

## שם הפרוייקט: Combat Tanks



תאריך ביצוע: 07/2021

קורס: יישומי מחשב להנדסת אלקטרוניקה

מרצה: ד"ר אלעזר פלקסר

מגישים:

עומר אביב - 205785876

רום הירש – 313288763

## תוכן עניינים:

2	1.הקדמה
3	2.נושאים שמימשנו בפרויקט
4	3.הסבר קצר על מימוש המשחק :
7	4.פונקציות
9	5.פאנלים
19	6.הוראות שימוש
20	7.קבצים בפרויקט
21	8.בעיות עיקריות במהלך הפרוייקט :
22	9.סיכום
23	10.נספח קוד :
53	11.נספח ורטואליזציה לתקשורת COM

# 1.הקדמה

## 1.1. מטרת הפרויקט:

מטרת הפרויקט הוא להראות את היכולות שנלמדו בקורס ויכולות נוספות שלמדנו בלמידה עצמית בעזרת האינטרנט וממשק העזרה של CVI, מימשנו את יכולות אלו ע"י יצירת משחק הכולל קרב טנקים במולטי פלייר בעזרת תקשורת RS232.

תיאור קצר של המשחק :

על השחקן לנצח בקרב את יריבו על ידי פגיעה בשחקן השני בעזרת יריות.

כמו כן באפשרות השחקן לצבור לבבות, מהירות – משפר את מהירות השחקן לזמן מוגבל, יריות ודברים נוספות.

## 2. נושאים שמימשנו בפרויקט

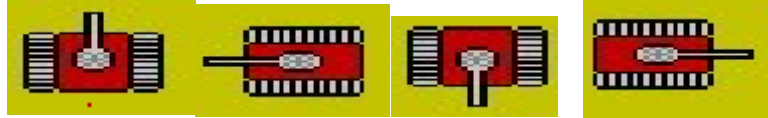
בפרויקט מימשנו את הנושאים הבאים:

1. RS232 – תקשורת בין שני המחשבים שמשחקים את המשחק.
2. עיבוד תמונה - שינוי זווית של התמונות במשחק.
3. כתיבה לקובץ - שמירת נתונים של המשחק.
4. Menu Bar - תתי ברים של הפאנל
5. Tabs – חלוקת פאנל בTABS
6. Threading - הרצת פונקציות במקביל
7. Canvas – תצוגת כל המשחק
8. Multi Panel - שימוש בכמה פאנלים
9. Tables - טבלאות נתונים ונצחונות
10. DLL – פונקציות עזר
11. Active X (לימוד עצמי) (WINDOWSMEDIAPLAYER) – צלילים של המשחק

### 3. הסבר קצר על מימוש המשחק :

1. הטנקים תזוזה ירי:

- a. הטנק עצמו הוא תמונה שנטענת ל bitmap .
- b. על מנת לסובב את הטנק לכל כיוון השתמשנו ב **בעיבוד תמונה** כלומר נבצע Rotate לתמונה ע"פ החץ שלחצנו במיקלדת .



- a. את התנועה של הטנק מבוצע ע"י שינוי המיקום התמונה ב canvas ניתן לשלוט בזה בקלות ע"י האובייקט שהוגדר לכל טנק כאשר נשנה את המיקום או לפי קלט מהמקלדת של המשתמש או לפי קליטת תקשורת מהמחשב השני(הזזת טנק השני).
  - b. ירי מבוצע ע"י עיגול המדמה ירי כדור(שמצויר ב canvas) שזז בגבולות המפה לפי כיוון הטנק מיקום הכדור שנורה זז ע"י פונקציה שרצה ב thread ומעדכנת את המיקום של הכדור ובמקביל רץ תמיד ה thread שמצייר את כל האובייקטים הקיימים לפי מיקומו של האובייקט.
  - c. הטנק הנבחר לפי player number זז במסך לפי קלט מהמקלדת והטנק שני לפי פקודות שמקבל מאיך שהמשחק שני הזיז אותו בתקשורת RS232.
  - d. פונקציות היעודיות לאתמול האובייקטים ובדיקות שונות ממשות בעזרת DLL
2. Affects - אפקטים :



- a. כאשר יש פגיעה של כדור בטנק נצייר שריפה על גבי הטנק ונשמיע מוזיקה של פגיעה.
- b. כאשר נורה כדור ונלקח element יש צלילים בהתאם.
- c. בנוסף ישנם צלילים רבים ששולבו בפעולות שונות במשחק.
- \* כל הצלילים ממומשים בעזרת activeX
- d. הרצנו אלמנטים המשפרים או מוסיפים דברים לטנק יריות מהירות וחיים.

3. ציור על ה canvas -

- a. כל התמונות נטענות ל bitmap ולאחר מכן מצוירות למסך.
- b. הציור על canvas מבוצע בעזרת thread שרץ כל עוד המשחק במצב פעיל ומצייר את המצב הנתון של המשחק כאשר פונקציות אחרות משנות את מיקום המאובייקטים .

4. מימוש מולטי פלייר בעזרת RS232 :

- a. כחלק ממימוש המולטי פלייר נדרש מאיתנו שהתוכנית תהיה אפשרית להרצה גם כשחקן מספר 1 וגם שחקן מספר 2 לכן הטנק שאתה שולט בוא נבחר וכל הפעולות עליו מבוצעת בעזרת מצביע לטנק הנבחר כנל לגבי הטנק שני.
- b. התקשורת בין שחקן 1 לשחקן 2 ממומש בעזרת RS232 על מנת לממש זאת בנינו "פרוטוקול" כך שכל מחשב ידבר עם ויבין מה השחקן השני עשה ובכך לעדכן את ה canvas בהתאם ואת המחשב שני

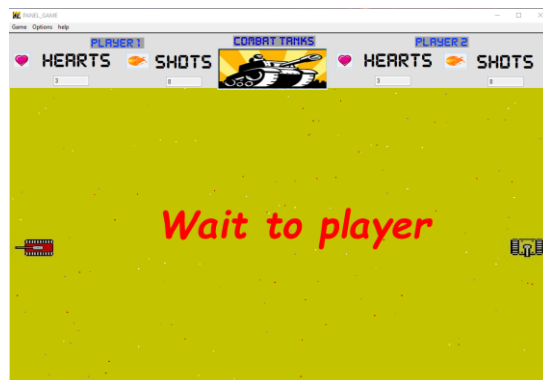
להלן הפרוטוקול והפועולות שנשלחות בין שני המחשבים :

command	operation
startGame1	נשלח על מנת להגיד לשחקן 2 להתחיל את המשחק
startGame2	נשלח על מנת להגיד לשחקן 1 להתחיל את המשחק
left	move tank left
right	move tank right
up	move tank up
down	move tank down
shot	shot
addspeed	add element speed + increase the tank speed
addhearts	add hearts
addshot	add shot

\*במידה ואחד המחשבים שולח את אחת הפקודות השני יבצע את הפעולה בטנק השני שהוא לא שולט בוא.

בתחילת המשחק יש בדיקה האם שני השחקים נכנסו ומידה וכן המשחק יתחיל ביחד בשני המחשבים. ( מבוצא תיאום בין המחשבים מתי להתחיל את המשחק)

תמונה – מסך המתנה לשחקן השני



5. אובייקטים :

יצרנו structure לאובייקטים במשחק כגון לטנק, elements, שניתן לאסוף ולשפר יכולות וכו.. הכולל טריגר לפעולות ומידע שנדרש לדעת לגבי האובייקט.

```
}typedef struct DLLIMPORT object
```

bool exist; - האם הוא קיים במפה

int y; - מיקום בציר y

int x; - מיקום בציר x

int state; - כיוון שאליו הטנק נמצא (הקנה כלפי ימין/שמאל/למעלה או למטה)

bool rotate; - האם לסובב את תמונת הטנק

int hearts; - כמה חיים יש לטנק

int moveToState; - לאיזה מצב לסובב את ההאובייקט

int Fire; - האם לבמצע יריה – מתאפס לאחר הביצוע

int playerNum; - מספר השחקן ששולט בטנק

int Number\_shot; - כמות הכדורים

int Eelement\_bomb; - כמות האלמנטים שאסף האובייקט מסוג bomb

```

int Eelement_speed - כמות האלמנטים שאסף האובייקט מסוג speed
double speed - מהירות הטנק בפועל.
int hits - כמות הפגיעות שביצע הטנק בטנק השני
int Eelement_hearts - כמות האלמנטים שאסף האובייקט מסוג hearts
;OBJECT {

```

האובייקט כולל פעולות רבות שנדרש לשמור עליהם מידע כגון מיקומו והאם לירות עכשיו או האם לקח אלמנט שמשפר אותו וכו..

## 4. פונקציות

הסבר קצר על כל פונקציה:

function	מטרה
int CVICALLBACK GAME	קליטת קלט משתמש וביצוע פעולות בהתאם באובייקטים במסך
int load_image(int tank_num)	יצירת כל הפרמטרים הנדרשים להיפוך התמונה
int rotate_func ()	שינוי זווית של התמונה
int new_game	אתחול משחק חדש - קליטת כל התמונות (bitmap), הפעלת הפונקציות שרצות בthreading במהלך המשחק ואתחול משתנים למשחק.
int draw_game	הפונקציה האחראית להדפיס על CANVAS היא רצה בthread כל עוד המשחק בפעולה – זוהי פונקציה מרכזית בקוד כאשר נשנה משתנים גלובלים או ערכים של האובייקטים יודפסו דברים שונים בהתאם.
int updateTankspeed	עדכון מהירות הטנק ל10 שניות לאחר לקיחת אלמנט של מהירות
int RotateOBJECT	מזיז את הטנק לפי המיקום הנדרש הוא משתמש ב load image ו- rotate_func
int threadSound	משמיע קול לפי הקובץ שמועבר לו (מבוצע במקביל (threading))
int confStatics	עדכון הסטטיסטיקות על המשחק מבוצע בthread (בפנל של הסטטיסטיקות)
int displayWin	פעולות המבוצעות לאחר ניצחון של טנק (הצגה על המסך, הצגת לוח הסטטיסטיקות וכו.).
int displayLose	פעולות המבוצעות לאחר ניצחון של טנק (הצגה על המסך, הצגת לוח הסטטיסטיקות וכו.).
int create_hearts_element	יצירת אלמנטים מסוג חיים (מורץ במקביל (thread) לאורך כל משחק
int create_run_element	יצירת אלמנטים מסוג מהירות (מורץ במקביל (thread) לאורך כל משחק
int create_shot_element	יצירת אלמנטים מסוג יריה (מורץ במקביל (thread) לאורך כל משחק
int shot	הפונקציה אחראית על כל יצירת הכדור המדמה ירי כולל שינוי המיקום שלו מבוצע בthread – מבוצע גם בהתאם לכיוון הטנק.
int move_tank2	אחראי על תזוזת הטנק שני (שאתה לא שולט בו) הפונקציה מופעלת לפי פקודות שנשלחות מהמחשב השני
void CVICALLBACK NewGame	יצירת משחק חדש menu bar משתמש בint new_game
void MyPolling	הפונקציה האחראית על הקלט מהמחשב השני בRS232 וביצוע הפעולות בהתאם (קריאה לפונקציות אחרות) – הפרוטוקול שהגדרנו לתקשורת בין המחשבים ממומש בפונקציה הזו.
int Config	קינפוג ההגדרות לתקשורת RS232 כגון מס COM ומהירות .
int SendString	שליחת הודעה לשחקן השני שמחובר בRS232
CVICALLBACK RecTimer	טיימר הקובע מתי לבדוק האם יש הודעה חדשה
void CVICALLBACK BAR_RS232	פתיחת החלון של ההגדרות של RS232 מבוצע ע"י בר מהתפריט של המשחק.
int CVICALLBACK CB	סגירת החלון הראשי
int CVICALLBACK how_to_play_func	הצגת הפאנל של איך לשחק



int CVICALLBACK new_game_button	כניסה למשחק מהחלון הראשי
int CVICALLBACK QuitCallback	פוקצית יציאה מהחלון הראשי
void CVICALLBACK menu_bar_sound_flag	בר הכולל סימון האם להפעיל קול או לא
void CVICALLBACK Statics	הצגת חלון statics
void CVICALLBACK menu_bar_about	בר להצגת פרטים על מפתחי המשחק
void CVICALLBACK howtoplay_bar	הצגת איך לשחק מבר בפנאל של המשחק
int CVICALLBACK applycom	ישום ההגדרות החדשות פנאל ההגדרות של RS232

#### 4.1. קובץ-DLL

function	מטרה
check_hit_func	בדיקה האם יש פגיעה בין אלמנט לטנק
create_object	יצירת אובייקט במיקום רנדומלי
init_tank1	אתחול כל הערכים של האובייקטים של טנק 1
init_tank2	אתחול כל הערכים של האובייקטים של טנק 2
ReadWinsFromFile	קריאה מקובץ הנצחונות
WriteWinsToFile	כתיבה לקובץ הנצחונות
check_hit_func_tank	בדיקה האם יש פגיעה של כדור בטנק 1
check_hit_func_tank2	בדיקה האם יש פגיעה של כדור בטנק 2

## 5. פאנלים

המשחק מכיל את הפאנלים הבאים:

### 5.1. תפריט המשחק – Main Menu

#### התפקיד הראשי כולל :

1. כפתור New Game המכניס אותך למשחק
2. כפתור How to Play מעלה לך חלון המסביר כיצד לשחק.
3. COM – לקבוע את הCOM שאיתו אתה מחובר למשחק השני.

