בית הספר "תיכון המושבה"

"Flappy Bird"

שם תלמיד: אורון רוזנבלט

ת"ז: 331093559

שם מנחה: ענבר מור

בית ספר: תיכון המושבה, זכרון יעקב

תאריך הגשה: 20.05.25

עבודת גמר תכנון ותכנות מערכות 883599

תוכן עניינים

דרישות חובה עבור פרויקט 5 יח"ל

סעיף 6: יצירת אנימציה (לא כולל שימוש במחלקת אנימציה).

סעיף 7: אחסון וטיפול בנתונים בעזרת Firebase

סעיף 9: אפליקציה מרובת משתתפים

שימוש ב- SurfaceView

שימוש ב- Threads

מבוא

תיאור תכולת הספר

בספר הפרויקט שלי יוצג תהליך העבודה שלי על הפרויקט, החל מבחירת הנושא ועד לקשיים שחוויתי במהלך העבודה על הפרויקט. אפרט על מבנה האפליקציה ועל אופן השימוש בה, בנוסף לממשק הפעולות והמחלקות שבונות את האפליקציה.

רקע לפרויקט

שם הפרויקט הינו "Flappy Bird". מדובר במשחק מסוג ארקייד שפותח בוייטנאם שיצא לאור בשנת 2013. במשחק מתוארת ציפור אשר קופצת בין צינורות, מטרת השחקן לגרום לציפור להתחמק מהצינורות בעזרת לחיצה על המסך, כל לחיצה גורמת לציפור לקפוץ "flap"ובכך השחקן שולט במיקומה על המסך. כיום קיימות עשרות גרסאות של המשחק והוא נחשב לאחד מן המשחקים הכי פופולאריים בעולם. קהל היעד של המשחק הינו ילדים אך כל אדם בכל גיל ייהנה מהמשחק.

מאז שאני צעיר התעניינתי מאוד במשחקי וידאו ולכן חשבתי שלהכין אחד כזה יכול להיות חוויה משמעותית עבורי, לכן בחרתי במשחק פלאפי בירד, משחק אשר כמעט כל אחד מכיר ויודע איך לשחק בו. למשחק יש הרבה גרסאות ורציתי להכין אחת משלי אשר כוללת את כל מה שהיה לי חסר במשחק המקורי. לאפליקציה שיצרתי יש 2 אפשרויות משחק: שחקן אחד ושני שחקנים. באפשרות של שחקן אחד מטרת המשתמש להשיג כמה שיותר נקודות בכך ששורד כמה שיותר זמן מבלי להיתקע בצינורות. באפשרות של שני שחקנים ניתן להתחרות נגד חברים בתחרות של מי ישרוד הכי הרבה זמן. לאפליקציה הוספתי אפשרות לשנות את סוג הציפור איתה משחקים ואפשרות לשנות את הרקע. בנוסף ניתן לרשום את שם השחקן ובכך לשמור את הישגיו במשחק של שחקן יחיד.

תהליך מחקר

תחום הידע שלי על המשחק נובע מכך ששיחקתי במשחק הזה מספר פעמים לאורך השנים. אני יודע איך המשחק האמיתי נראה ומתפקד ולכן הידע שלי בנושא הוא רחב.

במהלך העבודה חקרתי רבות על הנושא של מימוש אנימציה בקוד והנושא של FireBase. לפני העבודה על הפרויקט לא היה לי כלל ידע בנושאים אלו ולכן בכדי לייצר את הפרויקט הייתי חייב להכיר את הנושאים האלה בצורה מעמיקה. למדתי איך להכין אנימציה בשימוש ב-Bitmap ו-SurfaceView ובכך יצירת האנימציה במשחק הייתה לי יותר קלה. למדתי איך להשתמש ב-Database למטרת העברת נתונים בין אמולטורים שונים ובכך יצרתי את המשחק בין 2 שחקנים. בדקתי גרסאות נוספות הקיימות של המשחק בשביל לקבל השראה לאופן התצוגה והתכונות אשר קיימות במשחק. מהן קיבלתי את הרעיון להוסיף שינוי ציפור ושינוי רקע. חקרתי לעומק את כל הנושאים האלו ובכך הצלחתי להכין את המשחק בדרך הייחודית שלי תוך כדי שמירה על מאפיינים קיימים.

אתגרים מרכזיים

צרכים שעליהם עונה הפרויקט

הפרויקט שלי הינו אפליקציית משחק, כמו כל משחק מטרתו בשביל הנאה והפגת השעמום של המשתמש. אך בניגוד למשחקים אחרים פלאפי בירד מעניק תחושה הישג בגלל הנקודות שצוברים וגם יכול לשמש כמשחק תחרותי בין חברים. המשחק מקרב בין אנשים שאוהבים תחרות חברותית ומהנה.

בעיות בפיתוח הפרויקט

קושי שנתקלתי בו במהלך העבודה על הפרויקט היה ליצור התנגשות חלקה בין הציפור לבין הצינורות, התנגשות זו מהווה כמקור הפסילה העיקרי. מכיוון שתמונת הציפור אינה מלבנית אלא עגולה, נוצר קושי בקליטה מתי הציפור באמת נוגעת בצינור. בעיה זו גורמת לפסילה במשחק למרות שנראה לעין כי הציפור אינה נגעה בצינור כלל.

קושי נוסף שנתקלתי בו היה במהלך העבודה על המשחק בין 2 שחקנים. נוצר דיליי בגלל הזמן שלוקח להעביר נתונים ל-FireBase ועד שהאמולטור השני קולט את אותם נתונים ומשתמש בהם במשחק של השחקן השני. בעיה זו גורמת לשני השחקנים לראות שהציפור של השחקן האחר זזה לאט, בנוסף לכך שציפור אחת מגיעה לצינורות מהר יותר מהשנייה ובכך יש יתרון לשחקן אחד על פני השחקן האחר, יתרון ששחקן אחד יכול להיפסל לפני שלשחקן השני הייתה בכלל הזדמנות להיפסל.

פתרונות לבעיות

כדי לפתור את הבעיה הנוגעת להתנגשות הציפור בצינורות, שיניתי את התמונה של הציפור. הקטנתי את הגבולות של התמונה כך שהמלבן המקיף אותה יקיף את הציפור העגולה באופן הכי מדויק שאפשר ובכך הוקטנה משמעותית כמות הפסילות החסרות משמעות והמשחק נראה כחלק יותר. הקטנת הגבולות בהתחלה נראתה כפתרון אינו יעיל מכיוון שהקטנת הגבולות תקטין גם את הציפור ובכך המשחק יראה פחות טוב. הדבר אכן קרה אך תיקנתי את זה בעזרת הגדלת תוכן התמונה ולא את התמונה עצמה.

כדי לפתור את הבעיה של הדיליי שנוצרה בגלל האיטיות של העברת המשתנים ב-FireBase, השתמשתי בטיימר. הפעלתי את הטיימר לשני השחקנים באותו זמן. העברתי את הנתונים ב-FireBase ואכסנתי אותם במשתנה מסוים. במקום ישר להשתמש במשתנה הזה כמו שעשיתי קודם, כעת אני מחכה עד שהטיימר יגיע למספר מסוים ורק אז הקוד משתמש במשתנה שהועבר ב-FireBase. דבר זה הפחית בצורה משמעותית את הדיליי במשחק וכעת נראה כי כמעט ואין הפרשים בין השחקנים. העברת הנתונים ב-FireBase עדיין לוקחת זמן אך כעת בגלל הטיימר הזמן הזה אינו משנה בגלל שמשתמשים בנתונים אלה מאוחר יותר.

מבנה הפרויקט

מחלקות הפרויקט שלי והאינטראקציה ביניהן מאורגנות בשיטת MVC

MVC היא תבנית של ארכיטקטורת תוכנה המשמשת למימוש של מערכות עתירות ממשק משתמש. התפיסה מבוססת על הפרדה בין הייצוג הפנימי של המידע (Model) לממשק המשתמש (View).

Model –manager, classes

View –activity, layout

Controller – adapters, handlers

הוצאת ושמירת נתונים ב - DataBase

Database הוא כלי יעיל המשמש לאחסון של מידע ונתונים במחשב, ובעת הצורך ניתן לשמור או להוציא מן הDB את הנתונים ולעשות בהם שימוש. בפרויקט שלי השתמשתי באמצעי אחסון מסוג FireBase על מנת לשמור את נתוני המשתמשים, עדכוני ריצה, קבוצות ריצה ומסלולי ריצה.

מחלקות ראשיות

קיימת מחלקה בשם GameView אשר מנהלת את כל המשחק הרגיל. מחלקה זו בעצם מתפעלת את כל המשחק. היא מציירת את הציפור, הצינורות, הרקע, הנקודות. המחלקה הזו מפעילה את הלופ המרכזי אשר מגריל את מיקום הציפורים, מצייר את המסך, מזיז את הצינורות והציפור, בודק אם התקיימה פסילה. מחלקה זו מייצרת את המשחק וגורמת לו להתקיים לפי הכללים. זוהי המחלקה הכי חשובה בכל הפרויקט. המשתנים בה כוללים את התמונות של הציפור והצינורות, משתנים הבודקים אם המשחק נגמר, רשימה של הצינורות על המסך, גדלי המסך וכל משתנה אחר אשר עוזר למשחק להתקיים.

קיימת מחלקה בשם Bird אשר שומרת את הנתונים על הציפור המשחקת. ניתן לשמור את ערך ה-Y של הציפור ובכך להיעזר בו בהתנגשויות בין הציפור לצינורות או הקרקע. התכונות במרכזיות במחלקה הן: 1. birdVelocity משתנה מסוג int אשר מכיל את מהירות הציפור. בעזרת משתנה זה ניתן ליצור אנימציית נפילה חלקה לפי חוקי הגרביטציה. 2. GRAVITY משתנה קבוע מסוג double אשר מבטא את עוצמת הגרביטציה על הציפור. בכל פעם מגדילים את הגודל של birdVelocity בגודל של GRAVITY ובכך תנועת הציפור היא מעין פרבולה כלפי מטה. 3. BIRD_JUMP משתנה קבוע מסוג int אשר מכיל את גודל ההתנגדות לגרביטציה. עבודת המשתנה הזה היא כמו של GRAVITY רק שבמקום לייצר תנועה פרבולית כלפי מטה הוא מייצר אותה כלפי מעלה וגורם לאפקט של קפיצה.

קיימת מחלקה בשם PipesView אשר שומרת את הנתונים על הצינורות. בפעולה הבונה של המחלקה מכניסים 1 או 2 המבדיל בין הצינור העליון לצינור התחתון. במחלקה זו שומרים את ערך ה-Y של הצינורות ואת גובהם. בכך ניתן לבנות "מסלול מכשולים" לציפור המורכב משני צינורות ורווח ביניהם. גובה הצינורות וערך ה-Y שלהם חייב להיות מותאם למיקום הרווח ביניהם וגודל המסך. לכן במחלקה זו מחשבים את מיקום הרווח ומיקום הצינורות וגובהם ביחס לרווח.

הרחבות

Thread

בפרויקט שלי יש Thread נפרד מה Thread Ui. הוא הthread שמריץ את הלולאה הראשית של המשחק ברקע, הוא נמצא ב-Game View ו-GameViewMultiplayer הלולאה מבצעת באופן קבוע עדכון של מצב העצמים כמו מיקום הצינורות והציפור, ערכי ה-Firebase ועוד.

השימוש ב־Thread נפרד מאפשר להריץ את המשחק בצורה חלקה, מבלי להעמיס על הלולאה הראשית של המערכת (ui thread).

שימוש בגרפיקה

Canvas

אובייקט המשמש לציור של תמונות, עצמים ואובייקטים על המסך – במיקום ובזמן שאני קובע. הוא מאפשר שליטה מלאה על איך כל אובייקט מוצג על מסך. השתמשתי בו במחלקות Game View ו-GameViewMultiplayer לציור של הציפורים, צינורות והרקע.

SurfaceView

אובייקט שמורחב מ־View, ומאפשר ציור ישיר על המסך בלולאה נפרדת מה־UI, בצורה יעילה וחלקה. הוא אידיאלי למשחקים וגרפיקה בזמן אמת.

השתמשתי בו במסכי המשחק במחלקות Game View ו- GameViewMultiplayer שיורשים ממנו.

מדריך משתמש

מטרת המערכת

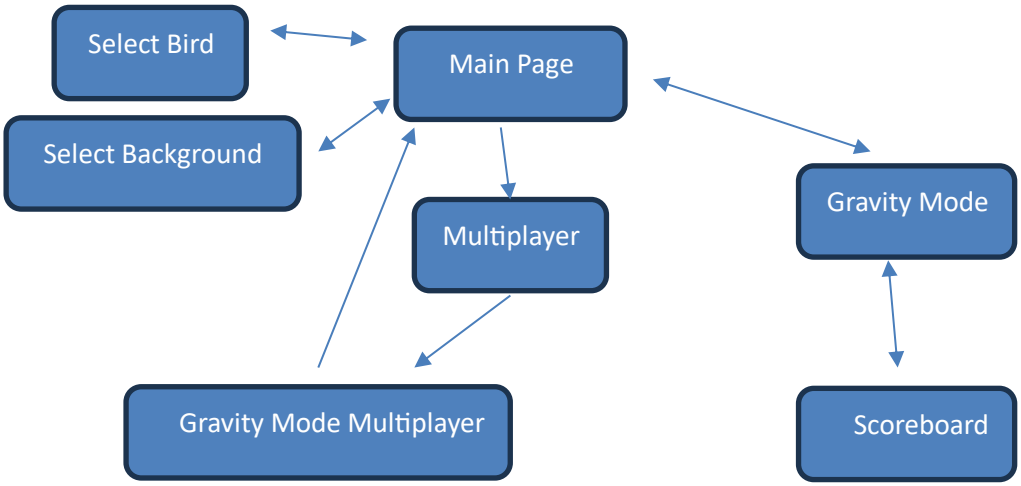
הפרויקט של הינו משחק פלאפי בירד רגיל. מטרת המשחק פלאפי בירד מאוד פשוטה: על המשתמש לשלוט בציפור כך שתעוף דרך פתחים בין צינורות מבלי להיתקע בהם או בקרקע. כל מעבר דרך הפתחים מזכה את השחקן בנקודה. המטרה היא להשיג כמה שיותר נקודות לפני שהציפור מתרסקת.

