**CONTENT**

Contents

[**USER GUIDE** 2](#_Toc440295234)

[MAIN MENU 2](#_Toc440295235)

[**CODES** 5](#_Toc440295236)

[Game\_View 5](#_Toc440295237)

[Game\_Activity 8](#_Toc440295238)

[**Main\_Menu** 9](#_Toc440295239)

[**Instruct** 11](#_Toc440295240)

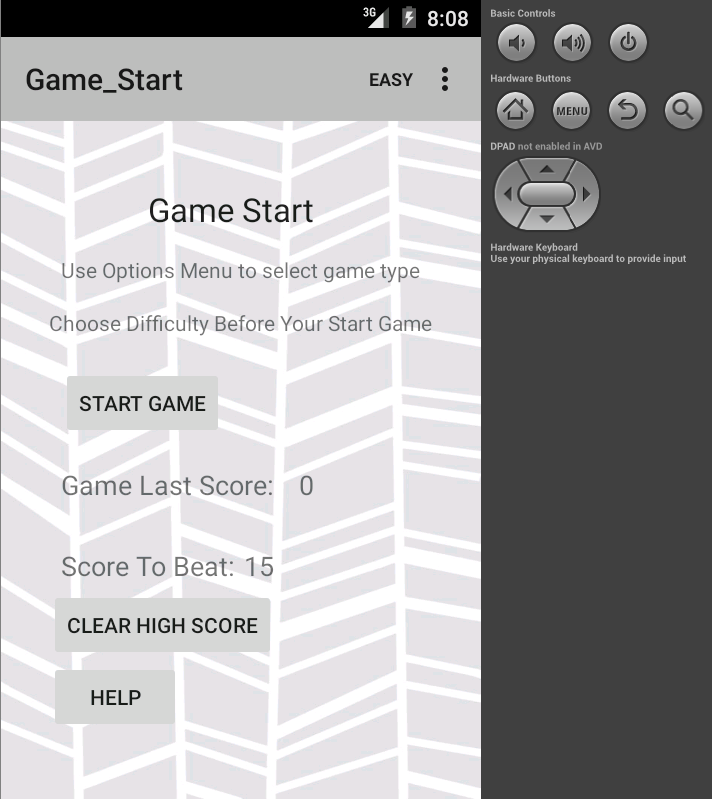
[**Sprite** 12](#_Toc440295241)

[Game\_Loop\_Thread 14](#_Toc440295242)

[**UML** 16](#_Toc440295243)

# **USER GUIDE**

## MAIN MENU



5

1

4

6

2

33

1. Press to start game.

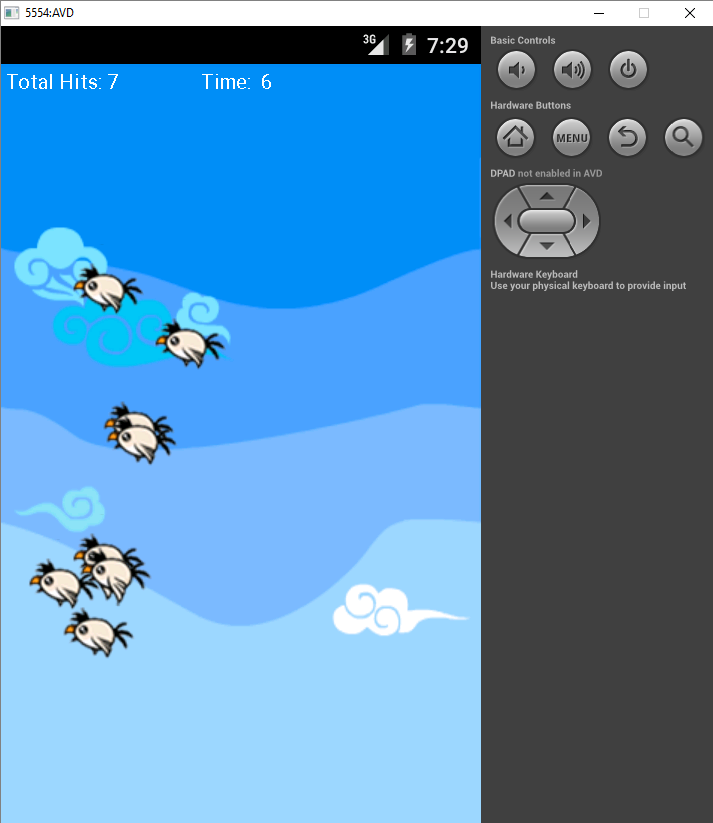
2. Press to clear high score.

