



שם בית הספר : אורט שחקים

שם העבודה : משחק פלאפי בירד

שם התלמיד : רפאל רייכרודל

ת"ז : 215059916

שם המנחה : יצחק ביליה

שם החלופה : פרויקט גמר בסביבת אנדרואיד

תאריך ההגשה : 04.06.23

תוכן עניינים

3	מבוא.....
3	רקע.....
3	תהליך המחקר.....
3	אתגרים מרכזיים.....
4	חידושים, התאמות ועדכונים של אלמנטים טכנולוגיים, עיצוביים ואחרים בפרויקט.....
5	תיאור תחום הידע.....
5	אובייקטים נחוצים.....
5	סוגי נתונים.....
5	ייצוג מידע.....
5	תיאור פעולות על המידע.....
6	מבנה / ארכיטקטורה.....
6	תכנון ותיעוד מסכי הפרויקט.....
14	תרשים היררכיית ומעברי המסכים, ותרשים תיאור מחלקות הפרויקט.....
16	מימוש הפרויקט.....
46	מדריך למשתמש.....
46	דרישות סף.....
47	הצגת המשחק.....
61	רפלקציה.....
62	ביבליוגרפיה.....
64	נספחים.....

מבוא

רקע

הפרויקט שלי הינו בעצם חיקוי של המשחק "Flappy Bird" שהיה מאוד פופולרי בסביבות 2013-2014. הנושא נבחר משום שבתור חובבן משחקים להוט רציתי ליצור וריאציה משלי למשחק ובנוסף לטבול את האצבעות שלי בפיתוח משחקים בסביבת Android Studio, שכן המשחק המקורי נכתב בסביבת Unity. קהל היעד הינו כל אדם שאוהב לשחק במשחקים בפלאפון.

תהליך המחקר

לצורך תחילת העבודה על הפרויקט קודם הייתי צריך בעצם להבין – כיצד ניתן לפתח משחקים ב-Android Studio? כמובן לאחר שידעתי כיצד המשחק המקורי מתנהל בתור שחקן הייתי צריך להבין מה קורה מאחורי הקלעים בתור מפתח. יצאתי למחקר מעמיק לגבי מגוון כלים שיכולים לעזור לי בנדון – בין אם אלה פונקציות חשובות או קונספטים של פיתוח משחקים לאנדרואיד, למיניהם. להלן טכנולוגיות שחקרתי, אשר אינן חלק מתוכנית הלימודים:

Bitmap, Rect, Matrix, draw(Canvas canvas), Handler, Runnable

Bitmap: Bitmap הוא אובייקט ב-Android Studio המייצג רשת מלבנית של פיקסלים, שניתן להשתמש בו כדי לאחסן ולתפעל תמונות. הוא משמש בדרך כלל להצגת תמונות ביישומי אנדרואיד.

Rect: Rect (קיצור של Rectangle) היא מחלקה ב-Android Studio המייצגת קואורדינטות וממדים של מלבן. היא משמשת לעתים קרובות להגדרת גבולות, מיקום אלמנטים או ביצוע זיהוי התנגשות ביישומים גרפיים.

Matrix: Matrix הוא אובייקט ב-Android Studio המשמש לביצוע טרנספורמציות שונות על אובייקטים גרפיים. ניתן להשתמש בו כדי לתרגם, לסובב, לשנות קנה מידה או להטות תמונות או אלמנטים גרפיים אחרים.

draw(Canvas canvas): draw היא פונקציה באובייקטים הגרפיים של Android Studio המאפשרת רינדור ("ציור") על המסך. הפרמטר canvas מסוג Canvas מייצג את משטח הציור עליו מוצג האובייקט שברצוננו לרנדור.

Handler: Handler היא מחלקה המאפשרת לתזמן ולהפעיל קוד ב-thread ספציפי. זה מספק דרך לתזמן הודעות או ריצה לעיבוד במועד מאוחר יותר.

Runnable: Runnable הוא ממשק ב-Android Studio המייצג מקטע קוד שניתן להפעיל. הוא משמש לעתים קרובות בשילוב עם Handler כדי לתזמן משימות לרוץ על thread נפרד או בזמן מסוים.

אתגרים מרכזיים

פיזיקת המשחק: יישום פיזיקה מציאותית עבור תנועת הציפור, כוח המשיכה וזיהוי התנגשות היה מאתגר. להבטיח שתנועת הציפור תרגיש טבעית ומגיבה דרשה חישוב וטיפול זהירים במהירויות וכד'.

אירועי מגע: טיפול במחוות מגע ומיפוי שלהן לפעולות הציפור דרש טיפול מדויק באירוע ותיאום עם היגיון המשחק.

הצבת מכשולים: המכשולים (צינורות) נוצרים באופן דינמי וזזים במהירות קבועה שמאלה, מה שהיה מורכב לפתח. יצירת חווית משחק חלקה ומאתגרת כרוכה בתכנון קפדני של אלגוריתמים כדי ליצור ולמקם מכשולים תוך הבטחה שהם לא חופפים או יוצרים מצבי משחק לא הוגנים.

לולאת משחק ואופטימיזציה: שמירה על קצב פריימים חלק ועקבי חיונית לחוויית משחק טובה. פיתוח לולאת משחק יעילה, אופטימיזציה של ביצוע הקוד, מזעור השימוש בזיכרון וטיפול בפעולות עתירות משאבים (כגון טעינה ורינדור גרפיקה) חיוניים להשגת ביצועים מיטביים.

מנגנון הניקוד וסיום המשחק: טיפול בהתנגשויות, מעקב אחרי התקדמות המשחק ואחר הניקוד של השחקן היו מאתגרים. קביעה מתי המשחק אמור להסתיים, הצגת המשחק עצמו, ושמירה על היגיון ניקוד מדויק דרשו שיקול ובדיקה מדוקדקים.

גרפיקה: מציאת גרפיקה מתאימה למשחק, כולל הציפור, הרקע והמכשולים, לקח זמן. להבטיח שהגרפיקה מושכת מבחינה ויזואלית, בקנה מידה נכון ותואמת לגדלים ורזולוציות מסך שונות היווה אתגר.

בדיקה ואיתור באגים: בדיקה יסודית של המשחק כדי לזהות ולתקן באגים, תקלות או בעיות ביצועים הייתה חלק חיוני, חשוב ומשמעותי מתהליך הפיתוח.

חידושים, התאמות ועדכונים של אלמנטים טכנולוגיים, עיצוביים ואחרים בפרויקט
בנוסף למשחק עצמו ישנם מספר דברים שהוספתי (בעיקר כחלק מדרישות משרד החינוך לפרויקט שהייתי צריך לענות עליהם): מערכת יצירת חשבונות והתחברות על מנת לשמור ולשלוח high scores בין משתמשים שונים, שינוי הרקע לתמונה של סוללה חלשה כאשר אחוז הסוללה של המשתמש הוא מתחת לאחוז מסוים, מוזיקת רקע, סאונד אפקטים, אינטגרציה ל-Google Maps API כך שישנה את תמונת הרקע ל-Street View של המקום הנוכחי של המשתמש בלחיצה על כפתור ושינוי תמונת הציפור במשחק לתמונה של מטוס כאשר המשתמש מפעיל מצב טיסה. כל אלה דרשו מחקר נוסף – שהיה מעמיק ומהנה לשלב בפרויקט.

תיאור תחום הידע

אובייקטים נחוצים

בעיקרון, אנו צריכים ציפור, צינורות ומערכת שתאפשר לנו לצייר את הציפור והצינורות על המסך, בנוסף ללוגיקה כמו הזזת הצינורות או collision detection.

בכמעט כל משחק יש שימוש במשהו שקוראים לו hitbox – צורה גיאומטרית שמקיפה איזה שהוא אובייקט, כדי לוודא מתי אותו אובייקט יוצר מגע עם אובייקט אחר.

לדוגמה, Super Mario Bros. (1985) :



במקרה שלנו יש לנו ציפור וצינורות אותם ניתן להקיף במלבנים, והקביעה של הלוגיקה שלהם היא לפי שיעורי ה-x ו-y הנוכחיים שלהם על המסך.

סוגי נתונים

אזכור כבר קודם לכן : Bitmap, Rect, Matrix, draw(Canvas canvas), Handler, Runnable. אלה הכלים העיקריים בהם נעזרתי כדי ליצור בסיס המשחק עצמו.

יש גם שימוש ב-ArrayList ששומר Bitmaps או את הצינורות (ArrayList<Bitmap>), סוג זה של נתונים מאפשר גישה ומעבר קלים על כל אחד מהתמונות בתוכו. (ArrayList<Pipe>).

ייצוג מידע

הציפור והצינורות מיוצגים כתמונות זזות על המסך, באמצעות Bitmap ו-Rect. הם "מצוירים" על המסך בצורה מתמדת כל 5 מילישניות, בעיקרון עם draw(Canvas canvas).

תיאור פעולות על המידע

הציפור : מתחילה ממקום קבוע במסך. יש החלפה של תמונת הציפור כל 5 מילישניות בין שלושה תמונות כדי ליצור אנימציה מתאימה. הציפור תמיד במצב של נפילה והשחקן מונע את זה עם לחיצה על המסך. הרוטאציה של התמונה של הציפור נקבעת לפי המגע של המשתמש במסך – כאשר הציפור עפה למעלה הציפור מסתכלת למעלה, וכאשר הציפור נופלת למטה הציפור מסתכלת למטה.

הצינורות : נוצרים כל שנייה או שתיים עם ערך y התחלתי אקראי ותמיד בצד הימני של המסך. ערך ה-x שלהם פוחת כל 5 מילישניות כדי להזיז אותם שמאלה.

יש בדיקה מתמדת באמצעות המתודה intersect(). כדי לבדוק האם המלבנים, או ה-hitboxים של הציפור ואחד מהצינורות נחתכים. במקרה כזה, כמובן, המשחק נגמר.

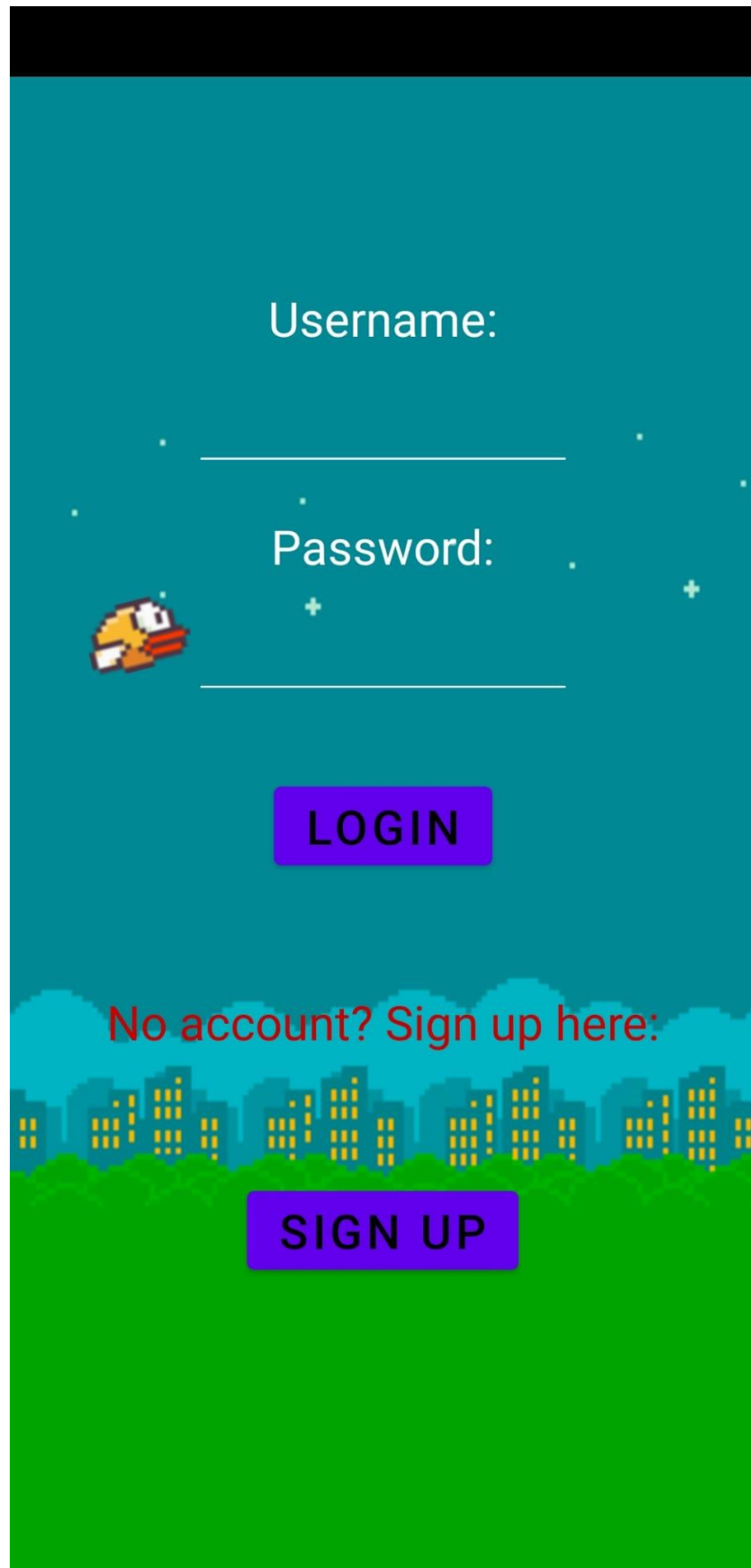
מבנה / ארכיטקטורה

תכנון ותיעוד מסכי הפרויקט

מסך פתיחה: המסך הראשון שמוצג עבור המשתמש. מסך זה דורש מהמשתמש להתחבר לחשבון קיים אם כבר יש לו, או ליצור אחד חדש.

לאחר הקלדת הפרטים, בלחיצה על כפתור Sign Up יוצר משתמש (במידה ולא קיים כבר חשבון שחולק את אותו היוזרניים עם המשתמש החדש שמנסה להיווצר) ותוצג הודעה מתאימה – בצבע ירוק במקרה של אישור יצירת המשתמש ובצבע אדום במקרה של שגיאה כלשהי.

לאחר מכן המשתמש יכול להזין את הפרטים של המשתמש הקיים באותו מקום שהזין את הפרטים של המשתמש שניסה ליצור. לאחר לחיצה על כפתור Login, במידה ולא הזין פרטים נכונים או קיימים תוצג הודעת שגיאה באדום, ובמידה ועשה הכל כשורה – תוצג הודעה בירוק והוא יועבר למסך הבא, מסך הקדם-משחק.



Username:

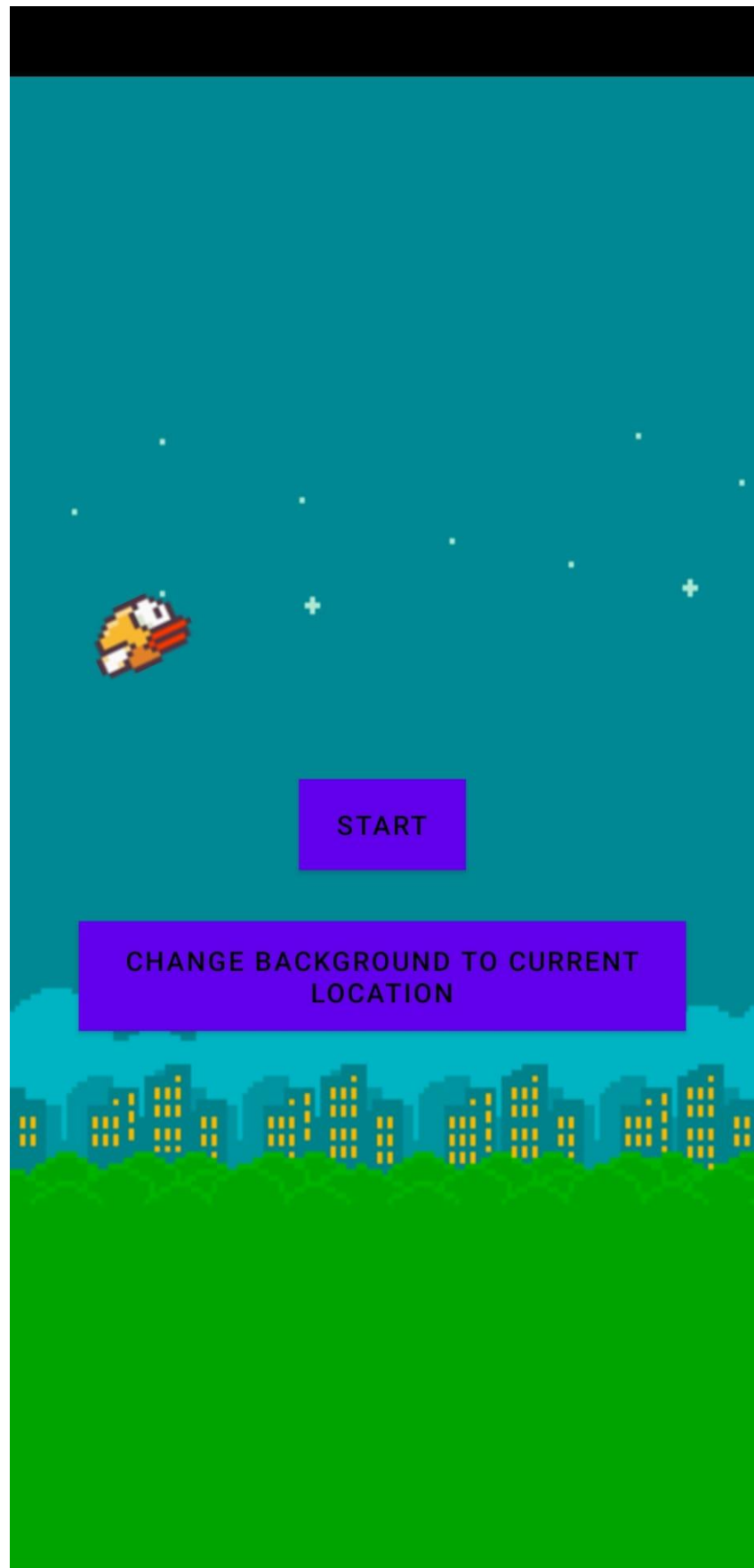
Password:

