImage("fun fact #25.png");

initORreset();

}

public void draw(){

//if(ch=='L'){ lc=1;}

//if(ch=='R'){ lc=0;}

if (keyPressed) {

if(key =='R'|| key =='L')//if(lc==1)

{

switch(scene)

{

case 0: //title screen

scene=1;

break;

case 1: //get ready

scene=2;

break;

case 2: //game

if(!stumpHit)

{

velocity.add(up);

wingSound.start();

}

break;

case 3: //game over

initORreset();

scene=1;

break;

}

}

}

TableRow row = tsv.addRow();

row.setInt("High Score", TopScore);

drawBackground();

textSize(35);

fill(255,100);

textAlign(CENTER);

switch(scene)

{

case 0:

case 1:

image(base,baseInc,BY);

if(scene==0)

{

image(imgTitle,width/2 - imgTitle.width/2,height/4);

image(bird[brd/10],width/2 - bird[0].width/2,height/2 + sin(radians(deg)\*10)\*5 -50);

}

else

{

image(imgGetReady,width/2 - imgGetReady.width/2,height/4);

image(bird[brd/10],width/4,height/2 + sin(radians(deg)\*10)\*5 -50);

image(imageClick,width/2 - imageClick.width/2,height/3 +50);

}

brd+=1;

deg+=1;

if(brd>20) brd=0;

if(deg>360) deg=0;

break;

case 2:

for(int k=0;k<stumps.size();k++){

Stump st = stumps.get(k);

st.checkHit();

if(!stumpHit && !hit)

{

st.checkPassed();

checkScored();

st.update();

}

else

{

gameOver=true;

}

st.display();

}

image(base,baseInc,BY);

flyingBird();

/\*stroke(0);

line(0,stumpMIN,width,stumpMIN);

stroke(#ffffff);

line(0,stumpMAX,width,stumpMAX);\*/

if(hitSnd){hitSound.start(); fill(255); for(int z=0;z<100;z++) rect(0,0,width,height); hitSnd=false; if(stumpHit) dieSound.start();}

printNum(score,width/2,height/6,'b');

break;

case 3:

for(int k=0;k<stumps.size();k++){

Stump st1 = stumps.get(k);

println(stumps.get(k));

st1.display();

}

image(base,baseInc,BY);

translate(birdPosition.x,birdPosition.y);

rotate(radians(90));

image(bird[0],-bird[0].width/2,-bird[0].height/2);

resetMatrix();

if(gameOverPosY<height/6) gameOverPosY+=5;

if(scoreCardPosY>height/6 +100) scoreCardPosY-=10;

image(imageGameOver,width/2 - imageGameOver.width/2,gameOverPosY + (100\*height)/1200);

if(score>TopScore)

image(goldScoreCard,width/2 - goldScoreCard.width/2,height/2 - (175\*height)/1200);

else

image(imageScoreCard,width/2 - imageScoreCard.width/2,height/2 - (175\*height)/1200);

//print Topscore

printNum(score>TopScore?score:TopScore,width/2 + (120\*width)/1920,scoreCardPosY+(215\*height)/1200,'s');

printNum(score,width/2 + (120\*width)/1920,scoreCardPosY+(138\*height)/1200,'s');

if(a == 1){

image(funFact1, (width\*3)/5+(25\*width)/1920, height/10);

funFact1.resize(665\*width/1920, 331\*height/1200);

funFact1.loadPixels();

}

else if(a == 2){

image(funFact2, (width\*3)/5+(25\*width)/1920, height/10);

funFact2.resize(665\*width/1920, 331\*height/1200);

funFact2.loadPixels();

}

else if(a == 3){

image(funFact3, (width\*3)/5+(25\*width)/1920, height/10);

funFact3.resize(665\*width/1920, 331\*height/1200);

funFact3.loadPixels();

}

else if(a == 4){

image(funFact4, (width\*3)/5+(25\*width)/1920, height/10);

funFact4.resize(665\*width/1920, 331\*height/1200);

funFact4.loadPixels();

