

הנדסת חשמל

מעבדה יישומי מחשב להנדסת אלקטרוניתיקה

פרויקט – המשחק סנייק

מרצה: מר דמיטרי טורצ'ינסקי

תאריך : 06.02.19

מאת:

**טל מיטרני 204738157
עadel falah**

רקע תיאורטי

מהות התוכנה- כתיבת קוד למשחק סנאי.

כבר בהתחלה החלנו לכתוב קוד למשחק לפרויקט. כאשר חשבנו על העיון רצינו שהמשחק יהיה קלאסי וממכר. כך, חשבנו לכתוב קוד שיממש את המשחק "Snake", שהעביר להרבה אנשים את הזמן בטלפון Nokia. המשחק שלנו כולל 2 אופציות:

משחק יחיד – נחש אחד שצובר נקודות כל אכילת תפוח. המטרה היא לצבור כמה שיוטר נקודות, כאשר כל צבירת נקודה מגדילה את הנחש. המשחק נגמר אם הנחש מתנגש בדפנות ה- canvas או אם התנגש בעצמו.

משחק של שני נחשים – שני שחנים שלכל אחד נחש משלו. מטרת השחקנים היא לצבור נקודות עברו כל אכילת תפוח, כאשר אכילת תפוח מגדילה את הנחש שאכל אותו. המנצח יהיה זה שיצבור הcci הרבה נקודות ולא יתנגש בנחש השני, עצמו, או בדפנות ה- canvas.

תוצאה: צירפנו לחץ INSTRUCTION שבו כתוב איך משחקים ו גם על חוקי המשחק ואיך התוצאה הסופית מוחשבת, התוצאה מבוססת על כמה הנחש אכל תפוחים ויש הורדת נקודות תלוי באיך הנחש מפסיד:
התנגשות בקייב:-1 נקודות לתוצאה הסופית.

נחש שמתנגש בעצמו:-2 נקודות מהתוצאה הסופית .

נחש שמתנגש בנחש השני :-3 נקודות מהתוצאה הסופית.

הקוד מבוסס על ציור ב- canvas של הנחשים והתפוחים. בהתחלה של המשחק הנחש/הනחים מתחילה ישרות תנעה ימינה מצד שמאל של ה- canvas ומופיע תפוח על ה- canvas בזרה רנדומלית. תפוח חדש מופיע רק לאחר התפוח הנוכחי שופיע. כמו כן, מופיע ניקוד על המסך של כל נחש. בתום המשחק, מוצגת טבלה של עשרת המתמודדים שצברו את הניקוד הcci גבואה. לצד הטבלה ישנה אפשרות לצאת מהמשחק או לחזור ל panel הראשו בו בחרים איזה סוג משחק לשחק (משחק יחיד או משחק של שני נחשים).

הקוד של המשחק פועל כל עוד אחת מהאפשרויות שפורטו לא תתקיים.

הגדרת המשתנים:

int panelhandle –

הפאנל הראשי של המשחק



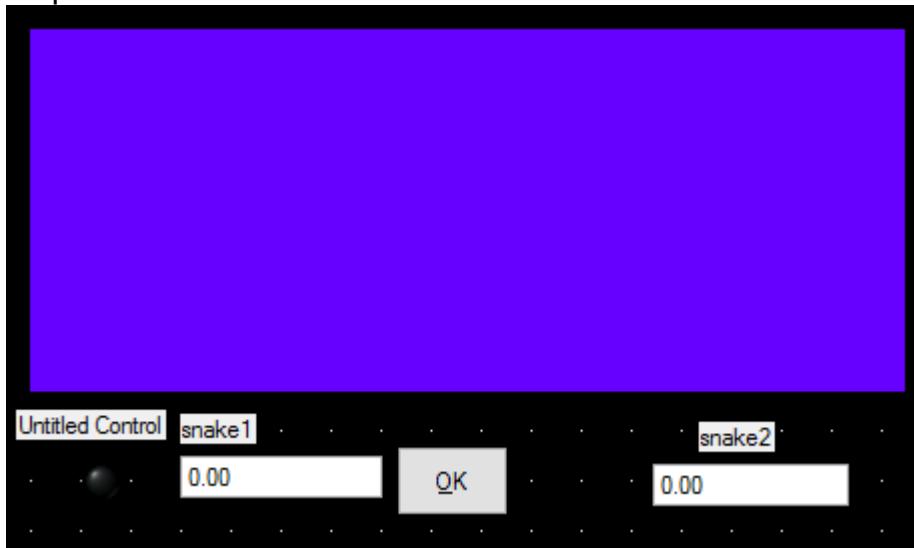
.INSTRUCTIONS-לחצן המסביר את חוקי המשחק.

