שם הפרוצדורה: DrawSpaceShip

פרוצדורה זו אחראית על ציור ומחיקת השחקן (חללית)

<u>טענת כניסה</u>: מקבלת את שלושה פרמטרים

1.[bp+6] - הפרמטר הזה קובע את הצבע של החללית

2.[bp+4] – הפרמטר הזה קובע את ה X של החללית

3.[bp+2] – הפרמטר הזה קובע את ה Y של החללית

<u>טענת יציאה</u>: ציור של חללית במקום ובצבע המתאים לפי הפרמטרים

<u>קוד הפרוצדורה</u>

```
;taking X pos then Y pos
; [bp + 2] - Y
; [bp + 4] - X
; [bp + 6] - SpaceShip_color / or black_color
proc DrawSpaceShip
;first pixel
        mov bp,sp
        pusha

        mov cx,[bp+4] ;X
        mov dx,[bp+2] ;Y

        mov ah,0ch
        mov bl,0
        mov al,[bp + 6] ;color
        int 10h

        ;adjast pos
        inc dx
        sub cx,2

        ;second row
        mov si,0 ;clearing si for the loop
```

```
second_row_draw:
        inc cx
        int 10h
        inc si
        cmp si,3
        jne second_row_draw

        ;adjast pos
        inc dx
        add cx,2
        ;third row
        mov si,0 ;clearing si for the loop
third_row_draw:
        dec cx
        int 10h
        inc si
        cmp si,5
        jne third_row_draw

        ;adjast pos
        inc dx
        dec cx
        ;forth row
forth_row_draw:
        inc cx
        int 10h
        dec si
        cmp si,0
        jne forth_row_draw
```

```
        ;adjast pos
        inc dx
        add cx,2
        ;fifth row
fifth_row_draw:
        dec cx
        int 10h
        inc si
        cmp si,7
        jne fifth_row_draw

        ;adjast pos
        inc dx
        dec cx
        ;Left booster
        mov si,0 ;clearing si for the loop
fifth:
        inc cx
        int 10h
        inc si
        cmp si,2
        jne fifth
        inc dx
        dec cx
        int 10h
        ;adjast pos
        mov si,0 ;clearing si for the loop
six:
        inc cx
        inc si
        cmp si,7
```

```
        jne six

        ;Right booster
        dec cx
        int 10h
        dec dx
        int 10h
        dec cx
        int 10h
;End of proc
        popa
ret 6
endp DrawSpaceShip
```

שם הפרוצדורה:DrawUnderLine

פרוצדורה זו מציירת את הקוו הלבן שנמצא מתחת לחללית לקוו אין שום מטרה במשחק הוא רק לאסטטיקה

טענת כניסה: *הפרוצדורה לא לוקחת פרמטרים*

טענת יציאה: קוו לבן בתחתית המסך

קוד הפרוצדורוה

```
;proc that draws the buttom line
proc DrawUnderLine
mov cx, 0
mov dx, 190
mov ah,0ch
mov bl,0
mov al,0fh ;color
int 10h
loop0:
inc cx
int 10h
cmp cx,319
jne loop0
ret
endp DrawUnderLine
```

שם הפרוצדורה: YellowIndicators

פרוצדורה זו מציירת את שני הקווים הצהובים הקטנים בצידי המסך, מטרת קווים אלו היא לסמן לשחקן עד איפה יכולים להגיע החיזרים לפני שהוא מפסיד.

