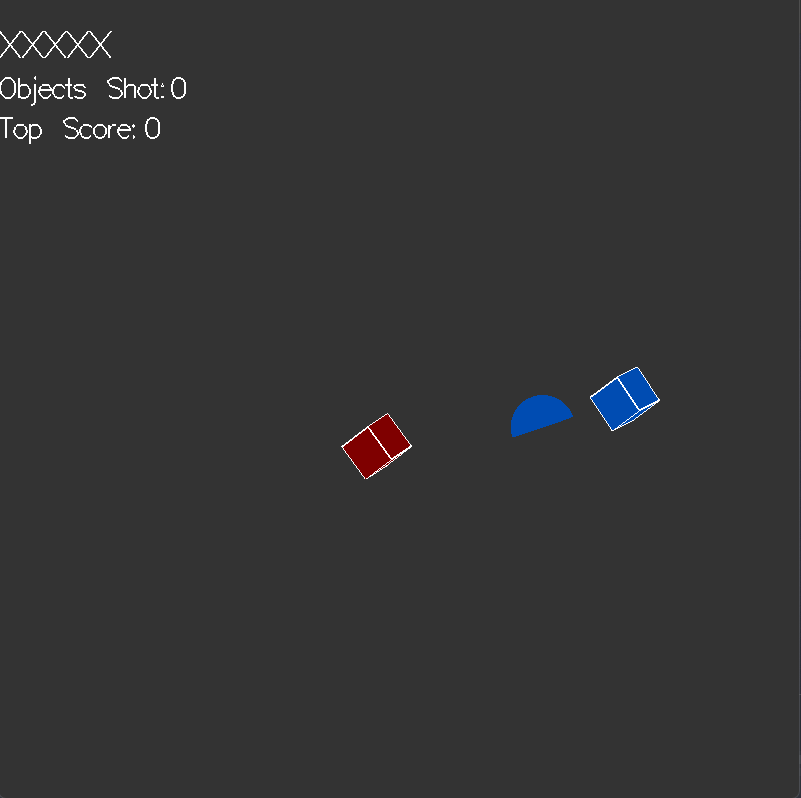
**בית הספר התיכון המקיף ו' ע"ש קוהל**

**פרויקט גמר**

**בגרפיקה ממוחשבת**

**נושא:**

משחקיירוט אובייקטים



**היקף העבודה: 5 יחידות לימוד**

**שם החלופה: הנדסת תוכנה 883589**

**מגיש: אלון שולדינר**

**ת"ז: 327646287**

**תאריך ההגשה: 22.6.2023**

**מנחה: יפים קיזנר**

תוכן עניינים

[מבוא 3](#_Toc136352534)

[הרקע לפרויקט 3](#_Toc136352535)

[תיאור הפרויקט 3](#_Toc136352536)

[אתגרים מרכזיים 3](#_Toc136352537)

[הסיבות לבחירת הפרויקט 3](#_Toc136352538)

[תהליך המחקר 4](#_Toc136352539)

[שלב 1 – תכנון 4](#_Toc136352540)

[שלב 2 – חקירה 4](#_Toc136352541)

[שלב 3 – פיתוח 4](#_Toc136352542)

[שלב 4 – מעקב אחרי פעולת המערכת 4](#_Toc136352543)

[מדריך למשתמש 5](#_Toc136352544)

[בסיס הנתונים 6](#_Toc136352545)

[מבוא ל- OpenGL 7](#_Toc136352546)

[מערכות צירים 8](#_Toc136352555)

[עקרונות פיזיקליים 10](#_Toc136352556)

[מטריצות 11](#_Toc136352557)

[סיבוב דו- ממדי(Rotation 2D) 11](#_Toc136352558)

[סיבוב תלת-ממדית(Rotation 3D) 12](#_Toc136352559)

[הזזה דו-ממדית(Translation 2D) 13](#_Toc136352560)

