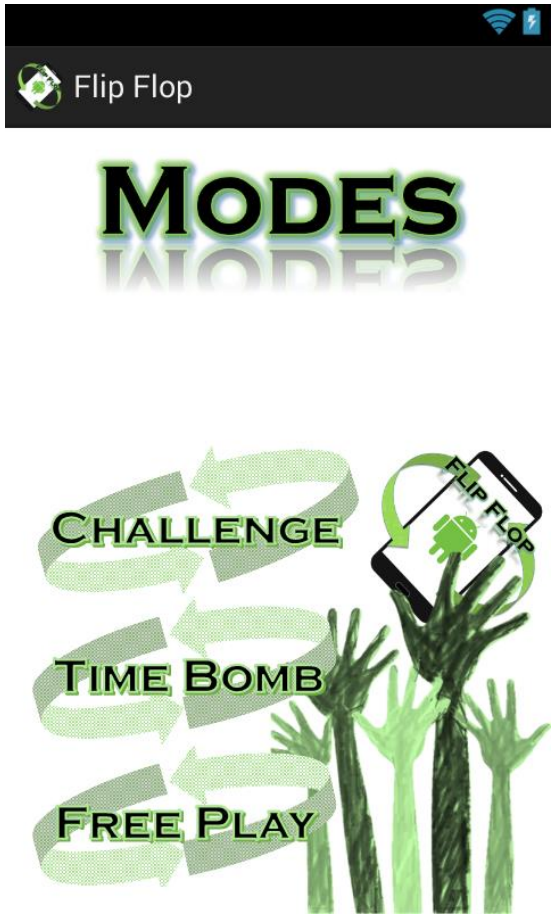
The Next Top 10 Android App

Flip Flop Description

- A super fun game based on Android
- The goal of the game is to flip one's phone in their hands as many times and as fast one can within a given period of time with the added bonus that higher flips in the air will be accompanied with a multiplier onto the score of that flip
- We're all proud of the work we've put in so far and we hope to release the finished product onto the Google Play Store

Interface


Flip Flop
Flip Flop
Play
Scores
Help
Options


Flip Flop
Flip Flop
Challenge
Time Bomb
Free Play


Flip Flop
Instructions
How to play? Spin your phone... That's pretty much it... The more spins, the higher your score.
Back
Next