LOGIN

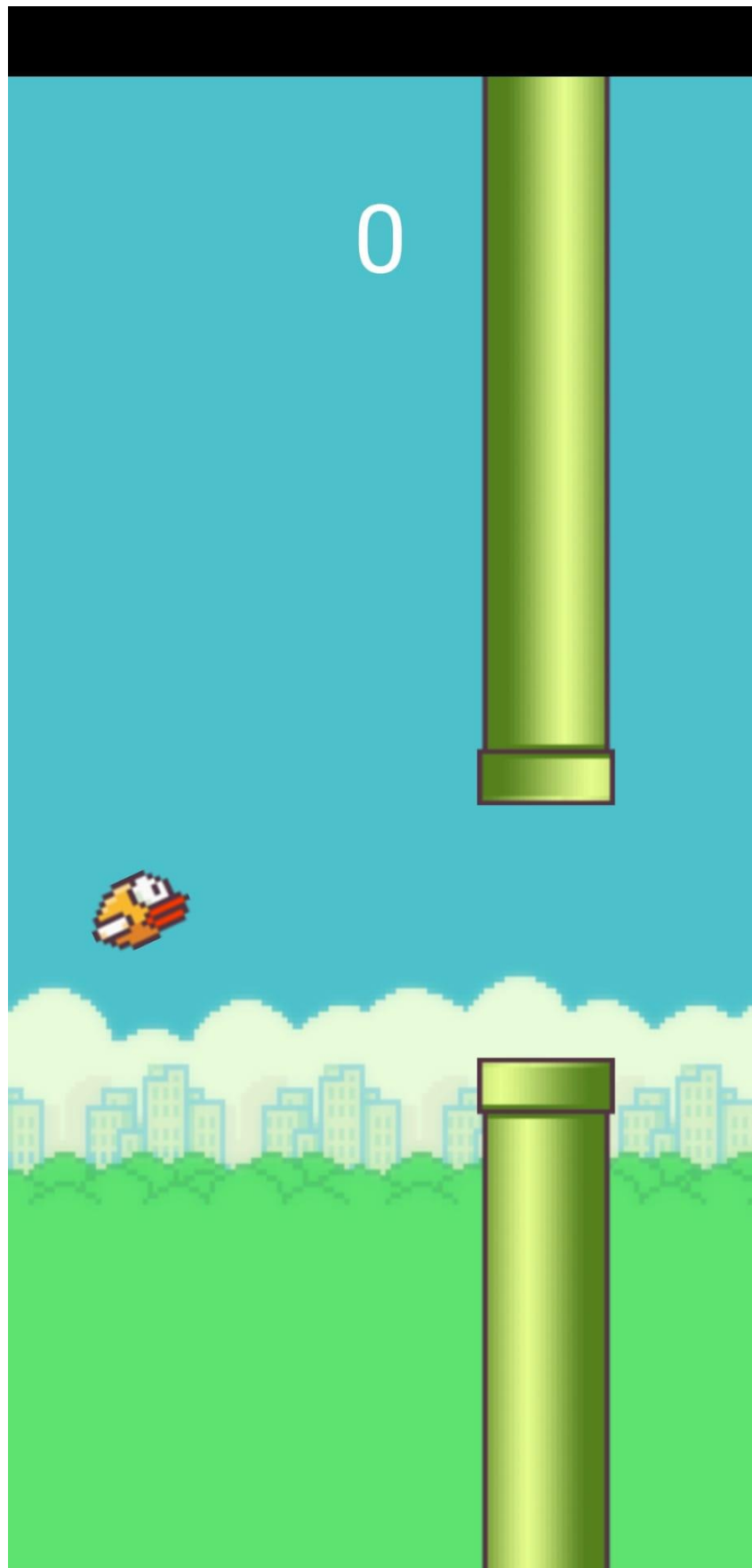
No account? Sign up here:

SIGN UP

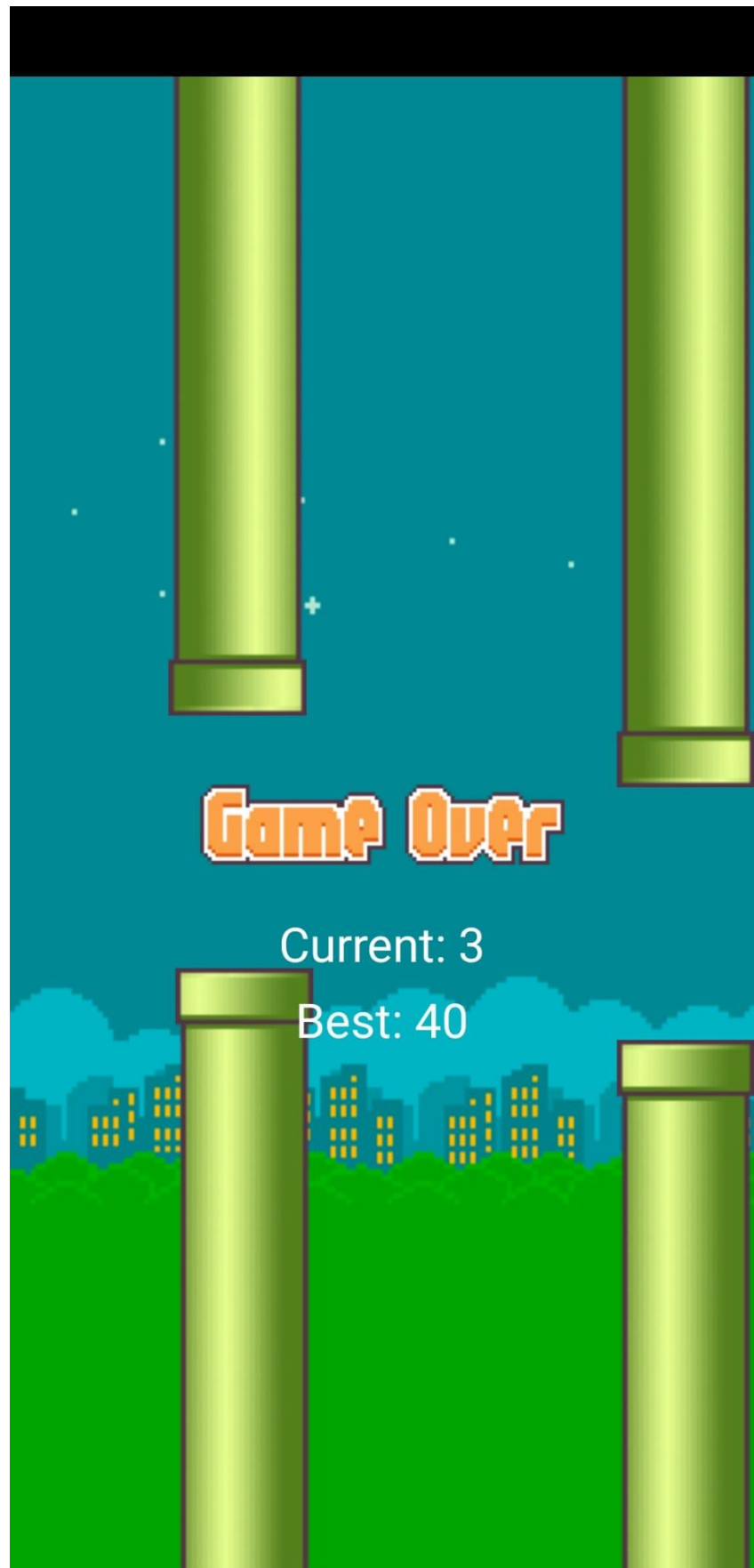
מסך קדם-משחק: מסך זה מאפשר למשתמש להתכונן לפני תחילת המשחק. בלחיצה על Start המשחק עצמו יתחיל. בלחיצה על Change Background to Current Location, במידה והמשתמש יסכים לחלוק את פרטי המיקום הנוכחי שלו עם האפליקציה, התוכנה תיצור קשר עם Google Maps API כדי לשלוף לפי קורדינאטות המיקום הנוכחי של המשתמש תמונת Street View של היכן הוא נמצא (חשוב לציין כי התמונה עלולה להיות לא עדכנית בגלל שגוגל לא שינו את התמונות בנהריה, העיר בה אני גר, מאז 2012 אם אני לא טועה).



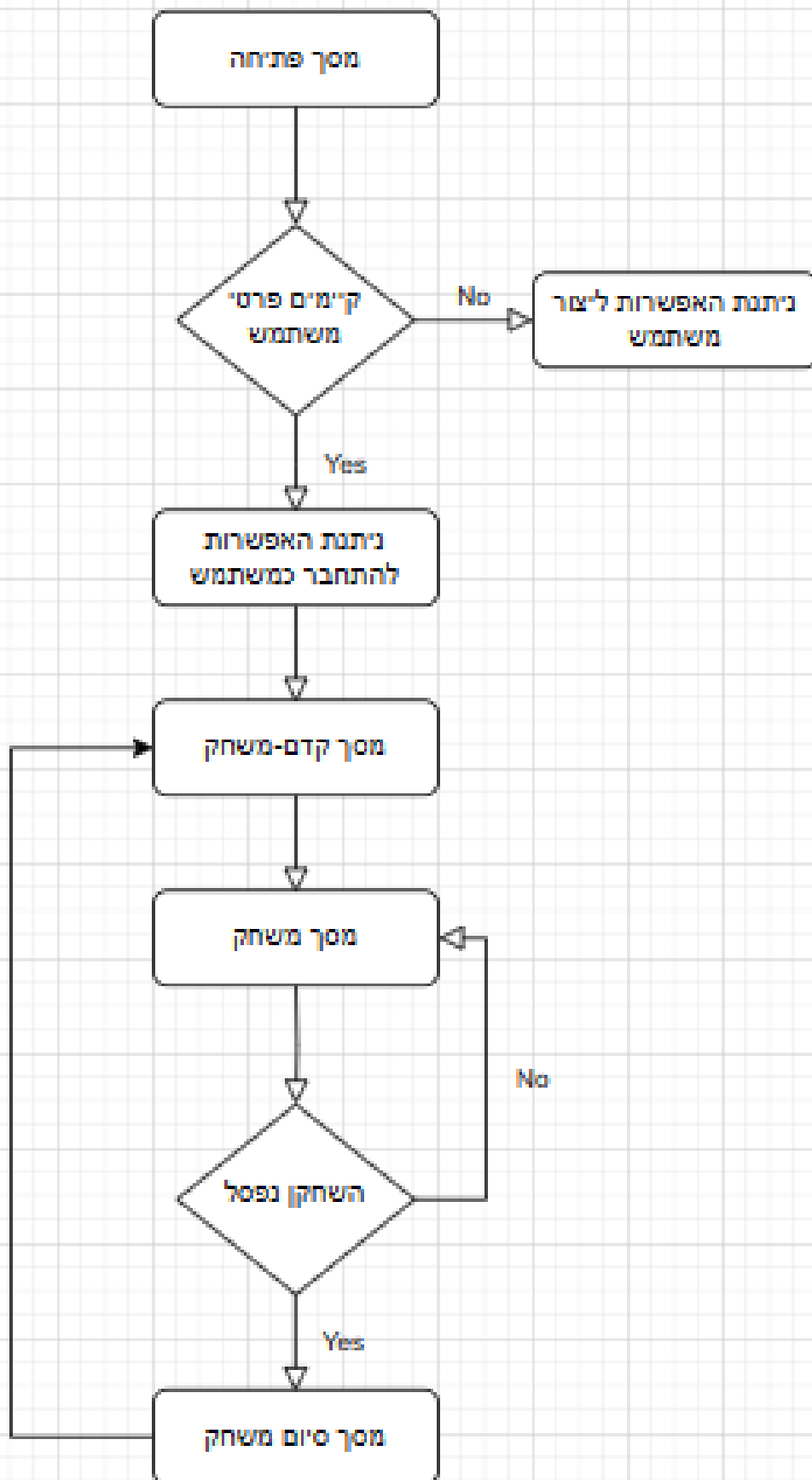
מסד משחק: המסך ה"ראשי" של האפליקציה והכי חשוב שלו – המשחק עצמו. השחקן יצטרך לעוף בזהירות דרך כל אחד מהמכשולים (צינורות) שמוצבות לפניו וכל פעם שעובר בהצלחה באמצע אחת מהן – תתווסף לו נקודה. המשחק אינסופי והמכשולים מוצבים ברנדומליות יחסית (שיעור ה-y ההתחלתי שבו מוצב מכשול משתנה כל פעם שמכשול חדש מתחולל).

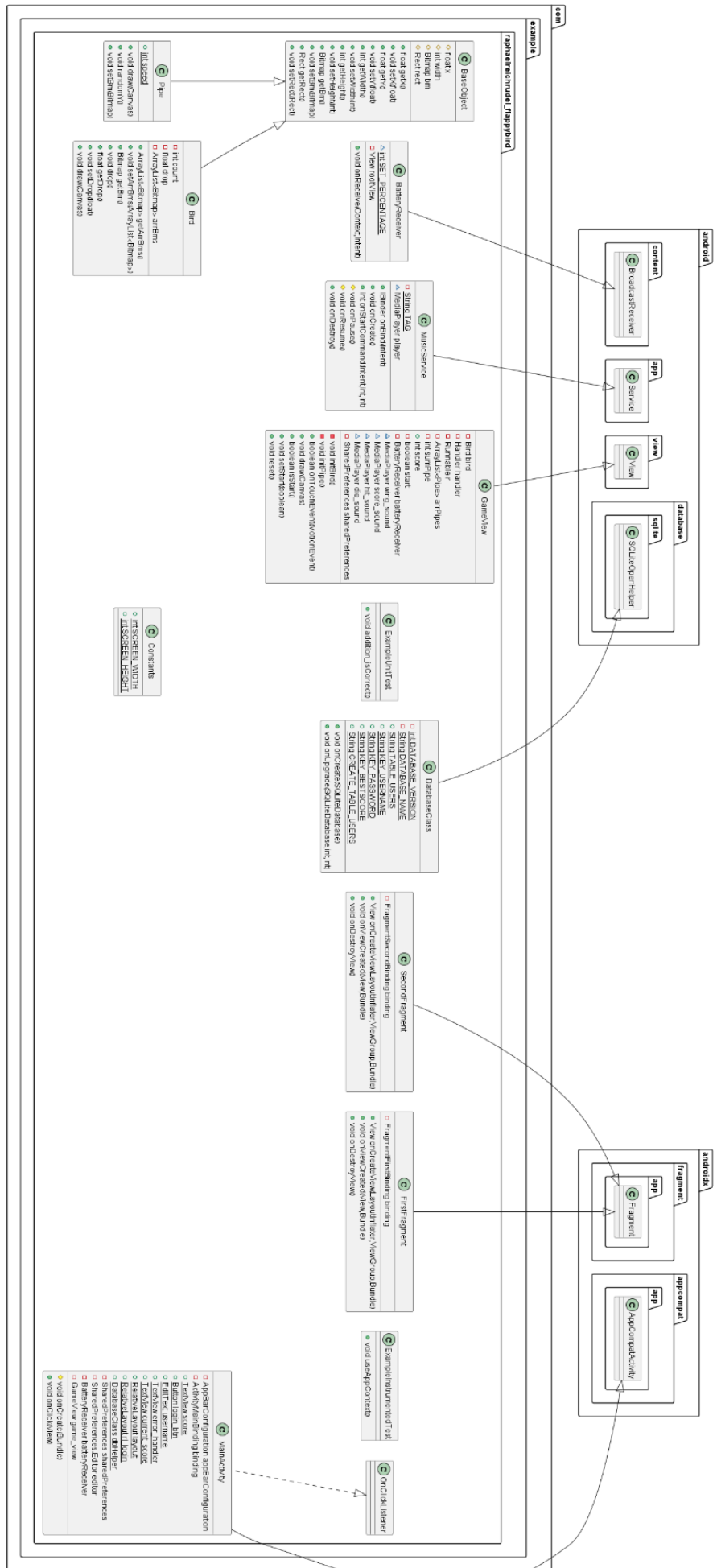


מסד סיום משחק: השחקן עתיד בסופו של דבר להיכשל במשחק. במסד זה מוצגים הניקוד שהמשתמש הצליח לרכוש במשחק הנוכחי, והניקוד הגבוה ביותר שהצליח לצבור אי פעם. בלחיצה על המסד הנ"ל המשתמש יועבר למסד הקדם-משחק, וחוזר חלילה.



