פרויקט סוף סמסטר- oop2

Legend4Life

מגישים:

נדב אטיס
אריאל אמון
נועם מרג'ני
אפרת אילוז

תיאור המשחק:

משחק זה הוא מז'אנר משחקי הרפתקאות, המטרה העיקרית היא לאסוף כמה שיותר מטבעות תוך סיום המשחק בזמן הקצר ביותר והריגת כמה שיותר אויבים.
המשחק כולל בחובו מספר שלבים, אשר על מנת לעבור אותם על השחקן להתמודד עם מכשולים רבים: אבנים ו'פצצות' שנופלות מהשמיים, מלכודות עכבר, מלכודות פרחים, אויבים תוקפים, לבה ומים, קופסאות בעלות משקל שמהוות 'חומה', אבנים מתפוררות וכן מכשולי גובה ומרחק.
על השחקן לחשוב בחוכמה איך להשתמש ביכולותיו המגוונות והרבות- הוא יכול להכות אויבים ולקפוץ בין מכשולים אך הכבירה מבניהם היא להכות במכה מיוחדת, שהורגת את האויב במהירות יותר גבוהה מהמכה הרגילה, ניתן להחליף את המכה במכה אחרת על ידי איסוף קריסטל.
כשיתמזל מזלו של השחקן והוא יצלח את מעבר המכשולים הוא יגיע לשער אשר יפתח לכבודו ויעביר אותו לעבר העולם הבא.

כללי המשחק ופיצ'רים מרכזיים:

- כדי ללמוד כיצד לשחק במשחק, ניתן ללחוץ על כפתור ה-help בתפריט המשחק, יוצגו הסברים רבים ומומחשים ועל ידי החץ ניתן לעבור ביניהם.
- על ידי exit \ esc נצא מהמשחק.
- על ידי האייקון של ההגדרות נוכל לכבות את המוזיקה, או להנמיך את הסאונדים.
- על ידי לחיצה על כפתור ה- miniMap יוכל השחקן לראות את כל המפה בבת אחת.
- טרם תחילת המשחק, יועבר השחקן למסך בחירת שלב, כלומר את העולם בו הוא ישחק – אדמה, אופל או פנטזיה. לאחר מכן יעבור מיידית למסך בחירת שחקן – הוא יבחר אחת מארבע דמויות המוצעות לו.
- השחקן, בעל סט היכולות: ריצה, קפיצה, תקיפה ותקיפת בירייה מיוחדת יכנס לעולם עם הדמות שנבחרה ויתחיל לשחק בשלב.
- תנועות השחקן בעזרת מקשי המקלדת.
- לכל שחקן יש בר חיים דינמי, עת הוא נפגע יורד לו חיים והוא ממשיך לשחק מנקודה מסוימת שקרובה אליו. אם יגמרו לו כל החיים שהיו לו – השחקן מסיים את תפקידו במשחק. ניתן לאסוף לבבות לאורך הדרך על מנת להאריך את שהות השחקן במשחק.
- גם לכל אויב יש בר חיים דינמי על מנת לתת לשחקן חיווי – כך השחקן ידע כמה החיים אזלו לו וכמה מכות עליו לתת על מנת להרוג את האויב ולהעלימו בצורה מוחלטת מן המשחק.
- על השחקן לאסוף לאורך הדרך מטבעות - שיצטברו לו לאורך הדרך, לבבות – שיתנו לו הזדמנות לנצח במשחק ולא להיפסל, חצים – יתנו לשחקן מהירות תנועה גבוהה יותר מהנוכחית, קריסטלים – יחליפו לשחקן את המכה המיוחדת הנוכחית.
- בהגעה לשער יופיע מסך בחירה: לחזור לתפריט או לעבור לשלב הבא.
- כשהשחקן ינצח יופיע לו מסך עם כמות המטבעות שאסף ואת כמות הזמן בו הצליח לסיים.

- כשהשחקן יפסיד יופיע לו מסך הפסד.

