**שם :** מיה עמרם

**מס. ת.ז. :** 322518655

**נושא** **:** Snake AI

**בית ספר ועיר :** עירוני י"ד תל אביב

**שם המנחה :** שמעון קוגן

**שם המורה המלווה :** אורית רביד רדה

העבודה נעשתה במסגרת התכנית "**מדעי המחשב, אקדמיה, ותעשייה**",

מכון דוידסון, מכון ויצמן

שנה : 2018

תוכן עניינים

תקציר.................................................................................................................3

מבוא...................................................................................................................4

מטרת העבודה.....................................................................................................5

ניתוח וניסוח הבעיה האלגוריתמית.........................................................................6

תיאור אלגוריתמים קיימים לפתרון הבעיה...............................................................7

הפתרון המוצע...................................................................................................13

תיאור המודולים ומבני הנתונים של התוכנה...........................................................14

שפת התכנות, סביבת העבודה והכלים..................................................................24

הערכת הפתרון..................................................................................................25

תיאור הממשק למשתמש.....................................................................................26

רפלקציה...........................................................................................................30

ביבליוגרפיה......................................................................................................31

תקציר

בפרויקט שלי בחרתי לעשות את המשחק סנייק הכולל בינה מלאכותית AI (Artificial Intelligent) שבה ניתנת האפשרות לתוכנה לשחק באופן עצמאי בעזרת אלגוריתם למציאת המסלול המוביל אל המטרה.

תמיד עיניין אותי נושא המחשבים ורציתי לשלב משהו שאני אוהבת לעשות, כמו לפתח ולשחק במשחק מחשב שפתחתי בעצמי.

כשהמנחה שלי הציע לי לעשות את משחק המחשב סנייק, התחברתי מאוד לרעיון כי מעניין אותי מאוד איך ליצור משחק מחשב בעצמי. התוספת של בינה מלאכותית למשחק שבו התוכנה תוכל להיות "חכמה" ולשחק לבד במהירות מוסיפה מיממד חדש לפרויקת ומאתגרת את המימוש ורמת הקושי.

הנושא הזה נתן לי אפשרות ללמוד ולהעשיר את הידע שלי באופן ניכר במחשבים בכלל ובפרט בשפת Java Script , להיחשף לעולם הבינה המלאכותית AI שהוא חדש לי לגמרי, לעולם האלגוריטמים , ואולי גם להנות.

מבוא

רקע כללי לנושא העבודה

במשחק שני ממשקים עיקרים :

הראשון בו שולט השחקן (מפעיל אנושי) בעזרת מקשי המקלדת על נחש ארוך ודק המשוטט באזור מבודד, אוסף אוכל, ומנסה להימנע מלהתנגש בזנב שלו או בקירות המקיפים את אזור המשחק. הנחש מתארך כשהוא אוכל, וכך קשה יותר להימנע מלהיתקע בזנב או בגוף עצמו. השחקן שולט על כיוון תנועת ראש הנחש (למעלה, ימינה, למטה או שמאלה), ויתר גופו של הנחש עוקב אחרי ראשו. הנחש נמצא בתנועה מתמדת ולא ניתן לעצור אותו במהלך המשחק.

ובחלק השני,אותם כללי משחק אלא שהתוכנה היא זו שתשחק ותנסה להגיע לנחש הארוך ביותר בזמן הקצר ביותר. זאת בעזרת אלגוריתם למציאת מסלול מתאים.

רוב המשחקים מסוג זה שקיימים כיום, מצריכים הורדה של המשחק למחשב, ואילו האפליקציה מתארחת בסרבר ברשת ותוכל לרוץ בדפדפנים שונים כולל טלפונים סלולארים ,בזכות העובדה שכתבתי אותה ב Java Scriptשזו שפה שדפדפנים יודעים להריץ אותה.

הקוד

<https://github.com/mymaya7/davidson>

המשחק:

<https://rawgit.com/mymaya7/davidson/master/index.html>

מטרת העבודה

לפתח משחק מהנה, שכל אחד יכול לשחק בו ובנוסף להפעילו בצורה אוטומטית בעזרת אלגוריתם AI שידע להתחשב באורך הנחש, מיקום הפיתיון ולעקוף מחשולים כך שבסופו של דבר יגיע אל הפיתיון.

.