תרשים היררכיית ומעברי המסכים, ותרשים תיאור מחלקות הפרויקט





מימוש הפרויקט
BaseObject.java

```
public class BaseObject {
    protected float x, y;
    protected int width, height;
    protected Bitmap bm;
    protected Rect rect;

    public BaseObject() {

    }

    public BaseObject(float x, float y, int width, int height) {
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
    }

    public float getX() {
        return x;
    }

    public void setX(float x) {
        this.x = x;
    }

    public float getY() {
        return y;
    }

    public void setY(float y) {
        this.y = y;
    }

    public int getWidth() {
        return width;
    }

    public void setWidth(int width) {
        this.width = width;
    }

    public int getHeight() {
        return height;
    }

    public void setHeight(int height) {
        this.height = height;
    }

    public Bitmap getBm() {
        return bm;
    }

    public void setBm(Bitmap bm) {
        this.bm = bm;
    }

    public Rect getRect() {
```



```

        return new Rect((int)(this.x), (int)(this.y), (int)(this.x +
this.width), (int)(this.y + this.height));
    }

    public void setRect(Rect rect) {
        this.rect = rect;
    }
}

```

מחלקה זו מייצגת "מחלקה כללית" שממנה יורשות שתי מחלקות אחרות, Bird ו-Pipe. היא מייצגת אובייקט שמצוייר על המסך;

```

protected float x, y;
protected int width, height;
protected Bitmap bm;
protected Rect rect;

```

x, y – מגדירים את הקורדינאטה הנוכחית של אובייקט על המסך.

width, height – מגדירים את הגודל של המלבן הסובב את האובייקט.

bm – שומר תמונה הקשורה לאובייקט, במקרה של הציפור לדוגמה יש 3 תמונות שמתחלפות ביניהן כל כמה מילישניות כדי להמחיש תעופה.

rect – המלבן עצמו הסובב את האובייקט.

במחלקה יש רק constructors / getters / setters רגילים אז אין מה להרחיב כאן בדיוק.

Bird.java

```

public class Bird extends BaseObject {
    private int count, vFlap, idCurrentBitmap;
    private float drop;
    private ArrayList<Bitmap> arrBms = new ArrayList<>();

    public Bird() {
        this.count = 0;
        this.vFlap = 5;
        this.idCurrentBitmap = 0;
        this.drop = 0;
    }

    public ArrayList<Bitmap> getArrBms() {
        return arrBms;
    }

    public void setArrBms(ArrayList<Bitmap> arrBms) {
        this.arrBms = arrBms;

        for (int i = 0; i < arrBms.size(); i++) {
            this.arrBms.set(i,
                Bitmap.createScaledBitmap(this.arrBms.get(i), this.width,
                this.height, true));
        }
    }

    @Override
    public Bitmap getBm() {
        count++;

        if (this.count == this.vFlap) {
            for (int i = 0; i < this.arrBms.size(); i++) {
                if (i == arrBms.size() - 1) {
                    this.idCurrentBitmap = 0;
                    break;
                }

                else if (this.idCurrentBitmap == i) {
                    idCurrentBitmap = i + 1;
                    break;
                }
            }

            count = 0;
        }

        if (this.drop < 0) {
            Matrix matrix = new Matrix();
            matrix.postRotate(-25);

            return Bitmap.createBitmap(arrBms.get(idCurrentBitmap),
            0, 0, arrBms.get(idCurrentBitmap).getWidth(),
            arrBms.get(idCurrentBitmap).getHeight(), matrix, true);
        }

        else if (this.drop >= 0) {
            Matrix matrix = new Matrix();

            if (this.drop < 70) {

```

```

        matrix.postRotate(-25 + (drop * 2));
    }

    else {
        matrix.postRotate(45);
    }

    return Bitmap.createBitmap(arrBms.get(idCurrentBitmap),
0, 0, arrBms.get(idCurrentBitmap).getWidth(),
arrBms.get(idCurrentBitmap).getHeight(), matrix, true);
    }

    return this.getArrBms().get(idCurrentBitmap);
}

public void drop() {
    this.drop += 0.6;
    this.y += this.drop;
}

public float getDrop() {
    return drop;
}

public void setDrop(float drop) {
    this.drop = drop;
}

public void draw(Canvas canvas) {
    drop();
    canvas.drawBitmap(this.getBm(), this.x, this.y, null);
}
}

```

מחלקה זה מייצגת את הציפור במשחק, ומייסדת "שלד" לפונקציונליות שלו.

```

private int count, vFlap, idCurrentBitmap;
private float drop;
private ArrayList<Bitmap> arrBms = new ArrayList<>();

```

count, vFlap, idCurrentBitmap – אחראים על ההתחלפות בין התמונות של הציפור.

drop – אחראי על הרוטאציה של הציפור. לדוגמה, כשלוחצים על המסך – הציפור עפה למעלה, וכך גם הגוף שלה פונה למעלה. דבר דומה קורה כשלא לוחצים על המסך בכלל בזמן המשחק – הציפור נופלת וכך גם הגוף שלה פונה למטה.

arrBms – שומר את התמונות, באמצעות ArrayList ו-Bitmap.

```

public void setArrBms(ArrayList<Bitmap> arrBms) {
    this.arrBms = arrBms;

    for (int i = 0; i < arrBms.size(); i++) {
        this.arrBms.set(i,
Bitmap.createScaledBitmap(this.arrBms.get(i), this.width,
this.height, true));
    }
}

```

setArrBms(ArrayList<Bitmap> arrBms) – אחראי על לקחת את התמונות של הציפור שאנו ניתן ב-GameView.java ולשים אותם ב-arrBms שישמור אותם להלאה.

```
@Override
public Bitmap getBm() {
    count++;

    if (this.count == this.vFlap) {
        for (int i = 0; i < this.arrBms.size(); i++) {
            if (i == arrBms.size() - 1) {
                this.idCurrentBitmap = 0;
                break;
            }

            else if (this.idCurrentBitmap == i) {
                idCurrentBitmap = i + 1;
                break;
            }
        }

        count = 0;
    }

    if (this.drop < 0) {
        Matrix matrix = new Matrix();
        matrix.postRotate(-25);

        return Bitmap.createBitmap(arrBms.get(idCurrentBitmap), 0, 0,
arrBms.get(idCurrentBitmap).getWidth(),
arrBms.get(idCurrentBitmap).getHeight(), matrix, true);
    }

    else if (this.drop >= 0) {
        Matrix matrix = new Matrix();

        if (this.drop < 70) {
            matrix.postRotate(-25 + (drop * 2));
        }

        else {
            matrix.postRotate(45);
        }

        return Bitmap.createBitmap(arrBms.get(idCurrentBitmap), 0, 0,
arrBms.get(idCurrentBitmap).getWidth(),
arrBms.get(idCurrentBitmap).getHeight(), matrix, true);
    }

    return this.getArrBms().get(idCurrentBitmap);
}
```

getBm() – החלק הזה קובע את המהירות שבה תמונות הציפור מתחלפות לפי vFlap, וכיצד התמונה עושה רוטאציות באוויר לפי המשתנה drop – כאשר הציפור נופלת התמונה מורכנת מטה, כשהיא עפה למעלה התמונה מורכנת מעלה.

חשוב: משום מה, האמצע של המסך משמש כ- $y = 0$, החצי העליון הוא $y < 0$ והחצי התחתון הוא $y > 0$. מטעה, אבל הייתי צריך לזכור את זה לאורך פיתוח כל דבר שקשור לתזוזה על המסך.

```
public void drop() {
    this.drop += 0.6;
```

```
this.y += this.drop;  
}
```

drop() – גורם לציפור ליפול למטה אוטומטית תמיד.

```
public void draw(Canvas canvas) {  
    drop();  
    canvas.drawBitmap(this.getBm(), this.x, this.y, null);  
}
```

draw(Canvas canvas) – מתפעל את הנפילה האוטומטית למטה, ומצייר את הציפור.

Pipe.java

```

public class Pipe extends BaseObject {
    public static int speed;

    public Pipe(float x, float y, int width, int height) {
        super(x, y, width, height);
        this.speed = 10 * Constants.SCREEN_WIDTH / 1080;
    }

    public void draw(Canvas canvas) {
        this.x -= this.speed;
        canvas.drawBitmap(this.bm, this.x, this.y, null);
    }

    public void randomY() {
        Random r = new Random();
        this.y = r.nextInt((0 + this.height / 4 + 1)) - this.height /
4;
    }

    @Override
    public void setBm(Bitmap bm) {
        this.bm = Bitmap.createScaledBitmap(bm, this.width,
this.height, true);
    }
}

```

מחלקה זו קובעת את הבסיס של המכשולים במשחק, הצינורות.

speed – קובע כמה מהר הצינורות זזים שמאלה.

draw(Canvas canvas) – מצייר את הצינור וקובע ששיעור ה-x שלו, כל עדכון מסך המשחק, ינמך ב-speed כדי שתהיה תזוזה שמאלה.

randomY() – מאתחל את שיעור ה-y של הצינור בצורה אקראית.

DatabaseClass.java

```

public class DatabaseClass extends SQLiteOpenHelper {
    public DatabaseClass(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(CREATE_TABLE_USERS);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_USERS);
        onCreate(db);
    }

    private static final int DATABASE_VERSION = 1;
    private static final String DATABASE_NAME = "my_database.db";

    public static final String TABLE_USERS = "users";

    public static final String KEY_USERNAME = "username";
    public static final String KEY_PASSWORD = "password";
    public static final String KEY_BESTSCORE = "bestscore";

    public static final String CREATE_TABLE_USERS = "CREATE TABLE "
        + TABLE_USERS + "(" + KEY_USERNAME + " TEXT PRIMARY KEY,"
        + KEY_PASSWORD
        + " TEXT NOT NULL," + KEY_BESTSCORE + " INTEGER NOT
        NULL)";
}

```

מחלקה זו מייצגת את מסד הנתונים :

קיים מסד נתונים אחד אשר מנהל את המשתמש, והוא נוצר ב-SQLite. במסד זה קיימים עמודות לשם המשתמש, סיסמת המשתמש, והניקוד המרבי שהצליח להשיג משלל כל המשחקים ששיחק. יש הכנסה ושליפה בגדר שם המשתמש וסיסמת המשתמש כאשר אנו רוצים ליצור משתמש, להתחבר בתור אחד או לשמור את ה-user session של המשתמש המחובר הנוכחי. ישנה הכנסה ושליפה לניקוד המרבי כאשר המשתמש שבר את שיא הניקוד שלו במשחק עצמו או כאשר אנו רוצים להציג את הניקוד המרבי שלו כאשר הוא נפסל.

דוגמה לטבלה שיוצר מסד נתונים :

Username	Password	Best Score
a	a	40
egor	goat	7
itzhak	bilial	18

MusicService.java

```

public class MusicService extends Service {
    private static final String TAG = null;
    MediaPlayer player;

    public IBinder onBind(Intent arg0) {
        return null;
    }

    @Override
    public void onCreate() {
        super.onCreate();
        player = MediaPlayer.create(this, R.raw.music);
        player.setLooping(true);
        player.setVolume(100,100);
    }

    public int onStartCommand(Intent intent, int flags, int startId)
    {
        player.start();
        return Service.START_STICKY;
    }

    protected void onPause() {
    }

    protected void onResume() {
    }

    @Override
    public void onDestroy() {
        player.stop();
        player.release();
    }
}

```

מחלקה זו אחראית על ניגון מוזיקה לאורך המשחק, פיצ'ר קטן ונחמד.

ב-AndroidManifest.xml יש להוסיף את השורה הזאת:

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


BatteryReceiver.java

```

public class BatteryReceiver extends BroadcastReceiver {
    static final int SET_PERCENTAGE = 80;

    private View rootView;

    public BatteryReceiver(View rootView) {
        this.rootView = rootView;
    }

    @Override
    public void onReceive(Context context, Intent intent) {
        int batteryLevel =
intent.getIntExtra(BatteryManager.EXTRA_LEVEL, -1);
        int batteryScale =
intent.getIntExtra(BatteryManager.EXTRA_SCALE, -1);
        float batteryPercent = (batteryLevel / (float)batteryScale) *
100;

        if (batteryPercent <= SET_PERCENTAGE) {
rootView.setBackgroundResource(R.drawable.background_low_battery);
        }

        else {
            Random r = new Random();
            int roll = r.nextInt(2);

            if (roll == 0) {
rootView.setBackgroundResource(R.drawable.background_day);
            }

            else {
rootView.setBackgroundResource(R.drawable.background_night);
            }
        }
    }
}

```

מחלקה זאת אחראית על שינוי הרקע של המסך לפי אחוז הסוללה של הפלאפון של השחקן. לצורכי בדיקה, ניתן לשנות את אחוז הסוללה שמתחתיו התמונה תשתנה. כאן לדוגמה מתחת ל-80% סוללה התמונה תהיה בטרייה כמעט ריקה, ומעל 80% סוללה התמונה תהיה רקע צבעוני עם גרסת יום / לילה שמתחלפים לפי משתנה אקראי.

ב-AndroidManifest.xml יש להוסיף את השורות האלה:

```

<receiver android:name=".BatteryReceiver" android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.BATTERY_CHANGED"
/>
    </intent-filter>
</receiver>

```

MainActivity.java

```

public class MainActivity extends AppCompatActivity implements
View.OnClickListener {
    private AppBarConfiguration appBarConfiguration;
    private ActivityMainBinding binding;

    public static TextView score;

    public static Button login_btn, signup_btn, start_btn,
location_btn;
    public static EditText username, password;
    public static TextView error_handler;

    public static TextView current_score, best_score;

    public static RelativeLayout layout;
    public static RelativeLayout rl_login, rl_gameover;

    public static DatabaseClass dbHelper;
    private SharedPreferences sharedPreferences;
    private SharedPreferences.Editor editor;

    private BatteryReceiver batteryReceiver;

    private GameView game_view;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);

        DisplayMetrics dm = new DisplayMetrics();
        this.getWindowManager().getDefaultDisplay().getMetrics(dm);
        Constants.SCREEN_WIDTH = dm.widthPixels;
        Constants.SCREEN_HEIGHT = dm.heightPixels;

        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        Intent musicService = new Intent(this, MusicService.class);
        startService(musicService);

        layout = (RelativeLayout) findViewById(R.id.background);

        batteryReceiver = new BatteryReceiver(layout);
        IntentFilter filter = new
IntentFilter(Intent.ACTION_BATTERY_CHANGED);
        getApplicationContext().registerReceiver(batteryReceiver,
filter);

        score = findViewById(R.id.score);

        login_btn = findViewById(R.id.login_btn);
        signup_btn = findViewById(R.id.signup_btn);
        start_btn = findViewById(R.id.start_btn);
        location_btn = findViewById(R.id.location_btn);
    }
}

```

```

        username = findViewById(R.id.username);
        password = findViewById(R.id.password);

        error_handler = findViewById(R.id.error_text);
        error_handler.setTypeface(null, Typeface.BOLD);

        current_score = findViewById(R.id.current_score);
        best_score = findViewById(R.id.best_score);

        rl_login = findViewById(R.id.rl_login);
        rl_gameover = findViewById(R.id.rl_gameover);

        dbHelper = new DatabaseClass(this);
        sharedPreferences = getSharedPreferences("UserSession",
Context.MODE_PRIVATE);
        editor = sharedPreferences.edit();

        editor.clear();
        editor.apply();

        game_view = findViewById(R.id.game_view);
    }

    @Override
    public void onClick(View view) {
        if (view == login_btn) {
            if (username.length() == 0 || password.length() == 0) {
                error_handler.setTextColor(Color.RED);
                error_handler.setText("Can't insert empty strings");
            }

            else if (sharedPreferences.getString("username",
"").length() > 0) {
                error_handler.setTextColor(Color.RED);
                error_handler.setText("User already logged in");
            }

            else {
                SQLiteDatabase db = dbHelper.getReadableDatabase();

                String[] projection = {
                    dbHelper.KEY_USERNAME,
                    dbHelper.KEY_PASSWORD
                };

                String selection = dbHelper.KEY_USERNAME + " = ? AND
" + dbHelper.KEY_PASSWORD + " = ?";
                String[] selectionArgs = {
username.getText().toString(), password.getText().toString() };

                Cursor cursor = db.query(
                    dbHelper.TABLE_USERS,
                    projection,
                    selection,
                    selectionArgs,
                    null,
                    null,
                    null
                );

                boolean result = cursor.getCount() > 0;
            }
        }
    }