3. Press to read game instructions.

4. Display score of previous game.

5. Click to select game difficulty.

6. Displays highest score to beat.

GAME ACTIVITY 

1

2

3

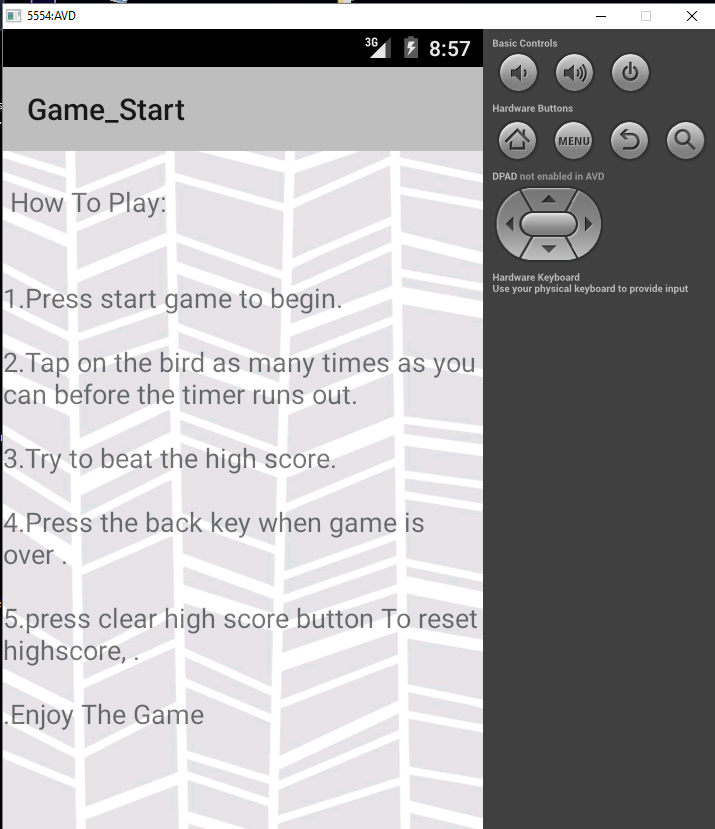
4

1. Tap on each bird to increase score.

2. Displays your total hit count.

3. Show’s time left in game.

4. Press back key (icon may differ on your device) to go back to menu.



1

Instructions for the game.

1. Press the back key to go back.

# **CODES**

## Game\_View

**package** cct.mad.lab;

**import** java.util.ArrayList;

**import** android.content.Context;

**import** android.content.Intent;

**import** android.content.SharedPreferences;

**import** android.graphics.Bitmap;

**import** android.graphics.BitmapFactory;

**import** android.graphics.Canvas;

**import** android.graphics.Color;

**import** android.graphics.Paint;

**import** android.os.Bundle;

**import** android.os.CountDownTimer;

**import** android.util.Log;

**import** android.view.MotionEvent;

**import** android.view.SurfaceHolder;

**import** android.view.SurfaceView;

**import** android.widget.TextView;

**import** android.widget.Toast;

**import** android.media.AudioManager;

**import** android.media.MediaPlayer;

**import** android.media.SoundPool;

**public** **class** GameView **extends** SurfaceView **implements** SurfaceHolder.Callback {

**int** value = 1;

SharedPreferences settings;

SharedPreferences.Editor editor;

/\* Member (state) fields \*/

**private** GameLoopThread gameLoopThread;

**private** Paint paint; //Reference a paint object

/\*\* The drawable to use as the background of the animation canvas \*/

**private** Bitmap mBackgroundImage;

**private** Sprite sprite;

**private** **int** hitCount;

/\* For the countdown timer \*/

**private** **long** startTime ; //Timer to count down from

**private** **final** **long** interval = 1 \* 1000; //1 sec interval

**private** CountDownTimer countDownTimer; //Reference to class

**private** **boolean** timerRunning = **false**;

**private** String displayTime; //To display time on the screen

**private** **boolean** gameOver;

**private** **int** highScore = 0;

**private** SoundPool sounds;

**private** **int** hit;

**public** **static** **final** String ***PREPS\_NAME*** = "Score";

**private** ArrayList<Sprite> spriteArrayList;

**private** **int** i, sprites = 8;

**public** GameView(Context context) {

**super**(context);

// Focus must be on GameView so that events can be handled.

