**תכנות מונחה עצמים ופיתוח משחקים**

**סמסטר ב', תשפ"ג**

***פרויקט גמר – Angry Birds***

**חברי הצוות:**

**- מיכאל זרגרי 208289082.**

**- עדי יעקובי .208773168**

**- רון אליאן .208834945**

**רקע :**

במשחק זה, השחקן שולט על להקה של [ציפורים](https://he.wikipedia.org/wiki/%D7%A2%D7%95%D7%A4%D7%95%D7%AA) צבעוניות שמנסות להחזיר את ה[ביצים](https://he.wikipedia.org/wiki/%D7%91%D7%99%D7%A6%D7%94) שלהן, אשר נגנבו על ידי קבוצה של [חזירים](https://he.wikipedia.org/wiki/%D7%97%D7%96%D7%99%D7%A8_%D7%94%D7%91%D7%99%D7%AA) ירוקים רעבים. בכל שלב של המשחק, החזירים מוגנים על ידי מבנים מחומרים שונים (כגון [עץ](https://he.wikipedia.org/wiki/%D7%A2%D7%A5_(%D7%97%D7%95%D7%9E%D7%A8_%D7%92%D7%9C%D7%9D)), [זכוכית](https://he.wikipedia.org/wiki/%D7%96%D7%9B%D7%95%D7%9B%D7%99%D7%AA) או [סלעים](https://he.wikipedia.org/wiki/%D7%A1%D7%9C%D7%A2)). מטרת המשחק היא לחסל את כל החזירים.

באמצעות [קלע](https://he.wikipedia.org/wiki/%D7%A7%D7%9C%D7%A2), השחקן משגר קבוצה מוגבלת של ציפורים במטרה לפגוע בחזירים ישירות או לפגוע במבנים הסובבים להם, לגרום לאבנים להתמוטט ולמחוץ את החזירים.

ישנם מספר סוגים של ציפורים במשחק, מכובד על ידי צבע וצורה.

כל רמה מתחילה עם כמות, סוגים, וסדר של הציפורים שנקבעו מראש. אם כל החזירים ימחצו לאחר הציפור האחרונה ששוגרה, השלב הושלם והשלב הבא נפתח.

**אופן המימוש:**

המשחק מומש בצורה הבאה:

* נוצרו 6 שלבים בעלי רמות קושי שונות.
* קיימים 4 סוגים של ציפורים בעלי יכולות שונות.
* מימוש הגרפיקה על ידי ספריית SFML .
* מימוש הפיזיקה על ידי ספריית BOX2D .

**מחלקות:**

**State :**

מחלקה אבסטרקטית המהווה מחלקת בסיס לכל מחלקות הstate- במשחק.

**MainMenuState :**

תפריט המשחק,תפקידו של התפריט הוא להיות שער הכניסה למשחק ולתת למשתמש לבחור אם הוא רוצה להתחיל את המשחק,

בלובי מימשנו ארבעה כפתורים:

הראשון start תפקידו הוא להכניס את המשתמש למסך בוחר השלבים.

השני הוא help תפקידו לתת למשתמש עזרה בלהבין את חוקי המשחק ואת מטרתו.

והשלישי הוא בעצם יציאה מהמשחק.

ורבעי הוא השתקת השמע שמתנגן ברקע

**: PlayState**

מצב המשחק,אחראית על שטף המשחק, קריאה ועדכון המשחק בכל תזוזה של הלוח

**: LevelSelectState**

לובי לבחירת שלבים, מציגה לשחקן לאיזה שלבים יש לו גישה ולאלו לא, מציגה את הניקוד הגבוהה ביותר של כל שלב.

**: TransitionScreensStaten**

מצב של מסכי מעבר, אחראי להציג לשחקן בעת סיום שלב האם הוא עבר או לא על ידי קבלת ערך המסך שצריך להציג.

**StateMAchine :**

מכונת מצבים, אחראי על ניהול כל המצבים במשחק, מחזיק את המצבים הקיימים באותו הרגע במשחק על ידי מחסנית(מצב שאין בו שימוש לא נמצא שם), על ידי פונקציות המשנות את ערכי הבוליאנים שהמחלקה מחזיקה המחלקה בכל איטרציה של המשחק בודקת האם יש עדכון של אחד המצבים.

אפשרויות פעולה של המכונה:

1. מחיקה – הסרה של המצב הנוכחי מהמחסנית(המצב בראש המחסנית).
2. הוספה עם החלפה – הוספה של מצב חדש למשחק אשר יחליף את המצב הנוכחי, אז המכונה מוחקת את המצב הנוכחי ומוסיפה את החדש.
3. הוספה ללא החלפה – הוספה של מצב חדש בלי למחוק את השלב הקיים כרגע, המכונה דוחפת את המצב החדש למחסנית ובכך המצב החדש הופך להיות המצב הנוכחי כאשר הישן נמצא מתחתיו בלי לאבד את המידע הקיים עליו כרגע.
4. החלפה בין מצבים – לעיתים קיימים 2 מצבים במחסנית ונדרשת החלפה בינהם(מצב משחק ובוחר השלבים לצורך הדוגמא שצריכים להתממשק אחד עם השני ולשמור נתונים שהועברו בינהם).