אופן השימוש

כאשר משתמש נכנס לתוך האפליקציה מוצגות בפניו כמה אפשרויות:

1. להתחיל משחק. המשתמש יכול לבחור בהתחלת משחק ובכך להתחיל לשחק במשחק לפי החוקים שלו. המשתמש צובר נקודות עד פסילה. כאשר נפסל מוצגות כמה אפשרויות: להתחיל את המשחק מחדש, לחזור למסך הבית או להסתכל על לוח התוצאות וההישגים.
2. להתחיל משחק multiplayer. ניתנת היכולת לשחק נגד שחקן נוסף. כאשר שחקן נכנס לאפשרות הזאת הוא בוחר האם להצטרף למשחק או ליצור משחק חדש. אם בוחר ליצור משחק חדש או מחכה עד שיצטרף שחקן נוסף, כאשר יש 2 שחקנים מתחיל המשחק ביניהם.
3. לשנות סוג ציפור. ניתן להחליף את תמונת הציפור במגוון של ציפורים שונות ובכך לאפשר לשחקן חופשיות ובחירה.
4. לשנות רקע. ניתן להחליף את הרקע של המשחק ברקעים שונים. בכך לאפשר למשתמש לעצב את המסך כפי שנראה לו יפה.
5. להיכנס כמשתמש. אין במשחק שמירת משתמשים אך שחקן יכול לכתוב את השם שלו ובכך בלוח התוצאות יהיה רשום שמו של השחקן. המטרה לגרום לשחקן תחושת הצלחה ביחס לאחרים המשחקים גם. אם השחקן לא מכניס שם המערכת מייצרת שם רנדומלי.

תרשים זרימה בין המסכים

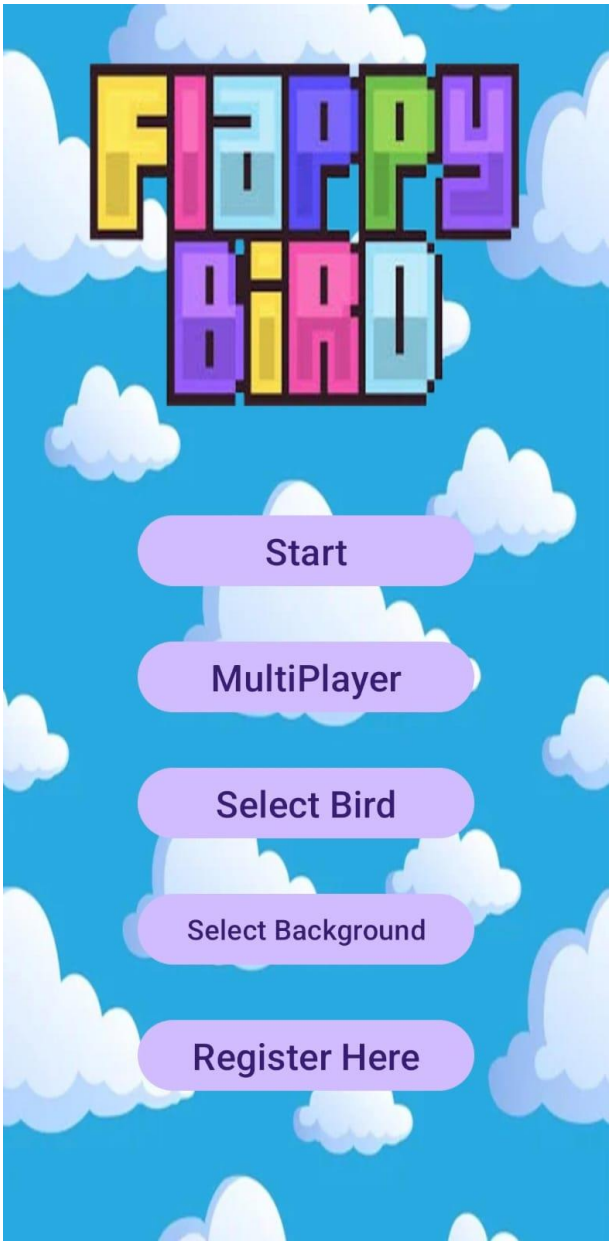


תפעול המערכת

תפריט ראשי

בתפריט מוצגים אפשרויות בחירה:

1. התחלת משחק רגיל
2. התחלת משחק מרובה משתתפים
3. בחירת סוג ציפור
4. בחירת רקע
5. הכנסת שם שחקן



מסכים

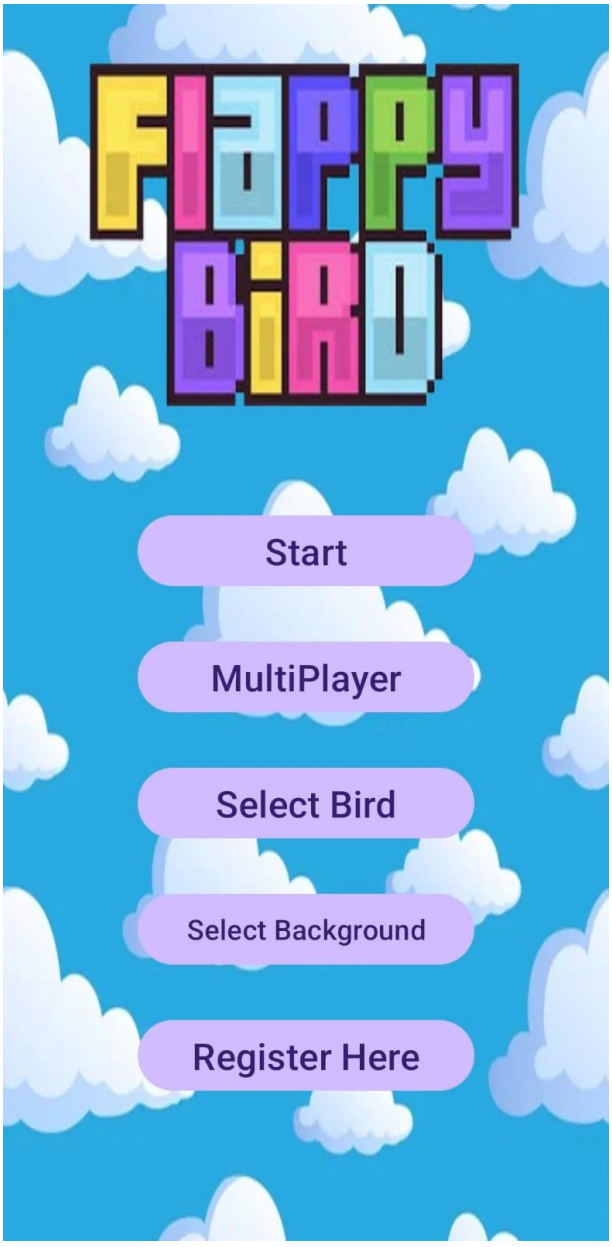
מסך ראשי – Main Activity

במסך זה מוצגות 5 אפשרויות:

1. התחלת משחק רגיל
2. התחלת משחק מרובה משתתפים
3. בחירת סוג ציפור
4. בחירת רקע
5. הכנסת שם שחקן

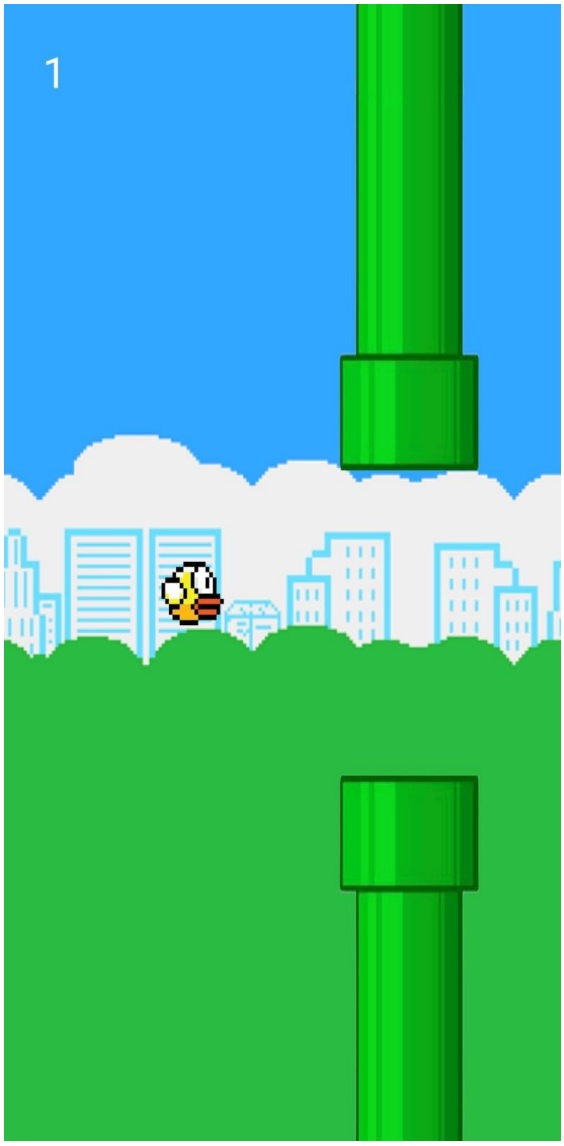
לאחר השלמת שינוי ציפור/רקע האפליקציה תחזור למסך הזה.

לאחר סיום משחק רגיל או מרובה משתתפים תינתן לשחקן אפשרות לחזור חזרה למסך זה ומשם לנווט את כל הפעילויות. מסך זה הוא התפריט הראשי של כל הפרויקט. ממנו מנווטים לכל האפשרויות והפונקציות בקיימות במשחק.



מסך המשחק הרגיל – Gravity Mode

במסך זה מתרחש המשחק הרגיל. במסך זה ניתן לראות את הציפור זהה, את העמודים זזים, את הרקע שנבחר למשחק ואת הנקודות שצבר השחקן. במסך זה ניתן לראות את מכניקת המשחק פועלת. הציפור נופלת בגרביטציה ואם השחקן לוחץ על המסך הציפור קופצת. אם הציפור נופלת לרצפה או פוגעת בצינורות אז המשחק נגמר. כשהמשחק נגמר מופיעה הודעה המאפשרת לשחקן להחליט מבין כמה אפשרויות: לחזור למסך הראשי, restart להתחיל את המשחק מחדש עם איפוס נקודות ואיפוס מהירות המשחק, ללכת למסך הבא של לוח התוצאות.

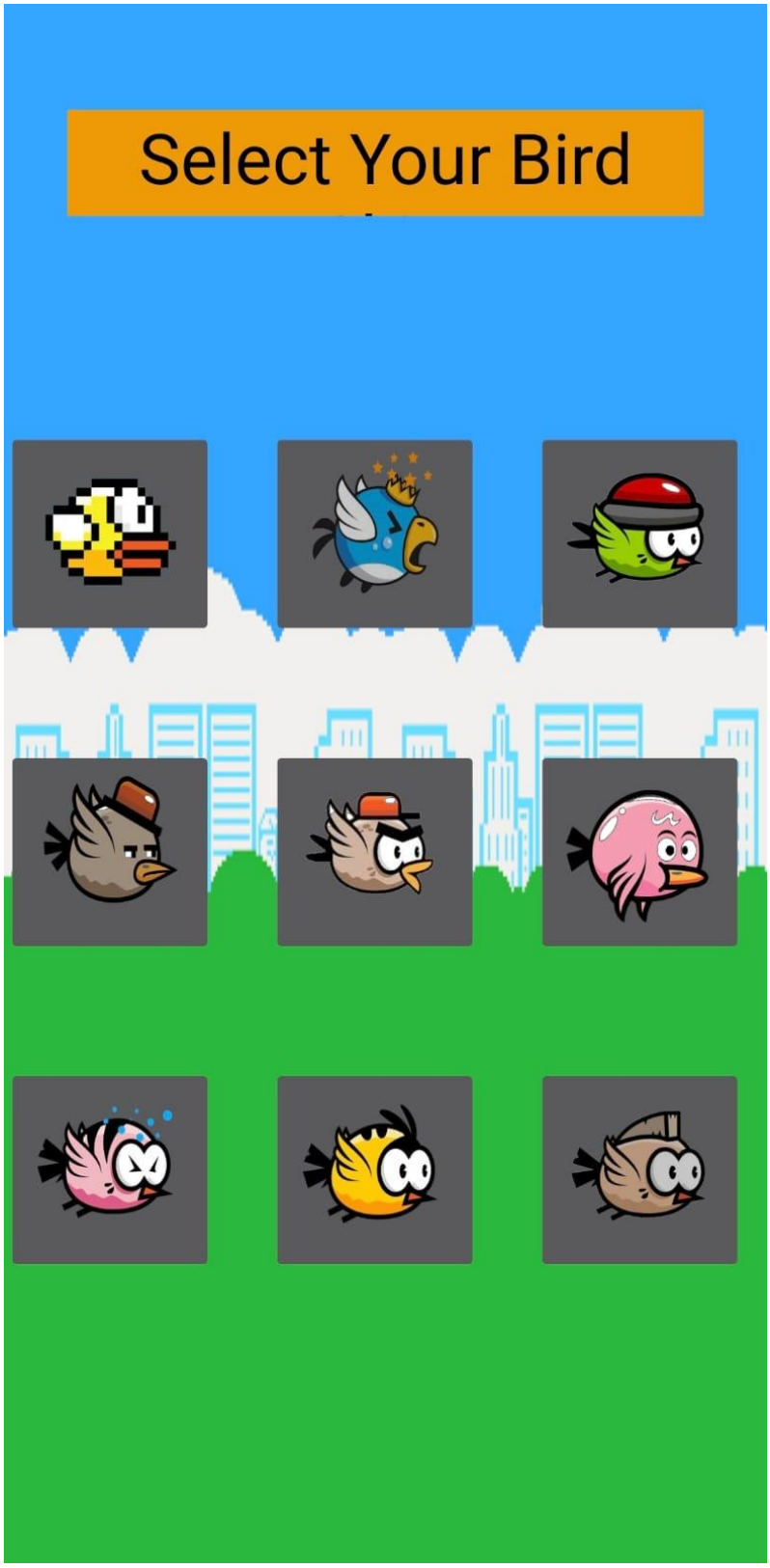


מסך לוח התוצאות – Score Board Activity

במסך זה ניתן לראות את כל התוצאות השמורות על ידי שחקנים. בכל פעם שנגמר משחק רגיל אז המשחק שומר את שם השחקן ואת הנקודות שהשיג ב-FireBase וכאשר נכנסים למסך הזה אז הנתונים של כל השחקנים ששיחקו במשחק מועברים לרשימה מה-FireBase שאני ממיין בסדר יורד לפי כמות הנקודות שעשה שחקן. בכך ניתן לראות את ההישגים של שאר השחקנים ולבדוק מי עשה הכי הרבה נקודות. קיים כפתור שמחזיר את המשתמש למסך הראשי.