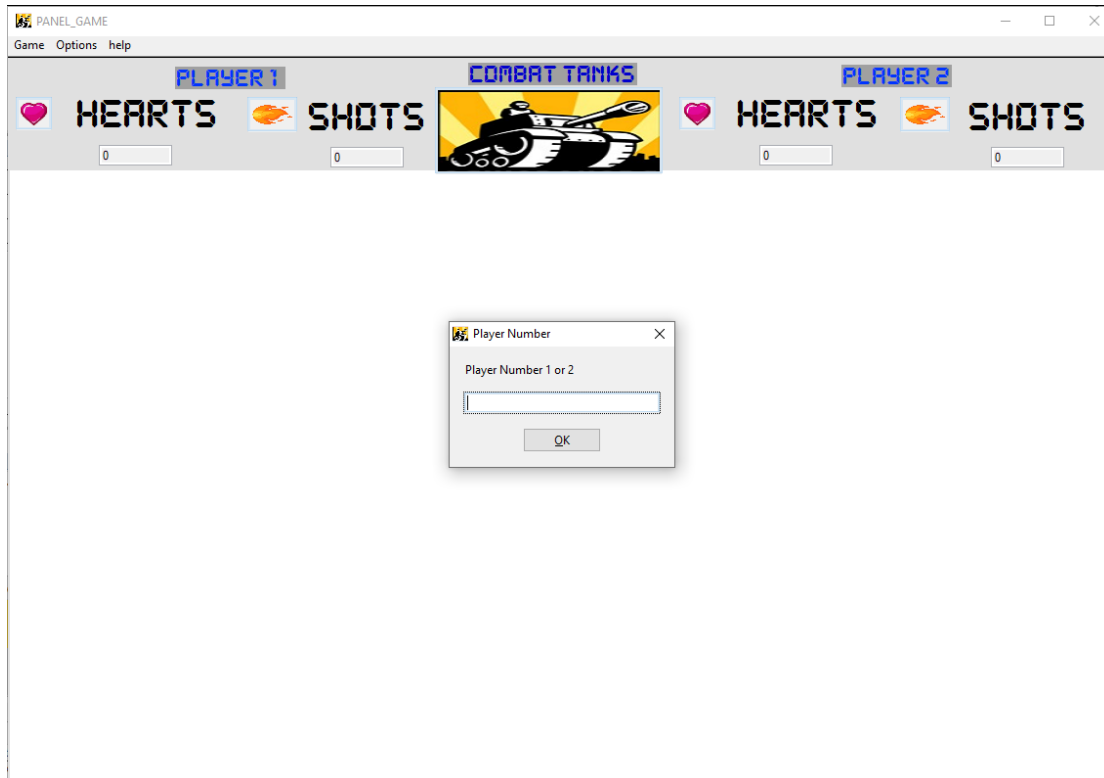
תמונה של תפריט ראשי:



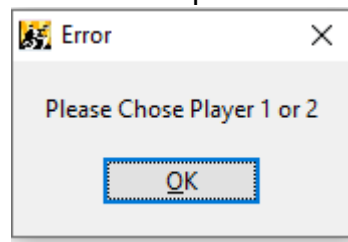
## 5.2. חלון המשחק – Game

a. מסך פתיחה – הזנת מספר שחקן (טנק 1 או טנק 2)  
ניתן להזין 1 או 2 במידה ותזין שגוי יקפוצ חלון שגיאה ותידרש להזין שנית.

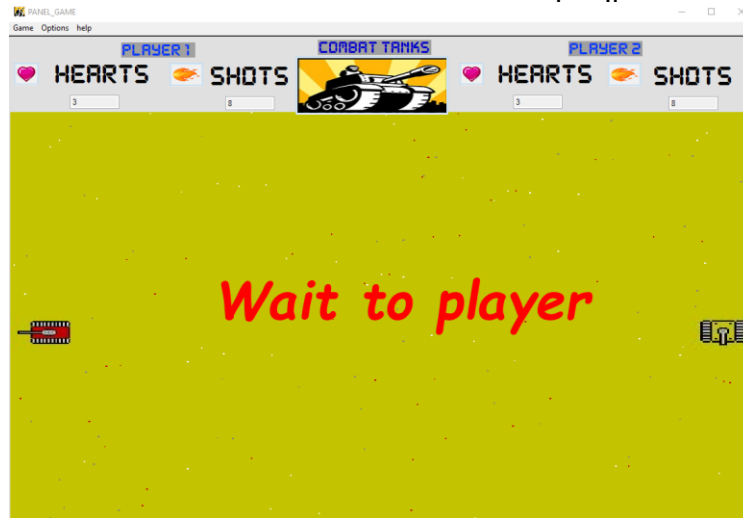
תמונה חלון בחירת שחקן:



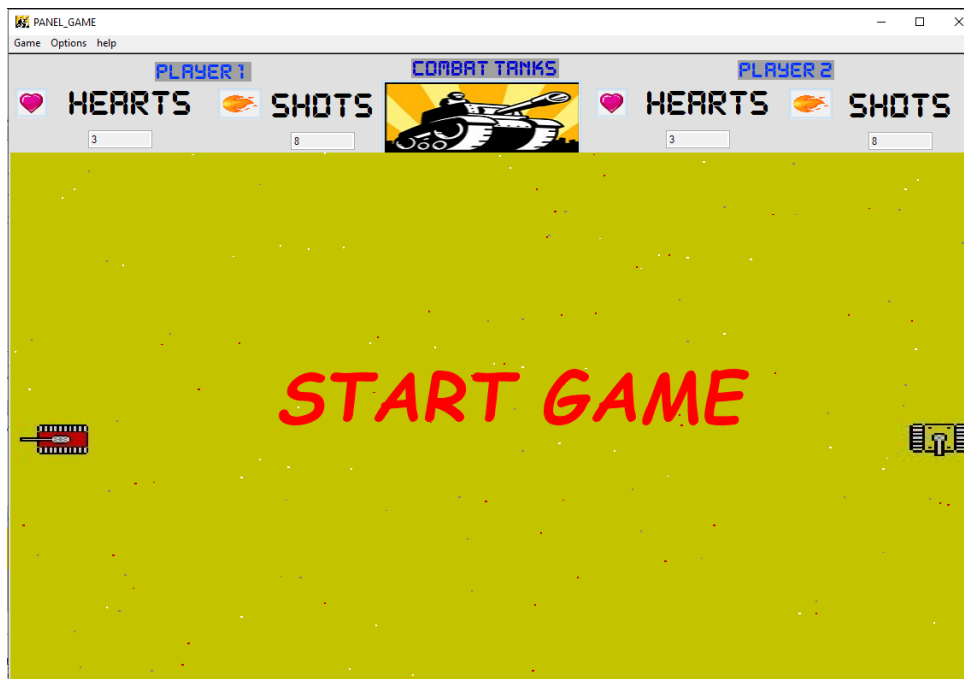
תמונה של חלון השגיאה :



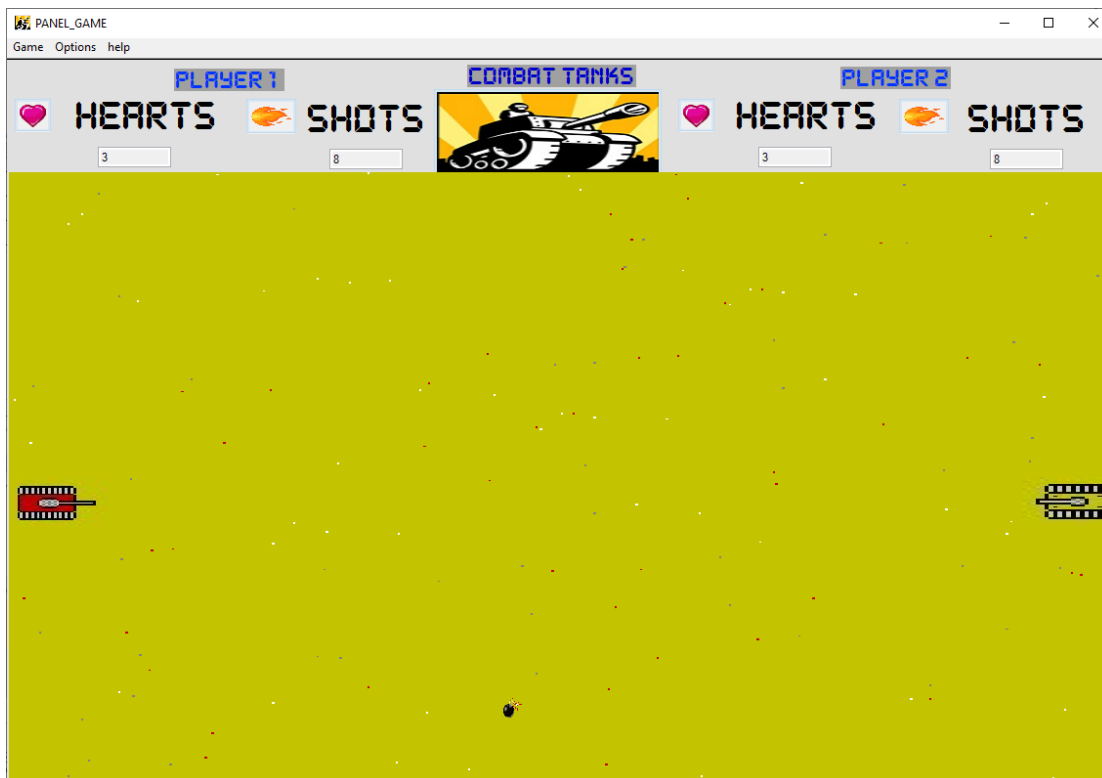
b. מחכה לשחקן נוסף



c. מסך תחילת משחק  
ניתן לראות את החיים והיריות שלך ושל היריב.  
תמונה מסך תחילת משחק:

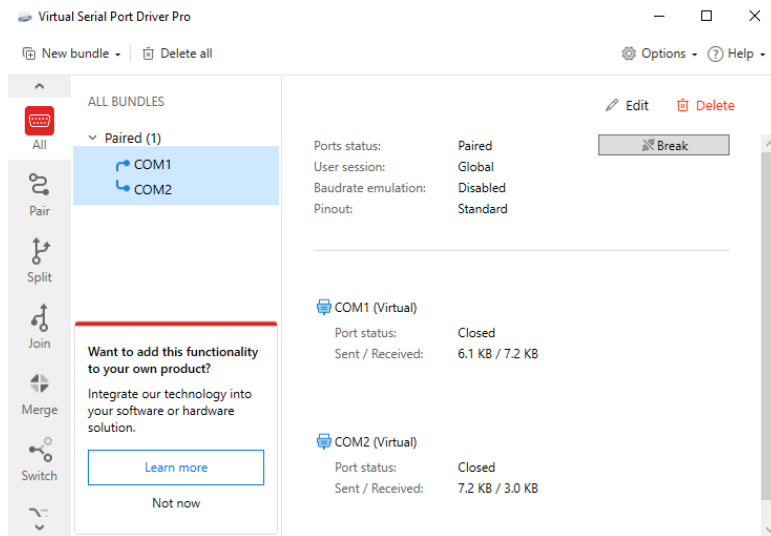


.d המשחק



מולטי פלייר מומש בעזרת RS232:

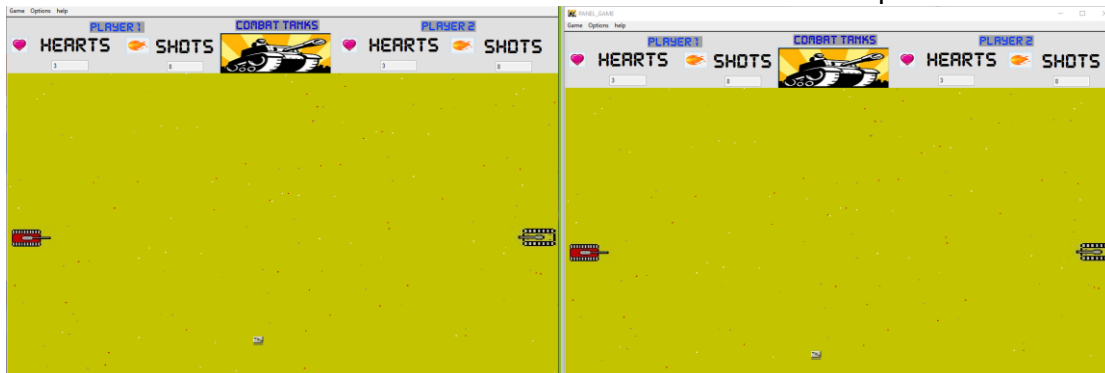
חיברנו שני com כך שאחד יהיה מחובר לשני וירטואלית :



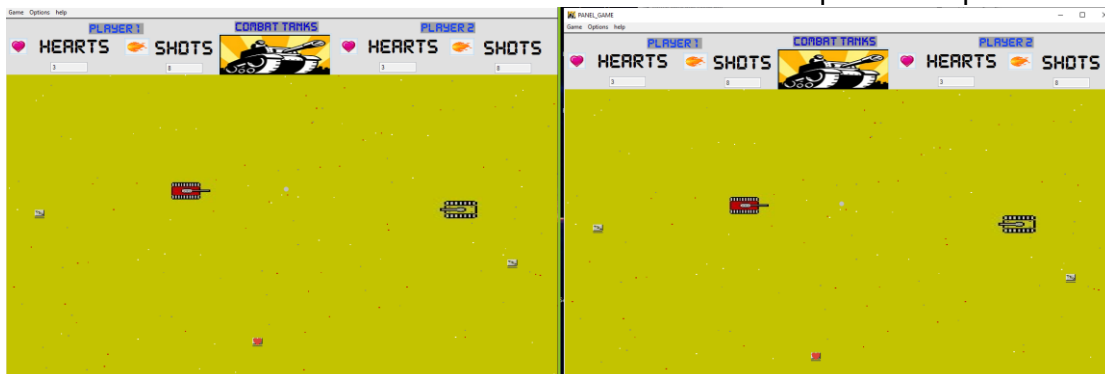
תמונה מולטי פלייר :

\*ניתן לראות שהזזנו טנק והוא זז בשני המחשבים

תמונת תחילת משחק:



תמונת משחק כאשר הטנקים זזו:



תמונת לאחר קרב (ניתן לראות שהחיים עודכנו שני המחשבים (החיים של טנק 1 ירדו ל1):



תמונת נצחון והפסד :

\*כולל sound מותאם לנצחון או הפסד.

