

הנדסת חשמל

מעבדה יישומי מחשב להנדסת אלקטרוניקה

פרויקט – המשחק סנייק

מרצה: מר דמיטרי טורצ'ינסקי

תאריך : 06.02.19

מאת:

טל מיטרני 204738157

עאדל פאלח

רקע תיאורטי

מהות התוכנה - כתיבת קוד למשחק סנייק.

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

משחק יחיד – נחש אחד שצובר נקודות כל אכילת תפוח. המטרה היא לצבור כמה שיותר 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 שחקנים שלכל אחד נחש משלו. מטרת השחקנים היא לצבור נקודות עבור כל אכילת תפוח, כאשר אכילת תפוח מגדילה את הנחש שאכל אותו. המנצח יהיה זה שיצבור הכי הרבה נקודות ולא יתנגש בנחש השני, בעצמו, או בדפנות ה-canvas. תוצאה: צירפנו לחצן INSTRUCTION שבו כתוב איך משחקים וגם על חוקי המשחק ואיך התוצאה הסופית מחושבת, התוצאה מבוססת על כמה הנחש אכל תפוחים ויש הורדת נקודות תלוי באיך הנחש מפסיד: התנגשות בקיר: 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/הנחשים מתחילים ישירות תנועה ימינה מצד שמאל של ה-canvas ומופיע תפוח על ה-canvas בצורה רנדומלית. תפוח חדש מופיע רק לאחר התפוח הנוכחי שמופיע. כמו כן, מופיע ניקוד על המסך של כל נחש. בתום המשחק, מוצגת טבלה של עשרת המתמודדים שצברו את הניקוד הכי גבוהה. לצד הטבלה ישנה אפשרות לצאת מהמשחק או לחזור ל-panel הראשון 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.

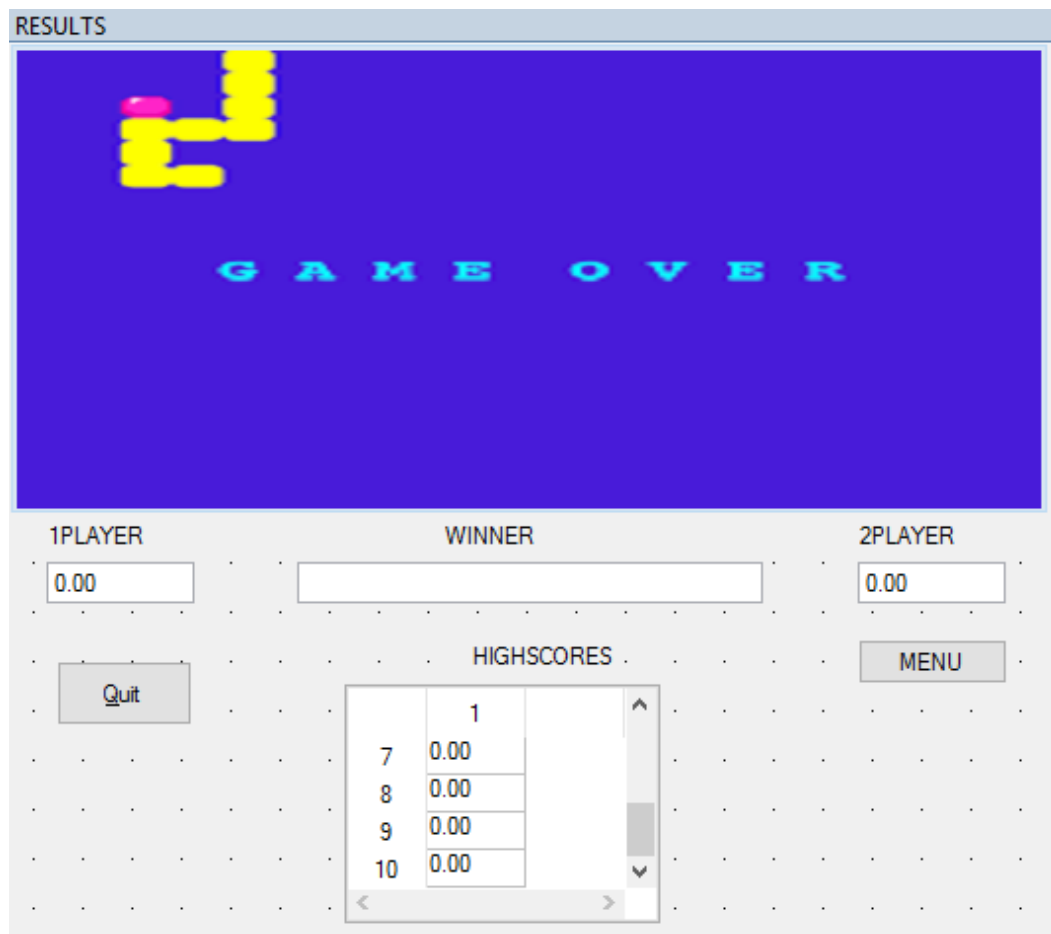
int panelhandle2 –



הפאנל של סביבת העבודה של המשחק

OK-הוא לחצן שמפעיל את המקלדת.
SNAKE1-סופר כמה תפוחים אכל הנחש הראשון.