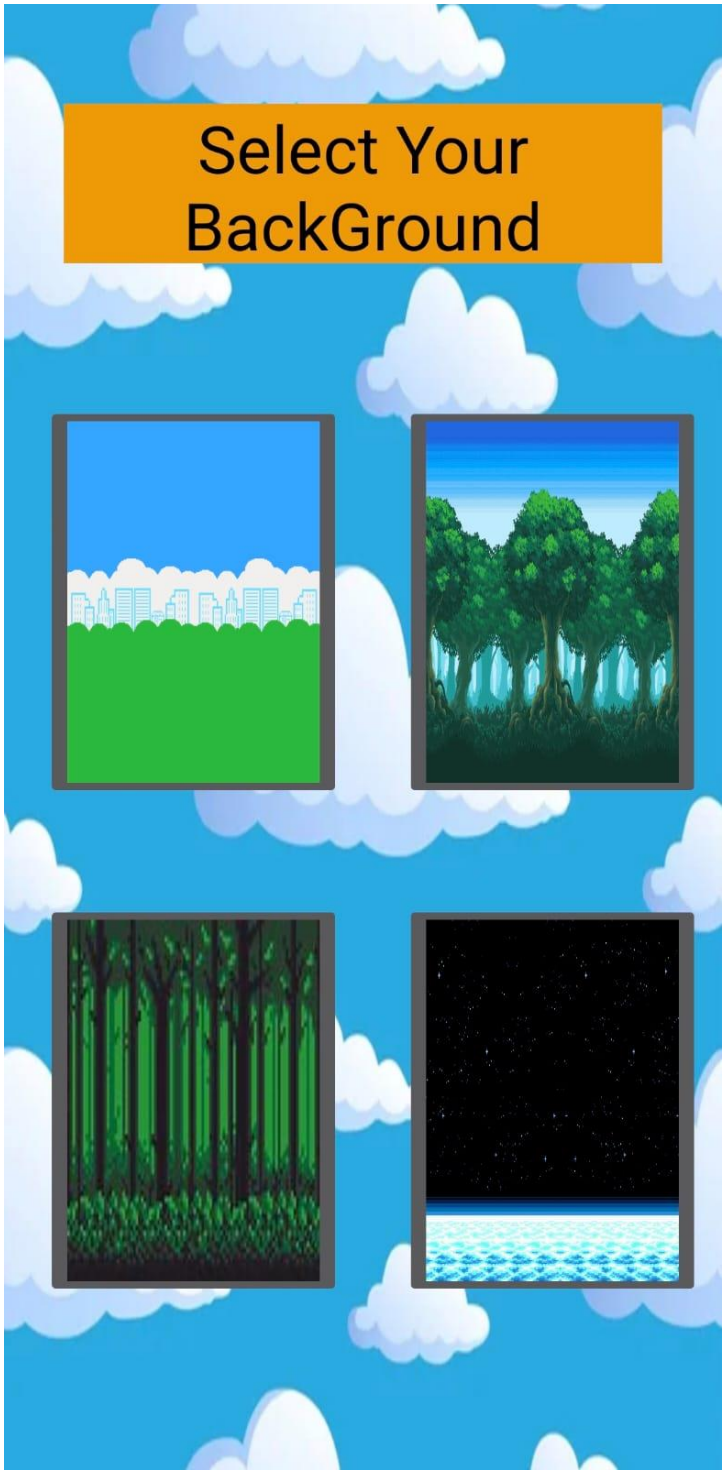
מסך בחירת הציפור – Bird Selection Activity

במסך זה ניתן לשנות את סוג הציפור שמשחקים איתה במשחק הרגיל. קיימת בחירה בין 9 ציפורים שונות ומגוונות. לכל ציפור צבע שונה, אישיות שונה, בגדים ועוד. לאחר שהשחקן לוחץ על הציפור שבחר הוא חוזר ישר למסך הראשי.



מסך בחירת הרקע – Background Selection Activity

במסך זה ניתן לשנות את הרקע של המשחק הרגיל, המרובה משתתפים וכל מסך אחר.
קיימת בחירה בין4 רקעים שונים. לכל רקע צבע שונה, אווירה שונה, מאפיינים שונים ועוד.
לאחר שהשחקן לוחץ על הרקע שבחר הוא חוזר ישר למסך הראשי.



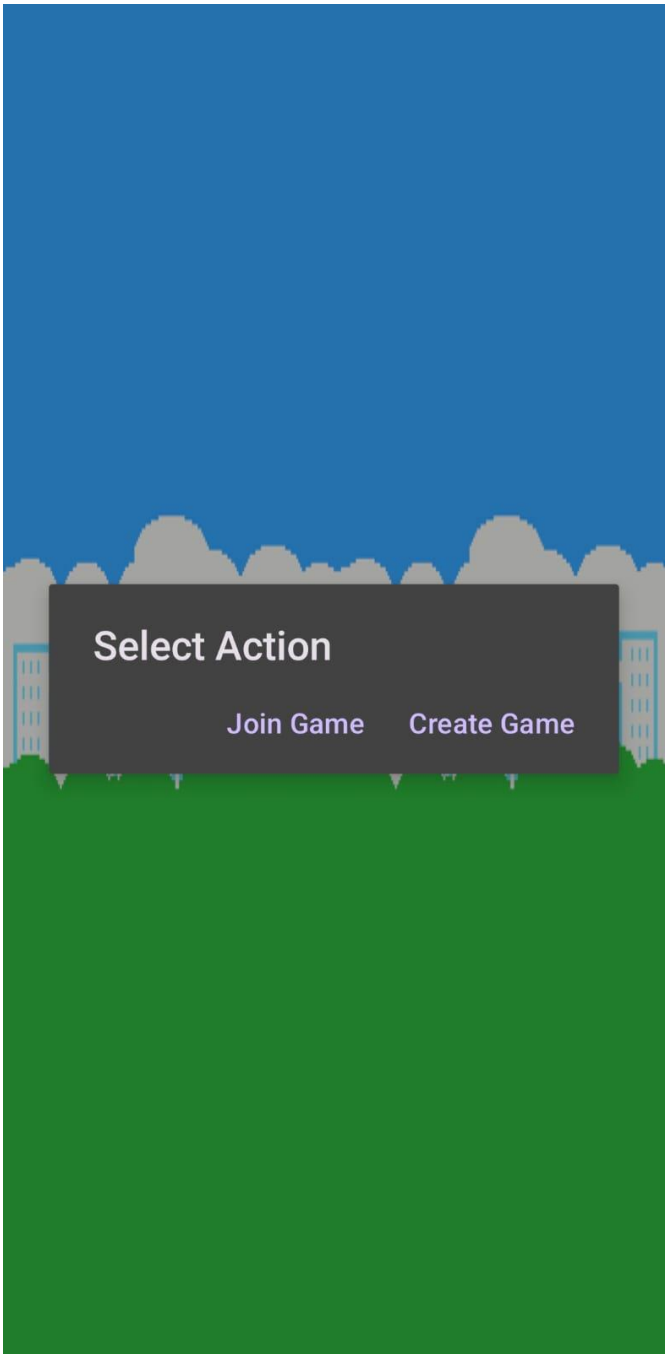
מסך בחירת אופציה למשחק המרובה משתתפים – Multiplayer Activity

במסך זה קיימת בחירה בין 2 אפשרויות.

אפשרות ראשונה היא ליצור משחק מרובה משתתפים חדש. אם אינו קיים כבר משחק, אז האפליקציה מייצרת משחק חדש והשחקן שבחר באופציה מחכה שיצטרף השחקן השני. אם קיים משחק אז אי אפשר ליצור עוד אחד.

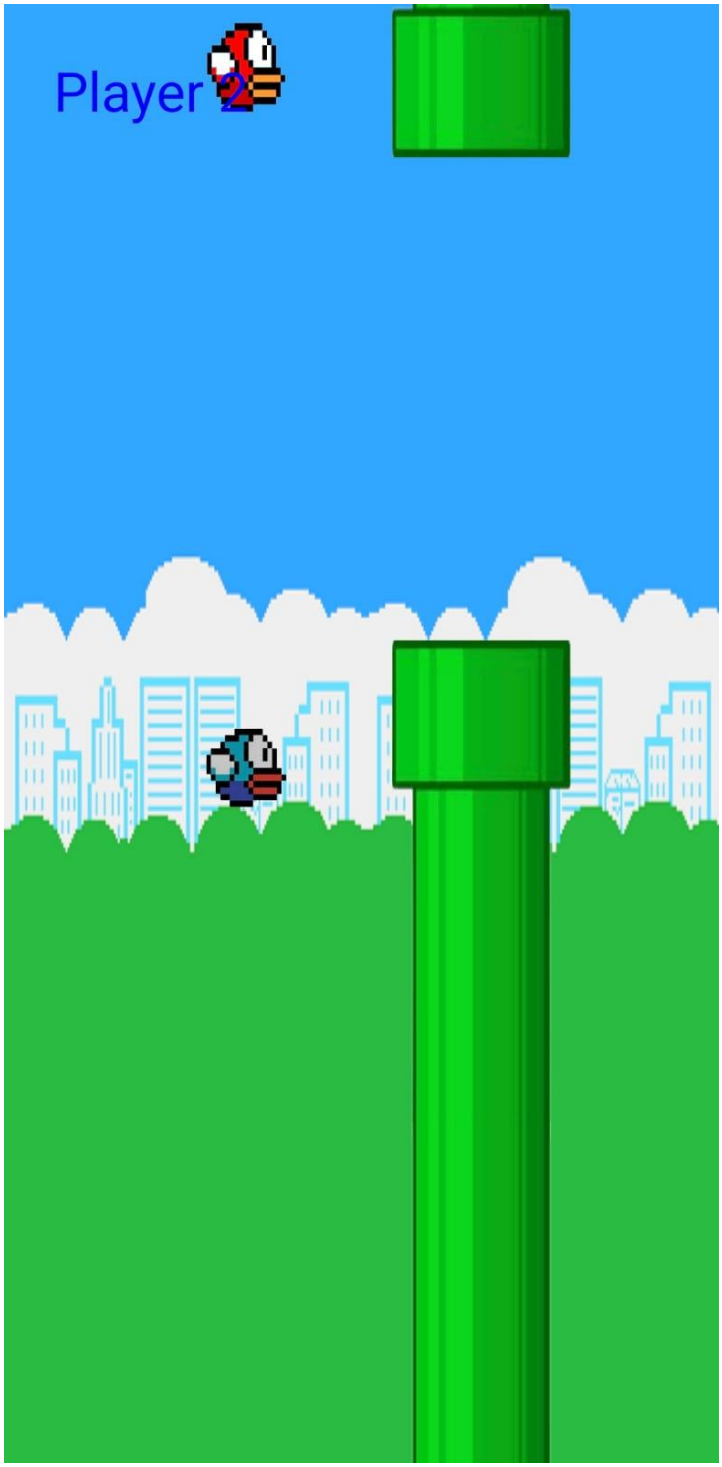
אפשרות שנייה היא להיכנס למשחק קיים. כאשר לוחצים על אופציה זו, האפליקציה בודקת אם נוצר משחק קיים. אם נוצר אז השחקן השני נכנס למשחק המולטי פלייר, אם לא קיים משחק אז האפליקציה מודיעה למשתמש שאין משחק קיים.

לאחר שנכנסו 2 שחקנים מתחיל המשחק ביניהם.



מסך המשחק המרובה משתתפים - Gravity Mode Multiplayer

במסך זה מופיע המשחק המרובה משתתפים עצמו. הציפור האדומה היא הציפור של השחקן שיצר את המשחק והציפור הכחולה היא של השחקן שהצטרף. ניתן לראות את תזוזות השחקן האחר בזמן המשחק. מיקום הפתחים בין הצינורות רנדומלי אך זהה בין 2 השחקנים בשביל שהמשחק יהיה הוגן. ברגע ששחקן אחד נפסל זה מפסיק את המשחק לשני השחקנים ומודיע לכל שחקן אם הפסיד או ניצח. המטרה לשרוד כמה שיותר זמן.



הרשאות

פירוט ההרשאות הנדרשות

android.permission.ACCESS_NETWORK_STATE- נדרשת כדי לגשת למצב האינטרנט.

