ASSEMBLER PROJECT THE BINDING OF ISAAC

Programmer name: Omer Golan

Id: 326247814

Teacher: Anatoly Peymaer

Grade: 10(3)

School: Neomi shemer

INTRO

Project name: the binding of Isaac

File name: project.exe

Workspace: tasm

Development space: notepad++

Running space: DOSBox

Required files:

Ab.pcx – about me file
BRC.pcx – boss room close
BRO.pcx – boss room open
ERC.pcx – enemy room close
ERO.pcx – enemy room open
Intro.pcx – intro (first window)
itemRI.pcx – item room with item
ItemRwi.pcx – item room without the item
loseW.pcx – lose screen
M1_p.pcx – menu phase 1
M2_p.pcx – menu phase 2
Room1.pcx – first room
Win.pcx – win screen
bitmap.dat - all the bit map

MAIN IDEA

My game based on very similar game that called the Binding of Isaac.
I spent a lot of hours in the game and I thought that its will Be very fun to do it in my own

In the game you are Isaac, the background of the Game is story that base on biding of Isaac (עקדת יצחק)
You fell into basement with a lot of enemies and you Need to go a cross the rooms and defat the devil.
Link for the full story –
(Warning The video is scary and recommended to watch over the age of 16)
https://www.youtube.com/watch?v=5d_rLgDhZfo

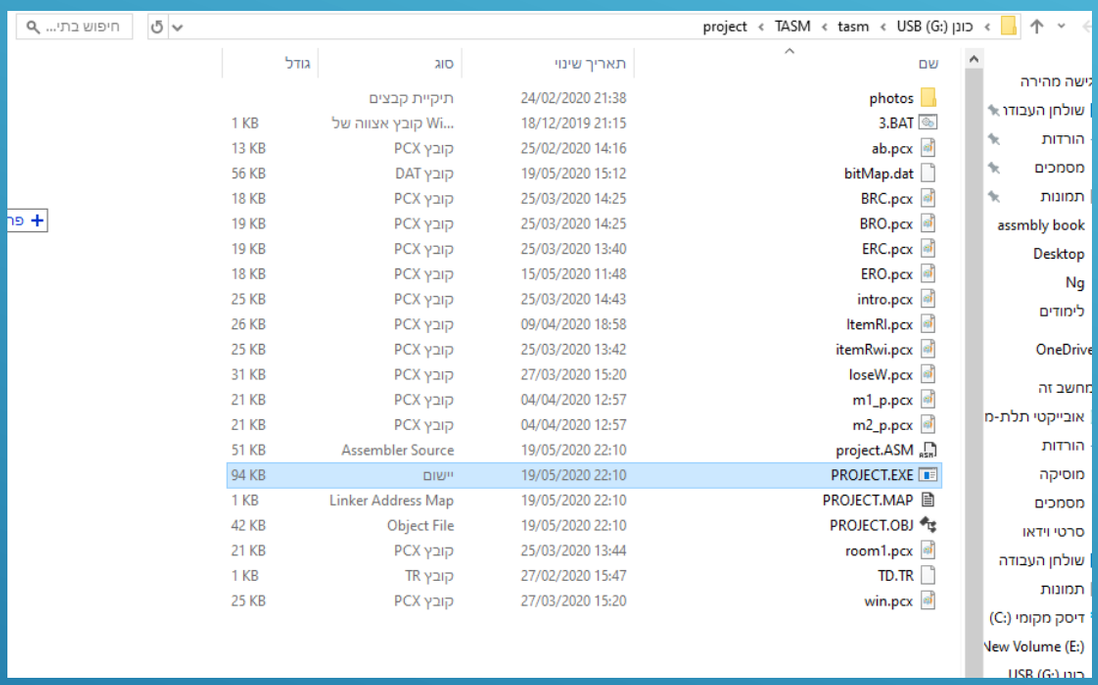
The purpose in my game is to go to the last room and Defeat the boss