.1PLAYER-משחק של שחקן אחד.

.2PLAYERS-משחק של שני שחקנים.

.QUIT-יציאה מהמשחק.

int panelhandle2 –



הפאנל של סביבת העבודה של המשחק

.OK-הוא לחצן שמבצעיל את המקלדת.

.SNAKE1-סופר כמה תפוחים אכל הנחש הראשון.

2E-SNAKE-סופר כמה תפוחים אכל הנחש השני.
הלו נדלק כל פעם שאנחנו לוחצים על משאנו במקלדת.
Int panelhandle3



הפאנל של התוצאה הסופית .
WINNER-מראה מי ניצח את המשחק .
שני הנקודות מראים את התוצאה של כל אחד .
MENU-אמור להחזיר אותנו לפאנל הראשי .
QUIT-מסיים את המשחק .
HIGHSCORE-מראה את התוצאות הכי טובות שהיו במשחק .

```
double cnt - סופר את הנקוד של הנחש הראשון
double cnt1 - סופר את הנקוד של הנחש השני
double result- מחשב את התוצאה של הנחש הראשון
double result1 - מחשב את התוצאה של הנחש השני
```

```

FILE * fp-
שומר את התוצאה הגבוהה יותר כל משחק.

Double arr[SIZE]={0}
מקבל את התוצאות מהקובץ ושם אותם בטבלת התוצאות הכי טובות.

CmtThreadFunctionID f1ID – תז' של ה- thread הראשון.

CmtThreadFunctionID f2ID- תז' של ה- thread השני.

CmtThreadPoolHandle p1handel – הגדרת שם של poolhandel

Int appleSize = 10 – הגדרת גודל הרוחב והארוך של התפוח בציור

Int snakeSize = 10 – הגדרת גודל הרוחב והארוך של איבר אחד במרחב של סנייק בציור.

Int MinimunX, MaximunX, MinimunY, MaximunY- מייצגים את גודל הקנבס.

Int NumX, NumY – מייצגים את מיקום התפוח בקנבס.

Int SizeOfSnakesPos, SizeOfSnakesPos2- מייצגים את מספר הערכים הרלוונטיים בכל מרחב (כלומר את גודל הנחש).

Int EatAnApple – משתנה שמייצג בוליאניות: אם אכל המשתנה יהיה שווה 1 אחרת 0.

Int numberOfSnakes – מייצג אם לחצנו על משחק יחיד או של שני שחקנים.

Enum snakedirection {NONE, UP, DOWN, LEFT, RIGHT, QUIT} – מקבל את אחד מהשמות בסוגרים המסלולים.

Enum snakedirection Direction = RIGHT- משתנה שמייצג את כיוון ראש הנחש הראשון. מתחילה בערך "ימינה" ויכול לקבל את אחד הערכים שבסוגרים המסלולים.

Enum snakedirection Direction = RIGHT- משתנה שמייצג את כיוון ראש הנחש השני. מתחילה בערך "ימינה" ויכול לקבל את אחד הערכים שבסוגרים המסלולים.

Struct SnakePos snakePositions [1000] – מערך לנחש הראשון שמכיל 1000 איברים אשר כל איבר מכיל את מיקום האיבר על ה- canvas וכיוונו.

Struct SnakePos snakePositions2[1000] –

```

מערך לנחש השני שמכיל 1000 איברים אשר כל איבר מכיל את מיקום האיבר על ה- canvas וכיונו.