<u>טענת כניסה:</u> * הפרוצדורה לא לוקחת פרמטרים*

<u>טענת יציאה:</u> שני פסים צהובים בצדדי המסך

<u>קוד הפרוצדוה</u>

```
;this proc created the yellow lines that indicates when you lose
proc YellowIndicators
add ax, 8
mov cx,0 ;X
mov dx,175 ;Y
mov ah,0ch
mov bl,0
mov al,0eh ;color
;need to inc cx twice
xor si,si
yellowLeftLoop:
        int 10h
        inc cx
        inc si
cmp si,2
jl yellowLeftLoop

mov cx,319
xor si,si
yellowRightLoop:
        int 10h
        dec cx
```

```
        inc si
cmp si,2
jl yellowRightLoop

ret
endp YellowIndicators
```

שם הפרוצדורה: Delay

פרוצדורה זו מוסיפה דילי למשחק. בעצם גורמת למשחק לרוץ לאט יותר (הזמן בין פריים לפריים)

טענת כניסה: *הפרוצדורה לא לוקחת פרמטרים*

טענת יציאה: אין..

קוד הפרוצדורה

```
;proc that adds delay
proc Delay

        mov cx,11
outer_waitloop:
        push cx

        mov cx,64000
        inner_waitloop:
                loop inner_waitloop

        pop cx
        loop outer_waitloop

;end of proc
        ret
endp Delay
```

שם הפרוצדורה DidYouWin

פרוצדורה זו בודקת אם כל האויבים מתים ומשנה משתנה בוליאני בהתאם

<u>טענת כניסה:</u> *הפרוצדורה לא לוקחת פרמטרים*

<u>טענת יציאה:</u> [win] יהיה שווה ל1 או 0 בסוף הפרוצדורה

<u>קוד הפרוצדורה</u>

```
proc DidYouWin
        mov [win],1
        mov si,0
        scanEnemies:

                cmp [Is_Enemy_Alive+si],1
                jne dontupdate
                mov [win],0
                dontupdate:
                inc si
                cmp si,9
        jl scanEnemies

ret
endp DidYouWin
```

שם הפרוצדורה LosingCondition

פרוצדורה זו בודקת אם האויבים הגיעו לפסים הצהובים שמסמנים הפסד ומשנה את [lost] בהתאם

<u>טענת כניסה:</u> *הפרוצדורה לא לוקחת פרמטרים*

<u>טענת יציאה:</u> הפרוצדורה תשנה את [lost]ל 1 או 0 בהתאם להשוואה

<u>קוד הפרוצדורה</u>

```
proc LosingCondition

cmp [First_Row_EnemiesY],165
jbe DidntLose
mov [Lost],1
DidntLose:

ret
endp LosingCondition
```

שם הפרוצדורה erase_mess

פרוצדורה זו מציירת קוו מתחת לשורת החייזרים בגלל שבמהלך התזוזה יש בעיה עם הציור

(*הערה הפרוצדורה היתה אמורה ליהיות תיקון זמני אבל לא הצלחתי לתקן את הבעיה בלי לפגוע בשאר הקוד)

<u>טענת כניסה:</u> *הפרוצדורה לא לוקחת פרמטרים*

<u>טענת יציאה:</u> קוו שחור מתחת לשורת החייזרים

<u>קוד הפרוצדורה</u>

```
proc erase_mess
        xor bh,bh
        mov ax,[First_Row_EnemiesY]
        add ax, 8
        mov cx,0 ;X
        mov dx,ax ;Y
        mov ah,0ch
        mov bl,0
        mov al,0  ;color (black)
        cleanMess:
        int 10h
        inc cx
        cmp cx,320
        jle cleanMess

ret
endp erase_mess
```

שם הפרוצדורה Clearing_Shot

פרוצדוה זו אחרית למחיקת היריה לאחר פגיע בחייזר

<u>טענת כניסה:</u> *הפרוצדורה לא לוקחת פרמטרים*

<u>טענת יציאה:</u> מחקית הפיקסל שנשאר לאחר פגיעה

<u>קוד הפרוצדורה</u>

```
proc Clearing_Shot
        cmp [IsShooting],1
        jne Stop_erasing_shot2
        mov cx,[BulletX]
        mov dx,[BulletY]

        mov al,0
        mov bl,0
        mov ah,0ch
        int 10h

        inc dx
        int 10h

        ;cmp dx,0
        ;jg Stop_erasing_shot2

        ;mov [IsShooting],0
Stop_erasing_shot2:
        ret
endp Clearing_Shot
```