[סילום (Scaling) 14](#_Toc136352562)

[גורמי סילום 14](#_Toc136352564)

[חשבון וקטורים ומטריצות 15](#_Toc136352566)

[פעולות חשבון בוקטורים ומטריצות 16](#_Toc136352567)

[רפלקציה 18](#_Toc136352568)

[ביבליוגרפיה 19](#_Toc136352569)

מבוא

הרקע לפרויקט

המשחק Fruit Ninja הינו משחק ארקייד ופעולה מצליח שפותח על ידי החברהHalfbrick Studios. המשחק הושק ב-21 באפריל 2010 בחנות האפליקציות של אפל (Appstore) לאייפון ולאייפוד טאץ' בהמשך, עקב ההצלחה הגדולה של משחק שוחררו גרסאות נוספות לאייפד ב-12 ביולי 2010, למכשירים מבוססי אנדרואיד ב-17 בספטמבר 2010 בנוסף המשחק שוחרר גם למערכת ההפעלה Windows Phone 7 בהמשך גם למערכת ההפעלה Bada של סמסונג ולמערכת ההפעלה Symbian.

במשחק השחקן יכול לבחור באפשרויות של כמה משחקים - לאחר 3 פסילות הוא נפסל, משחק לדקה, משחק לדקה וחצי ומשחק נגד שחקן שונה. מטרת המשחק היא לחתוך פירות, לדוגמה: אננס, תפוח, לימון, בננה, אבטיח, מלון וכו'. השחקן צריך לצבור כמה שיותר נקודות וכשהוא שובר שיאים הוא גם קונה כל מיני דברים בעזרת סטאר שניתן להשיג רגיל (בכל משחק) למשל: רקעים חדשים חרבות חדשות וכו', המשחק בנוי על סכינים שבעזרתם חותכים פירות ומגיעים לניקוד הגבוה ביותר.

<https://www.youtube.com/watch?v=COFeh96bfWI&t=300s>



תיאור הפרויקט

בפרויקט יצרתי משחק הדומה לFruit Ninja, במקום פירות השתמשתי במספר אובייקטים גאומטריים שונים בצבע כחול ובמקום פצצות השתמשתי באובייקטים בצבע אדום. במשחק מטרת השחקן היא להשיג את הניקוד הגבוה ביותר עד אשר יפסל. בנוסף ניתן לראות את התוצאה הגבוהה ביותר אשר הושגה.

אתגרים מרכזיים

האתגר המרכזי שלי היה למידת הספרייה OpenGL מכיוון שאת שפת C++ כבר ידעתי.

הסיבות לבחירת הפרויקט

כשהייתי ילד אהבתי לשחק במשחק זה.

תהליך המחקר

שלב 1 – תכנון

בשלב התכנון רשמתי

* חוקים: על המשתמש לא לתת לאובייקטים הכחולים לצאת מגבולות המסך בכך שהוא ילחץ עליהם, כל אובייקט כחול אשר התפספס מהווה פסילה. כל אובייקט אדום אשר ילחץ יהווה פסילה. במידה והשחקן הגיע ל3 פסילות המשחק נגמר.
* מנגנון פיזיקלי עליו מתבססת תנועת האובייקטים במשחק בו מתקיימים כל חוקי ניוטון
* יצירת UI המכיל מידע לגבי המשחק

שלב 2 – חקירה

העמקתי בלימודי ספריית OpenGL, למדתי כיצד להשתמש בה באופן בסיסי כמו יצירת חלונית, הגדרת צירים, ציור אובייקטים תלת ממדיים ועוד.

שלב 3 – פיתוח

התחלתי את פיתוח הפרויקט ביצירת המנגנון הפיזיקלי לתנועת האובייקטים תוך שימוש בחוקי ניוטון.