}

else if(a == 5){

image(funFact5, (width\*3)/5+(25\*width)/1920, height/10);

funFact5.resize(665\*width/1920, 331\*height/1200);

funFact5.loadPixels();

}

else if(a == 6){

image(funFact6, (width\*3)/5+(25\*width)/1920, height/10);

funFact6.resize(665\*width/1920, 331\*height/1200);

funFact6.loadPixels();

}

else if(a == 7){

image(funFact7, (width\*3)/5+(25\*width)/1920, height/10);

funFact7.resize(665\*width/1920, 331\*height/1200);

funFact7.loadPixels();

}

else if(a == 8){

image(funFact8, (width\*3)/5+(25\*width)/1920, height/10);

funFact8.resize(665\*width/1920, 331\*height/1200);

funFact8.loadPixels();

}

else if(a == 9){

image(funFact9, (width\*3)/5+(25\*width)/1920, height/10);

funFact9.resize(665\*width/1920, 331\*height/1200);

funFact9.loadPixels();

}

else if(a == 10){

image(funFact10, (width\*3)/5+(25\*width)/1920, height/10);

funFact10.resize(665\*width/1920, 331\*height/1200);

funFact10.loadPixels();

}

else if(a == 11){

image(funFact1, (width\*3)/5+(25\*width)/1920, height/10);

funFact11.resize(665\*width/1920, 331\*height/1200);

funFact11.loadPixels();

}

else if(a == 12){

image(funFact12, (width\*3)/5+(25\*width)/1920, height/10);

funFact12.resize(665\*width/1920, 331\*height/1200);

funFact12.loadPixels();

}

else if(a == 13){

image(funFact13, (width\*3)/5+(25\*width)/1920, height/10);

funFact13.resize(665\*width/1920, 331\*height/1200);

funFact13.loadPixels();

}

else if(a == 14){

image(funFact14, (width\*3)/5+(25\*width)/1920, height/10);

funFact14.resize(665\*width/1920, 331\*height/1200);

funFact14.loadPixels();

}

else if(a == 15){

image(funFact15, (width\*3)/5+(25\*width)/1920, height/10);

funFact15.resize(665\*width/1920, 331\*height/1200);

funFact15.loadPixels();

}

else if(a == 16){

image(funFact16, (width\*3)/5+(25\*width)/1920, height/10);

funFact16.resize(665\*width/1920, 331\*height/1200);

funFact16.loadPixels();

}

else if(a == 17){

image(funFact17, (width\*3)/5+(25\*width)/1920, height/10);

funFact17.resize(665\*width/1920, 331\*height/1200);

funFact17.loadPixels();

}

else if(a == 18){

image(funFact18, (width\*3)/5+(25\*width)/1920, height/10);

funFact18.resize(665\*width/1920, 331\*height/1200);

funFact18.loadPixels();

}

else if(a == 19){

image(funFact19, (width\*3)/5+(25\*width)/1920, height/10);

funFact19.resize(665\*width/1920, 331\*height/1200);

funFact19.loadPixels();

}

else if(a == 20){

image(funFact20, (width\*3)/5+(25\*width)/1920, height/10);

funFact20.resize(665\*width/1920, 331\*height/1200);

funFact20.loadPixels();

}

else if(a == 21){

image(funFact21, (width\*3)/5+(25\*width)/1920, height/10);

funFact21.resize(665\*width/1920, 331\*height/1200);

funFact21.loadPixels();

}

else if(a == 22){

image(funFact22, (width\*3)/5+(25\*width)/1920, height/10);

funFact22.resize(665\*width/1920, 331\*height/1200);

funFact22.loadPixels();

}

else if(a == 23){

image(funFact23, (width\*3)/5+(25\*width)/1920, height/10);

funFact23.resize(665\*width/1920, 331\*height/1200);

funFact23.loadPixels();

}

else if(a == 24){

image(funFact24, (width\*3)/5+(25\*width)/1920, height/10);

funFact24.resize(665\*width/1920, 331\*height/1200);

funFact24.loadPixels();

}

else {

image(funFact25, (width\*3)/5+(25\*width)/1920, height/10);

funFact25.resize(665\*width/1920, 331\*height/1200);

funFact25.loadPixels();