**this**.setFocusable(**true**);

// For intercepting events on the surface.

**this**.getHolder().addCallback(**this**);

mBackgroundImage = BitmapFactory.*decodeResource*(**this**.getResources(),R.drawable.***bg***);

sounds = **new** SoundPool(10, AudioManager.***STREAM\_MUSIC***, 0);

hit = sounds.load(context, R.raw.***hit***, 1);

spriteArrayList= **new** ArrayList<Sprite>();

}

/\* Called immediately after the surface created \*/

**public** **void** surfaceCreated(SurfaceHolder holder) {

ResetGame();//Set up a new game up - could be called by a 'play again option'

gameLoopThread = **new** GameLoopThread(**this**.getHolder(), **this**);

gameLoopThread.running = **true**;

gameLoopThread.start();

mBackgroundImage = Bitmap.*createScaledBitmap*(mBackgroundImage, getWidth(), getHeight(), **true**);

}

//To initialise/reset game

**private** **void** ResetGame(){

/\* Set paint details \*/

paint = **new** Paint();

paint.setColor(Color.***WHITE***);

paint.setTextSize(20);

sprite = **new** Sprite(**this**);

**for**(i=0; i<sprites; i++){

spriteArrayList.add(**new** Sprite(**this**));

}

//Set timer

startTime = 20;//Start at 10s to count down

//Create new object - convert startTime to milliseconds

countDownTimer=**new** MyCountDownTimer(startTime\*1000,interval);

countDownTimer.start();//Start it running

timerRunning = **true**;

gameOver = **false**;

}

//This class updates and manages the assets prior to drawing - called from the Thread

**public** **void** update(){

**if** ( gameOver == **false**){

sprite.update();

**for**(i=0; i<sprites; i++){

spriteArrayList.get(i).update();

}}

}

/\*\*

\* To draw the game to the screen

\* This is called from Thread, so synchronization can be done

\*/

**public** **void** doDraw(Canvas canvas) {

canvas.drawBitmap(mBackgroundImage, 0, 0, **null**);

//Draw all the objects on the canvas

canvas.drawText("Total Hits: "+ hitCount, 5, 25, paint);

canvas.drawText("Time: " + displayTime, 200, 25, paint);

**if** (gameOver == **true**){

canvas.drawText("Game Over Total Score: " + hitCount, 200, 50, paint);

canvas.drawText("Press The Back Key To Go To Menu", 110, 75, paint);

}

//sprite.draw(canvas);

**for**(i=0; i<sprites; i++){

spriteArrayList.get(i).draw(canvas);

}

}

//To be used if we need to find where screen was touched

**public** **boolean** onTouchEvent(MotionEvent event) {

**if** (sprite.wasItTouched(event.getX(), event.getY())){

sounds.play(hit, 1.0f, 1.0f, 0, 0, 2.5f);

}

**if** (gameOver == **false**){

**if** (sprite.wasItTouched(event.getX(), event.getY())){

/\* For now, just renew the Sprite \*/

sprite = **new** Sprite(**this**);

hitCount++;

}

**for**(i=0; i<sprites; i++){

**if** (spriteArrayList.get(i).wasItTouched(event.getX(), event.getY())){

/\* For now, just renew the Sprite \*/

spriteArrayList.set(i, **new** Sprite(**this**));

hitCount++;

sounds.play(hit, 1.0f, 1.0f, 0, 0, 2.5f);

}}

**return** **true**;

}

**return** gameOver;

}

**public** **void** surfaceDestroyed(SurfaceHolder holder) {

gameLoopThread.running = **false**;

// Shut down the game loop thread cleanly.

**boolean** retry = **true**;

**while**(retry) {

**try** {

gameLoopThread.join();

retry = **false**;

} **catch** (InterruptedException e) {}

}

}

/\* Countdown Timer - private class \*/

**private** **class** MyCountDownTimer **extends** CountDownTimer {

**public** MyCountDownTimer(**long** startTime, **long** interval) {

**super**(startTime, interval);

}

**public** **void** onFinish() {

displayTime = "Games Over!";

timerRunning = **false**;

countDownTimer.cancel();

gameOver = **true**;

}

**public** **void** onTick(**long** millisUntilFinished) {

displayTime = " " + millisUntilFinished / 1000;

}

}//End of MyCountDownTimer

**public** **int** getHitCount() {

**return** hitCount ;

}

**public** **void** surfaceChanged(SurfaceHolder holder, **int** format, **int** width, **int** height) {

}

}

## Game\_Activity

package cct.mad.lab;

import android.os.Bundle;

import android.app.Activity;

import android.content.Context;

import android.content.Intent;

import android.content.SharedPreferences;

import android.content.SharedPreferences.Editor;

import android.view.Window;

import android.widget.LinearLayout;

import android.widget.RelativeLayout;

/\* Activity class for the game.

