



שם התלמיד: גל מונדשיין

ת.ז. תלמיד: 215321753

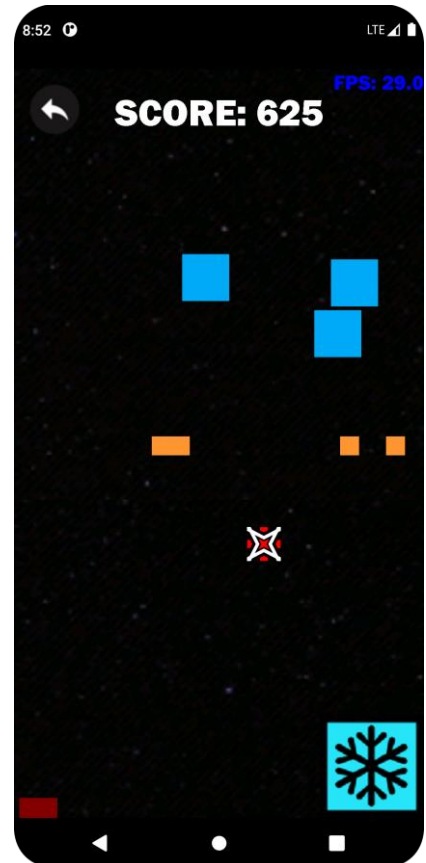
שם הפרויקט: Rush&Dash

שם החלופה: תכנון ותכנות מערכות טלפונים ניידים

שם המנחה: אורית רוזנבאום

שם בית הספר: קריית שרת חולון

תאריך הגשה: קיץ תשפ"ג



תוכן עניינים:

Contents

5	1. מבוא
5	1.1. הרקע לפרויקט
5	1.2. תהליך המחקר
5	1.3. אתגרים מרכזיים
6	2. תיאור תחום הידע:
7	3. מבנה / ארכיטקטורה של הפרויקט
7	3.1. תכנון ותיעוד מסכי הפרויקט
7	3.1.1. מסך ספלאש – SplashScreenActivity
7	3.1.2. מסך התחברות – LoginActivity
8	3.1.3. מסך איפוס סיסמה – ForgotActivity
8	3.1.4. מסך הרשמה – SignUpActivity
9	3.1.5. מסך ראשי – HubActivity
9	3.1.6. מסך הגדרות – OptionsActivity
10	3.1.7. מסך טבלת שיאים – RankingsActivity
10	3.1.8. מסך חנות – ShopActivity
10	3.1.8.1. פרגמנט תצוגה מקדימה – PreviewFragment
11	3.1.8.2. פרגמנט רשימת פריטים – InventoryFragment
11	3.1.8.3. פרגמנט רכישת פריט – ShopBuyFragment
12	3.1.9. מסך בחירת שלב – LevelSelectActivity
13	3.1.10. מסך המשחק – GameScreenActivity
14	3.2. דיאלוגים
16	3.3. תרשים מסכים המתאר את היררכיית המסכים והמעברים ביניהם:
17	3.4. תיאור מחלקות הפרויקט – UML
19	4. מימוש הפרויקט
19	4.1. המחלקה SplashScreenActivity
19	4.2. המחלקה LoginActivity
21	4.2.1. המחלקה User
22	4.2.2. המחלקה Statics
22	4.2.3. המחלקה VibrationService
23	4.3. המחלקה ForgotActivity
24	4.4. המחלקה SignUpActivity
25	4.5. המחלקה HubActivity

26.....	OptionsActivity	המחלקה	4.6
27.....	RankingsActivity	המחלקה	4.7
28.....	ShopActivity	המחלקה	4.8
29.....	PreviewFragment	המחלקה	4.8.1
29.....	InventoryFragment	המחלקה	4.8.2
30.....	ShopBuyFragment	המחלקה	4.8.3
31.....	PurchaseConfirmDialog	המחלקה	4.8.4
32.....	ShopInfoDialog	המחלקה	4.8.5
32.....	שימוש באדפטר באפליקציה על מנת ליצור חנות		4.8.6
33.....	ShopItem	המחלקה	4.8.6.1
33.....	ShopItemAdapter	המחלקה	4.8.6.2
34.....	LevelSelectActivity	המחלקה	4.9
34.....	DifficultyDialog	המחלקה	4.9.1
35.....	LevelInfoDialog	המחלקה	4.9.2
36.....	ObstInfoDialog	המחלקה	4.9.3
37.....	GameScreenActivity	המחלקה	4.10
39.....	BroadcastReceiver	שימוש במחלקה כדי לבדוק שינויים בחיבור של המשתמש לאינטרנט	4.10.1
39.....	BossLevelCompleteDialog	המחלקה	4.10.2
40.....	DemoGameOverDialog	המחלקה	4.10.3
40.....	GameOverDialog	המחלקה	4.10.4
41.....	Canvas	שימוש במחלקה כדי ליצור לוח משחק	4.10.5
42.....	GamePanel	המחלקה	4.10.5.1
44.....	MainThread	המחלקה	4.10.5.2
44.....	MainThread	תרשים זרימה של הרצה של התרד	4.10.5.2.1
45.....	ObstacleArray	המחלקה	4.10.5.3
45.....	ObstacleArrayManager	המחלקה	4.10.5.4
46.....	ObstacleThread	המחלקה	4.10.5.5
46.....	ScoreThread	המחלקה	4.10.5.6
47.....	PowerUp	המחלקה	4.10.6
48.....	PowerUp	תרשים זרימה של המחלקה	4.10.6.1
48.....	GameSensors	המחלקה	4.10.7
49.....	SFX	המחלקה	4.10.8
49.....	RectGame	המחלקה	4.10.9
50.....	PointGame	המחלקה	4.10.9.1
51.....	Player	המחלקה	4.10.9.2
51.....	BonusObj	המחלקה	4.10.9.3

53.....	BonusObj	המחלקה של זרימה	4.10.9.3.1
53.....	AlphaObj	המחלקה	4.10.9.4
53.....	MovingRectObj	המחלקה	4.10.9.5
54.....	GapObj	המחלקה	4.10.9.6
55.....	WarningObj	המחלקה	4.10.9.7
55.....	DownUpObj	המחלקה	4.10.9.8
56.....	MovingGapObj	המחלקה	4.10.9.9
57.....	CrossingObj	המחלקה	4.10.9.10
57.....	CrossingSingleObj	המחלקה	4.10.9.11
58.....	BounceObj	המחלקה	4.10.9.12
58.....	PositionSensorObj	המחלקה	4.10.9.13
60.....	Resources	תייקית	4.11
60.....	Layouts		4.11.1
61.....	Drawables		4.11.2
64.....	Anim		4.11.3
64.....	Fonts		4.11.4
64.....	Menu		4.11.5
64.....	Mipmap		4.11.6
64.....	Raws		4.11.7
65.....	Firebase	בסיס הנתונים	4.11.8
68.....		הרשאות	4.11.9
69.....		מדריך למשתמש	5
70.....		סיכום אישי / רפלקציה	6
71.....		ביבליוגרפיה	7
72.....		נספחים	8

1. מבוא

1.1. הרקע לפרויקט

שם הפרויקט – Rush&Dash

תיאור קצר של הפרויקט – האפליקציה Rush&Dash היא משחק שבו השחקן שולט על דמות שצריכה להתחמק ממכשולים, וככל שעולים בשלב השחקן יצטרך להתחמק ממכשולים מסובכים יותר מהשלב הקודמים. מטרתו של השחקן היא לסיים את השלב ולעבור לשלב הבא עד שלב 25 שהוא השלב האחרון. יש שני סוגי שלבים, שלב רגיל שצריך להגיע בו למספר נקודות מסויים כדי לפתוח את השלב הבא, ושלב בוס שמופיע בסוף העולם. בכל מסך משחק ניתן למצוא בונסים שונים שיכולים לעזור לשחקן או לעקב אותו.

קהל היעד – המשחק מיועד לאנשים מכל הגילאים.

הסיבות לבחירת הפרויקט – אני אוהב לשחק במשחקי מחשב בסגנון התחמקות ופעולה. לכן, רציתי ליצור משחק מהנה ומאתגר שדומה לסוג המשחקים שאני אוהב לשחק.

1.2. תהליך המחקר

כאשר תכננתי ליצור את המשחק הזה, ראיתי משחק דומה שנקרא "Geometry Dash" שאני נהנה לשחק בו, אבל שיניתי אותו כדי שהוא יהיה מקורי יותר, ודומה יותר לסוג המשחק שאני הייתי נהנה לשחק. בניגוד למשחק המקורי, גרמתי לדמות של המשחק לנוע בחופשיות באוויר במקום שהדמות תקפוץ רק על הרצפה. בנוסף, שיניתי את המשחק לכך שהוא יהיה משוחק לאורך במקום לרוחב, כי לדעתי יותר נוח לשחק משחקים כשהם לאורך. במשחק שלי הוספתי את השלבים ה"רגילים" ובהם השחקן מקבל ניקוד לכל שלב, מה שלא היה במשחק המקורי, כדי שאני אוכל להציג בטבלת שיאים את השחקנים הכי טובים לפי מספר הנקודות שלהם.

1.3. אתגרים מרכזיים

במהלך העבודה על הפרויקט נתקלתי בקשיים שחלקם היו מאתגרים מאוד. העבודה על הפרויקט הייתה במסגרת אנדרואיד סטודיו, שחלקה למדנו בכיתה, אבל עם רובה התמודדתי לבד במסגרת העבודה על הפרויקט.

האתגר המרכזי שאיתו התמודדתי היה לחשוב על דרך שבה אני אוכל לדעת אילו סוגי אובייקטים יוצרו על המסך, ולדעת איך לצייר את כולם על המסך מבלי להאט את האפליקציה, ומבלי לאבד חלק מהאובייקטים על ידי יצירת אובייקטים חדשים.

הפתרון שבחרתי לבעיה זו הוא שימוש ברשימה של מערכים מסוג מספרים שלמים, ובה כל מספר מייצג אובייקט על המסך, ובאמצעות לולאה הייתי יכול לגשת לפעולות של כל אחד מהאובייקטים שעל המסך כדי לצייר אותם על הקנבס, לעדכן אותם, או לבדוק אם השחקן פגע בהם, וכו'.

אתגר נוסף שהתמודדתי איתו היה למצוא דרך יצירתית להציג את חנות המשחק, שלא תצריך שימוש ביותר מדי מסכים חדשים עבור כל סוג חנות, ושתהיה קלה להבנה על ידי המשתמש.

הפתרון שבחרתי לבעיה זו הוא שימוש ב-Navigation Drawer Activity ופרגמנטים, כך שאני אוכל להחליף את תוכן המסך באמצעות האפשרויות של המגירה, וכשאני צריך לבחור סוג חנות, אני אוכל להשתמש בטאג שהפרגמנט מקבל מהאקטיביטי כדי לדעת מהו סוג החנות, במקום ליצור פרגמנט או אקטיביטי חדש לכל סוג חנות.

2. תיאור תחום הידע:

המשחק הוא לשחקן בודד, המשחק מורכב מלוח משחק שעליו השחקן חופשי לנוע בכל בכיוונים, במעלה לוח המשחק השחקן רואה כמה נקודות הוא אסף או כמה זמן נשאר לו לפי סוג השלב.

סוגי הנתונים שהשתמשתי בהם הם גודל הלוח, מחלקת המשתמש, ומחלקת מנהל מערך המכשולים בשביל האובייקטים שעל המסך.

הלוח מוצג באמצעות קנבס, תכונות הלוח תלויות במספר השלב ובמחלקת המשתמש ויקבעו את האובייקטים שיופיעו בשלב ואת המהירות שלהם.

השתמשתי ברשימה של מערכים דו מימדיים מסוג מספרים שלמים כדי לשמור את המידע של האובייקטים שמופיעים על המסך.

קיים במהלך הפרויקט עדכון וקריאה של מידע. קראתי נתונים כמו רמת הקושי שבחר המשתמש והשתמשתי בנתון הזה כדי לעצב את השלב בהתאם. עדכנתי נתונים כמו הנקודות שהשחקן השיג בשלב מסוים והשתמשתי בנתון זה כדי לעדכן טבלאות שיאים וההתקדמות שלו לקראת השלב הבא.

3. מבנה / ארכיטקטורה של הפרויקט

3.1. תכנון ותיעוד מסכי הפרויקט

3.1.1. מסך ספלאש – **SplashScreenActivity**

מסך פתיחה זה מופיע באופן אוטומטי כאשר פותחים את האפליקציה, ועובר באופן אוטומטי כעבור 3.3 שניות באמצעות האנדלר למסך ההתחברות.

מסך זה משתמש באנימציה ובה ניתן לראות את סמל המשחק גדל והבהירות שלו יורדת.



3.1.2. מסך ההתחברות – **LoginActivity**

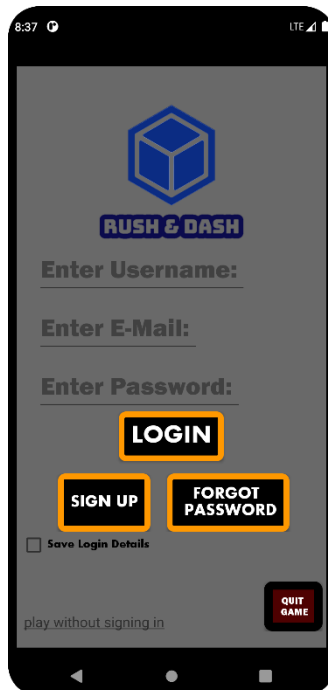
במסך זה ניתן לראות שלוש תיבות טקסט שניתן לכתוב בהם את שם המשתמש, המייל, והסיסמה.

במסך יש כפתור "LOGIN" שבודק אם הפרטים שהוכנסו לתיבות הטקסט נכונים. לאחר מכן, האפליקציה תבדוק אם המשתמש קיים בבסיס הנתונים, ואם הוא קיים, תחבר את המשתמש לחשבון שלו ותעביר אותו למסך הראשי. אם הוא נכשל בחיבור המשתמש האפליקציה תציג Toast בהתאם.

על המסך מוצגת תיבת סימון שבה המשתמש יכול לבחור אם לשמור את הנתונים, כדי שיוצגו אוטומטית בהתחברות הבאה, הנתונים נשמרים דרך SharedPreferences.

למסך יש שני כפתורים נוספים "SIGN UP" ו-"FORGOT PASSWORD", שניהם מעבירים למסך ההרשמה ומסך "שכחתי סיסמה" בהתאם.

לבסוף, בפניה מצד שמאל למטה של המסך יש כפתור שעליו כתוב "play without signing in" ו"כאשר לוחצים עליו הוא מעביר את המשתמש למסך המשחק ועליו "שלב הדמו" כדי שהשחקן יוכל להתרשם מהמשחק לפני שהוא נרשם אליו.



3.1.3. מסך איפוס סיסמה – ForgotActivity

מסך זה מופיע לאחר שלוחצים על הכפתור "FORGOT PASSWORD".

במסך זה יש ארבע תיבות טקסט ובהם המשתמש כותב את שם המשתמש והמייל שצריך שסיסמתו תתאפס.

לאחר שהמשתמש סיים לכתוב את פרטיו, הוא יוכל ללחוץ על הכפתור שינוי הסיסמה. לחיצה על כפתור זה תבדוק את נכונות הפרטים שהמשתמש כתב, לדוגמה, אם המייל ושם המשתמש מתאימים זה לזה וקיימים במערכת.

אם פרטי ההתחברות שגויים, יוצג למשתמש Toast בהתאם. אם פרטי ההתחברות נכונים, האפליקציה תשלח למייל של המשתמש מייל שיאפשר לו לשנות את הסיסמא.

בפינה שמאל למעלה יש כפתור שמחזיר את המשתמש למסך ההתחברות, אם הוא שינה את דעתו בקשר לשינוי הסיסמה.

3.1.4. מסך הרשמה – SignUpActivity

מסך זה מופיע לאחר שלוחצים על הכפתור "SIGN UP".

במסך זה יש ארבע תיבות טקסט ובהם המשתמש כותב את הפרטי התחברות שלו.

לאחר שהמשתמש סיים לכתוב את פרטיו, הוא יוכל ללחוץ על כפתור השמירה. לחיצה על כפתור זה תבדוק את נכונות הפרטים שהמשתמש כתב, לדוגמה, האם שם המשתמש או המייל כבר בשימוש, האם הסיסמה ואישור הסיסמה זהים.

אם פרטי ההתחברות שגויים, זה יוצג למשתמש באמצעות Toast. אם פרטי ההתחברות נכונים, האפליקציה תוסיף את המשתמש לבסיס הנתונים ותחזיר אותו למסך ההתחברות.

בפינה שמאל למעלה יש כפתור שמחזיר את המשתמש למסך ההתחברות, אם הוא שינה את דעתו בקשר ליצירת משתמש.

3.1.5. מסך ראשי – HubActivity

מסך זה מופיע לאחר שלוחצים על כפתור ההתחברות, וממנו אפשר לעבור למסך המשחק, ולמסכים אחרים באפליקציה.

במסך יש חמישה כפתורים שכל אחד מהם מעביר למסך אחר.

הכפתור הראשון, שעליו כתוב PLAY מעביר למסך בחירת שלב.

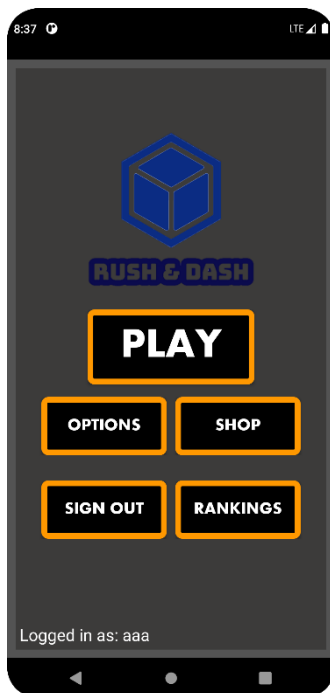
הכפתור השני, שעליו כתוב OPTIONS מעביר למסך ההגדרות.

הכפתור השלישי, שעליו כתוב SHOP מעביר למסך החנות.

הכפתור הרביעי, שעליו כתוב SIGN OUT מנתק את המשתמש ומחזיר את המשתמש למסך ההתחברות.

הכפתור החמישי, שעליו כתוב RANKINGS מעביר למסך טבלת השיאים.

בפינת המסך כתוב את השם של המשתמש שמחובר כרגע.



3.1.6. מסך הגדרות – OptionsActivity

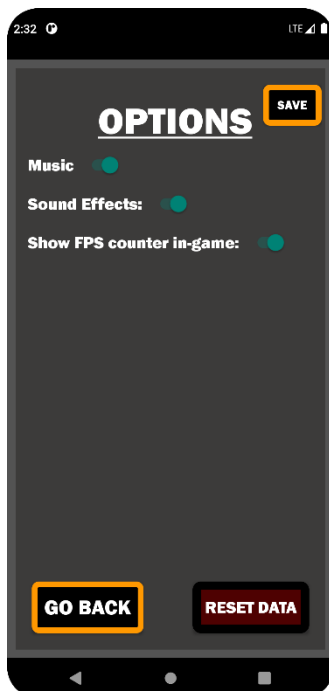
במסך זה השחקן יכול לשנות את הגדרות המשחק באמצעות שינוי מפסקים שמייצגים בוליאנים ואחרים על חלקים מסויימים במשחק.

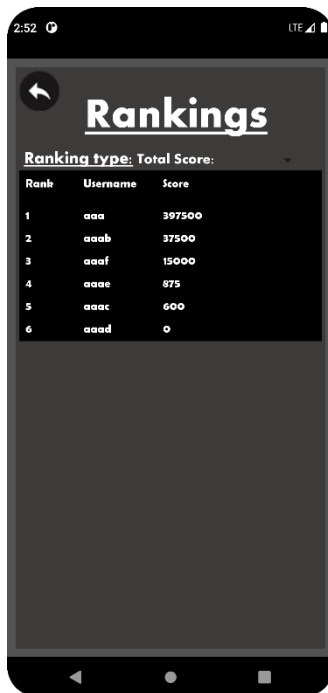
המפסקים הם: מפסק למוזיקה, מפסק לצלילים, ומפסק להצגת כמות הפריימים בשנייה בזמן המשחק.

השחקן יכול ללחוץ על כפתור השמירה בפינה העליונה של המסך כדי לשמור את ההגדרות.

בתחתית המסך יש כפתור שבו השחקן יכול לאתחל את כל הנתונים שלו אם הוא רוצה להתחיל הכל מחדש.

יש במסך גם כפתור חזרה למסך התפריט הראשי.





3.1.7. RankingsActivity – מסך טבלת שיאים

במסך זה מוצגות טבלאות שיאים בתחומים שונים והשחקן יכול לבחור באמצעות ספינר איזה סוג טבלת שיאים להראות.

בכל טבלת שיאים מוצגים עשרת המשתמשים הראשונים בקטגוריה.

סוגי טבלאות השיאים הם: כמות הנקודות הכוללת, כמות הפעמים שהשחקן נפסל, וכמות הפריטים שהמשתמש רכש.

במסך יש גם כפתור חזרה למסך התפריט הראשי.

3.1.8. ShopActivity - מסך חנות

מסך זה מחולק לפרגמנטים שמשתנים באמצעות מגירה שהמשתמש יכול ללחוץ על כל אחת מהאפשרויות שמופיעות בשביל להחליף את המסך.

הפרגמנטים הם: פרגמנט תצוגה מקדימה, רשימת פריטים שבבעלות השחקן, וחנות לדמות, רקע, בונוס, וכו'.

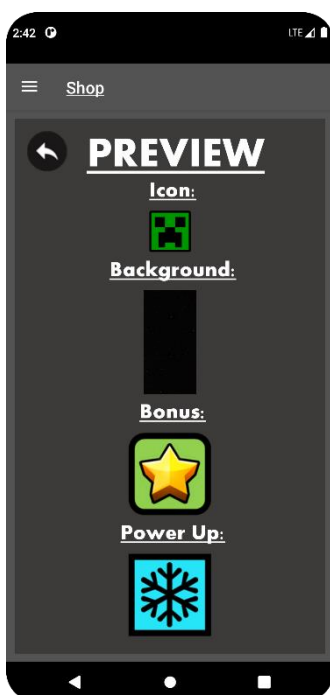
כאשר האקטיביטי של החנות מתחילה, הפרגמנט של התצוגה המקדימה נפתח אוטומטית.

בכל פרגמנט יש כפתור חזרה מהאקטיביטי של החנות אל האקטיביטי של התפריט הראשי.

3.1.8.1. PreviewFragment – פרגמנט תצוגה מקדימה

פרגמנט זה הוא הראשון שהמשתמש רואה כשהוא נכנס לאקטיביטי של החנות.

בפרגמנט זה מוצגים לשחקן הפריטים שהוא בחר.



3.1.8.2. InventoryFragment – פרגמנט רשימת פריטים

פרגמנט זה השחקן רואה ארבע תצוגות גלילה, אחת לכל סוג פריט, והוא יכול להסתכל על כל הפריטים שרכש.

3.1.8.3. ShopBuyFragment – פרגמנט רכישת פריט

פרגמנט זה מקבל את סוג החנות בתור פרמטר לפי הפרגמנט שהמשתמש לחץ עליו, ומציג חנות שמתאימה לאותו סוג הפריט.

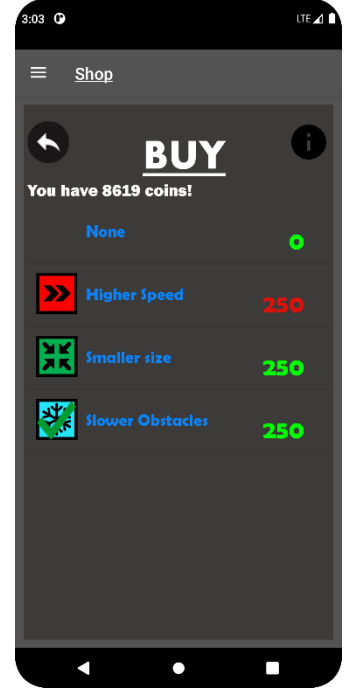
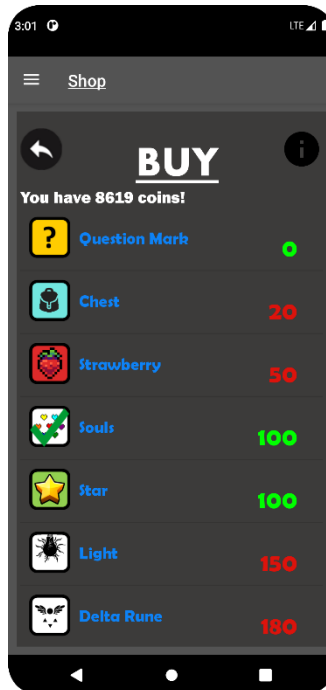
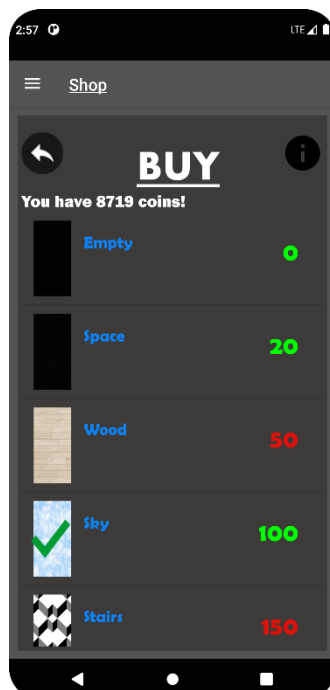
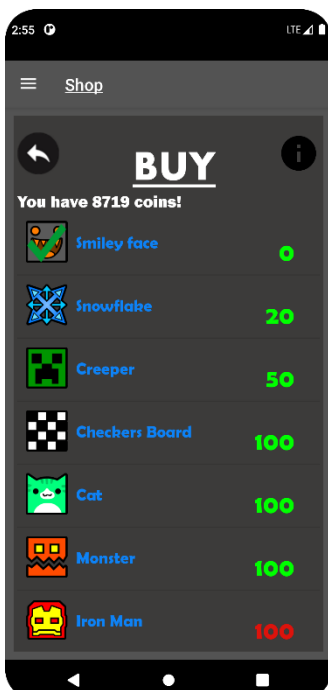
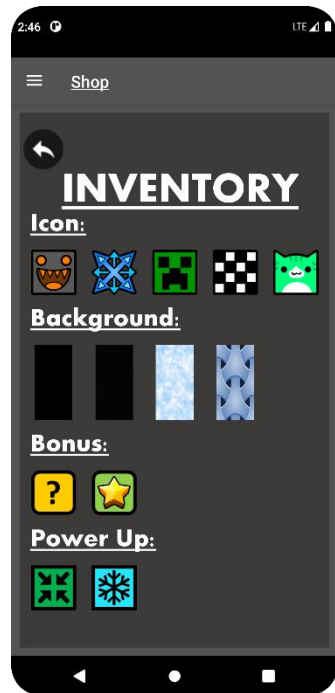
בתחילת המסך המשתמש רואה את כמות המטבעות שיש לו כרגע.

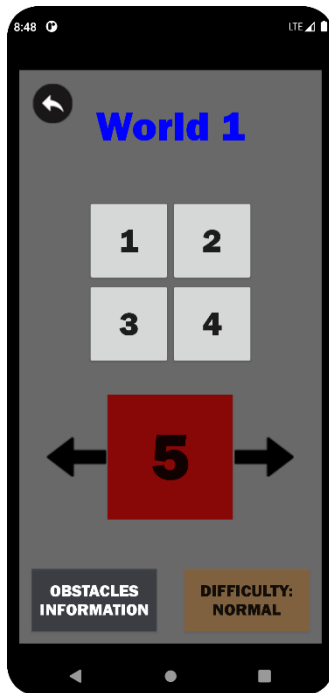
כל שורה בחנות נוצרת באמצעות המחלקה ShopItem, ולפיה, השורה מורכבת מתמונה, שם, ומחיר.

החנות משתמשת באדפטר ShopItemAdapter כדי ליצור את רשימת הפריטים.

צבע המחיר ישתנה מאדום לירוק אם המשתמש רכש אותו, ועל התמונה יהיה סימן של וי אם הוא הפריט שהשחקן בחר בו. אם השחקן לא קנה את הפריט, לחיצה עליו תפתח את הדיאלוג רכישת פריט PurchaseConfirmDialog.

בפינה הימנית למעלה יש כפתור יחודי לכל סוג חנות שיסביר על סוג הפריט.





3.1.9. מסך בחירת שלב – LevelSelectActivity

במסך זה השחקן יכול לבחור את השלב בו הוא רוצה לשחק.

במשחק יש עשרים וחמישה שלבים שמחולקים לחמישה עולמות כשבכל עולם יש חמישה שלבים. המסך הראשון שהשחקן רואה הוא העולם האחרון שפתוח עבור השחקן. השחקן יכול להשתמש בחיצים כדי לעבור קדימה ואחורה מהעולם הראשון עד לחמישי.

לחיצה על אחד השלבים יפתח את הדיאלוג LevelInfoDialog שמתאים לשלב שנבחר.

בתחתית המסך יש שני כפתורים שעליהם כתוב "DIALOG" על כפתור אחד, ו-"OBSTACLE INFORMATION" על הכפתור השני. הכפתורים פותחים את הדיאלוגים של רמת הקושי ושל המידע על המכשולים בהתאם.

במסך יש גם כפתור חזרה למסך התפריט הראשי.

3.1.10. מסך המשחק – **GameScreenActivity**

מסך זה הוא המסך ששולט במשחק עצמו.

ברגע שהמסך מופעל יהיה כתוב במרכז שהשלב נטען, ובזמן זה האפליקציה משיגה את השיר של השלב מהפייירבייס (אם השחקן קבע שיהיה מוזיקה בהגדרות), אם האפליקציה קיבלה את השיר, או שלא אמור להיות מוזיקה במשחק, למסך יתווסף פאנל המשחק, שהוא המשחק עצמו. הפרטים של השלב נקבעים לפי מספר גורמים: מספר השלב, דרגת הקושי שהשחקן בחר, והפרטים שהשחקן בחר בחנות, שהם הדמות שלו, הרקע, תמונת הבנוס, והכוח שבחר.

מספר השלב משפיע על הרשימה של המכשולים שיופיעו בשלב, ואם השלב הוא רגיל או שלב בוס, ועל השיר שאמור להופיע בו. דרגת הקושי משפיעה על מספר הנקודות שהשחקן מרוויח כל שנייה בשלב רגיל, על מהירות המכשולים, ועל הזמן בין יצירת כל מכשול. הפרטים שהשחקן בחר מהחנות הם הפרטים שיופיעו בשלב.

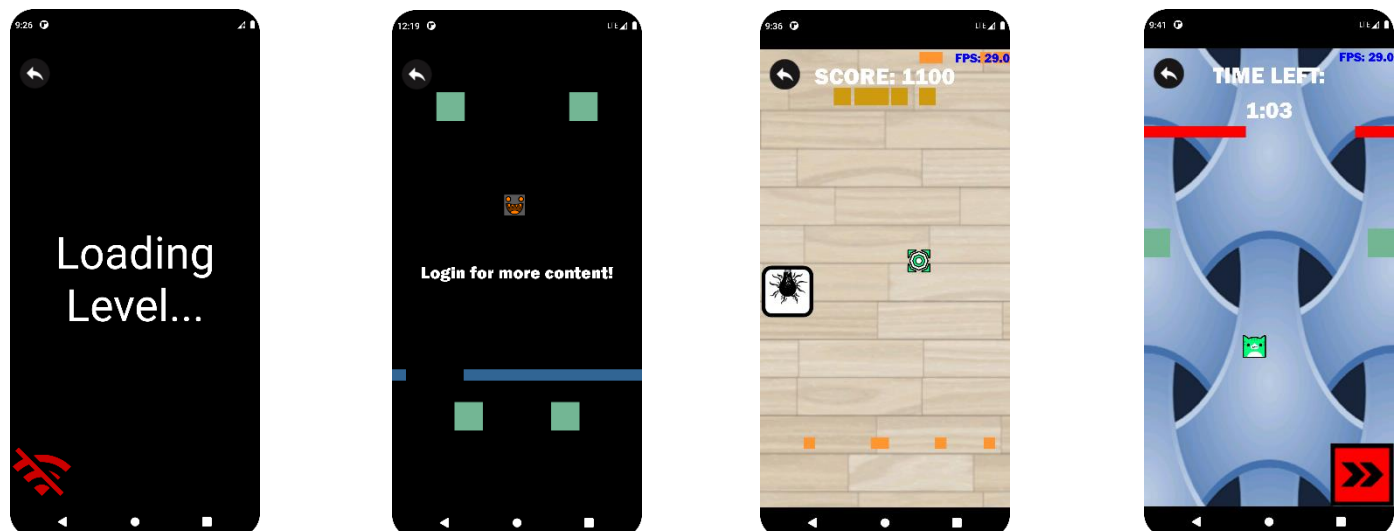
במסך עצמו יופיע טקסט שיגיד להתחבר למערכת בשביל עוד תוכן אם השחקן בשלב הדמו, את הניקוד שלו אם השלב הוא שלב רגיל, או את הזמן שנשאר אם הוא בשלב בוס.

בפינה העליונה מצד שמאל יש כפתור חזרה למסך בחירת השלב.

בפינה הימנית למעלה יש טקסט שמראה את כמות הפריימים לשנייה אם המשתמש בחר כך בהגדרות.

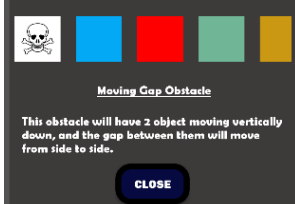
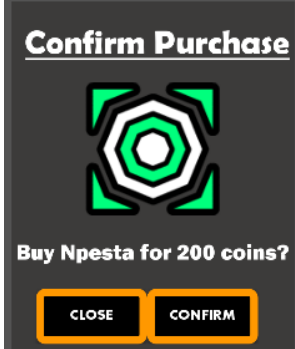
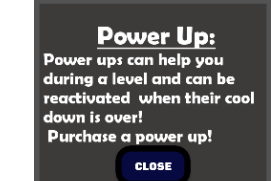
בפינה בצד ימין למטה יש את הכוח שבחר השחקן.

בפינה השמאלית למטה יש סימן של בעיות באינטרנט שיופיע אם לשחקן יש בעיות בחיבור לאינטרנט.

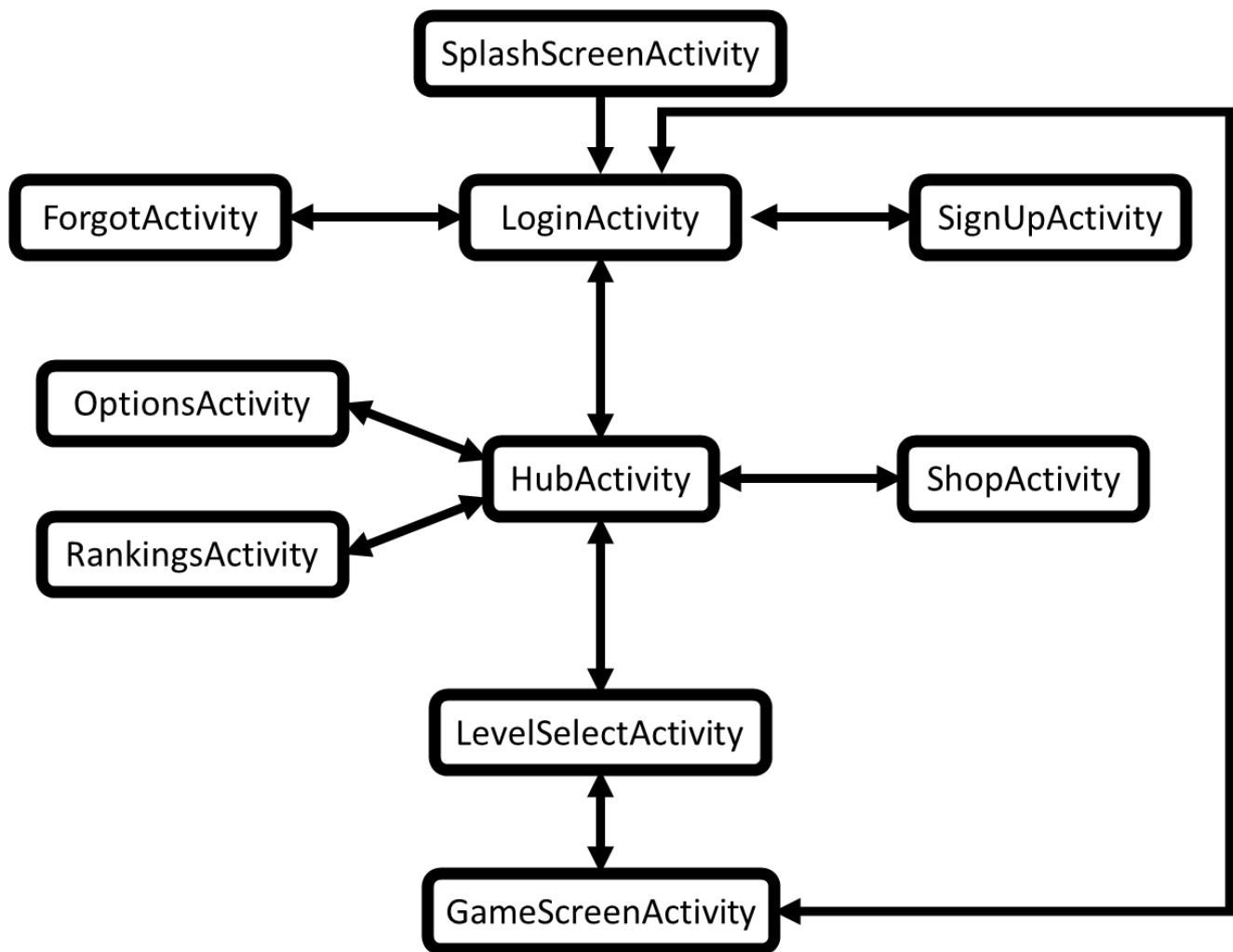


3.2. דיאלוגים

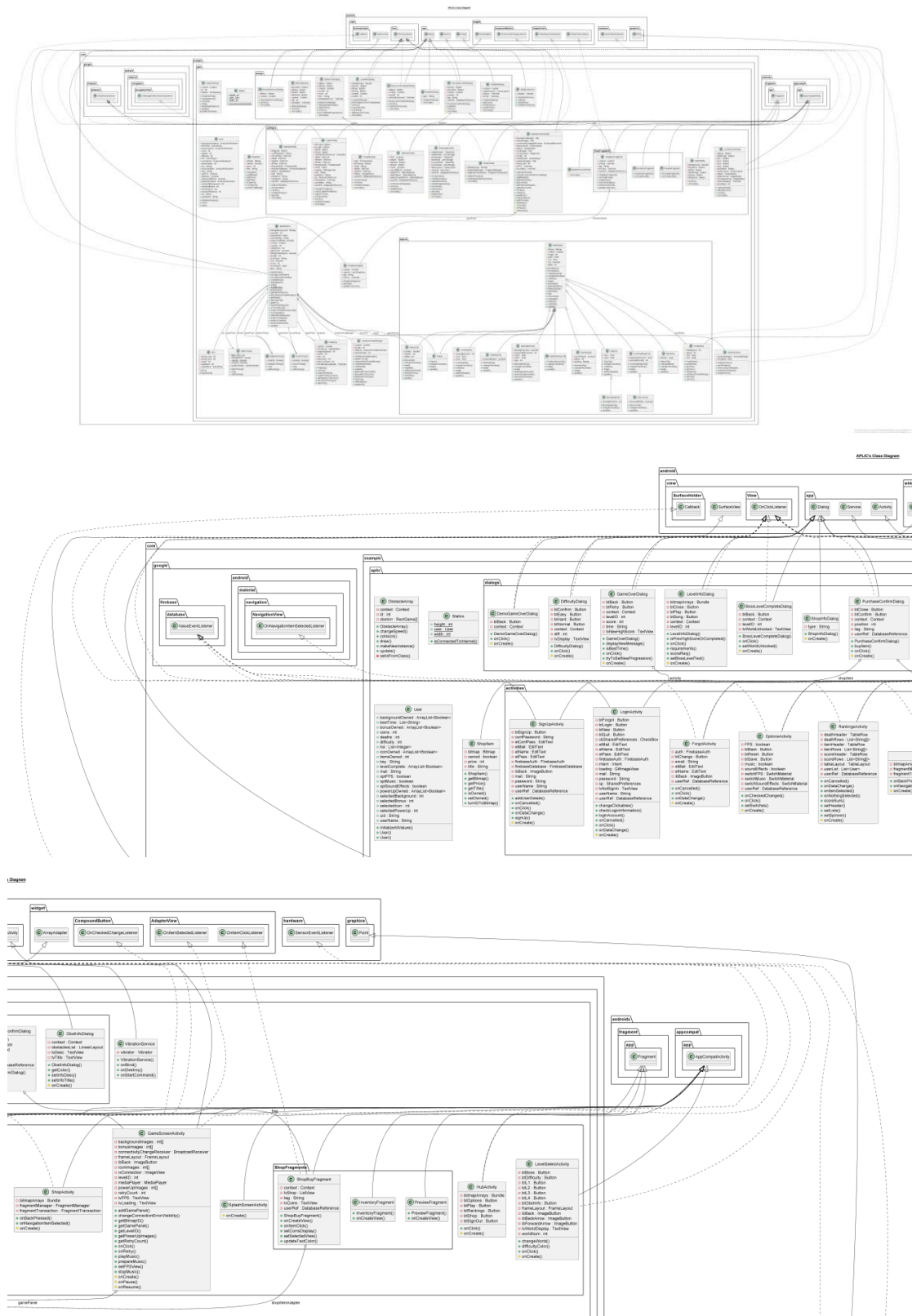
שם הדיאלוג	ה-Activity שאליו שייך	תיאור	דוגמא לתמונת הדיאלוג
BossLevelComplete	GameScreenActivity	מופיע כאשר השחקן מסיים שלב בוס (הטיימר הגיע ל0:00). הדיאלוג יודיע שהעולם הבא נפתח אם זו הפעם הראשונה שהוא סיים את השלב. על הדיאלוג יש כפתור חזרה למסך בחירת שלב.	
DemoGameOverDialog	GameScreenActivity	מופיע כאשר השחקן נפסל בשלב הדמו (השלב שמופיע כשהמשתמש בוחר לשחק מבלי להתחבר). על הדיאלוג יש כפתור שמחזיר את השחקן למסך ההתחברות.	
DifficultyDialog	LevelSelectActivity	מופיע כאשר לוחצים על הכפתור שעליו כתוב דרגת הקושי במסך. על הדיאלוג יש 3 כפתורים (קל, בינוני, קשה) כאשר לוחצים על כל אחד מהכפתורים מקבלים פירוט על דרגת הקושי. כאשר השחקן החליט איזה דרגת קושי הוא בחר הוא יכול ללחוץ על "אישור" ודרגת הקושי תשתנה בבסיס הנתונים.	
GameOverDialog	GameScreenActivity	מופיע כאשר השחקן נפסל, הדיאלוג מראה את כמות הנקודות שהשחקן השיג או כמה זמן נשאר לו לפי סוג השלב. על הדיאלוג יש כפתור חזרה למסך בחירת שלב או כפתור לנסות מחדש.	
LevelInfoDialog	LevelSelectActivity	מסביר לשחקן את הדרישות בשביל לפתוח את השלב. אם השלב פתוח, הדיאלוג יראה את כמות הנקודות הכי גבוהה שהשחקן הצליח להשיג באותו השלב, או את הזמן הכי נמוך שהצליח להשיג, לפי סוג השלב. בנוסף גם יהיה כפתור שיפתח באפליקציית יוטיוב את השיר של השלב. בדיאלוג יש כפתור לשחק שמעביר לאקטיביטי של מסך המשחק, וכפתור לסגור את הדיאלוג.	