ניסוח וניתוח של בעיות אלגוריתמיות

משחק הסנייק המשוחק ע"י המשתמש צריך להתמודד עם מספר משימות:

1. ציור משטח עבודה (קנבס) שעליו יתקיים המשחק
2. ציור פיתיון
3. ציור הנחש וקידומו לכיוון שהמשתמש בחר בקצב קבוע (או לפי הוראות המשתמש)
4. לבדוק בכל תזוזה אם: הפיתיון נאכל, הנחש פגע בעצמו, או שהנחש פגע בשוליים, ולהגיב בהתאם.
5. להאריך את הנחש אם אכל את הפיתיון.
6. מכוון שאחד היתרונות של המשחק הוא שהוא יכול לרוץ בכל דפדפן, חיפשתי דרך שיהיה אפשר לשחק איתו בטלפונים סלולארים. הבעיה היא שבטלפון אין מקלדת.
7. ניהול אוטומטי של המשחק , כלומר שהנחש ישחק בעצמו
8. מעבר בין מצב אוטומטי לידני

תיאור אלגוריתמים קיימים לפתרון הבעיה

* כדי לפתור את משימת הציור והרענון בתוכנית, השתמשתי בספרייה בשם p5.js, אשר מספקת לי תשתית של פונקציות כמו :

Setup- קורית פעם אחת בתוכנית לאחר עלייתה, ובה אני דאגתי לתכל את התוכנית שלי, למשל יצירת הקנבס, הגדרת מספר הפעמים שאנו קוראים ל Draw(כלומר קצב המשחק), בונים נחש חדש, ועדכון מיקום האוכל הראשון.

Draw- פונקציה שקורית במחזוריות ובה, כל פעם מחדש, מצוייר רקע(כל מה שהיה עד עכשיו נמחק), מתעדכן מיקום כל חולייה על מנת לקבל תזוזה, ציור הנחש, וציור האוכל.

בנוסף לאחר כל עדכון של מיקום החוליות יש צורך בבדיקה האם ראש הנחש "נפסל" כלומר האם פגע בעצמו או בשולי המשטח.

-כדי לפתור את משימת ציור הפיתיון, כתבתי פונקציה אשר מגרילה מיקום בצירx ו- y בהתחשב בגודל הקבוע של המשבצות במשחק, ובהם ממקמת את הפיתיון. [updateFood]

- כדי לפתור את משימת ציור הנחש וקידומו לכיוון שהמשתמש בחר בקצב קבוע, כתבתי פונקציה שבה כל חוליה בנחש מקבלת את מיקום החוליה שלפניה, והראש מקבל את מיקומו הקודם ועוד כיוון המהירות הנוכחי. הקצב של ההתקדמות נקבע על פי מספר הפעמים שאנו קוראים לפונקציה draw בשנייה. [updateLocation]

- כדי לבצע בדיקה בכל תזוזה אם: הפיתיון נאכל, הנחש פגע בעצמו, או שהנחש פגע בשוליים, כתבתי פונקציה אשר בודקת האם לראש הנחש יש אותו מיקום כמו לאוכל, אם כו סימן שהאוכל נאכל, או שלראש הנחש יש אותו מיקום כמו אחת החוליות של הנחש, או שמיקום ראש הנחש עבר את גבולות המשחק. [isSelfTouch] ,[ checkEdges] ,[ cheakEat].

* כדי לפתור את משימת הארכת הנחש במידה ואכל את הפיתיון כתבתי פונקציה שבה נוסך איבר חדש לנחש שמיקומו הוא מיקום זנב בנחש. בנחש, ישנה חוליה נוספת בסופו, שלא מצוירת ולא נכללת באורך הנחש שמטרת לשמור את המקום של זנב הנחש כדי לדעת איפה לצייר את החוליה הנוספת לאחר האכילה.[ addJoint]

-כדי לפתור את משימת ההתאמה של המחשק לטלפונים סלולארים, השתמשתי בפונקציה של p5.js וjava Script , שמידעת את התוכנית שלי כל פעם שהמשתמש נגע במסך, ונותנת את מקום הנגיעה. בזכות קבלת מאורעות אלו עשיתי שאם המשתמש נוגע בצד הימני של המסך, הנחש ילך ימינה וכן הלאה. כמו כך התוכנית מאזינה לאירוע שנקרא touchMoved, שמתקבל כאשר המשתמש מחליק את ידו על גבי המסך, במצב זה התוכנית עוברת למצב אוטומטי.(מקביל ללחיצה על מקש A במקלדת).

*המשחק האוטומטי*

- כדי לפתור את משימת המשחק האוטומטי ישמתי את הפונקציה הבאה:

הפונקציה הבאה היא בעצם האלגוריתם החכם. עד כה, בכל פעם המשתמש בחר את כיוון התנועה הבא, בעזרת החצים, והכיוון שבחר המשתמש בחצים, היה כיוון התנועה של הנחש.