```

```

        if (result) {
            editor.putString("username",
username.getText().toString());
            editor.apply();

            error_handler.setTextColor(Color.GREEN);
            error_handler.setText("Login successful!");

            final Handler handler = new Handler();

            handler.postDelayed(new Runnable() {
                @Override
                public void run() {
                    if
(sharedPreferences.getString("username", "") != null) {
                        SQLiteDatabase db =
MainActivity.dbHelper.getReadableDatabase();

                        String[] projection = {
MainActivity.dbHelper.KEY_BESTSCORE };

                        String selection =
MainActivity.dbHelper.KEY_USERNAME + " = ?";

                        String[] selectionArgs =
{sharedPreferences.getString("username", "")};

                        Cursor cursor = db.query(
MainActivity.dbHelper.TABLE_USERS,

                                projection,
                                selection,
                                selectionArgs,
                                null,
                                null,
                                null

                        );

                        if (cursor.moveToFirst()) {
                            game_view.best_score =
cursor.getInt(0);
                        }

                        cursor.close();
                        db.close();
                    }

                    start_btn.setVisibility(View.VISIBLE);
                    location_btn.setVisibility(View.VISIBLE);
                    rl_login.setVisibility(View.INVISIBLE);
                }
            }, 1000);
        }

        else {
            error_handler.setTextColor(Color.RED);
            error_handler.setText("User doesn't exist");
        }

        cursor.close();
    }

```

```

        db.close();
    }
}

if (view == signup_btn) {
    if (username.length() == 0 || password.length() == 0) {
        error_handler.setTextColor(Color.RED);
        error_handler.setText("Can't insert empty strings");
    }

    else {
        SQLiteDatabase db = dbHelper.getWritableDatabase();

        ContentValues values_users = new ContentValues();

        values_users.put(dbHelper.KEY_USERNAME,
username.getText().toString());
        values_users.put(dbHelper.KEY_PASSWORD,
password.getText().toString());
        values_users.put(dbHelper.KEY_BESTSCORE, 0);

        try {
            db.insertOrThrow(dbHelper.TABLE_USERS, null,
values_users);

            error_handler.setTextColor(Color.GREEN);
            error_handler.setText("User successfully
created!");
        }

        catch (SQLiteConstraintException e) {
            error_handler.setTextColor(Color.RED);
            error_handler.setText("User already exists");
        }

        catch (SQLiteException e) {
            error_handler.setTextColor(Color.RED);
            error_handler.setText("Error occurred, try
again");
        }

        db.close();
    }
}

if (view == location_btn) {
    if (ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
        FusedLocationProviderClient fusedLocationClient =
LocationServices.getFusedLocationProviderClient(this);

        fusedLocationClient.getLastLocation().addOnSuccessListener(new
OnSuccessListener<Location>() {
            @Override
            public void onSuccess(Location location) {
                if (location != null) {
                    DisplayMetrics displayMetrics = new
DisplayMetrics();

                    getWindowManager().getDefaultDisplay().getMetrics(displayMetrics);
                    int screenWidth =

```

```

displayMetrics.widthPixels;
                                int screenHeight =
displayMetrics.heightPixels;

                                double latitude = location.getLatitude();
                                double longitude =
location.getLongitude();

                                String streetViewUrl =
"https://maps.googleapis.com/maps/api/streetview?size=" + screenWidth
+ "x" + screenHeight + "&location=" + latitude + "," + longitude +
"&fov=110&heading=0&pitch=0&key=API_KEY_HERE";

                                ImageView imageView = new
ImageView(getApplicationContext());

Picasso.get().load(streetViewUrl).into(imageView, new Callback() {
    @Override
    public void onSuccess() {
        Drawable drawable =
imageView.getDrawable();

        if (drawable != null) {

layout.setBackground(drawable);

        }

    }

    @Override
    public void onError(Exception e) {
        // Handle error by displaying a
message... not REALLYLY necessary
    }

});

    }

});

    } else {
        ActivityCompat.requestPermissions(this, new String[]
{ Manifest.permission.ACCESS_FINE_LOCATION }, 1);
    }

    if (view == start_btn) {
        game_view.setStart(true);
        score.setVisibility(View.VISIBLE);
        start_btn.setVisibility(View.INVISIBLE);
        location_btn.setVisibility(View.INVISIBLE);
    }

    if (view == rl_gameover) {
        game_view.setStart(false);
        rl_gameover.setVisibility(View.INVISIBLE);
        start_btn.setVisibility(View.VISIBLE);
        location_btn.setVisibility(View.VISIBLE);
        game_view.reset();
    }

}
}

```

המחלקה הזו היא המחלקה ה"ראשית" של האפליקציה, שממנה ניתן לתפעל את כל המשחק.

```
public static TextView score;

public static Button login_btn, signup_btn, start_btn, location_btn;
public static EditText username, password;
public static TextView error_handler;

public static TextView current_score, best_score;

public static RelativeLayout layout;
public static RelativeLayout rl_login, rl_gameover;

public static DatabaseClass dbHelper;
private SharedPreferences sharedPreferences;
private SharedPreferences.Editor editor;

private BatteryReceiver batteryReceiver;

private GameView game_view;
```

score – מראה את הניקוד הנוכחי של המשתמש.

login_btn, signup_btn, start_btn, location_btn – כפתורים למיניהם שהשימוש שלהם מתקשר לשם שלהם.

username, password – מקומות בהם ניתן להכניס שם וסיסמה.

error_handler – טקסט שמוסתר תחילה וצץ בצבע אדום / ירוק במקרה של שגיאה או הצלחה בהקשר למערכת הלוגין.

current_score, best_score – מראים את הניקוד הנוכחי והניקוד הגבוה ביותר של המשתמש לאחר שהוא נפסל.

layout, rl_login, rl_gameover – שומרים מגוון relative layouts לשימושים שונים כגון שינוי מסך הרקע והסתרה / החבאה של מערכת הלוגין או מסך סיום המשחק.

dbHelper – קשור לניהול מסד הנתונים.

sharedPreferences, editor – קשורים לניהול ה-user session.

batteryReceiver – קשור לפיצ'ר שמשנה את רקע המסך לפי אחוז הסוללה שלו.

gameView – אובייקט של המחלקה GameView.java, שמשומש לאתחול המשחק.

```
@Override
public void onClick(View view) {
    if (view == login_btn) {
        if (username.length() == 0 || password.length() == 0) {
            error_handler.setTextColor(Color.RED);
            error_handler.setText("Can't insert empty strings");
        }

        else if (sharedPreferences.getString("username", "").length()
> 0) {
            error_handler.setTextColor(Color.RED);
            error_handler.setText("User already logged in");
        }

        else {
```

```

        SQLiteDatabase db = dbHelper.getReadableDatabase();

        String[] projection = {
            dbHelper.KEY_USERNAME,
            dbHelper.KEY_PASSWORD
        };

        String selection = dbHelper.KEY_USERNAME + " = ? AND " +
        dbHelper.KEY_PASSWORD + " = ?";
        String[] selectionArgs = { username.getText().toString(),
        password.getText().toString() };

        Cursor cursor = db.query(
            dbHelper.TABLE_USERS,
            projection,
            selection,
            selectionArgs,
            null,
            null,
            null
        );

        boolean result = cursor.getCount() > 0;

        if (result) {
            editor.putString("username",
        username.getText().toString());
            editor.apply();

            error_handler.setTextColor(Color.GREEN);
            error_handler.setText("Login successful!");

            final Handler handler = new Handler();

            handler.postDelayed(new Runnable() {
                @Override
                public void run() {
                    if (sharedPreferences.getString("username",
        "") != null) {
                        SQLiteDatabase db =
        MainActivity.dbHelper.getReadableDatabase();

                        String[] projection = {
        MainActivity.dbHelper.KEY_BESTSCORE };

                        String selection =
        MainActivity.dbHelper.KEY_USERNAME + " = ?";

                        String[] selectionArgs =
        {sharedPreferences.getString("username", "")};

                        Cursor cursor = db.query(
        MainActivity.dbHelper.TABLE_USERS,
                            projection,
                            selection,
                            selectionArgs,
                            null,
                            null,
                            null
                        );
                    }
                }
            }, 1000);
        }
    }
}

```



```

        if (cursor.moveToFirst()) {
            game_view.best_score =
cursor.getInt(0);
        }

        cursor.close();
        db.close();
    }

    start_btn.setVisibility(View.VISIBLE);
    location_btn.setVisibility(View.VISIBLE);
    rl_login.setVisibility(View.INVISIBLE);
}
}, 1000);
}

else {
    error_handler.setTextColor(Color.RED);
    error_handler.setText("User doesn't exist");
}

cursor.close();
db.close();
}
}

if (view == signup_btn) {
    if (username.length() == 0 || password.length() == 0) {
        error_handler.setTextColor(Color.RED);
        error_handler.setText("Can't insert empty strings");
    }

    else {
        SQLiteDatabase db = dbHelper.getWritableDatabase();

        ContentValues values_users = new ContentValues();

        values_users.put(dbHelper.KEY_USERNAME,
username.getText().toString());
        values_users.put(dbHelper.KEY_PASSWORD,
password.getText().toString());
        values_users.put(dbHelper.KEY_BESTSCORE, 0);

        try {
            db.insertOrThrow(dbHelper.TABLE_USERS, null,
values_users);
            error_handler.setTextColor(Color.GREEN);
            error_handler.setText("User successfully created!");
        }

        catch (SQLiteConstraintException e) {
            error_handler.setTextColor(Color.RED);
            error_handler.setText("User already exists");
        }

        catch (SQLiteException e) {
            error_handler.setTextColor(Color.RED);
            error_handler.setText("Error occurred, try again");
        }
    }
}

```

```

        db.close();
    }
}

if (view == location_btn) {
    if (ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
        FusedLocationProviderClient fusedLocationClient =
LocationServices.getFusedLocationProviderClient(this);

fusedLocationClient.getLastLocation().addOnSuccessListener(new
OnSuccessListener<Location>() {
    @Override
    public void onSuccess(Location location) {
        if (location != null) {
            DisplayMetrics displayMetrics = new
DisplayMetrics();

getWindowManager().getDefaultDisplay().getMetrics(displayMetrics);
            int screenWidth = displayMetrics.widthPixels;
            int screenHeight =
displayMetrics.heightPixels;

            double latitude = location.getLatitude();
            double longitude = location.getLongitude();

            String streetViewUrl =
"https://maps.googleapis.com/maps/api/streetview?size=" + screenWidth
+ "x" + screenHeight + "&location=" + latitude + "," + longitude +
"&fov=110&heading=0&pitch=0&key=API_KEY_HERE";

            ImageView imageView = new
ImageView(getApplicationContext());

Picasso.get().load(streetViewUrl).into(imageView, new Callback() {
    @Override
    public void onSuccess() {
        Drawable drawable =
imageView.getDrawable();

        if (drawable != null) {
            layout.setBackground(drawable);
        }
    }

    @Override
    public void onError(Exception e) {
        // Handle error by displaying a
message... not REALLY necessary
    }
});
        }
    }
});

} else {
    ActivityCompat.requestPermissions(this, new String[] {
Manifest.permission.ACCESS_FINE_LOCATION }, 1);
}
}

```

```

if (view == start_btn) {
    game_view.setStart(true);
    score.setVisibility(View.VISIBLE);
    start_btn.setVisibility(View.INVISIBLE);
    location_btn.setVisibility(View.INVISIBLE);
}

if (view == rl_gameover) {
    game_view.setStart(false);
    rl_gameover.setVisibility(View.INVISIBLE);
    start_btn.setVisibility(View.VISIBLE);
    location_btn.setVisibility(View.VISIBLE);
    game_view.reset();
}
}

```

onClick(View view) – חלק זה מתפרס למספר חלקים שונים. כאשר המשתמש לוחץ על אחד מהכפתורים / מסך סיום המשחק, הוא מועבר לפונקציה הזאת שמטפלת בכל מיני מקרים לפרט. להלן פירוט של כל אחד מהטיפולים הנ"ל:

view == login_btn : מתופעל כאשר המשתמש מנסה להתחבר לחשבון שלו. יש כל מיני בדיקות סף כדי לבדוק שהחיבור אכן תקין ופעולות שמוודאות שהחיבור יתבצע כהלכה, שבמידה והכל בסדר המשתמש מועבר למסך הקדם-משחק.

view == signup_btn : מתופעל כאשר המשתמש מנסה ליצור משתמש חדש. דומה מאוד ל- login_btn מבחינת בדיקות וכד'.

view == location_btn : הפיצ'ר שמשתמש ב-Google Maps API. מגבש מידע אודות גודל מסך המשתמש והמיקום הנוכחי שלו, ויוצר שאילתה שמבקשת תמונת Street View. שימוש ב-Picasso ממיר את התמונה שקיבלנו מהשאילתה לתמונה שניתן להשתמש בה כדי להחליף את רקע המסך. השורה key = API_KEY_HERE בסוף בעצם משתמשת ב-API Key ייחודי שקיבלתי מגוגל, אותו החבאתי כי הוא מהווה מידע רגיש.

יש לשים ב-AndroidManifest.xml את השורות הבאות:

```

<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />

```

view == start_btn : קיים במסך הקדם-משחק ומתפעל את המשחק בלחיצה עליו לאחר הסתרת אלמנטים למיניהם על המסך וכו'.

view == rl_gameover : כאשר המשתמש נפסל ומועבר למסך סיום המשחק, לחיצה על המסך תעשה שינויים לוגיים כאלה ואחרים כדי לאתחל את המשחק ולהעביר אותו למסך הקדם-משחק.

GameView.java

```

public class GameView extends View {
    private Bird bird;
    private Handler handler;
    private Runnable r;

    private ArrayList<Pipe> arrPipes;
    private int sumPipe, distance;

    public int score, best_score;

    private boolean start;

    private BatteryReceiver batteryReceiver;

    MediaPlayer wing_sound = MediaPlayer.create(getContext(),
R.raw.wing);
    MediaPlayer score_sound = MediaPlayer.create(getContext(),
R.raw.point);
    MediaPlayer hit_sound = MediaPlayer.create(getContext(),
R.raw.hit);
    MediaPlayer die_sound = MediaPlayer.create(getContext(),
R.raw.die);

    private SharedPreferences sharedPreferences =
getContext().getSharedPreferences("UserSession",
Context.MODE_PRIVATE);

    public GameView(Context context, @Nullable AttributeSet attrs) {
        super(context, attrs);

        score = 0;

        start = false;

        initBird();
        initPipe();

        handler = new Handler();

        r = new Runnable() {
            @Override
            public void run() {
                invalidate();
            }
        };
    }

    private void initBird() {
        bird = new Bird();
        bird.setWidth(150 * (Constants.SCREEN_WIDTH / 1080));
        bird.setHeight(100 * (Constants.SCREEN_HEIGHT / 1920));
        bird.setX(100 * (Constants.SCREEN_WIDTH / 1080));
        bird.setY(Constants.SCREEN_HEIGHT / 2 - bird.getHeight() /
2);

        ArrayList<Bitmap> arrBms = new ArrayList<>();

        arrBms.add(BitmapFactory.decodeResource(this.getResources(),
R.drawable.yellowbird_upflap));
    }
}

```

```

        arrBms.add(BitmapFactory.decodeResource(this.getResources(),
R.drawable.yellowbird_midflap));
        arrBms.add(BitmapFactory.decodeResource(this.getResources(),
R.drawable.yellowbird_downflap));

        if (android.os.Build.VERSION.SDK_INT >=
android.os.Build.VERSION_CODES.JELLY_BEAN_MR1) {
            if
(Settings.Global.getInt(getContext().getContentResolver(),
Settings.Global.AIRPLANE_MODE_ON, 0) != 0) {
                for (int i = 0; i < arrBms.size(); i++) {
                    arrBms.set(i,
BitmapFactory.decodeResource(this.getResources(),
R.drawable.airplane));
                }
            }
        }

        else {
            if
(Settings.System.getInt(getContext().getContentResolver(),
Settings.System.AIRPLANE_MODE_ON, 0) != 0) {
                for (int i = 0; i < arrBms.size(); i++) {
                    arrBms.set(i,
BitmapFactory.decodeResource(this.getResources(),
R.drawable.airplane));
                }
            }
        }

        bird.setArrBms(arrBms);
    }

    private void initPipe() {
        sumPipe = 4;
        distance = 325 * Constants.SCREEN_HEIGHT / 1920;
        arrPipes = new ArrayList<Pipe>();

        for (int i = 0; i < sumPipe; i++) {
            if (i < sumPipe / 2) {
                this.arrPipes.add(new Pipe(Constants.SCREEN_WIDTH + i
* ((Constants.SCREEN_WIDTH + 200 * Constants.SCREEN_WIDTH / 1080) /
(sumPipe / 2)), 0, 200 * Constants.SCREEN_WIDTH / 1080,
Constants.SCREEN_HEIGHT / 2));
                this.arrPipes.get(this.arrPipes.size() -
1).setBm(BitmapFactory.decodeResource(this.getResources(),
R.drawable.pipe_green_upside_down));
                this.arrPipes.get(this.arrPipes.size() -
1).randomY();
            }
            else {
                this.arrPipes.add(new Pipe(this.arrPipes.get(i -
sumPipe / 2).getX(), this.arrPipes.get(i - sumPipe / 2).getY() +
this.arrPipes.get(i - sumPipe / 2).getHeight() + this.distance, 200 *
Constants.SCREEN_WIDTH / 1080, Constants.SCREEN_HEIGHT / 2));
                this.arrPipes.get(this.arrPipes.size() -
1).setBm(BitmapFactory.decodeResource(this.getResources(),
R.drawable.pipe_green));
            }
        }
    }