Interface


Flip Flop
Time Bomb
Begin in

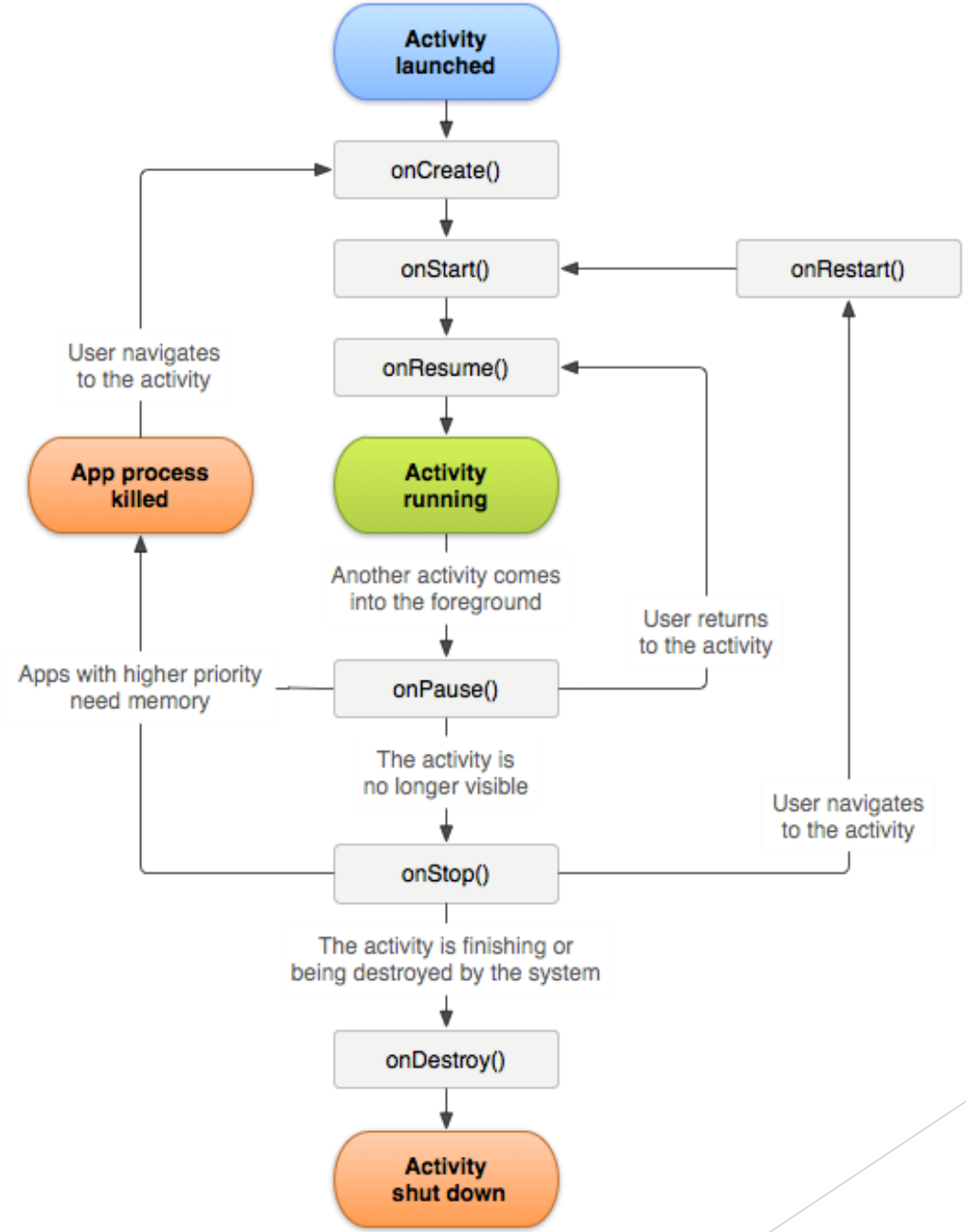

Flip Flop
Challenge
Challenge 1
Challenge 2
Challenge 3


Flip Flop
Free Play
Begin in

Music

Options

Sound: ON



Music (create service)

```
private ServiceConnection Scon = new ServiceConnection() {

    public void onServiceConnected(ComponentName name, IBinder binder) {
        mServ = ((MusicService.ServiceBinder) binder).getService();
    }

    public void onServiceDisconnected(ComponentName name) {
        mServ = null;
    }
};

    void doBindService(){
        bindService(new Intent(this,MusicService.class),
                Scon,Context.BIND_ADJUST_WITH_ACTIVITY);
        mIsBound = true;
    }

    public void doUnbindService()
    {
        if(mIsBound)
        {
            unbindService(Scon);
            stopService(music);
            mIsBound = false;
            continueMusic = false;
        }
```

```
<service
    android:name="edu.astate.cs.MusicService"
    android:enabled="true" >
</service>
```

Music Synchronized

```
public void options(View view) {
    if (mIsBound)
    {
        continueMusic = true;
    }
    else
    {
        continueMusic = false;
    }
    Intent viewIntent = new Intent(this, Options.class);
    viewIntent.putExtra("boolean", continueMusic);
    startActivity(viewIntent);
}

@Override
public boolean onKeyDown(int keyCode, KeyEvent event){
    if(keyCode == KeyEvent.KEYCODE_BACK) {

            Intent Act2Intent = new Intent(this, PlayOptions.class);
            if (Home.mIsBound)
                continueMusic = true;
            else
                continueMusic = false;
            Act2Intent.putExtra("boolean", continueMusic);
            startActivity(Act2Intent);

            finish();
            return true;
    }
    return false;
}
```

Music Synchronized

```
@Override
protected void onResume()
{
    super.onResume();
    if(Home.mIsBound == true)
    {
        continueMusic = false;
        Home.mServ.resumeMusic();
    }

}

@Override
public void onPause ()
{
    super.onPause();
    if(!continueMusic)
    {
        Home.mServ.pauseMusic();
    }
}
```

```
@Override
protected void onStop ()
{
    super.onStop();
    if(!continueMusic)
    {
        Home.mServ.pauseMusic();
    }
}

@Override
protected void onRestart ()
{
    super.onRestart();

    if(Home.mIsBound == true)
    {
        continueMusic = false;
        Home.mServ.resumeMusic();
    }

}
```

High Scores (SQLite database)

Database Implementation

- Each game mode has own database of scores assisted by OpenHelper classes
  - OpenHelpers create tables as well as add, delete, and return values from the databases themselves.
- Use simple Select-From-Where queries of the SQLite database to populate "ListView"
- After user finishes playing, score inserted into the database
- Home class gets instance of each database, provides get/set methods

Database (OpenHelper)

```java
package edu.astate.cs;

import java.util.ArrayList;

public class FreePlayOpenHelper extends SQLiteOpenHelper {

    private static final String DATABASE_NAME = "FPdb";
    private static final int DATABASE_VERSION = 1;
    private static final String FP_TABLE_NAME = "FreePlay";
    private static final String KEY_SCORE = "score";
    private SQLiteDatabase db = this.getWritableDatabase(); //get actual instance of DB




    FreePlayOpenHelper (Context context)
    {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
        //Call super constructor, pass database name, version
    }

    //oncreate will create tables inside database
    @Override
    public void onCreate(SQLiteDatabase db)
    {
        String FP_TABLE_CREATE =
                "CREATE TABLE if not exists FreePlay (" +
                 "score INTEGER );"; //create fpTable if not there already


        db.execSQL(FP_TABLE_CREATE);
    }
```