הפונקציה calcNextMove מחשבת בעצמה את כיוון התנועה הבא, בהתחשב באוכל, בקצוות, ושהנחש לא יפגע בעצמו:

חלק א:

**var** x=0,y=0;

**if**(**this**.headXpos>Xfood)

{

**if**(**this**.Xspeed!=1)

x=-1;

}

**else** **if**(**this**.headXpos<Xfood)

{

**if**(**this**.Xspeed!=-1)

x=1;

}

**if**(x==0)

{

**if**(**this**.headYpos>Yfood)

{

**if**(**this**.Yspeed!=1)

y=-1;

}

**else** **if**(**this**.headYpos<Yfood)

{

**if**(**this**.Yspeed!=-1)

y=1;

}

}

**if**(x==0 && y==0)

{

**if** (**this**.Xspeed!=0)

y=1;

**else** **if** (**this**.Yspeed!=0)

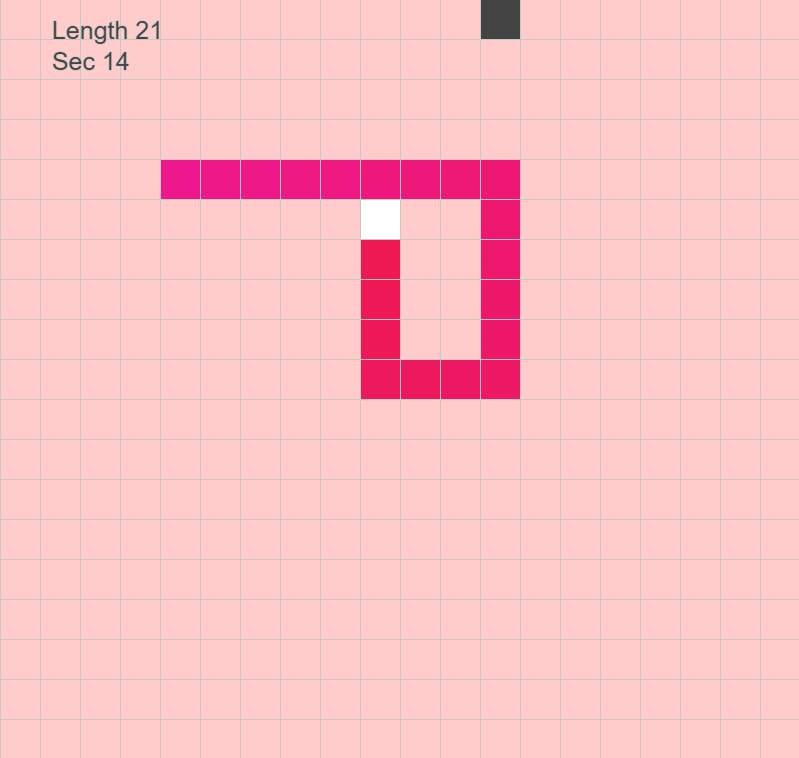
x=1;

בחלק זה, אנו מעדכנים את ערך הצעד הבא לפי מיקום האוכל. למשל אם האוכל מימינו, הנחש ילך ימינה, אם האוכל מתחתיו הוא ילך למטה. אך צריך לשים לב שכאשר הנחש הולך ימינה והאוכל משמאלו, לא נוכל לזוז שמאלה כי אחרת יכנס לתוך עצמו. במקרה זה, הנחש ילך לכיוון אחר ומשם יחשב את הצעד הבא.

בסיום חלק זה ישנו תנאי השואל האם הצעד הבא שחושב כעת, יגרום לנחש לפגוע בעצמו:

**if**(!**this**.isPartOfSnake(**this**.headXpos+x\*Scl,**this**.headYpos+y\*Scl))

**this**.updateDirection(x\*Scl,y\*Scl);



אם לא, זהו הצעד הבא שנבחר והפעולה תעדכן בהתאם (תשלח את ערכי המהירויות החדשים לפעולה updateDirection.

אם הנחש כן יפגע בעצמו לאחר שהנחש יתקדם לפי הצעד שחושב נעבור לחלק ב:

חלק ב:

תחילה, יש if שמטרתו לבדוק האם הנחש לכוד, כלומר מסביבו יש רק מכשולים : או את עצמו, או את הקצוות. אם הנחש נקלע למצד כזה, שהוא לא יכול לצאת ממנו, אין ברירה אלה להיפסל ולכן הוא הולך למקום רנדומלי שם יפסל.(אילו לא הייתי עושה קטע זה, הנחש נקלע ללולאה אין סופית בה הוא מנסה לצאת מהמצב הזה אך ללא הצלחה).