דרישות מיוחדות ומגבלות

על מנת להשתמש באפליקציה יש צורך בחיבור לאינטרנט ועל מנת לשחק משחק בעל שני שחקנים בצורה חלקה יש צורך גם בחיבור אינטרנט חזק כי אחרת השחקן השני יראה "נתקע" במסך של השחקן האחר.

דרישות תוכנה/חומרה

גרסת Android מינימלית

Android 4.4.2 גרסת 19 API

מכשירים עליהם נבדקה המערכת

Samsung Galaxy A55

Google Pixel 6a

בסיס הנתונים

שימוש ב – FireBase

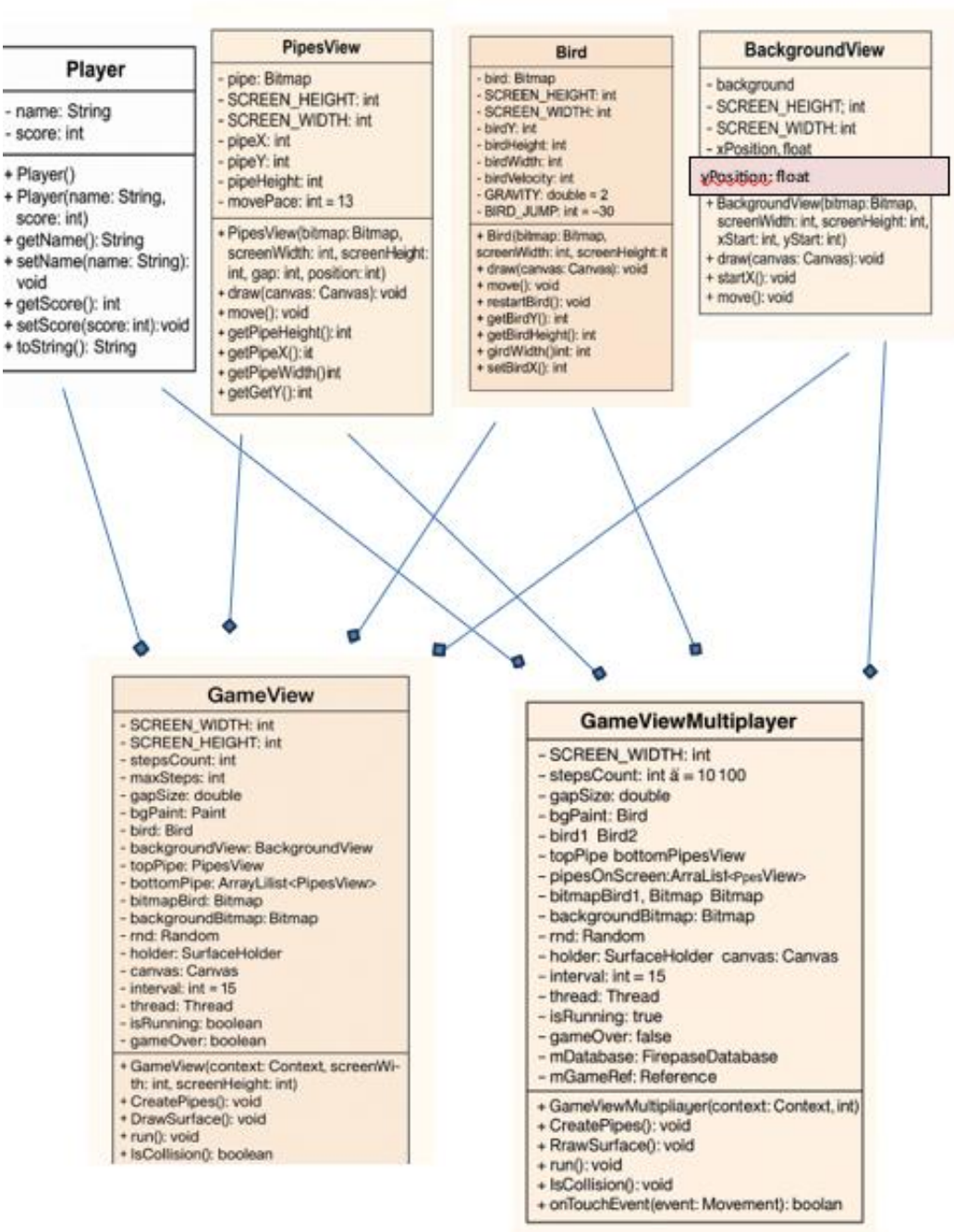
שם collection	שם משתנה ב collection	משמעות השדה	סוג ערך (double,int...)
Game	• isOver • Player2Y • randomGap • status	• האם נגמר המשחק • ערך הY של הציפור של שחקן 2 • המקום הרנדומלי של הפתח בצינורות • סטטוס המשחק (התחיל,מחכה)	• Boolean • int • int • String
Players	• Player1 • Player2 • Player3 וכן הלאה	מכיל את השם והניקוד שכל שחקן עשה	• String

https://flappybirdproject-4246c-default-rtdb.firebaseio.com

https://flappybirdproject-4246c-default-rtdb.firebaseio.com/
game
isOver: true
player1
Y: 2006
player2
Y: 484
randomGap: 534
status: "waiting"
players
Maayan Levi : 1
Oron : 11
Oron : 27

מדריך למפתח

תרשים מחלקות



תרשים activities

BackgroungSelectionActivity

+ RelativeLayout main
+ TextView title
+ ImageButton background1
+ ImageButton background2
+ ImageButton background3
+ ImageButton background4

+ onCreate(Bundle) : : void
+ ClickedBackground1(View) : : void
+ ClickedBackground2(View) : : void
+ ClickedBackground3(View) : : void
+ ClickedBackground4(View) : : void

BirdSelectionActivity

+ RelativeLayout main
+ TextView title
+ ImageButton bird1
+ ImageButton bird2
+ ImageButton bird3
+ ImageButton bird4
+ ImageButton bird5
+ ImageButton bird6
+ ImageButton bird7
+ ImageButton bird8
+ ImageButton bird9
- Drawable skybackground

+ onCreate(Bundle) : : void
+ ClickedBird1(View) : : void
+ ClickedBird2(View) : : void
+ ClickedBird3(View) : : void
+ ClickedBird4(View) : : void
+ ClickedBird5(View) : : void
+ ClickedBird6(View) : : void
+ ClickedBird7(View) : : void
+ ClickedBird8(View) : : void
+ ClickedBird9(View) : : void
+ ReturnHome(View) : : void

GravityMode

+ RelativeLayout main
+ FrameLayout frmLayout
+ ImageView GameOver
+ TextView score
- Drawable skybackground

+ onCreate(Bundle) : : void
+ updateScore(int) : : void
+ showGameOver() : : void

GravityModeMultiplayer

+ RelativeLayout main
+ FrameLayout frmLayout
+ ImageView GameOver
+ TextView playerInfo
+ Chronometer chronometer
- Drawable skybackground
- Drawable gameover

+ onCreate(Bundle) : : void
+ showGameOver() : : void
+ updatePlayerInfo(String) : : void
+ startChronometer() : : void
+ stopChronometer() : : void

MainActivity

- static String playerName
- RelativeLayout relativeLayout
- BitmapDrawable drawable

+ void onCreate(Bundle savedInstanceState)
+ void startButtonClicked(View view)
+ void multiplayerButtonClicked(View view)
+ void birdSelectionButtonClicked(View view)
+ void backgroungSelectionButtonClicked(View view)
+ void RegisterButtonClicked(View view)

MultiplayerActivity

- FirebaseDatabase mDatabase
- static boolean isPlayer2

+ void onCreate(Bundle savedInstanceState)
+ void StartGame()
+ void WaitToStart()
- void showCreateOrJoinDialog()
- void createNewGame()
- void joinGame()

ScoreBoardActivity

- ArrayList<Player> playersList
- ListView listView
- FirebaseDatabase mDatabase
- DatabaseReference mGameRef

+ void onCreate(Bundle savedInstanceState)
+ void CreateList()
+ void ReturnToGameButton(View view)

Activities – תיאור כל ה- activities

MainActivity

פעולות onClick שמעבירות אל האקטיבי בהתאם ללחיצה על הכפתור במסך. בנוסף פעולה שמקבלת את שם המשתמש ושומרת אותו במשתנה סטטי. תכונות עיקריות : playerName תכונה אשר מכילה את שם השחקן שמשחק. הפעולות במחלקה זו מנווטים בין המסכים בעזרת הכפתורים המוצגים במסך הראשי.

```
package com.example.flappybird;


import static java.security.AccessController.getContext;


import android.app.AlertDialog;
import android.content.Intent;
import android.graphics.drawable.BitmapDrawable;
import android.os.Bundle;
import android.text.InputType;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.RelativeLayout;


import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;


import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;


import java.util.Random;


public class MainActivity extends AppCompatActivity
}
    public static String playerName;" "=
```

```
@    Override
    protected void onCreate(Bundle savedInstanceState)
}
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets<- (
}
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
;({

        if(playerName.equals(" "))
}
            Random rnd=new Random;()
            playerName= "Player "+rnd.nextInt;(10000)
{

{
    public void startButtonClicked(View view)
}
        Intent myIntent=new Intent(MainActivity.this,GravityMode.class);
        startActivity(myIntent);
{
    public void multiplayerButtonClicked(View view)
}
        Intent myIntent=new Intent(MainActivity.this,MultiplayerActivity.class);
        startActivity(myIntent);
{
    public void birdSelectionButtonClicked(View view)
}
        Intent myIntent=new Intent(MainActivity.this,BirdSelectionActivity.class);
        startActivity(myIntent);
{
    public void backgroungSelectionButtonClicked(View view)
```

```
}
        Intent myIntent=new Intent(MainActivity.this,BackgroungSelectionActivity.class);
        startActivity(myIntent);
{
    public void RegisterButtonClicked(View view)     // an alert dialog that saves the players name
}
        final EditText editText = new EditText(this);
        AlertDialog.Builder dialogBuilder = new AlertDialog.Builder(this);
        dialogBuilder.setTitle("Enter your name")
.               setCancelable(false)
.               setView(editText)  // Add EditText to the dialog
.               setPositiveButton("Save", (dialog, which<- (
}
                  String name = editText.getText().toString;()
                  playerName=name.toString;()
({
.               setNegativeButton("Cancel", null)
.               create()
.               show;()
{
```

BirdSelectionActivity

פעולות onClick שבוחרות את סוג הציפור בהתאם ללחיצת המשתמש. משתנה עיקרי: selectedBird משתנה אשר שומר את תמונת הציפור שנבחרה על ידי השחקן. הפעולות במחלקה זו שומרות את תמונת הציפור שאותה בחר השחקן.

```
;package com.example.flappybird


;import android.content.res.Resources

;import android.graphics.Bitmap

;import android.graphics.BitmapFactory

;import android.graphics.drawable.BitmapDrawable

;import android.os.Bundle

;import android.view.View

;import android.widget.ImageView

;import android.widget.RelativeLayout


;import androidx.activity.EdgeToEdge

;import androidx.appcompat.app.AppCompatActivity

;import androidx.core.graphics.Insets

;import androidx.core.view.ViewCompat

;import androidx.core.view.WindowInsetsCompat


public class BirdSelectionActivity extends AppCompatActivity

}

;public static Bitmap selectedBird

public static boolean indicateBirdSelectionActivity=false;   // checks if the player entered the activity

Override@

protected void onCreate(Bundle savedInstanceState)

}

;super.onCreate(savedInstanceState)

;EdgeToEdge.enable(this)

;setContentView(R.layout.activity_bird_selection)

<- (ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets

}

;Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
```

```
;v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)

;return insets

;({



;RelativeLayout relativeLayout = findViewById(R.id.main)

BitmapDrawable drawable = new BitmapDrawable(getResources(),
;BackgroungSelectionActivity.selectedBackground)

if(BackgroungSelectionActivity.indicateBackgroundSelectionActivity)

;relativeLayout.setBackground(drawable)



;indicateBirdSelectionActivity=true

{

};()public void ReturnHome (View view) {finish



assign the birds images if chosen //

public void ClickedBird1 (View view){selectedBird = BitmapFactory.decodeResource(getResources(),
}  ;()R.drawable.bird);finish

public void ClickedBird2 (View view){selectedBird = BitmapFactory.decodeResource(getResources(),
};()R.drawable.cryingbird);finish

public void ClickedBird3 (View view){selectedBird = BitmapFactory.decodeResource(getResources(),
};()R.drawable.greenbirdwithhat);finish

public void ClickedBird4 (View view){selectedBird = BitmapFactory.decodeResource(getResources(),
};()R.drawable.nonchalant);finish

public void ClickedBird5 (View view){selectedBird = BitmapFactory.decodeResource(getResources(),
};()R.drawable.oldbirdwithhat);finish

public void ClickedBird6 (View view){selectedBird = BitmapFactory.decodeResource(getResources(),
};()R.drawable.pinkbird);finish

public void ClickedBird7 (View view){selectedBird = BitmapFactory.decodeResource(getResources(),
};()R.drawable.pinkbirdcrying);finish

public void ClickedBird8 (View view){selectedBird = BitmapFactory.decodeResource(getResources(),
};()R.drawable.yellowbird);finish

public void ClickedBird9 (View view){selectedBird = BitmapFactory.decodeResource(getResources(),
};()R.drawable.brownishbird);finish
```