גדרת ה- struct :

```
struct SnakePos
{
    int x,y
    enum snakedirection direct
}
```

כלומר מבנה בעל ערך שמייצגים את המיקום על ה- canvas, ומשתנה שיכל להציג את המילים NONE, UP, DOWN, LEFT, RIGHT, QUIT המיצגים את כיוון הנחש.

פונקציות ופעולות:

- Void DrawApple()

פונקציה פשוטה לציר של התפוח. קוראים לפונקציה לאחר מחיקת canvas.

- Int AddToTheBegining (struct SnakePos, struct SnakePos snakeArr [], int sizeOfSnake)

פונקציה שמקבלת את המיקום החדש של ראש הנחש, מערך הנחש, וכמות האיברים הרלוונטיים של הנחש (גודלו במשחק). הפונקציה מזיזה את האיבר במקומות 1-N למקומות N-1, וכך ממקום 2-N למקום 1-N וכן הלאה עד שהמקום האפס עובר למקומות 1-1 של המערך. נמייצג את sizeOfSnake. ולבסוף שמים את המיקום החדש שקיבלנו במקום האפס במערך. הפונקציה מחזירה את כמות האיברים הרלוונטיים פלוס 1 (return (sizeOfSanke+1))

- int CheckLose()

פונקציה שבודקת אם אחד מהנחיים הפסיק. הפסד מוגדר אם הנחשים התנגשו בקירות, עצם או אחד בשני. הפונקציה בודקת האם המיקום של הראש נמצא מחוץ לתחום של ה- canvas לכל אחד מהנחיים. כמו כן, היא עוברת ב- for האם אחד מהראשים של הנחשים נמצא באותה מקום של המערך של השני או של עצמו. הפונקציה מחזירה 1 אם הנחש הראשון הפסיק, 2 אם השני הפסיק ו-0 אם אף אחד לא הפסיק.

- Struct SnakePos UpdateDirect (enum snakedirection dir, struct SnakePos headOfSnake)

פונקציה שמקבלת איזה ללחץ נלחץ לפי המשמש (למעלה, למטה, ימינה או שמאל), ואת המיקום הנוכחי של הראש. הפונקציה בודקת לאיזה כיוון נלחץ ומעדכנת את המיקום הבא לפיה.

-Int CVICALLBACK KeyCallback (int panel, int control, int event, void *callbackData, int eventData1, int eventData2)

פונקציה שבודקת אם נלחץ מקש במקלדת. במידה וכן בודקת איזה מקש נלחץ. לפי המKeySpec שנלחץ המשתנה direction1-direction2 (במידה ושחקים שני שחקנים) מקבלים את סוג המKeySpec. כמו כן, בבדיקה אנו בודקים אם הכוון שנלחץ הוא ההפוך שאליו הוא כבר אז. אם כן המשתחנים direction1-direction2 לא מקבלים ערכים חדשים. (כלומר, אם הנחש אז למעלה, לחיצה על המKeySpec למטה לא תנסה כלום, וכן להפר. ואם הנחש אז ימינה לחיצה על המKeySpec שמאלת לא תנסה כלום, וכן להפר).

(-Void InitSnakePosition()

פונקציה שמכניסה ערכים ראשוניים לנחשים. אם המשחק הוא משחק יחיד, יכנס ערכים של x ו-y, לפי גודל ה-canvas (בערך x יהיה ב-0 ובציר x יהיה במרכז ה- canvas ובציר x יהיה 0 + גודל התפוח שעוטו קבועו שהיה 10x10). גם כל איבר במערך שלנו ציר בגודל 10x10, וכך כשמכניסים איבר למערך מכינים ערך של x ו-y פלוס 10. אם המשחק הוא משחק לשני שחקנים, נאתחל את המיקום של הנחש הראשון בציר x=0 , ו- אמצע הקנבס פחות 20 = y.