SNAKE2-סופר כמה תפוחים אכל הנחש השני.
הלד נדלק כל פעם שאנחנו לוחצים על משהו במקלדת.
Int panelhandle3



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

double cnt –

סופר את הניקוד של הנחש הראשון

double cnt1 –

סופר את הניקוד של הנחש השני

double result–

מחשב את התוצאה של הנחש הראשון

double result1 –

מחשב את התוצאה של הנחש השני

FILE * fp-

שומר את התוצאה הגובהה יותר כל משחק.

Double arr[SIZE]={0}

מקבל את התוצאות מהקובץ ושם אותם בטבלת התוצאות הכי טובות.

CmtThreadFunctionID f1ID -

ת.ז של ה- thread הראשון.

CmtThreadFunctionID f2ID-

ת.ז 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 במערך 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 מערך (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 בסוגריים המסולסלים.

Enum snakedirection Direction = RIGHT-

משתנה שמייצג את כיוון ראש הנחש הראשון. מתחיל בערך "ימינה" ויכול לקבל את אחד הערכים שבסוגריים המסולסלים.

Enum snakedirection Direction = RIGHT-

משתנה שמייצג את כיוון ראש הנחש השני. מתחיל בערך "ימינה" ויכול לקבל את אחד הערכים שבסוגריים המסולסלים.

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 וכיוונו.

גדרת ה-struct :

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

כלומר מבנה בעל x,y שמייצגים את המיקום על ה-canvas, ומשתנה שיכול להכין את המילים NONE, UP, DOWN, LEFT, RIGHT, QUIT המייצגים את כיוון הנחש.

פונקציות ופעולות:

- Void DrawApple ()

פונקציה פשוטה לציור של התפוח. קוראים לפונקציה לאחר מחיקת ה-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 במיקום N-1 למיקום N במערך, וכן ממיקום N-2 למיקום N-1 וכך האלה עד שהמיקום במקום האפס עובר למיקום במקום ה-1 של המערך. N מייצג את sizeOfSnake. ולבסוף שמים את המיקום החדש שקיבלנו במקום האפס במערך. הפונקציה מחזירה את כמות האיברים הרלוונטיים פלוס 1 (return (sizeOfSnake+1)).

-int CheckLose()

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

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

פונקציה שמקבלת איזה לחצן נלחץ לפי המשתמש (למעלה, למטה, ימינה או שמאלה), ואת המיקום הנוכחי של הראש. הפונקציה בודקת לאיזה כיוון נלחץ ומעדכנת את המיקום הבא לפי זה.

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

פונקציה שבודקת אם נלחץ מקש במקלדת. במידה וכן היא בודקת איזה מקש נלחץ. לפי המקש שנלחץ המשתנה direction ו-direction2 (במידה ומשחקים שני שחקנים) מקבלים את סוג המקש. כמו כן, בבדיקה אנו בודקים אם הכיוון שנלחץ הוא הכיוון ההפוך שאליו הוא כבר זז. אם כן המשתנים direction ו-direction2 לא מקבלים ערכים חדשים. (כלומר, אם הנחש זז למעלה, לחיצה על המקש למטה לא תשנה כלום, וכן להפך. ואם הנחש זז ימינה לחיצה על המקש שמאלה לא תשנה כלום, וכן להפך).

-Void InitSnakePosition()

פונקציה שמכניסה ערכים ראשוניים לנחשים. אם המשחק הוא משחק יחיד, יכניס ערכים של x ו-y, לפי גודל ה-canvas (בערך x יהיה ב-0 ובציר ט יהיה באמצע ה-canvas ובציר x יהיה +0 גודל התפוח שאותו קבענו שיהיה 10x10). גם כל איבר במערך שלנו ציור בגודל 10x10, ולכן כשמכניסים איבר למערך מכניסים 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 הראשוני של הנחש השני כמו הראשון רק שבציר y מוסיף 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 שמשמש אותנו להמשך 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, מכניסה ערכים למשתנים MaximunX ו-MaximunY את הגודל של ה-canvas, מפעילה את הפונקציות InitSnakePosition() ו-InitSnakePosition(2) , מפעילה את הפונקציה Apple() ויוצרת לנו Threadpool חדש שמכין 2 פונקציות שיכולות לרוץ במקביל. כמו כן, אנו מפעילים כל thread על אחד פונקציה keyCallback עם ID-f1ID ועל השני SnakeMove עם ID-f2ID .