לשם כך יצרתי מחלקה בסיסית של ווקטור תלת ממדי בשם **vector3** אשר ניתן לבצע בעזרתו פעולות מתמטיות כמו כפל עם סקלר, חילוק עם סקלר, חיבור וקטורים, חיסור וקטורים, השוואת וקטורים ועוד.

יצרתי מחלקה בשם **Game** אשר מכילה את וקטור כיוון כוח הכובד ומערך המכיל את כל האובייקטים אשר ינועו במהלך המשחק ופונקציה אשר מחשבת את המיקום של כל אובייקט בכל פסק זמן. בנוסף לכך הוספתי משתנים ופונקציות הדרושות למטרת המשחק כמו משתנים המכילים את כמות הפסילות, כמות הפגיעות וכמות הפגיעות המקסימלית ופונקציות אשר מעדכנות את משתנים אלה.

לאחר יצירת המחלקה בניתי מחלקה אבסטרקטית **Object**אשר מכילה את התכונות שכל אובייקט צריך במשחק: מיקום האובייקט במרחב, וקטור מהירות, וקטור הכוח השקול על האובייקט, מסת האובייקט, צבע האובייקט, דגל המסמן האם האובייקט נורה ופונקציה וירטואלית של ציור האובייקט על המסך.

כעט בחרתי במספר צורות בהן אני אשתמש בפרויקט והחלטתי על תיבה וחצי ספרה, **HalfSphere ,Rectangle**, כל אובייקט השתמש באובייקט האבסטרקטי ולכל אובייקט יש את פונקציית הציור שלו ומשתנים ופונקציות עזר משלו.

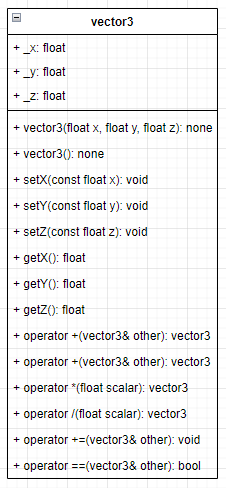
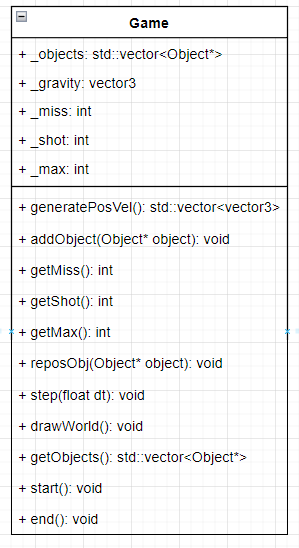
שלב 4 – מעקב אחרי פעולת המערכת

הבעיה העיקרית שבה נתקלתי היא בחירת המהירות ההתחלתי לאובייקטים בשביל ליצור תנועה של זריקה משופעת כך שכל טווח התנועה יכנס למסך

מדריך למשתמש

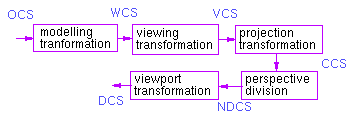
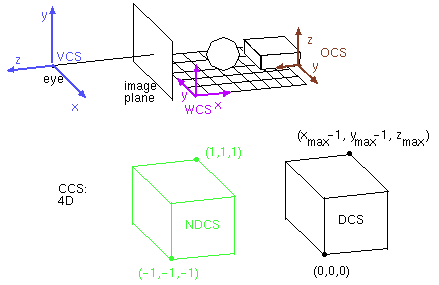
עם הפעלת הפרויקט המשחק יתחיל, בצד שמאל של המסך ניתן לראות את כמות הפסילות (איקסים אדומים), כמות האובייקטים שנורו בסיבוב הזה ואת הכמות המקסימלי שהשחקן הצליח להשיג בסיבוב אחד. על השחקן ללחוץ על כל האובייקטים הכחולים לפני שהם מספיקים לצאת מחוץ למסך אחרת יצבור פסילה. בנוסף אם השחקן ירה באובייקט אדום הוא יצבור פסילה. כשהשחקן יצבור 5 פסילות יפסל ויגמר הסיבוב, בהתאם לכמות האובייקטים שירה השחקן יתעדכן המספר המקסימלי שהשחקן הצליח להשיג בסיבוב אחד. בלחיצה על המקש R, ניתן לעשות להתחיל סיבוב חדש. בלחיצה על המקש Q ניתן לסגור את התוכנית.