**if**(((**this**.isPartOfSnake(**this**.headXpos+1\*Scl,**this**.headYpos)) || (**this**.headXpos+Scl>=width))

&&((**this**.isPartOfSnake(**this**.headXpos-1\*Scl,**this**.headYpos)) || (**this**.headXpos-Scl<0))

&&((**this**.isPartOfSnake(**this**.headXpos,**this**.headYpos+1\*Scl)) || (**this**.headYpos+Scl>=height))

&&((**this**.isPartOfSnake(**this**.headXpos,**this**.headYpos-1\*Scl)) || (**this**.headYpos-Scl<0)) )

{ // dead end we are going to die :(

x=1;

**this**.updateDirection(x\*Scl,y\*Scl);

}

אם הנחש לא נמצא במלכודת, אך הצעד הבא עומד לגרום לכך שהנחש יפגע בעצמו, נגריל אוכל דמיוני(ההגרלה דואגת שהאוכל לא יפגע בנחש, בכך שמגרילה אוכל חדש אם האוכל שהוגרל הוא על הנחש), את מיקום האוכל הדימיוני נשלח לפונקציה calcNextMove שוב, והנחש ילך לאוכל הדימיוני ובכך יזוז לכיוון אחר ולא יפגע בעצמו.

**else**

{

**var** tmpx,tmpy;

tmpx=int(random(0,width/Scl))\*Scl;

tmpy=int(random(0,height/Scl))\*Scl;

**while**(mySnake.isPartOfSnake(tmpx,tmpy))

{

tmpx=int(random(0,width/Scl))\*Scl;

tmpy=int(random(0,height/Scl))\*Scl;

}

**this**.calcNextMove(tmpx,tmpy);

}

-כדי לפתור את משימת המעבר בין מצב אוטומטי לידני, עשיתי שלחיצה על מקשי המקלדת עוברת בין מצבים. ההקשה על המקשים משנה את המשתנים הבוליאנים שיש בתוכנה וכך נוצר המעבר.

מעבר למצב אוטומטי- (A)

מעבר למצב ידני כלומר ע"י המשתמש- (M)

הפתרון המוצע

בחרתי בשפה Java Script מכוון שהיא מאפשרת ליצור מערך דינמי, וכך ניתן להוסיך חוליה בכל רגע מבלי להצהיר על גודל המערך בתחילת התוכנית.

ובנוסף כל ניתן להריץ את המשחק במכשירים שונים ומערכות הפעלה שונות כי מי שמריץ את התוכנה זה הדפדפן.

בחלק של המשחק האוטומטי, בחנתי מספר אלגוריתמים מסוג למציאת מסלול אופטימלי (path fining) שהפופולארי ביניהם הוא האלגוריתם A\* . אך החלטתי לממש אלגוריתם בעצמי, כדי להיות מקורית ולאתגר את היכולות שלי.

תיאור המודלים ומבני הנתונים של מערכת התוכנה

הנחש במשחק, הוא בעצם **מערך** של עצמים מסוג חוליה. כל איבר במערך מייצג חוליה בנחש, והחוליה היא עצם שקראתי לו joint לכל חוליה יש ערך בציר x וערך בציר y. כלומר הנחש הוא מערך של חוליות.

המשחק בנוי משתי מחלקות, main ו snake.

*Snake:*

מחלקה זו היא בעצם אוסף הפעולות שקשורות לנחש:

isPartOfSnake:

פונקציה שמטרת לבדוק האם משהו מסויים הוא חלק מהנחש. פעולה זו שימושית לבדיקה האם הנחש אכל, או שהנחש פגע בעצמו.

**this**.isPartOfSnake= **function**(x,y)

{

**for**(**var** i=**this**.joints.length-2 ; i >=0 ; i--)

{

**if**( **this**.joints[i].Xpos==x && **this**.joints[i].Ypos==y)

**return** **true**;

}

**return** **false**;

}

updateDirection:

פונקציה שמטרת לעדכן את כיוון התנועה הנוכחי. כיוון התנועה מתקבל או מהמשתש, ע"י החצים במקלדת, או ע"י האלגוריתם החכם שמחליט מהו כיוון התנועה הבא.

**this**.updateDirection = **function**(x,y)

{

**this**.Xspeed=x;

**this**.Yspeed=y;

}

updateLocation:

פונקציה שמטרת "לקדם את הנחש". הנחש אמור להיות בתנועה מתמדת בכיוון שהמשתמש או האלגוריתם החכם בחרו. כדי לעשות זאת, בכל סיבוב כל חוליה מקבלת את מיקום החוליה שלפניה, והראש מקבל את מיקומו ועוד המהירות בציר התנועה.

כדי לעשות פעולה זו אנו משתמשים בפונקציה updateJointLocation, פונקציה שמטרת לעדכן את המיקום של כל חוליה.