}

break;

}

}

public void mousePressed(){

switch(scene)

{

case 0: //title screen

scene=1;

break;

case 1: //get ready

scene=2;

break;

case 2: //game

if(!stumpHit)

{

velocity.add(up);

wingSound.start();

}

break;

case 3: //game over

initORreset();

scene=1;

a = (int) random(25);

break;

}

}

public void initORreset()

{

if(score>TopScore)

{

TopScore=score;

saveFile("high\_score\_data", tsv);

//topScoreFileUpdator();

}

hit=false;hitSnd=false;stumpHit=false;gameOver=false;

score=0;nextStump=0;

birdX=width/4;

birdPosition = new PVector(birdX,height/2 -50);

velocity = new PVector(0,0);

gravity = new PVector(0,0.4f);

up = new PVector(0,-8);

if((int)random(2)<1){

background=loadImage("turtle\_background.png");

background.resize(width, ((683\*height)/700));

background.loadPixels();

}

else{

background=loadImage("turtle\_background.png");

background.resize(width, ((683\*height)/700));

background.loadPixels();

}

BGY=-1\*(background.height-height+base.height);

BY=height-base.height;

stumpMIN = BY/6;

stumpMAX = BY-stumpDiffY-stumpMIN;

gameOverPosY=imageGameOver.height\*-1;

scoreCardPosY=height;

// print(stumps.size());

for(int k=0;k<stumps.size();k++){

Stump st2 = stumps.get(k);

st2.posY=random(stumpMIN,stumpMAX);

st2.posX=stumpDiffX\*k+width;

}

}

public void saveFile(String name, Table table){

String directory;

try{

directory = new String(Environment

.getExternalStorageDirectory()

.getAbsolutePath());

table.save(new File(directory+"/"+name+".tsv"), "tsv");

println("File write successful");

}catch(IOException iox){

println("Failed to write file: "+iox.getMessage());

}

}

public void drawBackground(){

image(background,0,BGY);

if(!hit)

{

baseInc-=10;

if(baseInc<-100) baseInc=0;

}

}

public void flyingBird(){

float posY=0;

if(!hit){

posY = sin(radians(deg)\*10);brd+=1;}

applyForces();

translate(birdPosition.x,birdPosition.y+posY\*5);

if(gravity.y/12 != 0)

rotate(velocity.y/12);

else rotate(radians(90));

image(bird[brd/10],-bird[brd/10].width/2,-bird[brd/10].height/2);

resetMatrix();

deg+=1;

if(brd>20) brd=0;

if(deg>360) deg=0;

floorHit();

}

public void applyForces(){

velocity.add(gravity);

velocity.limit(20);

birdPosition.add(velocity);

}

public void floorHit(){

if(birdPosition.y > BY && (!hit || stumpHit)){

birdPosition.sub(velocity);

velocity.set(0,0,0); //These are the two lines I changed when I tried to run the project in android mode.

gravity.set(0,0,0);

hit=true;

if(!stumpHit)hitSnd=true;

scene=3;

}

}

public void printNum(int n,int xPos,int Ypos, char bigOrSmall)

{

//xpos is not correct. score prints right align

int t;

if(n==0)

{

if(bigOrSmall=='s')

{

xPos-=imgNumSmall[0].width;

image(imgNumSmall[0],xPos,Ypos);

}

else if(bigOrSmall=='b')

{

xPos-=imgNumBig[0].width;

image(imgNumBig[0],xPos,Ypos);

}

}

while(n>0)

{

t= n%10;

if(bigOrSmall=='s')

{

xPos-=imgNumSmall[t].width;

image(imgNumSmall[t],xPos,Ypos);

}

else if(bigOrSmall=='b')

{

xPos-=imgNumBig[t].width;

image(imgNumBig[t],xPos,Ypos);

}

n=n/10;

}

}

/\*

void topScoreFileLoader(){

String lines[] = loadStrings("data1.aff");

TopScore=unhex(lines[2]);