(-Void InitSnakePosition2()

מעדכן את המיקום הראשון של הנחש השני כמו הראשון רק שבציר y מוסף 20 ולא מחסר.

-Int CVICALLBACK TowSnakes (int panel, int control, int event, void *callbackData, int eventData1, int eventData2)

פונקציה שבודקת אם נלחץ הכפתור ב-panel הראשון 2vs2 . אם כן, הוא שם במשתנה numberofsnakes=2 משמש אותנו להマーク לבדיקה כמה נחשים צריך לצייר ב-canvas. כמו כן הוא מפעיל את הפונקציה PLAY.

-Int CVICALLBACK PLAY (int panel, int control, int event, void *callbackData, int eventData1, int eventData2)

פונקציה שבודקת אם נלחץ ב-panel הראשון לחצן Play או 2vs2 . אם כן, היא מפעילה את הפאנל השני של המשחק עצמו, מכניתה ערכים למשתנים Maximum ו-YMaximun את הגודל של ה-canvas, מפעילה את הפונקציה InitSnakePosition () , מפעילה את הפונקציה () InitSnakePosition () Threadpool חדש שמכין 2 פונקציות שיכולות לרווח במקביל. כמו כן, אנו מפעילים כל thread על אחד פונקציה keyCallback עם ID-f1ID וועל השני SnakeMove עם ID-f2ID .

-void Apple()

פונקציה שיוצרת תפוח במקומות רנדומלי ב-canvas. הפונקציה Random מביאה ערכים רנדומליים בין 0 לגודל המקסימלי של ה-canvas. השתמשנו בפונקציה Random גם לציר x וגם לציר y. לאחר מכן אנו רצים על המערך הרלוונטי שלנו גם על הנחש הראשון וגם אם יש שני שחknim, על המערך של הנחש השני, כדי לבדוק אם המיקום הרנדומלי שנוצר שווה לאחד מהמקומות בהם מצורים הנחשים. הפונקציה Random ממחפשת ערכים כל עוד לא נמצא מקום שבו מיצר נחש. כאשר קיבלוינו ערכים אלו נצא מולאota while ואז במשתנים XNum ו-YNum יהיו המיקום של תפוח.

-Int CVICALLBACK QuitCallback (int panel, int control, int event, void *callbackData, int eventData1, int eventData2)

למשתנה Direction את המילה QUIT כדי שיעול ליצאת מהלולאה בפונקציה SnakeMove שפועלת כל עוד Direction!=QUIT, והפונקציה סוגרת את התוכנית (כל הפנלים).

-SnakeMove()

הפונקציה רצה כל עוד לא לחצנו Quit או שאחד מהנחשים לא נפסל. הפונקציה מוחקמת את ה-canvas כל סיבוב של ה-while כדי שנוכל לעדכן את מיקום הנחש מבלי לשכפל את הציור שהוא לפני. כל נחש מתחילה מכומות של 2 איברים ראשוניים במערך, המשטנה שהגדנו שסופר את כמות הרלוונטיים בכל מערך של נחש מתחילה מגודל 1 (כי אנחנו כוללים את הסירה 0 בספירה של כמות האיברים הרלוונטיים). ציור הנחשים נעשה על ידי פונקציית draw של מערכי הנחשים, אשר פועלת עבור כל איבר במערך באמצעות לולאת for שרצה מ-0 עד האיבר הרלוונטי. לפני כל ציור הנחש מחדש, מתעדכנים ערכי ה-*x* וה-*y* של כל איברים הרלוונטיים לפי כיוון הראש באמצעות הפונקציה AddToTheBigining . מכיוון שימוש בפונקציה AddToTheBigining מגדילה לנו את גודל המערך, אנו צריכים לבדוק אם אכן גודל הנחש או לא. לכן, שואלים אם המיקום של התפוח נמצא באותו מקום של אחד מהראשים .

