***פרויקט באסמבלי***

***מאת:*** אביב בן שוהם, י"ב4.

***נושא:*** משחק מבוך.

***בית ספר:*** תיכון הדרים הוד השרון.

***תאריך הגשה:*** 4.8.18

\*\*\*\*\*\*\*\*\*\*האם להסביר על הפעולות שמדפיסות ומעלות את התמונה\*\*

מידע על המשחק

משחק המבוך הוא משחק המבוסס על חידה, בה הפותר נאלץ למצוא את דרכו הנכונה אל עבר היציאה מהמבוך, הבנוי ממעברים מתפצלים.

סביבת עבודה: TASM

***מדוע בחרתי בנושא זה לפרויקט***

תחילה, כאשר התבקשתי לעשות פרויקט באסמבלי, רציתי אף מאוד לעשות משחק בעל גרפיקה מכיוון שהסתקרנתי בדרכי יצירת משחק גרפי בשפות תכנות נוספות באותו העת, כמו כן רציתי להציב לעצמי אתגר, לבנות משחק בעצמי לבד ללא עזרת אחרים ואף על כן גם רציתי לתכנת תוכנה לא פשוטה ומאתגרת בשפה בה התקשתי הכי הרבה, הלוא היא אסמבלי.

בנוסף לכך, שיחקתי בGTA (משחק מחשב ברמה כל כך גבוהה); היה לי מובן שאיני יכול לעשות דבר הדומה לכך, אך באחת מהמשימות שם ראיתי מין "שלב מבוך", ראיתי כי אני יכול להצליח לתכנת משהו לא משנה כמה זעיר הוא ביחס למשחק המפורסם והמוצלח הזה וכך קיבלתי השראה לתכנת מבוך.

לינק להשראה (0:20- 3:20):

https://www.youtube.com/watch?v=4r1xJE3y-ek&t=158s

*הוראות הפעלה*

על מנת להפעיל את המשחק תחילה יש לבדוק האם קובץ המשחק (avivbs1.asm) ו2 התמונות בהן יש שימוש במשחק נמצאות גם כן בתיקיה זו. לאחר מכן, יש להיכנס לדוסבוקס, לעבור אל כונן c

ולאחר מכן לעבור לתיקיית tasm, לכתוב tasm avivbs1.asm, אחר כך tlink avivbs1, ואז avivbs1 על מנת להריץ את המשחק.

בתחילת המשחק, יוצגו את הוראות המשחק ואיך משחקים ומנצחים ואחר כך המשחק יחל כאשר המשתמש ילחץ על מקש P.

בסיום המשחק תודפס הודעת הניצחון ויש ללחוץ על כל לחצן כדי לצאת מהמשחק אל מסך הוראות המשחק בחזרה. (ניתן לסגור את המשחק בעזרת מקש Q במסך הוראות המשחק).

*אלגוריתם הפרויקט*

בתחילת התוכנית, מתרחש אתחול של הפרמטרים (לשלב מאוחר יותר של התוכנית בה המשתמש נפסל ומעוניין להתחיל מחדש). לאחר מכן מתבצע הצג של האינטרודוקשיין והסבר על המשחק, כדי להתחיל את המשחק על המשתמש ללחוץ P, על מנת לצאת עליו ללחוץ Q (כמו כן במהלך המשחק במידה ורוצים לצאת ניתן ללחוץ על 0).

אחר כך, ישנו מעבר למצב גרפי, והתוכנה קוראת לפרוצדורה Loadpic המכילה קריאה לפרוצדורות נוספות אשר מטרתן להציג את התמונה. וכעת למעשה המשחק באמת מתחיל. כדי לאפשר לשחקן להסתגל לסביבת המשחק שהרגע הועלתה בפניו ישנו דיליי של כמה שניות בעזרת לולאה בתוך לולאה בתוך לולאה (שלמעשה לא עושה כלום מלבד לעכב את התוכנה).

והמשחק יוצא ואיתו לולאת המשחק, התוכנית בודקת תנועה של השחקן האם הוא שינה את כיוונו בעזרת הפעולה checkmovement בה מתבצעת בדיקה האם השחקן לחץ על אחד ממקשי החצים או הWASD ומשנה את כיוון תנועתו בהתאם.

לאחר מכן מתבצעת בדיקה האם השחקן ביקש לצאת או נפסל בעזרת השוואת הפרמטרים forceExit וlostGame בהם ערך שאינו 0 נחשב כtrue. במידה וlostGame מתקיים התוכנית תציג הודעת פסילה ולאחר לחיצה על מקש מסוים תחזור למסך האינטרודוקשיין, כאשר forceExit מתקיים התוכנית ישירות תעבור למסך האינטרו. במידה ושניהם לא מתקיימים השחקן יתקדם צעד אחד לכיוון הנוכחי שאליו פונה או שינה את כיוונו בעזרת הפעולה MoveOneStep.