```

```

    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        if (event.getAction() == MotionEvent.ACTION_DOWN) {
            bird.setDrop(-15);

            if (wing_sound.isPlaying()) {
                wing_sound.seekTo(0);
            }

            else {
                wing_sound.start();
            }
        }

        return true;
    }

    public void draw(Canvas canvas) {
        super.draw(canvas);

        if (start) {
            bird.draw(canvas);

            for (int i = 0; i < sumPipe; i++) {
                if
                (this.bird.getRect().intersect(arrPipes.get(i).getRect()) ||
                bird.getY() - bird.getHeight() < 0 || bird.getY() + bird.getHeight()
                > Constants.SCREEN_HEIGHT) {
                    bird.setWidth(0);
                    bird.setHeight(0);

                    Pipe.speed = 0;

                    if (!hit_sound.isPlaying()) {
                        hit_sound.start();
                        hit_sound.setOnCompletionListener(new
                MediaPlayer.OnCompletionListener() {
                            @Override
                            public void onCompletion(MediaPlayer
                mediaPlayer) {
                                mediaPlayer.stop();
                                mediaPlayer.reset();
                            }
                        });
                }

                    if (!die_sound.isPlaying()) {
                        die_sound.start();
                        die_sound.setOnCompletionListener(new
                MediaPlayer.OnCompletionListener() {
                            @Override
                            public void onCompletion(MediaPlayer
                mediaPlayer) {
                                mediaPlayer.stop();
                                mediaPlayer.reset();
                            }
                        });
                }
            }
        }
    }

```

```

        MainActivity.current_score.setText("Current: " +
Integer.toString(score));
        MainActivity.best_score.setText("Best: " +
Integer.toString(best_score));
        MainActivity.score.setVisibility(View.INVISIBLE);

MainActivity.rl_gameover.setVisibility(View.VISIBLE);
    }

    if ((this.bird.getX() + this.bird.getWidth() >
arrPipes.get(i).getX() + arrPipes.get(i).getWidth() / 2) &&
(this.bird.getX() + this.bird.getWidth() <= arrPipes.get(i).getX() +
arrPipes.get(i).getWidth() / 2 + Pipe.speed) && (i < sumPipe / 2)) {
        score++;

        if (score_sound.isPlaying()) {
            score_sound.seekTo(0);
        }

        else {
            score_sound.start();
        }

        if (score > best_score) {
            best_score = score;

            SQLiteDatabase db =
MainActivity.dbHelper.getWritableDatabase();

            ContentValues values = new ContentValues();

values.put(MainActivity.dbHelper.KEY_BESTSCORE, best_score);

            String selection =
MainActivity.dbHelper.KEY_USERNAME + " = ?";
            String[] selectionArgs = {
sharedPreferences.getString("username", "") };

            db.update(MainActivity.dbHelper.TABLE_USERS,
values, selection, selectionArgs);

            db.close();
        }

MainActivity.score.setText(Integer.toString(score));
    }

    if (this.arrPipes.get(i).getX() < -
arrPipes.get(i).getWidth()) {
this.arrPipes.get(i).setX(Constants.SCREEN_WIDTH);

        if (i < sumPipe / 2) {
            arrPipes.get(i).randomY();
        }

        else {
            arrPipes.get(i).setY(this.arrPipes.get(i -
sumPipe / 2).getY() + this.arrPipes.get(i - sumPipe / 2).getHeight()
+ this.distance);

```

```

        }
    }

    this.arrPipes.get(i).draw(canvas);
}

else {
    if (bird.getY() > Constants.SCREEN_HEIGHT / 2) {
        bird.setDrop(-15 * Constants.SCREEN_HEIGHT / 1920);
    }

    bird.draw(canvas);
}

handler.postDelayed(r, 5);
}

public boolean isStart() {
    return start;
}

public void setStart(boolean start) {
    this.start = start;
}

public void reset() {
    MainActivity.score.setText("0");

    score = 0;

    initPipe();
    initBird();

    hit_sound = MediaPlayer.create(getContext(), R.raw.hit);
    die_sound = MediaPlayer.create(getContext(), R.raw.die);

    Activity activity = (Activity)getContext();
    MainActivity mainActivity = (MainActivity)activity;

    RelativeLayout layout =
mainActivity.findViewById(R.id.background);

    batteryReceiver = new BatteryReceiver(layout);
    IntentFilter filter = new
IntentFilter(Intent.ACTION_BATTERY_CHANGED);

getContext().getApplicationContext().registerReceiver(batteryReceiver
, filter);
}
}

```

המחלקה הזו גם היא מהווה "עיקרית" וחשובה, שכן היא מנהלת את היגיון המשחק עצמו.

```

private Bird bird;
private Handler handler;
private Runnable r;

private ArrayList<Pipe> arrPipes;
private int sumPipe, distance;

```



```

public int score, best_score;

private boolean start;

private BatteryReceiver batteryReceiver;

MediaPlayer wing_sound = MediaPlayer.create(getContext(),
R.raw.wing);
MediaPlayer score_sound = MediaPlayer.create(getContext(),
R.raw.point);
MediaPlayer hit_sound = MediaPlayer.create(getContext(), R.raw.hit);
MediaPlayer die_sound = MediaPlayer.create(getContext(), R.raw.die);

private SharedPreferences sharedPreferences =
getContext().getSharedPreferences("UserSession",
Context.MODE_PRIVATE);

```

bird – אובייקט של הציפור שלנו.

handler, r – אחראים על תזמונים למיניהם וריצה עדכנית ורציפה של המשחק.

arrPipes, sumPipe, distance – קשורים ללוגיקת הצינורות.

score, best_score – מנהלים את הניקוד של המשתמש.

start – מדליק את המשחק עם הלחיצה על start_btn מהמחלקה הקודמת, MainActivity.java.

batteryReceiver – אחראי על פיצ'ר הסוללה.

wing_sound, score_sound, hit_sound, die_sound – סאונד אפקטים למיניהם שתומכים בחווית המשחק.

sharedPreferences – מתקשר ל-user session בשביל שליפה של ה-best score.

```

private void initBird() {
    bird = new Bird();
    bird.setWidth(150 * (Constants.SCREEN_WIDTH / 1080));
    bird.setHeight(100 * (Constants.SCREEN_HEIGHT / 1920));
    bird.setX(100 * (Constants.SCREEN_WIDTH / 1080));
    bird.setY(Constants.SCREEN_HEIGHT / 2 - bird.getHeight() / 2);

    ArrayList<Bitmap> arrBms = new ArrayList<>();

    arrBms.add(BitmapFactory.decodeResource(this.getResources(),
R.drawable.yellowbird_upflap));
    arrBms.add(BitmapFactory.decodeResource(this.getResources(),
R.drawable.yellowbird_midflap));
    arrBms.add(BitmapFactory.decodeResource(this.getResources(),
R.drawable.yellowbird_downflap));

    if (android.os.Build.VERSION.SDK_INT >=
android.os.Build.VERSION_CODES.JELLY_BEAN_MR1) {
        if (Settings.Global.getInt(getContext().getContentResolver(),
Settings.Global.AIRPLANE_MODE_ON, 0) != 0) {
            for (int i = 0; i < arrBms.size(); i++) {
                arrBms.set(i,
BitmapFactory.decodeResource(this.getResources(),
R.drawable.airplane));
            }
        }
    }
}

```

```

    }
}

else {
    if (Settings.System.getInt(getContext().getContentResolver(),
Settings.System.AIRPLANE_MODE_ON, 0) != 0) {
        for (int i = 0; i < arrBms.size(); i++) {
            arrBms.set(i,
BitmapFactory.decodeResource(this.getResources(),
R.drawable.airplane));
        }
    }

    bird.setArrBms(arrBms);
}
}

```

initBird() – מאתחל את הציפור. כאן יש שימוש בפיצ'ר נחמד שמשנה את תמונת הציפור במידה ומצב טיסה מופעל לתמונה של מטוס.

```

private void initPipe() {
    sumPipe = 4;
    distance = 325 * Constants.SCREEN_HEIGHT / 1920;
    arrPipes = new ArrayList<Pipe>();

    for (int i = 0; i < sumPipe; i++) {
        if (i < sumPipe / 2) {
            this.arrPipes.add(new Pipe(Constants.SCREEN_WIDTH + i *
(Constants.SCREEN_WIDTH + 200 * Constants.SCREEN_WIDTH / 1080) /
(sumPipe / 2)), 0, 200 * Constants.SCREEN_WIDTH / 1080,
Constants.SCREEN_HEIGHT / 2));
            this.arrPipes.get(this.arrPipes.size() -
1).setBm(BitmapFactory.decodeResource(this.getResources(),
R.drawable.pipe_green_upside_down));
            this.arrPipes.get(this.arrPipes.size() - 1).randomY();
        }

        else {
            this.arrPipes.add(new Pipe(this.arrPipes.get(i - sumPipe
/ 2).getX(), this.arrPipes.get(i - sumPipe / 2).getY() +
this.arrPipes.get(i - sumPipe / 2).getHeight() + this.distance, 200 *
Constants.SCREEN_WIDTH / 1080, Constants.SCREEN_HEIGHT / 2));
            this.arrPipes.get(this.arrPipes.size() -
1).setBm(BitmapFactory.decodeResource(this.getResources(),
R.drawable.pipe_green));
        }
    }
}
}

```

initPipe() – מאתחל את הצינורות, בין אם זה קצב התחוללות הצינורות, המרחק בין כל אחד מהם או המיקום ההתחלתי שבו הם נוצרים.

```

@Override
public boolean onTouchEvent(MotionEvent event) {
    if (event.getAction() == MotionEvent.ACTION_DOWN) {
        bird.setDrop(-15);

        if (wing_sound.isPlaying()) {
            wing_sound.seekTo(0);
        }
    }
}

```

```

        else {
            wing_sound.start();
        }
    }

    return true;
}

```

onTouchEvent(MotionEvent event) – כאשר יש לחיצה על המסך הציפור עולה למעלה ומתנגן סאונד אפקט.

```

public void draw(Canvas canvas) {
    super.draw(canvas);

    if (start) {
        bird.draw(canvas);

        for (int i = 0; i < sumPipe; i++) {
            if
            (this.bird.getRect().intersect(arrPipes.get(i).getRect()) ||
            bird.getY() - bird.getHeight() < 0 || bird.getY() + bird.getHeight()
            > Constants.SCREEN_HEIGHT) {
                bird.setWidth(0);
                bird.setHeight(0);

                Pipe.speed = 0;

                if (!hit_sound.isPlaying()) {
                    hit_sound.start();
                    hit_sound.setOnCompletionListener(new
                    MediaPlayer.OnCompletionListener() {
                        @Override
                        public void onCompletion(MediaPlayer
                    mediaPlayer) {
                            mediaPlayer.stop();
                            mediaPlayer.reset();
                        }
                    });
                }

                if (!die_sound.isPlaying()) {
                    die_sound.start();
                    die_sound.setOnCompletionListener(new
                    MediaPlayer.OnCompletionListener() {
                        @Override
                        public void onCompletion(MediaPlayer
                    mediaPlayer) {
                            mediaPlayer.stop();
                            mediaPlayer.reset();
                        }
                    });
                }

                MainActivity.current_score.setText("Current: " +
                Integer.toString(score));
                MainActivity.best_score.setText("Best: " +
                Integer.toString(best_score));
                MainActivity.score.setVisibility(View.INVISIBLE);
                MainActivity.rl_gameover.setVisibility(View.VISIBLE);
            }
        }
    }
}

```

```

    }

    if ((this.bird.getX() + this.bird.getWidth() >
arrPipes.get(i).getX() + arrPipes.get(i).getWidth() / 2) &&
(this.bird.getX() + this.bird.getWidth() <= arrPipes.get(i).getX() +
arrPipes.get(i).getWidth() / 2 + Pipe.speed) && (i < sumPipe / 2)) {
        score++;

        if (score_sound.isPlaying()) {
            score_sound.seekTo(0);
        }

        else {
            score_sound.start();
        }

        if (score > best_score) {
            best_score = score;

            SQLiteDatabase db =
MainActivity.dbHelper.getWritableDatabase();

            ContentValues values = new ContentValues();
            values.put(MainActivity.dbHelper.KEY_BESTSCORE,
best_score);

            String selection =
MainActivity.dbHelper.KEY_USERNAME + " = ?";
            String[] selectionArgs = {
sharedPreferences.getString("username", "") };

            db.update(MainActivity.dbHelper.TABLE_USERS,
values, selection, selectionArgs);

            db.close();
        }

        MainActivity.score.setText(Integer.toString(score));
    }

    if (this.arrPipes.get(i).getX() < -
arrPipes.get(i).getWidth()) {
        this.arrPipes.get(i).setX(Constants.SCREEN_WIDTH);

        if (i < sumPipe / 2) {
            arrPipes.get(i).randomY();
        }

        else {
            arrPipes.get(i).setY(this.arrPipes.get(i -
sumPipe / 2).getY() + this.arrPipes.get(i - sumPipe / 2).getHeight()
+ this.distance);
        }
    }

    this.arrPipes.get(i).draw(canvas);
}

}

else {
    if (bird.getY() > Constants.SCREEN_HEIGHT / 2) {

```

```

        bird.setDrop(-15 * Constants.SCREEN_HEIGHT / 1920);
    }

    bird.draw(canvas);
}

handler.postDelayed(r, 5);
}

```

draw(Canvas canvas) – כאן ה"קסם" מתחולל: תנועת הציפור, collision detection, ניגון סאונד אפקטים למיניהם, עדכון ה-best score במסד הנתונים לפי ה-user session וכו'.

```

public void reset() {
    MainActivity.score.setText("0");

    score = 0;

    initPipe();
    initBird();

    hit_sound = MediaPlayer.create(getContext(), R.raw.hit);
    die_sound = MediaPlayer.create(getContext(), R.raw.die);

    Activity activity = (Activity)getContext();
    MainActivity mainActivity = (MainActivity)activity;

    RelativeLayout layout =
mainActivity.findViewById(R.id.background);

    batteryReceiver = new BatteryReceiver(layout);
    IntentFilter filter = new
IntentFilter(Intent.ACTION_BATTERY_CHANGED);

getContext().getApplicationContext().registerReceiver(batteryReceiver
, filter);
}

```

reset() – מתרחש כאשר השחקן נפסל; מארגן את המשחק לסיבוב חוזר.

מדריך למשתמש

דרישות סף

האפליקציה נבדקה על Pixel 2 API 26 (אמולטור), Xiaomi Redmi Note 9 Pro (הפלאפון שלי) ו-Xiaomi Redmi Note 8 (פלאפון של חבר).

גרסת API מינימאלית – Lollipop (21), גרסת API רצויה – Snow Cone (32).

הרשאות: עבור Google Maps API –

```
<uses-permission  
android:name="android.permission.ACCESS_COARSE_LOCATION" />  
<uses-permission  
android:name="android.permission.ACCESS_FINE_LOCATION" />
```

עבור עדכון רקע לפי אחוז סוללה –

```
<receiver android:name=".BatteryReceiver" android:exported="true">  
  <intent-filter>  
    <action android:name="android.intent.action.BATTERY_CHANGED" />  
  </intent-filter>  
</receiver>
```

עבור מוזיקת רקע –

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

הצגת המשחק
התחלה (מסך פתיחה)

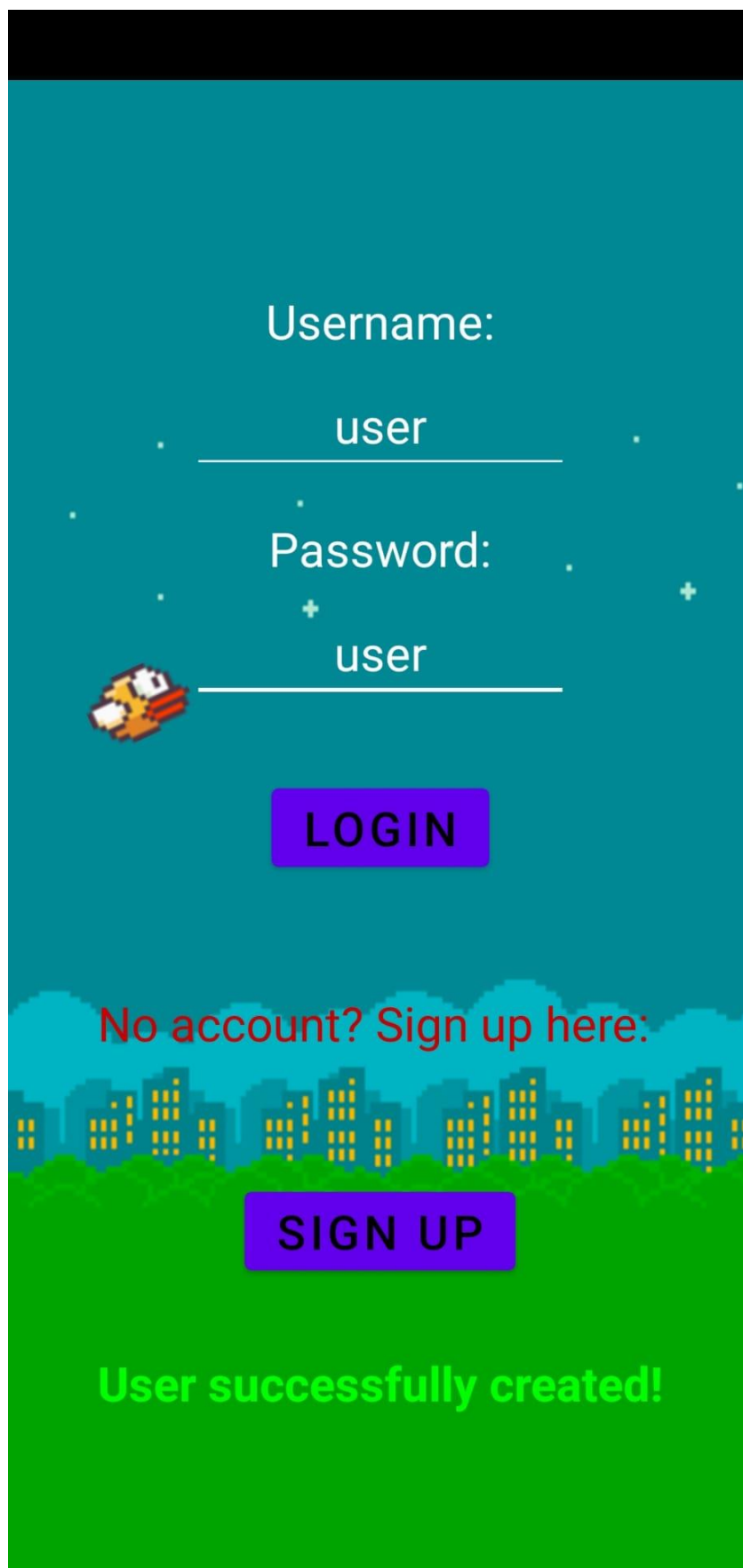
Username:

Password:

LOGIN

No account? Sign up here:

SIGN UP



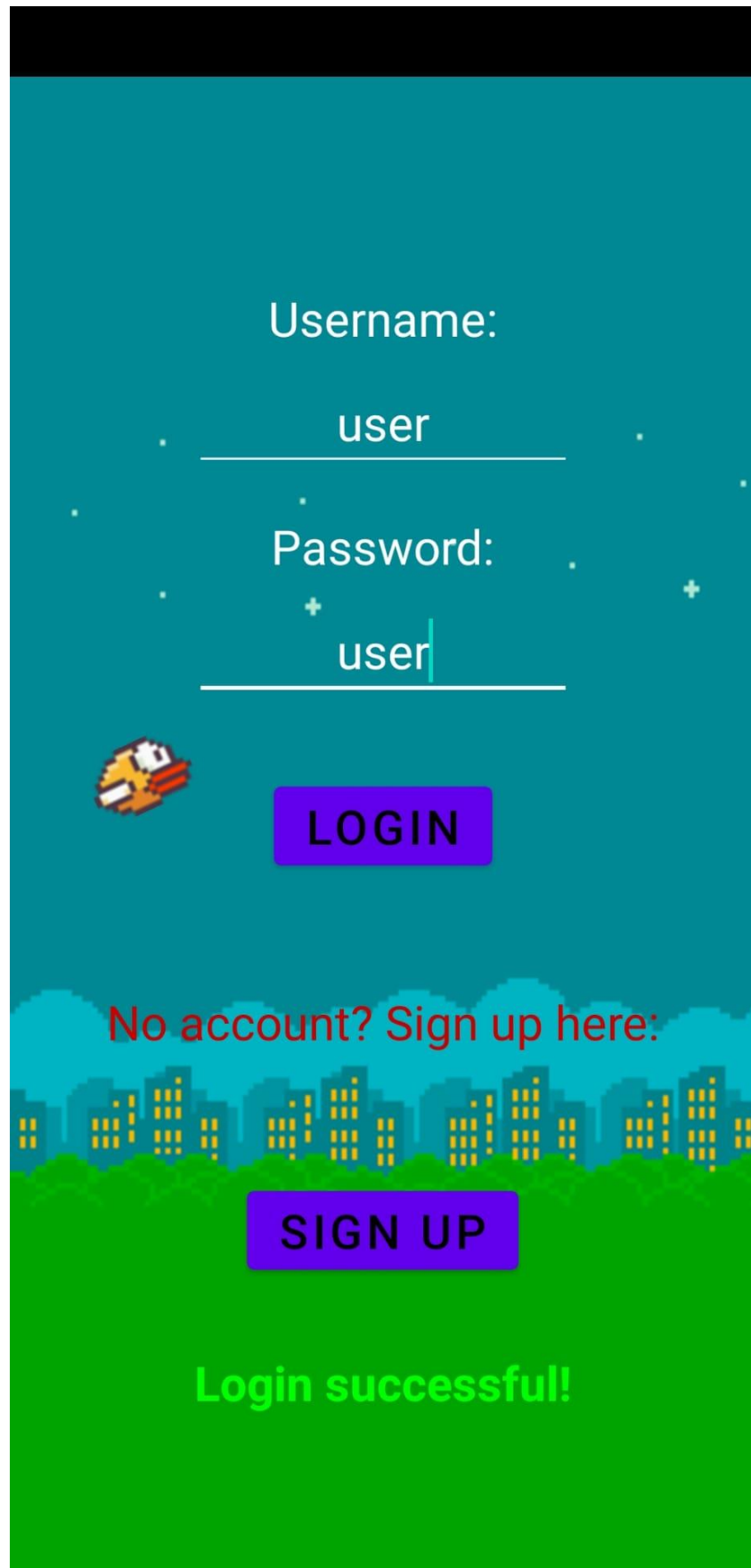
The image shows a mobile application interface for a game called 'Flappy Bird'. The background is a teal sky with small white stars and a city skyline with yellow-lit windows at the bottom. The interface includes a black header bar at the top. The main content area has a teal background with the following elements:

- Username:** A text label followed by an input field containing the text 'user'.
- Password:** A text label followed by an input field containing the text 'user'. To the left of the password field is a small pixel art character of a bird.
- LOGIN:** A purple rectangular button with the text 'LOGIN' in black.
- No account? Sign up here:** A red text link.
- SIGN UP:** A purple rectangular button with the text 'SIGN UP' in black.
- User successfully created!** A green text message at the bottom of the screen.

At the very bottom of the screen, there is a white bar containing three standard Android navigation icons: a square, a circle, and a triangle.

רפאל רייכרודל / פלאפי בירד

התחברות למשתמש

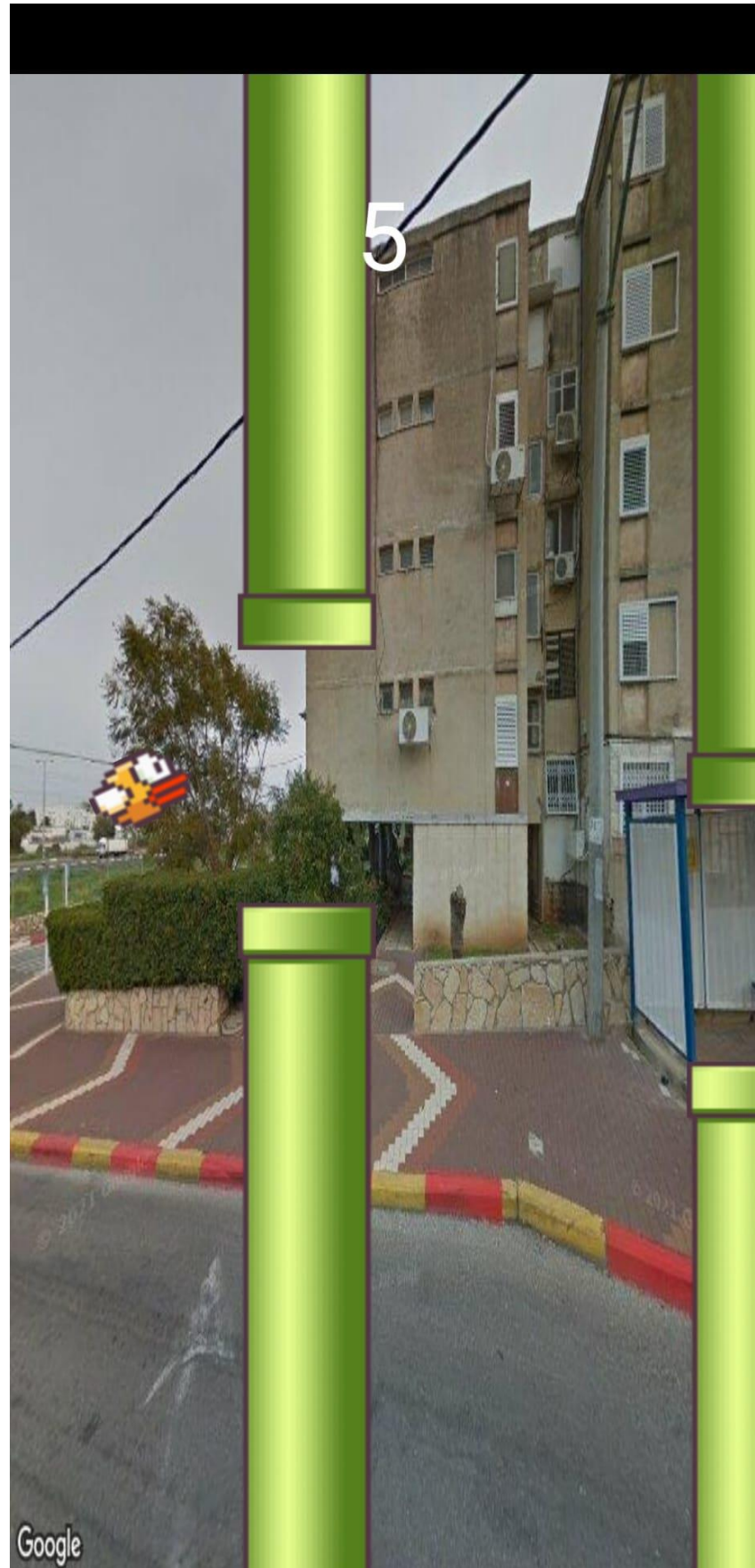


מסך קדם-משחק (עם לחיצה על כפתור location_btn)



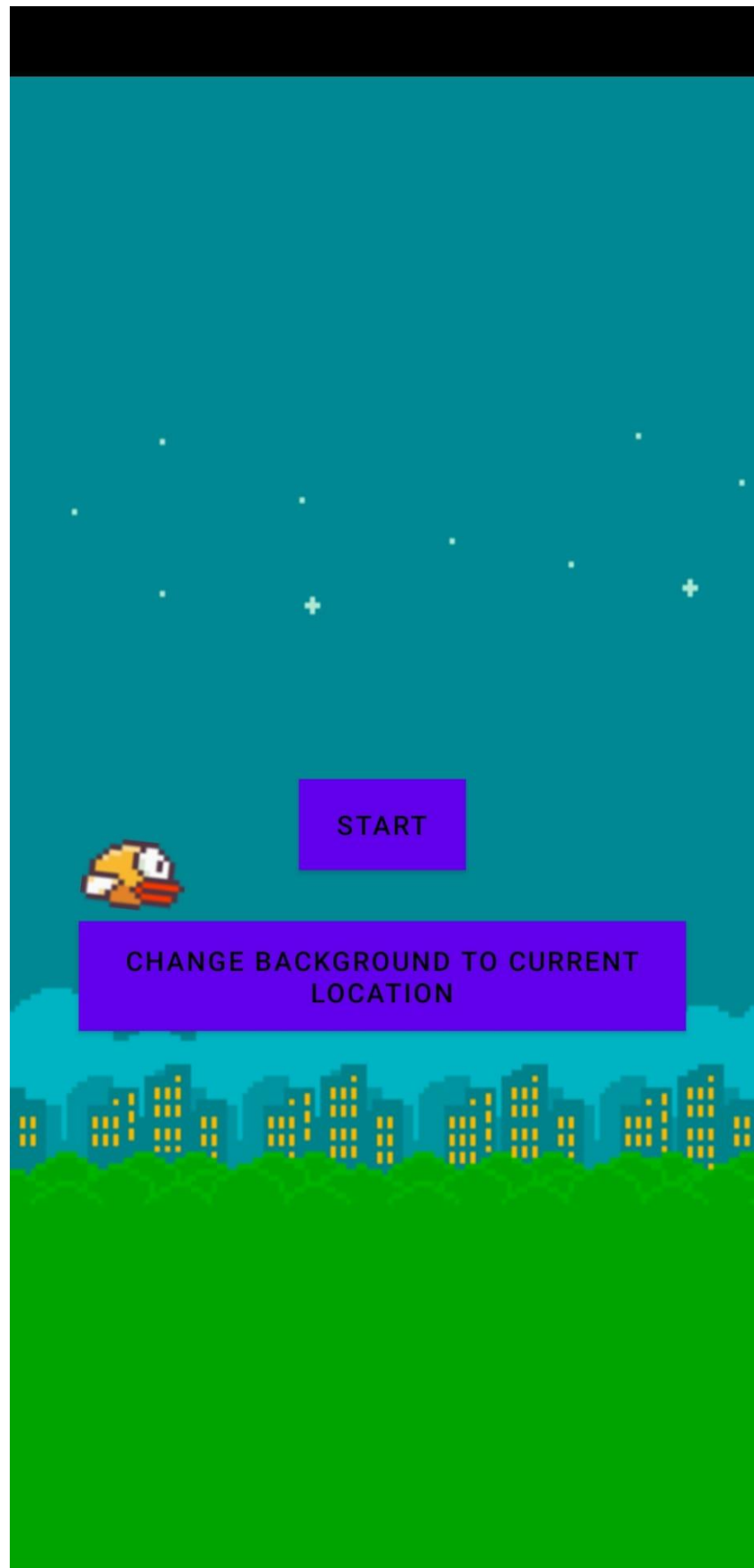
רפאל רייכרודל / פלאפי בירד

המשחק עצמו (מסך משחק)





מסך קדם-משחק שוב פעם לאחר פסילה



רפלקציה

במהלך הפיתוח של הפרויקט, יצאתי למסע מרתק ומאתגר שנפרש על פני מספר חודשים. הפרויקט כלל עיצוב והטמעה של משחק דומה ל-Flappy Bird באמצעות Android Studio. מתוך מחשבה על תהליך העבודה, ההצלחות, האתגרים והקשיים בהם נתקלתי, צברתי תובנות וחוויות חשובות שתרמו לצמיחתי כמפתח.

תהליך העבודה בפרויקט זה כלל מספר שלבים מרכזיים. בתחילה, התמקדתי בהבנת מכניקת המשחק של Flappy Bird ובמפוי הרכיבים והאינטראקציות הדרושים. שלב זה דרש ניתוח ותכנון קפדניים כדי להבטיח שכפול נאמן של תכונות הליבה של המשחק.

אחת ההצלחות הגדולות של פרויקט זה הייתה יישום מוצלח של פיזיקת המשחק. הדמיית תנועת הציפור, כוח המשיכה וזיהוי ההתנגשות של הציפור התבררה כמאתגרת, אבל הצלחתי להגיע לרמה מספקת של ריאליזם והיענות. באמצעות בדיקות איטרטיביות וכיוונון עדין, הצלחתי ליצור חווית משחק מהנה ומרתקת.

עם זאת, נתקלתי בכמה אתגרים במהלך תהליך הפיתוח. קושי משמעותי אחד היה טיפול באירועי מגע וקלט משתמש. לכידה ופירוש מחוות מגע בצורה מדויקת כדי לשלוט במעוף הציפור הצריכו בדיקה ועידון יסודיים. על ידי מינוף יכולות הטיפול באירועי מגע של Android Studio והתנסות בדפוס אינטראקציה שונים, התגברתי על האתגר הזה והשגתי בקורות חלקות ואינטואיטיביות.

אתגר נוסף היה יצירת והצבת המכשולים באופן דינמי ליצירת רמת קושי מגוונת ומאתגרת. נדרש מאמץ ניכר לעצב אלגוריתמים שיבטיחו שמיקום המכשולים יהיה הוגן וימנע חפיפה ומצבים לא פיזיים לשחקן. באמצעות איטרציה ובדיקות קפדניות, הצלחתי ליצור חווית משחק מאוזנת וקשה בהדרגה.

תהליך הלמידה במהלך פרויקט זה היה נרחב ומתגמל. העמקתי את ההבנה שלי בעקרונות פיתוח משחקים, חידדתי את כישורי פתרון הבעיות שלי, ושיפרתי את המיומנות שלי בתכנות Java וב-Android Studio. חקרתי באופן עצמאי משאבים רבים כדי להתגבר על אתגרים ספציפיים. למידה עצמית זו אפשרה לי להרחיב את הידע שלי וליישם מושגים חדשים ביעילות.

מבחינת כלים עתידיים, הייתי מעדיף לשלב מנוע משחק או פריימוורק בפרויקטים דומים. בעוד ש-Android Studio סיפק בסיס איתן, מינוף של מנוע משחק ייעודי כמו Unity או LibGDX יציע תכונות מתקדמות יותר, ייעל את הפיתוח וישפר את התאימות בין גרסאות אנדרואיד שונות.

במהלך הפרויקט, פנייה לעזרה ממומחים, השתתפות בקהילות מקוונות ושיתוף מידע עם עמיתים הוכיחו ערך רב. שיתוף פעולה עם מפתחים אחרים המתמודדים עם אתגרים דומים אפשר לי לקבל נקודות מבט חדשות, לגלות פתרונות חלופיים ולקבל משוב על בחירות היישום שלי. תהליך זה של שיתוף מידע ולמידת עמיתים העשיר מאוד את חווית הפיתוח שלי.