BackgroungSelectionActivity

פעולות onClick שבוחרות את הרקע בהתאם ללחיצת המשתמש. משתנה עיקרי:
selectedBackground משתנה אשר שומר את תמונת הרקע שנבחרה על ידי השחקן.
הפעולות במחלקה זו שומרות את הרקע אשר נחבר על ידי השחקן.

```
package com.example.flappybird;


import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.graphics.drawable.BitmapDrawable;

import android.os.Bundle;

import android.view.View;

import android.widget.RelativeLayout;


import androidx.activity.EdgeToEdge;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.graphics.Insets;

import androidx.core.view.ViewCompat;

import androidx.core.view.WindowInsetsCompat;


public class BackgroungSelectionActivity extends AppCompatActivity

}


    public static Bitmap selectedBackground;

    public static boolean indicateBackgroundSelectionActivity=false;   // checks if the player entered the activity


@    Override

    protected void onCreate(Bundle savedInstanceState)}

        super.onCreate(savedInstanceState);

        EdgeToEdge.enable(this);

        setContentView(R.layout.activity_backgroung_selection);

        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets<- (

}
```

```
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());

            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);

            return insets;

;({



        indicateBackgroundSelectionActivity=true;

{

    public void ClickedBackground1 (View view) {selectedBackground = BitmapFactory.decodeResource(getResources(),
R.drawable.skybackground);finish;()}

    public void ClickedBackground2 (View view) {selectedBackground = BitmapFactory.decodeResource(getResources(),
R.drawable.forestbackground);finish;()}

    public void ClickedBackground3 (View view) {selectedBackground = BitmapFactory.decodeResource(getResources(),
R.drawable.darkforestbackground);finish;()}

    public void ClickedBackground4 (View view) {selectedBackground = BitmapFactory.decodeResource(getResources(),
R.drawable.spacebackground);finish;()}

{
```

GravityMode

האקטיביטי שמייצרת את משתנה GameView אשר מייצר את כל המשחק הרגיל. בתוכה ישנם המשתנים האחראיים על ניהול הנקודות ופרטי השחקן. המשתנים העיקריים מכילים את התמונות אשר מופיעות מאחורי הרקע של המשחק (gameover,score). במחלקה זו נוצר משתנה gameView אשר אחראי לניהול כל המשחק הרגיל.

```
package com.example.flappybird;


import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.graphics.drawable.BitmapDrawable;

import android.os.Bundle;

import android.widget.FrameLayout;

import android.widget.ImageView;

import android.widget.RelativeLayout;

import android.widget.TextView;


import androidx.appcompat.app.AppCompatActivity;


import java.util.ArrayList;


public class GravityMode extends AppCompatActivity

}


    private FrameLayout frm;

    private Bitmap bitmap;

    private GameView GameView;

    public static int scoreCount=0;

    public static TextView score;

    public static ImageView gameOver;

    public static Player player;


    public static ArrayList<String> listNames;

    public static ArrayList<Integer> listScores;


@    Override

    protected void onCreate(Bundle savedInstanceState)
```

```
}

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_gravity_mode);

        RelativeLayout relativeLayout = findViewById(R.id.main);
        BitmapDrawable drawable = new BitmapDrawable(getResources(), BackgroungSelectionActivity.selectedBackground) ;
        if(BackgroungSelectionActivity.indicateBackgroundSelectionActivity)
            relativeLayout.setBackground(drawable);

        listNames=new ArrayList<String;()<
        listScores=new ArrayList<Integer;()<
        frm = findViewById(R.id.frmLayout);
        bitmap = BitmapFactory.decodeResource(getResources(), R.drawable.bird);
        bitmap = Bitmap.createScaledBitmap(bitmap, 30, 40, false);
        gameOver = findViewById(R.id.GameOver);

{

@    Override
    public void onWindowFocusChanged(boolean hasFocus)
}
        super.onWindowFocusChanged(hasFocus);
        if (hasFocus )
}
            int w = frm.getWidth;()
            int h = frm.getHeight;()
            score=findViewById(R.id.score);  // text view of the score displayed
            score.setText(Integer.toString(scoreCount));
            GameView = new GameView(this,w,h);
            frm.addView(GameView);
{
{

 {
```

ScoreBoardActivity

אחראית על יצירת לוח הנקודות וההישגים בסוף המשחק. ממיינת רשימה של כל השחקנים והנקודות שלהם ובונה לוח מנצחים. מציגה את התוצאות בעזרת רשימת adapter. משתנה עיקרי הוא (winnerList) הרשימה אשר שומרת את כל השחקנים ואז ממיינת אותם לפי נקודות. הפעולה הראשית במחלקה זו (()CreateList) יוצרת את הרשימה של השחקנים והניקוד וממיינת אותה לפי השחקנים עם הניקוד הגבוה ביותר.

```
package com.example.flappybird;


import android.content.Intent;

import android.graphics.drawable.BitmapDrawable;

import android.os.Bundle;

import android.view.View;

import android.widget.Adapter;

import android.widget.ArrayAdapter;

import android.widget.ListAdapter;

import android.widget.ListView;

import android.widget.RelativeLayout;


import androidx.activity.EdgeToEdge;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.graphics.Insets;

import androidx.core.view.ViewCompat;

import androidx.core.view.WindowInsetsCompat;


import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;


import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;


public class ScoreBoardActivity extends AppCompatActivity
```

```
}
    private ArrayList<Player> playersList = new ArrayList;()<>
    private ListView listView;
    private FirebaseDatabase mDatabase;
    private DatabaseReference mGameRef;

@   Override
    protected void onCreate(Bundle savedInstanceState)
}
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_score_board);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets<- (
}
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
;({

 //       change the background
        RelativeLayout relativeLayout = findViewById(R.id.main);
        BitmapDrawable drawable = new BitmapDrawable(getResources(), BackgroungSelectionActivity.selectedBackground);
        if(BackgroungSelectionActivity.indicateBackgroundSelectionActivity)
            relativeLayout.setBackground(drawable);

        mDatabase=FirebaseDatabase.getInstance;()
        mGameRef=mDatabase.getReference("players");

 //       add all players in firebase to the list
        mGameRef.addListenerForSingleValueEvent(new ValueEventListener()
}
@           Override
            public void onDataChange(@NonNull DataSnapshot snapshot)
}
                for (DataSnapshot snap : snapshot.getChildren())
}
```

```
                Player player=new Player(snap.getValue(Player.class).getName(),snap.getValue(Player.class).getScore());
                playersList.add(player);
{
            CreateList;()
{

@           Override
        public void onCancelled(@NonNull DatabaseError error){}
;({

{

    public void CreateList()
}
 //       sort the list by scores
        int maxScore=-1;
        int maxIndex=0;
        ArrayList<Player> winnerList=new ArrayList;()<>
        while(playersList.size()>0)
}
          for (int i=0;i<playersList.size();i++)
}
            if(playersList.get(i).getScore()>maxScore)
}
                maxIndex=i;
                maxScore=playersList.get(i).getScore;()
{
{
          winnerList.add(playersList.get(maxIndex));
          maxScore=-1;
          playersList.remove(maxIndex);
{

 //       create the list
        ArrayAdapter <Player> adapter;
        adapter = new ArrayAdapter<Player> (this,android.R.layout.simple_list_item_1, winnerList);
```

```
        listView = (ListView) findViewById(R.id.listViewScores);

        listView.setAdapter(adapter);

        listView.setAdapter(adapter);

{

    public void ReturnToGameButton(View view)

}

        Intent myIntent=new Intent(ScoreBoardActivity.this,MainActivity.class);

        startActivity(myIntent);

{
```

\

MultiplayerActivity

מייצרת את משחק המרובה משתתפים. בודקת אם שחקן מסויים התחיל משחק או שהוא מחכה. ממיינת שחקנים לפי שחקן 1 ושחקן 2. משתנה עיקרי הוא (isPlayer2) משתנה בוליאני אשר שומר בתוכו איזה מן השחקנים הוא שחקן מספר 2. תפקיד מחלקה זו היא לבדוק אם קיים משחק מרובה משתתפים ולהתחיל אותו. הפעולה הכי חשובה במחלקה היא (waitToStart()) הפעולה אשר מחכה ששחקן 2 יצטרף וכאשר הוא מצטרף המשחק מתחיל.

```
package com.example.flappybird;


import static android.view.View.INVISIBLE;

import static android.view.View.VISIBLE;


import android.content.DialogInterface;

import android.content.Intent;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.graphics.drawable.BitmapDrawable;

import android.os.Bundle;

import android.util.Log;

import android.view.View;

import android.widget.EditText;

import android.widget.FrameLayout;

import android.widget.ImageView;

import android.widget.RelativeLayout;

import android.widget.TextView;

import android.widget.Toast;


import androidx.activity.EdgeToEdge;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AlertDialog;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.graphics.Insets;

import androidx.core.view.ViewCompat;

import androidx.core.view.WindowInsetsCompat;


import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;
```

```
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;
import java.util.UUID;

public class MultiplayerActivity extends AppCompatActivity
{
    private FirebaseDatabase mDatabase;
    public static boolean isPlayer2;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_multiplayer);

        RelativeLayout relativeLayout = findViewById(R.id.main);
        BitmapDrawable drawable = new BitmapDrawable(getResources(), BackgroungSelectionActivity.selectedBackground);
        if (BackgroungSelectionActivity.indicateBackgroundSelectionActivity)
            relativeLayout.setBackground(drawable);

        mDatabase = FirebaseDatabase.getInstance();

        // show the initial dialog
        showCreateOrJoinDialog();
    }
    public void StartGame()
    {
        ImageView waiting=findViewById(R.id.Waiting);
        waiting.setVisibility(INVISIBLE);
        DatabaseReference gameRef = mDatabase.getReference("game");
        gameRef.child("game").removeValue();   // refresh the firebase
        Intent myIntent=new Intent(MultiplayerActivity.this,GravityModeMultiplayer.class);
```

```
    startActivity(myIntent);
  }
  public void WaitToStart()
  {
    // wait till status changes to "playing" to indicate player 2 joined
    DatabaseReference gameRef = mDatabase.getReference("game");
    gameRef.child("status").addValueEventListener(new ValueEventListener()
    {
      @Override
      public void onDataChange(@NonNull DataSnapshot dataSnapshot)
      {
        if(dataSnapshot.getValue(String.class)!=null && dataSnapshot.getValue(String.class).equals("playing"))
          StartGame();
      }
      @Override
      public void onCancelled(@NonNull DatabaseError error) {}
    });
  }
  private void showCreateOrJoinDialog()
  {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Select Action")
        .setPositiveButton("Create Game", new DialogInterface.OnClickListener()
        {
          @Override
          public void onClick(DialogInterface dialog, int which) {
            createNewGame();
          }
        })
        .setNegativeButton("Join Game", new DialogInterface.OnClickListener()
        {
          @Override
          public void onClick(DialogInterface dialog, int which) {
            joinGame();
          }
```

```
            })
            .setCancelable(false)
            .show();
    }

    private void createNewGame()
    {
        // create a new game
        DatabaseReference gameRef = mDatabase.getReference("game");

        // check if no game started else start new one
        gameRef.child("status").addListenerForSingleValueEvent(new ValueEventListener()
        {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot)
            {
                String status = dataSnapshot.getValue(String.class);
                if(status!=null)
                {
                    if (status.equals("waiting") || status.equals("playing"))
                    {
                        Toast.makeText(MultiplayerActivity.this, "Game already started", Toast.LENGTH_SHORT).show();
                        showCreateOrJoinDialog();
                    }
                }
                else
                {
                    // once game created wait for other player to join
                    Toast.makeText(MultiplayerActivity.this, "Game created successfully", Toast.LENGTH_SHORT).show();
                    isPlayer2=false;
                    gameRef.child("status").setValue("waiting");
                    ImageView waiting=findViewById(R.id.Waiting);
                    waiting.setVisibility(VISIBLE);
                    WaitToStart();
                }
```

```
        }
        @Override
        public void onCancelled(DatabaseError databaseError) {}
    });

}

private void joinGame()
{
    DatabaseReference gameRef = mDatabase.getReference("game");

    // check if the game exists and if it is waiting for player 2
    gameRef.child("status").addListenerForSingleValueEvent(new ValueEventListener()
    {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot)
        {
            String status = dataSnapshot.getValue(String.class);
            if (status.equals("waiting"))
            {
                gameRef.child("status").setValue("playing");
                isPlayer2=true;
                WaitToStart();
            }
            else
            {
                Toast.makeText(MultiplayerActivity.this, "No Game Exists", Toast.LENGTH_SHORT).show();
                showCreateOrJoinDialog();
            }
        }

        @Override
        public void onCancelled(DatabaseError databaseError) {}
    });
}
```

```
}
```

GravityModeMultiplayer

האקטיביטי שמייצרת את משתנה GameViewMultiplayer אשר מייצר את כל המשחק המרובה משתתפים. בתוכה ישנם המשתנים האחראיים על ניהול כל שחקן בנוסף לטיימר המסנכרן בין 2 השחקנים. משתנה עיקרי הוא הכרונומטר אשר אחראי לסנכרון בין 2 המשחקים המתקיימים במקביל. הפעולה העיקרית הינה זו אשר מייצרת משתנה gameViewMultiplayer אשר אחראי לניהול כל המשחק מרובה המשתתפים.

```
package com.example.flappybird;


import static android.view.View.VISIBLE;


import androidx.appcompat.app.AppCompatActivity;


import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.drawable.BitmapDrawable;
import android.os.Bundle;
import android.widget.Chronometer;
import android.widget.FrameLayout;
import android.widget.ImageView;
import android.widget.RelativeLayout;
import android.widget.TextView;


public class GravityModeMultiplayer extends AppCompatActivity
}
    private FrameLayout frm;
    private Bitmap bitmap1,bitmap2;
    private GameViewMultiplayer GameViewMultiplayer;
    public static ImageView gameOver;
    private TextView playerInfo;


    public static Chronometer chronometer;


@   Override
    protected void onCreate(Bundle savedInstanceState)
}
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_gravity_mode_multiplayer);

        RelativeLayout relativeLayout = findViewById(R.id.main);
        BitmapDrawable drawable = new BitmapDrawable(getResources(), BackgroungSelectionActivity.selectedBackground);
        if(BackgroungSelectionActivity.indicateBackgroundSelectionActivity)
            relativeLayout.setBackground(drawable);

        frm = findViewById(R.id.frmLayout);
        bitmap1 = BitmapFactory.decodeResource(getResources(), R.drawable.birdplayerone);
        bitmap1 = Bitmap.createScaledBitmap(bitmap1, 30, 40, false);
        bitmap2 = BitmapFactory.decodeResource(getResources(), R.drawable.birdplayertwo);
        bitmap2 = Bitmap.createScaledBitmap(bitmap2, 30, 40, false);
        gameOver = findViewById(R.id.GameOver);

 //       change the sign to show player 2
        playerInfo=findViewById(R.id.playerInfo);
        if(MultiplayerActivity.isPlayer2)
}
            playerInfo.setText("Player 2");
            playerInfo.setTextColor(0xFF0000FF);
{
        playerInfo.setVisibility(VISIBLE);

        chronometer = findViewById(R.id.chronometer);

{
@    Override
    public void onWindowFocusChanged(boolean hasFocus)
}
        super.onWindowFocusChanged(hasFocus);
        if (hasFocus)
}
            int w = frm.getWidth;()
            int h = frm.getHeight;()
```

```
        GameViewMultiplayer = new GameViewMultiplayer(this,h,w);
        frm.addView(GameViewMultiplayer);
{
{
{
```

Adapters

תפקידו להתאים מידע מורכב ומאורגן בין מבני הנתונים השונים לתצוגה כמו רשימה של פריטים או grid.