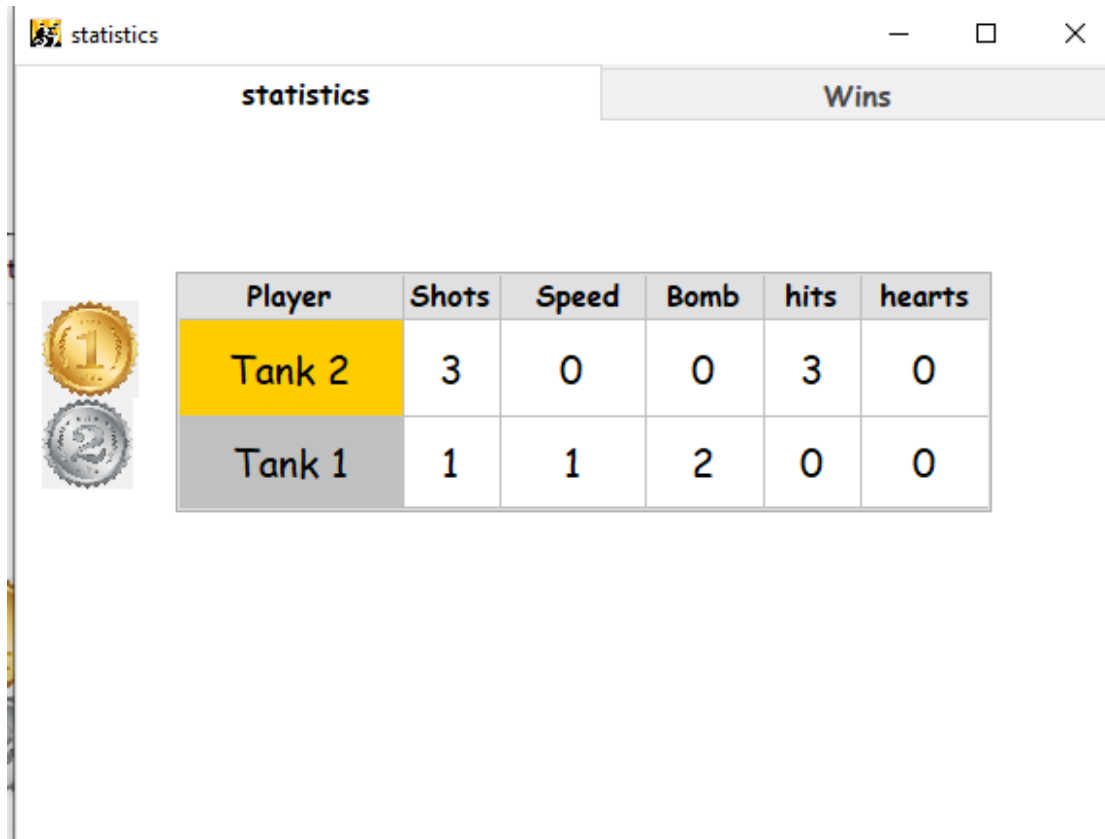


### 5.3. סטטיסטיקה:

מחולק לשני TABS

a. סטטיסטיקה:

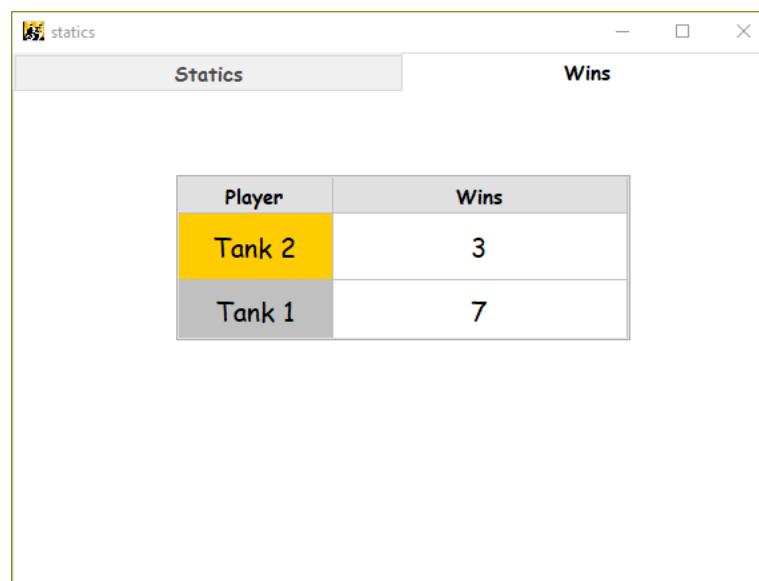
נתונים מהמשחק כגון כמה elements אספתה, כמה יריות ופגיעות וכו.



Player	Shots	Speed	Bomb	hits	hearts
Tank 2	3	0	0	3	0
Tank 1	1	1	2	0	0

b. Wins

נצחונות טנק 1 מול טנק 2 זהו סיכום כללי שנשמר בקובץ txt ומתעדכן בכל משחק.



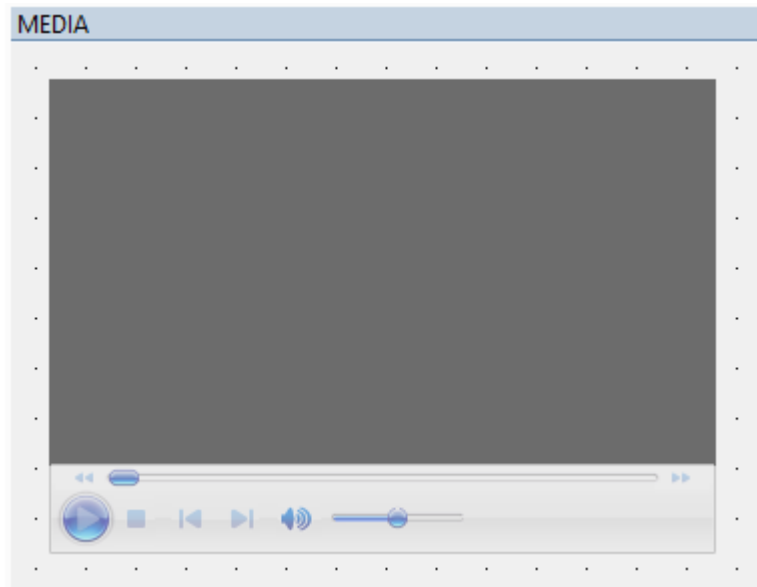
Player	Wins
Tank 2	3
Tank 1	7



---

## 5.4. חלון מכיל WINDOWSMEDIAPLAYER – Media

זהו חלון מוותר המשמש להשמעת הצלילים במשחק.



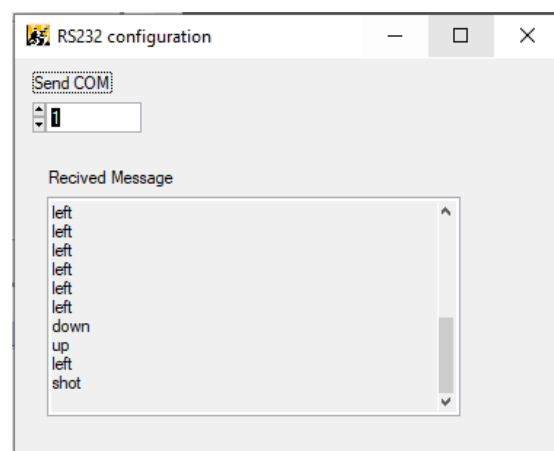
---

## 5.5. חלון RS232 Configuration

חלון לבחינת Com ובנוסף ניתן לראות את כל ההודעות בין שתי המחשבים :

חלון קונפיגורציה לתקשורת בין המחשבים.

ניתן לראות את הפעולות הנשלחות בין שני המחשבים

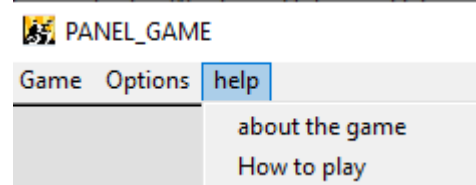
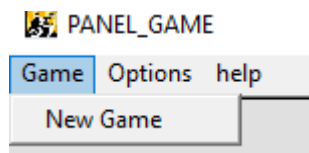
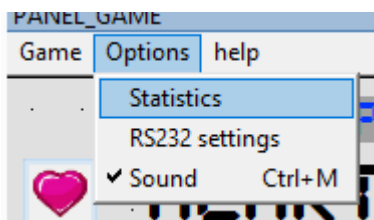
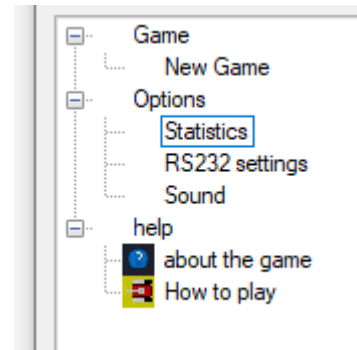


## 5.6. הוראות משחק – How to Play



## 5.7. תפריט בר – MenuBar

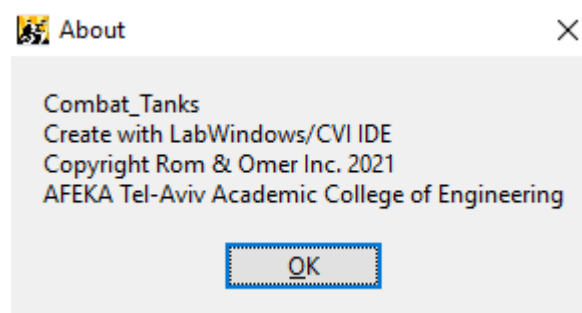
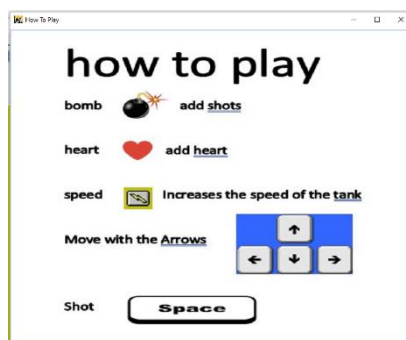
תפריטי בר הנמצאים בחלון המשחק -



New Game – מאתחל משחק חדש

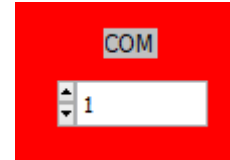
Options – פותח את פאנל של הסטטיסטיקות או של התקשורת וניתן לסמן האם רוצים עם sound או בלי.

עזרה – ניתן לפתוח איך לשחק ולקבל מידע על יצירת המשחק.



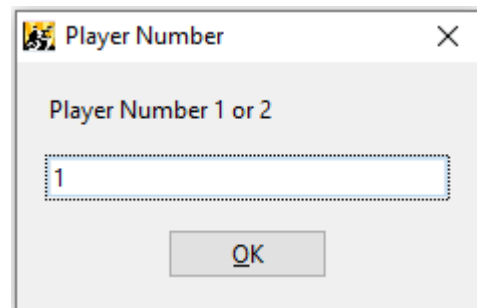
## 6. הוראות שימוש

בכניסה למשחק יש לבחור בחלון ה Com את החיבור שאיתו נעבוד על מנת לתקשר עם המחשב שני



בנוסף ניתן להיכנס ל How to play על מנת לראות את הכפתורים למשחק והאלמנטים שניתן לאסוף במשחק.

לאחר בחירת הCOM נלחץ על "New Game" נבחר את מספר השחקן 1 או 2 (כל מחשב צריך לבחור מספר שונה).



עכשיו ניתן להתחיל להילחם ביריב במידה וגם הוא התחבר למשחק.

בחלק העליון של המשחק מוצג כמות היריות שיש לכל שחקן והיום (ניתן לאסוף עוד חיים ויריות ע"י לקיחת אלמנטים שונים)

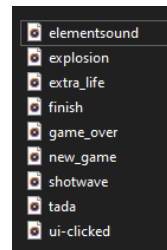


המנצח בקרב הוא מי שמוריד את היריב ל0 חיים.

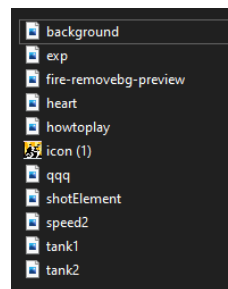
## 7. קבצים בפרויקט

### 7.1. ספריות

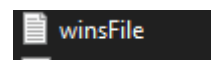
#### Sound



#### Pictures



#### קובץ txt :



#### קבצי קוד:

סוג	קיבצי קוד
C	gameuir.c
H	utils.h
H	gameuir.h
Uir	gameuir.uir
Lib	utils.lib
wmp.fp	wmp.c ו-wmp.h הבולל windows media playerעבור – wmp.fp

## 8. בעיות עיקריות במהלך הפרוייקט :

1. צריכת CPU גבוהה – לאור כך שהשתמשנו בהתחלה בפונקציות Delay למימוש של עיכוב בקוד גרם לצריכת CPU גבוהה מאוד. DELAY ממש מבצע DELAY באמצעות ה CPU ( מצאנו באינטרנט שניתן להשתמש בפונקציות Sleep של windows אשר תבצע את הנדרש ללא הצריכת CPU הגבוהה.
2. לאורך בניית המשחק נתקלנו רבות בצורך של threading לאורך מימוש הפרוייקט למדנו כיצד יהיה ניתן להשתמש בפונקציות שרצות במקביל. על מנת להיות גנרים בפונקציות גם כן לרוב השתמשנו במצביעים כך שהפונקציה תתאים לעבודה עם כל האובייקטים.
3. היו לנו קריסות במשחק בגלל out of memory של ה canvas מסתבר שלא מחקנו חלק מאלמנטים שיצרנו ויצאו מגבולות המפה לכן זה תוקן ע"י העלמת האלמנטים שאינם קיימים יותר במפה.
4. שמירת נתונים רבים על כל אובייקט כגון מספר חיים שנלקחו, כמות יריות של טנק ומהירות וכו... נשמרו בהתחלה ע"י משתנים גלובלים לאחר מכן על מנת לעשות את זה בצורה מסודרת יותר החלטתנו להמיר את זה לערכים שהיו חלק מהאובייקט (מבנה נתונים).
5. תיאום תחילת המשחק בין שני המחשבים – היה קושי לבצע תיאום של תחילת המשחק בין שני המחשבים זה יצר כל פעם באגים בסופו של דבר הצלחתה לכתוב תהליך שביצע את המבוקש.