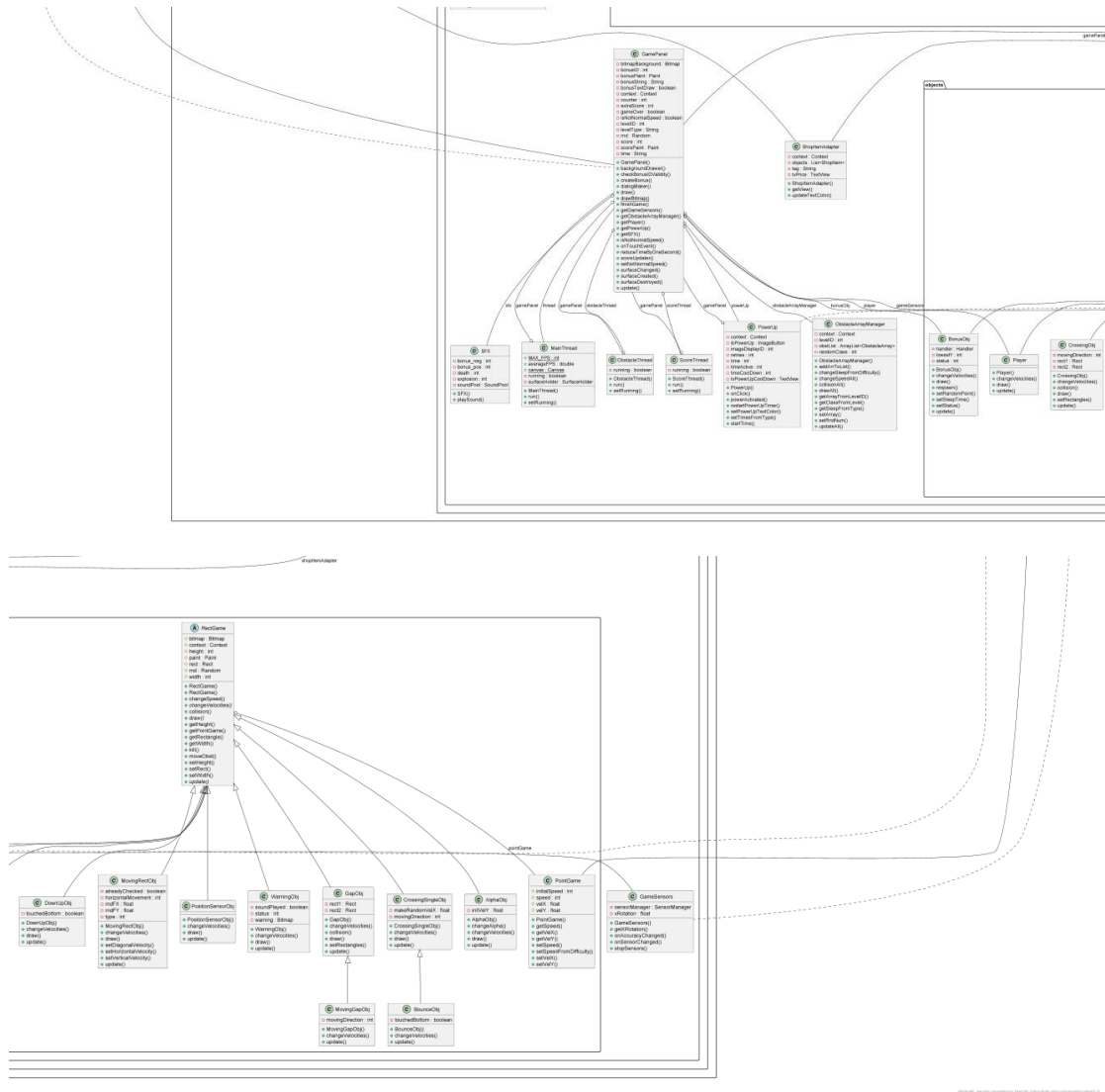
 <p>Moving Gap Obstacle</p> <p>This obstacle will have 2 object moving vertically down, and the gap between them will move from side to side.</p> <p>CLOSE</p>	<p>מופיע כאשר השחקן לוחץ על הכפתור של המידע על המכשולים. בדיאלוג יש תצוגת גלילה של כל המכשולים במשחק ולחיצה על כל אחד מהם תראה מידע על אותו המכשול. בדיאלוג יש גם כפתור סגירת דיאלוג.</p>	LevelSelectActivity	ObstInfoDialog
 <p>Confirm Purchase</p> <p>Buy Npesta for 200 coins?</p> <p>CLOSE CONFIRM</p>	<p>מופיע כאשר השחקן לוחץ על פריט שעדיין לא רכש. הדיאלוג יראה את תמונת הפריט, השם של הפריט והמחיר של הפריט. אם השחקן לוחץ על הכפתור של לקנות, הדיאלוג יבדוק אם יש לו מספיק מטבעות כדי לקנות, אם אין לשחקן מספיק יוצג לו Toast שיגיד לו את זה, ואם יש לו מספיק אז הוא יקנה את הפריט, יעדכן את כמות המטבעות שלו ויציג אותה מחדש למשתמש בפרגמנט החנות. המשתנה הבוליאני של האם הפריט נרכש יתעדכן בבסיס הנתונים.</p>	ShopBuyActivity	PurchaseConfirmDialog
<p>Are you sure This will reset all of your data!</p> <p>NO YES</p>	<p>מופיע כאשר השחקן לוחץ על כפתור אתחול הנתונים במסך ההגדרות. בדיאלוג, השחקן יכול ללחוץ על כן כדי לאתחל את כל הנתונים שלו מבסיס הנתונים, או ללחוץ על לא כדי לסגור את הדיאלוג. אם השחקן בוחר לאפס דרך הדיאלוג, יוצג לו הודעה שתגיד לו שהאיפוס הושלם.</p>	OptionsActivity	ResetDialog
 <p>Power Up:</p> <p>Power ups can help you during a level and can be reactivated when their cool down is over!</p> <p>Purchase a power up!</p> <p>CLOSE</p>	<p>מופיע בפרגמנט של הקנייה ונותן מידע לשחקן על כל פריט בהתאם לכל אחד מהפרגמנטים של הקנייה. בדיאלוג יש כפתור לסגור את הדיאלוג.</p>	ShopActivity	ShopInfoDialog

3.3. תרשים מסכים המתאר את היררכיית המסכים והמעברים ביניהם:



3.4. תיאור מחלקות הפרויקט - UML





4. מימוש הפרויקט

4.1. המחלקה SplashScreenActivity

המחלקה SplashScreenActivity היא המחלקה של האקטיביטי שהוא המסך הראשון שהשחקן רואה כשהוא פותח את האפליקציה. במסך זה המשתמש רואה אנימציה של לוגו האפליקציה במשך 3.3 שניות, ולאחר מכן, המשתמש מועבר למסך ההתחברות. המחלקה יורשת את המחלקה AppCompatActivity.

SplashScreenActivity – הפעולות	
הפעולה	תיאור הפעולה
@Override protected void onCreate(final Bundle savedInstanceState)	מציגה על האקטיביטי את הלוגו המתאים, משייכת את כל לוגו האפליקציה ללייאוט. יוצרת ומפעילה אנימציה לפי קובץ האנימציה R.anim.anim_splash. מתחילה האנדלר שיחכה 3.3 שניות ואז יעביר את השחקן למסך ההתחברות

4.2. המחלקה LoginActivity

המחלקה LoginActivity היא המחלקה של אקטיביטי, שממנו השחקן יכול להתחבר למערכת. השחקן מכניס את שם המשתמש, המייל, והסיסמא לתוך תיבות הטקסט שעל המסך, לוחץ על כפתור ההתחברות, ואם הנתונים מתאימים למשתמש במערכת, המשתמש יתחבר למערכת ויעבור למסך הראשי. השחקן יכול לשמור את נתונים אלו באמצעות תיבת סימון שתגרום לכך שהנתונים ישמרו דרך SharedPreferences. בנוסף, באקטיביטי יש כפתור יציאה מהאפליקציה, וכפתורים שיעבירו למסך השינוי סיסמא, ומסך יצירת משתמש חדש. המחלקה יורשת את המחלקה AppCompatActivity, ומיישמת את הממשקים View.OnClickListener, ValueEventListener.

LoginActivity – התכונות	
התכונה	תיאור התכונה
private CheckBox cbSharedPreferences	תיבת סימון שליחצה עליה ישמור את הנתונים של תיבות הטקסט במשתנה SharedPreferences
private SharedPreferences sp	אם תיבת הסימון מסומנת, משתנה זה יצור קובץ בטלפון שישמור את הנתונים שהשחקן כתב בתיבות הטקסט לאחר שהוא התחבר למערכת. בפעם הבאה שהשחקן נכנס לאפליקציה, נתונים אלו יוצגו באופן אוטומטי בתוך תיבות הטקסט המתאימות, ויאפשרו לשחקן להתחבר באופן מהיר למערכת
private Button btQuit, btLogin, btForgot, btNew	כפתור עבור סגירת האפליקציה, התחברות למערכת, מעבר למסך שינוי הסיסמא, ומעבר למסך יצירת משתמש חדש
private EditText etName, etMail, etPass	תיבת טקסט שבה השחקן יכול לכתוב את שם המשתמש שלו, המייל שלו, והסיסמא שלו
private TextView tvNotSignIn	טקסט לחץ שהשחקן יכול ללחוץ עליו כדי לשחק מבלי להתחבר למערכת
private String userName, mail, password	מחרוזת עבור שם המשתמש, מייל המשתמש, וסיסמת המשתמש

private GifImageView loading	גיפ טעינה שמוצג כאשר המשתמש מנסה להתחבר למערכת
private Intent intent	אינטנט שמעביר בין מסך זה למסכים אחרים
private FirebaseAuth firebaseAuth	רפרנס ל-Firebase Authentication
private DatabaseReference userRef	רפרנס ל-Firebase Database

LoginActivity – הפעולות	
הפעולה	תיאור הפעולה
@Override protected void onCreate(final Bundle savedInstanceState)	מציגה על האקטיביטי את הלייאווט המתאים, משייכת את תיבת הסימון, תיבות הטקסט, הטקסט והכפתורים ללייאווט וגורמת לכפתורים ולטקסט להיות לחיצים. משיגה את קובץ SharedPreferences מהאפליקציה, מאתחלת את הרפרנסים לפיירבייס, שמה על במחרוזות את הנתונים מהSharedPreferences. אם נתונים אלו הם לא null או ריקים, כלומר יש בהם תוכן של משתמש, והמשתמש בחר שהנתונים ישמרו. המחרוזות יוצגו על תיבות הטקסט ותיבת הסימון תהפוך למסומנת
@Override public void onClick(View view)	כאשר כפתור היציאה מהאפליקציה נלחץ, האקטיביטי נסגר. כפתור ההתחברות ירעיד את הטלפון ויבדוק אם המשתמש מחובר לאינטרנט. אם כן, יכניס את כל הנתונים של תיבות הטקסט למחרוזות, יבדוק שהנתונים תקינים באמצעות checkLoginInformation() הטעינה, יגרום לכפתורים לא להיות לחיצים, ויפעיל הרפרנס לדטאבייס את הפעולה addValueEventListener, שיגרום לפעולה של onDataChange להתחיל. אם השחקן לא מחובר לאינטרנט, תוצג לו הודעה בהתאם. לחיצה על כפתור איפוס הסיסמא או כפתור יצירת משתמש חדש יעבירו למסכים המתאימים. לחיצה על טקסט של לשחק מבלי להתחבר גם יבדוק אם השחקן מחובר לאינטרנט. אם כן, הפעולה תהפוך את המשתנה הסטטי של המשתמש למשתנה חדש, ויעביר את השחקן למסך המשחק שמספר השלב בו הוא 0. אם לא, הפעולה תציג לשחקן הודעה בהתאם
public boolean checkLoginInformation()	בודקת אם השדות שהשחקן הכניס מתאים, כלומר לא ריקים, ושהמייל תואם למבנה של כתובת מייל
public void loginAccount()	מנסה לחבר את המשתמש לפיירבייס דרך המייל והסיסמא, אם היא מצליחה, היא תעדכן/תאפס את הנתונים של SharedPreferences, לפי תיבת הסימון, תציג לשחקן הודעה שההתחברות הושלמה

	ותעביר אותו למסך הראשי. אם היא לא מצליחה, הפעולה תציג לשחקן הודעה בהתאם
Override@ public void onDataChange	פעולת חובה של הממשק ValueEventListener. הפעולה מקבלת את הנתונים של הפירבייס דטאבייס כפרמטר. הפעולה מורידה את ValueEventListener מהפרנס לדטאבייס, הפעולה בודקת אם קיים משתמש שהמייל ושם המשתמש שלו תואמים לאלו שבתיבות הטקסט, אם כן, משתנה המשתמש הסטטי יקבל את הנתונים של אותו המשתמש, והפעולה תקרא לפעולה loginAccount(), אם לא, הפעולה תציג לשחקן הודעה בהתאם, בשני המקרים הפעולה תגרום לכפתורים להיות שוב לחיצים, ותעלים את גיף הטעינה
Override@ public void onCancelled	פעולת חובה של הממשק ValueEventListener
public void changeClickables(boolean b)	מסנה את היכולת של המשתמש ללחוץ על הכפתורים לפי הבוליאן שהיא מקבלת כפרמטר, משמש על מנת שהשחקן לא יוכל לעבור מסכים, או לבדוק את המשתמש בזמן שהאפליקציה בתהליך של בדיקת נתוני משתמש

4.2.1. המחלקה User

המחלקה User שומרת את פרטי המשתמש שמשומשים לאורך האפליקציה, תכונות המחלקה מקושרות לבסיס הנתונים Firebase דרך Firebase Authentication ו־Firebase Database.

User – התכונות	
התכונה	תיאור התכונה
public String key	המפתח שהמשתמש מקבל מ־Firebase Database
public String uid	המפתח שהמשתמש מקבל מ־Firebase Authentication
public String userName, mail	השם והמייל של המשתמש
public int deaths	כמות הפעמים שהמשתמש מת
public int coins	כמות המטבעות שיש למשתמש
public int difficulty	רמת הקושי שהמשתמש בחר
public boolean optMusic, optSoundEffects, optFPS	משתנים עבור הגדרות המשחק, המשתנים מייצגים מוזיקה, אפקטי שמע, והצגת FPS
public List<Integer> hsl	רשימה עבור הניקוד של המשתמש בכל השלבים הרגילים של האפליקציה
public List<String> bestTime	רשימה עבור הזמן הכי טוב של השחקן בכל שלב בוס
public ArrayList<Boolean> levelComplete	רשימה עבור השלמת שלב בכל שלב בוס
public ArrayList<Boolean> iconOwned, backgroundOwned, bonusOwned, powerUpOwned	רשימה עבור כל סוג פריט שבדוק האם המשתמש שרכש את הפריט

public int selectedIcon, selectedBackground, selectedBonus, selectedPowerUp	האינדקס של הפריט שהמשתמש בחר עבור כל סוג פריט
public int itemsOwned	סך הפריטים שהמשתמש רכש

User – הפעולות	
הפעולה	תיאור הפעולה
public User()	יוצרת משתמש חדש עבור שלב דמו
public User(String uid, String mail, String userName, String key)	יוצרת משתמש חדש
public void InitializeAllValues()	מאתחלת את כל נתוני המשתמש

4.2.2. Statics המחלקה

המחלקה Statics שומרת משתנים סטטים ופעולה סטטית שמשומשים במקומות רבים באפליקציה.

Statics – התכונות	
התכונה	תיאור התכונה
public static int width	רוחב המסך
public static int height	אורך המסך
public static User user	המשתמש שמחובר לאפליקציה

Statics – הפעולות	
הפעולה	תיאור הפעולה
public static boolean isConnectedToInternet(Context context)	פעולה שבודקת האם המשתמש מחובר לאינטרנט ומחזירה אמת או שקר בהתאם

4.2.3. VibrationService המחלקה

המחלקה VibrationService גורמת לכך שהטלפון ירעד. המחלקה יורשת מהמחלקה Service.

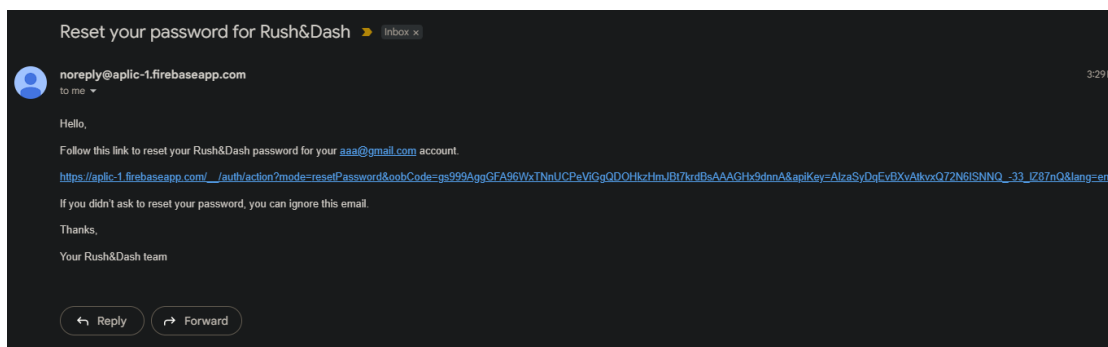
VibrationService – התכונות	
התכונה	תיאור התכונה
private Vibrator vibrator	גורם להרעדת הטלפון

VibrationService – הפעולות	
הפעולה	תיאור הפעולה
public VibrationService()	פעולה בונה של המחלקה
@Override public IBinder onBind(Intent intent)	פעולת חובה של המחלקה Service
@Override public int onStartCommand(Intent intent, int flags, int startId)	גורמת לטלפון לרעוד
@Override public void onDestroy()	מבטלת את הרטט

4.3. המחלקה ForgotActivity

המחלקה ForgotActivity היא המחלקה של האקטיביטי שהשחקן יכול לשנות את סיסמתו דרכה. השחקן מכניס לתיבות הטקסט את המייל ושם המשתמש שלו, ולאחר מכן, נשלח אליו מייל שדרכו הוא יכול לשנות את סיסמתו. המחלקה יורשת את המחלקה AppCompatActivity, ומיישמת את הממשקים ValueEventListener, View.OnClickListener.

דוגמא למייל שישלח למשתמש לאחר איפוס סיסמתו:



המשתמש יכול ללחוץ על הקישור ולהכנס לאתר שממנו הוא יכול לשנות את סיסמתו:

ForgotActivity – התכונות	
התכונה	תיאור התכונה
private EditText etName, etMail	תיבת טקסט עבור שם המשתמש, ומייל המשתמש
private String email	מחרוזת עבור המייל
private Button btChange	כפתור שינוי סיסמא
private ImageButton ibBack	כפתור חזרה למסך ההתחברות
private FirebaseAuth auth	פרנס ל-Firebase Authentication
private DatabaseReference userRef	פרנס ל-Firebase Database

ForgotActivity – הפעולות	
הפעולה	תיאור הפעולה
@Override protected void onCreate(final Bundle savedInstanceState)	מציגה על האקטיביטי את הלייאוט המתאים, מאתחלת את הרפרנסים לפיירבייס, משייכת את תיבות הטקסט והכפתורים ללייאוט, גורמת לכפתורים להיות לחיצים
@Override public void onClick(View view)	כאשר לוחצים על כפתור שינוי הסיסמא, הטלפון ירעד, ועל הרפרנס לדטאבייס יופעל הפעולה addValueEventListener, שיגרום לפעולה של onDataChange להתחיל

<pre>@Override public void onDataChange(@NonNull DataSnapshot snapshot)</pre>	פעולת חובה של הממשק ValueEventListener. הפעולה מקבלת את הנתונים של הפירבייס דטאבייס כפרמטר. הפעולה מורידה את ה ValueEventListener מהרפרנס לדטאבייס, שומרת את הנתונים של השם והמייל מתיבות הטקסט, ולאחר מכן, בודקת אם בדטאבייס יש שם שמתאים למייל. אם כן, הפעולה תשלח מייל שינוי סיסמא למייל, אם לא, הפעולה תציג למשתמש הודעה בהתאם
<pre>@Override public void onCancelled(@NonNull DatabaseError error)</pre>	פעולת חובה של הממשק ValueEventListener

4.4. המחלקה SignUpActivity

המחלקה SignUpActivity היא המחלקה של האקטיביטי שדרכה המשתמש יכול ליצור חשבון כדי להתחבר למערכת. במסך יש ארבע תיבות טקסט, שם משתמש, מייל, סיסמא, אישור סיסמא. לאחר שהמשתמש מילא את ארבע השדות, הוא יוכל לנסות להרשם למערכת באמצעות כפתור ההרשמה. המערכת תקבל את המשתמש רק אם השם והמייל הם יחודיים, ואין עוד משתמש עם אותו שם או מייל. באקטיביטי יש גם כפתור חזרה למסך ההתחברות. המחלקה יורשת את המחלקה AppCompatActivity, ומיישמת את הממשקים View.OnClickListener, ValueEventListener.

SignUpActivity – התכונות	
התכונה	תיאור התכונה
private EditText etName, etMail, etPass, etConfPass	תיבות טקסט עבור שם המשתמש, המייל, סיסמא, ואישור הסיסמא
private String userName, mail, password, confPassword	מחרוזות עבור שם המשתמש, המייל, סיסמא, ואישור הסיסמא
private ImageButton ibBack	כפתור חזרה למסך ההתחברות
private Button btSignUp	כפתור יצירת משתמש חדש
private FirebaseAuth firebaseAuth	רפרנס ל-Firebase Authentication
private FirebaseDatabase firebaseDatabase	רפרנס ל-Firebase Database
private DatabaseReference userRef	רפרנס לדטאבייס שנועד כדי ליצור משתמש חדש

SignUpActivity – הפעולות	
הפעולה	תיאור הפעולה
<pre>@Override protected void onCreate(final Bundle savedInstanceState)</pre>	מציגה על האקטיביטי את הלייאוט המתאים, משייכת את כל תיבות הטקסט והכפתורים ללייאוט, גורמת לכפתורים להיות לחיצים. מאתחלת את הרפרנסים לפירבייס
<pre>@Override public void onClick(View view)</pre>	כאשר כפתור החזרה נלחץ, האקטיביטי נסגר. כאשר כפתור ההירשמות נלחץ, אם למשתמש יש אינטרנט, הטלפון ירעד הפעולה תשים את הנתונים שבתיבות הטקסט במחרוזות, ותקרא לפעולת ההתחברות, אם למשתמש אין אינטרנט, תוצג למשתמש הודעה בהתאם

<code>public void signUp()</code>	בודקת שתיבות הטקסט לא ריקות, שהמייל כתוב במבנה של מייל, שתיבת הטקסט של הסיסמא ואישור הסיסמא תואמות, אם אחד מהתנאים לא מתקיימים, יוצג למשתמש הודעה בהתאם. אם כל התנאים מתקיימים, יופעל על הרפרנס לדטאבייס הפעולה <code>addValueEventListener</code> , שיגרום לפעולה של <code>onDataChange</code> להתחיל.
<code>public void addUserDetails()</code>	יוצרת משתמש לפי השם, המייל <code>uid</code> שהיא מקבלת מ <code>Firebase Authentication</code> , והיא מקבלת <code>key</code> שהיא מקבלת מ <code>FirebaseDatabase</code> .
<code>@Override</code> <code>public void onDataChange(@NonNull DataSnapshot snapshot)</code>	פעולת חובה של הממשק <code>ValueEventListener</code> . הפעולה מקבלת את הנתונים של הפיירבייס דטאבייס כפרמטר. הפעולה מורידה את הנתונים מ <code>EventListeners</code> מהרפרנס לדטאבייס, בודקת שאין משתמש בדטא בייס אם אותו שם או מייל שזהה למחרוזות המייל והשם, אם יש התאמה, תוצג לשחקן הודעה בהתאם והחשבון לא ייוצר. אם הנתונים התקבלו, הפעולה תיצור חשבון בפיירבייס באמצעות המייל והסיסמא, ותקרא לפעולה <code>addUserDetails()</code> .
<code>@Override</code> <code>public void onCancelled(@NonNull DatabaseError error)</code>	פעולת חובה של הממשק <code>ValueEventListener</code> .

4.5. המחלקה **HubActivity**

המחלקה **HubActivity** היא המחלקה של האקטיביטי הראשי של האפליקציה, שממנו המשתמש יכול ללחוץ על כפתורים כדי לעבור לחלקים השונים של האפליקציה כמו מסך בחירת השלב, החנות, האפשרויות, טבלת השיאים, וחזרה למסך ההתחברות. בנוסף, בפינת המסך כתוב את שם המשתמש של המשתמש שהתחבר למערכת. המחלקה יורשת את המחלקה `AppCompatActivity`, ומיישמת את הממשק `View.OnClickListener`.

HubActivity – התכונות	
התכונה	תיאור התכונה
<code>private Button btPlay, btOptions, btShop, btSignOut, btRankings</code>	כפתורים למסך בחירת שלב, הגדרות, חנות, התנתקות מהמערכת, טבלת שיאים
<code>private Bundle bitmapArrays</code>	חבילה של מערכים מסוג של מספרים שלמים ששומרת את התמונות של אייקון השחקן, הרקע, הבונוס, והכח

HubActivity – הפעולות	
הפעולה	תיאור הפעולה
<code>@Override</code> <code>protected void onCreate(final Bundle savedInstanceState)</code>	מציגה על האקטיביטי את הלייאוט המתאים, משייכת את הכפתורים ללייאוט וגורמת להם להיות לחיצים. מכינה את מערכי התמונות ומכניסה אותם לחבילה.

	משיכת טקסט וויו ללייאוט ומציגה בו את שם השחקן
@Override public void onClick(View view)	לחיצה על כל כפתור תגרום לרעידת הטלפון. הכפתור PLAY יעביר למסך בחירת השלב, כפתור OPTIONS יעביר למסך האפשרויות, כפתור SHOP יעביר למסך החנות, כפתור SIGN OUT יעביר למסך ההתחברות, כפתור RANKINGS יעביר למסך טבלת השיאים. כפתור המשחק והחנות יוסיפו אליהם את חבילת התמונות. כפתור ההתנתקות ינתק את המשתמש מפיירבייס, ויסיים את האקטיביטי, כפתור טבלת השיאים יבדוק את החיבור לאינטרנט, ויציג הודעה מתאימה אם המשתמש לא מחובר לאינטרנט

4.6. המחלקה OptionsActivity

המחלקה OptionsActivity היא המחלקה של האקטיביטי שממנו השחקן יכול לשנות את הגדרות המשחק. ההגדרות הם מוזיקת רקע במהלך המשחק, סאונד אפקטים במהלך המשחק, הצגת כמות הפריימים לשנייה. השחקן שולט על אפשרויות אלו דרך מתגים שהוא יכול להדליק ולכבות. במסך יש כפתור שמירה שתשמור את הנתונים של אותם המתגים. במסך יש גם כפתור אתחול שהשחקן יכול לאפס דרכו את כל הנתונים שלו, וכפתור חזרה למסך הראשי. המחלקה יורשת את המחלקה AppCompatActivity, ומיישמת את הממשקים View.OnClickListener, CompoundButton.OnCheckedChangeListener.

OptionsActivity – התכונות	
התכונה	תיאור התכונה
private DatabaseReference userRef	רפרנס ל-Firebase Database
private SwitchMaterial switchSoundLEffects, switchMusic, switchFPS	מתגים עבור הגדרות המוזיקה, האפקטים, וה-FPS
private Button btSave, btBack, btReset	כפתור שמירת שינויים, כפתור חזרה למסך הראשי, וכפתור אתחול הנתונים
private boolean music, soundEffects, FPS	בוליאנים עבור הגדרות המוזיקה, האפקטים, וה-FPS

OptionsActivity – הפעולות	
הפעולה	תיאור הפעולה
@Override protected void onCreate(final Bundle savedInstanceState)	מציגה על האקטיביטי את הלייאוט המתאים, משיכת את הכפתורים והמתגים ללייאוט וגורמת להם להיות לחיצים. מאתחלת את הרפרנס לדטאבייס. מכינה את המתגים לפי הנתונים שבדטאבייס
public void setSwitches()	משנה את המצב המתגים לפי הגדרות המשתמש
public void onCheckedChanged(CompoundButton compoundButton, boolean b)	משנה את בוליאן ההגדרות כל פעם שמפסק משתנה
@Override public void onClick(View view)	לחיצה על כפתור החזרה תסגור את האקטיביטי. כפתורי השמירה והאתחול

	דורשים חיבור לאינטרנט ויציגו לשחקן הודעה אם הוא לא מחובר. כפתור השמירה ישמור את הנתונים שבמתגים במשתנה המשתמש ובדטאבייס, ירעיד את הטלפון, ויציג לשחקן הודעה שהנתונים נשמרו. שבו Alert Dialog כפתור האתחול יפתח השחקן יוכל לבחור אם הוא רוצה לאתחל את כל המשתמש או שלא
--	---

4.7. המחלקה RankingsActivity

המחלקה RankingsActivity היא המחלקה של האקטיביטי של טבלת השיאים של המשחק. יש במשחק שלושה סוגים של טבלאות שיאים: סך נקודות, סך הפעמים שהשחקן נפסל, סך הפריטים שהשחקן קנה. הטבלה מציגה את עשרת המובילים בכל קטגוריה. המשתמש יכול להחליף בין טבלאות השיאים באמצעות Spinner. המחלקה יורשת את המחלקה AppCompatActivity, ומיישמת את הממשקים AdapterView.OnItemClickListener, ValueEventListener.

RankingsActivity – התכונות	
תיאור התכונה	התכונה
רפרנס ל-Firebase Database	private DatabaseReference userRef
רשימה של כל המשתמשים	private List<User> userList
שורה הכותרת של שלושת טבלאות השיאים	private TableRow scoreHeader, deathHeader, itemHeader
רשימות של שלושת טבלאות השיאים	private List<String[]> deathRows, scoreRows, itemRows
הטבלה שעליה מצוירת טבלת השיאים	private TableLayout tableLayout

RankingsActivity – הפעולות	
תיאור הפעולה	הפעולה
מציגה על האקטיביטי את הלייאוט המתאים, מאתחלת את הרפרנס לדטאבייס ומפעילה עליו את הפעולה addValueEventListener, שיגרום לפעולה של onDataChange להתחיל. הפעולה מאתחלת את הרשימות וטבלאות. יוצרת את כפתור החזרה ומשייכת אותו ללייאוט, גורמת לכך שלחיצה עליו תסגור את המסך. משייכת את הטבלה ללייאוט. יוצרת את הספינר ואת הכותרות לשלושת הטבלאות באמצעות הפעולות setSpinner(), setHeader()	@Override protected void onCreate(final Bundle savedInstanceState)
משייכת את הספינר ללייאוט, מוסיפה שורה של טבלה ריקה ושלוש שורות לשלושת הטבלאות אל אדפטר, ושמה על הספינר את אדפטר זה, קוראת לפעולה שגורמת לכל פריט בספינר להיות לחיצ	public void setSpinner()
הפעולה יוצרת שורת כותרת בטבלה עבור כל אחת משלושת הטבלאות	public void setHeader()
הפעולה מקבלת משתמש כפרמטר, הפעולה מחזירה את סך הנקודות של השלב בכל שלבי המשחק	public int scoreSum(User tempUser)

public void setLists()	יוצר רשימה של עשרת המובילים בכל קטגוריה
@Override public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l)	הפעולה גורמת לכך שכל פעם שנלחץ אחת מהשורות בספינר, הטבלה מתאפסת ועליה מצוירת טבלה חדשה בהתאם לאפשרות שנלחצה, או שלא יצויר עליה כלום אם האופציה של טבלה ריקה נבחרת
@Override public void onNothingSelected(AdapterView<?> adapterView)	פעולת חובה של הממשק AdapterView.OnItemSelectedListener
@Override public void onDataChange(@NonNull DataSnapshot snapshot)	פעולת חובה של הממשק ValueEventListener. הפעולה מקבלת את הנתונים של הפירבייס דטאבייס כפרמטר. הפעולה מורידה את הנתונים מהרפרנס לדטאבייס, יוצרת רשימה של כל המשתמשים בדטאבייס, וקוראת לפעולה setLists(), כדי להפוך את רשימה זו לרשימות הרצויות
@Override public void onCancelled(@NonNull DatabaseError error)	פעולת חובה של הממשק ValueEventListener

4.8. המחלקה ShopActivity

המחלקה ShopActivity היא מחלקה שממנה השחקן יכול לגשת לפרגמנטים שמייצגים חלקים שונים של חנות המשחק. המסך משתמש ב, Toolbar, DrawerLayout, ActionBarDrawerToggle, NavigationView כדי לתת למשתמש את האפשרות לבחור מגירה, ללחוץ על אחת מאפשרויות שלה, ולהחליף את הפרגמנט שכרגע מוצג על המסך בפרגמנט החדש. המסך נפתח, הוא יציג את פרגמנט התצוגה המקדימה לשחקן, ולאחר מכן השחקן יוכל להחליף את הפרגמנט באמצעות המגירה. המחלקה יורשת את המחלקה AppCompatActivity, ומיישמת את הממשק
NavigationView.OnNavigationItemSelectedListener.

ShopActivity – התכונות	
התכונה	תיאור התכונה
private FragmentManager fragmentManager	משומש כדי לנהל את הפרגמנטים של האקטיביטי
private FragmentTransaction fragmentTransaction	משומש כדי לחליף בין הפרגמנטים שעל המסך
private Bundle bitmapArrays	חבילה של מערכים מסוג של מספרים שלמים ששומרת את התמונות של אייקון השחקן, הרקע, הבונוס, והכח

ShopActivity – הפעולות	
הפעולה	תיאור הפעולה
@Override protected void onCreate(final Bundle savedInstanceState)	מציגה על האקטיביטי את הלייאווט המתאים, מאתחלת את ה, Toolbar, DrawerLayout, ActionBarDrawerToggle, מאתחלת את המסך כדי שיוציג את פרגמנט התצוגה המקדימה.

Override@ public void onBackPressed()	גורמת לכך שכאשר המגירה פתוחה וכפתור החזרה נסגר, המגירה תיסגר במקום שהאקטיביטי יסגר.
Override@ public boolean onNavigationItemSelectedListener(Menuitem item)	מחליפה את הפרגמנט הנוכחי בפרגמנט שהשחקן לחץ עליו, מעבירה לפרגמנט החדש את חבילת התמונות

4.8.1. המחלקה PreviewFragment

המחלקה PreviewFragment היא המחלקה של הפרגמנט שמראה לשחקן את הפריטים שהוא בחר. הפרגמנט בנוי מארבע תמונות שמייצגות את ארבעת סוגי הפריטים, אייקון, רקע, בונס, כח. אם למשתמש אין שום כח, יהיה כתוב במקום של הכח שלמשתמש אין כח. הפרגמנט כולל כפתור חזרה למסך הראשי. המחלקה יורשת את המחלקה Fragment.

InventoryFragment – הפעולות	
הפעולה	תיאור הפעולה
public InventoryFragment()	פעולה בונה לפרגמנט
@Override public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)	יוצרת משתנה מסוג View שמנפח את הלייאוט המתאים לפרגמנט. הפעולה יוצרת את כפתור החזרה שמחזיר את המשתמש למסך הראשי. הפעולה משייכת את ארבע התמונות ללייאוט, משיגה את ארבעת מערכי התמונות מהחבילה שהיא מקבלת מהאקטיביטי שאחראי עליה, ומציגה על כל תמונה את הפריט שהשחקן בחר. לבסוף, הפעולה מחזירה את עצם ה View כדי להציג את הפרגמנט על המסך

4.8.2. המחלקה InventoryFragment

המחלקה InventoryFragment היא המחלקה של הפרגמנט שמראה לשחקן את כל הפריטים שבבעלותו. הפרגמנט בנוי מארבע תצוגות גלילה אופקיות שמייצגות את ארבעת סוגי הפריטים, אייקון, רקע, בונס, כח. הפרגמנט כולל כפתור חזרה למסך הראשי. המחלקה יורשת את המחלקה Fragment.

InventoryFragment – הפעולות	
הפעולה	תיאור הפעולה
public InventoryFragment()	פעולה בונה לפרגמנט
@Override public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)	יוצרת משתנה מסוג View שמנפח את הלייאוט המתאים לפרגמנט. הפעולה יוצרת את כפתור החזרה שמחזיר את המשתמש למסך הראשי. הפעולה יוצרת 4 עצמים מסוג LinearLayout ומשייכת אותם ללייאוט, משיגה את ארבעת מערכי התמונות מהחבילה שהיא מקבלת מהאקטיביטי שאחראי עליה, ומציגה על כל LinearLayout את התמונות של הפריטים שבעלות המשתמש. לבסוף, הפעולה מחזירה את עצם ה View כדי להציג את הפרגמנט על המסך

4.8.3. ShopBuyFragment המחלקה

המחלקה ShopBuyFragment היא המחלקה של הפרגמנט שהוא החנות שדרכה השחקן יכול לקנות פריטים. יש ארבעה סוגים של חנויות, שהשחקן מגיע לכל אחת מהן דרך ארבעת הכפתורים במגרה של האקטיביטי שמציינים אייקון, רקע, בונס, כח. הפרגמנט מורכב מתצוגת גלילה ושם נמצאים כל הפריטים בחנות מאותו הסוג. תצוגת הגלילה משתמשת באדפטר ShopItemAdapter כדי להציג בכל שורה של תצוגת הגלילה את שם הפריט, תמונת הפריט, מחיר הפריט, שיהיה בצבע אדום אם הוא לא נרכש, וירוק אם הוא כן, וסימן אישור שיהיה ליד הפריט שהמשתמש בחר להשתמש בו. בנוסף, הפרגמנט כולל כפתור שמציג מידע למשתמש על סוג הפריט, וכפתור חזרה למסך הראשי. המחלקה יורשת את המחלקה Fragment ומיישמת את הממשק AdapterView.OnItemClickListener.

ShopBuyFragment – התכונות	
תיאור התכונה	התכונה
הקונטקסט של הפרגמנט	private final Context context
רפרנס ל-Firebase Database	private final DatabaseReference userRef
הטאג ששומר על סוג החנות	private final String tag
הטקסט שכתוב בו מספר המטבעות של השחקן	private TextView tvCoins
רשימת הפריטים שמוצגת על המסך	private ListView lvShop
האדפטר של רשימת הפריטים	private ShopItemAdapter shopItemAdapter

ShopBuyFragment – הפעולות	
תיאור הפעולה	הפעולה
פעולה בונה של הפרגמנט, מאתחלת את הקונטקסט לפי הפרמטר. מאתחלת את הטאג לפי הטאג שהפרגמנט קיבל מהאקטיביטי. מאתחלת את הרפרנס לדטאבייס.	public ShopBuyFragment(Context context)
יוצרת משתנה מסוג View שמנפח את הלייאוט המתאים לפרגמנט. הפעולה יוצרת את כפתור החזרה שמחזיר את המשתמש למסך הראשי. יוצרת את כפתור החזרה למסך הראשי, וכפתור הצגת דיאלוג המידע על החנות, משייכת אותם ללייאוט, וגורמת לכך שכאשר ילחצו עליהם הם יבצעו את תפקידן. משייכת את הטקסט שעליו כתוב את מספר המטבעות של השחקן ללייאוט ומאתחלת אותו. יוצרת את רשימת הפריטים לפי הטאג, ומוסיפה אליו את כל הפריטים. הפעולה משייכת את רשימת הפריטים ללייאוט, ובאמצעות האדפטר, מציגה אותם על ה-View, גורמת לכל הפריטים להיות לחיצים. לבסוף, הפעולה מחזירה את עצם ה-View כדי להציג את הפרגמנט על המסך	@Override public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
מעדכנת את הטקסט שמראה את מספר המטבעות של השחקן על המסך	public void setCoinsDisplay()
אם למשתמש יש אינטרנט והוא לוחץ על פריט שבבעלותו, הפעולה תקרא לפעולה setSelectedView כדי לעדכן את הפריט שבחר המשתמש. אם המשתמש לוחץ על	public void onItemClick(AdapterView<?> adapterView, View view, int i, long l)

	פריט שלא בבעלותו, יפתח דיאלוג קניית הפריט. אם למשתמש אין אינטרנט יוצג לו הודעה בהתאם
<code>public void setSelectedView(int i)</code>	משנה את מספר הפריט שהשחקן בחר מעדכנת אותו בפריירבייס
<code>public void updateTextColor(int position)</code>	מעדכנת את האדפטר כדי לשנות את צבע המחיר של פריט לירוק כאשר קונים אותו

4.8.4. המחלקה **PurchaseConfirmDialog**:

המחלקה **PurchaseConfirmDialog** היא מחלקה של הדיאלוג שהשחקן יכול לפתוח דרך מסך בחירת השלב. בדיאלוג זה, השחקן יכול לקרוא על כל אובייקט כדי לדעת איך לפעול נגדו. על הדיאלוג מופיע `horizontal scroll view` שהשחקן יכול לגלול דרכו וללחוץ כל אחד מהאובייקטים כדי לשנות את הטקסט שמופיע מתחת לדיאלוג. הדיאלוג כולל כפתור סגירת דיאלוג. המחלקה יורשת את המחלקה `Dialog`, ומיישמת את הממשק `View.OnClickListener`.