תיכון (design):

מחלקת gameObject**:** מחלקת אב שמורישה למחלקות הבאות:

1. **מחלקת MovingObject:** יורשת ממחלקת **gameObject**, היא מנהלת את כל האובייקטים שזזים במשחק ומורישה למחלקות הבאות:

מחלקת **Enemy:** המחלקה מנהלת את האויב: 'מציירת' אותו, מעדכנת, מנהלת תזוזות, מנהלת תקיפה ומנהלת פסילות. בנוסף לכך היא מעדכנת את הפיזיקליות של האויב ומהירות ההליכה.

מחלקת **Trap**: מחלקת בסיס שמנהלת את המלכודות ומורישה למחלקות הבאות:

מחלקת **fallingBlock**, מחלקת **guillotine**. שתיהן מחלקות שהן מלכודות שמטרתן להפריע לשחקן.

מחלקת **Player**: מחזיקה בתוכה את שק היכולות של השחקן: תזוזה, קפיצה, מתקפה, מתקפה מיוחדת, מוות, חיים, אנרגיה. בנוסף מגדירים פה את הכובד של משתמשים בה הרבה בניהול ההתנגשויות כי עלינו לנתר התנגשות של שחקן כמעט עם כל עצם בעולם.

מחלקת **MovingPlatform**: מנהלת את האריחים הזזים במשחק. מחזיקה **member** שמזהה שינוי בכיוון התזוזה וכן מהירות ו**member** מסוג שעון.

2. **מחלקת staticObject:** מנהלת את העצמים הסטטיים במשחק. היא מחלקה שמורישה למחלקות: **winGate, box, platform**. מחלקת **platform** מורישה למחלקת **goneTile** כי היא מחלקה שמשתמשת בפלטפורמר עם התנהגות מיוחדת: נעלם לאחר דריכה. בנוסף **staticObject** מורישה ל**staticObjectAnimation** שאחראית על האובייקטים הסטטיים שיש להם אנימציה והיא מורישה למחלקת **bonus** שמורישה למחלקות הבונוס השונות: בונוס של מטבעות, מהירות, מג'יק אפקט, חיים. היא גם מורישה ל-**water trap**שהיא גם אובייקט סטטי שמנהל תזוזה.
3. **מחלקת ground**: ייצור גוף שמסמן את הרצפה שאיתה יתנגש השחקן בעת נפילה, זיהוי התנגשות ומוות.

מחלקת animation: מורישה למחלקת **static animation**.

4. **מחלקת magic effect** שיורשת גם מgameObject

מחלקת **state** שמורישה למחלקות **leadboardState** , **gameState** ו- **loadingState**, היא מורישה גם ל-**baseMenuState** שהיא מחלקת בסיס לניהול התפריטים. היא מורישה למחלקות **settuingState, VictoryAndDefeat, chooseCharacter, instructionState, MainMenuState**. כל אלו מחלקות שמימושם הם המסכים השונים במשחק.

מחלקת **button** היא מחלקת הכפתורים והיא מורישה למחלקות שמיישמות את הכפתורים השונים במשחק והם: **stateButtonCharacter, stateButton, volumeButton, soundButton, pageButton,chooseLevelButton, chooseCharacter, miniMapButton**

מחלקות כלליות, שלא כוללת ירושה:

מחלקת **audio**- מחלקת סינגלטון לניהול האודיו במשחק.

מחלקת **cameraView**: גלגול המפה והתמרכזותה על השחקן.

- מחלקת **checkpoint**: חזרה לנקודת הציון האחרונה שבה היה המשתמש בעת התנגשות.
- מחלקת **collisionHandler** ו – **collisionListener** הן מחלקות שמנהלות התנגשויות, ההאזנה קורת על ידי **box2d**.
- מחלקת **controller**: מחזיקה את לולאת המשחק הראשית. יש עטיפה בחריגות.
- מחלקת **dataManagment**: ניהול מבני הנתונים המרכזיים במשחק.
- מחלקת **levelLoader** – טעינת השלבים במשחק בצורה גנרית , משתמשת בקבצי הטקסט שעליהם מקודדים השלבים (**factory**).
- מחלקת **lifeStatusBar** – מחלקה שאחראית על ניהול בר החיים.
- מחלקת **miniMap**- אחראית על התנהלות מיני מפה שנמצאת בתוך המשחק.
- מחלקת **playerStatus**- אחראית על ניהול הסטטוס של המשתמש.
- מחלקת **resources** מחלקת סינגלטון שאחראית לניהול עזרי המשחק.
- מחלקת **stateController**.