**this**.updateLocation = **function**()

{

**if**(**this**.Xspeed!=0 || **this**.Yspeed!=0)

{

**for**(**var** i=**this**.joints.length-1 ; i > 0 ; i--)

{

**this**.joints[i].updateJointLocation(**this**.joints[i-1].Xpos,**this**.joints[i-1].Ypos);

}

**this**.joints[0].updateJointLocation(**this**.joints[0].Xpos+**this**.Xspeed,**this**.joints[0].Ypos+**this**.Yspeed);

**this**.headXpos = **this**.joints[0].Xpos;

**this**.headYpos = **this**.joints[0].Ypos;

}

}

updateJointLocation:

פונקציה שמטרת לעדכן את המיקום של החוליה.

**this**.updateJointLocation = **function**(x,y)

{

**this**.Xpos=x; //ההתקדמות שלו בציר האיקס היא המיקום הקודם ועוד המהירות, אם יש, בציר האיקס

**this**.Ypos=y;

}

:

drawJoint, drawSnake:

פונקציה שמטרת לצייר את הנחש. למעשה הפונקציה מפנה לפנוקציה ששמה drawJoint , שבה מצויירים אברי הנחש בזה אחר זה.

בפונקציה זו נעזרתי בספריית p5.js שמאפשרת ציור של מלבן במיקום רצוע ובאורך ורוחב רצויים. כפי שניתן לראות, לפני ציור החוליות יש צורך להגדיר את הצבע הרצוי של מה שיצוייר, ואני בחרתי לצבוע את הראש בצבע שונה מזה של החוליות. בנוסף לכך, בכל פעם שנוסםה חוליה חדשה צבעה היה שונה מזה של הקודם כדי ליצור את הנחש יותר מעניין.

**this**.drawSnake = **function**()

{

// ציור הסנייק

**for**(**var** i=1 ; i < **this**.joints.length-1 ; i++)

{

fill(238,24,80+i\*3);

**this**.joints[i].drawJoint();

}

// draw the head

fill(255);

**this**.joints[0].drawJoint();

}

**this**.drawJoint = **function**(x,y)

{

rect( **this**.Xpos, **this**.Ypos,Scl,Scl);

}

**++++++++++++++++++++++++++++++++**

**this**.addJoint = **function**()

{

**this**.joints[**this**.joints.length] = **new** joint();

// this.joints[this.joints.length-1].updateJointLocation(this.joints[this.joints.length-2].Xpos,this.joints[this.joints.length-2].Ypos);

}

*main:*

מחלקה זו מנהלת את המשחק:

Setup:

קורית פעם אחת בתוכנית לאחר עלייתה, ובה אני דאגתי לתכל את התוכנית שלי, למשל יצירת הקנבס, וקריאה לפונקציה newgame שקורית בכל פעם שהמשחק מתחיל מחדש(גם לאחר פסילה).

**function** setup()

{

createCanvas(Scl\*Len, Scl\*Len); // המשטח בגודל של מספר המשבצות למשל 20 כפול גודל משבצת

//background(170); //הצבע של הרקע

newgame();

}

newgame :

פונקציה שמטרתה לאתחל את המשתנים הנחוצים, ליצור נחש, ולעדכן את מיקום האוכל הראשוני.

**function** newgame()

{

FPSspeed=5;

**if**(autoMode)

{

FPSspeed=20;

}

Yspeed = 0;

Xspeed = 0;

frameRate(FPSspeed); //מספר הפריימים בשנייה

mySnake = **new** snake();

updateFood();

startGameSecond = millis()/1000;

}

:Draw

קורה X פעמים בשנייה(FPS) ומצייר כל פעם מחדש את הקנבס ואת כל האלמנטים עליו כמו הנחש האוכל, הגבולות של הקנבס.

אם המצב הוא אוטומטי, האלגורים החכם יעדכן את כיוון הנחש, ואילו אם זהו לא מצב אוטומטי, הנחש ימשיך לנוע לכיוון האחרון שהוקש בעזרת החצים ע"י המשתמש. בכל מקרה, לאחר מכן נקראת הפונקציה updateLocation שכפי שהסברתי תגרום לתזוזת הנחש.

הפונקציה הבאה שנקראת היא ציור הנחש, שמצייירת את הנחש כל פעם מחדש כי אנו מוחקים בכל draw את כל מה שהיה לפני.

כעת, יש תנאי, אם הנחש פגע בעצמו או בקצוות, כלומר נפסל, יופיע מלל על המסך והקריאה לפונקצית draw תיפסק.

אם הנחש לא נפסל, יש תנאי האם הנחש אכל את הפיתיון. אם כן, נוספת חוליה בעזרת הפונקציה שעליה הסברתי addJoint, ומוגרל מיקום של אוכל חדש בעזרת הפונקציה updateFood.