PurchaseConfirmDialog – התכונות	
התכונה	תיאור התכונה
<code>private final Context context</code>	הקונטקסט של הדיאלוג
<code>private final ShopBuyFragment frag</code>	הפרגמנט שפתח את הדיאלוג
<code>private final int position</code>	המיקום של הפריט ברשימה
<code>private final ShopItem shopItem</code>	הפריט שהשחקן קנה
<code>private final String tag</code>	הטאג שמייצג את סוג החנות
<code>private Button btConfirm, btClose</code>	כפתור של אישור קניית הפריט, וכפתור סגירת הדיאלוג
<code>private final DatabaseReference userRef</code>	רפרנס לדטאבייס

PurchaseConfirmDialog – הפעולות	
הפעולה	תיאור הפעולה
<code>public PurchaseConfirmDialog(Context context, ShopBuyFragment frag, ShopItem shopItem, int position, String tag)</code>	פעולה שמאתחלת את הדיאלוג ואת משתנה הקונטקסט, את משתנה הפרגמנט, את משתנה הפריט, את משתנה מיקום הפריט ברשימה, את משתנה הטאג, ומאתחלת את הרפרנס לדטאבייס לפריירבייס דטאבייס
<code>@Override protected void onCreate(final Bundle savedInstanceState)</code>	מציגה על הדיאלוג את הלייאוט המתאים, גורמת לכך שלא יהיה אפשר ללחוץ מחוץ לדיאלוג כדי להעלים אותו, משייכת את הטקסט, התמונה, והכפתורים ללייאוט, גורמת לכפתורים להיות לחיצים, משנה את תמונת הפריט לתמונה המתאימה, ומציגה בטקסט את שם הפריט והמחיר שלו
<code>public void buyItem()</code>	קונה את הפריט המתאים, מעדכנת את הדטאבייס בהתאם, משנה את הטקסט של מחיר הפריט לירוק בפרגמנט, מפחיתה את מטבעות המשתמש לפי מחיר הפריט ומציגה את מספר המטבעות החדש בפרגמנט

@Override public void onClick(View view)	אם השחקן לוחץ על כפתור הקנייה, האפליקציה תבדוק אם יש לשחקן מספיק מטבעות, אם כן, האפליקציה תקנה את הפריט, אם לא תוצג לשחקן הודעה מתאימה. אם השחקן לוחץ על כפתור הסגירה, הדיאלוג יסגר
---	---

4.8.5. המחלקה ShopInfoDialog

המחלקה ShopInfoDialog היא מחלקה של הדיאלוג שהשחקן יכול לפתוח דרך החנות כדי לראות מה החנות מציעה לשחקן. המחלקה יורשת את המחלקה Dialog, ומיישמת את הממשק View.OnClickListener.

ShopInfoDialog – התכונות	
התכונה	תיאור התכונה
private final String type	סוג הפריט של החנות

ShopInfoDialog – הפעולות	
הפעולה	תיאור הפעולה
public ShopInfoDialog(Context context, String type)	פעולה שמאתחלת את הדיאלוג ואת משתנה סוג החנות
@Override protected void onCreate(final Bundle savedInstanceState)	מציגה על הדיאלוג את הלייאוט המתאים, גורמת לכך שלא יהיה אפשר ללחוץ מחוץ לדיאלוג כדי להעלים אותו, מייצרת 2 טקסט וויו עבור כותרת ותיאור סוג הפריט, וכפתור סגירת דיאלוג ומשייכת אותם ללייאוט, ומשנה את המחרוזות שכתובות על הטקסט וויוויים

4.8.6. שימוש באדפטר באפליקציה על מנת ליצור חנות

ב-Android Studio, מחלקת האדפטר משומשת כדי לספק נתונים ל-ListView או ל-RecyclerView. הוא משמש כגשר בין מקור הנתונים לרכיב ממשק המשתמש שמציג את הנתונים. האדפטר אחראי על יצירת תצוגה עבור כל פריט במקור הנתונים.

כדי ליצור חנות באמצעות מחלקת אדפטר, תחילה הגדרתי מקור נתונים המכיל מידע על הפריטים שבחנות, כגון שמותיהם, מחירים, תמונות שלהם, והאם הם בבעלות המשתמש. לאחר מכן יצרתי מחלקת אדפטר מותאמת אישית שיורשת את המחלקה ArrayAdapter.

במחלקת האדפטר, קיימת הפעולה getView, שאחראית ליצירת View עבור כל פריט בחנות. הפעולה תנפח לייאוט שיכיל את כל הViews של החנות.

כדי להציג את הפריטים, יצרתי ListView בלייאוט, והגדרתי את האדפטר. לאחר מכן, האדפטר יוסיף לרשימה את כל הפריטים שהוא קיבל מהArrayList של הפרגמנט שבו נוצרת החנות.

בנוסף להצגת מידע על המוצר, השתמשתי במחלקת האדפטר כדי לטפל באינטראקציה של המשתמש עם החנות שלך, כך שהמשתמש יוכל לבחור פריט על ידי לחיצה על פריט שבבעלותו, ולקנות פריט על ידי לחיצה על פריט שללא בבעלותו.

4.8.6.1. המחלקה ShopItem

המחלקה ShopItem היא מחלקה של פריט בודד בחנות, התכונות של המחלקה יקבעו מה השחקן יראה עבור כל מלבן שמייצג פריט בחנות.

ShopItem – התכונות	
התכונה	תיאור התכונה
private Bitmap bitmap	תמונת הפריט
private final String title	שם הפריט
private final int price	מחיר הפריט
private boolean owned	בוליאן שמציין האם הפריט בבעלות המשתמש

ShopItem – הפעולות	
הפעולה	תיאור הפעולה
public ShopItem(Context context, int bitmapID, String title, int price, boolean owned)	פעולה בונה שמאתחלת את נתוני הפריט ומשנה את המספר שמציין את הביטמאפ אל הביטמאפ עצמו
public Bitmap getBitmap()	פעולה שמחזירה את תמונת הפריט
public String getTitle()	פעולה שמחזירה את שם הפריט
public int getPrice()	פעולה שמחזירה את מחיר הפריט
public boolean isOwned()	פעולה שמחזירה את הבוליאן שמציין אם הפריט בבעלות המשתמש
public void setOwned	פעולה שמשנה את ערך הבוליאן שמציין אם הפריט בבעלות המשתמש
public void turnIDToBitmap(Context context, int id)	פעולה שמקבלת מספר כפרמטר ומשנה אותו לביטמאפ של הפריט

4.8.6.2. המחלקה ShopItemAdapter

המחלקה ShopItemAdapter היא Adaptern של החנות. Adaptern הוא גורם שהופך את הרשימה של פריטי החנות לרשימה של View שיוכלו להיות מוצגים בחנות. המחלקה יורשת את המחלקה ArrayAdapter<ShopItem>.

ShopItemAdapter – התכונות	
התכונה	תיאור התכונה
private final Context context	תמונת הפריט
private final List<ShopItem> objects	שם הפריט
private final String tag	מחיר הפריט
private TextView tvPrice	בוליאן שמציין האם הפריט בבעלות המשתמש

ShopItemAdapter – הפעולות	
הפעולה	תיאור הפעולה
public ShopItemAdapter(Context context, int resource, int textViewResourceId, List<ShopItem> objects, String tag)	פעולה בונה שמאתחלת
@SuppressWarnings("NonConstantResourceId") @Override public View getView(int position, View convertView, ViewGroup parent)	פעולה שלפי סוג הרשימה, הופכת את כל הפריטים בה לView
public void updateTextColor(int position)	משנה את צבע הפריט לירוק אם הוא נרכש

4.9. המחלקה LevelSelectActivity

המחלקה LevelSelectActivity היא המחלקה של האקטיביטי שבאמצעותו השחקן יכול לגשת לשלבים השונים של המשחק. במסך, השחקן יראה עולם אחד בכל פעם, כשהוא יכול להחליף בין העולמות 1-5 באמצעות החצים שבצד התחתון של המסך. המסך הראשון שהשחקן רואה הוא של העולם שבו השלב הבא שהשחקן צריך לעבור. לחיצה על כל שלב תפתח דיאלוג שמתאים לשלב לפי המספר שעל השלב שנלחץ. במסך יש כפתורים שחפתחו את דיאלוג דרגת הקושי, ומידע על האובייקטים, וכפתור חזרה למסך הראשי. המחלקה יורשת את המחלקה AppCompatActivity, ומיישמת את הממשק View.OnClickListener.

LevelSelectActivity – התכונות	
תיאור התכונה	התכונה
רקע המסך.	private FrameLayout frameLayout
הטקסט שעליו מוצג מספר העולם	private TextView tvWorldDisplay
כפתורים עבור כל שלב בעולם, פתיחת דיאלוג המידע על האובייקט, ופתיחת דיאלוג הרמת קושי.	private Button btL1, btL2, btL3, btL4, btBoss, btObstInfo, btDifficulty
כפתור חזרה למסך הראשי, ומעבר אחורה וקדימה בעולמות.	private ImageButton ibBack, ibBackArrow, ibForwardArrow
המספר שמציין את העולם שמוצג לשחקן	private int worldNum

LevelSelectActivity – הפעולות	
תיאור הפעולה	הפעולה
מציגה על האקטיביטי את הלייאווט המתאים, מאתחלת את הטקסט, הכפתורים, גורמת לכל הכפתורים להיות לחיצים, מאתחלת את העולם כך שיציג את העולם שבו השלב הבא שהשחקן צריך לעבור, משנה את כפתור הרמת הקושי לפי רמת הקושי.	@Override protected void onCreate(final Bundle savedInstanceState)
אם כפתור החזרה נלחץ, המסך יסגר, אם הכפתור שמשנה את מספר העולם קדימה או אחורה נלחץ, הפעולה תשנה את מספר העולם ותקרא ל changeWorld(). אם לשחקן יש אינטרנט והוא לוחץ על אחד השלבים, כפתור רמת הקושי, או כפתור המידע על האובייקטים, יפתח לשחקן דיאלוג בהתאם, אבל אם אין לו אינטרנט, תוצג לו הודעה בהתאם.	Override@ public void onClick(View view)
משנה את טקסט העולם, את כפתורי השלבים, ואת הרקע לפי מספר העולם.	public void changeWorld()
משנה את כפתור הרמת הקושי לפי רמת הקושי.	public void difficultyColor()

4.9.1. המחלקה DifficultyDialog

המחלקה DifficultyDialog היא מחלקה של הדיאלוג שהשחקן יכול לפתוח דרך מסך בחירת השלב. כפתור פתיחת הדיאלוג הוא כפתור שנראותו משתנה לפי דרגת הקושי שלו, כל דרגת קושי משפיעה על הטקסט שכתוב עליו, וצבע הדיאלוג (קל=ירוק, בינוני=כתום, קשה=אדום). הדיאלוג מאפשר לשחקן לשנות את דרגת הקושי שלו. רמות הקושי הם קל, בינוני, וקשה. במסך יש 3 כפתורים של דרגת קושי, ולחיצה על כל אחד מהם יראה מידע על דרגת הקושי, וכפתור שמירת השינוי וסגירת הדיאלוג. הדיאלוג כולל כפתור חזרה למסך ההתחברות. המחלקה יורשת את המחלקה Dialog, ומיישמת את הממשק View.OnClickListener.

DifficultyDialog – התכונות	
התכונה	תיאור התכונה
private final Context context	הקונטקסט של הדיאלוג
private Button btEasy, byNormal, btHard, btConfirm	כפתור רמת קושי קלה, בינונית, קשה ואישור
private int diff	מספר שמייצג דרגת קושי. מאותחל לפי רמת הקושי הנוכחית ומשתנה כל פעם שלוחצים על כפתור של דרגת קושי
private TextView tvDisplay	טקסט שכתוב עליו את ההבדלים בין רמות הקושי ומשתנה כל פעם שלוחצים על כפתור של רמת קושי

DifficultyDialog – הפעולות	
הפעולה	תיאור הפעולה
public DifficultyDialog (Context context)	פעולה שמאתחלת את הדיאלוג ואת משתנה הקונטקסט
@Override protected void onCreate(final Bundle savedInstanceState)	מציגה על הדיאלוג את הלליאווט המתאים, גורמת לכך שלא יהיה אפשר ללחוץ מחוץ לדיאלוג כדי להעלים אותו, משייכת את כל הכפתורים ללליאווט, גורמת לכפתורים להיות לחיצים, מאתחלת את משתנה דרגת הקושי
@Override public void onClick(View view)	כאשר לוחצים על כפתור של רמת קושי, הפעולה תציג את הטקסט המתאים ותשנה את המספר שמייצג אותה. כאשר לוחצים על כפתור האישור, הפעולה תעדכן את רמת הקושי בפירבייס ובמשתנה המשתמש של האפליקציה, הפעולה בנוסף תסגור את הדיאלוג ותעדכן את נראות כפתור שינוי דרגת הקושי דרך הפעולה של האקטיביטי LevelSelectActivity

4.9.2. המחלקה LevelInfoDialog

המחלקה LevelInfoDialog היא מחלקה של הדיאלוג שהשחקן יכול לפתוח על ידי לחיצה על שלב במסך בחירת השלב. אם השלב הקודם הושלם, הדיאלוג יציג כפתור לשיר של השלב, וכפתור לשחק את השלב. אם השלב הקודם לא הושלם, הדיאלוג יציג לשחקן את הדרישות כדי לפתוח את השלב. בנוסף, יש כפתור לסגור את הדיאלוג. המחלקה יורשת את המחלקה Dialog, ומיישמת את הממשק View.OnClickListener.

LevelInfoDialog – התכונות	
התכונה	תיאור התכונה
private final Context context	הקונטקסט של הדיאלוג
private final int levelID	מספר השלב שהשחקן משחק
private final Bundle bitmapArrays	חבילה של ביטמאפים של תמונות שמשמשות במשחק
private Button btSong, btClose, btPlay	כפתור למוזיקה, כפתור סגירת דיאלוג, כפתור לשחק

LevelInfoDialog – הפעולות	
הפעולה	תיאור הפעולה

<code>public LevelInfoDialog(Context context, int levelID, Bundle bitmapArrays)</code>	פעולה שמאתחלת את הדיאלוג, את משתנה הקונטקסט, את מספר השלב, ואת חבילת התמונות
<code>@Override</code> <code>protected void onCreate(final Bundle savedInstanceState)</code>	מציגה על הדיאלוג את הלייאוט המתאים, גורמת לכך שלא יהיה אפשר ללחוץ מחוץ לדיאלוג כדי להעלים אותו, מציגה את מספר השלב על הטקסט המתאים בלייאוט, משייכת את משתנה הניקוד/זמן ללייאוט, משייכת את הכפתורים ללייאוט וגורמת להם להיות לחיצים. הפעולה בודקת אם השלב הקודם הושלם באמצעות הפעולה <code>isPrevHighScoreOrCompleted</code> . אם כן, יוצג הטקסט המתאים על כפתור השיר, טקסט הניקוד/זמן, כפתור השיר, וכפתור הלשחק יהיו גלויים על המסך. אם לא, טקסט שכתוב עליו: Requirements, והטקסט שכתוב עליו הדרישות יהיו גלויים על המסך במקום.
<code>public boolean isPrevHighScoreOrCompleted()</code>	בודקת ומחזירה אמת עבור כל שלב שמתחלק 5 עם שארית של 1 אם השלב הקודם הושלם, ועבור כל שלב אחר, תבדוק אם השחקן השיג מספיק נקודות בשלב הקודם
<code>public String setBossLevelText()</code>	מחזירה מחרוזת שתכתוב את הזמן הכי טוב של השחקן בשלב בוס, או שתכתוב שהשלב הושלם אם השחקן סיים את השלב
<code>public int scoreReq()</code>	מחזירה את מספר הנקודות שיפתחו את השלב לפי מספר השלב הקודם
<code>public String requirements()</code>	מחזירה מחרוזת שתכתוב לשחקן את הדרישות לפתיחת השלב
<code>@Override</code> <code>public void onClick(View view)</code>	אם כפתור הסגירה נלחץ, הדיאלוג נסגר. אם כפתור המשחק נלחץ, הדיאלוג יפתח אינטנט למסך המשחק יעביר אליו את מספר השלב וחבילת התמונות, יתחיל את האקטיביטי ויסגור את הדיאלוג. אם כפתור המוזיקה נלחץ, הדיאלוג ימצא את כתובת הURL המתאימה של השיר לפי מספר השלב, ותפתח אינטנט מרומז שתפתח את אפליקציית היוטיוב לפי כתובת זו, או את הדפדפן אם למשתמש לא מותקן אפליקציית היוטיוב

4.9.3. המחלקה `ObstInfoDialog`

המחלקה `ObstInfoDialog` היא מחלקה של הדיאלוג שהשחקן יכול לפתוח דרך מסך בחירת השלב. בדיאלוג זה, השחקן יכול לקרוא על כל אובייקט כדי לדעת איך לפעול נגדו. על הדיאלוג מופיע מופיע `horizontal scroll view` שהשחקן יכול לגלול דרכו וללחוץ כל אחד מהאובייקטים כדי לשנות את הטקסט שמופיע מתחת לדיאלוג. הדיאלוג כולל כפתור סגירת דיאלוג. המחלקה יורשת את המחלקה `Dialog`, ומיישמת את הממשק `View.OnClickListener`.

תיאור התכונה	התכונה
הקונטקסט של הדיאלוג	private final Context context
מחזיק את כל התמונות שיוצגו ב horizontal scroll view	private LinearLayout obstaclesList
טקסט וויו עבור שם האובייקט, והתיאור שלו	private TextView tvTitle, tvDesc

ObstInfoDialog – הפעולות	
תיאור הפעולה	הפעולה
פעולה שמאתחלת את הדיאלוג ואת משתנה הקונטקסט	public ObstInfoDialog(Context context)
מציגה על הדיאלוג את הלייאוט המתאים, גורמת לכך שלא יהיה אפשר ללחוץ מחוץ לדיאלוג כדי להעלים אותו, משייכת את הכפתור ללייאוט, גורמת לכפתור להיות לחיצ, מייצרת ImageView לכל אובייקט אפשרי על פי הצבע, או על פי תמונה ספציפית לאובייקט האזהרה	@Override protected void onCreate(final Bundle savedInstanceState)
מחזירה צבע לפי אינדקס	public String getColor(int i)
משנה את הטקסט של שם האובייקט	public void setInfoTitle (int j)
משנה את טקסט של תיאור האובייקט	public void setInfoDesc (int j)

4.10. המחלקה GameScreenActivity

המחלקה GameScreenActivity היא המחלקה של האקטיביטי של המשחק, אקטיביטי זה מתחיל על ידי בדיקה של המשתנה הבוליאני שאומר האם השחקן בחר שיהיה מוזיקת רקע במשחק. אם כן, האפליקציה תקבל כתובת url מהFirebase Storage לפי מספר השלב, ותנגן אותו באמצעות מחלקת MediaPlayer, ולאחר מכן תוסיף את פאנל המשחק. בזמן טעינת השיר, המסך יהיה שחור ויהיה כתוב עליו שהמשחק נטען. אם השחקן בחר שלא יהיה מוזיקת רקע, פאנל המשחק יטען ישר. המסך כולל כפתור חזרה למסך בחירת שלב, טקסט להצגת כמות פריימים לשנייה, תמונה שמציגה שהשחקן לא מחובר לאינטרנט באמצעות BroadcastReceiver והתמונה+הטקסט של הכח של השחקן. המחלקה יורשת את המחלקה Activity, ומיישמת את הממשקים View.OnClickListener, SensorEventListener.

GameScreenActivity – התכונות	
תיאור התכונה	התכונה
טקסט הטעינה, וטקסט הפריימים לשנייה	private TextView tvLoading, tvFPS
פאנל המשחק	private GamePanel gamePanel
הלייאוט שעליו פאנל המשחק מתווסף	private FrameLayout frameLayout
תמונת החיבור לאינטרנט	private ImageView ivConnection
כפתור חזרה למסך בחירת שלב	private ImageButton ibBack
מספר השלב	private int levelID
נגן המוזיקה של המשחק	private MediaPlayer mediaPlayer
מערך תמונות עבור תמונת השחקן, הרקע, הבונוס, והכח	private int[] iconImages, backgroundImages, bonusImages, powerUpImages
כמות הפעמים שהמשחק אותחל, המשתנה פותר בעיות במחלקת הכח	private int retryCount = 0
בודק כל פעם שהשחקן מתנתק או מתחבר לאינטרנט וקורא לפעולה changeConnectionErrorVisibility	private final BroadcastReceiver connectivityChangeReceiver

כשהפרמטר הוא אמת אם השחקן מחובר, ושקר אם הוא לא	
---	--

GameScreenActivity – הפעולות	
הפעולה	תיאור הפעולה
@Override protected void onCreate(final Bundle savedInstanceState)	מציגה על האקטיביטי את הלייאווט המתאים, מאתחלת את הרפרנסים לפיירבייס, משייכת את הטקסטים, התמונות, והכפתורים ללייאווט, גורמת לכפתורים להיות לחיצים, מקבלת את מספר השלב דרך האינטנט שהעביר בין מסך בחירת השלב למסך המשחק באמצעות המספר שהיה כתוב על הכפתור שנלחץ. מקבלת את מערכי התמונות ממסך בחירת השלב. מכינה את המוזיקה או מוסיפה את פאנל המשחק לפי בחירת השחקן בהגדרות
public GamePanel getGamePanel()	מחזיר את פאנל המשחק
public int getLevelID()	מחזיר את מספר השלב
public int getRetryCount()	מחזיר את כמות הפעמים שהמשחק אותחל
public int[] getPowerUpImages()	מחזיר את מערך תמונות הכח
public void setFPSView(double FPS)	משנה את טקסט הפריימים לשנייה
public void onRetry(GamePanel newGamePanel)	מוסיפה אחד לכמות retryCount, מחליפה את פאנל המשחק הקודם באחד חדש
@Override public void onClick(View view)	גורמת לפאנל המשחק לסיים את המשחק וחוזרת למסך בחירת השלב כשהשלב הוא שלב רגיל או שלב בוס, ולמסך ההתחברות כשהשלב הוא שלב דמו
public void addGamePanel()	גורמת לטקסט הטעינה להעלים ומוסיפה פאנל משחק למסך
public void prepareMusic()	מכינה את המוזיקה לפי כתובת url שהוא מקבל מ Firebase Storage לפי מספר השלב
public void playMusic()	מתחילה את המוזיקה, מנגן אותה מחדש כשהשיר נגמר
public void stopMusic()	מפסיקה את המוזיקה אם היא מתנגנת
public void changeConnectionErrorVisibility(boolean connection)	הפעולה מציגה או מעלים את תמונת החיבור לפי הבוליאן שהיא מקבלת כפרמטר
protected void onResume()	מתחיל את BroadcastReceiver
protected void onPause()	מפסיק את BroadcastReceiver ואת המוזיקה. מרעיד את הטלפון
public int getBitmapID (int type)	מחזיר את התמונות בהתאם לתמונות שהשחקן בחר, אם השחקן משחק את שלב הדמו, הפעולה תחזיר את תמונות ברירות המחדל, 0 עבור הכח, כי שלחקן אין כח כשהוא משחק את שלב הדמו

4.10.1. שימוש במחלקה **BroadcastReceiver** כדי לבדוק שינויים בחיבור של המשתמש לאינטרנט

ב-Android Studio, Broadcast Receiver הוא רכיב שמקבל ומטפל בהודעות שידור בכל המערכת. הודעות שידור הן הודעות שנשלחות על ידי המערכת או על ידי אפליקציה כדי להודיע ליישומים או רכיבים אחרים על אירועים מסוימים. לדוגמה, במשחק שלי, כדי לבדוק חיבור לאינטרנט בכל רגע נתון באמצעות הפעולה הסטטית isConnectedToInternet שבפרויקט שלי.

כדי ליצור Broadcast Receiver ב-Android Studio, הוספתי שדה של BroadcastReceiver באקטיביטי ובאמצעות הפעולה onReceive שייכתי אותה לאקטיביטי. פעולה זו נקראת כאשר הודעת השידור מתקבלת על ידי המערכת. לאחר מכן קראתי לפעולה changeConnectionErrorVisibility כדי לטפל באירוע שהפעיל את הודעת השידור.

לאחר שיצרתי את Broadcast Receiver, עשיתי לו register במערכת באמצעות הפעולה registerReceiver.

4.10.2. המחלקה **BossLevelCompleteDialog**

המחלקה BossLevelCompleteDialog היא מחלקה של הדיאלוג שמופיע לשחקן אחרי שהוא מביס שלב בוס, שהוא שלב שמבוסס על שעון שסופר אחורה עד שמגיע ל00:00. אם השחקן מחובר לאינטרנט, הדיאלוג יעדכן את בסיס הנתונים שהשחקן עבר את השלב. אם זוהי הפעם הראשונה שהשחקן עבר את השלב הדיאלוג יציג לשחקן הודעה שהעולם הבא נפתח. אם השחקן לא מחובר לאינטרנט הדיאלוג יגיד לשחקן שהשלמת השלב לא נשמרה במערכת. הדיאלוג כולל גם כפתור חזרה למסך בחירת שלב. המחלקה יורשת את המחלקה Dialog, ומיישמת את הממשק View.OnClickListener.

BossLevelCompleteDialog – התכונות	
התכונה	תיאור התכונה
private final Context context	הקונטקסט של הדיאלוג
private final int levelID	מספר השלב שהשחקן סיים
private TextView tvWorldUnlocked	הטקסט שיציג לשחקן הודעה אם הוא פתח שלב חדש
private Button btBack	כפתור חזרה למסך בחירת שלב

BossLevelCompleteDialog – הפעולות	
הפעולה	תיאור הפעולה
public BossLevelCompleteDialog(Context context, int levelID)	הפעולה מאתחלת את הדיאלוג, משתנה הקונטקסט, ומשתנה מספר השלב
@Override protected void onCreate(final Bundle savedInstanceState)	מציגה על הדיאלוג את הלייאוט המתאים, גורמת לכך שלא יהיה אפשר ללחוץ מחוץ לדיאלוג כדי להעלים אותו, משייכת את tvWorldUnlocked ללייאוט, גורמת לכפתור להיות לחיצה, קוראת לפעולה setWorldUnlocked()
public void setWorldUnlocked ()	אם השחקן מחובר לאינטרנט, הפעולה מעדכנת את בסיס הנתונים שהשחקן עבר את השלב. אם זוהי הפעם הראשונה שהשחקן עבר את השלב הפעולה תציג לשחקן הודעה שהעולם הבא נפתח. אם השחקן לא מחובר לאינטרנט הפעולה תגיד לשחקן שהשלמת השלב לא נשמרה במערכת.

@Override public void onClick(View view)	מעלימה את הדיאלוג ומחזירה את השחקן למסך בחירת השלב
---	--

4.10.3. המחלקה DemoGameOverDialog:

המחלקה BossLevelCompleteDialog היא מחלקה של הדיאלוג שמופיע לשחקן אחרי שהוא נפסל בשלב הדמו של המשחק, שהוא השלב שהשחקן יכול לשחק בלי להתחבר למערכת. הדיאלוג כולל כפתור חזרה למסך ההתחברות. המחלקה יורשת את המחלקה Dialog, ומיישמת את הממשק View.OnClickListener.

DemoGameOverDialog – התכונות	
התכונה	תיאור התכונה
private final Context context	הקונטקסט של הדיאלוג
private Button btBack	כפתור חזרה למסך ההתחברות

DemoGameOverDialog – הפעולות	
הפעולה	תיאור הפעולה
public DemoGameOverDialog (Context context)	פעולה שמאתחלת את הדיאלוג ואת משתנה הקונטקסט
@Override protected void onCreate(final Bundle savedInstanceState)	מציגה על הדיאלוג את הלייאוט המתאים, גורמת לכך שלא יהיה אפשר ללחוץ מחוץ לדיאלוג כדי להעלים אותו, משייכת את btBack ללייאוט, גורמת לכפתור להיות לחיצ
@Override public void onClick(View view)	מעלימה את הדיאלוג ומחזירה את השחקן למסך ההתחברות

4.10.4. המחלקה GameOverDialog:

המחלקה GameOverDialog היא מחלקה של הדיאלוג שהשחקן רואה כשהוא נפסל בכל שלב שהוא לא שלב הדמו. אם השחקן שיחק שלב רגיל, הדיאלוג יציג לשחקן את כמות הנקודות שהשחקן קיבל ויבדוק אם זה שיא נקודות חדש. אם השחקן שיחק שלב בוס, המשחק יציג את הזמן שנשאר לשחקן ויבדוק אם זה שיא זמן חדש. על הדיאלוג יש כפתור לשחק שוב, וכפתור חזרה למסך בחירת שלב. המחלקה יורשת את המחלקה Dialog, ומיישמת את הממשק View.OnClickListener.

GameOverDialog – התכונות	
התכונה	תיאור התכונה
private final Context context	הקונטקסט של הדיאלוג
private final GameScreenActivity activity	האקטיביטי של המסך שעליו משוחק המשחק
private TextView tvNewHighScore	מספר שמייצג דרגת קושי. מאותחל לפי רמת הקושי הנוכחית ומשתנה כל פעם שלוחצים על כפתור של דרגת קושי
private TextView tvDisplay	טקסט שכתוב עליו את ההבדלים בין רמות הקושי ומשתנה כל פעם שלוחצים על כפתור של רמת קושי
private Button btBack, btRetry	כפתור חזרה למסך בחירת שלב, וכפתור נסה שוב
private final int score	הניקוד שקיבל השחקן
private final String time	הזמן שהשחקן הגיע אליו

מספר השלב שהשחקן משחק	private final int levelID
-----------------------	---------------------------

GameOverDialog – הפעולות	
הפעולה	תיאור הפעולה
public GameOverDialog(Context context, int score, String time, int levelID)	פעולה שמאתחלת את הדיאלוג, את משתנה הקונטקסט, את משתנה האקטיביטי לפי הקונטקסט, את הניקוד, הזמן, מספר השלב, את הרפרנס לדטאבייס לפי פיירבייס דטאבייס. הפעולה מעלה את מספר הפעמים שהשחקן מת ב-1 ומעדכנת את מספר זה בפיירבייס
@Override protected void onCreate(final Bundle savedInstanceState)	מציגה על הדיאלוג את הלייאווט המתאים, גורמת לכך שלא יהיה אפשר ללחוץ מחוץ לדיאלוג כדי להעלים אותו, משייכת את הטקסט והכפתורים ללייאווט, גורמת לכפתורים להיות לחיצים, מציגה על הטקסט tvDisplay של הלייאווט את הניקוד עבור שלב רגיל, והזמן עבור שלב בוס
@Override public void onClick(View view)	כאשר לוחצים על כפתור החזרה, הדיאלוג נסגר והאקטיביטי נסגרים והשחקן חוזר למסך בחירת שלב, אם לוחצים על כפתור הנסה שוב, המשחק מתחיל את עצמו מחדש
public void tryToSetNewProgression()	הפעולה בודקת אם השחקן מחובר לאינטרנט, אם השחקן מחובר, הפעולה תיצור משתנה שיתחבר לפיירבייס דטאבייס של המשחק, ולאחר מכן, תעלה בדוק אם הוא השיג שיא נקודות חדש (כשהוא משחק בשלב רגיל) או שיא זמן חדש (כשהוא משחק בשלב בוס), אם הושג שיא חדש, הפעולה תציג אותה מתאימה באמצעות הפעולה displayNewMessage. אם השחקן לא מחובר לאינטרנט
public boolean isBestTime(String oldBestTime)	הפעולה תפרק למספרים את מחרוזת הזמן שהשחקן קיבל, ומחרוזת הזמן של שיא הזמן הקודם, שאותו היא מקבלת כפרמטר, ותחזיר אמת אם הזמן נמוך יותר, ושקר אם לא
public void displayNewMessage(boolean type)	כותבת על המסך שהשחקן קיבל שיא חדש בהתאם לסוג השלב. הפעולה יודעת מהו סוג השלב דרך הפרמטר, אם הוא אמת, השלב הוא רגיל, אם הוא שקר, השלב הוא שלב בוס. הפעולה תגרום כך שהטקסט יהיה נראה על הדיאלוג

4.10.5. שימוש במחלקה Canvas כדי ליצור לוח משחק

שימוש בקנבס ובתרד היא גישה פופולרית ליצירת משחקים ב-Android Studio. הקנבס מספק משטח ציור עליו ניתן לעבד גרפיקה, בעוד שהתרד מאפשר עדכונים רציפים למצב המשחק ולממשק המשתמש.

בהתחלה, יצרתי מחלקת פאנל משחק תצוגה מותאמת אישית שמרחיבה את מחלקת ה-Android SurfaceView ועושה override את לפעולה draw. פעולה זו תטפל בכל העיבוד הגרפי באמצעות אובייקט הקנבס. ניתן להשתמש בקנבס כדי לצייר צורות, טקסט ותמונות.

לאחר מכן, יצרתי תרד חדש שיריץ את לולאת המשחק. לולאה זו תעדכן בלי הפסקה את מצב המשחק, תטפל בקלט משתמש ותצייר מחדש את הקנבס. חשוב לנהל את השרשור בצורה נכונה, להשהות ולחדש אותו לפי הצורך כדי להבטיח שימוש יעיל במשאבי המערכת.

בלולאת המשחק, עדכנתי את מצב המשחק על ידי בדיקת התנגשויות, עדכון מצבו של כל אובייקט שמצויר על המסך וטיפול במצבי משחק אלו באמצעות הפעולה update. טפילתי בקלט משתמש על ידי האזנה לאירועי מגע ועיבודם בהתאם. לבסוף, ציירתי מחדש את הקנבס עם מצב המשחק המעודכן באמצעות אובייקט הקנבס.

שימוש בקנבס ובתרד בדרך זו יכול לספק חווית משחק חלקה ומהנה למשתמשים.

4.10.5.1 המחלקה GamePanel

המחלקה GamePanel היא המחלקה הבסיסית של הלוח והמשחק. המחלקה הזו אחראית על האלגוריתם שמאחורי המשחק. המחלקה מציגה על קנבס את כל האובייקטים והטקסט שצריכים להופיע על המסך. המחלקה יורשת את המחלקה SurfaceView ומיישמת את הממשק SurfaceHolder.Callback.

GamePanel – התכונות	
תיאור התכונה	התכונה
התרד שאחראי על הרצת המשחק	private MainThread thread
הקונטקסט של GameScreenActivity שפתח את הפאנל	private final Context context
משתנה שגורם לאקראיות במשחק	private final Random rnd
משתנה של הסאונד אפקטים במשחק	private final SFX sfx
תמונת הרקע של המסך	private final Bitmap bitmapBackground
משתנה שעוזר לגרם לאפקט הגלילה של המסך	private int counter = 0
השחקן	private final Player player
אובייקט הבונוס	private final BonusObj bonusObj
הכח של השחקן	private final PowerUp powerup
מציג את טקסט הבונוס כאשר המשתנה הוא אמת	private boolean bonusTextDraw = false
מחזיק את התוכן שיוצג בטקסט הבונוס	private String bonusString = ""
מחליט איזה בונוס השחקן יקבל באקראיות	private int bonusID
מחזיק את התכונות של טקסט הנקודות והבונוס	private Paint scorePaint, bonusPaint
מחזיק את מספר הנקודות שהשחקן מקבל בכל שנייה לפי דרגת הקושי	private int extraScore
המשתנה יהיה אמת אם מהירות המכשולים לא רגילה	private boolean isNotNormalSpeed
החיישנים של המשחק	private final GameSensors gameSensors
תרד המכשולים של המשחק	private ObstacleThread obstacleThread
תרד הניקוד של המשחק	private ScoreThread scoreThread
תרד המשתנה שאחראי על האובייקטים	private final ObstacleArrayManager obstacleArrayManager
ניקוד השחקן בשלב רגיל	private int score = 0
זמן השחקן בשלב בוס	private String time
מחזיק את מספר השלב	private int levelID
מחזיק את סוג השלב לפי מספר השלב	private String levelType
האם המשחק נגמר	private boolean gameOver = false

GamePanel – הפעולות	
הפעולה	תיאור הפעולה
public GamePanel(Context context)	פעולה בונה למחלקה שמאתחלת את כל המשתנים במשחק
public SFX getSFX()	מחזירה את משתנה הסאונד אפקטים
public Player getPlayer()	מחזירה את משתנה השחקן
public PowerUp getPowerUp()	מחזירה את משתנה הכח של השחקן
public GameSensors getGameSensors()	מחזירה את משתנה של חיישני המשחק
public boolean isNotNormalSpeed()	מחזיר אם המשתנה של האם המכשולים לא במהירות הרגילה
public void setNotNormalSpeed(boolean notNormalSpeed)	שם בוליאן במשתנה של האם המכשולים לא במהירות הרגילה
public ObstacleArrayManager getObstacleArrayManager()	מחזיר את המשתנה שאחראי על האובייקטים
public void surfaceChanged(SurfaceHolder holder, int format, int width, int height)	חלק מפעולות החובה של הממשק
public void surfaceCreated(surfaceHolder holder)	מתחיל את כל התרדים כשהמסך נוצר, חלק מפעולות החובה של הממשק
public void surfaceDestroyed(SurfaceHolder holder)	חלק מפעולות החובה של הממשק
public boolean onTouchEvent(MotionEvent e)	מזיז את השחקן לכיוון האצבע של המשתמש כאשר האצבע של המשתמש זזה על המסך
public void draw(Canvas canvas)	מציירת על הקנבס את השחקן, המכשולים, טקסט של דמו/ניקוד/זמן בהתאם לסוג השלב, טקסט הבונוס. מסיים את המשחק שהשחקן נפסל.
public void update()	מעדכן את נתוני השחקן, הבונוס, המכשולים, מזיז את השחקן למיקומו ההתחלתי כשהוא נפסל, בודק אם השחקן נגע בבונוס
public void backgroundDrawer(Canvas canvas)	מצייר שני מלבנים שזזים לפי המשתנה counter כדי ליצור אפקט גלילה על המסך
public static synchronized void drawBitmap(Canvas canvas, Bitmap bitmap,int x, int y, int width, int height)	יוצר את המלבן לפי המשתנים
public void checkBonusIDValidity()	אם מהירות המכשולים לא רגילה המספר של הבונוס ישתנה ל-1
public void createBonus()	מגריל בונוס באקראיות, ורושם מה הבונוס על המסך לזמן קצר
public void dialogMaker(String dialogType)	יוצר דיאלוג לפי סוג, הדיאלוגים הם: משחק נגמר, שלב הדמו נגמר, סיום שלב בוס
public void scoreUpdater()	מעדכן את הניקוד או הזמן כל שנייה
public String reduceTimeByOneSecond(String time)	מקבל טקסט של זמן, מוריד את הזמן לשנייה וממיר את הזמן לטקסט
public void finishGame()	מפסיק את כל התרדים, החיישנים, והמוזיקה

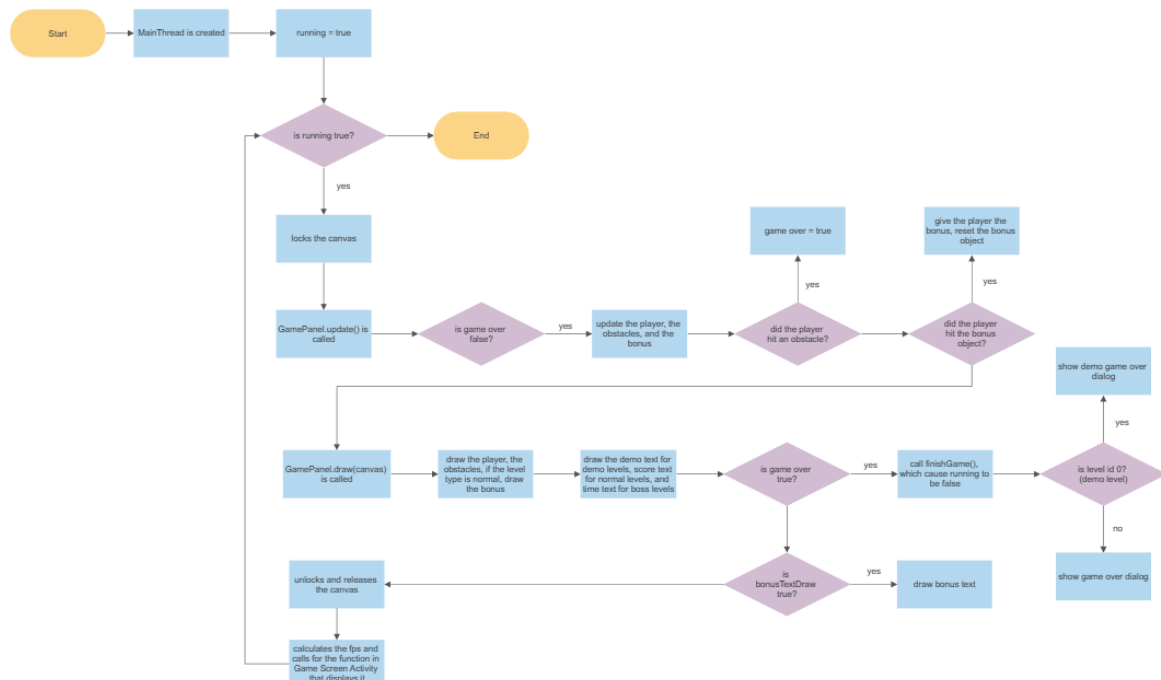
4.10.5.2. MainThread המחלקה

המחלקה MainThread היא המחלקה של התרד הראשי שמופעל כאשר השחקן בזמן משחק. המחלקה גורמת לקנבס להתעדכן בכל חלקיק שנייה שהמשחק רץ בו. המחלקה מורשה את המחלקה Thread.

MainThread – התכונות	
התכונה	תיאור התכונה
public static final int MAX_FPS = 30	מגדיר את כמות הפריימים לשנייה המקסימלי כ-30
public double averageFPS	שומר את כמות הפריימים לשנייה הממוצע בזמן שהמשחק רץ
private final SurfaceHolder surfaceHolder	שומר את המשתנה מסוג SurfaceHolder של פאנל המשחק
private final GamePanel gamePanel	שומר את פאנל המשחק
private boolean running	משתנה של האם התרד רץ
public static Canvas canvas	הקנבס של המשחק שמתעדכן ומצויר על המסך

MainThread – הפעולות	
הפעולה	תיאור הפעולה
public MainThread(SurfaceHolder surfaceHolder, GamePanel gamePanel)	פעולה בונה למחלקה שמשיגה את surface holder ואת פאנל המשחק
public void setRunning()	שם בוליאן במשתנה של האם התרד רץ
@Override public void run()	הפעולה רצה כל עוד המשחק רץ. הפעולה מכילה לולאה שמעדכנת ומציירת את פאנל המשחק, מחשבת את כמות הפריימים לשנייה ומשנה את המשתנה באקטיביטי של המשחק למספר זה

4.10.5.2.1. תרשים זרימה של הרצה של התרד MainThread



4.10.5.3. המחלקה **ObstacleArray**:

המחלקה ObstacleArray היא המחלקה של כל מופע של קבוצת אובייקטים על המסך. בפעולה יש תכונות שקובעות מה סוג האובייקט שיופיע ומה כמות האובייקטים במופע.