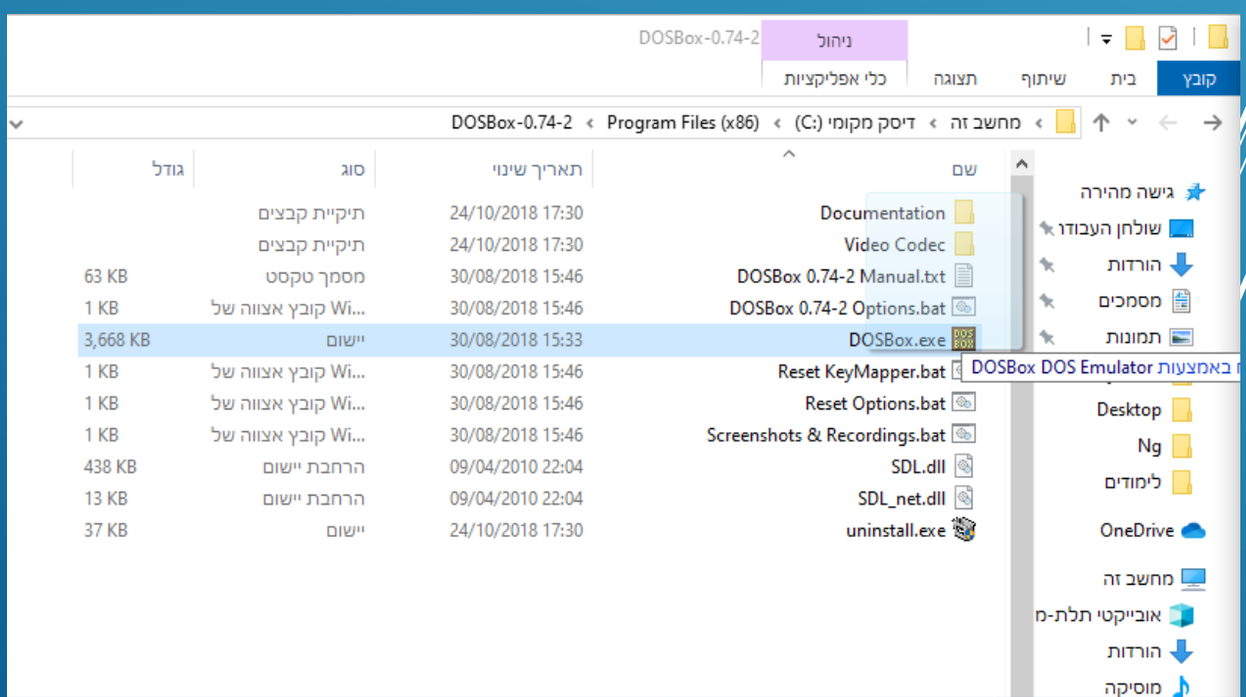
If the enemies hit you, you lose a heart and if your Hearts Are zero you lose

HOW TO RUN THE GAME

First of all ,to start play the game, you will need the software **DOSBOX** to run the exe file.

To allow the game to start, you must drag the exe
file on the **DOSBOX** software and the game starts.