בדיעבד, יש תחומים שבהם הייתי מיישם חלקים מהפרויקט אחרת. לדוגמה, הייתי לוקח יותר בחשבון את ארכיטקטורת הקוד והארגון שלו מההתחלה. אימוץ מבנה מודולרי וניתן להרחבה היה מקל על תחזוקה קלה יותר ועל שיפורים עתידיים.

בהינתן משאבים נוספים, הייתי מתמקד בשיפור ההיבטים החזותיים של המשחק. השקעה בגרפיקה מקצועית ובאנימציות עשויה לשפר משמעותית את המשיכה האסתטית הכללית ואת חווית המשתמש.

שאלות ללימוד עצמי למפתחים אחרים לשקול:

כיצד ניתן למנף מנועי משחק או פריימוורקים כאלה ואחרים כדי לייעל את תהליכי פיתוח המשחק? באילו טכניקות אפשר להשתמש כדי לשפר את אופטימיזציית המשחק אף עוד יותר? כיצד ניתן לשפר את ארכיטקטורת הקוד והארגון כדי להקל על מדרגיות ותחזוקה? כיצד אפשר לשלב התערבות בתוך קהילות מקוונות ולמידת עמיתים כדי להאיץ את תהליך הלמידה ופתרון הבעיות שלך?

לסיכום, הפרויקט סיפק לי חוויות, אתגרים והזדמנויות לצמיחה כמפתח שלא יסולאו בפז. דרך המסע הזה, העמקתי את ההבנה שלי בעקרונות פיתוח משחקים, חידדתי את הכישורים הטכניים שלי וטיפחתי תשוקה ליצירת חוויות אינטראקטיביות ומהנות לכל שכבת גיל.

ביבליוגרפיה

להלן מקורות מידע ועזרים למיניהם שהשתמשתי בהם לאורך העבודה על הפרויקט:

- ChatGPT (עזר המון בבניית קוד, נתן ייעוץ מתאים לדיבוג וכו')
- Android Studio Developer Documentation
 - : Bitmap
<https://developer.android.com/reference/android/graphics/Bitmap>
 - : Rect
<https://developer.android.com/reference/android/graphics/Rect>
 - : Matrix
<https://developer.android.com/reference/android/opengl/Matrix>
 - : draw(Canvas canvas)
<https://developer.android.com/develop/ui/views/layout/custom-views/custom-drawing>
 - : Canvas
<https://developer.android.com/reference/android/graphics/Canvas>
 - : Handler
<https://developer.android.com/reference/android/os/Handler>
 - : Runnable
<https://developer.android.com/reference/java/lang/Runnable>
 - : Background Music & Sound Effects (MediaPlayer)
<https://developer.android.com/reference/android/media/MediaPlayer>
 - : Battery Broadcast Receiver
<https://developer.android.com/training/monitoring-device-state/battery-monitoring>
 - : Airplane Mode
<https://developer.android.com/reference/android/provider/Settings>
- StackOverflow
 - : Bitmap
<https://stackoverflow.com/questions/41859989/what-is-a-bitmap-and-a-bitfactory-and-why-do-people-use-it-with-animations>
 - : Rect
<https://stackoverflow.com/questions/3035692/how-to-convert-a-drawable-to-a-bitmap>
 - : Rect
<https://stackoverflow.com/questions/19342538/how-to-draw-a-rectangle-in-specific-postion-in-android>
 - : Matrix
<https://stackoverflow.com/questions/7344497/android-canvas-draw-rectangle>
 - : draw(Canvas canvas)
<https://stackoverflow.com/questions/45500156/why-do-we-use-matrix-and-canvas-in-android-studio>
 - : draw(Canvas canvas)
<https://stackoverflow.com/questions/41317884/android-canvas-not-drawing>
 - : Handler
<https://stackoverflow.com/questions/48763030/draw-method-not-working>
 - : Handler
<https://stackoverflow.com/questions/3035692/how-to-convert-a-drawable-to-a-bitmap>
 - : Handler
<https://stackoverflow.com/questions/58881744/handler-not-executing-in-android-studio>
 - : Runnable
<https://stackoverflow.com/questions/3072173/how-to-call-a-method-after-a-delay-in-android>
 - : Runnable
<https://stackoverflow.com/questions/1921514/how-to-run-a-runnable-thread-in-android-at-defined-intervals>
 - : Runnable
<https://stackoverflow.com/questions/48674522/android-studio-java-runnable-not-running>
 - : Runnable
<https://stackoverflow.com/questions/12366448/how-to-use-runnable-in-android>

- : Background Music & Sound Effects (MediaPlayer)
<https://stackoverflow.com/questions/8209858/android-background-music-service>
- : Battery Broadcast Receiver
<https://stackoverflow.com/questions/13228849/how-to-detect-when-the-battery-is-low-android>
- : Airplane Mode
<https://stackoverflow.com/questions/4319212/how-can-one-detect-airplane-mode-on-android>
- : Google Maps Platform Documentation
<https://developers.google.com/maps/documentation>
- : GitHub
<https://github.com/samuelpcust/flappy-bird-assets>
- וכמובן... המורים למדמ"ח והחברים לכיתה 😊

נספחים

להלן הקוד המלא של הפרויקט – MainActivity.java, GameView.java, activity_main.xml, BaseObject.java, Bird.java, Pipe.java, DatabaseClass.java, MusicService.java, BatteryReceiver.java, AndroidManifest.xml

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/background"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <com.example.rafaelreichrudel_flappybird.GameView
        android:id="@+id/game_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

    <TextView
        android:id="@+id/score"
        android:visibility="invisible"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@color/white"
        android:textSize="50dp"
        android:text="0"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="50dp"/>

    <Button
        android:id="@+id/start_btn"
        android:visibility="invisible"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Start"
        android:padding="5dp"
        android:layout_centerInParent="true"
        android:background="@color/white"
        android:textColor="@color/black"
        android:onClick="onClick"/>

    <Button
        android:id="@+id/location_btn"
        android:visibility="invisible"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Change Background to Current Location"
        android:padding="5dp"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="445dp"
        android:background="@color/white"
        android:textColor="@color/black"
        android:onClick="onClick"/>

    <RelativeLayout
        android:id="@+id/rl_login"
        android:visibility="visible"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true">
```



```

<TextView
    android:id="@+id/username_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="80dp"
    android:text="Username:"
    android:textColor="@color/white"
    android:textSize="25sp" />

<EditText
    android:id="@+id/username"
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="125dp"
    android:text=""
    android:textAlignment="center"
    android:textColor="@color/white"
    android:backgroundTint="@color/white"
    android:textSize="25sp" />

<TextView
    android:id="@+id/password_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="200dp"
    android:text="Password:"
    android:textColor="@color/white"
    android:textSize="25sp" />

<EditText
    android:id="@+id/password"
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="245dp"
    android:text=""
    android:textAlignment="center"
    android:textColor="@color/white"
    android:backgroundTint="@color/white"
    android:textSize="25sp" />

<Button
    android:id="@+id/login_btn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="675dp"
    android:layout_centerInParent="true"
    android:text="Login"
    android:textColor="@color/black"
    android:textSize="25sp"
    android:onClick="onClick" />

<TextView
    android:id="@+id/disclaimer"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="450dp"

```

```

        android:text="No account? Sign up here:"
        android:textColor="@android:color/holo_red_dark"
        android:textSize="25sp" />

<Button
    android:id="@+id/signup_btn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="550dp"
    android:layout_centerHorizontal="true"
    android:text="Sign Up"
    android:textColor="@color/black"
    android:textSize="25sp"
    android:onClick="onClick" />

<TextView
    android:id="@+id/error_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="640dp"
    android:layout_centerHorizontal="true"
    android:text=""
    android:textColor="#FF0000"
    android:textSize="25sp" />
</RelativeLayout>

<RelativeLayout
    android:id="@+id/rl_gameover"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:onClick="onClick"
    android:visibility="invisible">

    <ImageView
        android:id="@+id/gameover"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:src="@drawable/gameover" />

    <TextView
        android:id="@+id/current_score"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="440dp"
        android:text="Current: 0"
        android:textColor="@color/white"
        android:textSize="25sp" />

    <TextView
        android:id="@+id/best_score"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="480dp"
        android:text="Best: 0"
        android:textColor="@color/white"
        android:textSize="25sp" />
</RelativeLayout>
</RelativeLayout>

```

MainActivity.java

```

package com.example.rafaelreichrudel_flappybird;

import android.Manifest;
import android.content.ContentValues;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteConstraintException;
import android.database.sqlite.SQLiteException;
import android.graphics.Color;
import android.graphics.Typeface;
import android.graphics.drawable.BitmapDrawable;
import android.graphics.drawable.Drawable;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.media.Image;
import android.os.Bundle;

import com.google.android.gms.location.FusedLocationProviderClient;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.material.snackbar.Snackbar;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Handler;
import android.provider.CalendarContract;
import android.util.DisplayMetrics;
import android.util.Log;
import android.view.Display;
import android.view.View;

import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.navigation.NavController;
import androidx.navigation.Navigation;
import androidx.navigation.ui.AppBarConfiguration;
import androidx.navigation.ui.NavigationUI;

import com.example.rafaelreichrudel_flappybird.databinding.ActivityMainBinding;
import com.squareup.picasso.Callback;
import com.squareup.picasso.Picasso;

import android.view.Menu;
import android.view.MenuItem;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;

```

```
import android.widget.RelativeLayout;
import android.widget.TextView;

import org.w3c.dom.Text;

import java.util.List;
import java.util.Locale;
import java.util.Random;

public class MainActivity extends AppCompatActivity implements
View.OnClickListener {
    private AppBarConfiguration appBarConfiguration;
    private ActivityMainBinding binding;

    public static TextView score;

    public static Button login_btn, signup_btn, start_btn,
location_btn;
    public static EditText username, password;
    public static TextView error_handler;

    public static TextView current_score, best_score;

    public static RelativeLayout layout;
    public static RelativeLayout rl_login, rl_gameover;

    public static DatabaseClass dbHelper;
    private SharedPreferences sharedPreferences;
    private SharedPreferences.Editor editor;

    private BatteryReceiver batteryReceiver;

    private GameView game_view;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);

        DisplayMetrics dm = new DisplayMetrics();
this.getWindowManager().getDefaultDisplay().getMetrics(dm);
        Constants.SCREEN_WIDTH = dm.widthPixels;
        Constants.SCREEN_HEIGHT = dm.heightPixels;

        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        Intent musicService = new Intent(this, MusicService.class);
        startService(musicService);

        layout = (RelativeLayout) findViewById(R.id.background);

        batteryReceiver = new BatteryReceiver(layout);
        IntentFilter filter = new
IntentFilter(Intent.ACTION_BATTERY_CHANGED);
        getApplicationContext().registerReceiver(batteryReceiver,
filter);
    }
}
```

```

        score = findViewById(R.id.score);

        login_btn = findViewById(R.id.login_btn);
        signup_btn = findViewById(R.id.signup_btn);
        start_btn = findViewById(R.id.start_btn);
        location_btn = findViewById(R.id.location_btn);

        username = findViewById(R.id.username);
        password = findViewById(R.id.password);

        error_handler = findViewById(R.id.error_text);
        error_handler.setTypeface(null, Typeface.BOLD);

        current_score = findViewById(R.id.current_score);
        best_score = findViewById(R.id.best_score);

        rl_login = findViewById(R.id.rl_login);
        rl_gameover = findViewById(R.id.rl_gameover);

        dbHelper = new DatabaseClass(this);
        sharedPreferences = getSharedPreferences("UserSession",
Context.MODE_PRIVATE);
        editor = sharedPreferences.edit();

        editor.clear();
        editor.apply();

        game_view = findViewById(R.id.game_view);
    }

    @Override
    public void onClick(View view) {
        if (view == login_btn) {
            if (username.length() == 0 || password.length() == 0) {
                error_handler.setTextColor(Color.RED);
                error_handler.setText("Can't insert empty strings");
            }

            else if (sharedPreferences.getString("username",
"".length() > 0) {
                error_handler.setTextColor(Color.RED);
                error_handler.setText("User already logged in");
            }

            else {
                SQLiteDatabase db = dbHelper.getReadableDatabase();

                String[] projection = {
                    dbHelper.KEY_USERNAME,
                    dbHelper.KEY_PASSWORD
                };

                String selection = dbHelper.KEY_USERNAME + " = ? AND
" + dbHelper.KEY_PASSWORD + " = ?";
                String[] selectionArgs = {
                    username.getText().toString(), password.getText().toString()
                };

                Cursor cursor = db.query(
                    dbHelper.TABLE_USERS,
                    projection,
                    selection,

```

```

        selectionArgs,
        null,
        null,
        null
    );

    boolean result = cursor.getCount() > 0;

    if (result) {
        editor.putString("username",
username.getText().toString());
        editor.apply();

        error_handler.setTextColor(Color.GREEN);
        error_handler.setText("Login successful!");

        final Handler handler = new Handler();

        handler.postDelayed(new Runnable() {
            @Override
            public void run() {
                if
(sharedPreferences.getString("username", "") != null) {
                    SQLiteDatabase db =
MainActivity.dbHelper.getReadableDatabase();

                    String[] projection = {
MainActivity.dbHelper.KEY_BESTSCORE };

                    String selection =
MainActivity.dbHelper.KEY_USERNAME + " = ?";

                    String[] selectionArgs =
{sharedPreferences.getString("username", "")};

                    Cursor cursor = db.query(
MainActivity.dbHelper.TABLE_USERS,

                        projection,
                        selection,
                        selectionArgs,
                        null,
                        null,
                        null

                    );

                    if (cursor.moveToFirst()) {
                        game_view.best_score =
cursor.getInt(0);
                    }

                    cursor.close();
                    db.close();
                }

                start_btn.setVisibility(View.VISIBLE);
                location_btn.setVisibility(View.VISIBLE);
                rl_login.setVisibility(View.INVISIBLE);
            }
        }, 1000);
    }
}

```

```

        else {
            error_handler.setTextColor(Color.RED);
            error_handler.setText("User doesn't exist");
        }

        cursor.close();
        db.close();
    }
}

if (view == signup_btn) {
    if (username.length() == 0 || password.length() == 0) {
        error_handler.setTextColor(Color.RED);
        error_handler.setText("Can't insert empty strings");
    }

    else {
        SQLiteDatabase db = dbHelper.getWritableDatabase();

        ContentValues values_users = new ContentValues();

        values_users.put(dbHelper.KEY_USERNAME,
username.getText().toString());
        values_users.put(dbHelper.KEY_PASSWORD,
password.getText().toString());
        values_users.put(dbHelper.KEY_BESTSCORE, 0);

        try {
            db.insertOrThrow(dbHelper.TABLE_USERS, null,
values_users);
            error_handler.setTextColor(Color.GREEN);
            error_handler.setText("User successfully
created!");
        }

        catch (SQLiteConstraintException e) {
            error_handler.setTextColor(Color.RED);
            error_handler.setText("User already exists");
        }

        catch (SQLiteException e) {
            error_handler.setTextColor(Color.RED);
            error_handler.setText("Error occurred, try
again");
        }

        db.close();
    }
}

if (view == location_btn) {
    if (ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
        FusedLocationProviderClient fusedLocationClient =
LocationServices.getFusedLocationProviderClient(this);

        fusedLocationClient.getLastLocation().addOnSuccessListener(new
OnSuccessListener<Location>() {
            @Override

```

```

        public void onSuccess(Location location) {
            if (location != null) {
                DisplayMetrics displayMetrics = new
DisplayMetrics();

getWindowManager().getDefaultDisplay().getMetrics(displayMetrics);
                int screenWidth =
displayMetrics.widthPixels;
                int screenHeight =
displayMetrics.heightPixels;

                double latitude = location.getLatitude();
                double longitude =
location.getLongitude();

                String streetViewUrl =
"https://maps.googleapis.com/maps/api/streetview?size=" + screenWidth
+ "x" + screenHeight + "&location=" + latitude + "," + longitude +
"&fov=110&heading=0&pitch=0&key=API_KEY_HERE";

                ImageView imageView = new
ImageView(getApplicationContext());

Picasso.get().load(streetViewUrl).into(imageView, new Callback() {
                    @Override
                    public void onSuccess() {
                        Drawable drawable =
imageView.getDrawable();

                        if (drawable != null) {

layout.setBackground(drawable);
                        }
                    }

                    @Override
                    public void onError(Exception e) {
                        // Handle error by displaying a
message... not REALLY necessary
                    }
                });
            }
        }
    }

    } else {
        ActivityCompat.requestPermissions(this, new String[]
{ Manifest.permission.ACCESS_FINE_LOCATION }, 1);
    }
}

if (view == start_btn) {
    game_view.setStart(true);
    score.setVisibility(View.VISIBLE);
    start_btn.setVisibility(View.INVISIBLE);
    location_btn.setVisibility(View.INVISIBLE);
}

if (view == rl_gameover) {
    game_view.setStart(false);
    rl_gameover.setVisibility(View.INVISIBLE);
    start_btn.setVisibility(View.VISIBLE);
}

```