ObstacleArray – התכונות	
התכונה	תיאור התכונה
private final Context context	הקונטקסט של GameScreenActivity שהתחיל את המשחק
private final RectGame[] obstArr	מערך של מלבני משחק
private int id	מחזיק מספר שמציין את סוג האובייקט של מלבני המערך (1-MovingRect, 2-Alpha, 3-MovingRect, 4-Gap, 5-Warning, 6-DownUp, 7-Crossing, 8-MovingGap, 9-CrossingSingle, 10-Bounce, PositionSensor)

ObstacleArray – הפעולות	
הפעולה	תיאור הפעולה
public ObstacleArray(Context context, Class<?> classType, int numOfObts)	פעולה בונה למחלקה שמקבלת את הקונטקסט, את המחלקה שתהיה סוג המערך, ואורך המערך. הפעולה ממירה את המחלקה למספר ויוצרת את המערך
public boolean collision(Player player)	בודקת עבור כל איבר במערך האם פגע בשחקן
public void draw(Canvas canvas)	מציירת כל איבר במערך על הקנבס
public void update()	מעדכנת כל איבר במערך
public void changeSpeed(int type)	משנה את המהירות של כל איבר במערך לפי הסוג שהפעולה מקבלת כפרמטר
public void setIdFromClass<?>(classType)	ממירה את המחלקה למספר
public RectGame makeNewInstance()	יוצרת מופע חדש של האובייקט לפי המספר

4.10.5.4. המחלקה **ObstacleArrayManager**:

המחלקה ObstacleArrayManager היא המחלקה של הרשימה של כל מערכי האובייקטים במשחק. המחלקה דואגת שכל אובייקט ימשיך להיות מצויר, ולהתעדכן על המסך גם אחרי שנוצרו אובייקטים אחרים.

ObstacleArrayManager – התכונות	
התכונה	תיאור התכונה
private final Context context	הקונטקסט של GameScreenActivity שהתחיל את המשחק
private final int randomClass	סוג המחלקה שיבחר באקראיות לפי מספר השלב
private int ArrayList<ObstacleArray> obstList	הרשימה שאליה יתווספו המערכים של האובייקטים
Private final int levelID	מספר השלב

ObstacleArrayManager – הפעולות	
הפעולה	תיאור הפעולה

public ObstacleArrayManager(Context context)	פעולה בונה למחלקה שמקבלת את הקונטקסט, ומקבלת לפיו את מספר השלב. הפעולה מאתחלת את הרשימה של המערכים
public void addArrToList()	יוצרת מערך מסוג אקראי ומוסיפה אותו לרשימה
public int getClassFromLevel(int[] arr)	מחזירה באקראיות את המספר של אחת המחלקות שיכולות להופיע בשלב
public ObstacleArray setArray()	יוצרת ומחזירה מערך לפי הסוג שנבחר
public int getSleepFromType()	מחזירה את הפרש הזמן בין היווצריות של מערכים על פי סוג האובייקט שנוצר
public int changeSleepFromDifficulty(int originalSleep)	מעדכנת את הפרש הזמן על פי דרגת הקושי שהשחקן בחר
public boolean collisionAll(Player player)	בודקת עבור כל מערך ברשימה האם אחד האובייקטים שלה פגע בשחקן
public void drawAll(Canvas canvas)	מציירת את כל האובייקטים של כל המערכים שברשימה על הקנבס
public void updateAll()	מעדכנת את כל האובייקטים של כל המערכים שברשימה
public void changeSpeedAll(int type)	משנה את המהירות של כל האובייקטים של כל המערכים שברשימה לפי הסוג שהיא מקבלת כפרמטר
public int[] getArrayFromLevelID()	מחזירה את רשימת האובייקטים שיכולים להופיע בשלב לפי סוג השלב
public int setRndNum(int type)	מחזירה כמות אובייקטים באופן אקראי לפי הסוג שהיא מקבלת כפרמטר

4.10.5.5 ObstacleThread המחלקה

המחלקה ObstacleThread היא המחלקה של התרד שמייצר אובייקטים חדשים על המסך לפי הפרשי זמן שנקבעים באובייקט ObstacleArrayManager של המשחק. הפעולה יורשת מהמחלקה Thread.

ObstacleThread – התכונות	
התכונה	תיאור התכונה
private boolean running	בוליאן שערכו הוא אמת כשהמשחק רץ והופך לשקר כשהמשחק נגמר.
private final GamePanel gamePanel	פאנל המשחק של המשחק שהתרד שייך אליו

ObstacleThread – הפעולות	
הפעולה	תיאור הפעולה
public ObstacleThread(GamePanel gamePanel)	פעולה בונה למחלקה שמקבלת את הקונטקסט, ומקבלת לפיו את מספר השלב. הפעולה מאתחלת את הרשימה של המערכים
public void setRunning(boolean running)	משנה את ערך המשתנה הבוליאני
@Override public void run()	הפעולה רצה כל עוד המשחק רץ. הפעולה היא לולאה, ובו התרד מוסיף מערך לרשימה, מקבל את הזמן שהתרד ישן לפי המערך שהתווסף לרשימה, והתרד ישן לפי אותו הזמן

4.10.5.6 ScoreThread המחלקה

המחלקה ScoreThread היא המחלקה של התרד שמוסיף נקודות לשחקן כל שנייה. הפעולה יורשת מהמחלקה Thread.

ScoreThread – התכונות	
התכונה	תיאור התכונה
private boolean running	בוליאן שערכו הוא אמת כשהמשחק רץ והופך לשקר כשהמשחק נגמר.
private final GamePanel gamePanel	פאנל המשחק של המשתדל שייך אליו

ScoreThread – הפעולות	
הפעולה	תיאור הפעולה
public ScoreThread(GamePanel gamePanel)	פעולה בונה למחלקה שמקבלת את פאנל המשחק ומאתחלת את תכונת המחלקה לפיו משנה את ערך המשתנה הבוליאני
public void setRunning(boolean running)	הפעולה רצה כל עוד המשחק רץ. הפעולה היא לולאה, ובו התרד מוסיף נקודות לשחקן וישן לשנייה
@Override public void run()	

4.10.6 PowerUp המחלקה

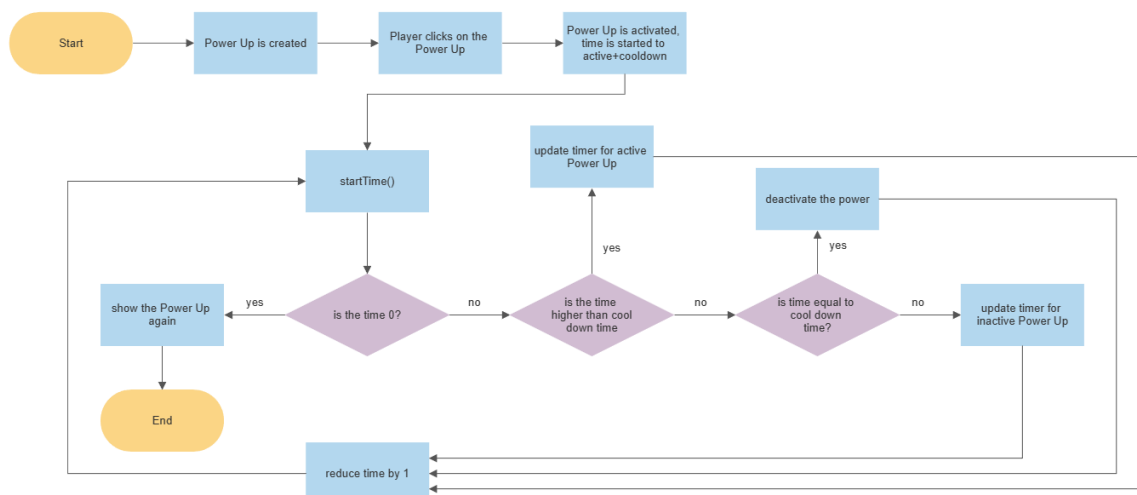
המחלקה PowerUp היא המחלקה של הכוחות שהשחקן יכול לקנות ולהשתמש בהם במהלך המשחק. במהלך המשחק, הכח יוצג לשחקן באמצעות תמונה בפינה הימנית למטה של המסך, שהשחקן יכול ללחוץ עליה כדי להפעיל את הכח, ולאחר מכן התמונה תתחלף לספירה לאחר של הזמן שבו הכח פועל, וספירה לאחר של הזמן שבו השחקן יצטרך לחכות על שיוכל להשתמש בכח שוב. המחלקה הפעולה מיישמת את הממשק View.OnClickListener.

PowerUp – התכונות	
התכונה	תיאור התכונה
private final Context context	הקונטקסט של GameScreenActivity שהתחיל את המשחק
private final GamePanel gamePanel	פאנל המשחק של המשחק שהמחלקה שייכת אליו
private final ImageButton ibPowerUp	הפנייה ל-ImageButton שנמצא על ה-GameScreenActivity
private final int imageDisplayID	מספר שמציין את הכח שהשחקן בחר, 0 אם השחקן לא בחר באף כח
private final TextView tvCountDown	הפנייה ל-TextView שנמצא על ה-GameScreenActivity
private int time, timeActive, timeCoolDown	הזמן שמוצג על המסך שמתעדכן כל שנייה, כמה זמן הכח מופעל, וכמה זמן צריך להמתין כדי שהכח יוכל להיות מופעל מחדש
private int retries = 0	כמות הפעמים שהמשחק אותחל מתעדכן כל פעם שמתחיל משחק חדש, הכרחי כדי שלא יופעל האנדלר לאחר דיליי שהתחיל במשחק הקודם

PowerUp – הפעולות	
הפעולה	תיאור הפעולה
public PowerUp(Context context, GamePanel gamePanel, ImageButton ibPowerUp)	פעולה בונה למחלקה שמקבלת את הקונטקסט, פאנל המשחק, הפנייה למיקום תמונת הכח. גורמת לתמונה להיות לחיצה, בודקת האם השחקן בחר כח ומשנה את הביטמאפ של התמונה בהתאם, אם השחקן

	לא בחר כח, התמונה לא תיראה על המסך בזמן המשחק.
<code>public void setTimesFromType()</code>	מתחלת את זמן ההפעלה, וזמן ההמתנה של הכח לפי סוגו
<code>public void powerActivated(boolean activateOrDeactivate)</code>	לפי הבוליאן שהיא מקבלת כפרמטר, הפעולה מחליטה מה יקרה במשחק כאשר השחקן לוחץ על הכח, ומה יקרה כאשר הכח מפסיק להיות מופעל
<code>@Override</code> <code>public void onClick(View view)</code>	כשהתמונה נלחצית, כמות הפעמים שהמשחק יתחיל תתעדכן אם הכמות השתנתה, המשחק יבדוק אם האובייקטים שעל המסך נעים במהירות הלא התחלתית שלהם והבנוס שהשחקן בחר הוא 3, משום שפעולות אלו יגרמו לניגודים ביניהם, לשחקן יוצג הודעה שאומרת לו שזה לא אפשרי. אם זה לא המקרה, הכח יופעל, התמונה תהפוך לבלתי נראית ולא תהיה לחיצה והמשחק יתחיל את הספירה לאחר
<code>public void setPowerUpTextColor(boolean activateOrDeactivate)</code>	משנה את צבע ההטקסט לירוק או לבן, בהתאם לבוליאן שהיא מקבלת כפרמטר שמציין אם הכח פועל או בהמתנה
<code>public void startTime()</code>	פעולה רקורסיבית שמורידה את הזמן באחד כל שנייה ומציגה את הזמן כראוי, כך שיראו את הזמן שהכח פעיל, או שהוא בהמתנה, כשהזמן הוא 0 השחקן יראה ויוכל ללחוץ על התמונה שוב
<code>public void restartPowerUpTimer()</code>	מתחלת את שעון העצר

4.10.6.1. תרשים זרימה של המחלקה PowerUp



4.10.7. המחלקה GameSensors

המחלקה GameSensors היא המחלקה של חיישן האפליקציה. תפקידה הוא למצוא מה מידת הסיבוב של הטלפון בציר האיקס ולהעביר אותו לפאנל המשחק, כדי שהמשחק יוכל להשתמש בנתון הזה. המחלקה מיישמת את הממשק `EventListener`.

GameSensors – התכונות	
התכונה	תיאור התכונה
private final SensorManager sensorManager	באמצעות חיישן זה אפשר למצוא את מידת הטלפון של המכשיר
private float xRotation	מחזיק את מידת הסיבוב של הטלפון בציר האיקס

GameSensors – הפעולות	
הפעולה	תיאור הפעולה
public GameSensors(Context context)	פעולה בונה למחלקה שמשיגה את חיישן מד התאוצה באפליקציה, ומאתחלת את משתנה מידת הסיבוב.
public float getXRotation()	מחזירה את מידת הסיבוב של הטלפון
public void stopSensors()	מפסיקה את עבודת החיישן
public void onSensorChanged(SensorEvent sensorEvent)	משיגה את מידת הסיבוב של הטלפון. חלק מפעולות החובה של הממשק
public void onAccuracyChanged(Sensor sensor, int i)	חלק מפעולות החובה של הממשק

4.10.8. המחלקה SFX

המחלקה SFX היא המחלקה של הצלילים שהשחקן יכול לשמוע במהלך המשחק.

SFX – התכונות	
התכונה	תיאור התכונה
private final SoundPool soundPool	משתנה להפעלת צלילים
private final int death, explosion, bonus_pos, bonus_neg	משתנים שעליהם יטענו הצלילים, הצלילים הם מוות, פיצוץ, בונוס חיובי, ובונוס שלילי

SFX – הפעולות	
הפעולה	תיאור הפעולה
public SFX(Context context)	פעולה בונה למחלקה שמקבלת את הקונטקסט, מכינה את המשתנה להשמעת צלילים לפי ה-SDK של השחקן, וטוענת על המשתנים את הצלילים
public void playSound(String sound)	מפעילה צליל לפי המחרוזת שהיא מקבלת כפרמטר

4.10.9. המחלקה RectGame

המחלקה RectGame היא מחלקה אבסטרקטית של אובייקט המשחק. כל אובייקטי המשחק יורשים את מחלקה. במחלקה זו יש את התכונות של כל אובייקט בסיסי, כמו הציור/תמונה שלו, נקודת המשחק שלו, הרוחב, והגובה שלו.

RectGame – התכונות	
התכונה	תיאור התכונה
protected Context context	הקונטקסט של האובייקט
protected Bitmap bitmap	התמונה של האובייקט

protected PointGame pointGame	נקודת המשחק של האובייקט
protected int width, height	הגובה והרוחב של האובייקט
protected Rect rect	המלבן של האובייקט
protected Random rnd	משומש על מנת להגריל מספרים אקראיים לכל אובייקט
protected Paint paint	מחלקה של השקיפות והצבע של האובייקט

RectGame – הפעולות	
הפעולה	תיאור הפעולה
public RectGame(Context context, int bitmapID, int width, int height)	פעולה בונה של המחלקה שמאתחלת את התכונות שלה. הפעולה מקבלת כפרמטרים את הקונטקסט, מספר שמציין את תמונת, הרוחב וגובה של המלבן. הפעולה יוצרת ביטמאפ מהמספר שמציין אותו
public RectGame(Context context, String color, int width, int height)	פעולה בונה של המחלקה שמאתחלת את התכונות שלה. הפעולה מקבלת כפרמטרים את הקונטקסט, הצבע, הרוחב וגובה של המלבן. הפעולה יוצרת ביטמאפ מהמספר שמציין אותו
public void setRect(Rect rectangle)	מקבלת מלבן כפרמטר, שמה את המלבן על המסך לפי צדדי המלבן
public Rect getRectangle()	מחזירה את המלבן
public int getWidth()	מחזירה את רוחב המלבן
public void setWidth(int width)	מציבה את ערך הפרמטר ברוחב המלבן
public int getHeight()	מחזירה את אורך המלבן
public void setHeight(int height)	מציבה את ערך הפרמטר באורך המלבן
public PointGame getPointGame()	מחזירה את נקודת המשחק של המלבן
public void kill()	שמה את האובייקט בנקודה על המסך כך שלא יראו, הפעולה ממומשת על ידי אובייקטים שלא נעים כך שלא צריך לדאוג שהמלבן יחזור למסך
public void moveObst()	מזיזה את האובייקט לפי מהירותו
public boolean collision (Player p)	בודקת אם המלבן פגע בשחקן ומחזירה אמת אם כן
public abstract void draw(Canvas canvas)	פעולה אבסטרקטית ששמה את המלבן על המסך ומציירת אותו על הקנבס
public abstract void update()	פעולה אבסטרקטית שמעדכנת את האובייקט
public abstract void changeVelocities()	פעולה אבסטרקטית שמשנה את המהירות הוקטורית של המלבן בשני הצירים
public void changeSpeed(int type)	משנה את רכיב המהירות הסקלרי לפי הסוג שינוי שהיא מקבלת כפרמטר, 0- קפוא, 1-איטי, 2-מהיר, 3-רגיל. קוראת לפעולה שמשנה את המהירות הוקטורית לפי רכיב המהירות הסקלרי החדש

4.10.9.1 המחלקה PointGame

המחלקה PointGame היא מחלקה של נקודת המשחק. לכל אובייקט במשחק יש נקודה שעוזרת למקם את האובייקט במרחב. המחלקה יורשת את המחלקה Point.

PointGame – התכונות

התכונה	תיאור התכונה
protected float velX, velY	המהירות הוקטורית של הנקודה בציר האיקס והוואי
protected int speed	המהירות הסקלרית של הגוף
protected int initialSpeed	המהירות הסקלרית הראשונית שהנקודה קיבלה

PointGame – הפעולות	
הפעולה	תיאור הפעולה
public PointGame(int x, int y)	מאתחלת את מיקום הנקודה לפי הפרמטרים שהיא מקבלת, מאתחלת את המהירות הסקלרית לפי הרמת קושי שהשחקן בחר
public float getVelX()	מחזירה את המהירות הוקטורית האופקית
public void setVelX(float velX)	משנה את המהירות הוקטורית האופקית לפי הפרמטר
public float getVelY()	מחזירה את המהירות הוקטורית האנכית
public void setVelY(float velY)	משנה את המהירות הוקטורית האנכית לפי הפרמטר
public int getSpeed()	מחזירה את המהירות הסקלרית
public void setSpeed(int speed)	משנה את המהירות הסקלרית לפי הפרמטר
public void setSpeedFromDifficulty()	קובעת את המהירות ההתחלתית של הנקודה לפי רמת הקושי

4.10.9.2. המחלקה Player

המחלקה Player היא האובייקט שהשחקן שולט עליו. השחקן שולט על האובייקט דרך הפעולה onTouchEvent(MotionEvent e). המחלקה יורשת את המחלקה RectGame.

Player – הפעולות	
הפעולה	תיאור הפעולה
public Player(Context context, int bitmapID, int width, int height)	פעולה בונה של המחלקה שמקבלת כפרמטרים את הקונטקסט, מספר שמציין את תמונת המלבן, הרוחב וגובה של המלבן. הפעולה יוצרת ביטמאפ מהמספר שמציין אותו, מאתחלת את נקודת התחלתו, ומאפסת את מהירותו
@Override public void draw(Canvas canvas)	מציירת את האובייקט על הקנבס. מאפסת את רכיבי המהירות הוקטוריים כדי שאובייקט האלפא לא יושפע מהשחקן גם כשהוא לא זז
public void createBitmap()	יוצרת ביטמאפ חדש לשחקן לפי הרוחב וגובה חדשים כאשר גודל השחקן משתנה
@Override public void update()	שמה את מלבן השחקן על המסך

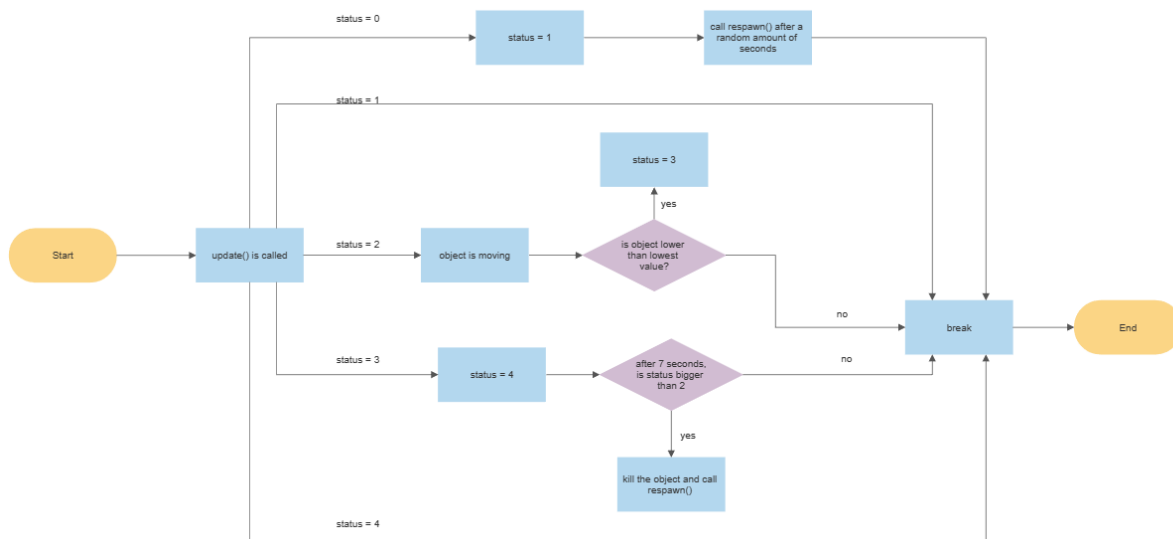
4.10.9.3. המחלקה BonusObj

המחלקה BonusObj היא אובייקט הבונוס, אובייקט זה מדי פעם יפול מהשמיים והשחקן יוכל לגעת בו כדי לקבל אחד מהבונוסים באקראיות. השחקן יכול לקבוע את התמונה של הבונוס בחנות המשחק. המחלקה יורשת מהמחלקה RectGame.

BonusObj – התכונות	
התכונה	תיאור התכונה
private int lowestY	הנקודה הכי נמוכה שהאובייקט יכול להגיע אליה
private int status	מסמן המצב שבו האובייקט נמצא. 0-מתחיל את ההאנדלר שמחשב את הזמן לפני שהאובייקט נוצר. 1-ממכה להיווצר. 2- זז מטה עד שמגיע לנקודה הכי נמוכה. 3-מעלה את הסטטוס ב1, מתחיל את ההאנדלר שממכה מספר שניות לפני שמחזיר את הסטטוס ל0. 4- משאיר את האובייקט במקום עד שהוא מופעל, או עד שעבר מספיק זמן ומחזיר את הסטטוס חזרה ל0
private Handler handler	משומש כדי להתחיל פעולות לאחר דיליי של מספר שניות

BonusObj – הפעולות	
הפעולה	תיאור הפעולה
public BonusObj(Context context, int bonusID, int width, int height)	פעולה בונה של המחלקה שמקבלת את הקונטקסט, מספר שמציין את ביטמאפ האובייקט, הרוחב והגובה שלו. הפעולה הופכת את מספר הביטמאפ לביטמאפ ומאתחלת את הסטטוס ל1
public void setStatus(int status)	משנה את הסטטוס למספר שהיא מקבלת כפרמטר
public void setRandomPoint()	משנה את ערכי הlowestY, x לערכים אקראיים ואת ערך הy לחלק העליון של המסך
public int setSleepTime()	מחזירה זמן אקראי לפני שהאובייקט ייווצר
@Override public void draw(Canvas canvas)	שמה את מלבן האובייקט על המסך ומציירת אותו על הקנבס אם הסטטוס בין 2 ל4
@Override public void update()	מזיזה את האובייקט לפי הסטטוס, ומשנה את הסטטוס בהתאם לתנאים
public void respawn()	קוראת לsetRandomPoint(), ומשנה את הסטטוס ל2 או 0 בהתאם לתנאי שבו האובייקט צריך להופיע מחדש

4.10.9.3.1 BonusObj תרשים זרימה של המחלקה



4.10.9.4 AlphaObj המחלקה

המחלקה AlphaObj היא מכשול במשחק. מכשול זה מוריד את השקיפות שלו כשהשחקן זז ומעלה את השקיפות כשהשחקן מפסיק לזוז. המחלקה יורשת מהמחלקה RectGame.

AlphaObj – התכונות	
תיאור הפעולה	הפעולה
המהירות ההתחלתית של האובייקט בציר הוואי	private final float initVelY

AlphaObj – הפעולות	
תיאור הפעולה	הפעולה
פעולה בונה של המחלקה, משנה את מיקום האובייקט על ציר האיקס באקראיות, מחליטה על המספר האקראי בין 1 ל-2 שישפיע על מהירות האובייקט. משנה את ערך שקיפותו ל-255 ואת מהירותו למהירות ההתחלתית	public AlphaObj(Context context, String color, int width, int height)
אם השחקן זז, שקיפות האובייקט יורדת עד למינימום של 0, ואם לא, מעלה את שקיפות האובייקט עד למקסימום של 255	public void changeAlpha()
שמה את מלבן האובייקט על המסך ומציירת אותו על הקנבס	@Override public void draw(Canvas canvas)
מזיזה את האובייקט לפי מהירותו, בודקת אם שקיפות האובייקט צריכה להשתנות	@Override public void update()
מחזיר את מהירות האובייקט למהירות המקורית	@Override public void changeVelocities()

4.10.9.5 MovingRectObj המחלקה

המחלקה MovingRectObj היא אחד משולשה אובייקטים אפשריים, הסוג הראשון נופל באלכסוניות ממעלה המסך מטה, השני נופל שיר מטה, והשלישי נע אופקית מקיר אחד לאחר. המחלקה באקראיות בוחרת איזה סוג אובייקט השחקן יפגוש כשנוצר מופע חדש של מחלקה זו. המחלקה יורשת את המחלקה RectGame.

MovingRectObj – התכונות	
התכונה	תיאור התכונה
private final int type	מוגדר באקראיות כדי לקבוע את סוג האובייקט. 0-אלכסוני, 1-אנכי, 2-אופקי
private final float rndFX = rnd.nextFloat()	משומש כדי לקבוע את המהירות האופקית של האובייקט כשהוא במצב של תנועה אלכסונית
private final float rndFY = 1 + rnd.nextFloat()	משומש כדי לקבוע את המהירות האנכית של האובייקט כשהוא במצב של תנועה אלכסונית
private boolean rightSide	משומש עבור האובייקט מסוג תנועה אלכסונית. בודק האם האובייקט נוצר בצידו הימני של המסך
private int horizontalMovement	מספר אקראי שקובע עם האובייקט מסוג תנועה אופקית ינוע ימינה או שמאלה

MovingRectObj – הפעולות	
הפעולה	תיאור הפעולה
public MovingRectObj(Context context, String color, int width, int height)	פעולה בונה של המחלקה שמקבלת כפרמטרים את הקונטקסט, הצבע, הרוחב וגובה של כל מלבן. הפעולה מגרילה באקראיות את סוג האובייקט וקובעת את מיקומו לפי הסוג, ובנוסף, קוראת לפעולה שקובעת את מהירות האובייקט לפי הסוג
public void setDiagonalVelocity()	מביאה לגוף מהירות בשני הכיוונים, אם האובייקט נוצר בצידו השמאלי של המסך, מהירותו הופקית תהייה ימינה, ואם נוצר בצידו הימני של המסך, מהירותו האופקית תהייה שמאלה
public void setVerticalVelocity()	מביאה לאובייקט מהירות מטה
public void setHorizontalVelocity()	מביאה לאובייקט מהירות אופקית לפי המשתנה horizontalMovement
@Override public void draw(Canvas canvas)	שמה את המלבן על המסך ומציירת אותו על הקנבס
@Override public void update()	מזיזה את האובייקט לפי מהירותו
@Override public void changeVelocities()	משנה את מהירות האובייקט באמצעות הפעולה שמתאימה לסוג האובייקט

4.10.9.6. המחלקה GapObj

המחלקה GapObj היא אובייקט שמורכב משני מלבנים שמתחילים בצד העליון של המסך ונעים מטה.
המחלקה יורשת את המחלקה RectGame.

GapObj – התכונות	
התכונה	תיאור התכונה
private final Rect rect1, rect2	שני המלבנים שעל המסך

GapObj – הפעולות	
הפעולה	תיאור הפעולה
public GapObj(Context context, String color, int width, int height)	פעולה בונה של המחלקה שמקבלת כפרמטרים את הקונטקסט, הצבע, הרוחב

	וגובה של כל מלבן. הפעולה שמה את האובייקט במקום אקראי במעלה המסך, מאפסת את המהירות האופקית ונותנת לו מהירות אנכית למטה
public void setRectangles (Rect rectangle1, Rect rectangle2)	מעדכנת את מיקום שני המלבנים
public boolean collision (Player p)	מחזירה אמת אם השחקן פוגע באחד מהמלבנים
@Override public void draw(Canvas canvas)	שמה את שני המלבנים על המסך ומציירת אותם על הקנבס
@Override public void update()	מזיז את המלבנים מטה
@Override public void changeVelocities()	משנה את מהירות האובייקט על פי הסקלר speed

4.10.9.7. WarningObj המחלקה

המחלקה WarningObj היא מחלקה של אובייקט שמופיע על המסך בהתחלה כסימן קריאה, שמתריע לשחקן על הסכנה, במצב הזה השחקן יכול לגעת האובייקט מבלי להפסל. לאחר שנייה וחצי, יופעל סאונד והאובייקט משנה את צורתו לגולגולת שהשחקן יפסיד אם יפגע בה. האובייקט נעלם כמה שניות לאחר מכן. המחלקה יורשת את המחלקה RectGame.

WarningObj – התכונות	
התכונה	תיאור התכונה
private Bitmap Warning	הביטמאפ של האזהרה שהשחקן מקבל
private int status = 0	שומר את המצב של האובייקט. 0-האובייקט נוצר ותמונתו היא סימן קריאה, מתחיל את ההאנדלר שיחכה שנייה וחצי ויעביר את הסטטוס ל-1. 1-האובייקט מראה גולגולת ויכול לפגוע בשחקן. 2-דואג שמקרה 0 או 1 לא יבוצעו
private boolean soundPlayed = false	מועבר לאמת כשהסאונד של האובייקט מופעל כדי שלא יופעל יותר מפעם אחת

WarningObj – הפעולות	
הפעולה	תיאור הפעולה
public RectGame(Context context, int bitmapID, int width, int height)	פעולה בונה של המחלקה שמאתחלת את התכונות שלה. הפעולה מקבלת כפרמטרים את הקונטקסט, מספר שמציין את תמונת, הרוחב וגובה של המלבן. הפעולה יוצרת ביטמאפ מהמספר שמציין אותו
@Override public void draw(Canvas canvas)	מציירת את האובייקט על הקנבס לפי התמונה שמתאימה לstatus, מפעילה את ההאנדלרים שמשנים את הסטטוס
@Override public void update()	פעולת חובה של המחלקה RectGame שאותה הוא יורש
@Override public void changeVelocities()	פעולת חובה של המחלקה RectGame שאותה הוא יורש

4.10.9.8. DownUpObj המחלקה

המחלקה DownUpObj היא אובייקט שמתחיל בצד העליון של המסך, נע מטה, פוגע ברצפה, ונע חזרה למעלה. המחלקה יורשת את המחלקה RectGame.

DownUpObj – התכונות	
התכונה	תיאור התכונה
private boolean touched bottom	שומר האם האובייקט פגע ברצפה, משומש כדי לקבוע את המהירות האנכית של האובייקט לאחר שמהירותו השתנתה

DownUpObj – הפעולות	
הפעולה	תיאור הפעולה
public DownUpObj(Context context, String color, int width, int height)	פעולה בונה של המחלקה שמקבלת כפרמטרים את הקונטקסט, הצבע, הרוחב וגובה של כל מלבן. הפעולה שמה את האובייקט במקום אקראי במעלה המסך, מאפסת את המהירות האופקית ונותנת לו מהירות אנכית למטה
@Override public void draw(Canvas canvas)	שמה את המלבן על המסך ומציירת אותו על הקנבס
@Override public void update()	מזיזה את האובייקט למטה. בודקת האם האובייקט פוגע ברצפה והופכת את מהירותו האנכית ואת הבוליאן touchedBottom כשהאובייקט פוגע ברצפה
@Override public void changeVelocities()	משנה את מהירות האובייקט על פי הסקלר speed והכיוון שנקבע על פי touchedBottom

4.10.9.9 MovingGapObj המחלקה

המחלקה MovingGapObj היא אובייקט שמורכב משני מלבנים שמתחילים בצד העליון של המסך, נעים מטה, וזזים ימינה ושמאלה כדי שהשחקן יצטרך לדאוג שהוא עובר ברווח ביניהם בזמן שהרווח הזה זז. המחלקה יורשת את המחלקה GapObj, שבה מוכרזים שני המלבנים והפעולות שדואגות שהם ייוצרו על המסך ויזזו למטה.

MovingGapObj – התכונות	
התכונה	תיאור התכונה
private int movingDirection	שומר על הכיוון של האובייקט. 0- המלבנים ינועו ימינה. 1- המלבני ינועו שמאלה

MovingGapObj – הפעולות	
הפעולה	תיאור הפעולה
public DownUpObj(Context context, String color, int width, int height)	פעולה בונה של המחלקה שמקבלת כפרמטרים את הקונטקסט, הצבע, הרוחב וגובה של כל מלבן. הפעולה שמה את האובייקט במקום אקראי במעלה המסך, מאפסת את המהירות האופקית ונותנת לו מהירות אנכית למטה
@Override public void update()	קוראת לפעולה update() של המחלקה GapObj שאותה היא יורשת, נוסף על כך, הפעולה בודקת אם אחד המלבנים במרחק

	של 300 פיקסלים מהקיר ומשנה את כיוון תנועת המלבנים אם כן
@Override public void changeVelocities()	משנה את מהירות האובייקטים באמצעות הסקלר speed.movingDirection משומש כדי לדעת מה כיוון התנועה האופקי

4.10.9.10. המחלקה CrossingObj

המחלקה BonusObj היא שני אובייקטים שנעים מטה, האובייקטים נעים בסימטריות כך שכל אחד מהם נע ימינה ומאלה והם נפגשים במרכז אחרי שהם עוברים חצי מסך פעמיים. המחלקה יורשת את המחלקה RectGame.

CrossingObj – התכונות	
התכונה	תיאור התכונה
private final Rect rect1, rect2	שני המלבנים שעל המסך
private int movingDirection = 0	שומר על הכיוון של שני המלבנים, מאותחל כ-0. מלבן 1 ינוע ימינה, ובמלבן 2 שמאלה. -1 מלבן 1 ינוע שמאלה ומלבן 2 ינוע ימינה

CrossingObj – הפעולות	
הפעולה	תיאור הפעולה
public CrossingObj(Context context, String color, int width, int height)	פעולה בונה של המחלקה שמקבלת כפרמטרים את הקונטקסט, הצבע, הרוחב וגובה של כל מלבן
public void setRectangles (Rect rectangle1, Rect rectangle2)	מעדכנת את מיקום שני המלבנים
@Override public boolean collision(Player p)	מחזירה אמת אם השחקן פוגע באחד מהמלבנים
@Override public void draw(Canvas canvas)	שמה את שני המלבנים על המסך ומציירת אותם על הקנבס
@Override public void update()	מעדכנת את movingDirection כשמלבן 1 פוגע בקירות ומזיזה את שני האובייקטים למטה ולצדדים בהתאם movingDirection
@Override public void changeVelocities()	מחזירה את מהירות האובייקטים למהירות המקורית. שומר באמצעות movingDirection שהכיוון האופקי של כל מלבן יהיה כמו הכיוון האופקי שלו לפני שהמהירות השתנתה

4.10.9.11. המחלקה CrossingSingleObj

המחלקה CrossingSingleObj היא אובייקט שנכנס מהצד העליון של המסך, כשהאובייקט פוגע בקירות המהירות האופקית שלו מתהפכת. המחלקה יורשת את המחלקה RectGame, שבודקת פגיעה בקירות, לכן מחלקה זו בודקת רק פגיעה ברצפה. המחלקה יורשת את המחלקה RectGame.

CrossingSingleObj – התכונות	
התכונה	תיאור התכונה
private int movingDirection	שומר על הכיוון של האובייקט. -0 מלבן 1 ינוע ימינה, ובמלבן 2 שמאלה. -1 מלבן 1 ינוע שמאלה ומלבן 2 ינוע ימינה

private final float makeRandomVelX	מהירות אקראית בציר ה-x
------------------------------------	------------------------

CrossingSingleObj – הפעולות	
הפעולה	תיאור הפעולה
public CrossingSingleObj(Context context, String color, int width, int height)	פעולה בונה של המחלקה שמקבלת כפרמטרים את הקונטקסט, הצבע, הרוחב וגובה של כל מלבן. הפעולה מגרילה מספר אקראי בשביל המהירות של האובייקט בציר ה-x, ומחליטה באקראיות על כיוון התנועה האופקי ההתחלתי שלו
@Override public void draw(Canvas canvas)	שם את האובייקט על המסך, ומצייר אותו על הקנבס
@Override public void update()	בודק אם האובייקט פגע בקיר והכיוון האופקי שלו צריך להשתנות. מזיז את האובייקט
@Override public void changeVelocities()	משנה את המהירות האופקית הוקטורית לפי סקלר המהירות של האובייקט

4.10.9.12. המחלקה BounceObj

המחלקה BonusObj היא אובייקט שנכנס מהצד העליון של המסך, כשהאובייקט פוגע בקירות המהירות האופקיות שלו מתהפכת, וכשהוא פוגע ברצפה המהירות האנכית מתהפכת. המחלקה יורשת את המחלקה CrossingSingleObj, שבודקת פגיעה בקירות, לכן מחלקה זו בודקת רק פגיעה ברצפה.

BounceObj – התכונות	
התכונה	תיאור התכונה
private boolean touchedGround	שומר האם האובייקט פגע ברצפה, משומש כדי לקבוע את המהירות האנכית של האובייקט לאחר שמהירותו השתנתה

BounceObj – הפעולות	
הפעולה	תיאור הפעולה
@Override public void update()	משנה את מהירות האנכית ואת הבוליאן שבודק את הפגיעה כאשר האובייקט פוגע ברצפה. בנוסף, קוראת לפעולה super.update() כדי לבדוק אם הגוף פגיעה בקירות ולגרום לגוף לנוע כמו המחלקה שהוא יורש ממנה.
@Override public void changeVelocities()	קוראת לפעולה super.changeVelocities() כדי לקבוע את המהירות האופקית המחלקה שהוא יורש ממנה. לפי touchedBottom, משנה את המהירות האנכית של האובייקט
public void setSpeedFromDifficulty()	קובעת את המהירות ההתחלתית של הנקודה לפי רמת הקושי

4.10.9.13. המחלקה PositionSensorObj

המחלקה PositionSensorObj היא אובייקט שמתחיל בצד העליון של המסך, נע מטה, אבל נוסף על כך, הוא נע ימינה ושמאלה בהתאם לתנועת המכשיר שעליו השחקן משחק. האובייקט מקבל את המידע הזה

באמצעות המחלקה GameSensors שנמצאת באקטיביטי של המשחק GameScreenActivity. המחלקה יורשת את המחלקה RectGame.

PositionSensorObj – הפעולות	
הפעולה	תיאור הפעולה
public MovingRectObj(Context context, String color, int width, int height)	פעולה בונה של המחלקה שמקבלת כפרמטרים את הקונטקסט, הצבע, הרוחב וגובה של כל מלבן. הפעולה מגרילה באקראיות את סוג האובייקט וקובעת את מיקומו לפי הסוג, ובנוסף, שמה את האובייקט בצד העליון של המסך במקום אקראי בציר האיקס
@Override public void draw(Canvas canvas)	שמה את המלבן על המסך ומציירת אותו על הקנבס
@Override public void update()	מזיזה את האובייקט מטה ובציר האיקס לפי תנועת המכשיר, שומרת שהאובייקט לא יצא מגבולות המסך
@Override public void changeVelocities()	משנה את מהירות האובייקט האנכית

4.11. תיקיית Resources

4.11.1 Layouts

שם ה-Layout	Activity אליו הוא משתייך	הסבר
activity_forgot	ForgotActivity	מסך איפוס הסיסמא
activity_game_screen	GameScreenActivity	מסך המשחק
activity_hub	HubActivity	מסך ראשי
activity_level_select	LevelSelectActivity	מסך בחירת השלב
activity_login	LoginActivity	מסך ההתחברות
activity_options	OptionsActivity	מסך האפשרויות
activity_ranikings	RankingsActivity	מסך טבלת השיאים
activity_shop	ShopActivity	מסך החנות
activity_sign_up	SignUpActivity	מסך ההרשמה
activity_splash_screen	SplashScreenActivity	מסך הכניסה
dialog_boss_level_complete	GameScreenActivity	דיאלוג לאחר השלמת שלב בוס
dialog_demo_game_over	GameScreenActivity	דיאלוג לאחר הפסד בשלב הדמו
dialog_difficulty	LevelSelectActivity	דיאלוג שינוי דרגת הקושי
dialog_game_over	GameScreenActivity	דיאלוג לאחר הפסד בשלב רגיל או שלב בוס
dialog_level_info	LevelSelectActivity	דיאלוג לקבלת מידע על שלב
dialog_obst_info	LevelSelectActivity	דיאלוג לקבלת מידע על האובייקטים במשחק
dialog_purchase_confirm	ShopActivity	דיאלוג לרכישת פריט
dialog_shop_info	ShopActivity	דיאלוג למידע על סוגי הפריטים בחנות
fragment_inventory	ShopActivity	פרגמנט רשימת הפריטים
fragment_preview	ShopActivity	פרגמנט תצוגה מקדימה
fragment_shop_buy	ShopActivity	פרגמנט לרכישת פריט לפי סוג
shop_app_bar	ShopActivity	הסרגל העליון של החנות
shop_content	ShopActivity	משומש עבור הסרגל העליון של החנות
shop_layout	ShopActivity	לייאוט לפריט בודד בחנות
spinner_dropdown_item	RankingsActivity	תיאור של הפריטים בספינר
spinner_selected_item	RankingsActivity	תיאור של הפריט הנבחר בספינר

Drawables 4.11.2

תמונות של הלוגו של המשחק:










סמלים של החנות:

symbol_shop_icon	symbol_shop_inventory	symbol_shop_preview	side_nav_bar
symbol_shop_tick	symbol_shop_power_up	symbol_shop_bonus	symbol_shop_background





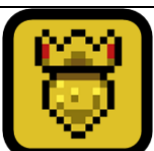



תמונות של דמויות שהשחקן יכול לבחור:

icon5	icon4	icon3	icon2	icon1
icon10	icon9	icon8	icon7	icon6




תמונות של הרקע שהשחקן יכול לבחור:

background4	background3	background2	background1
			
	background7	background6	background5
			






תמונות של הבונוסים שהשחקן יכול לבחור:

bonus4	bonus3	bonus2	bonus1
			
bonus8	bonus7	bonus6	bonus5
			


תמונות של הכוחות שהשחקן יכול לבחור:

power_up3	power_up2	power_up1
		

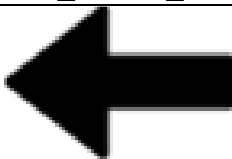



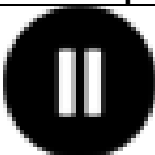


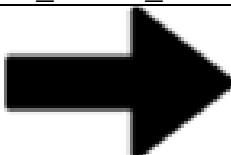
תבניות של רקעים למסכים וכפתורים באפליקציה:

button_red	button_blue	button_black	border_screen	border_hub
				

תמונות של אובייקט האזהרה:

warning_warn	warning_active
	

תמונות של סמלים נוספים שמשומשים באפליקציה:

level_select_back	info	ic_baseline_wifi_off_24	go_back
			
pause	locked_level	loading (gif)	level_select_forward
			

```

<scale
    android:duration="3000"
    android:fromXScale="0"
    android:toXScale="1"
    android:fromYScale="0"
    android:toYScale="1"
    android:pivotY="50%"
    android:pivotX="50%"/>
<alpha
    android:fromAlpha="0.0"
    android:toAlpha="1.0"
    android:duration="3000"/>

```

Anim 4.11.3

הקובץ anim_splash.xml הוא הקובץ של אנימצית הפתיחה של המשחק גורם ללוגו המשחק לגדול ולהיות פחות שקוף במשך שלוש שניות.