\* Creates the GameView and can pass parameters to the gameView.

\* Can also receive/return Intent data to configure game and report game status(eg score)

\*/

public class GameActivity extends Activity {

GameView gameView;// Reference the gameView

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

// remove title bar

requestWindowFeature(Window.FEATURE\_NO\_TITLE);

// Set the layout

setContentView(R.layout.game\_view\_container);

// Get the id of the layout

RelativeLayout mainscreen = (RelativeLayout) findViewById(R.id.mainscreen);

// Make the GameView

gameView = new GameView(this);

// Get data from intent and config gameView here

gameView.setLayoutParams(new LinearLayout.LayoutParams(

LinearLayout.LayoutParams.MATCH\_PARENT,

LinearLayout.LayoutParams.MATCH\_PARENT));

// Add GameView

mainscreen.addView(gameView);

}

/\* Called when activity is done and should be closed.

\* The ActivityResult is propagated back to whoever launched via onActivityResult()

\*/

public void finish(){

Intent returnIntent = new Intent();

System.out.println("LLL: "+gameView.getHitCount());

returnIntent.putExtra("GAME\_SCORE",gameView.getHitCount());

setResult(RESULT\_OK, returnIntent);

super.finish();

}

}

## **Main\_Menu**

**package** cct.mad.lab;

**import** android.os.Bundle;

**import** android.util.Log;

**import** android.app.Activity;

**import** android.content.Context;

**import** android.content.Intent;

**import** android.graphics.Bitmap;

**import** android.graphics.BitmapFactory;

**import** android.content.\*;

**import** android.media.MediaPlayer;

**import** android.view.Menu;

**import** android.view.MenuItem;

**import** android.view.SurfaceHolder;

**import** android.view.View;

**import** android.widget.Button;

**import** android.widget.EditText;

**import** android.widget.TextView;

**import** android.widget.Toast;

**import** android.media.MediaPlayer;

**import** android.view.View;

**import** android.view.View.OnClickListener;

**public** **class** MainMenu **extends** Activity {

**public** **static** **final** String ***PREPS\_NAME*** = "Score";

SharedPreferences settings;

SharedPreferences.Editor editor;

**int** high;

**int** gameScore;

MediaPlayer sound;

Button button;

**private** **static** **final** **int** ***SCORE\_REQUEST\_CODE*** = 1;// The request code for the intent

GameActivity gameActivity;

GameView gameView;

TextView tvScore;

String score;

Intent gameIntent;

TextView highScore;

Bitmap mBackgroundImage;

**static** **int** *xSpeed*;

**static** **int** *ySpeed*;

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

settings = getSharedPreferences(***PREPS\_NAME***, 0);

setContentView(R.layout.***game\_start***);

tvScore = (TextView)findViewById(R.id.***tvOutput***);

highScore =(TextView)findViewById(R.id.***textView3***);

high = settings.getInt("highscore", 0);

gameScore = 0;

highScore.setText(Integer.*toString*(high));

sound = MediaPlayer.*create*(MainMenu.**this**, R.raw.***music***);

sound.setLooping(**true**);

sound.start();

addListenerOnButton();

mBackgroundImage = BitmapFactory.*decodeResource*(**this**.getResources(),R.drawable.***background***);

}

**public** **void** onCreate1(Bundle savedInstanceState) {

}

**private** **void** addListenerOnButton() {

**final** Context context=**this**;

button = (Button) findViewById(R.id.***button3***);

button.setOnClickListener(**new** OnClickListener(){

**public** **void** onClick(View v) {

Intent intentMain = **new** Intent(MainMenu.**this** ,

Instruct.**class**);

MainMenu.**this**.startActivity(intentMain);

Log.*i*("Content "," Main layout ");

}

});

}

**public** **void** startGame(View v){

gameIntent = **new** Intent(**this**, GameActivity.**class**);

startActivityForResult(gameIntent, ***SCORE\_REQUEST\_CODE*** );