השתמשתי בו לעדכון מסך ה ScoreBoard להצגת רשימת פריטים.

Threads

תהליך הרץ ברקע שאחראי על הלולאה המרכזית של המשחק. אני משתמש בו במחלקת GameView GameViewMultiplayer כדי שיקרא לפעולה ()run שתרוץ במקביל שלאר הקוד בלי להפריע לו.

תיאור כל המחלקות

Player

במחלקה זו אני שומר מידע על המשתמש. יש 2 תכונות: שם המשתמש והנקודות שהרוויח. מחלקה זו עוזרת בהכנת לוח התוצאות בסוף המשחק.

```
package com.example.flappybird;


public class Player

}

    private String name;

    private int score;  // Firebase stores numbers in Long format


    public Player{} ()

    public Player(String name, int score){this.name = name;this.score = score;}

    public String getName(){return name;}

    public void setName(String name) {this.name = name;}

    public int getScore() {return score;}

    public void setScore(int score) {this.score = score;}

    public String toString() {return name + " : " + score; }

{
```

BackgroundView

במחלקה זו אני מנהל את הרקעים. את תזוזת הרקע ומיקומו על המסך. הפעולה העיקרית היא (Move()) הזאת האחראית על תזוזת הרקע.

```
package com.example.flappybird;

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

public class BackgroundView
{
   private Bitmap background;
   private int SCREEN_HEIGHT, SCREEN_WIDTH;
   private float xPosition,yPosition;
   public BackgroundView(Bitmap bitmap, int screenWidth, int screenHeight, int xStart, int yStart)
   {
      background=bitmap;
      SCREEN_HEIGHT=screenHeight;
      SCREEN_WIDTH=screenWidth;
      xPosition=(float)xStart;
      yPosition=(float)yStart;
      startX();
   }
   public void draw(Canvas canvas)
   {
      canvas.drawBitmap(background,xPosition,yPosition,null);
   }
   public void startX()
   {
     xPosition=0;
   }
```

```
public void move()
{
    xPosition-=10.48;   // adjust to phone width
    if(xPosition<=(-1*SCREEN_WIDTH))
        startX();
}
}
```

PipesView

מחלקה המנהלת את הצינורות. מנהלת את תזוזת הצינורות ומבדילה בין צינור עליון לתחתון. מזיזה את הצינורות ומחשבת את מיקומם על המסך. תכונה חשובה במחלקה זו היא (pipeX) התכונה השומרת את מיקומו של הצינור על המסך.

```
package com.example.flappybird;

import android.graphics.Bitmap;

import android.graphics.Canvas;

import android.graphics.Rect;


import java.util.Random;


public class PipesView

{

    private Bitmap pipe;

    private int SCREEN_HEIGHT, SCREEN_WIDTH;

    private int pipeX,pipeY,pipeHeight,pipeWidth;

    private int movePace=13; // moves pipes to the left  (13)


    public PipesView(Bitmap bitmap, int screenWidth, int screenHeight,int gap,int position)

    {


      this.SCREEN_WIDTH = screenWidth;

      this.SCREEN_HEIGHT = screenHeight;

      pipeX=SCREEN_WIDTH;


     if(position==1)  // 1 indicates top pipe

     {

        pipeY= -bitmap.getHeight()+gap;

        pipeHeight = bitmap.getHeight();

     }

     if(position==2)   // 2 indicates bottom pipe

     {
```

```
            pipeY=gap;
            pipeHeight = bitmap.getHeight();
        }


        pipeWidth = 270;     //setting width (270)
        pipe = bitmap;
        pipe = Bitmap.createScaledBitmap(bitmap,pipeWidth,pipeHeight,false);   //creating bitmap for the pipe
    }
    public void draw(Canvas canvas)
    {
        canvas.drawBitmap(pipe,pipeX,pipeY,null);
    }
    public void move() {pipeX-=movePace;}   //moves the pipe to the left and update the rect
  public int getPipeHeight() {return this.pipeHeight;}
  public int getPipeX() {return this.pipeX;}
  public int getPipeWidth() {return this.pipeWidth;}
  public int getPipeY() {return this.pipeY;}
}
```

Bird

מחלקה המנהלת את הציפורים. מנהלת את תזוזת הציפורים לפי קפיצות ונפילה לפי חוקי הגרביטציה. שומרת את מיקומה על שני הצירים ומהירותה בכל רגע. תכונה חשובה במחלקה זו היא (birdY) התכונה השומרת את מיקומה של הציפור על המסך.

```
package com.example.flappybird;


import android.graphics.Bitmap;

import android.graphics.Canvas;

import android.graphics.Matrix;

import android.graphics.Rect;

import android.widget.ImageView;

import android.widget.RelativeLayout;


public class Bird

{

   private Bitmap bird;

   private int SCREEN_HEIGHT, SCREEN_WIDTH;

   private int birdY,birdX;

   private int birdHeight,birdWidth;

   private int birdVelocity;

   private final double GRAVITY = 2;  // Gravity effect on the bird  (2)

   private final int BIRD_JUMP = -30;  // Jump effect on the bird  (-30)


   public Bird(Bitmap bitmap, int screenWidth, int screenHeight)

   {

      bird = bitmap;

      birdHeight = 200;  // (200)

      birdWidth = 150;   // (150)

      bird = Bitmap.createScaledBitmap(bitmap,birdWidth,birdHeight,false);

      this.SCREEN_WIDTH = screenWidth;

      this.SCREEN_HEIGHT = screenHeight;

      birdX=SCREEN_WIDTH/2-250;   // (-250)

      birdY=0;

   }
```

```
public void draw(Canvas canvas)
{
    canvas.drawBitmap(bird,birdX,birdY,null);
}
public void move()
{
    birdVelocity += GRAVITY;
    birdY += birdVelocity;

    // Prevent the bird from falling below the ground
    if (birdY > SCREEN_HEIGHT-100 )
        birdY=SCREEN_HEIGHT - 100;
    else if (birdY < 0)
    {
        birdY = 0;
        birdVelocity = 0;
    }
}
public void jump()
{
    birdVelocity = BIRD_JUMP;
    birdY += birdVelocity;

    // Prevent the bird from falling below the ground
    if (birdY > SCREEN_HEIGHT - 100)
    {
        birdY = SCREEN_HEIGHT-100;
        birdVelocity = 0;
    }
    else if (birdY < 0)
    {
        birdY = 0;
        birdVelocity = 0;
    }
}
```

```
    public void restartBird()
    {
        birdY=0;
        birdVelocity=0;
        birdX=SCREEN_WIDTH/2-250;
    }

    public int getBirdY() {return this.birdY;}
    public int getBirdHeight(){return this.birdHeight;}
    public int getBirdWidth(){return this.birdWidth;}
    public int getBirdX(){return this.birdX;}
    public void setBirdY(int birdY){this.birdY = birdY;}
}
```

GameView

המחלקה הכי חשובה בכל המשחק. מנהלת את כל המשחק הרגיל, החל מהפעולה הראשית אשר מציירת את הציפור, הצינורות והרקע, בודקת אם המשחק נגמר לפי התנגשויות ומנהלת את סיום המשחק. במחלקה זו מתנהל כל המשחק וכל החלק של הבדיקות. מחלקה זו מציירת את המסך וגורמת לאנימציה. בעצם מחלקה זו היא הלב של המשחק. פעולה מרכזית במחלקה היא הפעולה. run()