```

        location_btn.setVisibility(View.VISIBLE);
        game_view.reset();
    }
}

```

GameView.java

```

package com.example.rafaelreichrudel_flappybird;

import static android.content.Context.MODE_PRIVATE;

import android.app.Activity;
import android.content.ContentValues;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.SharedPreferences;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Rect;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.media.SoundPool;
import android.provider.Settings;
import android.util.AttributeSet;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;

import androidx.annotation.Nullable;

import android.content.Context;

import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.util.ArrayList;
import android.os.Handler;
import android.widget.RelativeLayout;

public class GameView extends View {
    private Bird bird;
    private Handler handler;
    private Runnable r;

    private ArrayList<Pipe> arrPipes;
    private int sumPipe, distance;

    public int score, best_score;

```

```

private boolean start;

private BatteryReceiver batteryReceiver;

MediaPlayer wing_sound = MediaPlayer.create(getContext(),
R.raw.wing);
MediaPlayer score_sound = MediaPlayer.create(getContext(),
R.raw.point);
MediaPlayer hit_sound = MediaPlayer.create(getContext(),
R.raw.hit);
MediaPlayer die_sound = MediaPlayer.create(getContext(),
R.raw.die);

private SharedPreferences sharedPreferences =
getContext().getSharedPreferences("UserSession",
Context.MODE_PRIVATE);

public GameView(Context context, @Nullable AttributeSet attrs) {
    super(context, attrs);

    score = 0;

    start = false;

    initBird();
    initPipe();

    handler = new Handler();

    r = new Runnable() {
        @Override
        public void run() {
            invalidate();
        }
    };
}

private void initBird() {
    bird = new Bird();
    bird.setWidth(150 * (Constants.SCREEN_WIDTH / 1080));
    bird.setHeight(100 * (Constants.SCREEN_HEIGHT / 1920));
    bird.setX(100 * (Constants.SCREEN_WIDTH / 1080));
    bird.setY(Constants.SCREEN_HEIGHT / 2 - bird.getHeight() /
2);

    ArrayList<Bitmap> arrBms = new ArrayList<>();

    arrBms.add(BitmapFactory.decodeResource(this.getResources(),
R.drawable.yellowbird_upflap));
    arrBms.add(BitmapFactory.decodeResource(this.getResources(),
R.drawable.yellowbird_midflap));
    arrBms.add(BitmapFactory.decodeResource(this.getResources(),
R.drawable.yellowbird_downflap));

    if (android.os.Build.VERSION.SDK_INT >=
android.os.Build.VERSION_CODES.JELLY_BEAN_MR1) {
        if
(Settings.Global.getInt(getContext().getContentResolver(),
Settings.Global.AIRPLANE_MODE_ON, 0) != 0) {
            for (int i = 0; i < arrBms.size(); i++) {

```

```

        arrBms.set(i,
BitmapFactory.decodeResource(this.getResources(),
R.drawable.airplane));
    }

    }

    else {
        if
(Settings.System.getInt(getContext().getContentResolver(),
Settings.System.AIRPLANE_MODE_ON, 0) != 0) {
            for (int i = 0; i < arrBms.size(); i++) {
                arrBms.set(i,
BitmapFactory.decodeResource(this.getResources(),
R.drawable.airplane));
            }
        }
    }

    bird.setArrBms(arrBms);
}

private void initPipe() {
    sumPipe = 4;
    distance = 325 * Constants.SCREEN_HEIGHT / 1920;
    arrPipes = new ArrayList<Pipe>();

    for (int i = 0; i < sumPipe; i++) {
        if (i < sumPipe / 2) {
            this.arrPipes.add(new Pipe(Constants.SCREEN_WIDTH + i
* ((Constants.SCREEN_WIDTH + 200 * Constants.SCREEN_WIDTH / 1080) /
(sumPipe / 2)), 0, 200 * Constants.SCREEN_WIDTH / 1080,
Constants.SCREEN_HEIGHT / 2));
            this.arrPipes.get(this.arrPipes.size() -
1).setBm(BitmapFactory.decodeResource(this.getResources(),
R.drawable.pipe_green_upside_down));
            this.arrPipes.get(this.arrPipes.size() -
1).randomY();
        }

        else {
            this.arrPipes.add(new Pipe(this.arrPipes.get(i -
sumPipe / 2).getX(), this.arrPipes.get(i - sumPipe / 2).getY() +
this.arrPipes.get(i - sumPipe / 2).getHeight() + this.distance, 200 *
Constants.SCREEN_WIDTH / 1080, Constants.SCREEN_HEIGHT / 2));
            this.arrPipes.get(this.arrPipes.size() -
1).setBm(BitmapFactory.decodeResource(this.getResources(),
R.drawable.pipe_green));
        }
    }
}

@Override
public boolean onTouchEvent(MotionEvent event) {
    if (event.getAction() == MotionEvent.ACTION_DOWN) {
        bird.setDrop(-15);

        if (wing_sound.isPlaying()) {
            wing_sound.seekTo(0);
        }
    }
}

```

```

        else {
            wing_sound.start();
        }
    }

    return true;
}

public void draw(Canvas canvas) {
    super.draw(canvas);

    if (start) {
        bird.draw(canvas);

        for (int i = 0; i < sumPipe; i++) {
            if
            (this.bird.getRect().intersect(arrPipes.get(i).getRect()) ||
            bird.getY() - bird.getHeight() < 0 || bird.getY() + bird.getHeight()
            > Constants.SCREEN_HEIGHT) {
                bird.setWidth(0);
                bird.setHeight(0);

                Pipe.speed = 0;

                if (!hit_sound.isPlaying()) {
                    hit_sound.start();
                    hit_sound.setOnCompletionListener(new
                    MediaPlayer.OnCompletionListener() {
                        @Override
                        public void onCompletion(MediaPlayer
                    mediaPlayer) {
                            mediaPlayer.stop();
                            mediaPlayer.reset();
                        }
                    });
                }

                if (!die_sound.isPlaying()) {
                    die_sound.start();
                    die_sound.setOnCompletionListener(new
                    MediaPlayer.OnCompletionListener() {
                        @Override
                        public void onCompletion(MediaPlayer
                    mediaPlayer) {
                            mediaPlayer.stop();
                            mediaPlayer.reset();
                        }
                    });
                }

                MainActivity.current_score.setText("Current: " +
                Integer.toString(score));
                MainActivity.best_score.setText("Best: " +
                Integer.toString(best_score));
                MainActivity.score.setVisibility(View.INVISIBLE);
                MainActivity.rl_gameover.setVisibility(View.VISIBLE);
            }

            if ((this.bird.getX() + this.bird.getWidth() >
            arrPipes.get(i).getX() + arrPipes.get(i).getWidth() / 2) &&

```

```

        (this.bird.getX() + this.bird.getWidth() <= arrPipes.get(i).getX() +
arrPipes.get(i).getWidth() / 2 + Pipe.speed) && (i < sumPipe / 2)) {
            score++;

            if (score_sound.isPlaying()) {
                score_sound.seekTo(0);
            }

            else {
                score_sound.start();
            }

            if (score > best_score) {
                best_score = score;

                SQLiteDatabase db =
MainActivity.dbHelper.getWritableDatabase();

                ContentValues values = new ContentValues();

values.put(MainActivity.dbHelper.KEY_BESTSCORE, best_score);

                String selection =
MainActivity.dbHelper.KEY_USERNAME + " = ?";
                String[] selectionArgs = {
sharedPreferences.getString("username", "") };

                db.update(MainActivity.dbHelper.TABLE_USERS,
values, selection, selectionArgs);

                db.close();
            }

MainActivity.score.setText(Integer.toString(score));
        }

        if (this.arrPipes.get(i).getX() < -
arrPipes.get(i).getWidth()) {
this.arrPipes.get(i).setX(Constants.SCREEN_WIDTH);

            if (i < sumPipe / 2) {
                arrPipes.get(i).randomY();
            }

            else {
                arrPipes.get(i).setY(this.arrPipes.get(i -
sumPipe / 2).getY() + this.arrPipes.get(i - sumPipe / 2).getHeight()
+ this.distance);
            }
        }

        this.arrPipes.get(i).draw(canvas);
    }
}

else {
    if (bird.getY() > Constants.SCREEN_HEIGHT / 2) {
        bird.setDrop(-15 * Constants.SCREEN_HEIGHT / 1920);
    }
}

```

```

        bird.draw(canvas);
    }

    handler.postDelayed(r, 5);
}

public boolean isStart() {
    return start;
}

public void setStart(boolean start) {
    this.start = start;
}

public void reset() {
    MainActivity.score.setText("0");

    score = 0;

    initPipe();
    initBird();

    hit_sound = MediaPlayer.create(getContext(), R.raw.hit);
    die_sound = MediaPlayer.create(getContext(), R.raw.die);

    Activity activity = (Activity)getContext();
    MainActivity mainActivity = (MainActivity)activity;

    RelativeLayout layout =
mainActivity.findViewById(R.id.background);

    batteryReceiver = new BatteryReceiver(layout);
    IntentFilter filter = new
IntentFilter(Intent.ACTION_BATTERY_CHANGED);

getContext().getApplicationContext().registerReceiver(batteryReceiver
, filter);
}
}

```

BaseObject.java

```

package com.example.raphaelreichrudel_flappybird;

import android.graphics.Bitmap;
import android.graphics.Rect;

public class BaseObject {
    protected float x, y;
    protected int width, height;
    protected Bitmap bm;
    protected Rect rect;

    public BaseObject() {

    }

    public BaseObject(float x, float y, int width, int height) {
        this.x = x;
    }
}

```

```

        this.y = y;
        this.width = width;
        this.height = height;
    }

    public float getX() {
        return x;
    }

    public void setX(float x) {
        this.x = x;
    }

    public float getY() {
        return y;
    }

    public void setY(float y) {
        this.y = y;
    }

    public int getWidth() {
        return width;
    }

    public void setWidth(int width) {
        this.width = width;
    }

    public int getHeight() {
        return height;
    }

    public void setHeight(int height) {
        this.height = height;
    }

    public Bitmap getBm() {
        return bm;
    }

    public void setBm(Bitmap bm) {
        this.bm = bm;
    }

    public Rect getRect() {
        return new Rect((int) (this.x), (int) (this.y), (int) (this.x +
this.width), (int) (this.y + this.height));
    }

    public void setRect(Rect rect) {
        this.rect = rect;
    }
}

```

Bird.java

```

package com.example.rafaelreichrudel_flappybird;

import android.graphics.Bitmap;

```

```

import android.graphics.Canvas;
import android.graphics.Matrix;

import java.util.ArrayList;

public class Bird extends BaseObject {
    private int count, vFlap, idCurrentBitmap;
    private float drop;
    private ArrayList<Bitmap> arrBms = new ArrayList<>();

    public Bird() {
        this.count = 0;
        this.vFlap = 5;
        this.idCurrentBitmap = 0;
        this.drop = 0;
    }

    public ArrayList<Bitmap> getArrBms () {
        return arrBms;
    }

    public void setArrBms(ArrayList<Bitmap> arrBms) {
        this.arrBms = arrBms;

        for (int i = 0; i < arrBms.size(); i++) {
            this.arrBms.set(i,
                Bitmap.createScaledBitmap(this.arrBms.get(i), this.width,
                this.height, true));
        }
    }

    @Override
    public Bitmap getBm() {
        count++;

        if (this.count == this.vFlap) {
            for (int i = 0; i < this.arrBms.size(); i++) {
                if (i == arrBms.size() - 1) {
                    this.idCurrentBitmap = 0;
                    break;
                }

                else if (this.idCurrentBitmap == i) {
                    idCurrentBitmap = i + 1;
                    break;
                }
            }

            count = 0;
        }

        if (this.drop < 0) {
            Matrix matrix = new Matrix();
            matrix.postRotate(-25);

            return Bitmap.createBitmap(arrBms.get(idCurrentBitmap),
            0, 0, arrBms.get(idCurrentBitmap).getWidth(),
            arrBms.get(idCurrentBitmap).getHeight(), matrix, true);
        }

        else if (this.drop >= 0) {

```



```

        Matrix matrix = new Matrix();

        if (this.drop < 70) {
            matrix.postRotate(-25 + (drop * 2));
        }

        else {
            matrix.postRotate(45);
        }

        return Bitmap.createBitmap(arrBms.get(idCurrentBitmap),
0, 0, arrBms.get(idCurrentBitmap).getWidth(),
arrBms.get(idCurrentBitmap).getHeight(), matrix, true);
    }

    return this.getArrBms().get(idCurrentBitmap);
}

public void drop() {
    this.drop += 0.6;
    this.y += this.drop;
}

public float getDrop() {
    return drop;
}

public void setDrop(float drop) {
    this.drop = drop;
}

public void draw(Canvas canvas) {
    drop();
    canvas.drawBitmap(this.getBm(), this.x, this.y, null);
}
}

```

Pipe.java

```

package com.example.raphaelreichrudel_flappybird;

import android.graphics.Bitmap;
import android.graphics.Canvas;

import java.util.Random;

public class Pipe extends BaseObject {
    public static int speed;

    public Pipe(float x, float y, int width, int height) {
        super(x, y, width, height);
        this.speed = 10 * Constants.SCREEN_WIDTH / 1080;
    }

    public void draw(Canvas canvas) {
        this.x -= this.speed;
        canvas.drawBitmap(this.bm, this.x, this.y, null);
    }

    public void randomY() {

```

```

        Random r = new Random();
        this.y = r.nextInt((0 + this.height / 4 + 1)) - this.height /
4;
    }

    @Override
    public void setBm(Bitmap bm) {
        this.bm = Bitmap.createScaledBitmap(bm, this.width,
this.height, true);
    }
}

```

DatabaseClass.java

```

package com.example.rafaelreichrudel_flappybird;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DatabaseClass extends SQLiteOpenHelper {
    public DatabaseClass(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(CREATE_TABLE_USERS);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_USERS);
        onCreate(db);
    }

    private static final int DATABASE_VERSION = 1;
    private static final String DATABASE_NAME = "my_database.db";

    public static final String TABLE_USERS = "users";

    public static final String KEY_USERNAME = "username";
    public static final String KEY_PASSWORD = "password";
    public static final String KEY_BESTSCORE = "bestscore";

    public static final String CREATE_TABLE_USERS = "CREATE TABLE "
        + TABLE_USERS + "(" + KEY_USERNAME + " TEXT PRIMARY KEY,"
" + KEY_PASSWORD
        + " TEXT NOT NULL," + KEY_BESTSCORE + " INTEGER NOT
NULL)";
}

```

MusicService.java

```

package com.example.rafaelreichrudel_flappybird;

import android.app.Activity;
import android.app.Service;
import android.content.Intent;

```

```
import android.media.MediaPlayer;
import android.os.IBinder;

public class MusicService extends Service {
    private static final String TAG = null;
    MediaPlayer player;

    public IBinder onBind(Intent arg0) {
        return null;
    }

    @Override
    public void onCreate() {
        super.onCreate();
        player = MediaPlayer.create(this, R.raw.music);
        player.setLooping(true); // Set looping
        player.setVolume(100,100);
    }

    public int onStartCommand(Intent intent, int flags, int startId)
    {
        player.start();
        return Service.START_STICKY;
    }

    protected void onPause() {

    }

    protected void onResume() {

    }

    @Override
    public void onDestroy() {
        player.stop();
        player.release();
    }
}
```

BatteryReceiver.java

```
package com.example.raphaelreichrudel_flappybird;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.BatteryManager;
import android.util.Log;
import android.view.View;
import android.widget.Toast;

import java.util.Random;

public class BatteryReceiver extends BroadcastReceiver {
    static final int SET_PERCENTAGE = 80;

    private View rootView;

    public BatteryReceiver(View rootView) {
        this.rootView = rootView;
    }
}
```

```

    }

    @Override
    public void onReceive(Context context, Intent intent) {
        int batteryLevel =
intent.getIntExtra(BatteryManager.EXTRA_LEVEL, -1);
        int batteryScale =
intent.getIntExtra(BatteryManager.EXTRA_SCALE, -1);
        float batteryPercent = (batteryLevel / (float)batteryScale) *
100;

        if (batteryPercent <= SET_PERCENTAGE) {
rootView.setBackgroundResource(R.drawable.background_low_battery);
        }

        else {
            Random r = new Random();
            int roll = r.nextInt(2);

            if (roll == 0) {
rootView.setBackgroundResource(R.drawable.background_day);
            }

            else {
rootView.setBackgroundResource(R.drawable.background_night);
            }
        }
    }
}

```

AndroidManifest.xml

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.raphaelreichrudel_flappybird">

    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@drawable/yellowbird_midflap"
        android:label="@string/app_name"
        android:roundIcon="@drawable/yellowbird_midflap"
        android:supportRtl="true"
        android:theme="@style/Theme.AppCompat.NoActionBar"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:screenOrientation="sensorPortrait"

```

```

android:theme="@style/Theme.RaphaelReichrudel_FlappyBird.NoActionBar"
>
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category
android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

    <receiver android:name=".BatteryReceiver"
android:exported="true">
        <intent-filter>
            <action
android:name="android.intent.action.BATTERY_CHANGED" />
        </intent-filter>
    </receiver>

    <service android:enabled="true" android:name=".MusicService"
/>
</application>
</manifest>

```