**תכנות מונחה עצמים ופיתוח משחקים**

**פרויקט סוף סמסטר**

**Pengi Adventures**

**פרטי הסטודנטים:**

* רנאל בן סימן טוב
* עידן כהן

**הסבר כללי:**

בפרויקט זה מימשנו את המשחק Pengi Adventures – משחק שבו הפינגווין פינגי מטייל בעולם ההרפתקאות שלו, תוך שהוא מתחמק משלל האויבים והמכשולים העומדים בדרכו ומפריעים לו, ומטרתו היא לאסוף כמה שיותר מטבעות ולהשלים את השלבים בהצלחה.

שמנו דגש על עקרונות תכנות מונחה עצמים – ירושה ופולימורפיזם, תבניות, שימוש במבני נתונים של STL, וכן מימשנו תבניות עיצוב מתקדמות כגון Singleton, Factory, Command ו-Multimethods. כמו כן העשרנו את הגרפיקה של המשחק והשתמשנו בספרייה הגרפית SFML תוך שילוב אנימציה למגוון הדמויות.

**תיכון:**

כפי שניתן לראות מדיאגרמת ה-UML המצורפת, עץ הירושה שלנו מורכב מהמחלקות הבאות:

* מחלקת **GameObject** המייצגת אובייקט כללי במשחק. מכילה בעיקר תכונות הקשורות לציור הספרייט של האובייקט. ממנה יורשות המחלקות StaticObject ו-MovingObject.
* מחלקת **StaticObject** - מחלקת אב עבור האובייקטים הסטטיים במשחק (אריח, מעלית, מקפצה, תותח, קוצים, קופסת הפתעה וכד').
* מחלקת **MovingObject** - מחלקת אב עבור האובייקטים הנעים במשחק – השחקן ושלל האויבים. מכילה תכונות ופונקציות הקשורות לתזוזה.
* מחלקת ביניים נוספת **Enemy** שמאגדת תחתיה את האויבים.

ממחלקות האב הללו נגזרות כלל המחלקות הקונקרטיות שמייצגות את האובייקטים במשחק.

בנוסף לעץ הירושה, קיימות מחלקות נוספות שאינן יורשות או מורישות והן משמשות לניהול המשחק והלוח ולענייני גרפיקה הקשורים בתפריט ובשורת המידע:

* מחלקת **Controller** – המחלקה העיקרית והראשית האחראית על ניהול המשחק. היא מריצה את לולאת המשחק הראשית ולולאת השלב. מחזיקה בתוכה את הדמויות הנעות במשחק – השחקן והאויבים – ומאזינה לאירועים של קלט המקשים מהמשתמש. מחלקה זו אחראית לניהול כל ההתרחשות הנובעת מהתזוזה, ומסתייעת במחלקת CollisionHandling (המממשת את תבנית העיצוב Multimethods) לניהול ההתנגשויות בין האובייקטים במשחק. עובדת יד ביד עם מחלקת Board ומתקשרת איתה באופן שוטף.
* מחלקת **Board** - המחלקה השנייה בחשיבותה במשחק. אחראית על לוח המשחק עצמו, כלומר כל האובייקטים הסטטיים במשחק. כמו כן, חלק עיקרי מתפקודה הוא קריאת קובץ הקלט של השלב. נעזרת במחלקה התבניתית Factory (שנפרשת עבור MovingObject ועבור StaticObject) לייצור האובייקטים בהתאם לתווים הנקראים מהקובץ.

מחלקות תפריט ושורת מידע:

* מחלקת **Menu** – ממומשת לפי תבנית העיצוב Command. אחראית על ייצוג התפריט המוצג לשחקן כאשר הוא פותח את המשחק, וכן מכילה את מסך הוראות המשחק.
* מחלקת **InfoBar** - אחראית על ההיבטים הגרפיים של שורת המידע המופיעה בראש כל שלב ומכילה את מספר השלב, מספר החיים שנותרו והניקוד שנצבר.

כמו כן, קיימת מחלקת Singleton מיוחדת **ResourcesManager** המוקדשת לניהול המשאבים הכבדים במשחק (טקסטורות, פונטים, מוזיקה וצלילים) שמעצם היותה Singleton שומרת את כל המשאבים הללו פעם אחת בלבד לאורך כל התכנית וכך חוסכת בזיכרון.

מחלקות נוספות:

* מחלקת **Animation** לניהול כלל ההיבטים הקשורים לאנימציה במשחק.
* מחלקת **CollisionHandling** לניהול ההתנגשויות בין האובייקטים במשחק. מממשת את תבנית העיצוב Multimethods.
* מחלקת **Factory** – שימוש בתבנית העיצוב Factory לייצור אובייקטים במשחק.

כדי שיהיה ניתן להוסיף שלבים חדשים בקלות, שלבי המשחק נשמרים כקובצי טקסט. עבור כל שלב ישנו קובץ טקסט נפרד הנקרא LevelN.txt כאשר N הוא מספר השלב.

כל קובץ מכיל בשורה הראשונה שני מספרים – גובה ורוחב השלב, ולאחר מכן התווים המתארים את השלב. בקובץ macros.h מפורטים התווים המייצגים כל אובייקט במשחק.