פעולה זו אחראית לכל המתרחש במחלקה : לצייר את הרקע, הציפור הצינורות, לבדוק אם נגמר המשחק, לנהל את הנקודות ועוד.

```
package com.example.flappybird;


import static androidx.core.content.ContextCompat.startActivity;


import android.app.Activity;

import android.app.AlertDialog;

import android.content.Context;

import android.content.DialogInterface;

import android.content.Intent;

import android.content.res.Resources;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Paint;

import android.graphics.Rect;

import android.media.MediaPlayer;

import android.os.Handler;

import android.os.Looper;

import android.text.InputType;

import android.view.MotionEvent;

import android.view.SurfaceHolder;

import android.view.SurfaceView;

import android.view.View;

import android.widget.EditText;

import android.widget.ImageView;

import android.widget.TextView;
```

```
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;
import java.util.Collection;
import java.util.Iterator;
import java.util.List;
import java.util.ListIterator;
import java.util.Random;

public class GameView extends SurfaceView implements Runnable
{
    private int SCREEN_WIDTH,SCREEN_HEIGHT;
    private int stepsCount=0,maxSteps=100;    //indicates when to make new tubes (100)
    private double gapSize;
    private Paint bgPaint;
    private Bird bird;
    private BackgroundView backgroundView;
    private PipesView topPipe,bottomPipe;
    private ArrayList<PipesView> pipesOnScreen;
    private Bitmap bitmapBird , backgroundBitmap;  // Bitmap for the bird and the background;
    private Random rnd;
    private SurfaceHolder holder;
    private Canvas canvas;
    private int interval = 15; // (15)
    private Thread thread;
    private boolean isRunning=true,gameOver=false;
```

```
private FirebaseDatabase mDatabase;

private DatabaseReference mGameRef;

private Player currentPlayer;

public GameView(Context context, int screenWidth, int screenHeight)

{

    super(context);


    SCREEN_WIDTH = screenWidth;

    SCREEN_HEIGHT = screenHeight;

    bgPaint = new Paint();

    bgPaint.setColor(Color.WHITE);


    pipesOnScreen=new ArrayList<PipesView>();  // list of all the pipes on the screen


    // checks to see if player changed background, if not it uses the deafult

    if(BackgroungSelectionActivity.indicateBackgroundSelectionActivity)

        backgroundBitmap = BackgroungSelectionActivity.selectedBackground;

    else

        backgroundBitmap = BitmapFactory.decodeResource(getResources(),R.drawable.skybackground);      // load the selected
bird image

    backgroundBitmap = Bitmap.createScaledBitmap(backgroundBitmap,2*SCREEN_WIDTH, SCREEN_HEIGHT, true);  // strech
the image

    backgroundView=new BackgroundView(backgroundBitmap, SCREEN_WIDTH, SCREEN_HEIGHT,0,0);


    // checks to see if player changed bird skin, if not it uses the deafult skin

    if(BirdSelectionActivity.indicateBirdSelectionActivity)

       bitmapBird=BirdSelectionActivity.selectedBird;

    else

        bitmapBird = BitmapFactory.decodeResource(getResources(),R.drawable.bird);      // load the selected bird image

    bird = new Bird(bitmapBird, SCREEN_WIDTH, SCREEN_HEIGHT);


    gapSize=bitmapBird.getHeight()/1.5;    // gap size


    CreatePipes();   // creating the first pipes


    holder=getHolder();

    thread = new Thread(this);
```

```
        thread.start();
    }
    public void CreatePipes()
    {
        // random place of gap
        // the gap indicates the bottom of the top pipe
        rnd=new Random();
        int gap=rnd.nextInt(SCREEN_HEIGHT-(int)(gapSize)-50)+50;  // add (50) so the gap wont be at the edges

        //create top pipe
        Bitmap bitmapTopPipe = BitmapFactory.decodeResource(getResources(), R.drawable.pipetop);
        topPipe=new PipesView(bitmapTopPipe, SCREEN_WIDTH, SCREEN_HEIGHT,gap,1);     // 1 indicates bottom pipe
        pipesOnScreen.add(topPipe);

        //create bottom pipe
        Bitmap bitmapBottomPipe = BitmapFactory.decodeResource(getResources(), R.drawable.pipebottom);
        bottomPipe=new PipesView(bitmapBottomPipe, SCREEN_WIDTH, SCREEN_HEIGHT,gap+(int)(gapSize),2);    // 2 indicates
bottom pipe
        pipesOnScreen.add(bottomPipe);
    }

    public void DrawSurface()   // drawing the game (bird,pipes, background)
    {
        if (holder.getSurface().isValid())
        {
            // draws the bird and the pipes
            canvas = holder.lockCanvas();
            canvas.drawPaint(bgPaint);
            backgroundView.draw(canvas);
            bird.draw(canvas);
            for (PipesView pipe : pipesOnScreen)
                pipe.draw(canvas);

            // code for viewing the rectangles on screen

//          Paint paint = new Paint();
```

```
//          paint.setColor(Color.RED);

//          paint.setStyle(Paint.Style.STROKE); // Set the style to STROKE (only outline)

//          paint.setStrokeWidth(5); // Set stroke width (optional, adjust as needed)

//          Rect topPipeRect=null;

//          Rect bottomPipeRect=null;

//          int rectConst=30;

//          Rect birdRect = new Rect(bird.getBirdX()+rectConst, bird.getBirdY()+rectConst, bird.getBirdX() + bird.getBirdWidth()-
rectConst, bird.getBirdY() + bird.getBirdHeight()-rectConst); // Bird's rectangle

//          for (PipesView pipe : pipesOnScreen)

//          {

//             if(pipe.getPipeY()<=0)

//                topPipeRect = new Rect(pipe.getPipeX(), pipe.getPipeY(), pipe.getPipeX() + pipe.getPipeWidth(), pipe.getPipeY() +
pipe.getPipeHeight()); // Top pipe's rectangle

//             else

//                bottomPipeRect = new Rect(pipe.getPipeX(), pipe.getPipeY(), pipe.getPipeX() + pipe.getPipeWidth(),
pipe.getPipeY() + pipe.getPipeHeight()); // Bottom pipe's rectangle

//          }

//          canvas.drawRect(birdRect, paint);

//          canvas.drawRect(topPipeRect, paint);

//          canvas.drawRect(bottomPipeRect, paint);


        holder.unlockCanvasAndPost(canvas);


    }

  }

  public void run()

  {

    gameOver=false;

    while (isRunning)

    {

        DrawSurface();


        backgroundView.move();  // move the background


        // ending the game if bird falls of the map

        if(bird.getBirdY()>=SCREEN_HEIGHT-bird.getBirdHeight()+50)   // (50) for it to look smooth

        {
```

```
    soundLose(this.getContext());   // sound when lose
    EndGame();
  }

  // ending the game if there is collision
  if(pipesOnScreen.size()>1)
    if(IsCollision())
    {
      soundLose(this.getContext());   // sound when lose
      EndGame();
    }

  // moves the bird and counts up when to generate new tubes
  bird.move();
  stepsCount++;

  if(stepsCount>=maxSteps)
  {
    CreatePipes();
    stepsCount=0;
    // make the game faster every point
    if(maxSteps>=70)
      maxSteps-=3;

    soundPoint(this.getContext());   // sound when make point

    // add point
    GravityMode.scoreCount++;

    // get the main thread's Looper
    Handler mainHandler = new Handler(Looper.getMainLooper());
    mainHandler.post(new Runnable()
    {
      @Override
      public void run()
```

```
                {
                    GravityMode.score.setText(Integer.toString(GravityMode.scoreCount));
                }
            });
        }

        // moves every pipe on screen
        for (PipesView pipe:pipesOnScreen)
            pipe.move();

        // removes pipes after they leave the screen
        for (int i=0;i<pipesOnScreen.size();i++)
            if(pipesOnScreen.get(i).getPipeX()+pipesOnScreen.get(i).getPipeWidth()<0)
                pipesOnScreen.remove(i);

        synchronized (this)
        {
            try {wait(interval);}
            catch (InterruptedException e) {e.printStackTrace();}
        }
    }
}
public boolean IsCollision()
{

    Rect topPipeRect=null;
    Rect bottomPipeRect=null;

    // Bird's rectangle
    int rectConst=30;   // (30) the image is a bit too big so it shrinks the borders of the rect
    Rect birdRect = new Rect(bird.getBirdX()+rectConst, bird.getBirdY()+rectConst, bird.getBirdX() + bird.getBirdWidth()-
rectConst, bird.getBirdY() + bird.getBirdHeight()-rectConst);

    // assigning each pipe to its position
    PipesView pipe=pipesOnScreen.get(0);
```

```
    topPipeRect = new Rect(pipe.getPipeX(), pipe.getPipeY(), pipe.getPipeX() + pipe.getPipeWidth(), pipe.getPipeY() +
pipe.getPipeHeight());

    pipe=pipesOnScreen.get(1);

    bottomPipeRect = new Rect(pipe.getPipeX(), pipe.getPipeY(), pipe.getPipeX() + pipe.getPipeWidth(), pipe.getPipeY() +
pipe.getPipeHeight());


    // true if bird touches any pipe

    return (birdRect.intersect(topPipeRect) || birdRect.intersect(bottomPipeRect));


  }

  public synchronized void EndGame()   // synchronized means only one thread can execute this method

  {

    // indicates game is over, and stops the running of the thread

    if (gameOver) return;

    gameOver = true;

    isRunning = false;


    updateScoreboardFireBase();


    // ensure the following code runs on the UI thread

    ((Activity) getContext()).runOnUiThread(new Runnable()

    {

      @Override

      public void run()

      {

        // show the game over image and call the game over dialog

        GravityMode.gameOver.setVisibility(VISIBLE);

        showGameOverDialog();

      }

    });

  }


  private void showGameOverDialog()

  {

    // ensure the UI thread is being used for the dialog creation

    ((Activity) getContext()).runOnUiThread(new Runnable()
```

```
    {
        @Override
        public void run()
        {
            new AlertDialog.Builder(getContext())
                .setTitle("Game Over")
                .setMessage("Name: "+currentPlayer.getName()+"\nScore: "+currentPlayer.getScore())
                .setCancelable(false)
                .setPositiveButton("Restart", (dialog, which) -> restartGame())
                .setNegativeButton("ScoreBoard", (dialog, which) ->
                {
                    // show Scoreboard Activity
                    Intent myIntent = new Intent(getContext(), ScoreBoardActivity.class);
                    getContext().startActivity(myIntent);
                    restartGame();     // restart after showing scoreboard
                })
                .setNeutralButton("Return Home", (dialog, which) ->
                {
                    GravityMode.scoreCount = 0;
                    GravityMode.score.setText("0");

                    // go to MainActivity
                    Intent myIntent = new Intent(getContext(), MainActivity.class);
                    getContext().startActivity(myIntent);
                })
                .show();
        }
    });
}

// updates the firebase with player name and score
private void updateScoreboardFireBase()
{
    mDatabase=FirebaseDatabase.getInstance();
    mGameRef=mDatabase.getReference("players");
```

```
        // update players score and name
        currentPlayer = new Player(MainActivity.playerName,GravityMode.scoreCount);
        mGameRef.child(currentPlayer.getName()+" : "+currentPlayer.getScore()).setValue(currentPlayer);
    }

    public void restartGame()
    {
        // restart the points and remove GameOver sign
        GravityMode.scoreCount = 0;
        GravityMode.score.setText("0");
        GravityMode.gameOver.setVisibility(INVISIBLE);


//      gameOver = false;
//      isRunning = true;
//      bird.restartBird();
//      // Clear all pipes on screen
//      while(pipesOnScreen.size()>0)
//          pipesOnScreen.remove(0);
//      CreatePipes();
//      stepsCount = 0;
//
//      // Stop the old thread safely
//      if (thread != null && thread.isAlive())
//      {
//          isRunning = false;  // Stop the game loop
//          try
//          {
//              thread.interrupt();  // Interrupt the thread to stop it
//              thread.join();      // Wait for the thread to finish
//          }
//          catch (InterruptedException e) {e.printStackTrace();}
//      }
//
```

```
//      // Start a new thread
//      isRunning = true;
//      holder=getHolder();
//      thread = new Thread(this); // Restart the thread
//      thread.start();
    }

    //makes bird jump when touch on screen
    public boolean onTouchEvent(MotionEvent event)
    {
        if (event.getAction() == MotionEvent.ACTION_DOWN)
        {
            bird.jump();
            //soundJump(this.getContext());
        }
        return super.onTouchEvent(event);  // Call the default handler for other touch events
    }
    public void soundJump(Context context)
    {
        MediaPlayer mp = MediaPlayer.create(context, R.raw.movesound);
        mp.start();
    }
    public void soundPoint(Context context)
    {
        MediaPlayer mp = MediaPlayer.create(context, R.raw.scoresound);
        mp.start();
        mp.setOnCompletionListener(mediaPlayer -> mediaPlayer.release());
    }
    public void soundLose(Context context)
    {
        MediaPlayer mp = MediaPlayer.create(context, R.raw.losingsound);
        mp.start();
        mp.setOnCompletionListener(mediaPlayer -> mediaPlayer.release());
    }
}
```