}

/\* Create Options Menu \*/

**public** **boolean** onCreateOptionsMenu(Menu menu) {

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

getMenuInflater().inflate(R.menu.***main\_menu***, menu);

**return** **true**;

}

// Respond to item selected on OPTIONS MENU

**public** **boolean** onOptionsItemSelected(MenuItem item) {

// Handle item selection

**switch** (item.getItemId()) {

//put data in Intent

**case** R.id.***easy***:

*xSpeed* = 5;

*ySpeed* = 3;

Toast.*makeText*(**this**, "Easy Selected!", Toast.***LENGTH\_SHORT***).show();

**return** **true**;

**case** R.id.***medium***:

*xSpeed* = 10;

*ySpeed* = 7;

Toast.*makeText*(**this**, "Medium Selected!", Toast.***LENGTH\_SHORT***).show();

**return** **true**;

**case** R.id.***hard***:

*xSpeed* = 20;

*ySpeed* = 17;

Toast.*makeText*(**this**, "Hard Selected!", Toast.***LENGTH\_SHORT***).show();

**return** **true**;

**default**:

**return** **super**.onOptionsItemSelected(item);

}

}

**protected** **void** onActivityResult(**int** requestCode, **int** resultCode, Intent retIntent) {

// Check which request we're responding to

**if** (requestCode == ***SCORE\_REQUEST\_CODE***) {

// Make sure the request was successful

**if** (resultCode == ***RESULT\_OK***) {

**if** (retIntent.hasExtra("GAME\_SCORE")) {

gameScore = retIntent.getExtras().getInt("GAME\_SCORE");

tvScore.setText(gameScore+"");

**if**(gameScore > high) {

editor = settings.edit();

editor.putInt("highscore", gameScore);

editor.commit();

Toast.*makeText*(MainMenu.**this**,"Congratulations! You Have Bet The Highscore " + high + " New Highscore " + gameScore + "\n", Toast.***LENGTH\_LONG***).show();

high = settings.getInt("highscore", 0);

highScore.setText(Integer.*toString*(high));

Toast.*makeText*(MainMenu.**this**,"High Score " + high, Toast.***LENGTH\_LONG***).show();

}

}

}

}

}

**public** **void** clearPreferences(View V) {

editor = settings.edit();// Create a new editor

editor.clear();

editor.commit();

highScore.setText("0");

Toast.*makeText*( MainMenu.**this**,"High Scoree Has Been Reset!",Toast.***LENGTH\_LONG***).show();

}

**public** **static** **int** setXSpeed(){

**return** *xSpeed*;

}

**public** **static** **int** setYSpeed(){

**return** *ySpeed*;

}

}

## **Instruct**

**package** cct.mad.lab;

**import** android.app.Activity;

**import** android.os.Bundle;

**import** android.widget.Button;

**import** android.widget.TextView;

**public** **class** Instruct **extends** Activity {

Button button;

TextView help;

**public** **void** onCreate(Bundle savedInstancestate){

**super**.onCreate(savedInstancestate);

setContentView(R.layout.***help***);

help =(TextView) findViewById (R.id.***textView1***);

help.setText("\n How To Play:\n \n \n1.Press start game to begin.\n \n2."

+ "Tap on the bird as many times as you can before the timer runs out.\n \n3."

+ "Try to beat the high score.\n \n4."

+ "Press the back key when game is over .\n \n5."

+ "press clear high score button To reset highscore, . \n \n."

+ "Enjoy The Game");

}

}

## **Sprite**

**package** cct.mad.lab;

**import** java.util.Random;

**import** android.graphics.Bitmap;

**import** android.graphics.BitmapFactory;

**import** android.graphics.Canvas;

**import** android.graphics.Rect;

**public** **class** Sprite {

// Needed for new random coordinates.