**Game :**

המחלקה הראשית של המשחק, המריצה את המשחק.

**GameResources :**

מחלקה זו אחראית על ניהול כל המקורות למשחק, טקסטורות פונטים מוזיקה.

**Window :**

מחלקה המייצרת את חלון המשחק אחראית על מתן גישה לחלון ועידכונו במהלך השלב.

**Bird** :

מחלקת "ציפור" מייצגת ציפור במשחק. זה הופך את הציפור ל"חפץ" שניתן לשגר על ידי השחקן. מחלקה זו היא מחלקה אבסרטרקתית וממנה יורשים ארבעה סוגי ציפורים : ציפור אדומה, ציפור צהובה, ציפור כחולה וציפור שחורה. לכל ציפור יש יכולת שונה, או במקרה של האדומה שהיא הציפור הפשוטה ביותר והיא ללא יכולת בכלל. **RedBird** :

מחלקת "ציפור אדומה" מייצגת את הציפור האדומה במשחק. הציפור האדומה יורשת מהציפור האדומה ומקבלת את היכולות הבסיסיות בלבד של הציפור. מחלקה זו ללא יכולת מיוחדת לעומת השאר כיוון שהיא הציפור הפשוטה ביותר.

**BlackBird** :

מחלקת "הציפור השחורה" - מייצגת את הציפור השחורה במשחק. הציפור הזו יורשת מ-BIRD את היכולות הבסיסיות שלה, ובנוסף יכולה "להתפוצץ" על ידי לחיצה על העכבר במהלך תעופה.

**YelowBird**:

מחלקת "הציפור הצהובה" - מייצגת את הציפור הצהובה במשחק. הציפור הזו יורשת מ-BIRD את היכולות הבסיסיות

ובנוסף בעלת יכולת לקבל boost במהלך תעופה על ידי לחיצה על העכבר.

**BlueBird**:

מחלקת "הציפור הכחולה" - מייצגת את הציפור הכחולה במשחק. הציפור הזו יורשת מ-BIRD את היכולות הבסיסיות גם כן ובנוסף בעלת יכולת "להתפצל" לשלושה ציפורים קטנות.

**World**:

מחלקה זו מייצגת את העולם - היא מגדירה את התחומים, הגרביטציה ומחזיקה את העולם שבו מתנהלים כל הגופים.

**StaticObject**:

מחלקה זו מייצגת את כל האובייקטים הסטטיים והיא מחלקה אבסרטקית. במילים אחרות אובייקטים שללא תנועה משל עצמם אלא רק אם נפגעו או לא ופועלים בהתאם לפי גרביטציה.

**ObjectFactory** :

מחלקת מפעל לאובייקטים ממומשת כמחלקה טמפלייטית ונפרשת על ידי שתי הצורות הבאות:

1. Birds .
2. StaticObject

**Ground**:

מחלקה זו מייצגת את האדמה ויורשת מ- STATICOBJECTS - המחלקה הזו מגדירה גוף של אדמה במשחק, ואובייקט זה מוגדר כסטטי ולא ניתן להזיזו על ידי פגיעה.

**Rogatka**:

מחלקה זו מייצגת את הרוגטקה ויורשת מ- STATICOBJECTS - באובייקט זה מונחים הציפור כאשר מגיע תורן לעוף וניתן לכוון אותן לכיוון הרצוי.

**Obstacle** :

מחלקה שנועדה לייצוג כל סוגי האובייקטים במשחק שניתן להרוס ושהשחקן יכול לבוא איתם במגע ולקבל תגובה מהם.

המחלקה מנהלת את עצמה וכל אובייקט גרפי עוקב אחרי האובייקט הפיזיקלי, והאובייקט הפיזיקלי מנוהל ע"י התזוזות במשחק עצמו וע"י פגיעות מהשחקן שמנוהלות ע"י השחקן.

**Circular Obstacle :**

מחלקה שנועדה לייצוג כל סוגי האובייקטים העגולים במשחק, ההבדלים בינה לבין Obstacle הוא גם במימוש הפונקציות הגרפיות והפונקציות הפיזיקליות הרי שהאובייקטים עגולים ולכן רצינו התנהגות שונה מאשר האובייקטים שאינם עגולים במשחק.

**GuideLine :**

מחלקה שנועדה לייצג את הכוונת של השחקן כאשר הוא מותח את הציפור, בעת מתיחה הכוונת תתנהג ותדמה את זריקת הציפור ולאיפה היא תעוף אילו השחקן יעזוב את העכבר וישלח את הציפור לדרכה, המחלקה משתמשת באובייקטים פיזיקליים כדי לדמות את זריקות הציפור בעזרת אובייקט פיזיקלי של Box2d שנקרא sensor ולכן איננו משתתף ב collisions במשחק ולכן איננו מפריע ופוגע באובייקטים.