GameViewMultiplayer

מחלקה שתפקידה כמו התפקיד של GameView רק בשביל המשחק מרובה משתתפים. בנוסף לתפקידים של מחלקת GameView יש למחלקה זו תפקיד חשוב נוסף, העברת כל הנתונים ההכרחיים ל- FireBase וקליטת אותם נתונים. נתונים אלה כוללים את המיקום הרנדומלי של הפתח בין הצינורות, האם המשחק נגמר ומיקום על ציר y של הציפורים.

```
package com.example.flappybird;


import android.app.Activity;
import android.app.AlertDialog;
import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.ColorMatrix;
import android.graphics.ColorMatrixColorFilter;
import android.graphics.Paint;
import android.graphics.Rect;
import android.media.MediaPlayer;
import android.os.CountDownTimer;
import android.os.Handler;
import android.os.Looper;
import android.os.SystemClock;
import android.renderscript.Allocation;
import android.renderscript.Element;
import android.renderscript.RenderScript;
import android.renderscript.ScriptIntrinsicBlur;
import android.text.InputType;
import android.util.Log;
import android.view.MotionEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.animation.AlphaAnimation;
import android.widget.Chronometer;
```

```
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;
import java.util.Random;
import java.util.Timer;
import java.util.TimerTask;
import java.util.UUID;

public class GameViewMultiplayer extends SurfaceView implements Runnable
}
    private int SCREEN_WIDTH,SCREEN_HEIGHT;
    private int stepsCount=0,maxSteps=100;    // indicates when to make new tubes(100)
    private double gapSize;
    private Paint bgPaint;
    private Bird bird1,bird2;
    private PipesView topPipe,bottomPipe;
    private ArrayList<PipesView> pipesOnScreen;
    private Bitmap bitmapBird1,bitmapBird2 , backgroundBitmap;  // Bitmap for the birds and the background;
    private Random rnd;
    private SurfaceHolder holder;
    private Canvas canvas;
    private int interval = 15(15) // ;
    private Thread thread;
    private boolean isRunning=true,gameOver=false,isLost=false;
```

```
    private FirebaseDatabase mDatabase;

    private DatabaseReference mGameRef;




    private boolean isRunning1 = false;

    private long pauseOffset = 0; // Stores paused time


    public GameViewMultiplayer(Context context,int height,int width)

}

        super(context);

        SCREEN_WIDTH = width;

        SCREEN_HEIGHT = 2206;  // the height of my phone

        bgPaint = new Paint;()

        bgPaint.setColor(Color.WHITE);


        mDatabase = FirebaseDatabase.getInstance;()

        mGameRef = mDatabase.getReference("game");

        mGameRef.removeValue;()

        mGameRef.child("isOver").setValue(false);


        pipesOnScreen=new ArrayList<PipesView>();  // list of all the pipes on the screen




        backgroundBitmap = BitmapFactory.decodeResource(getResources(),R.drawable.skybackground);       // load
the selected background image

        backgroundBitmap = Bitmap.createScaledBitmap(backgroundBitmap,SCREEN_WIDTH, SCREEN_HEIGHT, true);
// strech the image


 //        set up the birds

        bitmapBird1 = BitmapFactory.decodeResource(getResources(),R.drawable.birdplayerone);

        bitmapBird2 = BitmapFactory.decodeResource(getResources(),R.drawable.birdplayertwo);




        bird1 = new Bird(bitmapBird1, SCREEN_WIDTH, SCREEN_HEIGHT);

        bird2 = new Bird(bitmapBird2, SCREEN_WIDTH, SCREEN_HEIGHT);
```

```
        gapSize = bitmapBird1.getHeight()/1.5;    // gap size


        holder=getHolder;()

        thread = new Thread(this);

        thread.start;()

{

    public void CreatePipes()

}

 //        random place of gap

 //        the gap indicates the bottom of the top pipe

        if(MultiplayerActivity.isPlayer2)

}

            rnd = new Random;()

            int gap = rnd.nextInt(SCREEN_HEIGHT - (int) (gapSize) - 50) + 50;  // add (50) so the gap wont be at the edges

            mGameRef.child("randomGap").setValue(gap);

{


        mGameRef.child("randomGap").addListenerForSingleValueEvent(new ValueEventListener()

}

@            Override

            public void onDataChange(@NonNull DataSnapshot dataSnapshot)

}

                if(dataSnapshot.getValue()!=null)

}

                    int gap = dataSnapshot.getValue(int.class);


//                    create top pipe

                    Bitmap bitmapTopPipe = BitmapFactory.decodeResource(getResources(), R.drawable.pipetop);

                    topPipe=new PipesView(bitmapTopPipe, SCREEN_WIDTH, SCREEN_HEIGHT,gap,1);      // 1 indicates
bottom pipe

                    pipesOnScreen.add(topPipe);


//                    create bottom pipe

                    Bitmap bitmapBottomPipe = BitmapFactory.decodeResource(getResources(), R.drawable.pipebottom);

                    bottomPipe=new PipesView(bitmapBottomPipe, SCREEN_WIDTH,
SCREEN_HEIGHT,gap+(int)(gapSize),2);    // 2 indicates bottom pipe

                    pipesOnScreen.add(bottomPipe);
```

```
{
{

@           Override
            public void onCancelled(@NonNull DatabaseError error)}
{
;({
{
    public void DrawSurface()   // drawing the game (bird,pipes, background)
}
        if (holder.getSurface().isValid())
}
 //           draws the bird and the pipes
            canvas = holder.lockCanvas;()
            canvas.drawPaint(bgPaint);
            canvas.drawBitmap(backgroundBitmap, 0, 0, null);
            bird1.draw(canvas);
            bird2.draw(canvas);

            for (PipesView pipe : pipesOnScreen)
                pipe.draw(canvas);

            holder.unlockCanvasAndPost(canvas);
{
{

    public void run()
}
        gameOver=false;

        Handler uiHandler = new Handler(Looper.getMainLooper()); // Main thread handler
        uiHandler.post<- ())
}
            GravityModeMultiplayer.chronometer.setBase(SystemClock.elapsedRealtime()); // Reset base time
            GravityModeMultiplayer.chronometer.start(); // Start Chronometer
;({
```

```
        while (isRunning)
}

        long elapsedMillis = SystemClock.elapsedRealtime() - GravityModeMultiplayer.chronometer.getBase();   // to
synchronize the games

        DrawSurface;()

 //        ending the game if one of the birds bird falls of the map
 //        sending the player who won
        if(MultiplayerActivity.isPlayer2)
}
 //            check if player 1 lost to end the game
          mGameRef.child("isOver").addListenerForSingleValueEvent(new ValueEventListener()
}
@                Override
              public void onDataChange(@NonNull DataSnapshot snapshot)
}
                 if(snapshot.getValue()!=null&&snapshot.getValue().equals(true))
                    EndGame;()
{
@                Override
              public void onCancelled(@NonNull DatabaseError error)
{}
;({

          if(bird2.getBirdY()>=SCREEN_HEIGHT-bird2.getBirdHeight()+50)   // (50) for it to look smooth
}
              mGameRef.child("isOver").setValue(true);
              isLost=true;
{
{
        else
}
 //            check if player 2 lost to end the game
```

```
                mGameRef.child("isOver").addListenerForSingleValueEvent(new ValueEventListener()
}
@                   Override
                public void onDataChange(@NonNull DataSnapshot snapshot)
}
                    if(snapshot.getValue()!=null&&snapshot.getValue().equals(true))
                        EndGame;()
{

@                   Override
                public void onCancelled(@NonNull DatabaseError error)
{}
;({

                if(bird1.getBirdY()>=SCREEN_HEIGHT-bird1.getBirdHeight()+50)   // (50) for it to look smooth
}
                mGameRef.child("isOver").setValue(true);
                isLost=true;
{
{

 //             ending the game if there is collision
            if(pipesOnScreen.size()>1)
                if(IsCollision())
}
                if(MultiplayerActivity.isPlayer2)
}
                    mGameRef.child("isOver").setValue(true);
                    isLost=true;
{
                else
}
                    mGameRef.child("isOver").setValue(true);
                    isLost=true;
{
{
```

```
 //          move the birds according to the value in firebase
        if(MultiplayerActivity.isPlayer2)
}
            bird2.move;()
            mGameRef.child("player"+2).child("Y").setValue(bird2.getBirdY());
            mGameRef.child("player"+1).child("Y").addListenerForSingleValueEvent(new ValueEventListener()
}
@                Override
            public void onDataChange(@NonNull DataSnapshot snapshot)
}
                if(snapshot.getValue()!=null)
                    bird1.setBirdY(snapshot.getValue(int.class));
{
@                Override
            public void onCancelled(@NonNull DatabaseError error){}
;({
{
        else
}
            bird1.move;()
            mGameRef.child("player"+1).child("Y").setValue(bird1.getBirdY());
            mGameRef.child("player"+2).child("Y").addListenerForSingleValueEvent(new ValueEventListener()
}
@                Override
            public void onDataChange(@NonNull DataSnapshot snapshot)
}
                if(snapshot.getValue()!=null)
                    bird2.setBirdY(snapshot.getValue(int.class));
{
@                Override
            public void onCancelled(@NonNull DatabaseError error){}
;({
{
```

```
 //          up the steps to generate new pipes sync the games
          stepsCount;++
          if (elapsedMillis>=4000)
}
             uiHandler.post<- ())
}
                CreatePipes(); // Ensure CreatePipes() only modifies UI
                GravityModeMultiplayer.chronometer.setBase(SystemClock.elapsedRealtime());
                pauseOffset = 0;
                GravityModeMultiplayer.chronometer.stop;()
;({
{

 //          moves every pipe on screen
          for (PipesView pipe:pipesOnScreen)
             pipe.move;()

 //          removes pipes after they leave the screen
          for (int i=0;i<pipesOnScreen.size();i++)
             if(pipesOnScreen.get(i).getPipeX()+pipesOnScreen.get(i).getPipeWidth()<0)
                pipesOnScreen.remove(i);

          synchronized (this)
}
             try {wait(interval);}
             catch (InterruptedException e) {e.printStackTrace;()}
{
{
{
   public Boolean IsCollision()
}
 //       it checks collision only for the player who is playing
 //       this allows the game to work even when there is delay

       Rect topPipeRect=null;
       Rect bottomPipeRect=null;
```

```
//      Bird's rectangle

        int rectConst=30;   // (30) the image is a bit too big so it shrinks the borders of the rect

        Rect birdRect;

        if(!MultiplayerActivity.isPlayer2)

            birdRect = new Rect(bird1.getBirdX()+rectConst, bird1.getBirdY()+rectConst, bird1.getBirdX() +
bird1.getBirdWidth()-rectConst, bird1.getBirdY() + bird1.getBirdHeight()-rectConst);

        else

            birdRect = new Rect(bird2.getBirdX()+rectConst, bird2.getBirdY()+rectConst, bird2.getBirdX() +
bird2.getBirdWidth()-rectConst, bird2.getBirdY() + bird2.getBirdHeight()-rectConst);



//      assigning each pipe to its position

        PipesView pipe=pipesOnScreen.get;(0)

        topPipeRect = new Rect(pipe.getPipeX(), pipe.getPipeY(), pipe.getPipeX() + pipe.getPipeWidth(),
pipe.getPipeY() + pipe.getPipeHeight());

        pipe=pipesOnScreen.get;(1)

        bottomPipeRect = new Rect(pipe.getPipeX(), pipe.getPipeY(), pipe.getPipeX() + pipe.getPipeWidth(),
pipe.getPipeY() + pipe.getPipeHeight());



//      return the player who lost

        if(birdRect.intersect(topPipeRect) || birdRect.intersect(bottomPipeRect))

            return true;

        else if (birdRect.intersect(topPipeRect) || birdRect.intersect(bottomPipeRect))

            return true;

        else

            return false;



{

    public synchronized void EndGame()

}

//      indicates game is over, and stops the running of the thread

        if (gameOver) return;

        gameOver = true;

        isRunning = false;


//      ensure the following code runs on the UI thread

        ((Activity) getContext()).runOnUiThread(new Runnable()

}
```

```
@           Override
          public void run()
}
 //              show the game over image
            GravityModeMultiplayer.gameOver.setVisibility(VISIBLE);
            String message;""=
            if(MultiplayerActivity.isPlayer2)
}
                if(isLost)
                    message = "LOST;"
                else
                    message="WON;"
{
            else
}
                if(isLost)
                    message = "LOST;"
                else
                    message = "WON;"
{

             showGameOverDialog(message);
{
;({
{

    private void showGameOverDialog(String message)
}
 //       ensure the UI thread is being used for the dialog creation
       ((Activity) getContext()).runOnUiThread(new Runnable()
}
@           Override
          public void run()
}
            new AlertDialog.Builder(getContext())
.                   setTitle("Game Over")
```

```
.                    setMessage("YOU "+ message)
.                    setCancelable(false)
.                    setNeutralButton("Return Home", (dialog, which<- (
}
                       Intent myIntent = new Intent(getContext(), MainActivity.class);
                       getContext().startActivity(myIntent);
({
.                    show;()
{
;({
{

 //    makes bird jump when touch on screen
    public boolean onTouchEvent(MotionEvent event)
}
        if (event.getAction() == MotionEvent.ACTION_DOWN)
}
            if(MultiplayerActivity.isPlayer2)
                bird2.jump;()
            else
                bird1.jump;()
{
        return super.onTouchEvent(event);
{
{
```

Resources

שימוש בAdapters

בקובץ ScoreBoardActivity נעשה שימוש בArrayAdapter על מנת להעביר את כל שמות השחקנים והניקוד שלהם לרשימת Listview.

שימוש בThreads

בקבצים **GameView** ו-**GameViewMultiplayer** נעשה שימוש ב**Thread** בכדי לאפשר למערכת להיות יותר יעילה ולעשות מספר דברים בו זמנית.

שימוש בHandlers

נעשה שימוש בקבצים GameView **ו-**GameViewMultiplayer בכדי לאפשר להציג את ניקוד השחקן מעל לתצוגת View ולגשת לטיימר המופעל במשחק המרובה משתתפים.

Layout

activity_backgroung_selection.xml – מתאר את מסך בחירת הרקע ומציג אפשרויות שונות לבחירת הרקע.

activity_bird_selection.xml – מתאר את מסך בחירת הציפור ומציג אפשרויות שונות לבחירת הציפור במשחק.

activity_gravity_mode.xml – מתאר את המסך שמציג את המשחק הרגיל. המציג את הניקוד, שלט Game Over והמסך ששומר את FrameLayout שמציג את המשחק עצמו.

activity_gravity_mode_multiplayer.xml – מתאר את המסך שמציג את המשחק מרובה משתתפים. מציג את שלט Game Over, המסך ששומר את FrameLayout שמציג את המשחק עצמו ושומר את הטיימר האחראי לתזמון בין 2 השחקנים.

activity_main.xml – מתאר את הסמך הראשי של המשחק. מכיל את כל הכפתורים המנווטים את המשחק כמו התחלת המשחק, בחירת ציפור ובחירת רקע.

activity_multiplayer.xml – מתאר את המסך שבו קיימת הבחירה בין פתיחת משחק חדש והצטרפות למשחק קיים. מסך זה מנווט את המשחק מרובה המשתתפים.

activity_score_board.xml – מתאר את המסך שמציג את לוח התוצאות.

Drawables

bird.png – התמונה של הציפור הראשית.

birdplayerone.png – התמונה של הציפור של שחקן אחד במשחק מרובה משתתפים.

birdplayertwo.png – התמונה של הציפור של שחקן שתיים במשחק מרובה משתתפים.

brownishbird.png – אחת מן הציפורים שניתן לבחור.

cryingbird.png – אחת מן הציפורים שניתן לבחור.

greenbirdwithhat.png – אחת מן הציפורים שניתן לבחור.

nonchalant.png – אחת מן הציפורים שניתן לבחור.

oldbirdwithhat.png – אחת מן הציפורים שניתן לבחור.

pinkbird.png – אחת מן הציפורים שניתן לבחור.

pinkbirdcrying.png – אחת מן הציפורים שניתן לבחור.

yellowbird.png – אחת מן הציפורים שניתן לבחור.

clearsky.png – הרקע הכללי של המשחק.

skybackground.png – אחד מן הרקעים שניתן לבחור.

spacebackground.png – אחד מן הרקעים שניתן לבחור.

forestbackground.jpg – אחד מן הרקעים שניתן לבחור.

desertbackground.jpg - אחד מן הרקעים שניתן לבחור.

flappybirdlogo.png – לוגו המשחק.

gameover.png – שלט המשחק נגמר.

icon.png – תמונת האפליקציה.

pipebottom.png – תמונת הצינור התחתון.

pipetop.png – תמונת הצינור העליון.

waiting.png – השלט שמודיע שמחכים לשחקן 2 שיצטרף למשחק מרובה שחקנים.

Raw

losingsound.mp3 – הצליל שמשמיעים כשיש פסילה.

movesound.mp3 – הצליל שמשמיעים כשהציפור קופצת.

scoresound.mp3 – הצליל שמשמיעים כשיש נקודה.

סיכום אישי – רפלקציה

העבודה על הפרויקט הייתה תהליך מאוד מיוחד עבורי, תהליך מלא בתסכול וגם בתחושות של הצלחה. היו הרבה אתגרים בדרך הכוללים קושי ביצירת אנימציה, קושי ביצירת התנגשות חלקה בין הציפור לצינורות, קושי בהכנת משחק מרובה משתתפים ועוד הרבה. בכדי להתגבר על כל אתגר שכזה ישבתי ימים רבים מלאים במחשבה ועבודה קשה על מנת לפתור כל בעיה וליצור את הפרויקט הכי טוב שאני יכול. במהלך תהליך העבודה נתקלתי בנושאים חדשים לי אשר לא ידעתי עליהם דבר. חקרתי רבות על דרכי יצירת אנימציה ועל אופן השימוש ב – FireBase במטרה לממש את מטרותיי בפרויקט. החקר העצמאי, למרות שלעיתים היה מתסכל, העניק לי כלים חשובים לחיים הכוללים את היכולת ללמידה ועצמאית וחקירה והיכולת להתמודד לבד עם בעיות. למדתי מהתהליך שלפעמים פנייה לעזרה אינה דבר רע. קיימים פרויקטים רבים של תלמידים אחרים שהם דומים לפרויקט שלי לכן אם אחד מבין יותר בדבר שאני מבין בו פחות, פנייה לעזרה תתרום לשני הצדדים. למידת עמיתים אכן תורמת רבות להתקדמותם של שני האנשים.

כיום כשאני מסתכל על הפרויקט שלי אני מבין כי הייתי יכול לבצע דברים אחרת. הייתי משקיע יותר זמן ביצירת האנימציה כדי שתראה הכי חלק שניתן, נוסף על כך הייתי חוקר יותר על Bitmap ובכך מנסה ליצור התנגשות טובה יותר בין הגופים במשחק. אם היו לי משאבים נוספים הייתי משפר את המשחק בכך שיכלול בתוכו חנות שאפשר לרכוש בה את כל השיפורים לציפור והרקעים האחרים, נקודות לחנות צוברים בעזרת נקודות במשחק. אם היה לי יותר זמן הייתי מוסיף את החלק הזה.

לסיכום, העבודה על הפרויקט היוותה חלק משמעותי מתהליך הלמידה שלי בתחום המחשבים והתובנות שקיבלתי ממנה יישארו איתי גם בעתיד. למרות שהתוצר הסופי אינו מושלם, אני מרוצה שהצלחתי לייצר כמעט את כל הדברים שרציתי שיהיו במשחק ושהוא בסוף יצא משחק טוב ועובד.