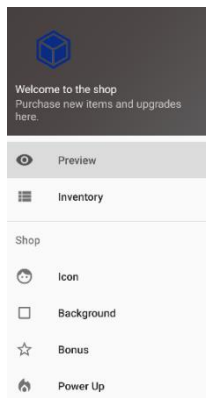
Fonts 4.11.4

במשחק יש שימוש בשלושה גופנים מותאמים אישית:

ahrbnd.ttf

berlin_sans_fb_demi_bold.ttf

franklin.ttf



Menu 4.11.5

באפליקציה יש תפריט שמושמש במסך החנות. תפריט זה הוא תפריט מסוג מגירה נפתחת. האפשרויות בתפריט זה הם תצוגה מקדימה, רשימת פריטים, וחנויות לדמות, לרקע, לבונוס, ולכח.

Mipmap 4.11.6



האפליקציה משתמשת בתמונה game_logo_cube שבתיקיית המיפמאפ כדי להציג את הלוגו של המשחק בתור לוגו האפליקציה.

Raws 4.11.7

במשחק יש 4 צלילים ויכולים להשמע במהלך המשחק דרך המחלקה SFX.

שם הצליל	תיאור
bonus_neg	מושמע כאשר השחקן משיג את אובייקט הבונוס ומקבל בונוס שלילי
bonus_pos	מושמע כאשר השחקן משיג את אובייקט הבונוס ומקבל בונוס חיובי
death	מושמע כאשר השחקן נפסל
explosion	מושמע כאשר אובייקט האזהרה מופעל

4.11.8. בסיס הנתונים Firebase

כשהייתי צריך לקבוע את סוג בסיס הנתונים שישימש בפרויקט שלי, הייתה לי את האפשרות לבחור ב-Firebase או ב-SQLite. בחרתי להשתמש בפיירבייס מכמה סיבות. ראשית, Firebase הוא מסד נתונים מבוסס ענן, כלומר ניתן לגשת אליו מכל מקום עם חיבור לאינטרנט. זה מקל על הניהול וההרחבה ככל שהפרויקט גדל, מכיוון שהנתונים אינם קשורים למכשיר או למיקום ספציפי. בנוסף, Firebase מספק מגוון תכונות אחרות מעבר לאחסון נתונים בלבד, כגון סנכרון בזמן אמת, אימות ואירוח. תכונות אלו יכולות לפשט מאוד את הפיתוח והתחזוקה, מכיוון שמפתחים אינם צריכים לבנות ולנהל את הרכיבים הללו בעצמם. לבסוף, Firebase מספקת אמצעי אבטחה חזקים, המבטיחים שהנתונים מוגנים מפני גישה או מניפולציה בלתי מורשית. בסך הכל, שימוש ב-Firebase יכול לייעל את תהליך הפיתוח, לשפר את יכולת ההרחבה והאבטחה ולספק תכונות נוספות מעבר לאחסון נתונים בלבד.

FIREBASE:

הסבר על הכלים בהם השתמשתי על מנת לאחסן את המידע באפליקציה:

Authentication Firebase:

שמירת הנתונים של המשתמש נעשית על ידי שימוש ב-Firebase, הרשמה והתחברות נעשים על ידי שימוש ב-Firebase Authentication באמצעות הזנת שם משתמש, מייל וסיסמה על ידי המשתמש במסכים המתאימים (LoginActivity | SignUpActivity).

Search by email address, phone number, or user UID					Add user		
Identifier	Providers	Created ↓	Signed In	User UID			
e@gmail.com	📧	Dec 30, 2022	Dec 30, 2022	qokLiQwyWEakKkOLrAq7sclHFCG3			
d@gmail.com	📧	Dec 30, 2022	Dec 30, 2022	3jyltpPINaWohDaKq2BQJQhvxDS2			
c@gmail.com	📧	Dec 30, 2022	Dec 30, 2022	i4Yw5dOgRDZpzyDgzzOxejE3RVV2			
b@gmail.com	📧	Dec 30, 2022	Dec 30, 2022	4zsalk70JmPvOk7IYdvXL08i2653			
a@gmail.com	📧	Dec 19, 2022	Dec 30, 2022	glxEXfv5G2S4A8Pur6kd0EtuChC3			
Rows per page: 50					1 – 5 of 5	<	>

Realtime Database:



























שמירת הנתונים של המשתמש (מלבד הסיסמה) נעשית על ידי שימוש ב-Realtime Database. לאחר ההרשמה, שאר הנתונים נשמרים ב-Firebase, בעזרת מחלקת User הכוללת את מפתח המשתמש, שעוזר לעדכן את נתוני המשתמש, UID שנוצר דרך ה-Authentication. השם והמייל של המשתמש, כמות הפעמים שהמשתמש מת במשחק, כמות המטבעות של המשתמש, דרגת הקושי שבחר המשתמש (מאותחל בתור "קל"), משתנים בוליאניים שהמשתמש בוחר בשביל ההגדרות של המשחק, שהם מוזיקה, סאונד אפקטים (שניהם מאותחלים כאמת), והצגת FPS (מאותחל כשקר). רשימה של שיאי הנקודות של המשתמש בעשרים השלבים הרגילים, שתי רשימות של חמשת שלבי הבוס במשחק, אחד בשביל שיא הזמן של המשתמש והשני בשביל לדעת האם השחקן סיים את השלב. ארבע רשימות של משתנים בוליאניים בשביל לדעת האם השחקן רכש את הפריט עבור כל סוג פריט בנפרד, משתנה מספרי עבור כל פריט שהשחקן בחר בו, ומשתנה מספרי שמראה את סך הפריטים שהמשתמש רכש.

תמונות מהפירבויס דטאבייס:

▼ — hsl					
0: 0	▼ — bestTime				coins: 0
1: 225000	0: ""				deaths: 30
2: 37500	1: "0:00"				difficulty: 0
3: 37500	2: "2:14"				itemsOwned: 13
4: 97500	3: "1:44"				key: "-NOZuMhM-8EuZXp_Nygz"
5: 0	4: "2:17"				mail: "a@gmail.com"
6: 0	5: "1:25"				optFPS: true
7: 0	▼ — levelComplete				optMusic: true
8: 0	0: true				optSoundEffects: true
9: 0	1: true				selectedBackground: 3
10: 0	2: false				selectedBonus: 3
11: 0	3: false				selectedIcon: 8
12: 0	4: false				selectedPowerUp: 2
13: 0	5: false				uid: "3dJxBDWbaUncDWIs2hi5uvAgOk1"
14: 0					userName: "aaa"
15: 0					
16: 0					
17: 0	▼ — iconOwned	▼ — bonusOwned	▼ — backgroundOwned		
18: 0	0: true	0: true	0: true		
19: 0	1: true	1: false	1: true		
20: 0	2: true	2: false	2: false		
21: 0	3: true	3: true	3: true		
22: 0	4: true	4: true	4: false		
23: 0	5: true	5: false	5: false		
24: 0	6: false	6: false	6: true	▼ — powerUpOwned	
25: 0	7: false	7: false		0: true	
	8: true			1: false	
	9: false			2: true	
				3: true	

Firestore Storage:

Firestore Storage הוא הדרך שבה השתמשתי על מנת אחסון מוזיקה.

gs://aplic-1.appspot.com > songs Upload file				
<input type="checkbox"/>	Name	Size	Type	Last modified
<input type="checkbox"/>	 BetrayalOfFear.mp3	4.21 MB	audio/mpeg	Dec 29, 2022
<input type="checkbox"/>	 BreadySteadyGo.ogg	6.78 MB	audio/ogg	Dec 29, 2022
<input type="checkbox"/>	 CantGoBack.ogg	6.23 MB	audio/ogg	Dec 29, 2022
<input type="checkbox"/>	 DarkEnd.mp3	4.14 MB	audio/mpeg	Dec 29, 2022
<input type="checkbox"/>	 Dimension.mp3	2.93 MB	audio/mpeg	Dec 29, 2022
<input type="checkbox"/>	 GhostHouse.mp3	3.25 MB	audio/mpeg	Dec 29, 2022
<input type="checkbox"/>	 Goldenvengeance.ogg	5.71 MB	audio/ogg	Dec 29, 2022
<input type="checkbox"/>	 ItMeansEverything.ogg	4.19 MB	audio/ogg	Dec 29, 2022
<input type="checkbox"/>	 Kenos.mp3	3.65 MB	audio/mpeg	Dec 29, 2022
<input type="checkbox"/>	 LingeringLightOfTheSettingSun.mp3	5.12 MB	audio/mpeg	Dec 29, 2022
<input type="checkbox"/>	 LostMemories.mp3	3.11 MB	audio/mpeg	Dec 29, 2022
<input type="checkbox"/>	 METALOVANIA.mp3	3.62 MB	audio/mpeg	Dec 29, 2022
<input type="checkbox"/>	 NineCircles.mp3	3.11 MB	audio/mpeg	Dec 29, 2022
<input type="checkbox"/>	 PartytArigang.mp3	2.71 MB	audio/mpeg	Dec 29, 2022
<input type="checkbox"/>	 PeerGynt.mp3	2.14 MB	audio/mpeg	Dec 30, 2022
<input type="checkbox"/>	 PhobiaWave.mp3	1.3 MB	audio/mpeg	Dec 29, 2022
<input type="checkbox"/>	 SonicBlaster.mp3	2.78 MB	audio/mpeg	Dec 29, 2022
<input type="checkbox"/>	 StayInsideMe.mp3	1.6 MB	audio/mpeg	Dec 29, 2022
<input type="checkbox"/>	 Swirly1000x.ogg	3.02 MB	audio/ogg	Dec 29, 2022
<input type="checkbox"/>	 TeeHeeTime.ogg	6.08 MB	audio/ogg	Dec 29, 2022
<input type="checkbox"/>	 TheVengeanceOfThoseForgottenInDarkness.ogg	3.24 MB	audio/ogg	Dec 29, 2022
<input type="checkbox"/>	 TheoryOfEverything3(DJNate).mp3	2.31 MB	audio/mpeg	Dec 29, 2022
<input type="checkbox"/>	 TheoryOfEverything3(domyeah).mp3	4.18 MB	audio/mpeg	Dec 29, 2022
<input type="checkbox"/>	 VsSusie.mp3	2.22 MB	audio/mpeg	Dec 29, 2022
<input type="checkbox"/>	 WorldsEndValentine.ogg	7.66 MB	audio/ogg	Dec 29, 2022
<input type="checkbox"/>	 YouWereWrongGoBack.ogg	3.94 MB	audio/ogg	Dec 29, 2022

4.11.9 הרשאות

האפליקציה שלי יש הרשאה להרעיד את הטלפון באמצעות ההרשאה שכתובה במניפסט.

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

5. מדריך למשתמש

התכנית משתמשת ב-API ברמה 30 עבור ה-SDK המינימלי, API ברמה 32 עבור SDK מיועד. התכנית נכתבה בסביבת Android Studio בגרסת אנדרואיד 11 ורצה גם על אנדרואיד 13. בוצעה הרצה על אמולטור Pixel 5 API 30, וגם על מכשיר סלולרי Samsung SM-A525F.

האפליקצייה Rush&Dash היא משחק שבו השחקן שולט על דמות שצריכה להתחמק ממכשולים. ככל שעולים בשלב השחקן יצטרך להתחמק ממכשולים מסובכים יותר מהשלים הקודמים.

מטרתו של השחקן היא לסיים את השלב ולעבור לשלב הבא עד שלב 25 שהוא השלב האחרון.

סיום השלב מותנה בסוג השלב.

יש שני סוגי שלבים, שלב רגיל שצריך להגיע בו למספר נקודות מסויים כדי לפתוח את השלב הבא, ושלב בוס שמופיע בסוף העולם (שלב 5, 10, 15, 20, 25), ובו השחקן יצטרך להתחמק מהמכשולים ולנסות לשרוד עד שהשעון שסופר את הזמן יגיע לאפס כדי לפתוח את העולם הבא. לכל שלב במשחק יש שיר שונה שמלווה אותו.

בכל מסך משחק ניתן למצוא בונסים שונים שיכולים לעזור לשחקן או לעקב אותו, לדוגמא: ישנם בונסים שיעזרו לשחקן, כמו הקטנת הדמות, כדי שיהיה לו קל יותר להתחמק ממכשולים, או הקפאת כל המכשולים שעל המסך. לעומת זאת, יש בונסים שיקשו על השחקן, כמו בונס שיגרום לאובייקטים לנוע מהר יותר, או שיגדילו את הדמות, כדי שיהיה לו קשה יותר להתחמק ממכשולים.

במסך הראשי של המשחק השחקן יכול לעבור לחלקים שונים באפליקציה: הגדרות, חנות, טבלת שיאים, בחירת שלב, והתנתקות מהמשחק וחזרה למסך ההתחברות.

במשחק יש מסך הגדרות.

במסך זה השחקן יכול לשנות הגדרות שונות של המשחק. אחת ההגדרות תקבע האם השחקן ישמע את השיר של השלב. הגדרה אחרת תקבע האם השחקן ישמע את אפקטי הסאונד של המשחק. ההגדרה האחרונה תקבע האם השחקן יראה את כמות הפריימים לשנייה של המשחק בזמן שהוא משחק.

במשחק יש חנות, ובה השחקן יכול לקנות פריטים לפי כמות הכסף שהוא משיג לאורך המשחק. הפריטים אותם הוא יכול לקנות הם לדוגמא, צורות שונות של דמויות, רקעים שונים, צורות שונות של בונסים, וכוחות שיעזרו לשחקן בשלבים השונים.

לפני תחילת המשחק, יש אפשרות לשחק בשלב הדמו של המשחק דרך מסך ההתחברות של האפליקצייה, כדי שהמשתמש יוכל לדעת אם המשחק מוצא חן בעיניו, ואם כן הוא יכול להתחיל ישר דרך מסך ההרשמה.

למשתמש מוצגות הודעות במהלך השימוש באפליקציה באמצעות השימוש ב-Toast. יש מספר מקרים שבהם יופיע הודעה למשתמש:

- כשהמשתמש לא מחובר לאינטרנט והאפליקציה צריכה לעדכן את הפיירבייס או לפתוח מסך שתלוי באינטרנט.
- כשהקלט שהמשתמש מכניס לתיבות הטקסט לא מתאים להשלמת הפעולה. לדוגמה, תיבות טקסט ריקות, מייל שלא נכתב בתור מבנה של מייל, שם משתמש שכבר קיים במערכת.
- במסכי ההתחברות - כשהתחברות למערכת הצליחה או נכשלה, כנשלח למשתמש מייל שינוי סיסמא בהצלחה, או כשנוצר משתמש חדש בהצלחה.
- כשהאפשרויות נשמרות במסך האפשרויות.
- כשהמשתמש מנסה להפעיל את הכח של האטת מהירות המכשולים בזמן שהמכשולים קפואים.
- כשלמשתמש אין מספיק מטבעות כדי לרכוש פריט.
- כשהמשתמש רוכש פריט בהצלחה.

יש שימוש ב-Alert Dialog כדי לוודא שהשחקן בטוח שהוא רוצה לאפס את ההתקדמות שלו דרך מסך האפשרויות. אם השחקן בוחר לאפס דרך הדיאלוג, יוצג לו הודעה שתגיד לו שהאיפוס הושלם.

6. סיכום אישי / רפלקציה

כאשר התחלתי לעבוד על האפליקציה, היעד ההתחלתי שלי היה ליצור משחק מעניין ומהנה.

לפי דעתי, היעד הושג ואף יותר מכך בגלל שכלל שהזמן עבר הוספתי למשחק עוד תוכן שלעיתים היה לי קשה לבנות אותם, אך בעזרת המורה למחשבים, הצלחתי.

העבודה על הפרויקט פיתחה את יכולותיי גם בתחום התכנות וגם מבחינה אישית.

למדתי לתכנת אפליקציית משחקים שלמה מההתחלה ועד הסוף ולמדתי להבין דרכה שגם דברים פשוטים ביותר וטריויאליים ביותר יכולים להיות קשים, אך למרות זאת הצלחתי להתגבר על כך ולפתור את הבעיות.

הפרויקט גרם לי להבין שכדי להגיע למטרה הסופית צריך להשקיע רבות לא לוותר ולכל בעיה קיים פתרון ואנחנו רק צריכים לחתור ולמצוא אותה ולהיעזר אם צריך באנשים אחרים.

הפרויקט גרם לי ללמוד לארגן טוב יותר את המחשבות והרעיונות שלי ולממשם, גם אם בהתחלה זה נראה לי כבלתי אפשרי, ישבתי ולמדתי עד שהצלחתי לעשות את הדברים כפי שחשבתי עליהם בתחילה.

העבודה על הפרויקט לקחה ממני מאמצים פיזיים ונפשיים רבים, בעיקר הזמן הרב שהשקעתי בפרויקט זה, למרות זאת, נהנתי מבניית האפליקציה, ובעיקר נהנתי לראות שכל מה שרציתי שיהיה במשחק בסופו של דבר הצליח.

הייתי צריך ללמוד לנהל את הזמן שלי בצורה טובה כדי שאוכל להשקיע שעות רבות בפרויקט זה, אך גם ללמוד מקצועות אחרים, להבחין בהם בהצלחה, וגם להיות עם חברים ולהתעסק בתחביבים שלי.

מבחינת התכנות, למדתי מושגים רבים, שמעולם לא חשבתי שאדע אותם, למדתי כיצד להשתמש בהם ולהביא אותם לידי ביטוי באפליקציית המשחק. אני אוכל לקחת את הכלים האלה להמשך כשאצטרך לתכנת משחקים ואפליקציות אחרות.

לפי דעתי, ניתן לבצע פיתוח עתידי למשחק שלי, על ידי הוספת שלבים חדשים עם אובייקטים חדשים, או הוספת פריטים חדשים בחנות שישפיעו יותר על חווית המשתמש.

בסופו של דבר, אני יכול להשוות את הפרויקט לטיפוס על הר. העלייה קשה, מפרכת ולוקחת שעות, ימים ואף חודשים רבים אך, בסיכומי של עניין כשרואים את התוצאה הסופית אפשר לנשום ולהבין שהתוצאה שווה את המאמץ.

7. ביבליוגרפיה

רשימת מקורות שעזרו לי במהלך הפרויקט:

https://www.youtube.com/playlist?list=PL2EfDMM6n_LYJdzaOQ5jZZ3Dj5L4tbAuM

https://www.youtube.com/playlist?list=PL2xjPbQaM7JZ_FmXwTAesiAciHEPIGmiW

https://firebase.google.com/docs/auth/android/manage-users#update_a_users_profile

<https://stackoverflow.com/questions/3875184/cant-create-handler-inside-thread-that-has-not-called-looper-prepare/16886486#16886486>

8. נספחים

- ```

public class ForgotActivity extends AppCompatActivity implements View.OnClickListener,
ValueEventListener {
 private EditText etName, etMail;
 private String email;
 private Button btChange;
 private ImageButton ibBack;
 private FirebaseAuth auth;
 private DatabaseReference userRef;
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_forgot);
 auth = FirebaseAuth.getInstance();
 userRef = FirebaseDatabase.getInstance().getReference("Users");
 etName = findViewById(R.id.etForgotName); etMail = findViewById(R.id.etForgotMail);
 btChange = findViewById(R.id.btChange);
 ibBack = findViewById(R.id.ibBack);
 btChange.setOnClickListener(this);
 ibBack.setOnClickListener(this);
 }
 @Override
 public void onClick(View view) {
 if (view == btChange)
 {
 startService(new Intent(this, VibrationService.class));
 if (isConnectedToInternet(this))
 userRef.addValueEventListener(this);
 else
 Toast.makeText(this, "Please connect to the internet!",
Toast.LENGTH_SHORT).show();
 }
 else if (view == ibBack) {
 finish();
 }
 }
 @Override
 public void onDataChange(@NonNull DataSnapshot snapshot) {
 userRef.removeEventListener(this);
 String userName = etName.getText().toString();
 email = etMail.getText().toString();

 // looks for the user and sends an email to the correct account
 for (DataSnapshot data : snapshot.getChildren()) {
 User tempUser = data.getValue(User.class);
 assert tempUser != null;
 Log.i(TAG, "onDataChange: " + tempUser.mail+" "+email+" "+tempUser.userName+"
"+userName);
 if (Objects.equals(tempUser.mail, email)) {
 if (Objects.equals(tempUser.userName, userName)) {
 auth.sendPasswordResetEmail(email)
 .addOnCompleteListener(task -> {
 if (task.isSuccessful()) {
 Toast.makeText(ForgotActivity.this, "Email sent to " + email,
Toast.LENGTH_SHORT).show();

```



```

 }
 }); finish();
 return;
}
}
}
 Toast.makeText(this, "Unable to find your account, please verify your information",
Toast.LENGTH_SHORT).show();
}
@Override
public void onCancelled(@NonNull DatabaseError error) {
}
}

```

- ```

public class GameScreenActivity extends Activity implements View.OnClickListener {
    private TextView tvLoading, tvFPS;
    private GamePanel gamePanel;
    private FrameLayout frameLayout;
    private ImageView ivConnection;
    private ImageButton ibBack;
    private int levelID;
    private MediaPlayer mediaPlayer;
    private int[] iconImages, backgroundImages, bonusImages, powerUpImages;
    // saves the retry count so the power up won't save after a game ends
    private int retryCount = 0;
    private final BroadcastReceiver connectivityChangeReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            Log.i(TAG, "onReceive: Connection changed");
            boolean isConnected = isConnectedToInternet(context);
            changeConnectionErrorVisibility(isConnected);
        }
    };
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_game_screen);
        levelID = getIntent().getIntExtra("LevelID", 0);
        Log.i(TAG, "onCreate: LevelID " + levelID);
        tvLoading = findViewById(R.id.tvLoading);
        tvFPS = findViewById(R.id.tvFPS);
        ivConnection = findViewById(R.id.ivConnection);
        ibBack = findViewById(R.id.ibBack);
        ibBack.setOnClickListener(this);
        // get images from HubActivity
        Bundle bitmapArrays = getIntent().getExtras();
        iconImages = bitmapArrays.getIntArray("icon_images");
        backgroundImages = bitmapArrays.getIntArray("background_images");
        bonusImages = bitmapArrays.getIntArray("bonus_images");
        powerUpImages = bitmapArrays.getIntArray("power_up_images");
        if (user.optMusic)
            prepareMusic();
        else
            addGamePanel();

```

```

    }
    public GamePanel getGamePanel() {
        return gamePanel;
    }
    public int getLevelID() {
        return levelID;
    }
    public int getRetryCount() {
        return retryCount;
    }
    public int[] getPowerUpImages() {
        return powerUpImages;
    }
    // displays the FPS in the textview if the user chose for it to happen in the options
    @SuppressWarnings("SetTextI18n")
    public void setFPSView(double FPS) {
        if (user.optFPS) {
            // uses handler to post messages to the UI thread's message queue
            // which will also ensure that the code is executed on the UI thread
            // and not give an android.view.ViewRootImpl$CalledFromWrongThreadException
            Handler handler = new Handler(Looper.getMainLooper());
            handler.post(() -> tvFPS.setText("FPS: " + FPS));
        }
    }
    // removes previous gamePanel and replace it with a new one
    public void onRetry(GamePanel newGamePanel) {
        frameLayout.removeView(this.gamePanel);
        frameLayout.addView(newGamePanel);
        this.gamePanel = newGamePanel;
        retryCount++;
    }
    // quits the game from the button
    @Override
    public void onClick(View view) {
        if (view == ibBack) {
            this.gamePanel.finishGame();
            this.finish();
        }
    }
    // adds a game panel and replaces the older one
    public void addGamePanel() {
        tvLoading.setVisibility(View.INVISIBLE);
        gamePanel = new GamePanel(this);
        frameLayout = findViewById(R.id.cvGame);
        frameLayout.addView(gamePanel);
    }
    public void prepareMusic() {
        mediaPlayer = new MediaPlayer();
        String url = "";
        // gets the songs url from firebase storage
        switch (levelID) {
            case 0: url = "https://firebasestorage.googleapis.com/v0/b/aplic-1.appspot.com/o/songs%2FStayInsideMe.mp3?alt=media&token=882438ba-a2af-4e80-a508-bad98a899ffd"; break;
            case 1: url = "https://firebasestorage.googleapis.com/v0/b/aplic-1.appspot.com/o/songs%2FPartytArlgang.mp3?alt=media&token=d4cf27f6-a449-436a-9f1c-"; break;
        }
    }

```

```

e765fc8fe352"; break;
    case 2: url = "https://firebasestorage.googleapis.com/v0/b/aplic-
1.appspot.com/o/songs%2FVsSusie.mp3?alt=media&token=70d465f4-7512-4820-90bf-
568dea1e02fd"; break;
    case 3: url = "https://firebasestorage.googleapis.com/v0/b/aplic-
1.appspot.com/o/songs%2FTheVengeanceOfThoseForgottenInDarkness.ogg?alt=media&tok
en=9bfa9f9f-92b6-4fca-8451-13e01800e40d"; break;
    case 4: url = "https://firebasestorage.googleapis.com/v0/b/aplic-
1.appspot.com/o/songs%2FDimension.mp3?alt=media&token=984bffef-9738-4c2b-b637-
c02cf685828e"; break;
    case 5: url = "https://firebasestorage.googleapis.com/v0/b/aplic-
1.appspot.com/o/songs%2FYouWereWrongGoBack.ogg?alt=media&token=d6c84f65-554a-
494c-a1cd-471c62e3a780"; break;
    case 6: url = "https://firebasestorage.googleapis.com/v0/b/aplic-
1.appspot.com/o/songs%2FPeerGynt.mp3?alt=media&token=cc1a7024-d596-4d1b-8db8-
a5404c4bd469"; break;
    case 7: url = "https://firebasestorage.googleapis.com/v0/b/aplic-
1.appspot.com/o/songs%2FLostMemories.mp3?alt=media&token=38f140d1-9722-44a5-
a548-4a5d814d92e3"; break;
    case 8: url = "https://firebasestorage.googleapis.com/v0/b/aplic-
1.appspot.com/o/songs%2FBreadySteadyGo.ogg?alt=media&token=650e5d7e-bb72-435e-
a930-6169121aa6ae"; break;
    case 9: url = "https://firebasestorage.googleapis.com/v0/b/aplic-
1.appspot.com/o/songs%2FBetrayalOfFear.mp3?alt=media&token=16b24faa-c546-40da-
b833-48f0178ae43b"; break;
    case 10: url = "https://firebasestorage.googleapis.com/v0/b/aplic-
1.appspot.com/o/songs%2FWorldsEndValentine.ogg?alt=media&token=2d2d29f4-3797-
4e3b-8b23-50582530b0c6"; break;
    case 11: url = "https://firebasestorage.googleapis.com/v0/b/aplic-
1.appspot.com/o/songs%2FGhostHouse.mp3?alt=media&token=9817b9e6-268a-4d5d-8454-
57d5178f4442"; break;
    case 12: url = "https://firebasestorage.googleapis.com/v0/b/aplic-
1.appspot.com/o/songs%2FMETALOVANIA.mp3?alt=media&token=b753a951-fc29-448d-
bfb9-25012cd9c634"; break;
    case 13: url = "https://firebasestorage.googleapis.com/v0/b/aplic-
1.appspot.com/o/songs%2FSwirly1000x.ogg?alt=media&token=727491d0-d8a7-46af-8d88-
290c1aaa9490"; break;
    case 14: url = "https://firebasestorage.googleapis.com/v0/b/aplic-
1.appspot.com/o/songs%2FKenos.mp3?alt=media&token=c5c7facd-fd6d-4ed1-8c3b-
740de18ea322"; break;
    case 15: url = "https://firebasestorage.googleapis.com/v0/b/aplic-
1.appspot.com/o/songs%2FTeeHeeTime.ogg?alt=media&token=68a28c93-5067-489e-a931-
135ac2a35f9e"; break;
    case 16: url = "https://firebasestorage.googleapis.com/v0/b/aplic-
1.appspot.com/o/songs%2FTheoryOfEverything3(DJNate).mp3?alt=media&token=e8ad7daa-
c429-4313-a7d6-66df717fac44"; break;
    case 17: url = "https://firebasestorage.googleapis.com/v0/b/aplic-
1.appspot.com/o/songs%2FLingeringLightOfTheSettingSun.mp3?alt=media&token=dbc601a
4-a2c1-46e6-957b-b1fde89f85d9"; break;
    case 18: url = "https://firebasestorage.googleapis.com/v0/b/aplic-
1.appspot.com/o/songs%2FIItMeansEverything.ogg?alt=media&token=6ffa99c0-6394-4c04-
a314-5c60c626b6bc"; break;
    case 19: url = "https://firebasestorage.googleapis.com/v0/b/aplic-
1.appspot.com/o/songs%2FSonicBlaster.mp3?alt=media&token=717a50b2-921b-48d3-ac0c-
d01bbf157c65"; break;
    case 20: url = "https://firebasestorage.googleapis.com/v0/b/aplic-

```

```

1.appspot.com/o/songs%2FGoldenvengence.ogg?alt=media&token=b4813fa3-5ee5-4c92-
a220-e060fe6f2381"; break;
    case 21: url = "https://firebasestorage.googleapis.com/v0/b/aplic-
1.appspot.com/o/songs%2FTheoryOfEverything3(domyeah).mp3?alt=media&token=68bf829
8-9af0-4170-8273-c1d96b4978db"; break;
    case 22: url = "https://firebasestorage.googleapis.com/v0/b/aplic-
1.appspot.com/o/songs%2FDarkEnd.mp3?alt=media&token=bcc34058-f880-4eb9-87be-
f4ede76c589e"; break;
    case 23: url = "https://firebasestorage.googleapis.com/v0/b/aplic-
1.appspot.com/o/songs%2FCantGoBack.ogg?alt=media&token=8fdeb4f0-512d-4775-a31a-
917b5c96a810"; break;
    case 24: url = "https://firebasestorage.googleapis.com/v0/b/aplic-
1.appspot.com/o/songs%2FNineCircles.mp3?alt=media&token=0b66d5b0-02b3-4e73-b731-
d03563cfc59c"; break;
    case 25: url = "https://firebasestorage.googleapis.com/v0/b/aplic-
1.appspot.com/o/songs%2FPhobiaWave.mp3?alt=media&token=de78d25c-dda7-4e82-b9af-
e7e273ebd858"; break;
    }
    try {
        //prepares the music
        Log.i(TAG, "prepareMusic: wow");
        mediaPlayer.setDataSource(url);
        mediaPlayer.prepare();
        mediaPlayer.setOnPreparedListener(mediaPlayer -> {
            addGamePanel();
            playMusic();
            Log.i(TAG, "onPrepared: playing music");
        });
    } catch (IOException e) {
        e.printStackTrace();
    }
}
// plays the music
public void playMusic() {
    if (mediaPlayer != null) {
        mediaPlayer.seekTo(0);
        mediaPlayer.start();
        // replays the song when it is over
        mediaPlayer.setOnCompletionListener(mediaPlayer -> {
            mediaPlayer.seekTo(0);
            mediaPlayer.start();
        });
    }
}
// stops the music
public void stopMusic() {
    if (mediaPlayer != null && mediaPlayer.isPlaying()) {
        mediaPlayer.pause();
    }
}
// turns on and off the connection symbol to alert the player if they have connection issues
public void changeConnectionErrorVisibility(boolean connection) {
    if (connection) {
        Log.i(TAG, "changeConnectionErrorVisibility: " + true);
        ivConnection.setVisibility(View.INVISIBLE);
    }
}

```

```

        else {
            Log.i(TAG, "changeConnectionErrorVisibility: " + false);
            ivConnection.setVisibility(View.VISIBLE);
        }
    }
    // broadcast receiver that changes the connection image view when connection status
    // changes
    // and broadcast receiver
    protected void onResume() {
        super.onResume();
        registerReceiver(connectivityChangeReceiver, new
IntentFilter("android.net.conn.CONNECTIVITY_CHANGE"));
    }
    // stops the broadcast receiver, the music, vibrates the phone
    protected void onPause() {
        super.onPause();
        unregisterReceiver(connectivityChangeReceiver);
        if (mediaPlayer != null && mediaPlayer.isPlaying()) {
            mediaPlayer.stop();
            mediaPlayer.release();
        }
        startService(new Intent(this, VibrationService.class));
    }
    // sets the bitmaps for the user from their selected items
    public int getBitmapID (int type) {
        // sets default values for when the player is a guest
        if (iconImages == null)
            switch (type) {
                case 1: return R.drawable.icon1;
                case 2: return R.drawable.background1;
                case 3: return R.drawable.bonus1;
                case 4: return 0;
            }
        switch (type) {
            case 1: return iconImages[user.selectedIcon];
            case 2: return backgroundImages[user.selectedBackground];
            case 3: return bonusImages[user.selectedBonus];
            case 4: return powerUpImages[user.selectedPowerUp];
        }
        return 0;
    }
}
}

```

- public class HubActivity extends AppCompatActivity implements View.OnClickListener {
 private Button btPlay, btOptions, btShop, btSignOut, btRankings;
 private Bundle bitmapArrays;
 @SuppressWarnings("SetTextI18n")
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_hub);
 btPlay = findViewById(R.id.btPlay);
 btOptions = findViewById(R.id.btOptions);

```

    btShop = findViewById(R.id.btShop);
    btSignOut = findViewById(R.id.btSignOut);
    btRankings = findViewById(R.id.btRankings);
    btPlay.setOnClickListener(this);
    btOptions.setOnClickListener(this);
    btShop.setOnClickListener(this);
    btSignOut.setOnClickListener(this);
    btRankings.setOnClickListener(this);
    int[] iconImages, backgroundImages, bonusImages, powerUpImages;
    iconImages = new int[10];
    backgroundImages = new int[7];
    bonusImages = new int[8];
    powerUpImages = new int[4];
    String[] str = new String[]{"icon", "background", "bonus", "power_up"};
    // sets all item arrays for the game and shop
    for (String type: str) {
        int i = 1;
        int imageKey = -1;
        while (imageKey != 0) {
            imageKey = getResources().getIdentifier(type + i, "drawable", getPackageName());
            if (imageKey != 0) {
                switch (type) {
                    case "icon":
                        iconImages[i - 1] = imageKey;
                        break;
                    case "background":
                        backgroundImages[i - 1] = imageKey;
                        break;
                    case "bonus":
                        bonusImages[i - 1] = imageKey;
                        break;
                    case "power_up":
                        powerUpImages[i] = imageKey;
                        break;
                }
            }
            i++;
        }
    }
    bitmapArrays = new Bundle();
    bitmapArrays.putIntArray("icon_images", iconImages);
    bitmapArrays.putIntArray("background_images", backgroundImages);
    bitmapArrays.putIntArray("bonus_images", bonusImages);
    bitmapArrays.putIntArray("power_up_images", powerUpImages);

    String currUserName = user.userName;
    TextView tvUserNameDisplay = findViewById(R.id.tvUserNameDisplay);
    tvUserNameDisplay.setText("Logged in as: " + currUserName);
}
@Override
public void onClick(View view) {
    startService(new Intent(this, VibrationService.class));
    Intent intent;
    if (view == btPlay) {
        intent = new Intent(view.getContext(), LevelSelectActivity.class);
        intent.putExtras(bitmapArrays);
    }
}

```

```

        startActivity(intent);
    } else if (view == btOptions) {
        intent = new Intent(HubActivity.this, OptionsActivity.class);
        startActivity(intent);
    } else if (view == btShop) {
        intent = new Intent(HubActivity.this, ShopActivity.class);
        intent.putExtras(bitmapArrays);
        startActivity(intent);
    } else if (view == btSignOut) {
        FirebaseAuth.getInstance().signOut();
        intent = new Intent(HubActivity.this, LoginActivity.class);
        startActivity(intent);
        finish();
    } else if (view == btRankings) {
        if (isConnectedToInternet(this)) {
            intent = new Intent(HubActivity.this, RankingsActivity.class);
            startActivity(intent);
        }
        else
            Toast.makeText(this, "Please connect to the internet first." ,
                Toast.LENGTH_SHORT).show();
    }
}
}
}

```

- public class LevelSelectActivity extends AppCompatActivity implements View.OnClickListener
{

```

    private FrameLayout frameLayout;
    private TextView tvWorldDisplay;
    private Button btL1, btL2, btL3, btL4, btBoss, btObstInfo, btDifficulty;
    private ImageButton ibBack, ibBackArrow, ibForwardArrow;
    private int worldNum;

```

```

    @Override

```

```

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_level_select);
        frameLayout = findViewById(R.id.frameLayout);

```

```

        tvWorldDisplay = findViewById(R.id.tvWorldDisplay); ibBack =
        findViewById(R.id.ibBack);
        btL1 = findViewById(R.id.ibLevel1); btL2 = findViewById(R.id.ibLevel2);
        btL3 = findViewById(R.id.ibLevel3); btL4 = findViewById(R.id.ibLevel4);
        btBoss = findViewById(R.id.ibBoss); ibBackArrow = findViewById(R.id.ibBackArrow);
        ibForwardArrow = findViewById(R.id.ibForwardArrow);
        btObstInfo = findViewById(R.id.btObstInfo);
        btDifficulty = findViewById(R.id.btDifficulty);

```

```

        btL1.setOnClickListener(this); btL2.setOnClickListener(this);
        btL3.setOnClickListener(this); btL4.setOnClickListener(this);
        btBoss.setOnClickListener(this); ibBack.setOnClickListener(this);
        ibBackArrow.setOnClickListener(this); ibForwardArrow.setOnClickListener(this);
        btObstInfo.setOnClickListener(this);

```

```

        btDifficulty.setOnClickListener(this);

        worldNum = 1;
        for (int i = 1; i <= 4; i++)
            if (user.levelComplete.get(i))
                worldNum = i + 1;
        changeWorld();

        difficultyColor();
    }

    @Override
    public void onClick(View view) {
        if (ibBack.equals(view)) { finish();
        }
        // allows the user to change between the different worlds
        else if (ibBackArrow.equals(view)) {
            worldNum--;
            changeWorld();
        } else if (ibForwardArrow.equals(view)) {
            worldNum++;
            changeWorld();
        }
        else if (isConnectedToInternet(this)) {
            if (btObstInfo.equals(view)) {
                // shows the user information about the game's objects
                ObstInfoDialog obstInfoDialog = new ObstInfoDialog(this);
                obstInfoDialog.show();
            }
            else if (btDifficulty.equals(view)) {
                // allows the user to change the difficulty
                DifficultyDialog difficultyDialog = new DifficultyDialog(this);
                difficultyDialog.show();
            }
            else {
                // if the button isn't any of the previous options it will be a "level" button
                // get the level id from the button and start the level from the id
                startService(new Intent(this, VibrationService.class));
                Button tempButton = (Button) view;
                int levelID = Integer.parseInt(tempButton.getText().toString());
                LevelInfoDialog levelInfoDialog = new LevelInfoDialog(this, levelID,
                getIntent().getExtras());
                levelInfoDialog.show();
            }
        }
        else
            Toast.makeText(this, "Please connect to a stable internet connection",
            Toast.LENGTH_SHORT).show();
    }
    // changes the world number and the text on the buttons
    @SuppressWarnings("SetTextI18n")
    public void changeWorld() {
        tvWorldDisplay.setText("World " + worldNum);
        // world number can't be less than 1 or more than 5
        ibBackArrow.setClickable(worldNum != 1);
        ibForwardArrow.setClickable(worldNum != 5);
        // changes level button text
    }

```



```

        btL1.setText(Integer.toString(5*worldNum-4));
        btL2.setText(Integer.toString(5*worldNum-3));
        btL3.setText(Integer.toString(5*worldNum-2));
        btL4.setText(Integer.toString(5*worldNum-1));
        btBoss.setText(Integer.toString(5*worldNum));
        //changes the background according to the world id
        switch (worldNum) {
            case 1: frameLayout.setBackgroundColor(Color.rgb(105,105,105)); break;
            case 2: frameLayout.setBackgroundColor(Color.rgb(80,80,80)); break;
            case 3: frameLayout.setBackgroundColor(Color.rgb(60,60,60)); break;
            case 4: frameLayout.setBackgroundColor(Color.rgb(40,40,40)); break;
            case 5: frameLayout.setBackgroundColor(Color.rgb(20,20,20)); break;
        }
    }
    // changes the difficulty button according to the user's chosen difficulty
    @SuppressWarnings("SetTextI18n")
    public void difficultyColor() {
        switch (user.difficulty) {
            case 0: btDifficulty.setBackgroundColor(Color.argb(50,50,255,50));
        btDifficulty.setText("DIFFICULTY: EASY"); break;
            case 1: btDifficulty.setBackgroundColor(Color.argb(100,177,97,11));
        btDifficulty.setText("DIFFICULTY: NORMAL"); break;
            case 2: btDifficulty.setBackgroundColor(Color.argb(60,234,11,11));
        btDifficulty.setText("DIFFICULTY: HARD"); break;
        }
    }
}

```

- public class LoginActivity extends AppCompatActivity implements View.OnClickListener, ValueEventListener {

```

    private CheckBox cbSharedPreferences;
    private SharedPreferences sp;
    private Button btQuit, btLogin, btForgot, btNew;
    private EditText etName, etMail, etPass;
    private TextView tvNotSignIn;
    private String userName, mail, password;
    private GifImageView loading;
    private Intent intent;
    private FirebaseAuth firebaseAuth;
    private DatabaseReference userRef;

```

@Override

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login);
    cbSharedPreferences = findViewById(R.id.cbSharedPreferences);
    sp = getSharedPreferences("loginDetails", 0);
    btQuit = findViewById(R.id.btQuit);
    btLogin = findViewById(R.id.btLogin);
    btForgot = findViewById(R.id.btForgot);
    btNew = findViewById(R.id.btNew);
    tvNotSignIn = findViewById(R.id.tvNotSignIn);
    etName = findViewById(R.id.etName);
}

```