**function** draw()

{

background(255,204,204); //נצבע את המשטח כל פעם מחדש(כל מה שהיה לפני נמחק)

drawGrid();

**if**(autoMode)

mySnake.calcNextMove(Xfood,Yfood);

mySnake.updateLocation();

mySnake.drawSnake();

drawFood();

**if**( mySnake.isSelfTouch() || mySnake.checkEdges())

{

**var** currLen = mySnake.joints.length-1;

textSize(52);

fill(100, 102, 153);

textAlign(CENTER);

text(' GameOver ' + second.toFixed(0) + 'Sec \n Total Length is ' + currLen, width/2, height/2);

//createAudio('assets/doorbell.mp3');

noLoop();

**return**;

}

**if**(cheakEat())

{

mySnake.addJoint();

updateFood(Xfood,Yfood);

}

putText();

}

keyPressed :

פונקציה זו היא פונקציה שנקראת אוטומטית כל פעם שנלחץ מקש על המקלדת. הפונקציה שייכת לספרייה p5.js ,והיא יודעת לבדוק איזה מקש במקלדת הוקש. כיוון המקש שהוקש במקלדת(החץ) הוא כיוון התנועה הבא של הנחש.

בנוסף לכך, יצרתי אפשרויות שונות שקורות בלחיצה על מקשי המקלדת:

הגדלת מהירות התנועה של הנחש- (+)

הקטנת מהירות התנועה של הנחש- (-)

מעבר למצב אוטומטי- (A)

מעבר למצב ידני כלומר ע"י המשתמש- (M)

nextMoveValue:

פונקציה זו, מקבלת את כיוון התנועה שהתקבל ע"י המקלדת או ע"י מסך המגע, ומעדכנת את כיוון התנועה ע"י שליחת הערך המתאים לפונקציה updateDirection. למשל אם הוקש החץ ימינה, כיוון התנועה הנוכחי יהיה ימינה( המיקום החדש יהיה המיקום הקודם ועוד SCL כלומר ועוד משבצת), אך צריך לשים לב שכאשר הנחש הולך לכיוון מסוים והתקבל הכיוון הנגדי למשל, הנחש הולך ימינה והכיוון שהתקבל הוא שמאלה למשל, לא נוכל לזוז שמאלה כי אחרת יכנס לתוך עצמו.

**function** nextMoveValue(x,y){

// מאפסים את מהירויות הכיוונים כדי שנישאר רק עם הלחיצה האחרונה

**if**(y === 1){//DOWN

**if**(Yspeed != -Scl)

Yspeed=Scl;

Xspeed=0;

}**else** **if**(y === -1){ //UP

**if**(Yspeed != Scl)

Yspeed=-Scl;

Xspeed=0;

}**else** **if**(x === -1){// LEFT

**if**(Xspeed != Scl)

Xspeed=-Scl;

Yspeed=0;

}**else** **if**(x === 1){ //RIGHT

**if**(Xspeed != -Scl)

Xspeed=Scl;

Yspeed=0;

}**else** **if**(x==0 && y==0){

Xspeed=0;

Yspeed=0;

}

mySnake.updateDirection(Xspeed,Yspeed);

}

drawFood:

פוקציה זו מציירת את האוכל במיקום שנקבע לו בפונקציה updateFood. האוכל מצויר בעזרת הספריה p5.js שמאפשרת לציר ריבוע במיקום רצוי ,בגודל רצוי וצבע רצוי.

**function** drawFood()

{

// ציור הפיתיון

fill(68)

rect(Xfood,Yfood,Scl,Scl);

}

cheakEat:

פונקציה שבודקת האם מיקום הראש זהה למיקום האוכל, אם כן סימן שהנחש אכל את הפיתיון , ומחזירה אמת אחרת מחזירה שקר.

**function** cheakEat()

{

// אם הסנייק אוכל את הפיתיון אז בחירות מיקום חדש לפיתיון

**return**(Xfood==mySnake.headXpos && Yfood==mySnake.headYpos)

}

updateFood:

פונקציה זו מגרילה מיקום של אוכל, בתחום המשחק, כלומר בין הקצוות. אם האוכל נופל על הנחש, הפונקציה מגרילה מיקום חדש לאוכל.

**function** updateFood()

{

**var** x,y;

x=int(random(0,width/Scl))\*Scl;

y=int(random(0,height/Scl))\*Scl;

**while**(mySnake.isPartOfSnake(x,y))

{

x=int(random(0,width/Scl))\*Scl;

y=int(random(0,height/Scl))\*Scl;

}

Xfood=x;

Yfood=y;

}

סביבת עבודה

שפת התכנות היא java script בשימוש בסיפריית p5.js .

דפדפןchrome מאפשר כלים עבור מפתחים, שם יש אפשרות לעקוב אחרי התוכנית, לשים נקודות עצירה(break points) ולהריץ את התוכנית צעד אחר צעד.