שם הפרוצדורה Shoot

פרוצדורה זו מציירת קוו ישר על ציר ה-Y מנקודת ההתחלה עד פגיעה או בסוף המסך או בחיזר (*הערה הפרוצדורה היא רק חצי ממהלך היריה ועובדת ביחד עם erase_shot)

<u>טענת כניסה:</u> *הפרוצדורה לא לוקחת פרמטרים*

<u>טענת יציאה:</u> קוו ישר בצבע אדום מהשחקן עד לפגיע

<u>קוד הפרוצדורה</u>

```
proc Shoot
        ;cmp [IsShooting],1
        ;jne Finish_Shooting
        mov cx,[BulletX]
        mov dx,[BulletY]
        mov al,4;color
        mov bl,0
        mov ah,0ch
        int 10h
        ;cmp dx,1
        ;jg Finish_Shooting
        ;mov [IsShooting],1
        ;Finish_Shooting:
ret
endp Shoot
```

שם הפרוצדורה erase_shot

פרוצדורה זו היא המשלימה לפרוצדורה Shoot המוזכרת למעלה ואחראית למחיקת השביל שהיריה יוצרת וכתוצאה מכך היריה נראת כמו פיקסל אחד

<u>טענת כניסה:</u> *הפרוצדורה לא לוקחת פרמטרים*

<u>טענת יציאה:</u> קוו ישר בצבע שחור בדיליי של פיקסל אחד מהיריה

<u>קוד הפרוצדורה</u>

```
proc erase_shot
        cmp [IsShooting],1
        jne Stop_erasing_shot
        mov cx,[BulletX]
        mov dx,[BulletY]
        inc dx; 1 delay
        mov al,0;color
        mov bl,0
        mov ah,0ch
        int 10h
        cmp dx,0
        jg Stop_erasing_shot
        mov [IsShooting],0
        Stop_erasing_shot:
ret
endp erase_shot
```

שם הפרוצדורה Find_Dead_enemy

לאחר שזוהתה פגיע בחיזר הפרוצדורה הזאת מחשבת באיזה שחקן פגע היריה ומעדכנת את המקום המתאים לו במערך. (הפרוצדורה הזאת הולכת ביחד עם**Check_For_Hit_First_Row**)

<u>טענת כניסה:</u> *הפרוצדורה לא לוקחת פרמטרים*

<u>טענת יציאה:</u> מערך Is_Enemy_Alive ומערך First_Row_EnemiesX יתאפסו במקום של של החייזר שנפגע .(העדכון הגרפי מגיע פרוצדורה המשלימה Check_For_Hit_First_Row)

<u>קוד הפרוצדורה</u>

```
proc Find_Dead_enemy
mov di,0
mov si,0

Not_This:
mov ax,[First_Row_EnemiesX+si]
add si,2
inc di
add ax,11
cmp ax,[BulletX]
jl Not_This
dec di
mov [Is_Enemy_Alive+di],0
add di,di
mov [First_Row_EnemiesX+di],0
mov [BulletY],0

ret
endp Find_Dead_enemy
```

שם הפרוצדורה Check_For_Hit_Firs_Row

פרוצדורה זו היא הפרוצדורה שמזהה פגיעה בחייזרים לפי השוואת צבעים. כאשר היא מזהה שאחד מהחיזרים נפגע היא קוראת לפרוצדורה Find_Dead_enemy כדי לאתר את האחד שנפגע ולאחר מכן היא מוחקת ומציירת מחדש את החייזרים בשביל העדכון הגרפי

<u>טענת כניסה:</u> Y - [bp+2]

X - [bp+4]