```

        etMail = findViewById(R.id.etMail);
        etPass = findViewById(R.id.etPass);
        btQuit.setOnClickListener(this);
        btLogin.setOnClickListener(this);
        btNew.setOnClickListener(this);
        btForgot.setOnClickListener(this);
        tvNotSignIn.setOnClickListener(this);
        loading = findViewById(R.id.gifLoading);
        loading.setVisibility(View.INVISIBLE);
        firebaseAuth = FirebaseAuth.getInstance();
        userRef = FirebaseDatabase.getInstance().getReference("Users");
        // sets the details from shared preferences if they exist
        userName = sp.getString("userName", null);
        mail = sp.getString("mail", null);
        password = sp.getString("password", null);
        if (userName != null && !userName.equals("")) {
            etName.setText(userName);
            etMail.setText(mail);
            etPass.setText(password);
            cbSharedPreferences.setChecked(true);
        }
    }

    @Override
    public void onClick(View view) {
        if (view == btQuit) {
            finish();
        }
        else if (view == btLogin)
        {
            startService(new Intent(this, VibrationService.class));
            if (isConnectedToInternet(this)) {
                userName = etName.getText().toString();
                mail = etMail.getText().toString();
                password = etPass.getText().toString();
                if (checkLoginInformation()) {
                    userRef.addValueEventListener(this);
                    loading.setVisibility(View.VISIBLE);
                    changeClickables(false);
                }
            }
            else {
                Toast.makeText(this, "Please check your connection!",
                    Toast.LENGTH_SHORT).show();
            }
        }
        else if (view == btNew)
        {
            intent = new Intent(this, SignUpActivity.class);
            startActivity(intent);
        }
        else if (view == btForgot)
        {
            intent = new Intent(this, ForgotActivity.class);
            startActivity(intent);
        }
        else if (view == tvNotSignIn) {

```

```

        if (isConnectedToInternet(this)) {
            user = new User();
            user.InitializeAllValues();
            intent = new Intent(this, GameScreenActivity.class);
            intent.putExtra("LevelID", 0);
            Log.i(TAG, "onClick: LEVEL - " + 0);
            startActivity(intent);
        }
        else
            Toast.makeText(this, "Please connect to the internet first!",
                Toast.LENGTH_SHORT).show();
    }
}

// checks if information is acceptable
public boolean checkLoginInformation() {
    if (TextUtils.isEmpty(userName)||
        TextUtils.isEmpty(mail)||TextUtils.isEmpty(password)) {
        Toast.makeText(LoginActivity.this,
            "Please enter all required details!",
            Toast.LENGTH_SHORT).show();
        Log.i(TAG, "onClick: empty");
        return false;
    }
    else if (!Patterns.EMAIL_ADDRESS.matcher(mail).matches()) {
        Toast.makeText(LoginActivity.this,
            "E-Mail is invalid.", Toast.LENGTH_SHORT).show();
        Log.i(TAG, "onClick: email is not valid");
        return false;
    }
    return true;
}

public void loginAccount() {
    // logs in the account if the password is correct
    firebaseAuth.signInWithEmailAndPassword(mail,password)
        .addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                // saves the information to shared preferences if checkbox is ticked
                @SuppressWarnings("CommitPrefEdits") SharedPreferences.Editor editor =
                    sp.edit();

                if (cbSharedPreferences.isChecked()) {
                    editor.putString("userName", userName);
                    editor.putString("mail", mail);
                    editor.putString("password", password);
                }
                else {
                    editor.putString("userName", "");
                    editor.putString("mail", "");
                    editor.putString("password", "");
                }

                editor.apply();
                Toast.makeText(LoginActivity.this,
                    "Login successful",
                    Toast.LENGTH_SHORT).show();
                intent = new Intent(LoginActivity.this, HubActivity.class);
                startActivity(intent);
            }
        });
}

```

```

        finish();
    }
    else {
        Toast.makeText(LoginActivity.this,
            "Details incorrect!",
            Toast.LENGTH_SHORT).show();
    }
});
}
// goes through each account and looks for the one with
@Override
public void onDataChange(@NonNull DataSnapshot snapshot) {
    Log.i(TAG, "onDataChange: onDataChange");
    userRef.removeEventListener(this);
    for (DataSnapshot data : snapshot.getChildren()) {
        User tempUser = data.getValue(User.class);
        assert tempUser != null;
        Log.i(TAG, "onDataChange: checking user " + tempUser.uid);
        if (Objects.equals(tempUser.mail, etMail.getText().toString())) {
            Log.i(TAG, "onDataChange: email correct");
            if (Objects.equals(tempUser.userName, etName.getText().toString())) {
                Log.i(TAG, "onDataChange: CORRECT");
                user = data.getValue(User.class);
                loginAccount();
                loading.setVisibility(View.INVISIBLE);
                changeClickables(true);
                return;
            }
            else Log.i(TAG, "onDataChange: incorrect 2");
        }
        else Log.i(TAG, "onDataChange: incorrect 1");
    }
    Toast.makeText(LoginActivity.this,
        "Details incorrect!",
        Toast.LENGTH_SHORT).show();
    loading.setVisibility(View.INVISIBLE);
    changeClickables(true);
}
@Override
public void onCancelled(@NonNull DatabaseError error) {
    Log.i(TAG, "onCancelled: ERROR");
}
// makes it so other button aren't clickable when firebase is checking the information
public void changeClickables(boolean b) {
    btLogin.setClickable(b);
    btForgot.setClickable(b);
    btNew.setClickable(b);
}
}
}

```

- public class OptionsActivity extends AppCompatActivity implements CompoundButton.OnCheckedChangeListener, View.OnClickListener {

private DatabaseReference userRef;

```

private SwitchMaterial switchSoundEffects, switchMusic, switchFPS;
private Button btSave, btBack, btReset;
private boolean music, soundEffects, FPS;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_options);

    userRef = FirebaseDatabase.getInstance().getReference("Users").child(user.key);

    switchSoundEffects = findViewById(R.id.switchSoundEffects);
    switchMusic = findViewById(R.id.switchMusic);
    switchFPS = findViewById(R.id.switchFPS);
    btSave = findViewById(R.id.btSave);
    btBack = findViewById(R.id.btBack);
    btReset = findViewById(R.id.btReset);

    switchSoundEffects.setOnCheckedChangeListener(this);
    switchMusic.setOnCheckedChangeListener(this);
    switchFPS.setOnCheckedChangeListener(this);
    btSave.setOnClickListener(this);
    btBack.setOnClickListener(this);
    btReset.setOnClickListener(this);

    setSwitches();
}
// sets the switches to the user's selected option
public void setSwitches() {
    switchMusic.setChecked(user.optMusic);
    switchSoundEffects.setChecked(user.optSoundEffects);
    switchFPS.setChecked(user.optFPS);
}

@Override
public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
    if (compoundButton == switchMusic) {
        music = b;
    }
    if (compoundButton == switchSoundEffects) {
        soundEffects = b;
    }
    if (compoundButton == switchFPS) {
        FPS = b;
    }
}

@Override
public void onClick(View view) {
    if (view == btBack)
        finish();
    else if (isConnectedToInternet(this)) {
        if (view == btSave) {
            // saves the information from the switches
            startService(new Intent(this, VibrationService.class));
            user.optMusic = music;
            userRef.child("optMusic").setValue(music);
        }
    }
}

```

```

        user.optSoundEffects = soundEffects;
        userRef.child("optSoundEffects").setValue(soundEffects);

        user.optFPS = FPS;
        userRef.child("optFPS").setValue(FPS);

        Toast.makeText(this, "Options saved!", Toast.LENGTH_SHORT).show();
    } else if (view == btReset) {
        // makes an alert dialog to reset all information
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Are you sure");
        builder.setMessage("This will reset all of your data!");
        builder.setPositiveButton("Yes", (dialog, which) -> {
            if (!isConnectedToInternet(OptionsActivity.this)) {
                Toast.makeText(OptionsActivity.this, "Please check your internet and try
again", Toast.LENGTH_SHORT).show();
            }
            else {
                startService(new Intent(OptionsActivity.this, VibrationService.class));
                // resets information
                user.InitializeAllValues();
                userRef.setValue(user);
                dialog.dismiss();
                finish();
                Toast.makeText(OptionsActivity.this, "All data has been reset!",
Toast.LENGTH_SHORT).show();
            }
        });
        builder.setNegativeButton("No", (dialog, which) -> dialog.dismiss());
        AlertDialog ResetDialog = builder.create();
        ResetDialog.show();
    }
}
else
    Toast.makeText(this, "Please check your internet and try again",
Toast.LENGTH_SHORT).show();
}
}

```

- public class RankingsActivity extends AppCompatActivity implements
AdapterView.OnItemClickListener, ValueEventListener {

```

    private DatabaseReference userRef;
    private List<User> userList;
    private TableRow scoreHeader, deathHeader, itemHeader;
    private List<String[]> deathRows, scoreRows, itemRows;
    private TableLayout tableLayout;

```

```

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_rankings);
    }

```

```

userRef = FirebaseDatabase.getInstance().getReference("Users");
userRef.addValueEventListener(this);
userList = new ArrayList<>();
scoreRows = new ArrayList<>();
deathRows = new ArrayList<>();
itemRows = new ArrayList<>();
scoreHeader = new TableRow(this);
deathHeader = new TableRow(this);
itemHeader = new TableRow(this);

ImageButton ibBack = findViewById(R.id.ibBack);
ibBack.setOnClickListener(view -> finish());
setSpinner();
tableLayout = findViewById(R.id.tableLayout);

setHeader();
}
// creates the spinner
public void setSpinner() {
    Spinner spinner = findViewById(R.id.spinner);
    // list for different types of ranking
    List<String> rankingTypes = new ArrayList<>();
    rankingTypes.add(" (none)");
    rankingTypes.add(" Total Score:");
    rankingTypes.add(" Total Deaths:");
    rankingTypes.add(" Total Items Owned:");

    ArrayAdapter<String> rankingTypesAdapter = new ArrayAdapter<>(this,
R.layout.spinner_selected_item, rankingTypes);
    rankingTypesAdapter.setDropDownViewResource(R.layout.spinner_dropdown_item);
    spinner.setAdapter(rankingTypesAdapter);
    spinner.setOnItemSelectedListener(this);
}
// creates for each ranking type it's header
public void setHeader() {
    scoreHeader.setLayoutParams(new TableRow.LayoutParams(
        TableRow.LayoutParams.MATCH_PARENT,
        TableRow.LayoutParams.MATCH_PARENT));
    String[] scoreHeaderText = {"Rank", "Username", "Score"};
    for (String header: scoreHeaderText) {
        TextView headerView = new TextView(this);
        headerView.setLayoutParams(new TableRow.LayoutParams(
            TableRow.LayoutParams.WRAP_CONTENT,
            TableRow.LayoutParams.WRAP_CONTENT));
        headerView.setText(header);
        headerView.setTextColor(getColor(R.color.white));
        headerView.setTypeface(ResourcesCompat.getFont(this,
R.font.berlin_sans_fb_demi_bold));
        headerView.setPadding(20, 20, 20, 50);
        scoreHeader.setBackgroundColor(getColor(R.color.black));
        scoreHeader.addView(headerView);
    }
    deathHeader.setLayoutParams(new TableRow.LayoutParams(
        TableRow.LayoutParams.MATCH_PARENT,
        TableRow.LayoutParams.MATCH_PARENT));
    String[] deathHeaderText = {"Rank", "Username", "Deaths"};

```

```

    for (String header: deathHeaderText) {
        TextView headerView = new TextView(this);
        headerView.setLayoutParams(new TableRow.LayoutParams(
            TableRow.LayoutParams.WRAP_CONTENT,
            TableRow.LayoutParams.WRAP_CONTENT));
        headerView.setText(header);
        headerView.setTextColor(getColor(R.color.white));
        headerView.setTypeface(ResourcesCompat.getFont(this,
            R.font.berlin_sans_fb_demi_bold));
        headerView.setPadding(20, 20, 20, 50);
        deathHeader.setBackgroundColor(getColor(R.color.black));
        deathHeader.addView(headerView);
    }
    itemHeader.setLayoutParams(new TableRow.LayoutParams(
        TableRow.LayoutParams.MATCH_PARENT,
        TableRow.LayoutParams.MATCH_PARENT));
    String[] itemHeaderText = {"Rank", "Username", "Items"};
    for (String header: itemHeaderText) {
        TextView headerView = new TextView(this);
        headerView.setLayoutParams(new TableRow.LayoutParams(
            TableRow.LayoutParams.WRAP_CONTENT,
            TableRow.LayoutParams.WRAP_CONTENT));
        headerView.setText(header);
        headerView.setTextColor(getColor(R.color.white));
        headerView.setTypeface(ResourcesCompat.getFont(this,
            R.font.berlin_sans_fb_demi_bold));
        headerView.setPadding(20, 20, 20, 50);
        itemHeader.setBackgroundColor(getColor(R.color.black));
        itemHeader.addView(headerView);
    }
}

// sums up the user's score
public int scoreSum(User tempUser) {
    List<Integer> userScoreList = tempUser.hsl;
    int sum = 0;
    for (int number : userScoreList) {
        sum += number;
    }
    return sum;
}

// creates rankings table for each type
public void setLists() {
    List<User> scoreList = userList;
    scoreList.sort(Comparator.comparingInt(this::scoreSum));
    for (int i = scoreList.size() - 1; i >= Math.max(scoreList.size() - 10, 0); i--) {
        scoreRows.add(new String[]{Integer.toString(scoreList.size() - i),
            scoreList.get(i).userName, String.valueOf(scoreSum(scoreList.get(i)))});
    }

    List<User> deathList = userList;
    deathList.sort(Comparator.comparingInt(u -> u.deaths));
    for (int i = deathList.size() - 1; i >= Math.max(deathList.size() - 10, 0); i--) {
        deathRows.add(new String[]{Integer.toString(deathList.size() - i),
            deathList.get(i).userName, String.valueOf(deathList.get(i).deaths)});
    }
}

```



```

List<User> itemList = userList;
itemList.sort(Comparator.comparingInt(u -> u.itemsOwned));
for (int i = itemList.size() - 1; i >= Math.max(itemList.size() - 10, 0); i--) {
    itemRows.add(new String[]{Integer.toString(itemList.size() - i),
itemList.get(i).userName, String.valueOf(itemList.get(i).itemsOwned)});
}
}
// every time an option is chosen the table changes according to the option chosen
@Override
public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
    String item = adapterView.getItemAtPosition(i).toString();
    tableLayout.removeAllViews();
    switch (item) {
        case " Total Score:":
            tableLayout.addView(scoreHeader);

            for (String[] row : scoreRows) {
                TableRow tableRow = new TableRow(this);
                tableRow.setLayoutParams(new TableRow.LayoutParams(
                    TableRow.LayoutParams.MATCH_PARENT,
                    TableRow.LayoutParams.WRAP_CONTENT));
                for (String value : row) {
                    TextView valueView = new TextView(this);
                    valueView.setLayoutParams(new TableRow.LayoutParams(
                        TableRow.LayoutParams.WRAP_CONTENT,
                        TableRow.LayoutParams.WRAP_CONTENT));
                    valueView.setText(value);
                    valueView.setTextColor(getColor(R.color.white));
                    valueView.setTypeface(ResourcesCompat.getFont(this,
R.font.berlin_sans_fb_demi_bold));
                    valueView.setPadding(20, 10, 150, 20);
                    tableRow.setBackgroundColor(getColor(R.color.black));
                    tableRow.addView(valueView);
                }
                tableLayout.addView(tableRow);
            }
            break;
        case " Total Deaths:":
            tableLayout.addView(deathHeader);

            for (String[] row : deathRows) {
                TableRow tableRow = new TableRow(this);
                tableRow.setLayoutParams(new TableRow.LayoutParams(
                    TableRow.LayoutParams.MATCH_PARENT,
                    TableRow.LayoutParams.WRAP_CONTENT));
                for (String value : row) {
                    TextView valueView = new TextView(this);
                    valueView.setLayoutParams(new TableRow.LayoutParams(
                        TableRow.LayoutParams.WRAP_CONTENT,
                        TableRow.LayoutParams.WRAP_CONTENT));
                    valueView.setText(value);
                    valueView.setTextColor(getColor(R.color.white));
                    valueView.setTypeface(ResourcesCompat.getFont(this,
R.font.berlin_sans_fb_demi_bold));
                    valueView.setPadding(20, 10, 150, 20);
                    tableRow.setBackgroundColor(getColor(R.color.black));

```

```

        tableRow.addView(valueView);
    }
    tableLayout.addView(tableRow);
}
break;
case " Total Items Owned:":
    tableLayout.addView(itemHeader);

    for (String[] row : itemRows) {
        TableRow tableRow = new TableRow(this);
        tableRow.setLayoutParams(new TableRow.LayoutParams(
            TableRow.LayoutParams.MATCH_PARENT,
            TableRow.LayoutParams.WRAP_CONTENT));
        for (String value : row) {
            TextView valueView = new TextView(this);
            valueView.setLayoutParams(new TableRow.LayoutParams(
                TableRow.LayoutParams.WRAP_CONTENT,
                TableRow.LayoutParams.WRAP_CONTENT));
            valueView.setText(value);
            valueView.setTextColor(getColor(R.color.white));
            valueView.setTypeface(ResourcesCompat.getFont(this,
                R.font.berlin_sans_fb_demi_bold));
            valueView.setPadding(20, 10, 150, 20);
            tableRow.setBackgroundColor(getColor(R.color.black));
            tableRow.addView(valueView);
        }
        tableLayout.addView(tableRow);
    }
    break;
}
}
@Override
public void onNothingSelected(AdapterView<?> adapterView) {
}
// creates a list of all users
@Override
public void onDataChange(@NonNull DataSnapshot snapshot) {
    Log.i("TAG", "onDataChange: ");
    userRef.removeEventListener(this);
    for (DataSnapshot data : snapshot.getChildren()) {
        User tempUser = data.getValue(User.class);
        userList.add(tempUser);
    }
    setLists();
    Log.i("TAG", "onDataChange: " + userList);
}
@Override
public void onCancelled(@NonNull DatabaseError error) {
}
}

```

- public class ShopActivity extends AppCompatActivity implements
NavigationView.OnNavigationItemSelectedListener {

```

private FragmentManager fragmentManager;
private FragmentTransaction fragmentTransaction;
private Bundle bitmapArrays;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_shop);
    Toolbar toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    fragmentManager = getSupportFragmentManager();

    DrawerLayout drawer = findViewById(R.id.drawer_layout);
    ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
        this, drawer, toolbar, R.string.navigation_drawer_open,
        R.string.navigation_drawer_close);
    drawer.addDrawerListener(toggle);
    toggle.syncState();

    NavigationView navigationView = findViewById(R.id.nav_view);
    navigationView.setCheckedItem(R.id.nav_preview);
    navigationView.setNavigationItemSelectedListener(this);

    bitmapArrays = getIntent().getExtras();
    // sets the activity to preview fragment when activity is created
    PreviewFragment previewFragment = new PreviewFragment();
    previewFragment.setArguments(bitmapArrays);
    fragmentTransaction = fragmentManager.beginTransaction();
    fragmentTransaction.replace(R.id.container, previewFragment, "tagPreview");
    fragmentTransaction.commit();
}
// if the drawer is open, pressing the back button will close it instead of finishing the activity
@Override
public void onBackPressed() {
    DrawerLayout drawer = findViewById(R.id.drawer_layout);
    if (drawer.isDrawerOpen(GravityCompat.START)) {
        drawer.closeDrawer(GravityCompat.START);
    } else {
        super.onBackPressed();
    }
}
// creates and replaces with previous fragment, the fragment according to navigation item
// selected, and gives it a tag
@SuppressWarnings("NonConstantResourceId")
@Override
public boolean onNavigationItemSelectedListener(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.nav_preview:
            PreviewFragment previewFragment = new PreviewFragment();
            previewFragment.setArguments(bitmapArrays);
            fragmentTransaction = fragmentManager.beginTransaction();
            fragmentTransaction.replace(R.id.container, previewFragment, "tagPreview");
            fragmentTransaction.commit();
            break;
        case R.id.nav_inventory:

```

```

        InventoryFragment inventoryFragment = new InventoryFragment();
        inventoryFragment.setArguments(bitmapArrays);
        fragmentTransaction = fragmentManager.beginTransaction();
        fragmentTransaction.replace(R.id.container, inventoryFragment, "tagInventory");
        fragmentTransaction.commit();
        break;
    case R.id.nav_icon:
        ShopBuyFragment iconFragment = new ShopBuyFragment(this);
        iconFragment.setArguments(bitmapArrays);
        fragmentTransaction = fragmentManager.beginTransaction();
        fragmentTransaction.replace(R.id.container, iconFragment, "tagIcon");
        fragmentTransaction.commit();
        break;
    case R.id.nav_background:
        ShopBuyFragment backgroundFragment = new ShopBuyFragment(this);
        backgroundFragment.setArguments(bitmapArrays);
        fragmentTransaction = fragmentManager.beginTransaction();
        fragmentTransaction.replace(R.id.container, backgroundFragment,
"tagBackground");
        fragmentTransaction.commit();
        break;
    case R.id.nav_bonus:
        ShopBuyFragment bonusFragment = new ShopBuyFragment(this);
        bonusFragment.setArguments(bitmapArrays);
        fragmentTransaction = fragmentManager.beginTransaction();
        fragmentTransaction.replace(R.id.container, bonusFragment, "tagBonus");
        fragmentTransaction.commit();
        break;
    case R.id.nav_power_up:
        ShopBuyFragment powerUpFragment = new ShopBuyFragment(this);
        powerUpFragment.setArguments(bitmapArrays);
        fragmentTransaction = fragmentManager.beginTransaction();
        fragmentTransaction.replace(R.id.container, powerUpFragment, "tagPowerUp");
        fragmentTransaction.commit();
        break;
    }
    DrawerLayout drawer = findViewById(R.id.drawer_layout);
    drawer.closeDrawer(GravityCompat.START);
    return true;
}
}

```

- public class SignUpActivity extends AppCompatActivity implements View.OnClickListener, ValueEventListener {
 private EditText etName, etMail, etPass, etConfPass;
 private String userName, mail, password, confPassword;
 private ImageButton ibBack;
 private Button btSignUp;
 private FirebaseAuth firebaseAuth;
 private FirebaseDatabase firebaseDatabase;
 private DatabaseReference userRef;

 @Override
 protected void onCreate(Bundle savedInstanceState) {

```

super.onCreate(savedInstanceState);
setContentView(R.layout.activity_sign_up);
etName = findViewById(R.id.etNewName);
etMail = findViewById(R.id.etNewMail);
etPass = findViewById(R.id.etNewPass);
etConfPass = findViewById(R.id.etConfNewPass);
btSignUp = findViewById(R.id.btSignUp);
btSignUp.setOnClickListener(this);
ibBack = findViewById(R.id.ibBack);
ibBack.setOnClickListener(this);
firebaseAuth = FirebaseAuth.getInstance();
firebaseDatabase = FirebaseDatabase.getInstance();
userRef = firebaseDatabase.getReference("Users").push();
}

@Override
public void onClick(View view) {
    if (view == ibBack)
        finish();
    if (view == btSignUp)
    {
        if (isConnectedToInternet(this)) {
            startService(new Intent(this, VibrationService.class));
            userName = etName.getText().toString();
            mail = etMail.getText().toString();
            password = etPass.getText().toString();
            confPassword = etConfPass.getText().toString();
            signUp();
        }
        else
            Toast.makeText(this,
                "Please connect to the internet!",
                Toast.LENGTH_SHORT).show();
    }
}

// checks if information is acceptable
public void signUp() {
    if (TextUtils.isEmpty(userName)|| TextUtils.isEmpty(mail)||
        TextUtils.isEmpty(password)||TextUtils.isEmpty(confPassword)) {
        Toast.makeText(SignUpActivity.this,
            "Please enter all required details!",
            Toast.LENGTH_SHORT).show();
        return;
    }
    else if (!Patterns.EMAIL_ADDRESS.matcher(mail).matches()) {
        Toast.makeText(getApplicationContext(),
            "E-Mail is invalid.", Toast.LENGTH_SHORT).show();
        return;
    }
    else if (!Objects.equals(password, confPassword)) {
        Toast.makeText(this, "Passwords do not match", Toast.LENGTH_SHORT).show();
        return;
    }
    firebaseDatabase.getReference("Users").addValueEventListener(this);
}

// adds a user to firebase database

```

```

    public void addUserDetails() {
        String uid =
Objects.requireNonNull(FirebaseAuth.getInstance().getCurrentUser()).getUid();
        String key = userRef.getKey();
        user = new User(uid,mail,userName,key);
        userRef.setValue(user);
        finish();
    }

    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        // looks for if username or email account are already being used
        userRef.removeEventListener(this);
        firebaseDatabase.getReference("Users").removeEventListener(this);
        for (DataSnapshot data : snapshot.getChildren()) {
            User tempUser = data.getValue(User.class);
            assert tempUser != null;
            Log.i(TAG, "onDataChange: checking user " + tempUser.uid);
            if (Objects.equals(tempUser.mail, etMail.getText().toString())) {
                Log.i(TAG, "onDataChange: email exists");
                Toast.makeText(SignUpActivity.this, "Email account already used",
Toast.LENGTH_SHORT).show();
                return;
            }
            else {
                if (Objects.equals(tempUser.userName, etName.getText().toString())) {
                    Toast.makeText(SignUpActivity.this, "Username already used",
Toast.LENGTH_SHORT).show();
                    Log.i(TAG, "onDataChange: username exists");
                    return;
                }
            }
        }
        // creates a firebase user
        firebaseAuth.createUserWithEmailAndPassword(mail, password)
            .addOnCompleteListener(this, task -> {
                if (task.isSuccessful()) {
                    Toast.makeText(SignUpActivity.this,
                        "Authentication success.",
                        Toast.LENGTH_SHORT).show();
                    addUserDetails();
                } else {
                    Toast.makeText(SignUpActivity.this,
                        "Authentication failed.",
                        Toast.LENGTH_SHORT).show();
                }
            });
    }

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

    }
}

```

- ```

public class SplashScreenActivity extends AppCompatActivity {

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_splash_screen);

 ImageView ivSplash = findViewById(R.id.SplashScreenImage);
 // shows an animation when the app is launched
 Animation splashAnim = AnimationUtils.loadAnimation(this, R.anim.anim_splash);
 ivSplash.startAnimation(splashAnim);
 Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
 // waits a few milliseconds after the animation ends and changes activity to login activity
 final Handler handler = new Handler(Looper.getMainLooper());
 handler.postDelayed(() -> {startActivity(intent); finish();}, 3300);
 }
}

```
- ```

public class InventoryFragment extends Fragment {

    public InventoryFragment() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_inventory, container, false);
        ImageButton ibBack = view.findViewById(R.id.ibBack);
        ibBack.setOnClickListener(viewBack -> requireActivity().finish());
        LinearLayout iconLayout = view.findViewById(R.id.icon_layout);
        LinearLayout backgroundLayout = view.findViewById(R.id.background_layout);
        LinearLayout bonusLayout = view.findViewById(R.id.bonus_layout);
        LinearLayout powerUpLayout = view.findViewById(R.id.power_up_layout);

        // gets images from ShopActivity
        assert getArguments() != null;
        int[] iconImages = getArguments().getIntArray("icon_images");
        int[] backgroundImages = getArguments().getIntArray("background_images");
        int[] bonusImages = getArguments().getIntArray("bonus_images");
        int[] powerUpBonusImages = getArguments().getIntArray("power_up_images");

        // make a view for each owned item and put it in its list

        for (int i = 0; i < iconImages.length; i++) {
            if (user.iconOwned.get(i)) {
                ImageView ivIcon;
                ivIcon = new ImageView(getContext());
                ivIcon.setImageResource(iconImages[i]);
                LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(
                    LinearLayout.LayoutParams.WRAP_CONTENT,
                    LinearLayout.LayoutParams.WRAP_CONTENT);
                params.setMargins(25, 0, 25, 0);
                params.width = 150;
                params.height = 150;
            }
        }
    }
}

```

```

        ivIcon.setLayoutParams(params);
        iconLayout.addView(ivIcon);
    }
}
for (int i = 0; i < backgroundImages.length; i++) {
    if (user.backgroundOwned.get(i)) {
        ImageView ivBackground;
        ivBackground = new ImageView(getContext());
        ivBackground.setImageResource(backgroundImages[i]);
        LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.WRAP_CONTENT,
            LinearLayout.LayoutParams.WRAP_CONTENT);
        params.setMargins(25, 0, 25, 0);
        params.width = 150;
        params.height = 250;
        ivBackground.setLayoutParams(params);
        backgroundLayout.addView(ivBackground);
    }
}
for (int i = 0; i < bonusImages.length; i++) {
    if (user.bonusOwned.get(i)) {
        ImageView ivBonus;
        ivBonus = new ImageView(getContext());
        ivBonus.setImageResource(bonusImages[i]);
        LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.WRAP_CONTENT,
            LinearLayout.LayoutParams.WRAP_CONTENT);
        params.setMargins(25, 0, 25, 0);
        params.width = 150;
        params.height = 150;
        ivBonus.setLayoutParams(params);
        bonusLayout.addView(ivBonus);
    }
}
for (int i = 1; i < powerUpBonusImages.length; i++) {
    if (user.powerUpOwned.get(i)) {
        ImageView ivPowerUp;
        ivPowerUp = new ImageView(getContext());
        ivPowerUp.setImageResource(powerUpBonusImages[i]);
        LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.WRAP_CONTENT,
            LinearLayout.LayoutParams.WRAP_CONTENT);
        params.setMargins(25, 0, 25, 0);
        params.width = 150;
        params.height = 150;
        ivPowerUp.setLayoutParams(params);
        powerUpLayout.addView(ivPowerUp);
    }
}
return view;
}
}

```


- ```

public class PreviewFragment extends Fragment {

 public PreviewFragment() {
 }

 @Override
 public View onCreateView(LayoutInflater inflater, ViewGroup container,
 Bundle savedInstanceState) {
 View view = inflater.inflate(R.layout.fragment_preview, container, false);
 ImageView ivIcon = view.findViewById(R.id.ivIcon);
 ImageView ivBackground = view.findViewById(R.id.ivBackground);
 ImageView ivBonus = view.findViewById(R.id.ivBonus);
 ImageView ivPowerUp = view.findViewById(R.id.ivPowerUp);
 TextView tvNoPowerUp = view.findViewById(R.id.tvNoPowerUp);
 ImageButton ibBack = view.findViewById(R.id.ibBack);
 ibBack.setOnClickListener(viewBack -> requireActivity().finish());

 // set the image of each item to owned item
 assert getArguments() != null;

 ivIcon.setImageResource(getArguments().getIntArray("icon_images")[user.selectedIcon]);

 ivBackground.setImageResource(getArguments().getIntArray("background_images")[user.selectedBackground]);

 ivBonus.setImageResource(getArguments().getIntArray("bonus_images")[user.selectedBonus]);

 // if user doesn't have a power up selected, switch the image to text saying "none"
 if (getArguments().getIntArray("power_up_images")[user.selectedPowerUp] != 0)

 ivPowerUp.setImageResource(getArguments().getIntArray("power_up_images")[user.selectedPowerUp]);
 else {
 ivPowerUp.setVisibility(View.GONE);
 tvNoPowerUp.setVisibility(View.VISIBLE);
 }
 return view;
 }
}

```
- ```

public class ShopBuyFragment extends Fragment implements
    AdapterView.OnItemClickListener {

    private final Context context;
    private final DatabaseReference userRef;
    private String tag;
    private TextView tvCoins;
    private ListView lvShop;
    private ShopItemAdapter shopItemAdapter;

    public ShopBuyFragment(Context context) {
        this.context = context;
        this.userRef = FirebaseDatabase.getInstance().getReference("Users").child(user.key);
    }
}

```

```

    }
    @SuppressWarnings("NonConstantResourceId")
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this.tag = getTag();
        View view = inflater.inflate(R.layout.fragment_shop_buy, container, false);
        ImageButton ibBack = view.findViewById(R.id.ibBack);
        ibBack.setOnClickListener(viewBack -> requireActivity().finish());
        ImageButton ibInfo = view.findViewById(R.id.ibInfo);
        ibInfo.setOnClickListener(viewInfo -> {
            ShopInfoDialog shopInfoDialog = new ShopInfoDialog(context, tag);
            shopInfoDialog.show();
        });

        tvCoins = view.findViewById(R.id.tvCoins);
        setCoinsDisplay();
        ArrayList<ShopItem> shopItemList = new ArrayList<>();

        Log.i("TAG", "onCreateView: tag is " + tag);
        // add item to listview for each shop item
        switch (tag) {
            case "tagIcon":
                assert getArguments() != null;
                int[] iconImages = getArguments().getIntArray("icon_images");
                ShopItem icon0 = new ShopItem(context, iconImages[0], "Smiley face", 0,
                    user.iconOwned.get(0));
                ShopItem icon1 = new ShopItem(context, iconImages[1], "Snowflake", 20,
                    user.iconOwned.get(1));
                ShopItem icon2 = new ShopItem(context, iconImages[2], "Creeper", 50,
                    user.iconOwned.get(2));
                ShopItem icon3 = new ShopItem(context, iconImages[3], "Checkers Board", 100,
                    user.iconOwned.get(3));
                ShopItem icon4 = new ShopItem(context, iconImages[4], "Cat", 100,
                    user.iconOwned.get(4));
                ShopItem icon5 = new ShopItem(context, iconImages[5], "Monster", 100,
                    user.iconOwned.get(5));
                ShopItem icon6 = new ShopItem(context, iconImages[6], "Iron Man", 100,
                    user.iconOwned.get(6));
                ShopItem icon7 = new ShopItem(context, iconImages[7], "Robot", 150,
                    user.iconOwned.get(7));
                ShopItem icon8 = new ShopItem(context, iconImages[8], "EVW", 180,
                    user.iconOwned.get(8));
                ShopItem icon9 = new ShopItem(context, iconImages[9], "Npesta", 200,
                    user.iconOwned.get(9));
                shopItemList.add(icon0); shopItemList.add(icon1); shopItemList.add(icon2);
                shopItemList.add(icon3); shopItemList.add(icon4); shopItemList.add(icon5);
                shopItemList.add(icon6); shopItemList.add(icon7); shopItemList.add(icon8);
                shopItemList.add(icon9);
                break;
            case "tagBackground":
                assert getArguments() != null;
                int[] backgroundImages = getArguments().getIntArray("background_images");
                ShopItem background0 = new ShopItem(context, backgroundImages[0], "Empty",

```

```

0, user.backgroundOwned.get(0));
    ShopItem background1 = new ShopItem(context, backgroundImages[1], "Space",
20, user.backgroundOwned.get(1));
    ShopItem background2 = new ShopItem(context, backgroundImages[2], "Wood",
50, user.backgroundOwned.get(2));
    ShopItem background3 = new ShopItem(context, backgroundImages[3], "Sky", 100,
user.backgroundOwned.get(3));
    ShopItem background4 = new ShopItem(context, backgroundImages[4], "Stairs",
150, user.backgroundOwned.get(4));
    ShopItem background5 = new ShopItem(context, backgroundImages[5], "Bricks",
180, user.backgroundOwned.get(5));
    ShopItem background6 = new ShopItem(context, backgroundImages[6], "Cyber",
200, user.backgroundOwned.get(6));
    shopItemList.add(background0); shopItemList.add(background1);
shopItemList.add(background2);
    shopItemList.add(background3); shopItemList.add(background4);
shopItemList.add(background5);
    shopItemList.add(background6);
    break;
case "tagBonus":
    assert getArguments() != null;
    int[] bonusImages = getArguments().getIntArray("bonus_images");
    ShopItem bonus0 = new ShopItem(context, bonusImages[0], "Question Mark", 0,
user.bonusOwned.get(0));
    ShopItem bonus1 = new ShopItem(context, bonusImages[1], "Chest", 20,
user.bonusOwned.get(1));
    ShopItem bonus2 = new ShopItem(context, bonusImages[2], "Strawberry", 50,
user.bonusOwned.get(2));
    ShopItem bonus3 = new ShopItem(context, bonusImages[3], "Souls", 100,
user.bonusOwned.get(3));
    ShopItem bonus4 = new ShopItem(context, bonusImages[4], "Star", 100,
user.bonusOwned.get(4));
    ShopItem bonus5 = new ShopItem(context, bonusImages[5], "Light", 150,
user.bonusOwned.get(5));
    ShopItem bonus6 = new ShopItem(context, bonusImages[6], "Delta Rune", 180,
user.bonusOwned.get(6));
    ShopItem bonus7 = new ShopItem(context, bonusImages[7], "Golden Strawberry",
200, user.bonusOwned.get(7));
    shopItemList.add(bonus0);
    shopItemList.add(bonus1);
    shopItemList.add(bonus2);
    shopItemList.add(bonus3);
    shopItemList.add(bonus4);
    shopItemList.add(bonus5);
    shopItemList.add(bonus6);
    shopItemList.add(bonus7);
    break;
case "tagPowerUp":
    assert getArguments() != null;
    int[] powerUpImages = getArguments().getIntArray("power_up_images");
    ShopItem powerUp0 = new ShopItem(context, powerUpImages[0], "None",
0, user.powerUpOwned.get(0));
    ShopItem powerUp1 = new ShopItem(context, powerUpImages[1], "Higher Speed",
250, user.powerUpOwned.get(1));
    ShopItem powerUp2 = new ShopItem(context, powerUpImages[2], "Smaller size",
250, user.powerUpOwned.get(2));

```

```

        ShopItem powerUp3 = new ShopItem(context, powerUpImages[3], "Slower
Obstacles", 250, user.powerUpOwned.get(3));
        shopItemList.add(powerUp0);
        shopItemList.add(powerUp1);
        shopItemList.add(powerUp2);
        shopItemList.add(powerUp3);
        break;
    }
    shopItemAdapter = new ShopItemAdapter(context, 0, 0, shopItemList, tag);
    lvShop = view.findViewById(R.id.lvShop);
    lvShop.setAdapter(shopItemAdapter);
    lvShop.setOnItemClickListener(this);

    Log.i("TAG", "onCreateView: " + user);
    return view;
}
@SuppressWarnings("SetTextI18n")
public void setCoinsDisplay() {
    tvCoins.setText("You have " + user.coins + " coins!");
}
@SuppressWarnings("NonConstantResourceId")
@Override
public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
    if (isConnectedToInternet(context)) {
        ShopItem item = (ShopItem) lvShop.getItemAtPosition(i);
        ShopItem lastSelected = shopItemAdapter.getItem(i);

        // if owned switch owned item to it and switch the item with a tick
        if (lastSelected.isOwned()) {
            setSelectedView(i);
            lvShop.setAdapter(shopItemAdapter);
        }
        // if not owned open purchase dialog
        else {
            PurchaseConfirmDialog purchaseConfirmDialog = new
PurchaseConfirmDialog(context, this, item, i, tag);
            purchaseConfirmDialog.show();
        }
    } else
        Toast.makeText(context, "Please connect to the internet!",
Toast.LENGTH_SHORT).show();
}
// changed user and firebase selected item
public void setSelectedView(int i) {
    switch (tag) {
        case "tagIcon":
            user.selectedIcon = i;
            userRef.child("selectedIcon").setValue(i);
            break;
        case "tagBackground":
            user.selectedBackground = i;
            userRef.child("selectedBackground").setValue(i);
            break;
        case "tagBonus":
            user.selectedBonus = i;
            userRef.child("selectedBonus").setValue(i);
    }
}

```

```

        break;
    case "tagPowerUp":
        user.selectedPowerUp = i;
        userRef.child("selectedPowerUp").setValue(i);
        break;
    }
}
// change color of price to green
public void updateTextColor(int position) {
    shopItemAdapter.updateTextColor(position);
    lvShop.setAdapter(shopItemAdapter);
}
}

```

- public class BossLevelCompleteDialog extends Dialog implements View.OnClickListener {

```

    private final Context context;
    private final int levelID;
    private TextView tvWorldUnlocked;
    private Button btBack;

    public BossLevelCompleteDialog(Context context, int levelID) {
        super(context);
        this.context = context;
        this.levelID = levelID;
    }

    @Override
    protected void onCreate(final Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.dialog_boss_level_complete);
        setCancelable(false);

        tvWorldUnlocked = findViewById(R.id.tvWorldUnlocked);
        tvWorldUnlocked.setVisibility(View.GONE);
        btBack = findViewById(R.id.btBack);
        btBack.setOnClickListener(this);
        setWorldUnlocked();
    }

    // checks if it was a first time completion and give the player a message if so
    @SuppressWarnings("SetTextI18n")
    public void setWorldUnlocked () {
        if (isConnectedToInternet(context)) {
            boolean check = false;
            DatabaseReference userRef =
                FirebaseDatabase.getInstance().getReference("Users").child(user.key);
            // checks for first time completion
            if (!user.levelComplete.get(levelID/5)) check = true;
            user.levelComplete.set(levelID/5, true);
            userRef.child("levelComplete").child(String.valueOf(levelID/5)).setValue(true);
            userRef.child("bestTime").child(String.valueOf(levelID/5)).setValue("0:00");
            //displays a message if it is
            if (check) {
                tvWorldUnlocked.setVisibility(View.VISIBLE);
            }
        }
    }
}

```

```

        if (levelID / 5 + 1 != 6)
            tvWorldUnlocked.setText("WORLD " + (levelID / 5 + 1) + " UNLOCKED!");
        else
            tvWorldUnlocked.setText("LAST LEVEL COMPLETED \n
CONGRATULATIONS!");
    }
}
else {
    tvWorldUnlocked.setVisibility(View.VISIBLE);
    tvWorldUnlocked.setText("CONNECTION LOST! \n PROGRESS NOT SAVED");
}
}
// a go back button
@Override
public void onClick(View view) {
    if (view == btBack) {
        this.dismiss();
        GameScreenActivity tempScreen = (GameScreenActivity) context;
        tempScreen.finish();
    }
}
}
}

```

```

public class DemoGameOverDialog extends Dialog implements View.OnClickListener {

```

```

    private final Context context;
    private Button btBack;

```

```

    // This dialog closes the game and returns to LoginActivity after the player loses the demo level

```

```

    public DemoGameOverDialog(Context context) {
        super(context);
        this.context = context;
    }

```

```

    @Override
    protected void onCreate(final Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.dialog_demo_game_over);
        setCancelable(false);

```

```

        btBack = findViewById(R.id.btBack);
        btBack.setOnClickListener(this);
    }

```

```

    @Override
    public void onClick(View view) {
        if (view == btBack) {
            this.dismiss();
            GameScreenActivity tempScreen = (GameScreenActivity) context;
            tempScreen.finish();
        }
    }
}
}

```

- ```

public class DifficultyDialog extends Dialog implements View.OnClickListener {

 private final Context context;
 private Button btEasy, btNormal, btHard, btConfirm;
 private int diff;
 private TextView tvDisplay;

 public DifficultyDialog(@NonNull Context context) {
 super(context);
 this.context = context;
 }