}

void topScoreFileUpdator(){

String words = "5df5745h5 @#SDG54541sfs "+hex(TopScore)+" YUGYU56%^$%tgrtYTFG% HJHDS45%$%$ 8674543423&&^(DSHFJU7451#Dd";

String[] list = split(words, ' ');

// Writes the strings to a file, each on a separate line

saveStrings("data1.aff", list);

}

\*/

class Stump{

float posX=0,posY;

Stump(float y,float x)

{

posY = y;

posX = x+width;

}

public void update(){

posX-=6;

//rect(posX-20,posY,stump.width,stumpDiffY);

}

public void display(){

stumper(posX,posY);

}

public void checkPassed()

{

if( posX+stump.width < 0){

posX = stumpDiffX+width;

stumpCount++;

}

}

public void checkHit(){

if(birdPosition.x>=(posX-15) && birdPosition.x <=(posX-15)+stump.width && !stumpHit)

{

if(birdPosition.y<posY || birdPosition.y>posY+stumpDiffY){

hit=true;

stumpHit=true;

hitSnd=true;

}

}

}

}

public void stumper(float x,float y){

image(stumpi,x,y-stumpi.height);

image(stump,x,y+stumpDiffY);

}

public void checkScored()

{

if(stumps.get(nextStump).posX+stump.width < birdX){

score++;

if(nextStump<4)nextStump++; else nextStump=0;

scoreSound.start();

}

}

public int sketchWidth() { return displayWidth; }

public int sketchHeight() { return displayHeight; }

}

**Appendix B - location\_mods.java**

package processing.test.location\_mods;

import processing.core.\*;

import processing.data.\*;

import processing.event.\*;

import processing.opengl.\*;

import apwidgets.\*;

import android.text.InputType;

import android.view.inputmethod.\*;

import ketai.sensors.\*;

import ketai.ui.\*;

import android.view.MotionEvent;

import android.location.Location;

import java.util.HashMap;

import java.util.ArrayList;

import java.io.File;

import java.io.BufferedReader;

import java.io.PrintWriter;

import java.io.InputStream;

import java.io.OutputStream;

import java.io.IOException;

public class location\_mods extends PApplet {

APWidgetContainer widgetContainer;

APEditText textField;

APButton submit;

boolean b = false;

KetaiLocation location;

KetaiSensor sensor;

KetaiGesture gesture;

Location destination;

PVector locationVector = new PVector();

float compass; //(1)

PImage img1;

int h, w;

int a;

public void setup() {

img1 = loadImage("floor1.png");

image(img1, 0, 0);

img1.resize(width, height);

img1.loadPixels();

orientation(PORTRAIT);

widgetContainer = new APWidgetContainer(this); //create new container for widgets

textField = new APEditText((width/2)-(100\*width)/1200, height/10, (200\*width)/1200, (100\*height)/1920); //create a textfield from x- and y-pos., width and height

submit = new APButton(((width\*4)/5), (50\*height)/1920, "submit");

widgetContainer.addWidget(textField); //place textField in container

widgetContainer.addWidget(submit);

gesture = new KetaiGesture(this);

location = new KetaiLocation(this);

sensor = new KetaiSensor(this);

sensor.start();

orientation(PORTRAIT);

textAlign(CENTER, CENTER);

textSize((28\*width)/1200);

smooth();

}

public void draw(){

image(img1, 0, 0);

ellipse(w, h, 100, 100);

fill(0);

text("LONG PRESS THE SCREEN TO ENTER NAVIGATION MODE", width/2, (4\*height)/7);

text("PLEASE ENTER WHICH DESTINATION YOU WOULD LIKE TO TRAVEL TO", width/2, (4\*height)/7 - ((100\*height)/1920));

text("WHEN YOU ARE IN NAVIGATION MODE, PLACE DEVICE FLAT FOR BEST RESULTS", width/2, ((4\*height)/7) + ((75\*height)/1920));

text("WHEN YOU ARE IN NAVIGATION MODE, TAP SCREEN TO RETURN TO MAP", width/2, ((4\*height)/7) + ((150\*height)/1920));