## 9. סיכום

מטרת הפרויקט הייתה להראות את היכולות שנלמדו בקורס ויכולות נוספות שלמדנו בלמידה עצמית בעזרת האינטרנט וממשק העזרה של CVI, מימשנו את יכולות אלו ע"י יצירת משחק הכולל קרב טנקים.

בפרויקט השתמשנו ביכולות כגון סיבוב אובייקטים (שינוי כיוון הטנק) מומש ע"י עיבוד תמונה, הזזת האובייקטים במרחב מומש ע"י CANVAS ויצירת אובייקט יעודי לשליטה במיקומם, הרצת הרבה פונקציות במקביל לכן השתמשנו threading, השמעת sound לכל מיני מצבים שונים במשחק שמומש בעזרת WINDOWS MEDIA PLAYER – Active X, כתיבה לקובץ לצורך שמירת מידע על המשחק, שימוש בכמה פאנלים, שימוש ב menu bar לצורך ביצוע פעולות מפאנל המשחק, תקשורת בין שני מחשבים בעזרת RS232 (מולטי פלייר) בנוסף נעזרנו בקבצי DLL שכתבנו הכוללים פעולות על האובייקטים שיצרנו לטובת המשחק.

חשוב לנו לציין שעבודה הייתה מאתגרת ולמדנו כלים רבים שנעזר בהם בהמשך, הרגשנו כי אנחנו ממשים את החומר שנלמד במהלך כל הסמסטר ואף נתקלים בבעיות חדשות שאתגרו אותנו .

נהנו לראות את המוצר הסופי שיצרנו ממה שתיכננו בהתחלה.

אנו בטוחים שהכלים שלמדנו מהפרויקט ישמשו אותנו רבות.

תודה על קורס מעניין ושימושי 😊.

## 10. נספח קוד :

הקוד gameuir.c :

```
1 #include "wmp.h"
2 #include <rs232.h>
3 #include "toolbox.h"
4 #include <utils.h>
5
6 #include <ansi_c.h>
7 #include <cvirte.h>
8 #include <userint.h>
9 #include "gameuir.h"
10 #include <stdio.h>
11 #include <Windows.h>
12
13 //DeFine variable
14 #define RAND_MAX 100;
15 #define TIMER_DELAY 50
16 #define PI 3.14159
17 #define SHOT_ELEMENT_HEIGHT 20
18 #define SHOT_ELEMENT_WIDTH 20
19 #define HEART_HEIGHT 25
20 #define HEART_WIDTH 25
21 #define BOMB_HEIGHT 35
22 #define BOMB_WIDTH 40
23 #define BULLET_SIZE 10
24 #define SHOT_SPEED 3.5
25 #define MAX_CHARS 15
26
27
28 int confStatics(void *params);
29 static int playerWin=0;
30 int Config();
31 void MyPolling () ;
32 int SendString (char *SendBuff) ;
33 int displayWin();
34 int displayLose() ;
35
36 //RS232
37 int SendCom = 1;
38 int RecvCom = 1;
39 int Err;
40 int LineNum = 0;
41 static int panelRS;
42
43 int menuBarHandle;
44
45 //struictions
46 typedef enum { right,left,up,down } clickmoute;
47 typedef union mColor {
48 int value;
```



```

49 unsigned char bytes[4];
50 } mColor;
51
52 //function declaration
53 int create_hearts_element();
54 int windowsGame();
55 int RotateOBJECT(OBJECT *T,int state,int tank_num);
56 int threadSound(void *params);
57 int rotate_func() ;
58 int load_image();
59 void restore_pixelArray();
60 int create_shot_element();
61 int shot(void *functionData);
62 int updateTankSpeed(void *params);
63 int create_run_element();
64
65
66 //image process
67 int imageWidth, imageHeight, pixelDepth, pixels, cWidth, cHeight;
68 static int bitsSize,bpr;
69 unsigned char* byteArray;
70 int* rotateArray;
71 int* pixelArray;
72 bool Rotate,Rotate2;
73
74 mColor color;
75 static clickmoute last_click1=left,last_click2=left;
76 static int STATE,STATE2;
77 static int panelGame,panelMedia,panelMain,panelStat,panelHow;
78 static int player_bm,background_bm,fire,explosion,wall,player_bm2,speed_bm,hearts_bm;
79 static int* *player_im;
80 CmtThreadFunctionID thID = 0;
81 int new_game(void);
82 int draw_game (void *functionData);
83 //int draw_game() ;
84 bool windowsGameStart=false;
85 //sound :
86 CAObjHandle mediaHandle;
87 bool statGame=false;
88
89
90
91
92 //shot
93 #define SHOT_SIZE 30
94 static OBJECT SHOT_ARRAY[SHOT_SIZE+1];
95 static OBJECT SPEED_ELEMENT[10];
96 static OBJECT HEARTS_ELEMENT[10];
97 static OBJECT SHOT_ELEMENT[10];
98 static int INDEX_SHOT_ELEMENT=0;
99 static int INDEX_SHOT=0,NUM_SHOTS=SHOT_SIZE,HEARTS=4;
100
101 static bool SHOT=false,FIRE=false;
102 int sound_flag = 1, diff=1, control_conf = 0;
103 bool game_on=true;
104 static OBJECT tank;
105 static OBJECT tank2;

```

```

106 static OBJECT *second_tank;
107 static OBJECT *current_tank;
108 static bool gameon=false ;
109 static playerNum=0;
110 int pause=0;
111 bool CREATE_ELEMENT_RUN = false;
112 bool CREATE_ELEMENT_SHOT = false;
113 bool GamelsStated=false;
114
115 int main (int argc, char *argv[])
116 {
117 tank2.state=0;
118 if (InitCVIRTE (0, argv, 0) == 0)
119 return -1; /* out of memory */
120 if ((panelGame = LoadPanel (0, "gameuir.uir", PANEL_GAME)) < 0)
121 return -1;
122 if ((panelMedia = LoadPanel (0, "gameuir.uir", MEDIA)) < 0)
123 return -1;
124 if ((panelRS = LoadPanel (0, "gameuir.uir", PANEL_RS)) < 0)
125 return -1;
126 if ((panelMain = LoadPanel (0, "gameuir.uir", PANEL_MAIN)) < 0)
127 return -1;
128 if ((panelStat = LoadPanel (0, "gameuir.uir", PANEL_STAT)) < 0)
129 return -1;
130 if ((menuBarHandle = LoadMenuBar(panelGame, "gameuir.uir", BAR_NEWGAM)) < 0)
131 return -1;
132
133 if ((panelHow = LoadPanel(0, "gameuir.uir", PANEL_how)) < 0)
134 return -1;
135 GetObjHandleFromActiveXCtrl (panelMedia, MEDIA_WINDOWSMEDIAPLAYER, &mediaHandle);
136 if(sound_flag)
137 WMPLib_IWMPPlayer4SetURL(mediaHandle, NULL, "sounds//new_game.wav");
138 DisplayPanel (panelMain);
139 RunUserInterface ();
140 DiscardPanel (panelGame);
141 Err = CloseCom (SendCom);
142 Err = CloseCom (RecvCom);
143 return 0;
144 }
145
146
147
148 int CVICALLBACK GAME (int panel, int event, void *callbackData,
149 int eventData1, int eventData2)
150 {
151 static bool notchose=true;
152 if (event == EVENT_CLOSE){
153 game_on=false;
154 QuitUserInterface (0);
155 free(bytesArray);
156 free(pixelArray);
157 }
158 if (gameon){
159 gameon=false;
160 char ResBuff[100];
161 while(notchose){
162

```

```

163 PromptPopup ("Player Number", "Player Number 1 or 2 ",ResBuff,10 );
164 if (strcmp("1",ResBuff)==0){
165 current_tank=&tank;
166 second_tank=&tank2;
167 playerNum=1;
168 notchose=false;
169
170 }
171 else if (strcmp("2",ResBuff)==0){
172 current_tank=&tank2;
173 second_tank=&tank;
174 playerNum=2;
175 notchose=false;
176 }
177 else{
178 MessagePopup("Error", "Please Chose Player 1 or 2");
179 }
180 }
181 SetCtrlVal (panelRS, PANEL_RS_COM, RecvCom);
182 Config();
183 char str[10];
184 sprintf(str, "%d", playerNum);
185 char send[20]="startGame";
186 SendString(strcat(send, str));
187 new_game();
188
189
190
191 }
192 if(statGame){
193 switch ( event)
194 {
195 case EVENT_KEYPRESS:
196 switch(eventData1)
197 {
198 case 32:
199 SendString("shot");
200 CmtScheduleThreadPoolFunction
(DEFAULT_THREAD_POOL_HANDLE,shot,current_tank, &thID);
201 if(sound_flag)
202 CmtScheduleThreadPoolFunction
(DEFAULT_THREAD_POOL_HANDLE,threadSound,"sounds//shotwave.wav",
&thID);
203 current_tank->Number_shot++;
204 break;
205 case 1536: //0x800 arrow up
206 SendString("up") ;
207 if(current_tank->y>0)
208 current_tank->y -= (current_tank->speed);
209 if (last_click1!=up){
210 STATE = 1;
211 current_tank->moveToState = 1;
212 current_tank->rotate=true;
213 Rotate = true;
214 }
215 last_click1=up;
216

```

```

217 break;
218 case 1792: //0x800 arrow down
219 SendString("down");
220 if(current_tank->y<(CANVAS_HEIGHT-PLAYER_HEIGHT))
221 current_tank->y += (current_tank->speed);
222 if (last_click1!=down){
223 STATE = 3 ;
224 current_tank->moveToState = 3;
225 current_tank->rotate=true;
226
227 Rotate = true;
228
229 }
230 last_click1=down;
231
232 break;
233 case 2048: //0x800 arrow left
234 SendString("left");
235 if(current_tank->x>0)
236 current_tank->x -= (current_tank->speed);
237 if (last_click1!=left){
238 STATE = 0 ;
239 current_tank->moveToState = 0;
240 current_tank->rotate=true;
241
242 Rotate = true;
243 }
244 last_click1=left;
245 break;
246 case 2304: //0x900 arrow right
247 SendString("right");
248 if(current_tank->x<(CANVAS_WIDTH-PLAYER_WIDTH))
249 current_tank->x += (current_tank->speed);
250 if (last_click1!=right){
251 STATE = 2 ;
252 current_tank->moveToState = 2;
253 current_tank->rotate=true;
254 Rotate = true;
255 }
256 last_click1=right;
257 break;
258 case 97: // a left
259 if(tank2.x>=0)
260 tank2.x -= (SPEED*2);
261 break;
262 case 100: // d right
263 if(tank2.x<(CANVAS_WIDTH-50))
264 tank2.x += (SPEED*2);
265 break;
266 case 112: // p key
267 pause = !pause;
268 break;
269 default:
270 break;
271 }
272
273 }

```

```

274 }
275 return 0;
276 }
277
278 //image processing
279
280 int load_image(int tank_num){
281 if (tank_num==1)
282 player_im = &player_bm;
283 else if (tank_num==2)
284 player_im = &player_bm2;
285 GetBitmapInfoEx (*player_im, NULL, &bitsSize, NULL, NULL);
286 byteArray=(unsigned char *)malloc(bitsSize);
287 GetBitmapDataEx (*player_im, &bpr, &pixelDepth, &imageWidth, &imageHeight, NULL,
byteArray, NULL, NULL);
288 pixels = imageWidth*imageHeight;
289 pixelArray=(int*)malloc(pixels*sizeof(int));
290 restore_pixelArray();
291 return 0;
292 }
293
294 void restore_pixelArray()
295 {
296 if(pixelDepth==24){
297 for(int i=0, j=0; i<pixels; i++, j+=3){
298 color.bytes[0]=byteArray[j+2];
299 color.bytes[1]=byteArray[j+1];
300 color.bytes[2]=byteArray[j];
301 pixelArray[i]=color.value;
302 }
303 }
304 else if(pixelDepth==32){
305 for(int i=0, j=0; i<pixels; i++, j+=4){
306 color.bytes[0]=byteArray[j];
307 color.bytes[1]=byteArray[j+1];
308 color.bytes[2]=byteArray[j+2];
309 pixelArray[i]=color.value;
310 }
311 }
312 else{
313 QuitUserInterface (0);
314 if(pixelArray!=NULL) free(pixelArray);
315 if(byteArray!=NULL) free(byteArray);
316 if(*player_im) *player_im=DiscardBitmap(*player_im);
317 }
318 }
319
320 int rotate_func ()
321 {
322 int horizontal, vertical, pindex=0;
323 rotateArray=(int*)malloc(pixels*sizeof(int));
324 for(horizontal=0; horizontal<imageWidth; horizontal++)
325 {
326 for(vertical=imageHeight-1; vertical>=0; vertical--)
327 {
328 rotateArray[pindex]=pixelArray[horizontal+vertical*imageWidth];
329 pindex++;