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.dialog_difficulty);
 setCancelable(false);

 btEasy = findViewById(R.id.btEasy);
 btNormal = findViewById(R.id.btNormal);
 btHard = findViewById(R.id.btHard);
 btConfirm = findViewById(R.id.btConfirm);
 btEasy.setOnClickListener(this);
 btNormal.setOnClickListener(this);
 btHard.setOnClickListener(this);
 btConfirm.setOnClickListener(this);
 tvDisplay = findViewById(R.id.tvDisplay);
 diff = user.difficulty;
 }

 @Override
 public void onClick(View view) {
 // gives information to the player about each difficulty
 if (view == btEasy) {
 diff = 0;
 tvDisplay.setText("-lower obstacle movement speed \n -gain less score");
 }
 if (view == btNormal) {
 diff = 1;
 tvDisplay.setText("-normal obstacle movement \n -gain normal score");
 }
 if (view == btHard) {
 diff = 2;
 tvDisplay.setText("-higher obstacle movement speed \n -gain more score");
 }
 // saves the chosen difficulty to firebase, changes the color of the button in
 LevelSelectActivity
 // to the new color, closes the dialog
 if (view == btConfirm) {
 user.difficulty = diff;
 DatabaseReference userRef =
 FirebaseDatabase.getInstance().getReference("Users").child(user.key);
 userRef.child("difficulty").setValue(user.difficulty);
 LevelSelectActivity levelSelectActivity = (LevelSelectActivity) context;

```

```

 levelSelectActivity.difficultyColor();
 dismiss();
 }
}
}

```

- public class GameOverDialog extends Dialog implements View.OnClickListener {

```

 private final Context context;
 private final GameScreenActivity activity;
 private TextView tvNewHighScore;
 private Button btBack, btRetry;
 private final int score;
 private final String time;
 private final int levelID;

```

```

 public GameOverDialog(Context context, int score, String time, int levelID) {
 super(context);
 this.context = context;
 this.activity = (GameScreenActivity) context;
 this.score = score;
 this.time = time;
 this.levelID = levelID;
 }

```

```

 @SuppressWarnings("SetTextI18n")

```

```

 @Override

```

```

 protected void onCreate(final Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.dialog_game_over);
 setCancelable(false);

```

```

 tvNewHighScore = findViewById(R.id.tvNewHighScore);
 btBack = findViewById(R.id.btBack);
 btRetry = findViewById(R.id.btRetry);
 btBack.setOnClickListener(this);
 btRetry.setOnClickListener(this);
 // displays the score or the time depending on the type of the level
 TextView tvDisplay = findViewById(R.id.tvDisplay);
 if (levelID % 5 != 0) {
 tvDisplay.setText("Total Score: " + score);
 }
 else
 tvDisplay.setText("TIME LEFT: " + time);
 tryToSetNewProgression();
 }

```

```

 @Override

```

```

 public void onClick(View view) {
 // a go back button
 if (view == btBack) {
 this.dismiss();
 this.activity.finish();
 }
 }

```



```

// a retry button
else if (view == btRetry) {
 if (isConnectedToInternet(context)) {
 this.dismiss();
 GamePanel newGamePanel = new GamePanel(activity);
 this.activity.onRetry(newGamePanel);
 activity.playMusic();
 activity.getGamePanel().getPowerUp().restartPowerUpTimer();
 }
 else
 Toast.makeText(context, "Please connect to the internet first",
Toast.LENGTH_SHORT).show();
}
}

// checks for if the new score is better than the old score
public void tryToSetNewProgression() {
 if (isConnectedToInternet(context)) {
 DatabaseReference userRef =
FirebaseDatabase.getInstance().getReference("Users").child(user.key);
 user.deaths++;
 userRef.child("deaths").setValue(user.deaths);
 if (levelID % 5 != 0 && score > user.hsl.get(levelID)) {
 user.hsl.set(levelID, score);
 userRef.child("hsl").child(String.valueOf(levelID)).setValue(score);
 displayNewMessage(true);
 }
 else if (levelID % 5 == 0 && isBestTime(user.bestTime.get(levelID/5))) {
 user.bestTime.set(levelID/5, time);
 userRef.child("bestTime").child(String.valueOf(levelID/5)).setValue(time);
 displayNewMessage(false);
 }
 }
 else
 Toast.makeText(context, "Please reconnect to the internet! \n progression will not be
saved without internet connection! ", Toast.LENGTH_SHORT).show();
}

// checks for if the new time is better than the old time
public boolean isBestTime(String oldBestTime) {
 String[] timeSplit = time.split(":");
 String[] oldBestTimeSplit = oldBestTime.split(":");
 int minute1 = Integer.parseInt(timeSplit[0]);
 int second1 = Integer.parseInt(timeSplit[1]);
 int minute2 = Integer.parseInt(oldBestTimeSplit[0]);
 int second2 = Integer.parseInt(oldBestTimeSplit[1]);

 return (minute1 < minute2 || (minute1 == minute2 && second1 < second2));
}

// displays a text to the player if they got a new high score or a better time
@SuppressWarnings("SetTextI18n")
public void displayNewMessage(boolean type) {
 // true = normal, false = boss
 if (type)
 tvNewHighScore.setText("New High Score!");
 else
 tvNewHighScore.setText("New Best Time!");
}

```

```

 tvNewHighScore.setVisibility(View.VISIBLE);
 }
}

```

- public class LevelInfoDialog extends Dialog implements View.OnClickListener {

```

 private final Context context;
 private final int levelID;
 private final Bundle bitmapArrays;
 private Button btSong, btClose, btPlay;

 public LevelInfoDialog(Context context, int levelID, Bundle bitmapArrays) {
 super(context);
 this.context = context;
 this.levelID = levelID;
 this.bitmapArrays = bitmapArrays;
 }
 @SuppressWarnings("SetTextI18n")
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.dialog_level_info);
 setCancelable(false);

 TextView tvLevel = findViewById(R.id.tvLevel);
 tvLevel.setText("LEVEL " + levelID);
 TextView tvScoreOrTime = findViewById(R.id.tvScoreOrTime);
 btSong = findViewById(R.id.btSong);
 btClose = findViewById(R.id.btClose);
 btPlay = findViewById(R.id.btPlay);
 btPlay.setOnClickListener(this);
 btClose.setOnClickListener(this);
 btSong.setOnClickListener(this);
 // checks for if previous level is done
 if (isPrevHighScoreOrCompleted()) {
 switch (levelID) {
 // sets the song text and the text for score or time
 case 0: default: btSong.setText("Song - Stay Inside Me - OcularNebula"); break;
 case 1: btSong.setText("Song - Partyt Är Igång! - Bossfight");
 tvScoreOrTime.setText("Highest Score: " + user.hsl.get(1)); break;
 case 2: btSong.setText("Song - Vs Susie - NyxTheShield");
 tvScoreOrTime.setText("Highest Score: " + user.hsl.get(2)); break;
 case 3: btSong.setText("Song - The Vengeance Of Those Forgotten In Darkness - Jami Lynne");
 tvScoreOrTime.setText("Highest Score: " + user.hsl.get(3)); break;
 case 4: btSong.setText("Song - Dimension - Creo");
 tvScoreOrTime.setText("Highest Score: " + user.hsl.get(4)); break;
 case 5: btSong.setText("Song - You Were Wrong. Go Back - Jami Lynne");
 tvScoreOrTime.setText(setBossLevelText()); break;
 case 6: btSong.setText("Song - Peer Gynt - cYsmix");
 tvScoreOrTime.setText("Highest Score: " + user.hsl.get(6)); break;
 case 7: btSong.setText("Song - Lost Memories - MULTI BGM STUDIO");
 tvScoreOrTime.setText("Highest Score: " + user.hsl.get(7)); break;
 case 8: btSong.setText("Song - Bready Steady Go - Pedro Silva");
 tvScoreOrTime.setText("Highest Score: " + user.hsl.get(8)); break;
 }
 }
 }
}

```

```

 case 9: btSong.setText("Song - Betrayal Of Fear - Goukisan");
tvScoreOrTime.setText("Highest Score: " + user.hsl.get(9)); break;
 case 10: btSong.setText("Song - World's End Valentine - Pedro Silva");
tvScoreOrTime.setText(setBossLevelText()); break;
 case 11: btSong.setText("Song - Ghost House - schtiffles");
tvScoreOrTime.setText("Highest Score: " + user.hsl.get(11)); break;
 case 12: btSong.setText("Song - METALOVANIA - Metal Socks");
tvScoreOrTime.setText("Highest Score: " + user.hsl.get(12)); break;
 case 13: btSong.setText("Song - Swirly 1000x - Pedro Silva");
tvScoreOrTime.setText("Highest Score: " + user.hsl.get(13)); break;
 case 14: btSong.setText("Song - Iron God Sakupen Hell Yes RMX - mr-jazzman");
tvScoreOrTime.setText("Highest Score: " + user.hsl.get(14)); break;
 case 15: btSong.setText("Song - Tee-Hee Time - Pedro Silva");
tvScoreOrTime.setText(setBossLevelText()); break;
 case 16: btSong.setText("Song - Theory Of Everything 3 - DJ Nate");
tvScoreOrTime.setText("Highest Score: " + user.hsl.get(16)); break;
 case 17: btSong.setText("Song - Lingering Light Of The Setting Sun - Snow");
tvScoreOrTime.setText("Highest Score: " + user.hsl.get(17)); break;
 case 18: btSong.setText("Song - It Means Everything - Pedro Silva");
tvScoreOrTime.setText("Highest Score: " + user.hsl.get(18)); break;
 case 19: btSong.setText("Song - Sonic Blaster - F-777");
tvScoreOrTime.setText("Highest Score: " + user.hsl.get(19)); break;
 case 20: btSong.setText("Song - Goldenvengeance - Jami Lynne");
tvScoreOrTime.setText(setBossLevelText()); break;
 case 21: btSong.setText("Song - Theory Of Everything - Domyeah");
tvScoreOrTime.setText("Highest Score: " + user.hsl.get(21)); break;
 case 22: btSong.setText("Song - Dark_End - RsnowJ");
tvScoreOrTime.setText("Highest Score: " + user.hsl.get(22)); break;
 case 23: btSong.setText("Song - You Can't Go Back - Jami lynne");
tvScoreOrTime.setText("Highest Score: " + user.hsl.get(23)); break;
 case 24: btSong.setText("Song - Nine Circles - NightKilla");
tvScoreOrTime.setText("Highest Score: " + user.hsl.get(24)); break;
 case 25: btSong.setText("Song - Phobiawave - Jami Lynne");
tvScoreOrTime.setText(setBossLevelText()); break;
 }
 btSong.setVisibility(View.VISIBLE);
 tvScoreOrTime.setVisibility(View.VISIBLE);
 btPlay.setVisibility(View.VISIBLE);
}
else {
 // sets the text for requirements
 TextView tvShowReq = findViewById(R.id.tvShowReq);
 tvShowReq.setVisibility(View.VISIBLE);
 TextView tvReq = findViewById(R.id.tvReq);
 tvReq.setText(requirements());
 tvReq.setVisibility(View.VISIBLE);
}
}
// checks for if previous level is done
public boolean isPrevHighScoreOrCompleted() {
 if (levelID % 5 == 1) {
 return user.levelComplete.get((levelID-1)/5);
 }
}

try {
 return (user.hsl.get(levelID - 1) >= scoreReq());
}

```

```

 } catch (IndexOutOfBoundsException e) {
 return true;
 }
}
// set the text to show best time a player's had on a boss level
public String setBossLevelText() {
 if (user.levelComplete.get(levelID/5))
 return "Level Completed";
 else return "Lowest Time: " + user.bestTime.get(levelID/5);
}
// returns the required points to beat the previous level from the number of level
public int scoreReq() {
 return 3000 + (levelID - 1) * 1000;
}
// tells the player the requirements to unlocking the level if it is not unlocked
public String requirements() {
 if (levelID % 5 != 1) {
 int score = scoreReq();
 return "- All previous levels unlocked \n- Score of " + score + " on level " + (levelID -
1);
 }
 return "- All previous levels unlocked \n- Complete world " + (levelID - 1) / 5;
}
@Override
public void onClick(View view) {
 if (view == btClose)
 // close
 dismiss();
 else if (view == btPlay) {
 if (isConnectedToInternet(context)) {
 // starts the game activity and adds the bitmaps to it's extras
 Intent intent = new Intent(view.getContext(), GameScreenActivity.class);
 intent.putExtra("LevelID", levelID);
 Log.i(TAG, "onClick: LEVEL - " + levelID);
 intent.putExtras(bitmapArrays);
 context.startActivity(intent);
 dismiss();
 }
 else
 Toast.makeText(context, "Please connect to a stable internet connection",
Toast.LENGTH_SHORT).show();
 }
 else if (view == btSong) {
 String url;
 // changes url to the song link from youtube
 switch (levelID) {
 case 0: default: url = "https://youtu.be/ix6e9h40qMg"; break;
 case 1: url = "https://youtu.be/AECcMhgSEsQ"; break;
 case 2: url = "https://youtu.be/eJHsEmtWiyM"; break;
 case 3: url = "https://youtu.be/9hqWe-Fr4ko"; break;
 case 4: url = "https://youtu.be/MrD05HVGVIQ"; break;
 case 5: url = "https://youtu.be/TyHXvJdyMUo"; break;
 case 6: url = "https://youtu.be/w4dLTLW6dJ0"; break;
 case 7: url = "https://youtu.be/qN56222aBWQ"; break;
 case 8: url = "https://youtu.be/QL_MA9kQ-0o"; break;
 case 9: url = "https://youtu.be/eXv0tqBtv3E"; break;
 }
 }
}

```

```

 case 10: url = "https://youtu.be/rIQd9qWKjLM"; break;
 case 11: url = "https://youtu.be/UHGvNoRPCQA"; break;
 case 12: url = "https://youtu.be/H5rbS0VuX1o"; break;
 case 13: url = "https://youtu.be/Hww-EBjdKvg"; break;
 case 14: url = "https://youtu.be/nTUVLuKAAkA"; break;
 case 15: url = "https://youtu.be/n9gm9IP_4Gk"; break;
 case 16: url = "https://youtu.be/fiAz__CBDIU"; break;
 case 17: url = "https://youtu.be/6VXd_Axc4oc"; break;
 case 18: url = "https://youtu.be/8wKWwUzgCE8"; break;
 case 19: url = "https://youtu.be/lvYD-HLAI8E"; break;
 case 20: url = "https://youtu.be/PqQC_gI5k0w"; break;
 case 21: url = "https://youtu.be/0T3XU6zh5No"; break;
 case 22: url = "https://youtu.be/qhKDQIPfsJM"; break;
 case 23: url = "https://youtu.be/Pe---DN2TmY"; break;
 case 24: url = "https://youtu.be/gvdH-gEOSB8"; break;
 case 25: url = "https://youtu.be/kdVU4FbSmu0"; break;
 }
 // implicit intent to play the song
 Intent link = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
 // tries to open the song in the youtube app and opens it on google if youtube app was
 not found
 link.setPackage("com.google.android.youtube");
 try {
 context.startActivity(link);
 } catch (ActivityNotFoundException ex) {
 context.startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse(url)));
 }
}
}
}

```

- public class ObstInfoDialog extends Dialog {

```

 private final Context context;
 private LinearLayout obstaclesList;
 private TextView tvTitle, tvDesc;

```

```

 public ObstInfoDialog(Context context) {
 super(context);
 this.context = context;
 }

```

@Override

```

 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.dialog_obst_info);
 setCancelable(false);
 }

```

```

 Button btClose = findViewById(R.id.btClose);
 btClose.setOnClickListener(view -> dismiss());
 obstaclesList = findViewById(R.id.obstacles_list);
 tvTitle = findViewById(R.id.tvTitle);
 tvDesc = findViewById(R.id.tvDesc);

```

```

LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(
 LinearLayout.LayoutParams.WRAP_CONTENT,
 LinearLayout.LayoutParams.WRAP_CONTENT);
params.setMargins(25, 0, 25, 0);
params.width = 150;
params.height = 150;
// turn every obstacle into a view and add it to the list
for (int i = 0; i < 12; i++) {
 ImageButton ibObstacle;
 ibObstacle = new ImageButton(context);
 if (i != 5)
 ibObstacle.setBackgroundColor(Color.parseColor(getColor(i)));
 else {
 // changes the picture of warning to it's bitmap because it doesn't use a color
 @SuppressWarnings("UseCompatLoadingForDrawables") Drawable originalDrawable =
context.getResources().getDrawable(R.drawable.warning_active, null);
 Bitmap originalBitmap = ((BitmapDrawable) originalDrawable).getBitmap();
 Bitmap resizedBitmap = Bitmap.createScaledBitmap(originalBitmap, 150, 150,
false);
 Drawable resizedDrawable = new BitmapDrawable(context.getResources(),
resizedBitmap);
 ibObstacle.setBackground(resizedDrawable);
 }
 ibObstacle.setLayoutParams(params);
 ibObstacle.setOnClickListener(view -> {
 tvTitle.setVisibility(View.VISIBLE);
 tvDesc.setVisibility(View.VISIBLE);
 int index;
 for (int j = 0; j < 12; j++) {
 if (obstaclesList.getChildAt(j) == view) {
 index = j;
 setInfoTitle(index);
 setInfoDesc(index);
 break;
 }
 }
 });
 obstaclesList.addView(ibObstacle);
}
}
// get the color for each object
public String getColor(int i) {
 switch (i) {
 case 0: return "#800000";
 case 1: return "#00FF00";
 case 2: return "#9600FF";
 case 3: return "#646464";
 case 4: return "#326496";
 case 6: return "#03A9F4";
 case 7: return "#FF0000";
 case 8: return "#70B596";
 case 9: return "#CB9813";
 case 10: return "#FF9632";
 case 11: return "#FF33FF";
 default: return "";
 }
}

```

```

 }
}
// set title for each object
public void setInfoTitle (int j) {
 switch (j) {
 case 0: tvTitle.setText(R.string.alpha); break;
 case 1: tvTitle.setText(R.string.diagonal); break;
 case 2: tvTitle.setText(R.string.vertical); break;
 case 3: tvTitle.setText(R.string.horizontal); break;
 case 4: tvTitle.setText(R.string.gap); break;
 case 5: tvTitle.setText(R.string.warning); break;
 case 6: tvTitle.setText(R.string.floor); break;
 case 7: tvTitle.setText(R.string.moving_gap); break;
 case 8: tvTitle.setText(R.string.cross); break;
 case 9: tvTitle.setText(R.string.wall); break;
 case 10: tvTitle.setText(R.string.wall_floor); break;
 case 11: tvTitle.setText(R.string.sensor); break;
 }
}
// set description for each object
public void setInfoDesc (int j) {
 String desc;
 switch (j) {
 case 0: desc = "This obstacle will become invisible if you move, but will turn visible again if you stand still."; break;
 case 1: desc = "This obstacle will move diagonally from the top."; break;
 case 2: desc = "This obstacle will move vertically from the top."; break;
 case 3: desc = "This obstacle will move horizontally from either sides."; break;
 case 4: desc = "This obstacles will create 2 object with a gap between them that you will need to go through."; break;
 case 5: desc = "This obstacle will give you a short warning before summoning a square that will hit you if you stand on it."; break;
 case 6: desc = "This obstacle will move down vertically and then bounce back up when they hit the bottom"; break;
 case 7: desc = "This obstacle will have 2 object moving vertically down, and the gap between them will move from side to side."; break;
 case 8: desc = "This obstacle is 2 objects that move together and cross each other."; break;
 case 9: desc = "This obstacle is an object that goes diagonally from top to bottom and bounces on the walls when it hits them."; break;
 case 10: desc = "This obstacle is an object that goes diagonally from top to bottom and bounces on the walls and the bottom floor when it hits them."; break;
 case 11: desc = "This obstacle will move based on the phone's position."; break;
 default: desc = ""; break;
 }
 tvDesc.setText(desc);
}
}
}

```

- public class PurchaseConfirmDialog extends Dialog implements View.OnClickListener {
 

```

private final Context context;
private final ShopBuyFragment frag;
private final int position;

```

```

private final ShopItem shopItem;
private final String tag;
private Button btConfirm, btClose;
private final DatabaseReference userRef;

public PurchaseConfirmDialog(Context context, ShopBuyFragment frag, ShopItem
shopItem, int position, String tag) {
 super(context);
 this.context = context;
 this.frag = frag;
 this.shopItem = shopItem;
 this.position = position;
 this.tag = tag;
 this.userRef = FirebaseDatabase.getInstance().getReference("Users").child(user.key);
}

@SuppressLint("SetTextI18n")
@Override
protected void onCreate(final Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.dialog_purchase_confirm);
 setCancelable(false);

 btConfirm = findViewById(R.id.btConfirm);
 btClose = findViewById(R.id.btReturn);
 btConfirm.setOnClickListener(this);
 btClose.setOnClickListener(this);

 ImageView ivProduct = findViewById(R.id.ivProduct);
 ivProduct.setImageBitmap(shopItem.getBitmap());
 TextView tvDisplay = findViewById(R.id.tvDisplay);
 tvDisplay.setText("Buy " + shopItem.getTitle() + " for " + shopItem.getPrice() + " coins?");
}

@SuppressLint("NonConstantResourceId")
public void buyItem() {
 // change the bought item to true in firebase
 switch (tag) {
 case "tagIcon":
 user.iconOwned.set(position, true);
 userRef.child("iconOwned").child(String.valueOf(position)).setValue(true);
 frag.updateTextColor(position);
 break;
 case "tagBackground":
 user.backgroundOwned.set(position, true);
 userRef.child("backgroundOwned").child(String.valueOf(position)).setValue(true);
 frag.updateTextColor(position);
 break;
 case "tagBonus":
 user.bonusOwned.set(position, true);
 userRef.child("bonusOwned").child(String.valueOf(position)).setValue(true);
 frag.updateTextColor(position);
 break;
 case "tagPowerUp":
 user.powerUpOwned.set(position, true);
 userRef.child("powerUpOwned").child(String.valueOf(position)).setValue(true);
 }
}

```



```

 frag.updateTextColor(position);
 break;
 }
 // set item "owned" field to true
 shopItem.setOwned(true);
 // reduces the player's coins and displays it
 user.coins = user.coins - shopItem.getPrice();
 userRef.child("coins").setValue(user.coins);
 user.itemsOwned++;
 userRef.child("itemsOwned").setValue(user.itemsOwned);
 frag.setCoinsDisplay();
}

@Override
public void onClick(View view) {
 if (view == btConfirm) {
 // checks if the player has enough coins
 if (user.coins < shopItem.getPrice())
 Toast.makeText(context, "You don't have enough coins!",
 Toast.LENGTH_SHORT).show();
 else {
 context.startService(new Intent(context, VibrationService.class));
 buyItem();
 Toast.makeText(context, "Item successfully purchased!",
 Toast.LENGTH_SHORT).show();
 dismiss();
 }
 }
 else if (view == btClose) {
 dismiss();
 }
}
}

```

- public class ShopInfoDialog extends Dialog {

```
 private final String type;
```

```
 public ShopInfoDialog(@NonNull Context context, String type) {
 super(context);
 this.type = type;
 }

```

```
 @SuppressWarnings("SetTextI18n")

```

```
 @Override

```

```
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.dialog_shop_info);
 setCancelable(false);
 }

```

```

 Button btClose = findViewById(R.id.btClose);
 btClose.setOnClickListener(view -> dismiss());
 TextView tvTitle = findViewById(R.id.tvTitle);
 TextView tvInfo = findViewById(R.id.tvInfo);
 }

```

```
 // set the text to give information to the player about the properties of the item that they
```

```

are buying
 switch (type) {
 case "tagIcon": tvTitle.setText(R.string.icon); tvInfo.setText("Customize the way your
character looks!"); break;
 case "tagBackground": tvTitle.setText(R.string.background); tvInfo.setText("Customize
the way the background looks!"); break;
 case "tagBonus": tvTitle.setText(R.string.bonus); tvInfo.setText("A bonus block is an
item that you can pick up occasionally during a level, " +
 "up a bonus can either make the level easier or harder, so be careful if you
decide to pick it up \n !Customize the way your bonus block looks!"); break;
 case "tagPowerUp": tvTitle.setText(R.string.power_up); tvInfo.setText("Power ups can
help you during a level and can be reactivated " +
 "when their cool down is over! \n Purchase a power up!"); break;
 }
}
}
}

```

- ```

public class AlphaObj extends RectGame {
    private final float initVelY;

    public AlphaObj(Context context, String color, int width, int height) {
        super(context, color, width, height);
        this.pointGame.x = rnd.nextInt(Statics.width);
        initVelY = (1 + rnd.nextFloat());
        changeVelocities();
        paint.setAlpha(255);
    }
    // reduces alpha when player is moving, increases it when player is not moving
    public void changeAlpha() {
        int currAlpha = this.paint.getAlpha();
        int newAlpha;
        Player player = ((GameScreenActivity) context).getGamePanel().getPlayer();
        if (player.getPointGame().velX == 0 && player.getPointGame().velY == 0)
            newAlpha = currAlpha + 25;
        else
            newAlpha = currAlpha - 25;
        newAlpha = Math.min (255, Math.max(0, newAlpha));
        this.paint.setAlpha(newAlpha);
    }

    @Override
    public void draw(Canvas canvas) {
        setRect(this.rect);
        canvas.drawRect(rect, paint);
    }

    @Override
    public void update() {
        this.moveObst();
        this.changeAlpha();
    }

    @Override

```

```

    public void changeVelocities() {
        this.pointGame.velX = 0;
        this.pointGame.velY = this.pointGame.speed * initVelY;
    }
}

```

- ```

public class BonusObj extends RectGame {
 private int lowestY;
 private int status;
 final Handler handler;

 public BonusObj(Context context, int bitmapID, int width, int height) {
 super(context, bitmapID, width, height);
 handler = new Handler(Looper.getMainLooper());
 status = 0;
 this.pointGame.velX = 0;
 this.pointGame.velY = pointGame.speed;
 }

 public void setStatus(int status) {
 this.status = status;
 }
 // puts the object slightly above the screen on a random X value
 // gives it a random value that will be the lowest point it can be for "case 2" in update
 method
 public void setRandomPoint() {
 lowestY = Statics.height/5 + rnd.nextInt(3* Statics.height/5);
 this.pointGame.x = width/2 + rnd.nextInt(-width + Statics.width);
 this.pointGame.y = -this.width/2;
 }

 public int setSleepTime() {
 return (5000 + rnd.nextInt(20000));
 }

 @Override
 public void draw(Canvas canvas) {
 if(status == 2 || status == 3 || status == 4) {
 canvas.drawBitmap(bitmap, this.pointGame.x - (float) this.width / 2, this.pointGame.y -
(float) this.height / 2, null);
 setRect(this.rect);
 }
 }

 @Override
 public void update() {
 switch (status) {
 // change status to 1, then activate handler once and wait to be respawned,
 case 0:
 status++;
 handler.postDelayed(this::respawn, setSleepTime()); break;
 // waiting to be respawned, handler will make status go to 2

```

```

 case 1: break;
 // object is moving and detecting lowest point, change to 3
 case 2:
 moveObst();
 if (this.pointGame.y >= lowestY) {
 status++;
 } break;
 // keeps the object at lowest y, despawn it if not touched for the delay time
 case 3:
 status++;
 handler.postDelayed(() -> {
 if (status > 2) {
 kill();
 respawn();
 }
 }, 7000); break;
 // object not moving
 case 4: break;
 }
}

// respawns the object when the player touches it or when enough time passed and the
// player didn't touch it
public void respawn() {
 setRandomPoint();
 if (status == 1)
 status++;
 else
 status = 0;
}
@Override
public void changeVelocities() {
}
}

```

- public class BounceObj extends CrossingSingleObj {
 private boolean touchedBottom;

 public BounceObj(Context context, String color, int width, int height) {
 super(context, color, width, height);
 }
 // same as CrossingSingleObj, but checks for collision with the ground
 @Override
 public void update() {
 if (pointGame.y >= Statics.height - height/2) {
 this.pointGame.velY \*= -1;
 touchedBottom = true;
 }
 super.update();
 }

 @Override

```

 public void changeVelocities() {
 super.changeVelocities();
 if (touchedBottom)
 pointGame.velY = -pointGame.speed;
 else
 pointGame.velY = pointGame.speed;
 }
}

```

- public class CrossingObj extends RectGame{

```

 private final Rect rect1, rect2;
 private int movingDirection = 0;

```

```

 public CrossingObj(Context context, String color, int width, int height) {
 super(context, color, width, height);
 this.pointGame.x = this.width/2;
 this.pointGame.y = -this.height;
 changeVelocities();
 rect1 = new Rect();
 rect2 = new Rect();
 }

```

// rectangle 1 is created on the object's point, rectangle 2 is created on a horizontally mirrored point

```

 public void setRectangles (Rect rectangle1, Rect rectangle2) {
 int l1 = this.pointGame.x-width/2;
 int t1 = this.pointGame.y-this.height/2;
 int r1 = this.pointGame.x+this.width/2;
 int b1 = this.pointGame.y+this.height/2;
 rectangle1.set(l1,t1,r1,b1);
 int revX = Statics.width - this.pointGame.x;
 int l2 = revX-this.width/2;
 int t2 = this.pointGame.y-this.height/2;
 int r2 = revX+this.width/2;
 int b2 = this.pointGame.y+this.height/2;
 rectangle2.set(l2,t2,r2,b2);
 }

```

@Override

```

 public boolean collision(Player p) {
 return Rect.intersects(this.rect1, p.getRectangle()) || Rect.intersects(rect2,
p.getRectangle());
 }

```

@Override

```

 public void draw(Canvas canvas) {
 setRectangles(rect1, rect2);
 canvas.drawRect(rect1, paint);
 canvas.drawRect(rect2, paint);
 }

```

// every time the objects hit the wall, "movingDirection" changes and make the object's point go in the other direction horizontally

// the object keeps moving down vertically

```

@Override
public void update() {
 if (movingDirection == 0) {
 pointGame.velX = pointGame.speed;
 if (pointGame.x > Statics.width-this.width/2)
 movingDirection = 1;
 }
 if (movingDirection == 1) {
 pointGame.velX = -pointGame.speed;
 if (pointGame.x < this.width/2)
 movingDirection = 0;
 }
 this.moveObst();
}
@Override
public void changeVelocities() {
 if (movingDirection == 0)
 pointGame.velX = pointGame.speed;
 else if (movingDirection == 1)
 pointGame.velX = -pointGame.speed;
 this.pointGame.velY= (float)pointGame.speed/2;
}
}

```

- public class CrossingSingleObj extends RectGame{

```

private int movingDirection;
private final float makeRandomVelX;

```

```

public CrossingSingleObj(Context context, String color, int width, int height) {
 super(context, color, width, height);
 this.pointGame.x = width/2 + rnd.nextInt(-1*width + Statics.width);
 this.pointGame.y = -width/2;
 makeRandomVelX = this.pointGame.speed * (1 + rnd.nextFloat());
 Log.i(TAG, "CrossingSingleObj: " + makeRandomVelX);
 if (rnd.nextInt(2) == 0)
 movingDirection = 0;
 else
 movingDirection = 1;
 this.pointGame.velY = this.pointGame.speed;
}

```

```

@Override
public void draw(Canvas canvas) {
 setRect(this.rect);
 canvas.drawRect(rect, paint);
}

```

```

@Override
public void update() {
 // checks for if the object touched a wall and changes its horizontal direction
 if (movingDirection == 0) {
 if (pointGame.x > Statics.width-this.width/2)
 movingDirection = 1;
 }
}

```

```

 }
 if (movingDirection == 1) {
 if (pointGame.x < this.width/2)
 movingDirection = 0;
 }
 changeVelocities();
 this.moveObst();
}

@Override
public void changeVelocities() {
 float newSpeed;
 switch (pointGame.speed) {
 case 0: newSpeed = 0; break;
 case 3: newSpeed = 3; break;
 default: newSpeed = makeRandomVelX; break;
 }
 pointGame.velY = pointGame.speed;
 if (movingDirection == 0) {
 pointGame.velX = newSpeed;
 }
 else if (movingDirection == 1) {
 pointGame.velX = -newSpeed;
 }
}
}
}

```

- public class DownUpObj extends RectGame {
 

```

private boolean touchedBottom;

public DownUpObj(Context context, String color, int width, int height) {
 super(context,color, width, height);
 this.pointGame.x = width/2 + rnd.nextInt(-1*width + Statics.width);
 this.pointGame.y = -this.width/2 - rnd.nextInt(200);
 this.pointGame.velX = 0;
 this.pointGame.velY = pointGame.speed;
}

@Override
public void draw(Canvas canvas) {
 setRect(this.rect);
 canvas.drawRect(rect, paint);
}

@Override
public void update() {
 this.moveObst();
 // checks for ground collision and change the direction of the vertical velocity when
 detects it
 if (pointGame.y >= Statics.height - height/2) {
 this.pointGame.velY *= -1;
 touchedBottom = true;
 }

```

```

 }

 @Override
 public void changeVelocities() {
 if (touchedBottom)
 pointGame.velY = -pointGame.speed;
 else
 pointGame.velY = pointGame.speed;
 }
}

```

- ```

public class GapObj extends RectGame {
    private final Rect rect1, rect2;

    public GapObj(Context context, String color, int width, int height) {
        super(context, color, width, height);
        this.pointGame.x = width/2 + rnd.nextInt(Statics.width-width);
        this.pointGame.y = -height;
        this.pointGame.velX = 0;
        this.pointGame.velY = pointGame.speed;
        rect1 = new Rect();
        rect2 = new Rect();
    }

    public void setRectangles (Rect rectangle1, Rect rectangle2) {
        // puts a "fake" rectangle around the gamePoint, rectangle 1 is to the left of that
        // rectangle,
        // rectangle 2 is to the right of that rectangle
        int l1 = 0;
        int t1 = this.pointGame.y-this.height/2;
        int r1 = this.pointGame.x-this.width/2;
        int b1 = this.pointGame.y+this.height/2;
        rectangle1.set(l1,t1,r1,b1);
        int l2 = this.pointGame.x + this.width/2;
        int t2 = this.pointGame.y-this.height/2;
        int r2 = Statics.width;
        int b2 = this.pointGame.y+this.height/2;
        rectangle2.set(l2,t2,r2,b2);
    }

    @Override
    public boolean collision (Player p) {
        return Rect.intersects(this.rect1, p.getRectangle()) || Rect.intersects(rect2,
        p.getRectangle());
    }

    @Override
    public void draw(Canvas canvas) {
        setRectangles(rect1, rect2);
        canvas.drawRect(rect1, paint);
        canvas.drawRect(rect2, paint);
    }
}

```



```

@Override
public void update() {
    this.moveObst();
}

@Override
public void changeVelocities() {
    this.pointGame.velY = pointGame.speed;
}
}

```

- ```

public class MovingGapObj extends GapObj {
 // 0 - left 1 - right
 private int movingDirection;
 public MovingGapObj(Context context, String color, int width, int height) {
 super(context, color, width, height);
 movingDirection = rnd.nextInt(2);
 this.width *= (0.25 + (1+ rnd.nextFloat()));
 }

 @Override
 public void update() {
 super.update();
 // also checks for if the "fake" rectangle is about to hit the wall, and change its horizontal
 moving direction
 if (movingDirection == 0) {
 pointGame.velX = -pointGame.speed;
 if (pointGame.x < 300) {
 movingDirection = 1;
 }
 }
 if (movingDirection == 1) {
 pointGame.velX = pointGame.speed;
 if (pointGame.x > Statics.width-300) {
 movingDirection = 0;
 }
 }
 }

 @Override
 public void changeVelocities() {
 pointGame.velY = pointGame.speed;
 if (movingDirection == 0) {
 pointGame.velX = -pointGame.speed;
 }
 else if (movingDirection == 1) {
 pointGame.velX = pointGame.speed;
 }
 }
}

```

- ```

public class MovingRectObj extends RectGame {

```

```

private final int type;
private final float rndFX = rnd.nextFloat();
private final float rndFY = 1 + rnd.nextFloat();
private boolean rightSide;
private int horizontalMovement;

public MovingRectObj(Context context, String color, int width, int height) {
    super(context, color, width, height);
    // this object will be randomly chosen as 1 of the 3: "Diagonal Obstacle", "Vertical
Obstacle", "Horizontal Obstacle"
    // type: 0 = diagonal from top, 1 = vertical from top, 2 = horizontal from either sides
    type = rnd.nextInt(3);
    switch (type) {
        case 0:
            this.pointGame.y = -this.height;
            this.pointGame.x = -width + rnd.nextInt(2*width + Statics.width);
            if (this.pointGame.x >= Statics.width/2)
                rightSide = true;
            break;
        case 1:
            this.paint.setColor(Color.parseColor("#9600FF"));
            this.pointGame.y = -this.height;
            this.pointGame.x = rnd.nextInt(Statics.width);
            break;
        case 2:
            this.paint.setColor(Color.parseColor("#646464"));
            horizontalMovement = rnd.nextInt(2);
            if (horizontalMovement == 0) {
                this.pointGame.x = -2 * width;
            }
            else {
                this.pointGame.x = Statics.width + 2 * width;
            }
            this.pointGame.y = rnd.nextInt(Statics.height);
            break;
    }
    changeVelocities();
}

// makes it so if the object spawned on the right side, it will move left horizontally and vice
versa
public void setDiagonalVelocity() {
    this.pointGame.velX = rndFX * this.pointGame.speed;
    this.pointGame.velY = rndFY * this.pointGame.speed;
    if (rightSide) {
        this.pointGame.velX *= -1;
    }
}

// sets the velocity for vertical points
public void setVerticalVelocity() {
    this.pointGame.velX = 0;
    this.pointGame.velY = this.pointGame.speed;
}

// sets the velocity of the horizontal object based on if it should go left or right
public void setHorizontalVelocity() {
    this.pointGame.velX = this.pointGame.speed;
}

```

```

        if (horizontalMovement == 1) {
            this.pointGame.velX *= -1;
        }
    }
    @Override
    public void draw(Canvas canvas) {
        setRect(this.rect);
        canvas.drawRect(rect, paint);
    }

    @Override
    public void update() {
        this.moveObst();
    }

    @Override
    public void changeVelocities() {
        switch (type) {
            case 0: setDiagonalVelocity(); break;
            case 1: setVerticalVelocity(); break;
            case 2: setHorizontalVelocity(); break;
        }
    }
}

```

- ```

public class Player extends RectGame {
 public Player(Context context, int bitmapID, int width, int height) {
 super(context, bitmapID, width, height);
 pointGame = new PointGame(Statics.width/2, (Statics.height*4)/5);
 pointGame.velX = 0; pointGame.velY = 0; pointGame.speed = 17;
 }
 @Override
 public void draw(Canvas canvas) {
 canvas.drawBitmap(bitmap, this.pointGame.x - (float) this.width / 2, this.pointGame.y -
(float) this.height / 2, null);
 pointGame.velX = 0; pointGame.velY = 0;
 }
 public void createBitmap() {
 bitmap = Bitmap.createScaledBitmap(bitmap, width, height, true);
 }
 @Override
 public void update() {
 setRect(this.rect);
 }
 @Override
 public void changeVelocities() {
 }
}

```
- ```

public class PointGame extends Point {
    protected float velX, velY;

```

```

protected int speed;
protected int initialSpeed;

public PointGame(int x, int y) {
    super(x, y);
    setSpeedFromDifficulty();
}

public float getVelX() {
    return velX;
}
public void setVelX(float velX) {
    this.velX = velX;
}
public float getVelY() {
    return velY;
}
public void setVelY(float velY) {
    this.velY = velY;
}
public int getSpeed() {
    return speed;
}
public void setSpeed(int speed) {
    this.speed = speed;
}
// each point will have a speed based on the player's chosen difficulty
public void setSpeedFromDifficulty() {
    switch (user.difficulty) {
        case 0: speed = 10; break;
        case 1: default: speed = 15; break;
        case 2: speed = 20; break;
    }
    initialSpeed = speed;
}
}

```

- ```

public class PositionSensorObj extends RectGame {
 public PositionSensorObj(Context context, String color, int width, int height) {
 super(context, color, width, height);
 pointGame.velY = pointGame.speed;
 pointGame.x = this.width/2 + rnd.nextInt(Statics.width + this.width);
 }
 @Override
 public void draw(Canvas canvas) {
 setRect(this.rect);
 canvas.drawRect(rect, paint);
 }

 @Override
 public void update() {
 // moves the object's x value based on the phone's rotation
 float deltaX = (float) (0.3 * pointGame.speed * ((GameScreenActivity)

```

```

context).getGamePanel().getGameSensors().getXRotation());
 // makes it so the object can't move out of the screen
 this.pointGame.x = (int)Math.min(Statics.width-this.width/2f, Math.max(this.width/2f,
this.pointGame.x - deltaX));
 this.pointGame.y += pointGame.velY;
}
@Override
public void changeVelocities() {
 pointGame.velY = pointGame.speed;
}
}

```

- public abstract class RectGame {
 

```

protected Context context;
protected Bitmap bitmap;
protected PointGame pointGame;
protected int width, height;
protected Rect rect;
protected Random rnd;
protected Paint paint;
// for objects that are images
public RectGame(Context context, int bitmapID, int width, int height) {
 this.context = context;
 this.width = width;
 this.height = height;
 Bitmap temp;
 temp = BitmapFactory.decodeResource(context.getResources(), bitmapID);
 temp = Bitmap.createScaledBitmap(temp, width, height, true);
 this.bitmap = temp;
 this.pointGame = new PointGame(0, 0);
 this.rect = new Rect();
 this.rnd = new Random();
}
// for objects that are colors
public RectGame(Context context, String color, int width, int height) {
 this.context = context;
 this.width = width;
 this.height = height;
 this.paint = new Paint();
 this.paint.setColor(Color.parseColor(color));
 this.pointGame = new PointGame(0, 0);
 this.rect = new Rect();
 this.rnd = new Random();
}
// sets the rectangle with the width, height around the point
public void setRect(Rect rectangle) {
 int l = this.pointGame.x-this.width/2;
 int t = this.pointGame.y-this.height/2;
 int r = this.pointGame.x+this.width/2;
 int b = this.pointGame.y+this.height/2;
 rectangle.set(l, t, r, b);
}

```

```

 public Rect getRectangle() {
 return this.rect;
 }
 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 PointGame getPointGame() {
 return pointGame;
 }
 // moves an object to a point out of the screen where it can't be seen
 public void kill() {
 this.pointGame.x = -500;
 this.pointGame.y = -500;
 setRect(this.rect);
 }
 // move the object by changing it's x, y values
 public void moveObst() {
 this.pointGame.x += this.pointGame.velX;
 this.pointGame.y += this.pointGame.velY;
 }
 // checks for collision between the player and the object
 public boolean collision (Player p) {
 return Rect.intersects(this.rect, p.getRectangle());
 }
 // draws the object on the game panel's canvas which the function gets as a parameter
 public abstract void draw(Canvas canvas);
 // update the object properties depending on it's purpose
 public abstract void update();
 // changes the velocity of the object depending on it's properties
 public abstract void changeVelocities();
 // changes speed of the object
 // type 0: freeze
 // type 1: slow
 // type 2: fast
 // type 3: back to previous

 public void changeSpeed(int type) {
 switch (type) {
 case 0: pointGame.speed = 0; break;
 case 1: pointGame.speed = 3; break;
 case 2: pointGame.speed = pointGame.initialSpeed * 2; break;
 case 3: pointGame.speed = pointGame.initialSpeed; break;
 }
 changeVelocities();
 }
}

```