**LevelManager :**

מחלקה שנועדה לקרוא את השלבים מקובץ הטקסט שמייצג את השלבים וליצור את האובייקטים המתאימים ולהכניסם לוקטורים שיתנו לו ע"י התוכנית. התוכנית ממקמת את האובייקטים ע"פ כללים שנקבעו על ידינו בקובץ הטקסט שלנו ועבור סימנים מוסכמים מראש.

המיקומים נקבעים בהתאמה ובאופן יחסי לגודל המסך שלנו.

**MyContactListner:**

מחלקה זו מייצגת את ההתנגשויות במשחק ומנהלת אותם במשחק. מחלקה זו יורשת מb2listen.

b2Listener שמאזין לאירועים של תכנית הפיזיקה. ירושה ממחלקה זו נותנת את היכולת לציין ממשק מותאם אישית שישמש כמאזן ויפעל ברגעים מסוימים במהלך התנגשות.

מחלקה זו יורשת מb2listen על מנת שנוכל לממש את ההתנגשויות כרצוננו. את ההתנגשויות מימשנו על ידי השיטה multimethods.

**הערות אחרות**:

* בכל פתיחת שלב, השלב נמצא במצב "הצגה". המשתמש רואה איך השלב נראה – איך המבנה נראה, מרחק מהרוגטקה, מיקום חזירים וכו' וע''י כך לתכנן את אופן הזריקה ומימוש נכון של הציפורים שהוא מקבל בכל שלב. ניתן לדלג על "הצגת השלב" ע''י לחיצה על כל מקש כלשהו.
* קיימים 4 סוגים של ציפורים, ול- 3 מהם קיימת יכולת ייחודית המופעלת על ידי הקלקה על המסך בזמן שליחת הציפור על הבניין.

ציפור אדומה – זוהי הציפור הפשוטה אין יכולת מיוחדת.

ציפור כחולה – מתפצלת ל3 ציפורים קטנות.

ציפור צהובה – מקבלת בוסט של כח במהלך הזריקה.

ציפור שחורה – מתפוצצת, ההדף שנוצר מספיק חזק כדי להוריד בניין.

* כל השלבים נעולים עד שהם נפתחים על ידי מעבר השלב הקודם, ניתן לגשת לכל שלב פתוח במהלך המשחק ולשפר את הניקוד שם.

**תבניות עיצוב:**

**State :**

המשחק מומש על ידי תבנית העיצוב State , כל מצב של המשחק הוא מחלקה בפני עצמה.

המשחק רץ באותה הצורה כאשר על ידי מכונת מצבים אנו משנים את המצב שבו המשחק נמצא כרגע. לדוגמא: אמצע שלב, מסך בחירת שלב, תפריט המשחק, מסכי מעבר וכו'.

מחלקות קשורות :

State,MainMenuState,PlayState,LevelSelectState,TransitionScreensState,StateMAchine

**Factory :**

תבנית עיצוב לייצור אובייקטים (בית חרושת לאובייקטים) ,בפרוייקט המפעל הוא טמפלייתי ונפרש על פי שתי משפחות:

1)ציפורים. 2)אובייקטים סטטיים.

מחלקת המפעל במשחק : ObjectFactory .

**Singelton :**

מחלקה שמצד אחד צריך גישה לאובייקט מהסוג שלה מכל מקום בקוד ומצד שני, מעצם הגדרתה אמור להיות רק אובייקט אחד כזה בתוכנית.

מחלקות שמומשו כסינגלטון – GameResources,GuideLine,ObjectFactory .

**MultyMethods :**

היא גישה לתכנות מונחה עצמים שבה ניתן לקשור פונקציות ולקרוא לפונקציה המתאימה לפי טיפוסי המשתנים.

כאשר מתרחש מגע בין שני גופים במנוע הפיזיקה, הפונקציה "handleCollision" נקראת ומקבלת את שני העצמים המעורבים ומטפלת בהם לפי המימוש שנתנו.

מימוש הMultyMethods שלנו בוצע כך:

החזקנו Map שהמפתח שלה הוא זוג מצביעים של אובייקטים. כל זוג אובייקטים מקבל פונקצייה ייחודית משלו שבמהלך התנגשות הוא קורא לפונקציית ההתנגשות המתאימה שלו.

חיפוש הפונקציה מתבצע על ידי פונק חיפוש, במידה ולא נמצאה פונק התנגשות, האיטרטור יחזיר NULL מהמבנה נתונים ולא יקרה דבר. נציין ששילבנו את הגישה עם BOX2D וכאשר מתרחש מגע, מחקלת MyContactListner תזמן את הפונק BeginContact תאתר את המצביעים שהתנגשו, ותקרא לHandleCollision.