```

```

330 if(pindex==pixels)
331 break;
332 }
333 }
334 for (pindex = 0; pindex<pixels; pindex++)
335 pixelArray[pindex]=rotateArray[pindex];
336 free(rotateArray);
337 // if(player_bm) player_bm=DiscardBitmap(player_bm);
338 bpr=imageHeight*4;
339 imageHeight = imageWidth;
340 imageWidth = bpr/4;
341 NewBitmapEx (bpr, 32, imageWidth, imageHeight, NULL,
342 (unsigned char*)pixelArray, NULL, NULL, player_im);
343
344 return 0;
345 }
346
347 //game function
348
349 int new_game(void){
350 playerWin=0;
351 init_tank1(&tank);
352 init_tank2(&tank2);
353 CanvasStartBatchDraw (panelGame, PANEL_GAME_CANVAS);
354 CanvasClear (panelGame, PANEL_GAME_CANVAS, VAL_ENTIRE_OBJECT);
355 //load images
356 GetBitmapFromFile ("pictures//heart.png", &hearts_bm);
357 GetBitmapFromFile ("pictures//speed2.jpg", &speed_bm);
358 GetBitmapFromFile ("pictures//tank1.jpg", &player_bm);
359 GetBitmapFromFile ("pictures//tank2.jpg", &player_bm2);
360 GetBitmapFromFile ("pictures//background.png", &background_bm);
361 GetBitmapFromFile ("pictures//shotElement.png", &fire);
362 GetBitmapFromFile ("pictures//exp.png", &explosion);
363 CanvasDrawBitmap (panelGame, PANEL_GAME_CANVAS, background_bm,
VAL_ENTIRE_OBJECT, VAL_ENTIRE_OBJECT);
364 CanvasDrawBitmap(panelGame, PANEL_GAME_CANVAS, player_bm, VAL_ENTIRE_OBJECT ,
MakeRect(tank.y, tank.x, PLAYER_HEIGHT, PLAYER_WIDTH));
365 CanvasDrawBitmap(panelGame, PANEL_GAME_CANVAS, player_bm2, VAL_ENTIRE_OBJECT ,
MakeRect(tank2.y, tank2.x-100, PLAYER_HEIGHT, PLAYER_WIDTH));
366 CanvasEndBatchDraw(panelGame, PANEL_GAME_CANVAS);
367 windowsGameStart=true;
368 CmtScheduleThreadPoolFunction (DEFAULT_THREAD_POOL_HANDLE, draw_game, NULL,
&thID);
369 //rotate tank1
370 tank.moveToState=2;
371 tank.rotate=true;
372 //rotate tank2
373 tank2.moveToState=0;
374 tank2.rotate=true;
375 //CmtScheduleThreadPoolFunction
(DEFAULT_THREAD_POOL_HANDLE,create_shot_element, NULL, &thID);
376 //CmtScheduleThreadPoolFunction (DEFAULT_THREAD_POOL_HANDLE,create_run_element,
NULL, &thID);
377 //CmtScheduleThreadPoolFunction
(DEFAULT_THREAD_POOL_HANDLE,create_hearts_element, NULL, &thID);
378 return 0;
379 }

```

```

380
381
382 int draw_game (void *functionData){
383 static bool OneTime=true;
384 bool static exp=false,exp2=false ;
385 static float count_exp=0,count_exp2=0;
386 int j;
387 while(game_on)
388 {
389 SetCtrlVal (panelGame,PANEL_GAME_NUMERIC_SHOTS,tank.shots);
390 SetCtrlVal (panelGame,PANEL_GAME_NUMERIC_HEARTS,tank.heart);
391 SetCtrlVal (panelGame,PANEL_GAME_NUMERIC_SHOTS_2,tank2.shots);
392 SetCtrlVal (panelGame,PANEL_GAME_NUMERIC_HEARTS_2,tank2.heart);
393 while(pause){
394 SetCtrlAttribute (panelGame, PANEL_GAME_TXT, ATTR_VISIBLE, 1);
395 SetCtrlVal (panelGame, PANEL_GAME_TXT, "pause");
396 }
397 SetCtrlAttribute (panelGame, PANEL_GAME_TXT, ATTR_VISIBLE, 0);
398 while(!statGame){
399 SetCtrlAttribute (panelGame, PANEL_GAME_TXT, ATTR_VISIBLE, 1);
400 SetCtrlVal (panelGame, PANEL_GAME_TXT, "Wait to player");
401
402 }
403
404 if (windowsGameStart&&statGame){
405 windowsGameStart=false;
406 SetCtrlAttribute (panelGame, PANEL_GAME_TXT, ATTR_VISIBLE, 1);
407 SetCtrlVal (panelGame, PANEL_GAME_TXT, "START GAME");
408 Sleep(1e3*1);
409 SetCtrlVal (panelGame, PANEL_GAME_TXT, " Fight!!!!");
410 Sleep(1e3*1);
411 SetCtrlAttribute (panelGame, PANEL_GAME_TXT, ATTR_VISIBLE, 0);
412 CmtScheduleThreadPoolFunction (DEFAULT_THREAD_POOL_HANDLE,create_shot_element,
NULL, &thID);
413 CmtScheduleThreadPoolFunction (DEFAULT_THREAD_POOL_HANDLE,create_run_element,
NULL, &thID);
414 CmtScheduleThreadPoolFunction
(DEFAULT_THREAD_POOL_HANDLE,create_heart_element, NULL, &thID);
415 }
416 //Rotate tank1
417 if(tank.rotate){
418 RotateOBJECT(&tank,tank.moveToState,1);
419 Rotate=false;
420
421 }
422 //Rotate tank2
423 if(tank2.rotate){
424 int state2;
425 switch(tank2.moveToState){
426 case 0:
427 state2=2 ;
428 break;
429 case 1:
430 state2=3 ;
431 break;
432 case 2:
433 state2=0;

```

```

434 break;
435 case 3:
436 state2=1;
437 break;
438 }
439 RotateOBJECT(&tank2,state2,2);
440 tank2.rotate=false;
441 }
442 CmtScheduleThreadPoolFunction (DEFAULT_THREAD_POOL_HANDLE,confStatics,
NULL, &thID);
443 CanvasStartBatchDraw (panelGame, PANEL_GAME_CANVAS);
444 CanvasClear (panelGame, PANEL_GAME_CANVAS, VAL_ENTIRE_OBJECT);
445 CanvasDrawBitmap (panelGame, PANEL_GAME_CANVAS, background_bm,
VAL_ENTIRE_OBJECT, VAL_ENTIRE_OBJECT);
446 CanvasDrawBitmap(panelGame, PANEL_GAME_CANVAS, player_bm, VAL_ENTIRE_OBJECT
, MakeRect(tank.y, tank.x, PLAYER_HEIGHT, PLAYER_WIDTH));
447 CanvasDrawBitmap(panelGame, PANEL_GAME_CANVAS, player_bm2,
VAL_ENTIRE_OBJECT, MakeRect(tank2.y, tank2.x-100, PLAYER_HEIGHT,
PLAYER_WIDTH));
448 //#####win_lose#####
449 if (tank.hearts==0){
450 playerWin=2;
451 if (playerNum==2){
452 displayWin(&tank2);
453 }
454 }
455 else{
456 displayLose(&tank);
457 }
458 game_on=false;
459 }
460 }
461 }
462 if (tank2.hearts==0){
463 playerWin=1;
464 if (playerNum==1){
465 displayWin(&tank);
466 }
467 else{
468 displayLose(&tank2);
469 }
470 game_on=false;
471 }
472 }
473 }
474 //##### explosion #####
475 }
476 if (exp&&count_exp<0.07){
477 count_exp+=0.01;
478 CanvasDrawBitmap(panelGame, PANEL_GAME_CANVAS, explosion,
VAL_ENTIRE_OBJECT, MakeRect(tank.y, tank.x, PLAYER_HEIGHT,
PLAYER_WIDTH));
479 }
480 else{
481 count_exp=0;
482 exp=false;
483 }

```



```

484
485 if (exp2&&count_exp2<0.07){
486 count_exp2+=0.01;
487 CanvasDrawBitmap(panelGame, PANEL_GAME_CANVAS, explosion,
VAL_ENTIRE_OBJECT , MakeRect(tank2.y, tank2.x-100, PLAYER_HEIGHT,
PLAYER_WIDTH));
488 }
489 else{
490 count_exp2=0;
491 exp2=false;
492 }
493
494
495 //##### element #####
496
497 for(j=0;j<10;j++){
498 if (SHOT_ELEMENT[j].exist){
499 if
(check_hit_func(&(SHOT_ELEMENT[j]),current_tank,PLAYER_HEIGHT,PLA
YER_WIDTH)){
500 SHOT_ELEMENT[j].exist=false;
501 if(sound_flag)
502 CmtScheduleThreadPoolFunction
(DEFAULT_THREAD_POOL_HANDLE,threadSound,"sounds//elements
ound.wav", &thID);
503 current_tank->shots++;
504 current_tank->Eelement_bomb++;
505 SendString("addshot");
506
507 }
508 if
(check_hit_func(&(SHOT_ELEMENT[j]),second_tank,PLAYER_HEIGHT,PLA
YER_WIDTH)){
509 SHOT_ELEMENT[j].exist=false;
510 if(sound_flag)
511 CmtScheduleThreadPoolFunction
(DEFAULT_THREAD_POOL_HANDLE,threadSound,"sounds//elements
ound.wav", &thID);
512
513
514 }
515 CanvasDrawBitmap(panelGame, PANEL_GAME_CANVAS, fire,
VAL_ENTIRE_OBJECT , MakeRect(SHOT_ELEMENT[j].y,
SHOT_ELEMENT[j].x, SHOT_ELEMENT_HEIGHT,SHOT_ELEMENT_WIDTH));
516 }
517 if (SPEED_ELEMENT[j].exist){
518 if
(check_hit_func(&(SPEED_ELEMENT[j]),current_tank,PLAYER_HEIGHT,PL
AYER_WIDTH)){
519 SPEED_ELEMENT[j].exist=false;
520 if(sound_flag)
521 CmtScheduleThreadPoolFunction
(DEFAULT_THREAD_POOL_HANDLE,threadSound,"sounds//elements
ound.wav", &thID);
522 CmtScheduleThreadPoolFunction
(DEFAULT_THREAD_POOL_HANDLE,updateTankSpeed,current_tank,
&thID);

```

```

523 current_tank->Eelement_speed++;
524 SendString("addspeed");
525
526
527 }
528 if
(check_hit_func(&(SPEED_ELEMENT[j]),second_tank,PLAYER_HEIGHT,PL
AYER_WIDTH)){
529 SPEED_ELEMENT[j].exist=false;
530 if(sound_flag)
531 CmtScheduleThreadPoolFunction
(DEFAULT_THREAD_POOL_HANDLE,threadSound,"sounds//elements
ound.wav", &thID);
532 }
533 CanvasDrawBitmap(panelGame, PANEL_GAME_CANVAS, speed_bm,
VAL_ENTIRE_OBJECT , MakeRect(SPEED_ELEMENT[j].y,
SPEED_ELEMENT[j].x, SHOT_ELEMENT_HEIGHT,SHOT_ELEMENT_WIDTH));
534 }
535
536 if (HEARTS_ELEMENT[j].exist){
537 if
(check_hit_func(&(HEARTS_ELEMENT[j]),current_tank,PLAYER_HEIGHT,P
LAYER_WIDTH)){
538 HEARTS_ELEMENT[j].exist=false;
539 if(sound_flag)
540 CmtScheduleThreadPoolFunction
(DEFAULT_THREAD_POOL_HANDLE,threadSound,"sounds//extra_li
fe.wav", &thID);
541 current_tank->hearts++;
542 current_tank->Eelement_hearts++;
543 SendString("addhearts");
544
545
546 }
547 if
(check_hit_func(&(HEARTS_ELEMENT[j]),second_tank,PLAYER_HEIGHT,P
LAYER_WIDTH)){
548 HEARTS_ELEMENT[j].exist=false;
549 if(sound_flag)
550 CmtScheduleThreadPoolFunction
(DEFAULT_THREAD_POOL_HANDLE,threadSound,"sounds//extra_li
fe.wav", &thID);
551 }
552 CanvasDrawBitmap(panelGame, PANEL_GAME_CANVAS, hearts_bm,
VAL_ENTIRE_OBJECT , MakeRect(HEARTS_ELEMENT[j].y,
HEARTS_ELEMENT[j].x, SHOT_ELEMENT_HEIGHT,SHOT_ELEMENT_WIDTH));
553 }
554
555 }
556
557
558 //##### shot #####
559
560 for(j=0;j<SHOT_SIZE;j++)
561 if (SHOT_ARRAY[j].exist){
562 if (check_hit_func_tank(&SHOT_ARRAY[j],&tank,50,80)){
563 SHOT_ARRAY[j].exist=false;

```

```

564 tank.hearts--;
565 if(sound_flag)
566 CmtScheduleThreadPoolFunction
(DEFAULT_THREAD_POOL_HANDLE,threadSound,"sounds//explosion.wa
v", &thID);
567 exp=true;
568 tank2.hits++;
569
570 }
571 else if (check_hit_func_tank2(&SHOT_ARRAY[j],&tank2,50,80)){
572 SHOT_ARRAY[j].exist=false;
573 tank2.hearts--;
574 exp2=true;
575 if(sound_flag)
576 CmtScheduleThreadPoolFunction
(DEFAULT_THREAD_POOL_HANDLE,threadSound,"sounds//explosion.wa
v", &thID);
577 tank.hits++;
578 }
579 else
580 CanvasDrawOval (panelGame, PANEL_GAME_CANVAS,
MakeRect(SHOT_ARRAY[j].y+5,SHOT_ARRAY[j].x,BULLET_SIZE,BULLET_SIZ
E), VAL_DRAW_INTERIOR);
581 }
582
583 CanvasEndBatchDraw(panelGame, PANEL_GAME_CANVAS);
584 Sleep(0.05*1e3);
585
586 }
587 return 0;
588 }
589
590
591 int updateTankspeed(void *params){
592 OBJECT* T = (OBJECT*)params;
593 T->speed=T->speed+0.3;
594 Sleep(1e3*10);
595 T->speed=(SPEED*2);
596 return 0;
597 }
598
599
600
601 int RotateOBJECT(OBJECT *T,int state,int tank_num){
602 int tmp = T->state;
603 while(tmp!=state){
604 load_image(tank_num);
605 rotate_func();
606 tmp++;
607 if(tmp==4)
608 tmp=0;
609 }
610 T->state=tmp;
611 return 0;
612 }
613
614