רשימת הקבצים:

main.cpp

Resources.cpp – מחזיק את העזרים שנוספו למשחק כמו דמויות, סאונדים וכו.. מחלקת סינגלטון.

-State

BaseMenuState.cpp – ניהול שוטף של תפריט המשחק, ממנו יורשים ה-states של התפריט.

ChooseCharacterState.cpp – אחראי על בחירת השחקן.

GameState.cpp – ניהול שוטף של המשחק.

InsrtuctionsState.cpp – אחראי על לוח העזרה.

LoadingState.cpp – טעינת ה-state .

MainMenuState.cpp – מנהל את הכפתורים של התפריט.

State.cpp – מחלקת בסיס אבסטרקטית שמכילה את כל המימושים הדיפולטיביים.

StateController.cpp – ניהול המחסנית.

VictoryAndDefeat.cpp – אחראי על מסכי המעבר של ניצחון והפסד.

-buttons

Button.cpp – מחלקת בסיס אבסטרקטית, מנטרת את לחיצת הכפתורים.

ChooseCharacterButton.cpp – ניהול לחיצת השחקן על הדמות הרצויה למשחק.

MiniMapButton.cpp – ניהול לחיצת השחקן על כפתור ה- miniMap במפת המשחק.

-gameManage

Animation.cpp – ניהול האנימציות עבור האובייקטים הזזים והסטטיים.

Audio.cpp – ניהול האודיו במשחק. מחלקת סינגלטון.

CheckPoint.cpp – ניתור הצ'ק פוינט, כלומר לאן ישוב השחקן לאחר ירידה בחיים.

CollisionHandler.cpp – ניהול התנגשויות במשחק על ידי גישת collision filtering ומיסוך ביטים של box2d.

CollisionListener.cpp – האזנה להתנגשויות.

Controller.cpp – ניהול הלולאה הראשית של המשחק.

DataManagment.cpp – ניהול מבני הנתונים המרכזיים במשחק.

Direction.cpp – ניהול וניתור הכיוונים במשחק וממשוק עם מקשי המקלדת.

GameObject.cpp – ניהול האובייקטים במשחק.

LevelLoader.cpp – ניהול טעינת השלבים ומעבר בין שלבים.

-gameUi

CameraView.cpp – מימוש המפה המתגלגלת תוך התמרכזות בדמות המשחקת.

LifeStatusBar.cpp – מימוש בר חיים.

MiniMap.cpp – מימוש הפיצ'ר minimap.

PlayerStatus.cpp – מימוש הסטטוס של השחקן, כולל את השם, החיים, האנרגיה, המטבעות והטיימר.

-MovingObject

Enemy.cpp – ניהול ההתנהלות של האויב, כולל את התזוזה שלו ואת התקיפה החכמה.

FallingBlock.cpp – ניהול האבנים הנופלות מן השמיים.

GuillotioneTrap.cpp – ניהול הגליוטינה.

MagicEffect.cpp – ניהול המכה החכמה של השחקן.

MovingObject.cpp – מחלקת אב אבסטרקטית לכל האובייקטים הזזים.

MovingPlatform.cpp – ניהול האריח הזז, משמש כמעין 'נדנדה'.

Player.cpp – ניהול השחקן, כולל את כל שק היכולות שיש לשחקן.

Trap.cpp – מחלקת בסיס של המלכודות.

-StaticObject

Bonus.cpp – מחלקת אב לכל המתנות במשחק.

BonusCoin.cpp – ניהול אובייקט המטבעות.

BonusLive.cpp – ניהול אובייקט הלבבות.

BonusMagicEffect.cpp – ניהול אובייקט הקריסטלים שנאספים כדי לשנות את המכה המיוחדת של השחקן.