בסיס הנתונים



מבוא ל- OpenGL

Coordinate Systems in the Graphics Pipeline



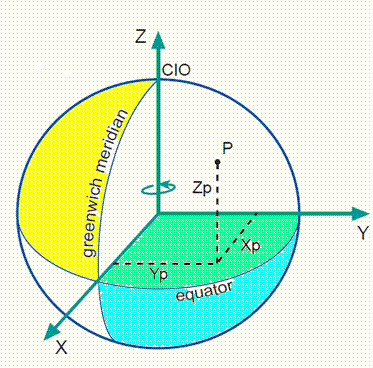
OCS - object coordinate system    
WCS - world coordinate system    
VCS - viewing coordinate system    
CCS - clipping coordinate system    
NDCS - normalized device coordinate system    
DCS - device coordinate system

**OpenGL functions for setting up transformations**

|  |  |
| --- | --- |
| modelling transformation *modelview matrix* | glTranslatef()   glRotatef()   glScalef() |
| viewing transformation  *modelview matrix* | gluLookAt() |
| projection transformation   *projection matrix* | glFrustum()   gluPerspective()   glOrtho()   gluOrtho2D() |
| viewing transformation | glViewport() |

מערכות צירים

ב- OpenGl משתמשים ב-4 סוגי מערכות צירים: ימנית, שמאלית, חלוני וגישתי.



Y

X

Z

Z

X

**ימנית שמאלית חלונית(מסך) גישתית**

X

X

Y

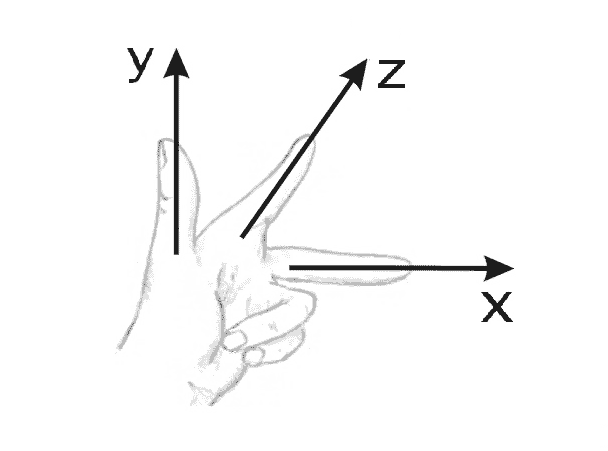
Y

Y

מערכת צירים ימנית

( World (problem) coordinate system)

מיועדת ליצירת ותנועת עצמים (כולל מצלמה) במרחב.



מערכת צירים שמאלית - מיועדת להגדרת הראיה של בן אדם או המצלמה בעזרת הפונקציות :

glOrtho(), glPerspective() .

מערכת צירים של מסך - מיועדת לתצוגת עולם תלת מימדי בדו מימדי.

מערכת צירים גישתית -מיועדת ליצירת קשר בין העכבר והעצמים שנצאים במערכת צירים של מסך ומוגדרת בעזרת הפונקציה :

glutInitWindowSize(Width in pixels , Height in pixels)

למשל לפי ההגדרות:

מסך

glutInitWindowSize(800,800);