```

```

615 int threadSound(void *params){
616 char* Data = (char*)params;
617 WMPLib_IWMPPlayer4SetURL(mediaHandle, NULL, Data);
618 return 0;
619 }
620
621
622 int confStatics(void *params){
623 int tab_handle;
624 int winstank1,winstank2;
625 FILE *fp;
626 fp = fopen("winsFile.txt", "r"); // read mode
627 fscanf(fp, "%d,%d", &winstank1,&winstank2);
628 fclose(fp);
629 GetPanelHandleFromTabPage(panelStat, PANEL_STAT_TAB, 0, &tab_handle);
630 if (playerWin==1){
631 SetTableCellVal (tab_handle, 2, MakePoint (1, 2), "Tank 1");
632 SetTableCellVal (tab_handle, 2, MakePoint (2, 2), tank.Number_shot);
633 SetTableCellVal (tab_handle, 2, MakePoint (3, 2), tank.Eelement_speed);
634 SetTableCellVal (tab_handle, 2, MakePoint (4, 2), tank.Eelement_bomb);
635 SetTableCellVal (tab_handle, 2, MakePoint (5, 2), tank.hits);
636 SetTableCellVal (tab_handle, 2, MakePoint (6, 2), tank.Eelement_hearts);
637 SetTableCellVal (tab_handle, 2, MakePoint (1, 3), "Tank 2");
638 SetTableCellVal (tab_handle, 2, MakePoint (2, 3), tank2.Number_shot);
639 SetTableCellVal (tab_handle, 2, MakePoint (3, 3), tank2.Eelement_speed);
640 SetTableCellVal (tab_handle, 2, MakePoint (4, 3), tank2.Eelement_bomb);
641 SetTableCellVal (tab_handle, 2, MakePoint (5, 3), tank2.hits);
642 SetTableCellVal (tab_handle, 2, MakePoint (6, 3), tank2.Eelement_hearts);
643 winstank1++;
644 GetPanelHandleFromTabPage(panelStat, PANEL_STAT_TAB, 1, &tab_handle);
645 if (winstank1>winstank2){
646 SetTableCellVal (tab_handle, 2, MakePoint (1, 2), "Tank 2");
647 SetTableCellVal (tab_handle, 2, MakePoint (2, 2),winstank2 );
648 SetTableCellVal (tab_handle, 2, MakePoint (1, 3), "Tank 1");
649 SetTableCellVal (tab_handle, 2, MakePoint (2, 3),winstank1 );
650 }
651 else{
652 SetTableCellVal (tab_handle, 2, MakePoint (1, 2), "Tank 1");
653 SetTableCellVal (tab_handle, 2, MakePoint (2, 2),winstank1 );
654 SetTableCellVal (tab_handle, 2, MakePoint (1, 3), "Tank 2");
655 SetTableCellVal (tab_handle, 2, MakePoint (2, 3),winstank2 );
656 }
657 WriteWinsToFile(winstank1,winstank2);
658
659 }
660 else if (playerWin==2){
661 SetTableCellVal (tab_handle, 2, MakePoint (1, 2), "Tank 2");
662 SetTableCellVal (tab_handle, 2, MakePoint (2, 2), tank2.Number_shot);
663 SetTableCellVal (tab_handle, 2, MakePoint (3, 2), tank2.Eelement_speed);
664 SetTableCellVal (tab_handle, 2, MakePoint (4, 2), tank2.Eelement_bomb);
665 SetTableCellVal (tab_handle, 2, MakePoint (5, 2), tank2.hits);
666 SetTableCellVal (tab_handle, 2, MakePoint (6, 2), tank2.Eelement_hearts);
667 SetTableCellVal (tab_handle, 2, MakePoint (1, 3), "Tank 1");
668 SetTableCellVal (tab_handle, 2, MakePoint (2, 3), tank.Number_shot);
669 SetTableCellVal (tab_handle, 2, MakePoint (3, 3), tank.Eelement_speed);
670 SetTableCellVal (tab_handle, 2, MakePoint (4, 3), tank.Eelement_bomb);
671 SetTableCellVal (tab_handle, 2, MakePoint (5, 3), tank.hits);

```

```

672 SetTableCellVal (tab_handle, 2, MakePoint (6, 3), tank.Eelement_hearts);
673 winstank2++;
674 GetPanelHandleFromTabPage(panelStat, PANEL_STAT_TAB, 1, &tab_handle);
675 if (winstank1>winstank2){
676 SetTableCellVal (tab_handle, 2, MakePoint (1, 2), "Tank 2");
677 SetTableCellVal (tab_handle, 2, MakePoint (2, 2),winstank2 );
678 SetTableCellVal (tab_handle, 2, MakePoint (1, 3), "Tank 1");
679 SetTableCellVal (tab_handle, 2, MakePoint (2, 3),winstank1 );
680 }
681 else{
682 SetTableCellVal (tab_handle, 2, MakePoint (1, 2), "Tank 1");
683 SetTableCellVal (tab_handle, 2, MakePoint (2, 2),winstank1 );
684 SetTableCellVal (tab_handle, 2, MakePoint (1, 3), "Tank 2");
685 SetTableCellVal (tab_handle, 2, MakePoint (2, 3),winstank2 );
686
687 }
688 WriteWinsToFile(winstank1,winstank2);
689 }
690 else{
691 SetTableCellVal (tab_handle, 2, MakePoint (1, 2), "Tank 1");
692 SetTableCellVal (tab_handle, 2, MakePoint (2, 2), tank.Number_shot);
693 SetTableCellVal (tab_handle, 2, MakePoint (3, 2), tank.Eelement_speed);
694 SetTableCellVal (tab_handle, 2, MakePoint (4, 2), tank.Eelement_bomb);
695 SetTableCellVal (tab_handle, 2, MakePoint (5, 2), tank.hits);
696 SetTableCellVal (tab_handle, 2, MakePoint (6, 2), tank.Eelement_hearts);
697 SetTableCellVal (tab_handle, 2, MakePoint (1, 3), "Tank 2");
698 SetTableCellVal (tab_handle, 2, MakePoint (2, 3), tank2.Number_shot);
699 SetTableCellVal (tab_handle, 2, MakePoint (3, 3), tank2.Eelement_speed);
700 SetTableCellVal (tab_handle, 2, MakePoint (4, 3), tank2.Eelement_bomb);
701 SetTableCellVal (tab_handle, 2, MakePoint (5, 3), tank2.hits);
702 SetTableCellVal (tab_handle, 2, MakePoint (6, 3), tank2.Eelement_hearts);
703 GetPanelHandleFromTabPage(panelStat, PANEL_STAT_TAB, 1, &tab_handle);
704 if (winstank1>winstank2){
705 SetTableCellVal (tab_handle, 2, MakePoint (1, 2), "Tank 2");
706 SetTableCellVal (tab_handle, 2, MakePoint (2, 2),winstank2 );
707 SetTableCellVal (tab_handle, 2, MakePoint (1, 3), "Tank 1");
708 SetTableCellVal (tab_handle, 2, MakePoint (2, 3),winstank1 );
709 }
710 else{
711 SetTableCellVal (tab_handle, 2, MakePoint (1, 2), "Tank 1");
712 SetTableCellVal (tab_handle, 2, MakePoint (2, 2),winstank1 );
713 SetTableCellVal (tab_handle, 2, MakePoint (1, 3), "Tank 2");
714 SetTableCellVal (tab_handle, 2, MakePoint (2, 3),winstank2 );
715 }
716
717 }
718
719 return 0;
720
721 }
722
723
724 int displayWin(OBJECT* T){
725 SetCtrlAttribute (panelGame, PANEL_GAME_TXT, ATTR_VISIBLE, 1);
726 SetCtrlVal (panelGame, PANEL_GAME_TXT, " WINNER !!!");
727 CmtScheduleThreadPoolFunction
(DEFAULT_THREAD_POOL_HANDLE,threadSound,"sounds//tada.wav", &thID);

```

```

728 Sleep(1e3*2);
729 DisplayPanel(panelStat);
730 return 0;
731 }
732
733
734 int displayLose(){
735 SetCtrlAttribute (panelGame, PANEL_GAME_TXT, ATTR_VISIBLE, 1);
736 SetCtrlVal (panelGame, PANEL_GAME_TXT, " LOSER !!!");
737 CmtScheduleThreadPoolFunction
(DEFAULT_THREAD_POOL_HANDLE,threadSound,"sounds//game_over.wav", &thID);
738 Sleep(1e3*2);
739 DisplayPanel(panelStat);
740 return 0;
741
742 }
743
744
745 int create_hearts_element(){
746 Sleep(1e3*20);
747 int i,x=50,y=50;
748 while(game_on){
749 for(i=0;i<10;i++){
750 OBJECT * current_element=&HEARTS_ELEMENT[i];
751 create_object(current_element,SHOT_ELEMENT_WIDTH,SHOT_ELEMENT_HEIGHT);
752 Sleep(1e3*50);
753 }
754 }
755 for(i=0;i<10;i++)
756 HEARTS_ELEMENT[i].exist=false;
757
758 return 0;
759 }
760
761 int create_run_element(){
762 Sleep(1e3*10);
763 int i,x=50,y=50;
764 while(game_on){
765 for(i=0;i<10;i++){
766 OBJECT * current_element=&SPEED_ELEMENT[i];
767 create_object(current_element,SHOT_ELEMENT_WIDTH,SHOT_ELEMENT_HEIGHT);
768 Sleep(1e3*20);
769 }
770 }
771 for(i=0;i<10;i++)
772 SPEED_ELEMENT[i].exist=false;
773 return 0;
774 }
775
776 int create_shot_element(){
777 int i,x=10,y=10;
778 while(game_on){
779 for(i=0;i<10;i++){
780 OBJECT * current_element=&SHOT_ELEMENT[i];
781 create_object(current_element,SHOT_ELEMENT_WIDTH,SHOT_ELEMENT_HEIGHT);
782 Sleep(1e3*20);
783 }

```

```

784 for(i=0;i<10;i++)
785 SHOT_ELEMENT[i].exist=false;
786 }
787 return 0;
788 }
789
790
791 int shot(void *param){
792 OBJECT* Ttank = (OBJECT*)param;
793 if (Ttank->shots>0){
794 OBJECT *current_shot=&SHOT_ARRAY[INDEX_SHOT];
795 if (current_shot->exist==false){
796 current_shot->exist=true ;
797 if (INDEX_SHOT+1== SHOT_SIZE)
798 INDEX_SHOT=0;
799 else
800 INDEX_SHOT++;
801 Ttank->shots--;
802 SetCtrlVal (panelGame,PANEL_GAME_NUMERIC_SHOTS,Ttank->shots);
803 Ttank->Fire=true;
804 Sleep (0.2);
805 Ttank->Fire=false;
806 if (Ttank->playerNum==1){
807 switch(Ttank->state){
808 case 0:
809 current_shot->x=Ttank->x-120;
810 current_shot->y=Ttank->y+13;
811 while( (current_shot->x-10)>0){
812 if(current_shot->exist){
813 Sleep(1e3*0.01) ;
814 current_shot->x-=(SHOT_SPEED);
815 }
816 }
817 break;
818 case 1:
819 current_shot->x=Ttank->x+PLAYER_WIDTH/2;
820 current_shot->y=Ttank->y-50;
821 while( (current_shot->y-10)>0){
822 if(current_shot->exist){
823 Sleep(1e3*0.01) ;
824 current_shot->y-=(SHOT_SPEED);
825 }
826 }
827
828 break;
829 case 2:
830 current_shot->x=Ttank->x+120;
831 current_shot->y=Ttank->y+13;
832 while( (current_shot->x-10)<CANVAS_WIDTH){
833 if(current_shot->exist){
834 Sleep(1e3*0.01) ;
835 current_shot->x+=(SHOT_SPEED);
836 }
837 }
838 break;
839 case 3:
840 current_shot->x=Ttank->x+PLAYER_WIDTH/2;

```

```

841 current_shot->y=Ttank->y+PLAYER_HEIGHT;
842 while( (current_shot->y-10)<CANVAS_HEIGHT){
843 if(current_shot->exist){
844 Sleep(1e3*0.01) ;
845 current_shot->y+=(SHOT_SPEED);
846 }
847 }
848 break;
849 }
850 }
851 else if (Ttank->playerNum==2){
852 switch(Ttank->state){
853 case 2:
854 current_shot->x=Ttank->x-120;
855 current_shot->y=Ttank->y+13;
856 while( (current_shot->x-10)>0){
857 if(current_shot->exist){
858 Sleep(1e3*0.01) ;
859 current_shot->x-=(SHOT_SPEED);
860 }
861 }
862 break;
863 case 1:
864 current_shot->x=Ttank->x-PLAYER_WIDTH/2;
865 current_shot->y=Ttank->y+PLAYER_HEIGHT;
866 while( (current_shot->y-10)<CANVAS_HEIGHT){
867 if(current_shot->exist){
868 Sleep(1e3*0.01) ;
869 current_shot->y+=(SHOT_SPEED);
870 }
871 }
872 break;
873 case 0:
874 current_shot->x=Ttank->x+120;
875 current_shot->y=Ttank->y+13;
876 while( (current_shot->x-10)<CANVAS_WIDTH){
877 if(current_shot->exist){
878 Sleep(1e3*0.01) ;
879 current_shot->x+=(SHOT_SPEED);
880 }
881 }
882 break;
883 case 3:
884 current_shot->x=Ttank->x-PLAYER_WIDTH/2;
885 current_shot->y=Ttank->y-10;
886 while( (current_shot->y-10)>0){
887 if(current_shot->exist){
888 Sleep(1e3*0.01) ;
889 current_shot->y-=(SHOT_SPEED);
890 }
891 }
892
893 break;
894 }
895
896 }
897 Sleep(1e3*1) ;