**רשימת הקבצים:**

* Controller.h+Controller.cpp - מחלקת הניהול העיקרית של המשחק.
* Board.h+Board.cpp - מחלקה המנהלת את לוח המשחק (את האובייקטים הסטטיים במשחק).
* Menu.h+Menu.cpp - מחלקה לניהול התפריט.
* Command.h+Command.cpp – מחלקת אב אבסטרקטית לייצוג פקודה בתפריט (לפי תבנית העיצוב Command).
* Exit.h+Exit.cpp – לייצוג פקודת היציאה בתפריט.
* Instructions.h+Instructions.cpp – לייצוג מסך ההוראות של המשחק.
* Play.h+Play.cpp – לייצוג פקודת התחלת משחק חדש בתפריט.
* InfoBar.h+InfoBar.cpp – מחלקה לייצוג שורת המידע המופיעה בראש החלון של כל שלב.
* ResourcesManager.h+ResourcesManager.cpp – מחלקת סינגלטון לניהול המשאבים הכבדים במשחק (טקסטורות, פונטים, מוזיקה וצלילים).
* main.cpp - המחלקה הראשית, ממנה מריצים את התכנית.
* macros.h - קובץ עבור קבועים בהם עושים שימוש במחלקות שונות.
* GameObject.h+GameObject.cpp - מחלקת אב לייצוג אובייקט במשחק.
* MovingObject.h+MovingObject.cpp – מחלקת אב לייצוג אובייקט נע במשחק.
* Player.h+Player.cpp – מחלקה לייצוג השחקן (הפינגווין) במשחק.
* Enemy.h+Enemy.cpp – מחלקת אב לייצוג אויב במשחק.
* FlyingMonster.h+FlyingMonster.cpp – מחלקה לייצוג האויב המעופף במשחק.
* Jellyfish.h+Jellyfish.cpp – מחלקה לייצוג האויב הימי (המדוזה) במשחק.
* Orange.h+Orange.cpp – מחלקה לייצוג התפוז האויב.
* RollingEnemy.h+RollingEnemy.cpp – מחלקה לייצוג הלימון האויב.
* StaticObject.h+StaticObject.cpp - מחלקת אב לייצוג אובייקט סטטי במשחק.
* Bone.h+Bone.cpp – מחלקה לייצוג העצם שהפינגווין יורה על האויבים.
* Bubble.h+Bubble.cpp – מחלקה לייצוג בועת המגן שהפינגווין מקבל וכך הופך להיות מוגן מאויבים.
* Bullet.h+Bullet.cpp – מחלקה לייצוג הפגז הנורה מהתותח.
* Cannon.h+Cannon.cpp – מחלקה לייצוג התותח.
* Carnivorous.h+Carnivorous.cpp – מחלקה לייצוג צמח טורף שיוצא מהאדמה.
* Coin.h+Coin.cpp – מחלקה לייצוג מטבע.
* Door.h+Door.cpp – מחלקה לייצוג הדלת בסוף כל שלב.
* Escalator.h+Escalator.cpp – מחלקה לייצוג המדרגה הנעה.
* Spike.h+Spike.cpp – מחלקה לייצוג הקוצים.
* SpringBoard.h+SpringBoard.cpp – מחלקה לייצוג מקפצה.
* SurpriseBox.h+SurpriseBox.cpp – מחלקה לייצוג קופסת הפתעה ממנה יוצא מטבע.
* Tile.h+Tile.cpp – מחלקה לייצוג אריח.
* Water.h+Water.cpp – מחלקה לייצוג אריח של ים.
* Animation.h+Animation.cpp – מחלקה לניהול האנימציה.
* AnimationData.h – לשמירת נתוני האנימציה (מפה הממפה כיוון לווקטור של הספרייט שיט הרלוונטי).
* Direction.h+Direction.cpp – מחלקה לייצוג כיוון האובייקט הנע.
* Factory.h+Factory.cpp – מחלקה המממשת את תבנית העיצוב Factory לייצור אובייקטים במשחק.
* CollisionHandling.h+CollisionHandling.cpp – מחלקה המממשת את תבנית העיצוב Multimethods ומכילה את פונקציות ההתנגשויות בין האובייקטים השונים.

**מבני נתונים עיקריים ותפקידיהם:**

* במחלקת Board, וקטור השומר את כל האובייקטים הסטטיים בלוח המשחק.
* במחלקת Controller, וקטור לשמירת האויבים.
* במחלקת Animation, AnimationData – מפה הממפה בין כל כיוון לווקטור של ה-IntRects הרלוונטיים לכיוון זה בספרייט שיט.
* במחלקת Factory, מפה הממפה בין התו המייצג את האובייקט לפונקציה המפעילה את הבנאי שלו ומחזירה אותו.
* ב- CollisionHandling.cpp מפה הממפה בין זוגות (pairs) של סוגי אובייקטים, לפונקציית ההתנגשות המתאימה.
* במחלקת ResourcesManager, unordered\_maps הממפות בין קידודים של משאבים למשאבים עצמם.

**אלגוריתמים הראויים לציון:**

**באגים ידועים:**

**הערות אחרות:**