חלון

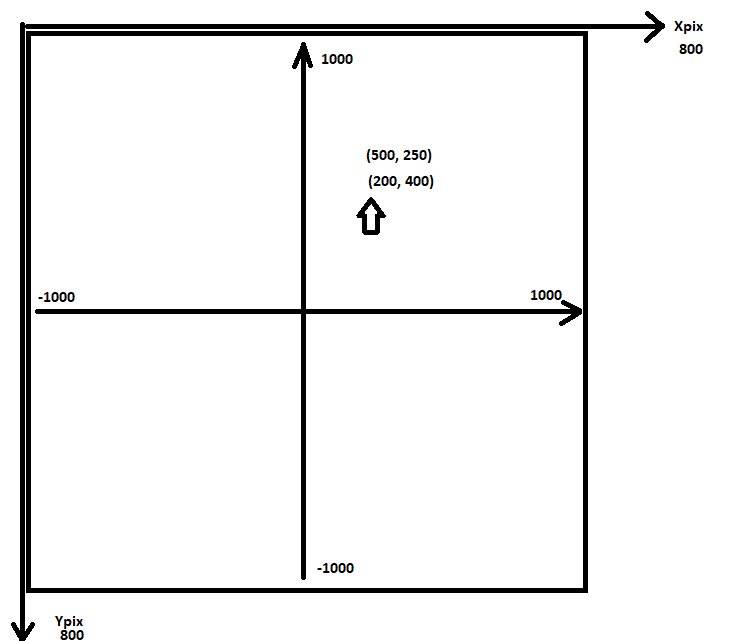
glOrtho(-1000, 1000, -1000, 1000, -1000, 1000);

גדלי החלון הם: 800x800

במערכת גישתי הרוחב הוא pixels800 לפי ציר ה-X והגובה הוא 800 pixels לפי ציר ה- .Y

במערכת חלונית הרוחב הוא 2000 לפי ציר ה-X והגובה הוא 2000 לפי ציר ה-Y.

המעבר בין המערכות נעשה ע"י חישוב הבא :



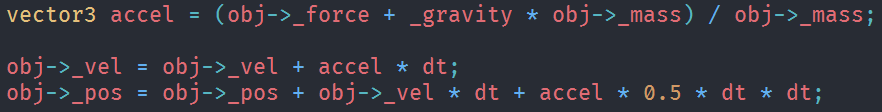
xm = -1000 + x \* 1000 / WIDTH \* 2;

ym = 1000 - y \* 1000 / HIEGHT \* 2;

WIDTH= 800

HIEGHT = 800

עקרונות פיזיקליים



מטריצות

תנועת העצמים במרחב נעשית ע"י חישוב מטריצות.

ב-OpenGL ניתן להגדיר 3 קבוצות מטריצות בגודל 4\*4 כל אחת, בעזרת פונקציה:

void glMatrixMode(enum mode)

GL\_PROJECTION GL\_MODELVIEW GL\_TEXTURE

קבוצת המטריצות GL\_MODELVIEW כוללת פעולות ליישום:

א. סיבוב (Rotate)

ב. תזוזה (Translate)

ג. סילום (Scale)

ד. וכו'

Y

X

y’

P’

y

P

x’

x

O

α

β

סיבוב דו- ממדי(Rotation 2D)

נסמן:

רדיוס 

זוויות 

מטריצה סיבוב סביב ציר Z היא 

למשל:

נקודה  לאחר פקודה glRotatef(30,0,0,1); תקבל מקום חדש:



# סיבוב תלת-ממדית(Rotation 3D)

XXXX

ZXXX

YXXX

A'

AXXX

β

מטריצה סיבוב סביב ציר Y היא 

מטריצה סיבוב סביב ציר X היא 

למשל :נקודה A בעלת שערים (5,2,3) לאחר פקודה glRotatef(30,0,0,1); תקבל מקום חדש:







הזזה דו-ממדית(Translation 2D)

P

P’

X

YT

### מטריצה תזוזה היא

` 

סילום (Scaling)

### 

AXXX

A'

XXXX

YXXX

xXXX

x'XXX

yXXX

y'XXX

### גורמי סילום

### ( scaling factors)



מטריצה סילום **ה**יא 

****

XXXX

ZXXX

YXXX

***סילום של הגוף*** *,*

*כאשר מרכז הכובד שלו מתלכד עם מרכז מערכת הצירים לא גורם ל"תנועת" הגוף.*

חשבון וקטורים ומטריצות

בשלב הראשון נתייחס לוקטורים ולמטריצות כאל דרכים נוחות המאפשרות לייצג סטים של מספרים, ונגדיר את ההגדרות היסודיות של פעולות החשבון המתבצעות באמצעותם.