לאחר מכן תתבצע בדיקה האם השחקן הצליח וצלח את השלב הראשון, במידה ולא לולאת המיין תחזור על עצמה שוב. אם הצליח התוכנית תטען את שלב 2 בעזרת loadLevel2 אשר בתוכה נמצא LoadPic בו השתמשנו בתחילת התוכנית, כמו כן התוכנית תאתחל את הפרמטרים למצב התחלה בשלב 2.

במקום לולאת המיין תרוץ כעת לולאת מיין משנית המותאמת לשלב שני.היא תפעל באותו סדר פעולות ובאותן פעולות כפי שתיארתי פה אך במקום שבמצב הצלחה התוכנית תטען את שלב 2 היא תדפיס הודעת ניצחון ולאחר לחיצה על מקש היא תחזור למסך ההוראות.

פירוט הפרוצדורות:

checkMovement: פרוצדורה בודקת האם הקלט שהתקבל (במידה והתקבל) שווה לאחד מערכי מקשי התנועה WASD, במידה ואכן אחד ממקשים אלו התקבל כקלט הפרוצדורה משנה את פרמטר כיוון התנועה. כמו כן היא בודקת אם המקש 0 התקבל, אשר עוצר את התוכנית. במידה ולא היא קוראת לפרוצדורה ReplaceKey.

ReplaceKey: פרוצדורה זו בודקת האם המקש שנקלט הוא אחד מהחצים על ידי ערך האסקי שלו, ומעדכנת בהתאם את כיוון התנועה העתידי.

MoveOneStep: פעולה זו מעדכנת את הפיקסל הנוכחי מצבע ראש השחקן לצבע זנבו בעזרת הפרוצדורה pixel ומעדכנת את פרמטרי מיקום הx והy של השחקן הנמצא במבוך לפי כיוון התנועה הנוכחי של השחקן. לאחר מכן היא קוראת לפרוצדורה CheckWhatNextCell: בה התוכנית בודקת האם המיקום אליו השחקן עומד להתקדם עכשיו פנוי, לפי הצבע הנמצא בו, התוכנית תעדכן את ax אותו נבדוק כשהפעולה תסתיים ונחזור לMoveOneStep, לפי המצבים הבאים: הפיקסל פנוי, הפיקסל תפוס בצבע שגורם להפסד, הפיקסל תפוס בצבע המסמן ניצחון.

MoveOneStep בודקת את ערך ax המוחזר מהפעולה ומעדכנת את פרמטרי הניצחון וההפסד לפיה, ולאחר מכן מדפיסה את הפיקסל שכעת נבדק בצבע ראש השחקן במידה והפיקסל אכן פנוי ואינו גורם לניצחון או פסילה.

\_100MiliSecDelay: פעולה זו מכילה לולאה בתוך לולאה כאשר שתיהן לולאות גדולות שלמעשה אינן עושות דבר מלבד "להעסיק" את התוכנית בהרצתן ובכך להעביר 100 אלפיות שנייה.

pixel: מצייר פיקסל לפי הפרמטרים xplace yplace.

LoadLevel2: פעולה זו מעדכנת את הפרמטרים לפי הערכים ההתחלתים של שלב 2 וקוראת לLoadPic על מנת להעלות את התמונה של השלב השני במבוך.

**תיאור המשתנים בתוכנית**

|  |  |  |
| --- | --- | --- |
| שם המשתנה | טיפוס | תיאור המשתנה |
| SHOVAL\_COLOR | Byte | מספר המתאר את צבע גוף השחקן |
| HEAD\_COLOR | Byte | מספר המתאר את צבע הראש של השחקן |
| BG\_COLOR | Byte | מספר המתאר את הצבע עליו השחקן יכול לדרוך |
| SUCCESS\_COLOR | Byte | מספר המתאר את הצבע על השחקן להגיע על מנת לעבור בהצלחה את השלב |
| xplace | Byte | מספר המתאר את מיקומו על ציר הx של השחקן |
| yplace | Byte | מספר המתאר את מיקומו על ציר הy של השחקן |
| Direction | Byte | תו המתאר את כיוון התנועה אליו נע השחקן, תו זה יכול להיות אך ורק WASD |
| level1Maze | Byte | מחרוזת בה כתוב את שמה של התמונה של השלב הראשון במבוך |
| level2Maze | Byte | מחרוזת בה כתוב את שמה של התמונה של השלב השני במבוך |
| LostMsg | Byte | מחרוזת בה כתובה הודעת הפסד |
| CongratsMsg | Byte | מחרוזת בה כתובה הודעת ניצחון |
| Instructions | Byte | מחרוזת המציגה את הוראות המשחק |
| forceExit | Byte | מספר המתחיל כ0 המייצג האם השחקן ביקש לצאת מהמשחק, במידה וביקש הפרמטר יהפוך ל1 |
| LostGame | Byte | מספר המתחיל ב0 המייצג האם השחקן נפסל או לא, במידה וכן הפרמטר יהפוך ל1 |
| levelFinished | Byte | מספר המתחיל ב0 המייצג האם השחקן עבר את השלב או לא,במידה וכן הפרמטר ישונה ל1. |