- ```

public class WarningObj extends RectGame {
    private Bitmap warning;
    private int status = 0;
    private boolean soundPlayed = false;
    public WarningObj(Context context, int bonusID, int width, int height) {
        super(context, bonusID, width, height);
        warning = BitmapFactory.decodeResource(context.getResources(),
R.drawable.warning_warn);
        warning = Bitmap.createScaledBitmap(warning, width, height, true);
        this.pointGame.x = width/2 + rnd.nextInt(-1*width + Statics.width);
        this.pointGame.y = height/2 + rnd.nextInt(-1*height + Statics.height);
    }

    @Override
    public void draw(Canvas canvas) {
        // firstly, warn the player by putting warning_warn on a random point on the screen
        if (status == 0) {
            canvas.drawBitmap(warning, this.pointGame.x - (float) this.width / 2, this.pointGame.y
- (float) this.height / 2, null);
            // wait for 1.5 seconds
            final Handler handler = new Handler(Looper.getMainLooper());
            handler.postDelayed(() -> {
                status = 1;
            }, 1500);
        }
        // play explosion sound effect, draw the object on the canvas and change the image to
warning_active
        if (status == 1) {
            GameScreenActivity gsa = (GameScreenActivity) context;
            if (!soundPlayed) {
                gsa.getGamePanel().getSFX().playSound("explosion");
                soundPlayed = true;
            }
            setRect(this.rect);
            canvas.drawBitmap(bitmap, this.pointGame.x - (float) this.width / 2, this.pointGame.y -
(float) this.height / 2, null);
            final Handler handler = new Handler(Looper.getMainLooper());
            // wait for 3 seconds and then the object disappears
            handler.postDelayed(() -> {
                status = 2;
                kill();
            }, 3000);
        }
    }

    @Override
    public void update() {

    }

    @Override
    public void changeVelocities() {

```

```

    }
}

```

- `public class GamePanel extends SurfaceView implements SurfaceHolder.Callback {`

```

    private MainThread thread;
    private final Context context;
    private final Random rnd;
    private final SFX sfx;
    private final Bitmap bitmapBackground;
    private int counter = 0;

    private final Player player;
    private final BonusObj bonusObj;
    private final PowerUp powerUp;
    private boolean bonusTextDraw = false;
    private String bonusString = "";
    private int bonusID;
    private final Paint scorePaint, bonusPaint;
    private int extraScore;
    private boolean isNotNormalSpeed;
    private final GameSensors gameSensors;

    private ObstacleThread obstacleThread;
    private ScoreThread scoreThread;

    private final ObstacleArrayManager obstacleArrayManager;
    private int score = 0;
    private String time;
    private final int levelID;
    private String levelType;

    private boolean gameOver = false;

    public GamePanel(Context context) {
        super(context);
        getHolder().addCallback(this);
        this.context = context;

        // used to get the width and height of the phone
        DisplayMetrics metrics = this.getResources().getDisplayMetrics();
        width = metrics.widthPixels;
        height = metrics.heightPixels;
        System.out.println("w" + width);
        System.out.println("h" + height);
        rnd = new Random();
        sfx = new SFX(context);

        levelID = ((GameScreenActivity) context).getLevelID();
        Log.i(TAG, "GamePanel: level id " + levelID);
        if (levelID == 0) {
            levelType = "demo";
        } else if (levelID % 5 != 0) {
            levelType = "normal";
        }
    }
}

```



```

    } else {
        levelType = "boss";
        switch (levelID) {
            case 5: time = "1:16"; break;
            case 10: time = "2:14"; break;
            case 15: time = "1:44"; break;
            case 20: time = "2:17"; break;
            case 25: time = "1:25"; break;
        }
    }
    scorePaint = new Paint();
    scorePaint.setColor(Color.WHITE);
    scorePaint.setTextSize(100);
    if (levelID == 0) {
        scorePaint.setTextSize(75);
    }
    scorePaint.setTextAlign(Paint.Align.CENTER);
    scorePaint.setTypeface(ResourcesCompat.getFont(context, R.font.franklin));

    bonusPaint = new Paint();
    bonusPaint.setTextSize(100);
    bonusPaint.setTextAlign(Paint.Align.CENTER);
    bonusPaint.setTypeface(ResourcesCompat.getFont(context,
R.font.berlin_sans_fb_demi_bold));
    switch (user.difficulty) {
        case 0: extraScore = 75; break;
        case 1: extraScore = 100; break;
        case 2: extraScore = 125; break;
    }

    if (levelID == 0) {
        levelType = "demo";
    } else if (levelID % 5 != 0) {
        levelType = "normal";
    } else {
        levelType = "boss";
    }
    player = new Player(context, ((GameScreenActivity) context).getBitmapID(1), 100, 100);
    bitmapBackground =
BitmapFactory.decodeResource(context.getResources(), ((GameScreenActivity)
context).getBitmapID(2));
    bonusObj = new BonusObj(context, ((GameScreenActivity)
context).getBitmapID(3), 225, 225);
    powerUp = new PowerUp(context, this, ((GameScreenActivity)
context).findViewById(R.id.ibPowerUp));

    obstacleArrayManager = new ObstacleArrayManager(context);

    gameSensors = new GameSensors(context);

    setFocusable(true);
}
public SFX getSFX() {
    return sfx;
}
public Player getPlayer() {

```

```

        return player;
    }
    public PowerUp getPowerUp() {
        return powerUp;
    }
    public GameSensors getGameSensors() {
        return gameSensors;
    }
    public boolean isNotNormalSpeed() {
        return isNotNormalSpeed;
    }
    public void setNotNormalSpeed(boolean notNormalSpeed) {
        isNotNormalSpeed = notNormalSpeed;
    }
    public ObstacleArrayManager getObstacleArrayManager() {
        return obstacleArrayManager;
    }
    @Override
    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
    }
    @Override
    public void surfaceCreated(SurfaceHolder holder){
        // starts the thread when the game panel is created
        thread = new MainThread(getHolder(), this);
        thread.setRunning(true);
        thread.start();

        obstacleThread = new ObstacleThread(this);
        obstacleThread.setRunning(true);
        obstacleThread.start();

        scoreThread = new ScoreThread(this);
        scoreThread.setRunning(true);
        scoreThread.start();
    }
    @Override
    public void surfaceDestroyed(SurfaceHolder holder) {
    }
    @SuppressWarnings("ClickableViewAccessibility")
    @Override
    public boolean onTouchEvent(MotionEvent e) {
        // moves the player to where the user is touching
        float x = e.getX();
        float y = e.getY();

        int w = player.getWidth(); int h = player.getHeight();
        // keeps the player in the bounds of the screen
        if (x < (float)w/2) {x = (float)w/2;}
        if (x > width - (float)w/2) {x = width - (float)w/2;}
        if (y < (float)h/2) {y = (float)h/2;}
        if (y > height - (float)h/2) {y = height - (float)h/2;}
        PointGame playerPoint= player.getPointGame();
        float previousX = playerPoint.x;
        float previousY = playerPoint.y;
        if (e.getAction() != MotionEvent.ACTION_UP)
        {

```

```

        float dX = x - previousX;
        float dY = y - previousY;

        float distance = (float)Math.sqrt(Math.pow(x - playerPoint.x, 2) + Math.pow(y -
playerPoint.y, 2));

        float directionX = dX / distance;
        float directionY = dY / distance;

        if (distance > 5) {
            playerPoint.setVelX(directionX * playerPoint.getSpeed());
            playerPoint.setVelY(directionY * playerPoint.getSpeed());
        }
        playerPoint.set((int)previousX + (int)playerPoint.getVelX(), (int)previousY +
(int)playerPoint.getVelY());
    }
    return true;
}
@Override
public void draw(Canvas canvas) {
    super.draw(canvas);

    // draws the canvas background
    backgroundDrawer(canvas);
    player.draw(canvas);
    // bonus objects can only appear on normal levels
    if (levelType.equals("normal"))
        bonusObj.draw(canvas);
    // draws all objects
    obstacleArrayManager.drawAll(canvas);
    switch (levelType) {
        // tells the player to login if they are playing the demo
        case "demo":
            canvas.drawText("Login for more content!", (float) width / 2, (float) height / 2,
scorePaint);
            break;
        // tells the player their score if they playing a normal level
        case "normal":
            canvas.drawText("SCORE: " + score, (float) width / 2, 150, scorePaint);
            break;
        case "boss":
            // tells the player the time left when they are playing a boss level
            canvas.drawText("TIME LEFT:", (float) width / 2, 150, scorePaint);
            canvas.drawText(time, (float) width / 2, 300, scorePaint);
            break;
    }
    // stops the game and shows the dialog according to if the level is a demo or not
    if (gameOver) {
        finishGame();
        sfx.playSound("death");
        Log.i(TAG, "draw: about to stop the thread");
        if (levelID != 0)
            new Handler(Looper.getMainLooper()).post(() -> dialogMaker("over"));
        else
            new Handler(Looper.getMainLooper()).post(() -> dialogMaker("demo"));
    }
}

```

```

        // displays the text that tells the player which bonus they got
        // bonusTextDraw will only be true for 2 seconds after the player gets a bonus
        if (bonusTextDraw) {
            canvas.drawText(bonusString, player.getPointGame().x, player.getPointGame().y -
125, bonusPaint);
        }
    }

    public void update() {
        // updates the game object's positions when the game is running
        if (!gameOver) {
            player.update();
            bonusObj.update();
            obstacleArrayManager.updateAll();
            // if the games detects a collision with the player the game will stop and move the
player to its original position
            if (obstacleArrayManager.collisionAll(player)) {
                gameOver = true;
                Log.i(TAG, "update: collision -> game over!");
                player.getPointGame().set(width/2, height - 100);
                player.update();
            }
            // if the game detects a collision with the player it will kill the bonus object and give the
player a bonus
            if (bonusObj.collision(player)) {
                createBonus();
                bonusObj.kill();
                bonusObj.setStatus(0);
            }
        }
    }

    // the 2 functions create the scrolling effect of the canvas
    public void backgroundDrawer(Canvas canvas) {
        drawBitmap(canvas, bitmapBackground, 0, -counter, width, height);
        drawBitmap(canvas, bitmapBackground, 0, -counter+height, width, height);
        counter = counter % height;
        counter+=5;
    }

    public static synchronized void drawBitmap(Canvas canvas, Bitmap bitmap, int x, int y, int
width, int height) {
        Rect source = new Rect(0,0, bitmap.getWidth(), bitmap.getHeight());
        Rect target = new Rect(x,y,x+width, y+height);
        canvas.drawBitmap(bitmap,source,target,null);
    }

    // if the player has a power up that changes object speed, the bonus will change to extra
score
    public void checkBonusIDValidity() {
        if (isNotNormalSpeed)
            bonusID = 1;
    }

    // starts the bonus
    public void createBonus() {
        bonusID = rnd.nextInt(7);
        checkBonusIDValidity();
        bonusTextDraw = true;
        final Handler handler = new Handler(Looper.getMainLooper());
        handler.postDelayed(() -> bonusTextDraw = false, 2000);
    }

```

```

switch (bonusID) {
    // adds coins
    case 0:
        int bonusCoins = levelID + rnd.nextInt(10);
        bonusString = "+" + bonusCoins;
        bonusPaint.setColor(Color.rgb(255,215,0));
        if (isConnectedToInternet(context)) {
            user.coins += bonusCoins;
            DatabaseReference userRef =
                FirebaseDatabase.getInstance().getReference("Users").child(user.key);
            userRef.child("coins").setValue(user.coins);
        }
        else
        {
            handler.post(() -> Toast.makeText(context, "Coins were not actually added due
to connectivity issues with the internet!", Toast.LENGTH_SHORT).show());
        }
        sfx.playSound("bonus_pos");
        break;
    // adds score
    case 1:
        int bonusPoints = 500 + rnd.nextInt(501);
        bonusString = "+" + bonusPoints;
        bonusPaint.setColor(Color.GREEN);
        score += bonusPoints;
        sfx.playSound("bonus_pos");
        break;
    // lowers score
    case 2:
        int minusPoints = -500 - rnd.nextInt(501);
        bonusString = String.valueOf(minusPoints);
        bonusPaint.setColor(Color.RED);
        score += minusPoints;
        sfx.playSound("bonus_neg");
        break;
    // temp freeze objects
    case 3:
        bonusString = "FREEZE ALL OBSTACLES";
        isNotNormalSpeed = true;
        obstacleArrayManager.changeSpeedAll(0);
        handler.postDelayed(() -> {obstacleArrayManager.changeSpeedAll(3);
isNotNormalSpeed = false;}, 3000);
        bonusPaint.setColor(Color.GREEN);
        sfx.playSound("bonus_pos");
        break;
    // objects speed up
    case 4:
        bonusString = "OBSTACLES ACCELERATED";
        isNotNormalSpeed = true;
        obstacleArrayManager.changeSpeedAll(2);
        handler.postDelayed(() -> {obstacleArrayManager.changeSpeedAll(3);
isNotNormalSpeed = false;}, 3000);
        bonusPaint.setColor(Color.RED);
        sfx.playSound("bonus_neg");
        break;
    // decreases player size

```

```

        case 5:
            bonusString = "SIZE DECREASE!";
            bonusPaint.setColor(Color.GREEN);
            sfx.playSound("bonus_pos");
            player.setWidth(50); player.setHeight(50);
            player.createBitmap();
            handler.postDelayed(() -> {
                player.setWidth(100);
                player.setHeight(100);
                player.createBitmap();
            }, 5000);
            break;
        // increases player size
        case 6:
            bonusString = "SIZE INCREASE!";
            bonusPaint.setColor(Color.RED);
            sfx.playSound("bonus_neg");
            player.setWidth(200); player.setHeight(200);
            player.createBitmap();
            handler.postDelayed(() -> {
                player.setWidth(100);
                player.setHeight(100);
                player.createBitmap();
            }, 5000);
            break;
    }
}
// creates and shows the dialogs
public void dialogMaker(String dialogType) {
    switch (dialogType) {
        case "over" :
            GameOverDialog gameOverDialog = new GameOverDialog(context, score, time,
            levelID); gameOverDialog.show(); break;
        case "demo" :
            DemoGameOverDialog demoGameOverDialog = new
            DemoGameOverDialog(context); demoGameOverDialog.show(); break;
        case "complete" :
            BossLevelCompleteDialog bossLevelCompleteDialog = new
            BossLevelCompleteDialog(context, levelID); bossLevelCompleteDialog.show(); break;
    }
}
// updates the score
public void scoreUpdater() {
    if (levelType.equals("normal")) {
        this.score += extraScore;
    }
    else if (levelType.equals("boss")) {
        time = reduceTimeByOneSecond(time);
        if (time.equals("0:00")) {
            finishGame();
            new Handler(Looper.getMainLooper()).post(() -> dialogMaker("complete"));
        }
    }
}
// reduces the time by a second for boss levels
public String reduceTimeByOneSecond(String time) {

```

```

String[] timeParts = time.split(":");
int minutes = Integer.parseInt(timeParts[0]);
int seconds = Integer.parseInt(timeParts[1]);
if (seconds > 0) {
    seconds -= 1;
} else {
    seconds = 59;
    minutes -= 1;
}
String secondsString = String.valueOf(seconds);
if (seconds < 10) {
    secondsString = "0" + secondsString;
}

return minutes + ":" + secondsString;
}
// stops everything that is running when the game needs to be finished
public void finishGame() {
    thread.setRunning(false);
    scoreThread.setRunning(false);
    obstacleThread.setRunning(false);
    gameSensors.stopSensors();
    ((GameScreenActivity) context).stopMusic();
}
}

```

- public class GameSensors implements SensorEventListener {


```

private final SensorManager sensorManager;
private float xRotation;

public GameSensors(Context context) {
    sensorManager = (SensorManager)
context.getSystemService(Context.SENSOR_SERVICE);
    sensorManager.registerListener(this,
sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
SensorManager.SENSOR_DELAY_NORMAL);
    xRotation = 0;
}
// gives the x rotation to the position sensor object
public float getXRotation() {
    return xRotation;
}
// stops the sensor
public void stopSensors() {
    sensorManager.unregisterListener(this);
}
@Override
public void onSensorChanged(SensorEvent sensorEvent) {
    Sensor sensor = sensorEvent.sensor;
    // get the phone's x rotation and save it in a variable
    if (sensor.getType()==Sensor.TYPE_ACCELEROMETER) {
        xRotation = sensorEvent.values[0];
    }
}

```

```

    }
    @Override
    public void onAccuracyChanged(Sensor sensor, int i) {

    }
}

```

- ```

public class MainThread extends Thread {
 public static final int MAX_FPS = 30;
 public double averageFPS;
 private final SurfaceHolder surfaceHolder;
 private final GamePanel gamePanel;
 private volatile boolean running;
 public static Canvas canvas;

 public MainThread(SurfaceHolder surfaceHolder, GamePanel gamePanel) {
 super();
 this.surfaceHolder = surfaceHolder;
 this.gamePanel = gamePanel;
 }

 public void setRunning(boolean running) {
 this.running = running;
 }

 // when the thread is running, boolean "running" is true
 @Override
 public void run() {
 long startTime;
 long timeMillis;
 long waitTime;
 int frameCount = 0;
 long totalTime = 0;
 long targetTime = 1000/MAX_FPS;
 while (running) {
 startTime = System.nanoTime();
 canvas = null;

 try {
 synchronized (surfaceHolder) {
 // locks the canvas and lets the game panel draw on it
 canvas = this.surfaceHolder.lockCanvas();
 this.gamePanel.update();
 this.gamePanel.draw(canvas);
 }
 } catch (Exception e) {
 e.printStackTrace();
 } finally {
 if (canvas != null) {
 try {
 // unlocks the canvas and updates the display
 surfaceHolder.unlockCanvasAndPost(canvas);
 } catch (Exception e) {
 e.printStackTrace();
 }
 }
 }
 }
 }
}

```



```

 }
 }
}
timeMillis = (System.nanoTime() - startTime) / 1000000;
waitTime = targetTime - timeMillis;

try {
 if (waitTime > 0)
 sleep(waitTime);
} catch (Exception e) {
 e.printStackTrace();
}
// calculates the average frames per second
// displays it on the screen if the player want it to from the options menu
totalTime += System.nanoTime() - startTime;
frameCount++;
if (frameCount == MAX_FPS) {
 averageFPS = 1000 / ((totalTime / frameCount) / 1000000);
 frameCount = 0;
 totalTime = 0;
// System.out.println(averageFPS);
 ((GameScreenActivity)gamePanel.getContext()).setFPSView(averageFPS);
}
}

Log.i(TAG, "run: finish running, out of run loop");
}
}

```

```

public class ObstacleArray {
 private final Context context;
 private final RectGame[] obstArr;
 private int id;

 public ObstacleArray(Context context, Class<?> classType, int numOfObst) {
 this.context = context;
 setIdFromClass(classType);
 this.obstArr = new RectGame[numOfObst];
 // fills the array with different instance of the object it got as a parameter
 for (int i = 0; i < numOfObst; i++) {
 RectGame currObst = makeNewInstance();
 this.obstArr[i] = currObst;
 }
 }

 // checks if any of the objects in the array touched the player, returns yes if any of them did
 public boolean collision(Player player) {
 for (RectGame obj : obstArr)
 if (obj.collision(player))
 return true;
 return false;
 }

 // draws all objects in the array on the canvas
 public void draw(Canvas canvas) {
 for (RectGame obj: obstArr) {
 obj.draw(canvas);
 }
 }
}

```

```

 }
}
// updates all objects in the array
public void update() {
 for (RectGame obj : obstArr) {
 obj.update();
 }
}
// changes speed to all objects in the array based on the type
public void changeSpeed(int type) {
 for (RectGame obj : obstArr) {
 obj.changeSpeed(type);
 }
}
// gives an id number depending on the type of the object
private void setIdFromClass(Class<?> classType) {
 if (classType == AlphaObj.class)
 this.id = 1;
 else if (classType == MovingRectObj.class)
 this.id = 2;
 else if (classType == GapObj.class)
 this.id = 3;
 else if (classType == WarningObj.class)
 this.id = 4;
 else if (classType == DownUpObj.class)
 this.id = 5;
 else if (classType == MovingGapObj.class)
 this.id = 6;
 else if (classType == CrossingObj.class)
 this.id = 7;
 else if (classType == CrossingSingleObj.class)
 this.id = 8;
 else if (classType == BounceObj.class)
 this.id = 9;
 else if (classType == PositionSensorObj.class)
 this.id = 10;
}
// makes a new instance of the object based on the id
public RectGame makeNewInstance() {
 RectGame currObj;
 switch (this.id) {
 case 1: currObj = new AlphaObj(context, "#800000", 100, 100); break;
 case 2: currObj = new MovingRectObj(context, "#00FF00", 100, 100); break;
 case 3: currObj = new GapObj(context, "#326496", 250, 50); break;
 case 4: currObj = new WarningObj(context, R.drawable.warning_active, 450, 450); break;
 case 5: currObj = new DownUpObj(context, "#03A9F4", 125, 125); break;
 case 6: currObj = new MovingGapObj(context, "#FF0000", 250, 50); break;
 case 7: currObj = new CrossingObj(context, "#70B596", 125, 125); break;
 case 8: currObj = new CrossingSingleObj(context, "#CB9813", 75, 75); break;
 case 9: currObj = new BounceObj(context, "#FF9632", 50, 50); break;
 case 10: currObj = new PositionSensorObj(context, "#FF33FF", 400, 150); break;
 default: currObj = null; break;
 }
 return currObj;
}
}

```

- ```
public class ObstacleArrayManager {

    private final Context context;
    private int randomClass;
    private final ArrayList<ObstacleArray> obstList;
    private final int levelID;

    public ObstacleArrayManager(Context context) {
        this.context = context;
        this.obstList = new ArrayList<>();
        this.levelID = ((GameScreenActivity) context).getLevelID();
    }
    // add an array of a randomly chosen object to the list
    public void addArrToList() {
        randomClass = getClassFromLevel(getArrayFromLevelID());
        ObstacleArray obstacleArray = setArray();
        obstList.add(obstacleArray);
    }
    // returns a random class from the array of objects that can appear in the level
    public int getClassFromLevel(int[] arr) {
        int rnd = new Random().nextInt(arr.length);
        return arr[rnd];
    }
    // makes the array of objects based on the random class it generated
    public ObstacleArray setArray() {
        ObstacleArray obstArr;
        switch (this.randomClass) {
            case 1: obstArr = new ObstacleArray(context, AlphaObj.class, setRndNum(2)); break;
            case 2: obstArr = new ObstacleArray(context, MovingRectObj.class, setRndNum(2));
break;
            case 3: obstArr = new ObstacleArray(context, GapObj.class, 1); break;
            case 4: obstArr = new ObstacleArray(context, WarningObj.class, 1); break;
            case 5: obstArr = new ObstacleArray(context, DownUpObj.class, setRndNum(1));
break;
            case 6: obstArr = new ObstacleArray(context, MovingGapObj.class,1); break;
            case 7: obstArr = new ObstacleArray(context, CrossingObj.class, 1); break;
            case 8: obstArr = new ObstacleArray(context,
CrossingSingleObj.class,setRndNum(2)); break;
            case 9: obstArr = new ObstacleArray(context, BounceObj.class, setRndNum(2));
break;
            case 10: obstArr = new ObstacleArray(context, PositionSensorObj.class,
setRndNum(1)); break;
            default: obstArr = new ObstacleArray(context, MovingRectObj.class,1); break;
        }
        return obstArr;
    }
    // sets the sleep time after the object spawns depending on the object that spawned
    public int getSleepFromType() {
        int sleepTime;
        switch (this.randomClass) {
            case 4: sleepTime = changeSleepFromDifficulty(1000); break;
            case 6:
            case 3: sleepTime = changeSleepFromDifficulty(1600); break;
            case 5:
```

```

        case 1: sleepTime = changeSleepFromDifficulty(2000); break;
        case 8: case 10:
        case 2: sleepTime = changeSleepFromDifficulty(2500); break;
        case 7 : sleepTime = changeSleepFromDifficulty(3000); break;
        case 9: sleepTime = changeSleepFromDifficulty(3500); break;
        default: sleepTime = changeSleepFromDifficulty(500000); break;
    }
    return sleepTime;
}
// changes the sleep time between spawning objects depending on the player's chosen
difficulty
public int changeSleepFromDifficulty(int originalSleep) {
    if (user.difficulty == 0)
        return (int)(originalSleep * 1.75);
    if (user.difficulty == 1)
        return originalSleep;
    if (user.difficulty == 2)
        return (int)(originalSleep*0.75);
    return originalSleep;
}
// checks if any of the objects in the list touched the player, returns yes if any of them did
public boolean collisionAll (Player player) {
    for (ObstacleArray obstArr : obstList)
        if (obstArr.collision(player))
            return true;
    return false;
}
// draws all objects in the list on the canvas
public void drawAll(Canvas canvas) {
    for (ObstacleArray obstacleArray : obstList) {
        obstacleArray.draw(canvas);
    }
}
// updates all objects in the list
public void updateAll() {
    for (ObstacleArray obstacleArray: obstList) {
        obstacleArray.update();
    }
}
// changes speed to all objects in the list based on the type
public void changeSpeedAll(int type) {
    for (ObstacleArray obstacleArray: obstList) {
        obstacleArray.changeSpeed(type);
    }
}

// sets the array of objects that can appear in the level
public int[] getArrayFromLevelID() {
    int[] arr;
    switch (levelID) {
        // level 0 - play without signing in
        case 0: arr = new int[]{2}; break;
        case 1: arr = new int[]{1,2,3}; break;
        case 2: arr = new int[]{1,2,4}; break;
        case 3: arr = new int[]{1,3,4}; break;
        case 4: arr = new int[]{2,3,4}; break;
    }
}

```

```

        case 5: arr = new int[]{1,2,3,4}; break;
        case 6: arr = new int[]{2,3,4,5,6}; break;
        case 7: arr = new int[]{4,5,6}; break;
        case 8: arr = new int[]{1,3,5,6}; break;
        case 9: arr = new int[]{1,2,4,6}; break;
        case 10: arr = new int[]{1,2,3,4,5,6}; break;
        case 11: arr = new int[]{1,4,5,7}; break;
        case 12: arr = new int[]{2,3,4,5,8}; break;
        case 13: arr = new int[]{2,4,6,7}; break;
        case 14: arr = new int[]{1,3,5,6,8}; break;
        case 15: arr = new int[]{1,2,3,4,5,6,7,8}; break;
        case 16: arr = new int[]{3,4,6,8,9}; break;
        case 17: arr = new int[]{1,2,5,7,9}; break;
        case 18: arr = new int[]{1,3,5,6,8,9}; break;
        case 19: arr = new int[]{6,7,8,9}; break;
        case 20: arr = new int[]{1,2,3,4,5,6,7,8,9}; break;
        case 21: arr = new int[]{1,2,3,4,10}; break;
        case 22: arr = new int[]{3,5,6,9,10}; break;
        case 23: arr = new int[]{4,5,6,7,10}; break;
        case 24: arr = new int[]{5,6,7,8,9,10}; break;
        case 25: arr = new int[]{1,2,3,4,5,6,7,8,9,10}; break;
        default: arr = new int[]{1000}; break;
    }
    return arr;
}
// returns a random number in a range to be the length of the array
public int setRndNum(int type) {
    switch (type) {
        case 1: return 1 + new Random().nextInt(3);
        case 2: return 3 + new Random().nextInt(3);
        default: return 1;
    }
}
}
}

```

- ```

public class ObstacleThread extends Thread{
 private boolean running;
 private final GamePanel gamePanel;

 public ObstacleThread (GamePanel gamePanel) {
 this.gamePanel = gamePanel;
 }

 public void setRunning(boolean running) {
 this.running = running;
 }

 @Override
 public void run() {
 while (running) {
 try {
 // add an array of a randomly chosen object to the list
 gamePanel.getObstacleArrayManager().addArrToList();
 // sleep after adding the array
 }
 }
 }
}

```

```

 int sleepTime = gamePanel.getObstacleArrayManager().getSleepFromType();
 Thread.sleep(sleepTime);
 } catch (InterruptedException e) {
 e.printStackTrace();
 }
}
}
}
}

```

- public class PowerUp implements View.OnClickListener {

```

 private final Context context;
 private final GamePanel gamePanel;
 private final ImageButton ibPowerUp;
 private final int imageDisplayID;
 private final TextView tvPowerUpCoolDown;
 private int time, timeActive, timeCoolDown;
 // retries makes each runnable only work when the game is running
 private int retries = 0;

 public PowerUp(Context context, GamePanel gamePanel, ImageButton ibPowerUp) {
 this.context = context;
 this.gamePanel = gamePanel;
 this.ibPowerUp = ibPowerUp;
 ibPowerUp.setOnClickListener(this);
 imageDisplayID = user.selectedPowerUp;
 if (imageDisplayID == 0) {
 ibPowerUp.setVisibility(View.GONE);
 }
 else {
 @SuppressWarnings("UseCompatLoadingForDrawables") Drawable drawable =
 context.getResources().getDrawable(((GameScreenActivity)
 context).getPowerUpImages()[user.selectedPowerUp], null);
 Bitmap imageBitmap = ((BitmapDrawable) drawable).getBitmap();
 Bitmap resizedBitmap = Bitmap.createScaledBitmap(imageBitmap, 100, 100, false);
 ibPowerUp.setImageBitmap(resizedBitmap);
 }
 tvPowerUpCoolDown = ((GameScreenActivity)
 context).findViewById(R.id.tvCountDown);
 setTimesFromType();
 }
 // sets the amount of time that the power up is active depending on the chosen power up
 public void setTimesFromType() {
 switch (imageDisplayID) {
 case 1: timeActive = 4; timeCoolDown = 20; break;
 case 2: timeActive = 3; timeCoolDown = 17; break;
 case 3: timeActive = 5; timeCoolDown = 15; break;
 }
 }
 // activates or deactivates the power up
 public void powerActivated(boolean activateOrDeactivate) {
 // param: true is to activate, false to deactivate
 switch (imageDisplayID) {
 // player higher speed

```

```

case 1:
 if (activateOrDeactivate) {
 gamePanel.getPlayer().getPointGame().setSpeed(75);
 gamePanel.setNotNormalSpeed(true);
 }
 else {
 gamePanel.getPlayer().getPointGame().setSpeed(17);
 gamePanel.setNotNormalSpeed(false);
 }
 break;
// player smaller
case 2:
 if (activateOrDeactivate) {
 gamePanel.getPlayer().setWidth(50);
 gamePanel.getPlayer().setHeight(50);
 } else {
 gamePanel.getPlayer().setWidth(100);
 gamePanel.getPlayer().setHeight(100);
 }
 gamePanel.getPlayer().createBitmap();
 break;
// obstacles move slower
case 3:
 if (activateOrDeactivate) {
 gamePanel.getObstacleArrayManager().changeSpeedAll(1);
 gamePanel.setNotNormalSpeed(true);
 }
 else {
 gamePanel.getObstacleArrayManager().changeSpeedAll(3);
 gamePanel.setNotNormalSpeed(false);
 } break;

}
}
// activate power up and timer when clicked
@Override
public void onClick(View view) {
 // checks for if retries need to be updated
 retries = ((GameScreenActivity) context).getRetryCount();
 // if the speed already changed from the bonus object, the player cannot activate the
 freeze power up
 if (gamePanel.isNotNormalSpeed() && imageDisplayID == 3) {
 Toast.makeText(context, "Cannot activate this power up when objects are frozen",
 Toast.LENGTH_SHORT).show();
 return;
 }
 powerActivated(true);
 ibPowerUp.setClickable(false);
 ibPowerUp.setVisibility(View.INVISIBLE);
 time = timeActive + timeCoolDown;
 startTime();
 Log.i("TAG", "onClick: activated");
}
// changes the color of the text depending on the type of timer
public void setPowerUpTextColor(boolean activateOrDeactivate) {
 if (activateOrDeactivate)

```

```

 tvPowerUpCoolDown.setTextColor(Color.rgb(11,102,35));
 else
 tvPowerUpCoolDown.setTextColor(Color.WHITE);
 }
 // starts the timer
 @SuppressWarnings("SetTextI18n")
 public void startTime() {
 if (time == 0) {
 tvPowerUpCoolDown.setText("");
 this.ibPowerUp.setClickable(true);
 this.ibPowerUp.setVisibility(View.VISIBLE);
 }
 else {
 // displays time as a text, if statement is to check if power up is active or on cool down
 if (time > timeCoolDown) {
 setPowerUpTextColor(true);
 tvPowerUpCoolDown.setText(Integer.toString(time - timeCoolDown));
 }
 else {
 if (time == timeCoolDown)
 powerActivated(false);
 setPowerUpTextColor(false);
 tvPowerUpCoolDown.setText(Integer.toString(time));
 }
 // wait a second and reduce time by 1 second, call for the function again
 new Handler(Looper.getMainLooper()).postDelayed(() -> {
 if (retries == ((GameScreenActivity) context).getRetryCount()) {
 if (time != 0) {
 time--;
 startTime();
 }
 }
 }, 1000);
 }
 }
 // when the game restarts, the function restarts the timer that show the power ups's cool
 down
 public void restartPowerUpTimer() {
 tvPowerUpCoolDown.setText("");
 if (user.selectedPowerUp != 0) {
 ibPowerUp.setClickable(true);
 ibPowerUp.setVisibility(View.VISIBLE);
 }
 time = 0;
 }
}

```

- public class ScoreThread extends Thread{
 private boolean running = false;
 private final GamePanel gamePanel;

 public ScoreThread (GamePanel gamePanel) {
 this.gamePanel = gamePanel;
 }



```

public void setRunning(boolean running) {
 this.running = running;
}

// updates the score every second
@Override
public void run() {
 while (running) {
 try {
 Thread.sleep(1000);
 gamePanel.scoreUpdater();
 } catch (InterruptedException e) {
 e.printStackTrace();
 }
 }
}
}

```

- ```

public class SFX {
    private final SoundPool soundPool;
    private final int death, explosion, bonus_pos, bonus_neg;

    @SuppressWarnings("ObsoleteSdkInt")
    public SFX(Context context) {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
            AudioAttributes aa = new AudioAttributes.Builder()
                .setContentType(AudioAttributes.CONTENT_TYPE_MUSIC)
                .setUsage(AudioAttributes.USAGE_GAME)
                .build();
            soundPool = new SoundPool.Builder()
                .setMaxStreams(10)
                .setAudioAttributes(aa)
                .build();
        }
        else {
            soundPool = new SoundPool(10, AudioManager.STREAM_MUSIC, 1);
        }
        death = soundPool.load(context, R.raw.death, 1);
        explosion = soundPool.load(context, R.raw.explosion, 1);
        bonus_pos = soundPool.load(context, R.raw.bonus_pos, 1);
        bonus_neg = soundPool.load(context, R.raw.bonus_neg, 1);
    }

    public void playSound(String sound) {
        if (user.optSoundEffects) {
            switch (sound) {
                case "death": soundPool.play(death, 1, 1, 0, 0, 1); break;
                case "explosion": soundPool.play(explosion, 1, 1, 0, 0, 1); break;
                case "bonus_pos": soundPool.play(bonus_pos, 1, 1, 0, 0, 1); break;
                case "bonus_neg": soundPool.play(bonus_neg, 1, 1, 0, 0, 1); break;
            }
        }
    }
}

```

```
    }
}
```

- ```

public class ShopItem {
 private Bitmap bitmap;
 private final String title;
 private final int price;
 private boolean owned;

 public ShopItem(Context context, int bitmapID, String title, int price, boolean owned) {
 turnIDToBitmap(context, bitmapID);
 this.title = title;
 this.price = price;
 this.owned = owned;
 }

 public Bitmap getBitmap() {
 return bitmap;
 }
 public String getTitle() {
 return title;
 }
 public int getPrice() {
 return price;
 }
 public boolean isOwned() {
 return owned;
 }
 public void setOwned(boolean owned) {
 this.owned = owned;
 }

 public void turnIDToBitmap(Context context, int id) {
 this.bitmap = BitmapFactory.decodeResource(context.getResources(), id);
 }
}

```
- ```

public class ShopItemAdapter extends ArrayAdapter<ShopItem> {
    private final Context context;
    private final List<ShopItem> objects;
    private final String tag;
    private TextView tvPrice;
    public ShopItemAdapter(Context context, int resource, int textViewResourceId,
List<ShopItem> objects, String tag) {
        super(context, resource, textViewResourceId, objects);
        this.context = context;
        this.objects = objects;
        this.tag = tag;
    }
    @SuppressWarnings("NonConstantResourceId")
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        LayoutInflater inflater = ((Activity)context).getLayoutInflater();

```

```

    @SuppressWarnings("ViewHolder") View view =
    inflater.inflate(R.layout.shop_layout, parent, false);
    ImageView ivProduct = view.findViewById(R.id.ivProduct);
    if (tag.equals("tagBackground")) {
        ViewGroup.LayoutParams params = ivProduct.getLayoutParams();
        params.width = 150;
        params.height = 250;
        ivProduct.setLayoutParams(params);
    }
    TextView tvTitle = view.findViewById(R.id.tvTitle);
    tvPrice = view.findViewById(R.id.tvPrice);
    ImageView ivSelected = view.findViewById(R.id.ivSelected);
    ShopItem temp = objects.get(position);

    ivProduct.setImageBitmap(temp.getBitmap());
    tvTitle.setText(String.valueOf(temp.getTitle()));
    tvPrice.setText(String.valueOf(temp.getPrice()));
    // changes price color to green if owned, add a tick to selected item
    switch (tag) {
        case "tagIcon":
            if (user.iconOwned.get(position))
                tvPrice.setTextColor(Color.GREEN);
            if (user.selectedIcon == position)
                ivSelected.setVisibility(View.VISIBLE);
            break;
        case "tagBackground":
            if (user.backgroundOwned.get(position))
                tvPrice.setTextColor(Color.GREEN);
            if (user.selectedBackground == position)
                ivSelected.setVisibility(View.VISIBLE);
            break;
        case "tagBonus":
            if (user.bonusOwned.get(position))
                tvPrice.setTextColor(Color.GREEN);
            if (user.selectedBonus == position)
                ivSelected.setVisibility(View.VISIBLE);
            break;
        case "tagPowerUp":
            if (user.powerUpOwned.get(position))
                tvPrice.setTextColor(Color.GREEN);
            if (user.selectedPowerUp == position)
                ivSelected.setVisibility(View.VISIBLE);
            break;
    }
    return view;
}
// updates the price text color
public void updateTextColor(int position) {
    switch (tag) {
        case "tagIcon":
            if (user.iconOwned.get(position)) {
                tvPrice.setTextColor(Color.GREEN);
            } break;
        case "tagBackground":

```

```

        if (user.backgroundOwned.get(position)) {
            tvPrice.setTextColor(Color.GREEN);
        } break;
    case "tagBonus":
        if (user.bonusOwned.get(position)) {
            tvPrice.setTextColor(Color.GREEN);
        } break;
    case "tagPowerUp":
        if (user.powerUpOwned.get(position)) {
            tvPrice.setTextColor(Color.GREEN);
        } break;
    }
}
}

```

- ```

public class Statics {
 public static int width;
 public static int height;
 public static User user;
 // checks if the phone is connected to the internet
 public static boolean isConnectedToInternet(Context context) {
 ConnectivityManager cm =
 (ConnectivityManager)
context.getSystemService(Context.CONNECTIVITY_SERVICE);
 NetworkCapabilities nc = cm.getNetworkCapabilities(cm.getActiveNetwork());
 return nc != null &&
nc.hasCapability(NetworkCapabilities.NET_CAPABILITY_INTERNET);
 }
}

```
- ```

public class User {
    public String key;
    public String uid;
    public String userName, mail;
    public int deaths;
    public int coins;
    public int difficulty; // 0 - easy, 1 - normal, 2 - hard
    // options
    public boolean optMusic, optSoundEffects, optFPS;
    // saves high score for every normal level
    public List<Integer> hsl;
    // saves best time for boss levels
    public List<String> bestTime;
    // saves if boss level was already beaten
    public ArrayList<Boolean> levelComplete;
    // boolean for if item is owned
    public ArrayList<Boolean> iconOwned, backgroundOwned, bonusOwned, powerUpOwned;
    // saves the selected item from the shop
    public int selectedIcon, selectedBackground, selectedBonus, selectedPowerUp;
    public int itemsOwned;
}

```

```

public User() {
    InitializeAllValues();
}

public User(String uid, String mail, String userName, String key) {
    this.key = key;
    this.uid = uid;
    this.userName = userName;
    this.mail = mail;
    InitializeAllValues();
}

public void InitializeAllValues() {
    deaths = 0;
    coins = 9999;
    itemsOwned = 0;
    difficulty = 1;
    optMusic = true; optSoundEffects = true; optFPS = false;

    hsl = new ArrayList<>();
    for (int i = 0; i <= 25; i++) {
        hsl.add(0);
    }
    bestTime = new ArrayList<>();
    bestTime.add(""); // index 0 is unused
    bestTime.add("1:16");bestTime.add("2:14");
    bestTime.add("1:44");bestTime.add("2:17");bestTime.add("1:25");
    levelComplete = new ArrayList<>();
    for (int i = 0; i <=5; i++) {
        levelComplete.add(false);
    }
    levelComplete.set(0, true); // index 0 is unused

    iconOwned = new ArrayList<>();
    for (int i = 0; i < 10; i++) {
        iconOwned.add(false);
    } iconOwned.set(0, true);

    backgroundOwned = new ArrayList<>();
    for (int i = 0; i < 7; i++) {
        backgroundOwned.add(false);
    } backgroundOwned.set(0, true);

    bonusOwned = new ArrayList<>();
    for (int i = 0; i < 8; i++) {
        bonusOwned.add(false);
    } bonusOwned.set(0, true);

    powerUpOwned = new ArrayList<>();
    for (int i = 0; i < 4; i++) {
        powerUpOwned.add(false);
    } powerUpOwned.set(0, true);

    selectedIcon = 0; selectedBackground = 0; selectedBonus = 0; selectedPowerUp = 0;

```

```
}  
}
```

- ```
public class VibrationService extends Service {

 private Vibrator vibrator;

 public VibrationService() {
 }

 @Override
 public IBinder onBind(Intent intent) {
 // TODO: Return the communication channel to the service.
 throw new UnsupportedOperationException("Not yet implemented");
 }

 @Override
 public int onStartCommand(Intent intent, int flags, int startId) {
 vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
 VibrationEffect effect = VibrationEffect.createOneShot(200,
VibrationEffect.DEFAULT_AMPLITUDE);
 vibrator.vibrate(effect);
 return START_STICKY;
 }

 @Override
 public void onDestroy() {
 super.onDestroy();
 vibrator.cancel();
 }
}
```