```



```

898 current_shot->exist=false;
899 }
900 }
901 return 0;
902 }
903
904
905 //command to tank2
906 int move_tank2(OBJECT *T,int move){
907 switch(move){
908 case 32:
909 CmtScheduleThreadPoolFunction (DEFAULT_THREAD_POOL_HANDLE,shot,T, &thID);
910 if(sound_flag)
911 CmtScheduleThreadPoolFunction
912 (DEFAULT_THREAD_POOL_HANDLE,threadSound,"sounds//shotwave.wav",
913 &thID);
914 T->Number_shot++;
915 break;
916 case 1536: //0x800 arrow up
917 if(T->y>0)
918 T->y -= (T->speed);
919 if (last_click2!=up){
920 T->moveToState = 1;
921 T->rotate = true;
922 }
923 last_click2=up;
924
925 break;
926 case 1792: //0x800 arrow down
927 if(T->y<(CANVAS_HEIGHT-PLAYER_HEIGHT))
928 T->y += (T->speed);
929 if (last_click2!=down){
930 T->moveToState = 3 ;
931 T->rotate = true;
932 }
933 last_click2=down;
934
935 break;
936 case 2048: //0x800 arrow left
937 if(T->x>0)
938 T->x -= (T->speed);
939 if (last_click2!=left){
940 T->moveToState = 0 ;
941 T->rotate = true;
942 }
943 last_click2=left;
944
945 break;
946 case 2304: //0x900 arrow right
947 if(T->x<(CANVAS_WIDTH-PLAYER_WIDTH))
948 T->x += (T->speed);
949 if (last_click2!=right){
950 T->moveToState = 2 ;
951 T->rotate = true;
952 }
953 last_click2=right;
954 break;
955
956

```

```

953 }
954 return 0;
955 }
956
957
958 void CVICALLBACK NewGame (int menuBar, int menuItem, void *callbackData,
959 int panel)
960 {
961 statGame=false;
962 game_on=false;
963 Sleep(1e3*1);
964 game_on=true;
965 char str[10];
966 sprintf(str, "%d", playerNum);
967 char send[20]="startGame";
968 SendString(strcat(send, str));
969 new_game();
970
971 }
972
973
974
975 void MyPolling ()
976 {
977 char RecBuff[100];
978 int N;
979 N = ComRdTerm (RecvCom, RecBuff, 100, 0);
980 if (N>0){
981 RecBuff[N] = 0;
982 if (playerNum==1){
983 if ((strcmp("startGame2",RecBuff)==0)&&(!statGame)){
984 statGame=true;
985 SendString("startGame1");
986 }
987 }
988 else{
989 if ((strcmp("startGame1",RecBuff)==0)&&(!statGame)){
990 statGame=true;
991 SendString("startGame2");
992 }
993 }
994 if (strcmp("left",RecBuff)==0)
995 move_tank2(second_tank,2048);
996 else if (strcmp("right",RecBuff)==0)
997 move_tank2(second_tank,2304);
998 else if (strcmp("up",RecBuff)==0)
999 move_tank2(second_tank,1536);
1000 else if (strcmp("down",RecBuff)==0)
1001 move_tank2(second_tank,1792);
1002 else if (strcmp("shot",RecBuff)==0)
1003 move_tank2(second_tank,32);
1004 else if (strcmp("addhearts",RecBuff)==0){
1005 second_tank->hearts++;
1006 second_tank->Element_hearts++;
1007 }
1008 else if (strcmp("addshot",RecBuff)==0){
1009 second_tank->shots++;

```

```

1010 second_tank->Element_bomb++;
1011 }
1012 else if (strcmp("addspeed",RecBuff)==0) {
1013 CmtScheduleThreadPoolFunction
(DEFAULT_THREAD_POOL_HANDLE,updateTankspeed,second_tank, &thID);
1014 second_tank->Element_speed++;
1015 }
1016 InsertTextBoxLine (panelRS, PANEL_RS_RECIVER, -1, RecBuff);
1017 SetCtrlAttribute (panelRS, PANEL_RS_RECIVER, ATTR_FIRST_VISIBLE_LINE, LineNum++);
1018 }
1019
1020 return;
1021 }
1022
1023
1024 int Config (){
1025 Err = OpenComConfig (SendCom, "", 9600, 0, 7, 1, 512, 512);
1026 Err = OpenComConfig (RecvCom, "", 9600, 0, 7, 1, 512, 512);
1027 SetCtrlAttribute (panelRS, PANEL_RS_TIMER, ATTR_ENABLED, 1);
1028 InstallComCallback (RecvCom, LWRS_RXFLAG, 0, 0, 0, 0);
1029
1030 return 0;
1031 }
1032
1033
1034 int SendString (char *SendBuff)
1035 {
1036 int Len;
1037 Len = strlen(SendBuff);
1038 //
1039 ComWrt (SendCom, SendBuff, Len+1);
1040 return 0;
1041 }
1042
1043
1044 int CVICALLBACK RecTimer (int panel, int control, int event,
1045 void *callbackData, int eventData1, int eventData2)
1046 {
1047 char RecBuff[100];
1048 //Config ();
1049 int N;
1050 switch (event)
1051 {
1052 case EVENT_TIMER_TICK:
1053 N = GetInQLen (RecvCom);
1054 if (!N) return(0);
1055 MyPolling ();
1056 break;
1057 }
1058 return 1;
1059 }
1060
1061
1062 void CVICALLBACK BAR_RS232 (int menuBar, int menuItem, void *callbackData,
1063 int panel)
1064 {
1065 GetCtrlVal (panelRS, PANEL_RS_COM, &SendCom);

```

```

1066 DisplayPanel (panelRS);
1067
1068 }
1069
1070
1071 int CVICALLBACK CB (int panel, int event, void *callbackData,
1072 int eventData1, int eventData2)
1073 {
1074 switch (event)
1075 {
1076 case EVENT_CLOSE:
1077 HidePanel (panelRS);
1078 break;
1079 }
1080 return 0;
1081 }
1082
1083
1084 //main PANEL :
1085
1086
1087 int CVICALLBACK how_to_play_func (int panel, int control, int event,
1088 void *callbackData, int eventData1, int eventData2)
1089 {
1090 switch (event)
1091 {
1092 case EVENT_COMMIT:
1093 DisplayPanel (panelHow);
1094
1095 break;
1096 }
1097 return 0;
1098 }
1099
1100
1101 int CVICALLBACK new_game_button (int panel, int control, int event,
1102 void *callbackData, int eventData1, int eventData2)
1103 {
1104 switch (event)
1105 {
1106 case EVENT_COMMIT:
1107 GetCtrlVal (panelMain, PANEL_MAIN_COM_main, &SendCom);
1108 GetCtrlVal (panelMain, PANEL_MAIN_COM_main, &RecvCom);
1109 SetCtrlVal (panelRS, PANEL_RS_COM, SendCom);
1110
1111 CmtScheduleThreadPoolFunction
(DEFAULT_THREAD_POOL_HANDLE,threadSound,"sounds//ui-clicked.wav", &thID);
1112 gameon=true;
1113 DisplayPanel (panelGame);
1114 HidePanel (panelMain);
1115 break;
1116 }
1117 return 0;
1118 }
1119
1120
1121 int CVICALLBACK QuitCallback (int panel, int control, int event,

```

```

1122 void *callbackData, int eventData1, int eventData2)
1123 {
1124 switch (event)
1125 {
1126 case EVENT_COMMIT:
1127 QuitUserInterface (0);
1128 break;
1129 }
1130 return 0;
1131 }
1132
1133
1134
1135 void CVICALLBACK menu_bar_sound_flag (int menuBar, int menuItem, void *callbackData,
int panel){
1136 sound_flag = !sound_flag;
1137 SetMenuBarAttribute (menuBarHandle, BAR_NEWGAM_Options_sound_bar, ATTR_CHECKED,
sound_flag);
1138 }
1139
1140
1141
1142 int CVICALLBACK QuitStatics(int panel, int event, void *callbackData, int eventData1,
int eventData2)
1143 {
1144 {
1145 switch (event)
1146 {
1147 case EVENT_CLOSE:
1148 HidePanel (panelStat);
1149 break;
1150 }
1151 return 0;
1152 }
1153
1154
1155 int CVICALLBACK QuitMain (int panel, int event, void *callbackData,
1156 int eventData1, int eventData2)
1157 {
1158 switch (event)
1159 {
1160 case EVENT_CLOSE:
1161 QuitUserInterface (0);
1162
1163 break;
1164 }
1165 return 0;
1166 }
1167
1168
1169 void CVICALLBACK Statics (int menuBar, int menuItem, void *callbackData,
1170 int panel)
1171 {
1172 DisplayPanel(panelStat);
1173 }
1174
1175

```

```

1176 int CVICALLBACK quitHow (int panel, int event, void *callbackData,
1177 int eventData1, int eventData2)
1178 {
1179     switch (event)
1180     {
1181     case EVENT_CLOSE:
1182         HidePanel(panelHow);
1183         break;
1184     }
1185     return 0;
1186 }
1187
1188
1189 void CVICALLBACK menu_bar_about (int menuBar, int menuItem, void *callbackData,
1190 int panel)
1191 {
1192     if(sound_flag)
1193         CmtScheduleThreadPoolFunction
1194         (DEFAULT_THREAD_POOL_HANDLE,threadSound,"sounds//ui-clicked.wav", &thID);
1194     MessagePopup("About", "Combat_Tanks \nCreate with LabWindows/CVI IDE\nCopyright Rom
& Omer Inc. 2021\nAFEKA Tel-Aviv Academic College of Engineering");
1195 }
1196
1197
1198 void CVICALLBACK howtoplay_bar (int menuBar, int menuItem, void *callbackData,
1199 int panel)
1200 {
1201     DisplayPanel (panelHow);
1202 }
1203 }
1204
1205 int CVICALLBACK applycom (int panel, int control, int event,
1206 void *callbackData, int eventData1, int eventData2)
1207 {
1208     switch (event)
1209     {
1210     case EVENT_COMMIT:
1211         GetCtrlVal (panelRS, PANEL_RS_COM, &SendCom);
1212         GetCtrlVal (panelMain, PANEL_MAIN_COM_main, &RecvCom);
1213         Config ();
1214     }
1215     break;
1216 }
1217 return 0;
1218 }
1219
1220

```

1221

: uirgame.h 717

```
1 /*****  
2 /* LabWindows/CVI User Interface Resource (UIR) Include File */  
3 /* */  
4 /* WARNING: Do not add to, delete from, or otherwise modify the contents */  
5 /* of this include file. */  
6 /*****  
7  
8 #include <userint.h>  
9  
10 #ifdef __cplusplus  
11 extern "C" {  
12 #endif  
13  
14 /* Panels and Controls: */  
15  
16 #define MEDIA 1  
17 #define MEDIA_WINDOWSMEDIAPLAYER 2 /* control type: activeX, callback  
function: (none) */  
18  
19 #define PANEL_GAME 2 /* callback function: GAME */  
20 #define PANEL_GAME_CANVAS 2 /* control type: canvas, callback  
function: (none) */  
21 #define PANEL_GAME_PICTURE_4 3 /* control type: picture, callback  
function: (none) */  
22 #define PANEL_GAME_NUMERIC_HEARTS_2 4 /* control type: numeric, callback  
function: (none) */  
23 #define PANEL_GAME_NUMERIC_SHOTS_2 5 /* control type: numeric, callback  
function: (none) */  
24 #define PANEL_GAME_PICTURE 6 /* control type: picture, callback  
function: (none) */  
25 #define PANEL_GAME_NUMERIC_HEARTS 7 /* control type: numeric, callback  
function: (none) */  
26 #define PANEL_GAME_NUMERIC_SHOTS 8 /* control type: numeric, callback  
function: (none) */  
27 #define PANEL_GAME_TEXTMSG_3 9 /* control type: textMsg, callback  
function: (none) */  
28 #define PANEL_GAME_TEXTMSG 10 /* control type: textMsg, callback  
function: (none) */  
29 #define PANEL_GAME_TEXTMSG_2 11 /* control type: textMsg, callback  
function: (none) */  
30 #define PANEL_GAME_TXT 12 /* control type: textMsg, callback  
function: (none) */  
31 #define PANEL_GAME_PICTURE_6 13 /* control type: picture, callback  
function: (none) */  
32 #define PANEL_GAME_PICTURE_5 14 /* control type: picture, callback  
function: (none) */  
33 #define PANEL_GAME_PICTURE_3 15 /* control type: picture, callback  
function: (none) */  
34  
35 #define PANEL_how 3 /* callback function: quitHow */  
36 #define PANEL_how_PICTURE 2 /* control type: picture, callback  
function: (none) */
```

```

37
38 #define PANEL_MAIN 4 /* callback function: QuitMain */
39 #define PANEL_MAIN_BUTTON_how 2 /* control type: pictButton, callback
function: how_to_play_func */
40 #define PANEL_MAIN_COM_main 3 /* control type: numeric, callback
function: (none) */
41 #define PANEL_MAIN_EXIT_BUTTON 4 /* control type: pictButton, callback
function: QuitCallback */
42 #define PANEL_MAIN_START_BUTTON 5 /* control type: pictButton, callback
function: new_game_button */
43 #define PANEL_MAIN_PICTURE_2 6 /* control type: picture, callback
function: (none) */
44 #define PANEL_MAIN_PICTURE_6 7 /* control type: picture, callback
function: (none) */
45
46 #define PANEL_RS 5 /* callback function: CB */
47 #define PANEL_RS_TIMER 2 /* control type: timer, callback
function: RecTimer */
48 #define PANEL_RS_COM 3 /* control type: numeric, callback
function: (none) */
49 #define PANEL_RS_RECIVER 4 /* control type: textBox, callback
function: (none) */
50 #define PANEL_RS_applycom 5 /* control type: command, callback
function: applycom */
51
52 #define PANEL_STAT 6 /* callback function: QuitStatics */
53 #define PANEL_STAT_TAB 2 /* control type: tab, callback
function: (none) */
54 #define PANEL_STAT_PICTURE 3 /* control type: picture, callback
function: (none) */
55 #define PANEL_STAT_PICTURE_3 4 /* control type: picture, callback
function: (none) */
56 #define PANEL_STAT_PICTURE_2 5 /* control type: picture, callback
function: (none) */
57
58 /* tab page panel controls */
59 #define TABPANEL_TABLE_stat 2 /* control type: table, callback
function: (none) */
60 #define TABPANEL_PICTURE 3 /* control type: picture, callback
function: (none) */
61 #define TABPANEL_PICTURE_2 4 /* control type: picture, callback
function: (none) */
62
63 /* tab page panel controls */
64 #define TABPANEL_2_TABLE_stat_2 2 /* control type: table, callback
function: (none) */
65
66
67 /* Control Arrays: */
68
69 /* (no control arrays in the resource file) */
70
71
72 /* Menu Bars, Menus, and Menu Items: */
73
74 #define BAR_NEWGAM 1
75 #define BAR_NEWGAM_BAR_GAME 2