Database (OpenHelper)

```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + FP_TABLE_NAME);
        onCreate(db); //create new table
}

//add new score to DB
public void addScore (int newScore)
{


    ContentValues vals = new ContentValues();
    vals.put(KEY_SCORE, newScore); //put score into value container

    //Insert Row into Database
    db.insert(FP_TABLE_NAME, null, vals); //insert into table


}



//use to get list of all scores in database, sorted in string format
public List<String> getAllScores ()
{

    List<Integer> actualScores = new ArrayList<Integer>();
    //Select All Query, Sort by Score
    String selectQuery = "Select * From " + FP_TABLE_NAME +
                         " ORDER BY " + KEY_SCORE + " DESC";
```

Database (OpenHelper)

```
Cursor cursor = db.rawQuery(selectQuery, null);
//use cursor object to store queried results

//loop through each row and them to list

if (cursor.moveToFirst())
{
    do{
        actualScores.add(Integer.parseInt(cursor.getString(0)));


    } while (cursor.moveToNext()); //while more rows to parse
}

List<String> stringList = new ArrayList<String>();
for (Integer i: actualScores)
{
    stringList.add(i.toString());
} //convert integer values to string, used in listview


//return scoresList;
return stringList; //return list of strings
```

Options

- Uses toggleButton to turn Sound (ON/OFF)
  - Buttons for ON/OFF are custom images, bound in separate xml file


```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android" >

    <!-- When selected, use grey -->
    <item android:drawable="@drawable/on"
          android:state_checked="true" />
    <!-- When not selected, use white -->
     <item android:drawable="@drawable/off"
          android:state_checked="false" />

</selector>
```

Options (continued)

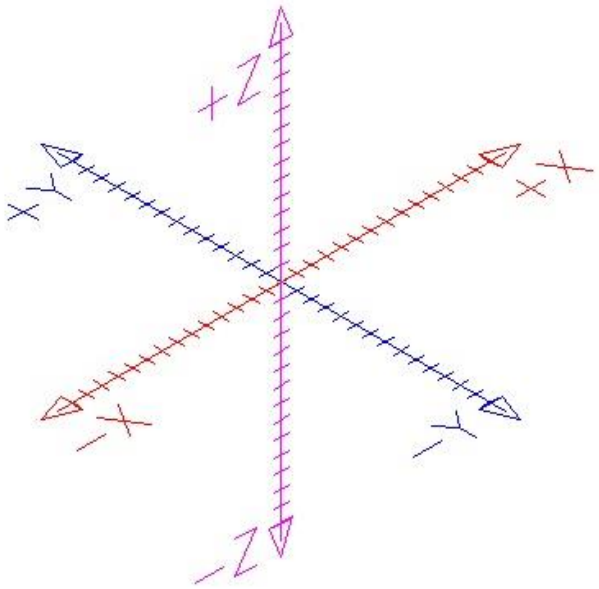
- Toggle button used to turn music service on or off

```
public void soundToggleClick (View view)
{
    // boolean on = ((ToggleButton) view).isChecked();

    if (tg.isChecked() == true) {
        // resume music from where you paused it
        Home.mServ.resumeMusic();
        Home.mIsBound = true;
    } else {
        // pause media player from main activity
        Home.mServ.pauseMusic();
        Home.mIsBound = false;
    }
}
```