**private** Random random = **new** Random();

//x,y position of sprite - initial position (0,50)

**private** **int** x = random.nextInt(300)+1;

**private** **int** y = random.nextInt(200)+1;

**static** **int** *xSpeed* = MainMenu.*setXSpeed*();//Horizontal increment of position (speed)

**static** **int** *ySpeed* = MainMenu.*setYSpeed*();// Vertical increment of position (speed)

// READ FROM dhared prefreemce

//xSpeed = (xSpeed \* "what u read from shared preference");

// ySpeed = (ySpeed \* "what u read from shared preference");

**private** GameView gameView;

**private** Bitmap spritebmp;

//Width and Height of the Sprite image

**private** **int** bmp\_width;

**private** **int** bmp\_height;

**private** **static** **final** **int** ***ANIMATION\_ROWS*** = 2;

**private** **static** **final** **int** ***ANIMATION\_COLUMNS*** = 8;

**private** **int** currentFrame = 0;

**private** **int** frameCount = 8;

**private** **long** lastFrameChangeTime = 0;

**private** **int** frameLengthInMilliseconds = 100;

**public** Sprite(GameView gameView) {

**this**.gameView=gameView;

spritebmp = BitmapFactory.*decodeResource*(gameView.getResources(),

R.drawable.***birdanimation***);

**this**.bmp\_width = spritebmp.getWidth()/***ANIMATION\_COLUMNS***;

**this**.bmp\_height= spritebmp.getHeight()/***ANIMATION\_ROWS***;

}

//update the position of the sprite

**public** **void** update() {

x = x + *xSpeed*;

y = y + *ySpeed*;

wrapAround(); //Adjust motion of sprite.

}

**public** **void** getCurrentFrame(){

**long** time = System.*currentTimeMillis*();

**if** ( time > lastFrameChangeTime + frameLengthInMilliseconds) {

lastFrameChangeTime = time;

currentFrame ++;

**if** (currentFrame >= frameCount) {

currentFrame = 0;

}

}

}

**public** **void** draw(Canvas canvas) {

**int** srcX = currentFrame \* bmp\_width;//frame - x direction

**int** srcY; //row

**if** (*xSpeed* > 0){//Sprite going right; row = 0

srcY = 0 \* bmp\_height;

}

**else** { //Going left; row = 1+

srcY = 1 \* bmp\_height;

}

//Create Rect around the source image to be drawn

Rect src = **new** Rect(srcX, srcY, srcX+bmp\_width, srcY + bmp\_height);

//Rect for destination image

Rect dst = **new** Rect(x, y, x + bmp\_width, y + bmp\_height);

//draw the image frame

getCurrentFrame();

canvas.drawBitmap(spritebmp, src, dst, **null**);

}

**public** **void** wrapAround(){

//Code to wrap around

**if** (x < 0) x = x + gameView.getWidth(); //increment x whilst not off screen

**if** (x >= (gameView.getWidth() - bmp\_width)){ //if gone of the right sides of screen

*xSpeed* = -MainMenu.*setXSpeed*(); //Reset x

}

**if** (x + *xSpeed* < 0){

*xSpeed* = MainMenu.*setXSpeed*();

}

**if** (y < 0) y = y + gameView.getHeight();//increment y whilst not off screen

**if** (y >=( gameView.getHeight()- bmp\_height)){//if gone of the bottom of screen

*ySpeed* = -MainMenu.*setYSpeed*();//Reset y

}

**if** (y + *ySpeed* < 70){

*ySpeed* = MainMenu.*setYSpeed*();

}

}

/\* Checks if the Sprite was touched. \*/

**public** **boolean** wasItTouched(**float** ex, **float** ey){

**boolean** touched = **false**;

**if** ((x <= ex) && (ex < x + bmp\_width) &&

(y <= ey) && (ey < y + bmp\_height)) {

touched = **true**;

}

**return** touched;

}//End of wasItTouched

}

## Game\_Loop\_Thread

**package** cct.mad.lab;

**import** android.graphics.Canvas;

**import** android.view.KeyEvent;

**import** android.view.SurfaceHolder;

/\*\*

\* Game loop thread. This loop is the main driver for the

\* Game and calls methods of Game object such as update and draw.

\*

\*/

**public** **class** GameLoopThread **extends** Thread {

**private** **final** **static** **int** ***FPS*** = 60; // How many times per second the game should be updated, drawn?

**private** **final** **static** **int** ***MAX\_FRAME\_SKIPS*** = 5; // Maximum number of frames to be skipped

**private** **final** **static** **int** ***FRAME\_PERIOD*** = 100 / ***FPS***; // The time for one frame

// Surface holder that can access the physical surface.

**private** SurfaceHolder surfaceHolder;

**private** GameView gameView;//To link to GameView

// Holds the state of the game loop.