```



```

76 #define BAR_NEWGAM_BAR_GAME_BAR_NEWGAME 3 /* callback function: NewGame */
77 #define BAR_NEWGAM_Options 4
78 #define BAR_NEWGAM_Options_ITEM1 5 /* callback function: Statics */
79 #define BAR_NEWGAM_Options_BAR_RS232 6 /* callback function: BAR_RS232 */
80 #define BAR_NEWGAM_Options_sound_bar 7 /* callback function:
menu_bar_sound_flag */
81 #define BAR_NEWGAM_help 8
82 #define BAR_NEWGAM_help_ITEM2 9 /* callback function: menu_bar_about */
83 #define BAR_NEWGAM_help_howtoplay 10 /* callback function: howtoplay_bar */
84
85 #define MENUBAR 2
86 #define MENUBAR_MENU1 2
87 #define MENUBAR_MENU2 3
88
89
90 /* Callback Prototypes: */
91
92 int CVICALLBACK applycom(int panel, int control, int event, void *callbackData, int
eventData1, int eventData2);
93 void CVICALLBACK BAR_RS232(int menubar, int menuitem, void *callbackData, int panel);
94 int CVICALLBACK CB(int panel, int event, void *callbackData, int eventData1, int
eventData2);
95 int CVICALLBACK GAME(int panel, int event, void *callbackData, int eventData1, int
eventData2);
96 int CVICALLBACK how_to_play_func(int panel, int control, int event, void
*callbackData, int eventData1, int eventData2);
97 void CVICALLBACK howtoplay_bar(int menubar, int menuitem, void *callbackData, int panel);
98 void CVICALLBACK menu_bar_about(int menubar, int menuitem, void *callbackData, int
panel);
99 void CVICALLBACK menu_bar_sound_flag(int menubar, int menuitem, void *callbackData, int
panel);
100 int CVICALLBACK new_game_button(int panel, int control, int event, void *callbackData,
int eventData1, int eventData2);
101 void CVICALLBACK NewGame(int menubar, int menuitem, void *callbackData, int panel);
102 int CVICALLBACK QuitCallback(int panel, int control, int event, void *callbackData,
int eventData1, int eventData2);
103 int CVICALLBACK quitHow(int panel, int event, void *callbackData, int eventData1, int
eventData2);
104 int CVICALLBACK QuitMain(int panel, int event, void *callbackData, int eventData1, int
eventData2);
105 int CVICALLBACK QuitStatics(int panel, int event, void *callbackData, int eventData1,
int eventData2);
106 int CVICALLBACK RecTimer(int panel, int control, int event, void *callbackData, int
eventData1, int eventData2);
107 void CVICALLBACK Statics(int menubar, int menuitem, void *callbackData, int panel);
108
109
110 #ifdef __cplusplus
111 }
112 #endif

```

113

:DLL

: Utils.h

```
1 #ifndef PROJECT_UTILS
2 #define PROJECT_UTILS
3
4 #include <windows.h>
5 #include <cvirte.h>
6
7 #define PLAYER_HEIGHT 50
8 #define PLAYER_WIDTH 100
9 #define CANVAS_HEIGHT 624
10 #define CANVAS_WIDTH 1137
11 #define SPEED 5
12
13
14 typedef enum { false, true } bool;
15
16 typedef struct DLLIMPORT object{
17     bool exist;
18     int y;
19     int x;
20     int z;
21     int state;
22     int shots;
23     bool rotate;
24     int hearts;
25     int moveToState;
26     int Fire;
27     int playerNum;
28     int Number_shot;
29     int Eelement_bomb;
30     int Eelement_speed;
31     double speed;
32     int hits;
33     int Eelement_hearts;
34 } OBJECT;
35
36
37 int DLLEXPORT check_hit_func (OBJECT* object1, OBJECT* object2, int height, int width);
38 int DLLEXPORT create_object(OBJECT *objects ,int object_width,int object_height);
39 int DLLEXPORT init_tank1(OBJECT *T);
40 int DLLEXPORT init_tank2(OBJECT *T);
41 int DLLEXPORT ReadWinsFromFile();
42 int DLLEXPORT WriteWinsToFile(int wintank1,int wintank2);
43 int DLLEXPORT check_hit_func_tank (OBJECT* object1, OBJECT* object2, int height, int
width) ;
44 int DLLEXPORT check_hit_func_tank2 (OBJECT* object1, OBJECT* object2, int height, int
width);
```

```
45
46
47 #endif
```

:Utils.c

```
1 #include "utils.h"
2 #include <ansi_c.h>
3 #include "toolbox.h"
4
5 int DLLEXPORT check_hit_func (OBJECT* object1, OBJECT* object2, int height, int width){
6 if (!(object1->exist) || !(object2->exist)) return 0;
7 if (((object1->x < object2->x + width && object1->x > object2->x) ||
8 (object1->x + width/2 < object2->x + width && object1->x + width/2 >
object2->x) ||
9 (object1->x + width < object2->x + width && object1->x + width >
object2->x)) &&
10 ((object1->y < object2->y + height && object1->y > object2->y) ||
11 (object1->y + height/2 < object2->y + height && object1->y + height/2 >
object2->y) ||
12 (object1->y + height < object2->y + height && object1->y + height >
object2->y))) return 1;
13 else return 0;
14 }
15
16
17 int DLLEXPORT create_object(OBJECT *objects, int object_width, int object_height)
18 {
19 srand(clock());
20 objects->exist=true;
21 objects->x=(int)Random(object_width, CANVAS_WIDTH - object_width);
22 objects->y=(int)Random(object_height, CANVAS_HEIGHT - object_height);
23 return 1;
24 }
25
26
27 int DLLEXPORT init_tank1(OBJECT *T){
28 T->y=CANVAS_HEIGHT/2;
29 T->x = 0;
30 T->exist=true;
31 T->shots=8;
32 T->state=0;
33 T->hearts=3;
34 T->rotate=false;
35 T->Fire=false;
36 T->playerNum=1;
37 T->speed=(SPEED*2);
38 T->Number_shot=0;
39 T->Eelement_bomb=0;
40 T->Eelement_speed=0;
41 T->Eelement_hearts=0;
42 T->hits=0;
43 return 0;
44 }
45
46
```

50

```

47 int DLLEXPORT init_tank2(OBJECT *T){
48 T->y= CANVAS_HEIGHT/2;
49 T->x = CANVAS_WIDTH;
50 T->exist=true;
51 T->shots=8;
52 T->state=1;
53 T->hearts=3;
54 T->rotate=false;
55 T->Fire=false;
56 T->playerNum=2;
57 T->speed=(SPEED*2);
58 T->Number_shot=0;
59 T->Eelement_bomb=0;
60 T->Eelement_speed=0;
61 T->Eelement_hearts=0;
62 T->hits=0;
63 return 0;
64 }
65
66 int DLLEXPORT ReadWinsFromFile(){
67 FILE *fp;
68 int wintank1,wintank2;
69 fp = fopen("winsFile.txt", "r"); // read mode
70 fscanf(fp, "%d,%d", &wintank1,&wintank2);
71 fclose(fp);
72 return wintank1,wintank2 ;
73
74 }
75
76 int DLLEXPORT WriteWinsToFile(int wintank1,int wintank2){
77 FILE *fp;
78 fp = fopen("winsFile.txt", "w"); // read mode
79 fprintf(fp, "%d,%d", wintank1,wintank2);
80 fclose(fp);
81 return wintank1,wintank2 ;
82
83
84 }
85
86 int DLLEXPORT check_hit_func_tank2 (OBJECT* object1, OBJECT* object2, int height, int
width)
87 {
88 if (!(object1->exist) || !(object2->exist)) return 0;
89 if (((object1->x > object2->x - width) && (object1->x < object2->x)) &&
((object1->y < object2->y + height) && (object1->y > object2->y)) )
90 return 1;
91 else return 0;
92 }
93
94
95 int DLLEXPORT check_hit_func_tank (OBJECT* object1, OBJECT* object2, int height, int
width)
96 {
97 if (!(object1->exist) || !(object2->exist)) return 0;
98 if (((object1->x < object2->x + width) && (object1->x > object2->x)) &&
((object1->y < object2->y + height) && (object1->y > object2->y)) )
99 return 1;

```

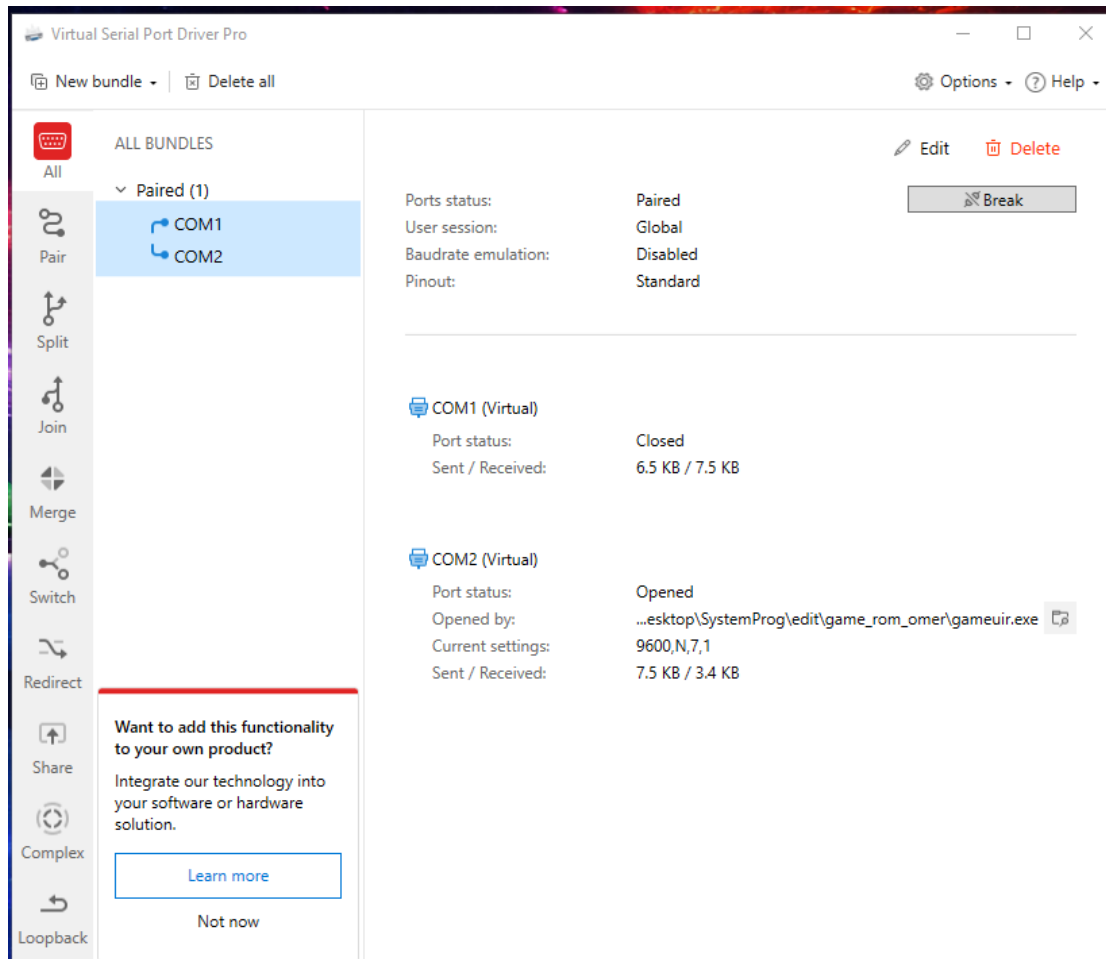
```
100 else return 0;  
101 }
```

102

## 11. נספח וירטואליזציה לתקשורת COM:

השתמשנו בתוכנת : virtual Serial port Driver pro

בתוכנה הגדנו חיבור בין com1 לcom2 כך שמידע שיוצא מ com1 יגיע לcom2



התוכנה מציגה את כמות המידע העוברת בחיבור והאם הוא מחובר או סגור .

ניתן למצוא את התוכנה להורדה באינטרנט.

קובץ ההתקנה מצורף בRAR.