<u>טענת יציאה:</u> החייזר שנפגע ימחק

<u>קוד הפרוצדורה</u>

```
proc Check_For_Hit_First_Row
mov bp,sp

mov cx,[bp+4]
mov dx,[bp+2]
xor bx,bx
mov ah,0dh
dec dx
int 10h
cmp al, [purple_color]
jne EndCheck
        call Clearing_Shot

        ; reset shooting status
        mov [IsShooting],0

        mov [color],0
        call Draw_First_Enemy_Row
        call erase_mess
        call Find_Dead_enemy
```

```
        mov [color],5
        call Draw_First_Enemy_Row
        call erase_mess

EndCheck:
ret 4
endp Check_For_Hit_First_Row
```

שם הפרוצדורה Draw_First_Enemy_Row

פרוצדורה זו מציירת את שורת החייזרים בהתאם למערך שאומר אם הם חיים או מתים, בהתאם למערך של המקומות שלהם, בהתאם למערך של הפיקסלים ובהתאם למיקם ב-Y (ניתן למצוא תרשים זרימה של פרוצדורה זו בסוף הספר)

<u>טענת כניסה:</u> הפרוצדוה לא מקבלת פרמטרים אך משתמשת בהרבה נתונים מ**DATASEG**

- **Is_Enemy_Alive**
- **First_Row_EnemiesX**
- **First_Row_EnemiesY**
- **Enemy_Color_Map**
- **Color**

<u>טענת יציאה:</u> השורה של החייזרים תצויר

<u>קוד הפרוצדורה</u>

```
proc Draw_First_Enemy_Row
        mov si,0
        mov di,0
    Draw_First_Enemy_Row_loop:
        cmp [Is_Enemy_Alive+di],1; checks if the enemy is dead or alive
        jne Skip_Enemy; skips if dead
        mov cx,[First_Row_EnemiesX+si];X
        mov dx,[First_Row_EnemiesY];Y
        mov ah,0ch
        mov bl,0
        mov al,[color] ;color
        ;pushing di to the the stack to use it saperatly to scan
Enemy_Color_Map
        push di
        ;<Drawing the Enemy>
        mov di,0
        mov [i],0
```

```
            Yloop:
                mov [j],0
                mov cx,[First_Row_EnemiesX+si]
                Xloop:
                        cmp [Enemy_Color_Map+di],0
                        je Skip_Pixel
                        int 10h
                        Skip_Pixel:
                        inc cx
                        inc [j]
                        inc di
                cmp [j],10
                jle Xloop
                inc dx
                inc [i]
                cmp [i],8
            jle Yloop
            ;</Drawing the Enemy>
            ;Poping di out to use it again for the Living Status of the enemy
            pop di
            Skip_Enemy:
            add si,2
            inc di
    cmp si,16
    jle Draw_First_Enemy_Row_loop
ret
endp Draw_First_Enemy_Row
```

שם הפרוצדורה Move_Enemies

הפרוצדורה הזאת אחראית על **כל התזוזה** של החייזרים (ימינה שמאלה למטה)

בנוסף היא גם אחראית על ה**עדכון הגרפי** (מוחקת ומציירת כדי לעדכן)

(ניתן למצוא תרשים זרימה של פרוצדורה זו בסוף הספר ~~וגם בהערות~~)

<u>טענת כניסה:</u> הפרוצדוה לא מקבלת פרמטרים אך משתמשת בהרבה נתונים מ**DATASEG**

- **leftORRight**
- **Is_Enemy_Alive**
- **First_Row_EnemiesX**
- **First_Row_EnemiesY**
- **Color**
- **S**