//display the text in the text field

if(b) {

if(w == width/2 && h == height/2) {

widgetContainer.removeWidget(textField);

widgetContainer.removeWidget(submit);

background(78, 93, 75);

print("IT WORKS");

float bearing = location.getLocation().bearingTo(destination); //(2)

float distance = location.getLocation().distanceTo(destination);

translate(width/2, height/2); //(3)

rotate(radians(bearing) - radians(compass)); //(4)

stroke(255);

fill(0);

triangle(-width/4, 0, width/4, 0, 0, -width/2); //(5)

text((int)distance + " m", 0, 50);

text(nf(distance\*3.28084f, 0, 2) + " ft", 0, 100); //(6)

fill(0);

text("TAP TO GO BACK TO PREVIOUS SCREEN", width/2, height\*5/7);

//text("Tap to exit this screen", width/2,((3\*height)/4));

//text("Lay your device flat to achieve the best results", width/2+100,((3\*height)/4)+100);

}

}

}

public void onClickWidget(APWidget widget){

if(widget == submit){ //if it was button1 that was clicked

w = width/2;

h = height/2;

if((textField.getText().equals("restroom") == true) || (textField.getText().equals("Restroom") == true) || (textField.getText().equals("restroom ") == true) || (textField.getText().equals("Restroom ") == true)) {

destination = new Location("restroom");

destination.setLatitude(26.121337f);

destination.setLongitude(-80.147814f);

fill(82);

}

else if((textField.getText().equals("airboat") == true) || (textField.getText().equals("airboat ") == true) || (textField.getText().equals("Airboat ") == true) || (textField.getText().equals("Airboat ") == true)|| (textField.getText().equals("Airboat Ride ") == true)|| (textField.getText().equals("Airboat Ride") == true) || (textField.getText().equals("airboat ride ") == true) || (textField.getText().equals("airboat ride") == true)) {

destination = new Location("airboat ride");

destination.setLatitude(26.121357f);

destination.setLongitude(-80.148092f);

}

else if((textField.getText().equals("prehistoric florida") == true) || (textField.getText().equals("prehistoric florida ") == true) || (textField.getText().equals("Prehistoric Florida ") == true) || (textField.getText().equals("Prehistoric Florida") == true)|| (textField.getText().equals("prehistoric ") == true)|| (textField.getText().equals("prehistoric") == true) || (textField.getText().equals("Prehistoric ") == true) || (textField.getText().equals("Prehistoric") == true)) {

destination = new Location("prehistoric florida");

destination.setLatitude(26.121364f);

destination.setLongitude(-80.147918f);

}

else if((textField.getText().equals("otter viewing") == true) || (textField.getText().equals("otter viewing ") == true) || (textField.getText().equals("Otter Viewing ") == true) || (textField.getText().equals("Otter Viewing") == true)|| (textField.getText().equals("otters ") == true)|| (textField.getText().equals("otters") == true) || (textField.getText().equals("Otters ") == true) || (textField.getText().equals("Otters") == true)) {

destination = new Location("otter viewing");

destination.setLatitude(26.121404f);

destination.setLongitude(-80.147778f);

}

else if((textField.getText().equals("storm center") == true) || (textField.getText().equals("storm center ") == true) || (textField.getText().equals("Storm Center ") == true) || (textField.getText().equals("Storm Center") == true)|| (textField.getText().equals("storm ") == true)|| (textField.getText().equals("storm") == true) || (textField.getText().equals("Storm ") == true) || (textField.getText().equals("Storm") == true)) {

destination = new Location("storm center");

destination.setLatitude(26.12128f);

destination.setLongitude(-80.148053f);

}

else if((textField.getText().equals("go green") == true) || (textField.getText().equals("go green ") == true) || (textField.getText().equals("Go Green ") == true) || (textField.getText().equals("Go Green") == true)|| (textField.getText().equals("green ") == true)|| (textField.getText().equals("green") == true) || (textField.getText().equals("Green ") == true) || (textField.getText().equals("Green") == true)) {

destination = new Location("go green");

destination.setLatitude(26.121232f);

destination.setLongitude(-80.148052f);