בנוסף לכך,לצורך הכתיבה וההרצה השתמשתי באקליפס, התקנתי תוסף לגאבה סקריפט באקליפס.דבר שאפשר לי לנווט התוכנית בצורה נעימה ונוחה.

ניהול הקבצים-

כדי לנהל את הקבצים והשינויים, וכדי שיהיה לי האפשרות לחזור לאחורה, לראות את השינויים שעשיתי לאורך זמן, השתמשתי בתוכנת ניהול קבצים git.

ואת הקבצים עצמם שמרתי בסרבר פתוח ברשת שנקרא github

ובתור השרת השתמשתי בrawgit שיודע להיות שרת לקבצין שנשמרים ב github.

הערכת פתרון

התכנון המקורי שלי היה להשתמש באלגוריתם למציאת מסלול A\* אך החלטתי לממש אלגוריתם בעצמי, כדי להיות מקורית ולאתגר את היכולות שלי.

שנית, התכנון שלי היה לעשות את הפרויקט java אך בתחילת הפרויקט הבנתי שעדיף לעשות אותו ב java script כי הוא מאפשר ליצור מערך דינמי וכך יותר קל להאריך את הנחש. בנוסף לכך רציתי להרחיב את אופקי וללמוד שפה חדשה שאני לא לומדת בבית ספר.

תיאור ממשק למשתמש



במשחק שני ממשקים עיקרים :

-הראשון בו שולט השחקן (מפעיל אנושי) בעזרת מקשי המקלדת על נחש ארוך ודק המשוטט באזור מבודד, אוסף אוכל, ומנסה להימנע מלהתנגש בזנב שלו או בקירות המקיפים את אזור המשחק. הנחש מתארך כשהוא אוכל, וכך קשה יותר להימנע מלהיתקע בזנב או בגוף עצמו. השחקן שולט על כיוון תנועת ראש הנחש (למעלה, ימינה, למטה או שמאלה) בעזרת מקשי במקלדת או מסך מגע בטלפון, ויתר גופו של הנחש עוקב אחרי ראשו. הנחש נמצא בתנועה מתמדת ולא ניתן לעצור אותו במהלך המשחק.

בחלק האוטומטי אותם כללי משחק אלא שהתוכנה היא זו שתשחק ותנסה להגיע אל המטרה.

תפעול:

במחשב- החצים במקלדת מכתיבים את כיוון התנועה, לחיצה על החץ למעלה הנחש יזוז למעלה וכו..

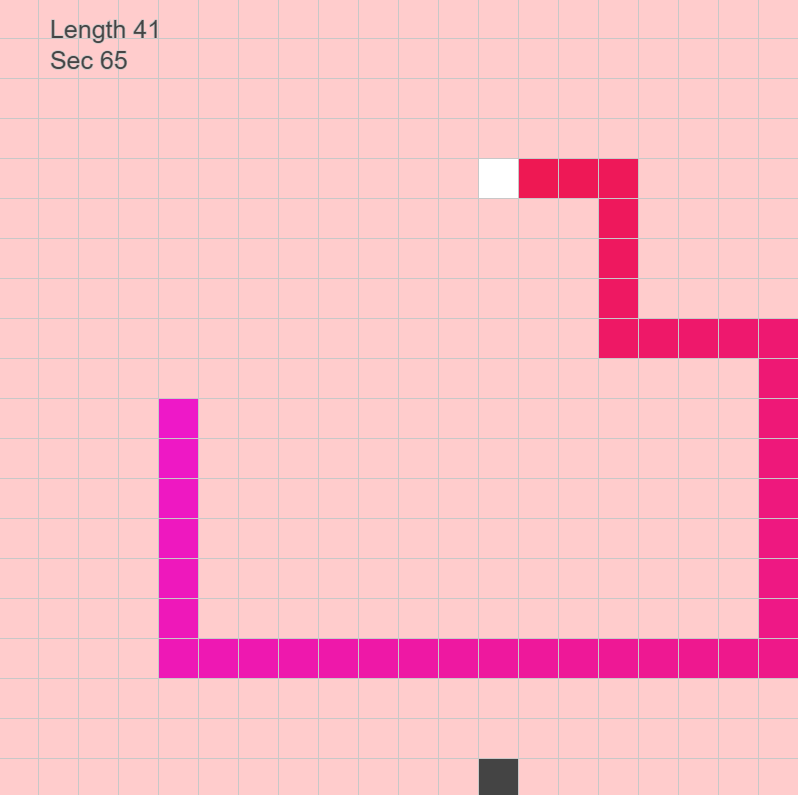
בטלפון- נגיעה בחלק הימני של המסך יזיז את הנחש ימינה נגיעה בחלק העליון של המסך יזיז את הנחש למעלה וכן הלאה.