Box.cpp – ניהול הקופסאות בעלות המשקל.

GoneTile.cpp – ניהול האריחים שמתפוררים בעת דריכה עליהם.

Ground.cpp – ניהול הקרקע של המשחק.

Minetrap.cpp – ניהול האובייקט שמשמש מלכודת – כשדורכים עליו הוא נסגר על השחקן.

Platform.cpp – ניהול האריחים שמהווים בסיס למשחק – עליהם ירוץ השחקן.

StaticObject.cpp – מחלקת אב לכל האובייקטים הסטטיים.

StaticAnimation.cpp – מחלקת אב לכל האובייקטים הסטטיים שיש להם גם אנימציה.

StaticObjectAnimation.cpp – מחלקת אב לכל האובייקטים הסטטיים שיש להם גם אנימציה.

WaterTrap.cpp – ניהול אובייקט המים – נפילה למים תוביל למוות.

WinGate.cpp – ניהול שער הניצחון בעת סיום שלב.

Macros.h – מכיל את כל ה- consts שהשתמשנו בהם.

HashHandler.h – שני struct-ים תבניתיים שונים, למען שימוש טבלת הגיבוב ומולטי methods.

Factory.h – מחלקה תבניתית, מנהלת את בניית האובייקטים, משתמשים בזה כשבונים שלב. מחלקת סינגלטון.

לכל קובץ cpp קיים גם קובץ h בסה"כ יש 52 + 55 = 107 קבצים בסה"כ.

<u>Desing patterns **ומבני נתונים:**</u>

- **<u>Factory</u>** – תבנית עיצוב לבניית אובייקטי המשחק. כל האובייקטים שיורשים מ-gameObject במשחק מכילים member פרטי בוליאני סטטי אשר עת אנו קוראים שלב חדש, עבור כל אובייקט מצויים כל המידע שאנו צריכים לשלוח לבנאי. המידע נשלח לאחר פירוק השורה לנתונים למחלקת factory אשר מריצה חיפוש ב- **map**, שמורכבת מ**functor string** . אם האות המייצגת מצויה כבר במפה אזי המצביע לפונקציה מחזיר uniqePtr של הטיפוס המתאים, לפנינו 3 טיפוסים: staticObject**,** movingObject **ו-**player.למען הגנריות בנינו את factory כ- **תבניתית**.
- **<u>Command</u>** – תבנית עיצוב לניהול תפריטים, מחלקת בסיס היא baseMenuState היא מכילה וקטור של uniqePtr של מחלקת בסיס button במחלקה זו יש ניהול כללי של כל הפונקציונליות שאמורה להיות בתפריט, בין אם ניתור לחיצה או עכבר שנמצא מעל כפתור, הדפסות ועדכונים.
- **<u>State</u>** – ניהול של כל המסכים הקיימים במשחק, הכוונה במסך זה תצוגה של 'מצב' שיש לו ניהול event-ים, ברינדור שונה ובכל מיני אלמנטים שמרכיבים מסך. מנוהל על ידי **מחסנית** שנמצאת ב-stateController, state היא מחלקה אבסטרקטית ממנה יורשות כל המחלקות. לולאת המשחק בפונקציית run מפעילה את הפונקציות render ,handleEvent ,update על ה-state האחרון שנכנס למחסנית.