**public** **boolean** running;

**public** GameLoopThread(SurfaceHolder surfaceHolder, GameView gameView) {

**super**();

**this**.surfaceHolder = surfaceHolder;

**this**.gameView = gameView; // To link to GameView

}

@Override

**public** **void** run() {

Canvas canvas;

**long** startTime; // the time when the cycle begun

**long** timeDiff; // the time it took for the cycle to execute

**int** sleepTime; // ms to sleep (<0 if we're behind)

**int** framesSkipped; // number of frames being skipped

sleepTime = 0;

**while**(running){

canvas = **null**;

**try** {

canvas = **this**.surfaceHolder.lockCanvas(); // Try locking the canvas for exclusive pixel editing in the surface.

**synchronized**(surfaceHolder) {

startTime = System.*currentTimeMillis*();//of canvas locked

framesSkipped = 0; // Resetting the frames skipped.

gameView.update();//Update the assets

gameView.doDraw(canvas);//Draw on the surface

/\* SYNCHRONISE THE LOOP CYCLE WITH THE FRAME PERIOD \*/

// First calculate how long the drawing/frame took.

timeDiff = System.*currentTimeMillis*() - startTime;

//Calculate the time to sleep to synch with frame time(period)

sleepTime = (**int**)(***FRAME\_PERIOD*** - timeDiff);

**if** (sleepTime > 0) {//Loop going faster than Frame, so sleep

**try** {

// Send the thread to sleep for a short period.

Thread.*sleep*(sleepTime);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

**while** (sleepTime < 0 && framesSkipped < ***MAX\_FRAME\_SKIPS***) {

// We need to catch up, so we update without drawing the game to the screen.

sleepTime += ***FRAME\_PERIOD***; // Add FRAME\_PERIOD to check while condition again.

framesSkipped++;

}

}

} **catch**(Exception e) {

e.printStackTrace();

} **finally** {

// In case of an exception the surface is not left in an inconsistent state.

**if** (canvas != **null**) {

surfaceHolder.unlockCanvasAndPost(canvas);//All in synch - unlock canvas

}

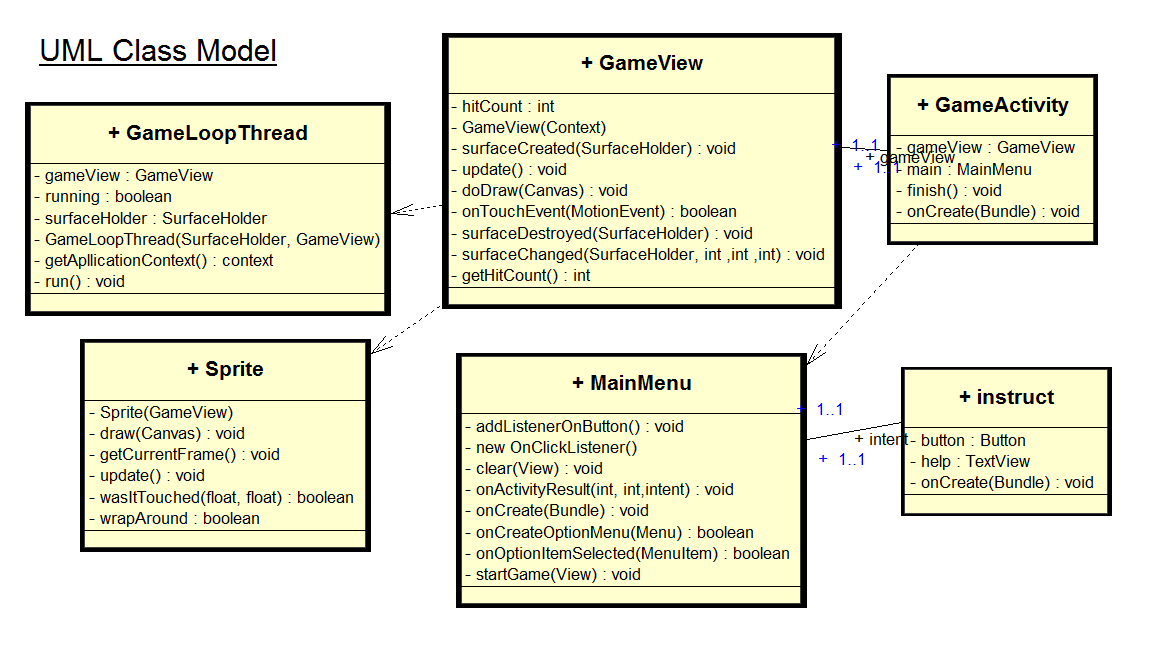
}

}

}

}

# **UML**

****