}

else if((textField.getText().equals("discovery center") == true) || (textField.getText().equals("discovery center ") == true) || (textField.getText().equals("Discovery Center ") == true) || (textField.getText().equals("Discovery Center") == true)|| (textField.getText().equals("discovery ") == true)|| (textField.getText().equals("discovery") == true) || (textField.getText().equals("Discovery ") == true) || (textField.getText().equals("Discovery") == true)) {

destination = new Location("discovery center");

destination.setLatitude(26.121072f);

destination.setLongitude(-80.148069f);

}

else if((textField.getText().equals("florida ecoscapes") == true) || (textField.getText().equals("florida ecoscapes ") == true) || (textField.getText().equals("Florida Ecoscapes ") == true) || (textField.getText().equals("Florida Ecoscapes") == true)|| (textField.getText().equals("ecoscapes ") == true)|| (textField.getText().equals("ecoscapes") == true) || (textField.getText().equals("Ecoscapes ") == true) || (textField.getText().equals("Ecoscapes") == true)) {

destination = new Location("florida ecoscapes");

destination.setLatitude(26.121039f);

destination.setLongitude(-80.147998f);

}

else if((textField.getText().equals("elevator") == true) || (textField.getText().equals("elevator ") == true) || (textField.getText().equals("Elevator ") == true) || (textField.getText().equals("Elevator") == true)|| (textField.getText().equals("stairs ") == true)|| (textField.getText().equals("stairs") == true) || (textField.getText().equals("Stairs ") == true) || (textField.getText().equals("Stairs") == true)) {

destination = new Location("elevator");

destination.setLatitude(26.121237f);

destination.setLongitude(-80.147805f);

}

else if((textField.getText().equals("store") == true) || (textField.getText().equals("store ") == true) || (textField.getText().equals("Store ") == true) || (textField.getText().equals("Store") == true)|| (textField.getText().equals("shop ") == true)|| (textField.getText().equals("shop") == true) || (textField.getText().equals("Shop ") == true) || (textField.getText().equals("Shop") == true)) {

destination = new Location("store");

destination.setLatitude(26.120757f);

destination.setLongitude(-80.148014f);

}

else if((textField.getText().equals("help") == true) || (textField.getText().equals("Help ") == true) || (textField.getText().equals("help ") == true) || (textField.getText().equals("Help") == true)|| (textField.getText().equals("information ") == true)|| (textField.getText().equals("information") == true) || (textField.getText().equals("Information ") == true) || (textField.getText().equals("Information") == true)) {

destination = new Location("help desk");

destination.setLatitude(26.120924f);

destination.setLongitude(-80.147887f);

}

else if((textField.getText().equals("box office") == true) || (textField.getText().equals("box office ") == true) || (textField.getText().equals("Box Office ") == true) || (textField.getText().equals("Box Office") == true)|| (textField.getText().equals("tickets ") == true)|| (textField.getText().equals("tickets") == true) || (textField.getText().equals("Tickets ") == true) || (textField.getText().equals("Tickets") == true)) {

destination = new Location("box office");

destination.setLatitude(26.120822f);

destination.setLongitude(-80.147999f);

}

else if((textField.getText().equals("imax theater") == true) || (textField.getText().equals("imax theater ") == true) || (textField.getText().equals("Imax Theater ") == true) || (textField.getText().equals("Imax Theater") == true)|| (textField.getText().equals("theater ") == true)|| (textField.getText().equals("theater") == true) || (textField.getText().equals("Theater ") == true) || (textField.getText().equals("Theater") == true)) {

destination = new Location("imax theater");

destination.setLatitude(26.120784f);

destination.setLongitude(-80.147894f);

}

else if((textField.getText().equals("entrance") == true) || (textField.getText().equals("entrance ") == true) || (textField.getText().equals("Entrance ") == true) || (textField.getText().equals("Entrance") == true)) {

destination = new Location("imax theater");

destination.setLatitude(26.120884f);

destination.setLongitude(-80.147919f);

}

fill(82);

//set the smaller size

}

}

public void onLongPress(float x, float y) {

b = true;

print("HI");

}

public void onTap (float x, float y) {

println("WORKS YAY");

b = false;

setup();

}

public void onLocationEvent(Location \_location) {

println("onLocation event: " + \_location.toString());

locationVector.x = (float)\_location.getLatitude(); //(7)

locationVector.y = (float)\_location.getLongitude();

}

public void onOrientationEvent(float x, float y, float z, long time, int accuracy) { //(8)

compass = x;

// Azimuth angle between magnetic north and device y-axis, around z-axis.