נשתמש כעת בהגדרות הבאות:

1. *סקלר (*Scalar*)* - מספר יחיד כגון: s = 47½
2. *וקטור(*Vector*)* - מערך מספרים עם אינדקס יחיד כגון:  כאשר במקום השלישי  נאמר כי הרכיב השלישי של  שווה ל-7. במתמטיקה( לפי תכנית הלימודים של ביטת הספר) משתמשים רק בוקטורים תלת מימדיים כגון: 

* *מטריצה (* Matrix*)* - מערך מספרים עם שני אינדקסים כגון:

 נאמר כי : 

האינדקס הראשון מייצג שורה. האינדקס השני מייצג עמודה.

בדרך כלל נסמן ב- את מספר השורות, וב- את מספר העמודות.

(ניתן כמובן לבנות מערכות עם יותר אינדקסים כמו מערכות תלת מימדיות).

פעולות חשבון בוקטורים ומטריצות

כפל מטריצות

נגדיר תחילה *מכפלה סקלרית* של שני וקטורים:

נניח כי  ו- הנם וקטורים בעלי n רכיבים כל אחד, לכן:



תוצאת המכפלה הסקלרית היא סקלר (ולא וקטור) ומכאן שמה (מכפלה סקלרית נקראת גם מכפלה פנימית).

דוגמא: .

מכפלת המטריצות  מוגדרת באמצעות המכפלות הסקלריות של השורות  בעמודות  באופן הבא:



k רץ על הרכיבים k רץ על הרכיבים

בעמודה ה-j של בשורה ה-i של

נסמן לשם הנוחות:  , 





כאשר: C = А•В למשל ,



ראש הטופס

תחתית הטופס

הערות:

1. המכפלה מוגדרת רק אם מספר העמודות של שווה למספר השורות של.
2. לא מתקיים חוק החילוף - כלומר, . השוויון יתכן רק כאשר ו- ריבועיות (מספר השורות והעמודות זהה) וגם אז השוויון לא בהכרח מתרחש.  
   אם מתקיים  נאמר כי ו- מתחלפות.

מתקיימת אסוציאטיביות: 

רפלקציה

תהליך הלמידה עבור הפרויקט היה מעניין, בהתחלה ספריית OpenGL הרגישה כמו ספרייה נורא קשה שתיקח הרבה זמן ללמוד אך כשהתחלתי לעשות קטעי קוד קצרים וניסוי וטעיה על מנת להבין את הספרייה מאוד מהר הצלחתי להשתלט עלייה.  
בתהליך יצירת הפרויקט עצמו חביתי כמה קשיים אך בעזרת השקעת זמן הצלחתי לפתור את כל הבעיות שהיו לי.

אני מאוד שמח עם התוצר הסופי שלי וגאה בזה שכל הפרויקט שלי נכתב מאפס על ידי.

ביבליוגרפיה

[1] <https://en.wikipedia.org/wiki/Fruit_Ninja>

[2] <http://www.opengl.org/resources/code/samples/redbook>

[3] <http://www.glprogramming.com/red/>

[4] <http://www.cs.uccs.edu/~semwal/indexGLTutorial.html>

[5] Interactive Computer Graphics – A top-down approach with OpenGL (First Edition)

מאת: Professor Edward S. Angel

אוניברסיטת ניו מקסיקו, ארה"ב.