<u>טענת יציאה:</u> השורה של החייזרים תזוז בהתאם למצב הקודם שלהם

<u>קוד הפרוצדורה</u>

```
proc Move_Enemies
        cmp [leftORRight],0
        je EnemiesLeft
        cmp [leftORRight],1
        je EnemiesRight

        EnemiesLeft:
        xor si, si
        loop7:
        cmp [Is_Enemy_Alive+si],1
        je end_loop7
        inc si
        jmp loop7
        end_loop7:
        cmp si,9
```

```
jae fakeProcEnd



add si,si
;now Si stores the location of the MOST LEFT ENEMY ALIVE
;saving si to be used later
mov [s],si
;updating pos to the left by 10
mov [color],0
call Draw_First_Enemy_Row
call erase_mess
;moving si back into si from [s]
mov si,[s]
cmp [First_Row_EnemiesX+si],10
jle MovingDownFR
mov di,si
mov ax,18

loop8:
sub [First_Row_EnemiesX+di],10
add di,2
cmp di,ax
jne loop8

jmp procEnd
MovingDownFR:
add [First_Row_EnemiesY],10
mov [leftORRight],1
```

```
fakeProcEnd:
jmp procEnd
;=====Right=====
EnemiesRight:
mov si,8
loop10:
cmp [Is_Enemy_Alive+si],1
je end_loop10
dec si
jmp loop10
end_loop10:

cmp si,0
jle ProcEnd

add si,si
;now Si stores the location of the MOST Right ENEMY ALIVE
;saving si to be used later
mov [s],si
;updating pos to the Right by 10
mov [color],0
call Draw_First_Enemy_Row
call erase_mess
;moving si back into si from [s]
mov si,[s]
cmp [First_Row_EnemiesX+si],300
jae MovingDownFR_
mov di,si
```

```
loop11:
add [First_Row_EnemiesX+di],10
sub di,2
cmp di,0
jne loop11

jmp procEnd
MovingDownFR_:
add [First_Row_EnemiesY],10
mov [leftORRight],0
;==
ProcEnd:
mov [color],5
call Draw_First_Enemy_Row
call erase_mess
```

שם הפרוצדורה Handle_Input

פרוצדורה זו אחראית על עיבוד הקלט מהמשתמש.

<u>טענת כניסה:</u> *הפרוצדורה לא לוקחת פרמטרים*

<u>טענת יציאה:</u> תלוי במשתמש... אם הוא לחץ חץ למעלה אז יריה חץ ימינה תזוזה ימינה וכו...

<u>קוד הפרוצדורה</u>

```
proc handle_input

WaitForData:
        in al,64h
        cmp al,10b
        je intermediate_endmovements
        in al,60h
        cmp al,4bh
        je goLeft
        cmp al,4dh
        je goRight
        cmp al,48h
        je Shooting
intermediate_endmovements:
        jmp endmovments

Shooting:
        cmp [IsShooting],1
        je endmovments
                ; init shooting condition
                mov [IsShooting],1
                mov ax,[SpaceShipX]
```

```
            mov bx,[SpaceShipY]
            sub bx,2
            mov [BulletX], ax
            mov [BulletY], bx
        jmp endmovments

goLeft:
        ;boundry
        cmp [SpaceShipX],5
        jle endmovments
            ;<erasing>
            push [black_color]
            push [SpaceShipX]
            push [SpaceShipY]
            call DrawSpaceShip
            ;</erasing>
            ;<draw new Space Ship>
            dec [SpaceShipX]
            push [SpaceShip_color]
            push [SpaceShipX]
            push [SpaceShipY]
            call DrawSpaceShip
            ;</draw new Space Ship>
        jmp endmovments

goRight:
        ;boundry
        cmp [SpaceShipX],314
```

```
        jge endmovments
                ;<erasing>
                push [black_color]
                push [SpaceShipX]
                push [SpaceShipY]
                call DrawSpaceShip
                ;</erasing>
                ;<draw new Space Ship>
                inc [SpaceShipX]
                push [SpaceShip_color]
                push [SpaceShipX]
                push [SpaceShipY]
                call DrawSpaceShip
                ;</draw new Space Ship>

endmovments:
;End of proc
        ret
endp handle_input
```