// Range: 0 to 359 degrees

// 0=North, 90=East, 180=South, 270=West

}

public boolean surfaceTouchEvent(MotionEvent event) {

super.surfaceTouchEvent(event);

return gesture.surfaceTouchEvent(event);

}

}

**Appendix C - Survey.java**

package processing.test.survey;

import processing.core.\*;

import processing.data.\*;

import processing.event.\*;

import processing.opengl.\*;

import ketai.sensors.\*;

import ketai.ui.\*;

import android.view.MotionEvent;

import com.parse.Parse;

import java.util.List;

import com.parse.ParseAnalytics;

import com.parse.ParseObject;

import apwidgets.\*;

import android.text.InputType;

import android.view.inputmethod.\*;

import java.util.HashMap;

import java.util.ArrayList;

import java.io.File;

import java.io.BufferedReader;

import java.io.PrintWriter;

import java.io.InputStream;

import java.io.OutputStream;

import java.io.IOException;

public class survey extends PApplet {

APWidgetContainer widgetContainer;

APButton submit;

PImage img1;

PImage ystar;

PImage gstar;

PImage image1, image2, image3, image4, image5;

KetaiGesture gesture;

List<Record> recordList;

public void setup(){

Parse.initialize(this, "nLxtTiMaekR4RNoLdiL9DslnxN32ORLP77NW2pRh", "M97zU7eeGn2go5mwMY45OXJYVgpJxwNhoyixGvR8");

recordList = new ArrayList<Record>();

widgetContainer = new APWidgetContainer(this); //create new container for widgets

submit = new APButton(width/2, 9\*height/10, "submit");

widgetContainer.addWidget(submit);

img1 = loadImage("Survey Screen.png");

ystar = loadImage("yellow\_star.png");

gstar = loadImage("grey\_star.png");

ystar.resize(150\*(width)/1200, 150\*(height)/1920);

gstar.resize(150\*(width)/1200, 150\*(height)/1920);

ystar.loadPixels();

gstar.loadPixels();

gesture = new KetaiGesture(this);

image1 = gstar;

image2 = gstar;

image3 = gstar;

image4 = gstar;

image5 = gstar;

orientation(PORTRAIT);

}

public void draw() {

image(img1, 0, 0);

img1.resize(width, height);

img1.loadPixels();

image(image1, 100\*(width)/1200, 1565\*(height)/1920);

image(image2, 330\*(width)/1200, 1565\*(height)/1920);

image(image3, 563\*(width)/1200, 1565\*(height)/1920);

image(image4, 753\*(width)/1200, 1565\*(height)/1920);

image(image5, 1018\*(width)/1200, 1565\*(height)/1920);

text("What do you think \n " + " of this app? Rate it \n " + " down below! \n \n" + "Double tap to \n" + " set stars to 0." , width/2, 4\*height/7);

textSize(90\*(width)/1200);

textAlign(CENTER, CENTER);

}

public void onTap(float x, float y) {

if((mouseX <= 300\*width/1200) && (mouseX >= 100\*width/1200) &&(mouseY >= 1450\*height/1920) && (mouseY <= 1650\*height/1920)){

image1 = ystar;

recordList.add(new Record(1));

}

else if((mouseX <= 500\*width/1200) && (mouseX >= 300\*width/1200) &&(mouseY >= 1450\*height/1920) && (mouseY <= 1650\*height/1920)){

image1 = ystar;

image2 = ystar;

recordList.add(new Record(2));

}

else if((mouseX <= 750\*width/1200) && (mouseX >= 550\*width/1200) &&(mouseY >= 1450\*height/1920) && (mouseY <= 1650\*height/1920)){

image1 = ystar;

image2 = ystar;

image3 = ystar;

recordList.add(new Record(3));