אם כן:

- בודקים איזה מהראשים אכל את התפוח. במידה והנחש הראשון אכל, אנו לא מחסרים מהמשטנה שמקיל את כמות האברים

הרלוונטיים של נחש זה וכן מחסרים מהמשתנה שמכיל את כמות האברים הרלוונטיים של הנחש השני (הוא לא גדול ולכך מחסרים מה שגדל כאשר השתמשנו בפונקציה AddToTheBigining). הדבר קורה הפור אם השני זה שאכל את התפו.

- קוראים לפונקציה (Apple כדי ליצור תפוח חדש.
- ומגדילים את הניקוד של הנחש אשר אכל את התפו.

במידה ולא:

- אנו מחסירים משנה הנחשים את הגודל הרלוונטי (מן שגדל כשקראנו לפונקציה AddToTheBigining והוא לא באמת צריך לגודל מפני שאף אחד לא אכל את התפו).
- מצירים את שני הנחשים לפי המיקומים החדשם על ידי לולאת for שרצה על כל הערכים הרלוונטיים.

וכאשר לחצנו על ZQ או שאחד מהנחשים נפסל בודקים اذا נחש נפסל ומחשבים את התוצאה של כל נחש לפי חוקי המשחק, ואחריו זה בודקים اذا מבין הנחשים ניצח ושמים מודעה לכך על המסך אחרי שהמשחק נגמר וגם שומרים את התוצאה שמצות בתוך קובץ ומכוונים את הערכים שיש כבר בקובץ לטור מערך וממינים את המערך במילון BUBBLE SORT ואז שמים את הערכים שיש בתוך המערך לטבלת HIGHSCORE בכדי להראות את התוצאות היכי טובות שעד עכשו הגענו אליהם.

קשיים בהם נתקלנו:

- הקשי הראשון שנתקלנו היה לדעת איך לקבל לחוץ שנלץ. לאחר בירור באינטרנט ודוגמאות של CVI הצלחנו למש את הפונקציה באמצעות KeyCallback וה- event: EVENT_KEYPRESS .
- הקשי השני היה ליצור תפוח בצורה רנדומלית. לאחר הסתכלות האינטרנט מצאתי כמה פונקציות שמאפשרות להוציא מספר רנדומלי והיחידה שהצלחנו לעובד אותה היא Random. פונקציה זו אמרה לספק איבר רנדומלי בין המספרים שונים לה. למרות שהוא אכן מספקת מספר רנדומלי, היא חזרה על אותו ערך באופן סדר כל משחק חדש. לכן, לצערנו מימוש התוכנית עם פונקציה זו אינו אופטימלי.
- קשי נוסף היה לסגור את התוכנית עוד לפני שהפונקציה draw נסתה לצייר. פתרנו זאת באמצעות if לפני הציור.
- הקשי העיקרי היהקדם את הנחש. בתחילת חטיבנו לקדם כל איבר במערך בלבד. הדבר היה מאד מסובך ולא יעיל. לכן, לאחר חיפש באינטרנט הבנו שצריך רק למחוק את האיבר האחרון ולעדכן את הראש ואז לצייר את המערך. הפתרון יצר ייעילות רבה לתוכניות ונעשה על ידי הדברים שפירטנו בכל פונקציה.
- קשי נוסף שלא נפתר הוא כאשר לוחצים על כמה לחצנים במקביל אחת (לדוגמה חץ ימינה ושמאל ביחד).
- התקשינו גם בלחזור לפאנל הראשי ולשחק שוב אחרי שהמשחק נגמר.

טריקים בהם השתמשנו:

-השתמשנו בכל מיני פונקציות מיוחדות של CVI כמו הפונקציה לדעת את ערכי המקלדת ולדעת להשתמש בה.

-השתמשנו בפונקציית POPOUT שהוא מסבירה את חוקי המשחק .

-לחזור לפונקציה הראשית ולשחק שוב השתמשנו בפונקציה DEFAULTPANEL.