-void Apple()

פונקציה שיוצרת תפוח במקום רנדומלי ב-canvas. הפונקציה Random מביאה ערכים רנדומלים 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 שחקנים, על המערך של הנחש השני, כדי לבדוק אם המיקום הרנדומלי שנוצר שווה לאחד מהמקומות בהם מציורים הנחשים. הפונקציה Random מחפשת ערכים כל עוד לא נמצא מקום שבו מיוצר נחש. כאשר קיבלנו ערכים אלו נצא מלולאת ה-while ואז במשתנים NumX ו-NumY יהיו הערכים של מיקום התפוח.

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

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

-SnakeMove()

הפונקציה רצה כל עוד לא לחצנו Quit או שאחד מהנחשים לא נפסל. הפונקציה מוחקת את ה-canvas כל סיבוב של ה-while כדי שנוכל לעדכן את מיקום הנחש מבלי לשכפל את הציור שהיה לפניו. כל נחש מתחיל מכמות של 2 איברים ראשוניים במערך, המשתנה שהגדרנו שסופר את כמות הרלוונטיים בכל מערך של נחש מתחיל מגודל 1 (כי אנחנו כוללים את הספרה 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 ששימוש בפונקציה AddToTheBigining מגדילה לנו את גודל המערך, אנו צריכים לבדוק אם אכן גדל גודל הנחש או לא. לכן, שואלים אם המיקום של התפוח נמצא במיקום של אחד מהראשים .

אם כן:

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

- הרלוונטיים של נחש זה וכן מחסרים מהמשתנה שמכיל את כמות האברים הרלוונטיים של הנחש בשני (הוא לא גדל ולכן מחסרים מה שגדל כאשר השתמשנו בפונקציה AddToTheBigining). הדבר קורה הפוך אם השני זה שאכל את התפוח.
- קוראים לפונקציה Apple() כדי ליצור תפוח חדש.
- ומגדילים את הניקוד של הנחש אשר אכל את התפוח.

במידה ולא:

- אנו מחסירים משני הנחשים את הגודל הרלוונטי (מפני שגדל כשקראנו לפונקציה AddToTheBigining והוא לא באמת צריך לגדול מפני שאף אחד לא אכל את התפוח).
- מציירים את שני הנחשים לפי המיקומים החדשים על ידי לולאת for שרצה על כל הערכים הרלוונטיים.
- וכאשר לחצנו על QUIT או שאחד מהנחשים נפסל בודקים איזה נחש נפסל ומחשבים את התוצאה של כל נחש לפי חוקי המשחק, ואחרי זה בודקים איזה מבין הנחשים ניצח ושמים מודעה לכך על המסך אחרי שהמשחק נגמר וגם שומרים את התוצאה שמצחת בתוך קובץ ומכניסים את הערכים שיש כבר בקובץ לתוך מערך וממינים את המערך במיון BUBBLE SORT ואז שמים את הערכים שיש בתוך המערך לטבלת HIGHSCORE בכדי להראות את התוצאות הכי טובות שעד עכשיו הגענו אליהם.

קשיים בהם נתקלנו:

- הקושי הראשוני שנתקלנו היה לדעת איך לקבל לחצן שנלחץ. לאחר בירור באינטרנט ודוגמאות של CVI הצלחנו לממש את הפונקציה באמצעות KeyCallback וה- event: EVENT_KEYPRESS.
- הקושי השני היה ליצור תפוח בצורה רנדומלית. לאחר הסתכלות האינטרנט מצאתי כמה פונקציות שמאפשרות להוציא מספר רנדומלי והיחידה שהצלחנו לעבוד איתה היה Random. פונקציה זו אמורה לספק איבר רנדומלי בין המספרים שנותנים לה. למרות שהיא אכן מספקת מספר רנדומלי, היא חוזרת על אותם ערכים באותו סדר כל משחק חדש. לכן, לצערנו מימוש התוכנית עם פונקציה זו אינו אופטימלי.
- קושי נוסף היה לסגור את התוכנית עוד לפני שהפונקציה draw ניסתה לצייר. פתרנו זאת באמצעות if לפני הציור.
- הקושי העיקרי היה לקדם את הנחש. בתחילה חשבנו לקדם כל איבר במערך בנפרד. הדבר היה מאוד מסובך ולא יעיל. לכן, לאחר חיפוש באינטרנט הבנו שצריך רק למחוק את האיבר האחרון ולעדכן את הראש ואז לצייר את המערך. הפתרון יצר יעילות רבה לתוכניות ונעשה על ידי הדברים שפירטנו בכל פונקציה.
- קושי נוסף שלא נפתר הוא כאשר לוחצים כעל כמה לחצנים במכה 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 שהיא מסבירה את חוקי המשחק .
- לחזור לפונקציה הראשית ולשחק שוב השתמשנו בפונקציה DEFAULTPANEL.