}

else if((mouseX <= 1000\*width/1200) && (mouseX >= 800\*width/1200) &&(mouseY >= 1450\*height/1920) && (mouseY <= 1650\*height/1920)){

image1 = ystar;

image2 = ystar;

image3 = ystar;

image4 = ystar;

recordList.add(new Record(4));

}

else if((mouseX <= 1300\*width/1200) && (mouseX >= 1000\*width/1200) &&(mouseY >= 1450\*height/1920) && (mouseY <= 1650\*height/1920)){

image1 = ystar;

image2 = ystar;

image3 = ystar;

image4 = ystar;

image5 = ystar;

recordList.add(new Record(5));

}

print("YES");

}

public void onDoubleTap(float x, float y){

image1 = gstar;

image2 = gstar;

image3 = gstar;

image4 = gstar;

image5 = gstar;

}

public void onClickWidget(APWidget widget){

if(widget == submit){

saveToParse("Ratings", recordList);//if it was button1 that was clicked

//set the smaller size

}

}

public void saveToParse(String className, List<Record> recordList)

{

int i = recordList.size() -1;

ParseObject pObj = new ParseObject(className);

pObj.put("Rating", recordList.get(i).getRating());

pObj.saveInBackground();

}

public boolean surfaceTouchEvent(MotionEvent event) {

super.surfaceTouchEvent(event);

return gesture.surfaceTouchEvent(event);

}

class Record

{

int rating;

/\*\*

\* Class constructor.

\*

\* @param timestamp The timestamp in nanoseconds

\* @param x The x force in m/s^2

\* @param y The y force in m/s^2

\* @param z The z force in m/s^2

\*/

Record(int rating)

{

this.rating = rating;

}

public int getRating() {

return rating;

}

}

}

**Appendix D - Exhibit\_info.java**

package com.example.modsoffun;

import android.support.v7.app.ActionBarActivity;

import android.support.v7.app.ActionBar;

import android.support.v4.app.Fragment;

import android.os.Bundle;

import android.view.LayoutInflater;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.view.ViewGroup;

import android.os.Build;

public class Exhibit\_info extends ActionBarActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_exhibit\_info);

if (savedInstanceState == null) {

getSupportFragmentManager().beginTransaction()

.add(R.id.container, new PlaceholderFragment()).commit();

}

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.exhibit\_info, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

// Handle action bar item clicks here. The action bar will

// automatically handle clicks on the Home/Up button, so long

// as you specify a parent activity in AndroidManifest.xml.

int id = item.getItemId();

if (id == R.id.action\_settings) {

return true;

}

return super.onOptionsItemSelected(item);

}

/\*\*

\* A placeholder fragment containing a simple view.

\*/

public static class PlaceholderFragment extends Fragment {

public PlaceholderFragment() {

}

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

View rootView = inflater.inflate(R.layout.fragment\_exhibit\_info,

container, false);

return rootView;

}

}

}

**Appendix E - AboutUs.java**

package com.example.modsoffun;

import android.support.v7.app.ActionBarActivity;

import android.support.v7.app.ActionBar;

import android.support.v4.app.Fragment;

import android.os.Bundle;

import android.view.LayoutInflater;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.view.ViewGroup;

import android.os.Build;

public class AboutUs extends ActionBarActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_about\_us);

if (savedInstanceState == null) {

getSupportFragmentManager().beginTransaction()

.add(R.id.container, new PlaceholderFragment()).commit();

}

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.about\_us, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

// Handle action bar item clicks here. The action bar will

// automatically handle clicks on the Home/Up button, so long

// as you specify a parent activity in AndroidManifest.xml.

int id = item.getItemId();

if (id == R.id.action\_settings) {

return true;

}

return super.onOptionsItemSelected(item);

}

/\*\*

\* A placeholder fragment containing a simple view.

\*/

public static class PlaceholderFragment extends Fragment {

public PlaceholderFragment() {

}

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

View rootView = inflater.inflate(R.layout.fragment\_about\_us,

container, false);

return rootView;

}

}

}

**Appendix F - Screenshots**