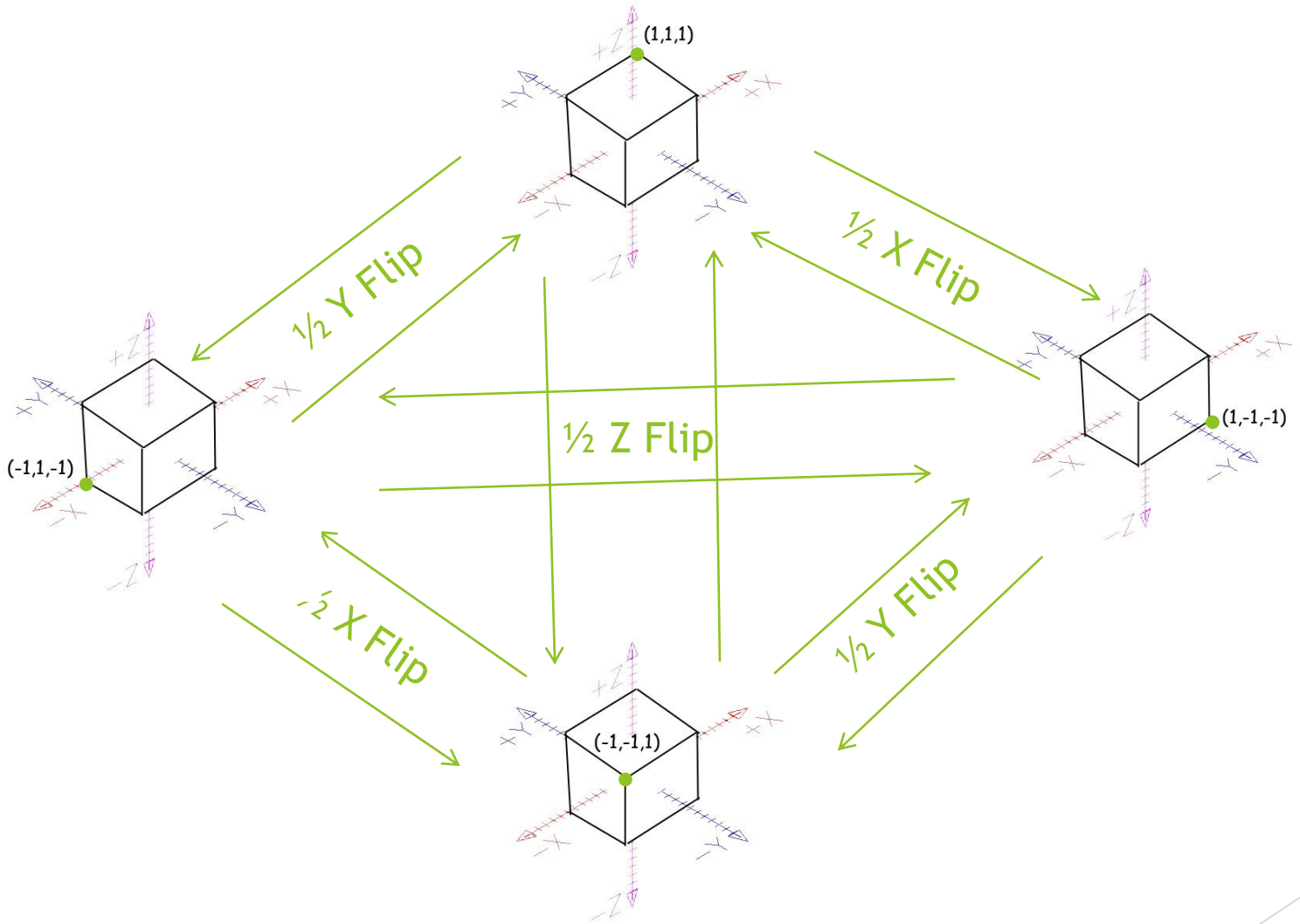
Scoring / Sensor Implementation

- Original Plan: Capture the exact number of flips
  - Gyroscope



X	Y	Z
+	+	+
+	+	-
+	-	+
+	-	-
-	+	+
-	+	-
-	-	+
-	-	-

Scoring / Sensor Implementation



Scoring / Sensor Implementation

- Hardware Limitations
- New approach: Capture acceleration in each direction
  - Accelerometer
- Score = (Σ ΔxAccel + Σ ΔyAccel) * zMultiplier

Scoring / Sensor Implementation (Code)

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    .
    .
    mSensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
    mAccelSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
}

@Override
protected void onResume() {
    .
    .
    mSensorManager.registerListener(this, mAccelSensor, 1);
}

@Override
public void onPause () {
    .
    .
    mSensorManager.unregisterListener(this);
}

@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {
    // Required for the interface, does nothing in our program
}
```

Scoring / Sensor Implementation (Code)

```
@Override
public void onSensorChanged(SensorEvent event) {
    if (event.sensor.getType()==Sensor.TYPE_ACCELEROMETER)
    {
        float xA = event.values[0];
        float yA = event.values[1];
        float zA = event.values[2];
        zA = zA-9.81f;

        double deltaX = xA - xPrevAccel; //same for deltaY and deltaZ

        xPrevAccel = xA; //same for yPrev and zPrev

        deltaX = Math.abs(deltaX); //same for deltaY and deltaZ

        if (deltaX > 1) //same for deltaY/yMax
            xMaxAccel += deltaX;

        if (deltaZ < .7)
            zCurAccel = 0;

        zCurAccel += deltaZ;

        if (zCurAccel > zMaxAccel)
        {
            zMaxAccel = zCurAccel;
        }

        mult = (int) ((zMaxAccel / 50) + 1);
        total = (int)(mult*(xMaxAccel + yMaxAccel));
    }
}
```

Thank you for your attention!

Presentation by:

Biljana Miloshevska

Michael Morton

Spencer Davis