- **Box2d –** הגדרת עולם פיזיקאלי המחשב את הדמיית חוקי הפיזיקה של המשחק. חיישן העולם הפיזיקאלי מזהה ומתריע על התנגשויות בעולם הפיזיקאלי. כל אובייקט הוחזק גם עם המידע של box2d וגם עם המידע של sfml כדי ליצור מוחשיות.
- **MultiMethod**- תבנית עיצוב שמנהלת את כלל ההתנגשויות במשחק, ההתנגשויות מזוהות על ידי collisionListener של box2d ומשם מועברים הטיפוסים של שני האובייקטים שזוהו כמתנגשים, במחלקת collisionHandler מתבצע חיפוש של שני טיפוסים האובייקטים הללו, במידה ונמצא אזי נרצה לנתר את ההתנגשות הזאת וניגש לפונקציה המתאימה. בחרנו לממש על ידי מבנה נתונים **unorderd map** אשר הוא **טבלת גיבוב**. העיקרון שלו הוא שחיפוש הפונקציה מתבצע ב-**o**(1) , המפתח של טבלת הגיבוב מורכב משני ערכים (טיפוס האובייקטים) לכן במחלקת **HashHandler** מימשנו פונקציית גיבוב מתאימה אשר מגבבת תחילה את הטיפוס הראשון, אחר כך את השני ועל שני הערכים שהפונקציות מחזירות אנו מבצעים **xor**. בנוסף מימשנו אופרטור סוגריים עגולים () למען השוואה של המפתחות.
- **Vector uniqe ptr** של static object ושל moving object
- **פולימורפיזם** – שימוש למען ניהול המשחק.
- **Map-** על מנת לשמור את טבלת השיאים השתמשנו במבנה נתונים ממויין אשר החיפוש בו הוא ב-**log(N)**.
- **מחלקות singeltons** – שימוש בשיטה השלישית כלומר, אובייקט סטטי המוחזק בפונקציה סטטית (get instance).

****

אלגוריתם לציין:

- **אויבים**- מנגנון ייחודי לכל אויב ואויב על מנת לייצר התנהגות טבעית.
- **ניהול התנגשויות –** על ידי multiMethods **ו**collision listener **(מ-**box2d**)** על ידי **מיסוך ביטים.**
- קריאה וייצור אובייקטים על ידי שיטת **factory.**
- ניהול מצבים על ידי שילוב שיטת **command** ושיטת **stateManager.**
- **Box2d –** בעזרתו מימשנו את האבנים הזזות (נדנדות), העצמים שנופלים מהשמיים וכן המלכודות. על מנת לממש את ההתנהגות של ה**אויבים** השתמשנו בשעון ובתנאים מיוחדים שקשורים לאנימציה של כל אובייקט. על מנת לממש את המתקפה של האויב עם השחקן ושל השחקן עם האויב הגדרנו לפי כיוונים: אם אתה מזהה את אויבך בין אם מקדמתו ובין אם בגבו: תתקוף.
- **מפה מתגלגלת** – מתגלגלת למעלה, למטה ולצדדים וכן מתמרכזת על השחקן. טיפול במקרי קיצון ובעיקר שימוש במעקב אחרי השחקן על ידי setCenter.

****

באגים ידועים: לא קיימים.

****

הערות:

1. שימוש נרחב ב-**Box2d** כלומר, לכל האובייקטים נבנה גוף, ההתנגשויות נותרו באמצעות **box2d** .
2. בתחילת המשחק מופיע מסך **help** כדי לאפשר לשחקן לזהות את השחקנים והאויבים ולהבדיל ביניהם במשחק, כולל הסברים.
3. שימוש רב באנימציות, **sprite** וספריית **SFML**.
4. ניתן להשתיק את המוזיקה, להנמיך ולהגביר סאונדים באמצעות הכפתורים המתאימים.

5. בעת תכנות המשחק השתמשנו בחומרים שנלמדו במהלך השנה כולה: **תבניות**, **חריגות**, **איטרטורים**, **מצביעים חכמים**, **סינגלטון**, **תבניות עיצוביות מיוחדות** ודאגה לשמירה על **עקרון האינקפסולציה**.
6. לכל דבר במשחק קיים חיווי בצורת כל ברי המידע והחיים שקיימים שהוזכרו לעיל.
7. יש שלושה שלבים וניתן להוסיף בצורה גנרית עוד שלבים למשחק. בעת קריאת שלב ישנה הנחה לתקינות הקובץ.
8. כדי ללמוד את ה-**state** בצורה עצמית, השתמשנו במקורות המידע שמצויים בפירוט במחלקת state.h.
9. טבלת גיבוב – השתמשנו במידע המצוי ב- **cpp reference**. קישור נמצא במחלקת hashHandler.