ניתן לעבור בין המצבים באמצעות המקש במקלדת A או M

1. מצב אוטומטי

M- מצב ידני

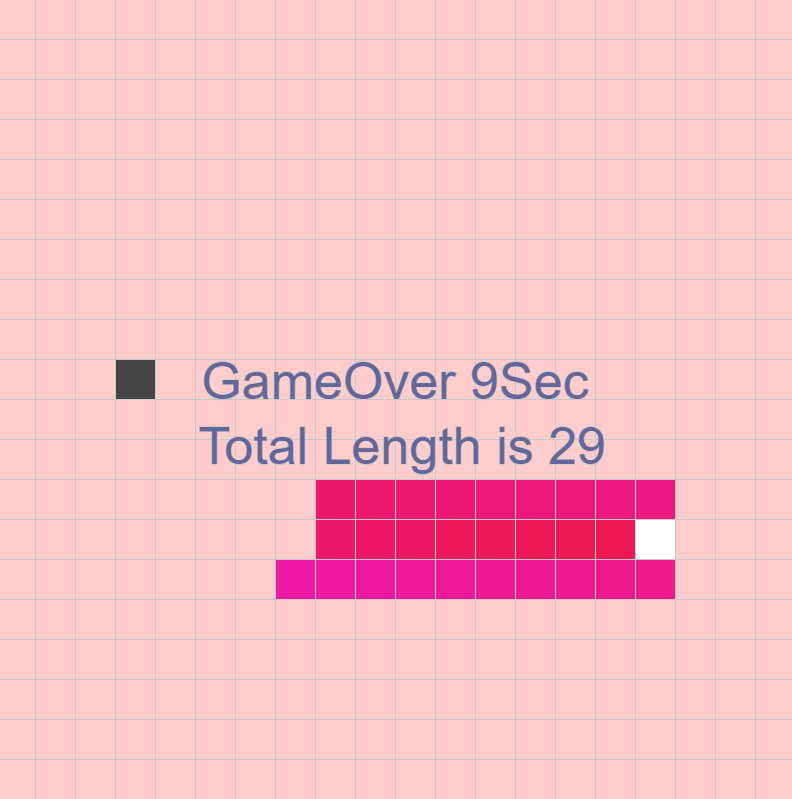
ניתן לשנות את מהירות הנחש באמצעות המקשין +,-

בטלפון, ניתן לעבור למצב אוטומטי באמצעות החלקת האצבע על המסך.





בסיום המשחק יופיע מסך הכולל נתונים לגבי אורך הנחש והזמן הכולל של המשחק:



כדי לצאת ממצב זה ולהתחיל משחק חדש, ניתן להקיש על כל מקש במקלדת.

רפלקציה

הפרויקט הציב בפני אתגרים ומשימות בהם לא התנסיתי בעבר. אפליקציה זו הינה הראשונה שפיתחתי. החלטתי כרעיון לאפליקציה לבחור דבר שבעיניי מהנה והרבה אנשים יכולים להשתמש בו ולהנות ממנו. התמודדתי באופן עצמאי.

האתגר הראשון היה לימוד השפה גאווה סקריפט באופן מקיף. לימוד זה הצריך זמן רב ומשמעת עצמית. בזכות כך, העמקתי את הידע שלי באופן משמעותי בתחום. יתר על כן, למדתי איך לנהל את פרויקט מבחינת ניהול קבצים.

בשנה זו, נטלתי על עצמי משימות רבות על מנת לקדם את לימודי בעיקר בתחום המחשבים. במקביל לפיתוח הפרויקט, למדתי באינטנסיביות הן בבית הספר במגמות שבחרתי יחד עם מתמטיקה ומחשבים.

לסיום, פרויקט זה העמיד אותי בפני אתגרים חדשים וקשים. למדתי להעמיד לעצמי רף גבוה אליו אני שואפת להגיע. ההתנסות בפיתוח התוכנה, מציאת כלים מתאימים לפיתוח(כמו תוסף java script לאקליפס (, ללמוד לעבוד עם כלים עבור מפתחים של דפדפן ה chrome , כלים לניהול קבצים, הפעלת שרתים ועוד, תרמה לי רבות והעניקה לי ניסיון, נהנתי רבות מפיתוח הפרויקט, שפתח לי עולמות חדשים ובזכותו הבנתי איך כל החלקים שאנו לומדים מתחברים, וניתן ליצור בעזרתם משהו שימושי.

ביבליוגרפיה

1. Github <https://github.com/>
2. <https://p5js.org> p5.js/
3. לימוד java script ותכנות האופן רחב <https://www.youtube.com/user/shiffman>
4. לימוד java script <https://www.codecademy.com/en/tracks/teachyourself>
5. java script לאקליפס https://www.eclipse.org/downloads/packages/eclipse-ide-javascript-web-developers/indigosr2