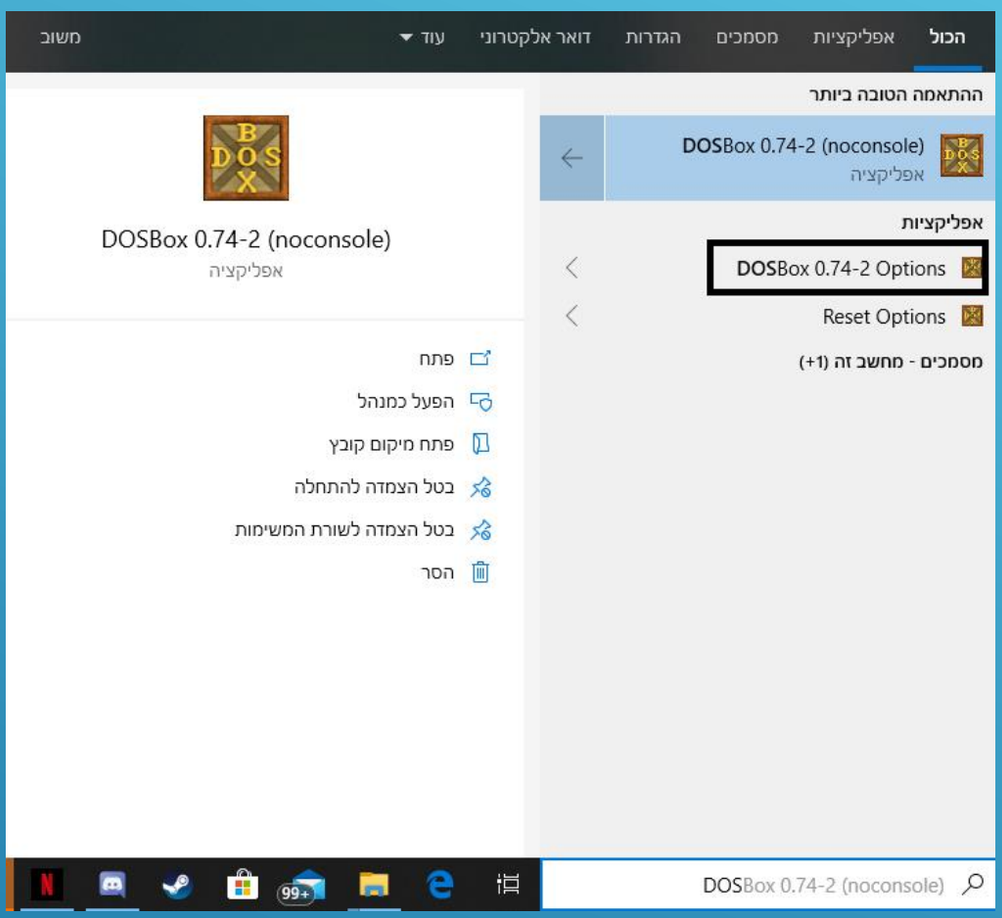




Continue in the next page

I recommended to set the cycles of the cpu to max
You can do it like that:

.1



.2

```
[cpu]
core: CPU Core used in emulation. auto will switch to dynamic if available and #
appropriate.                                                                    #
Possible values: auto, dynamic, normal, simple.                                 #
cputype: CPU Type used in emulation. auto is the fastest choice.                #
Possible values: auto, 386, 386_slow, 486_slow, pentium_slow, 386_prefetch.     #
cycles: Amount of instructions DOSBox tries to emulate each millisecond.        #
Setting this value too high results in sound dropouts and lags.                 #
Cycles can be set in 3 ways:                                                    #
'auto'          tries to guess what a game needs'                               #
It usually works, but can fail for certain games.                               #
fixed #number' will set a fixed amount of cycles. This is what you usually'     #
need if 'auto' fails. (Example: fixed 4000).                                    #
max'            will allocate as much cycles as your computer is able to'       #
handle.                                                                         #
Possible values: auto, fixed, max.                                              #
cycleup: Amount of cycles to decrease/increase with keycombos.(CTRL-F11/CTRL-F12) #
cycledown: Setting it lower than 100 will be a percentage.                      #

core=auto
cputype=auto
cycles=max
cycleup=10
cycledown=20
```

HOW TO PLAY

The keys:
W-move up
A-move left
D-move right
S-move down

Up arrow – shot up
down arrow – shot down
left arrow – shot left
right arrow – shot right

Esc – to return to the menu

The game start in the intro and after the intro you are going to be in the menu you can chose Between the options and if you chose "new run" And press enter you start the game.
The game start in room with 2 doors you can go To the left or to the right every door leads you To another room one with enemy and the other With item.
If you pick up the item your tears(shots) become Bigger and stronger
When you kill the enemy the doors are open And you go the boss room and if you defeat The boss and go to the trap door you win the Game
If you lose all your hearts you lose and you Can try again

GAME VERSION

The game's first version is build from 4 rooms: Item room, enemy room ,boss room and base room
The game containing one enemy and 1 boss also The game containing 2 kinds of shoots, hearts system, Map and sound

In the next version I would like to add more rooms and Enemies and also to improve to system and fix bugs

Flow chart

function

Condition

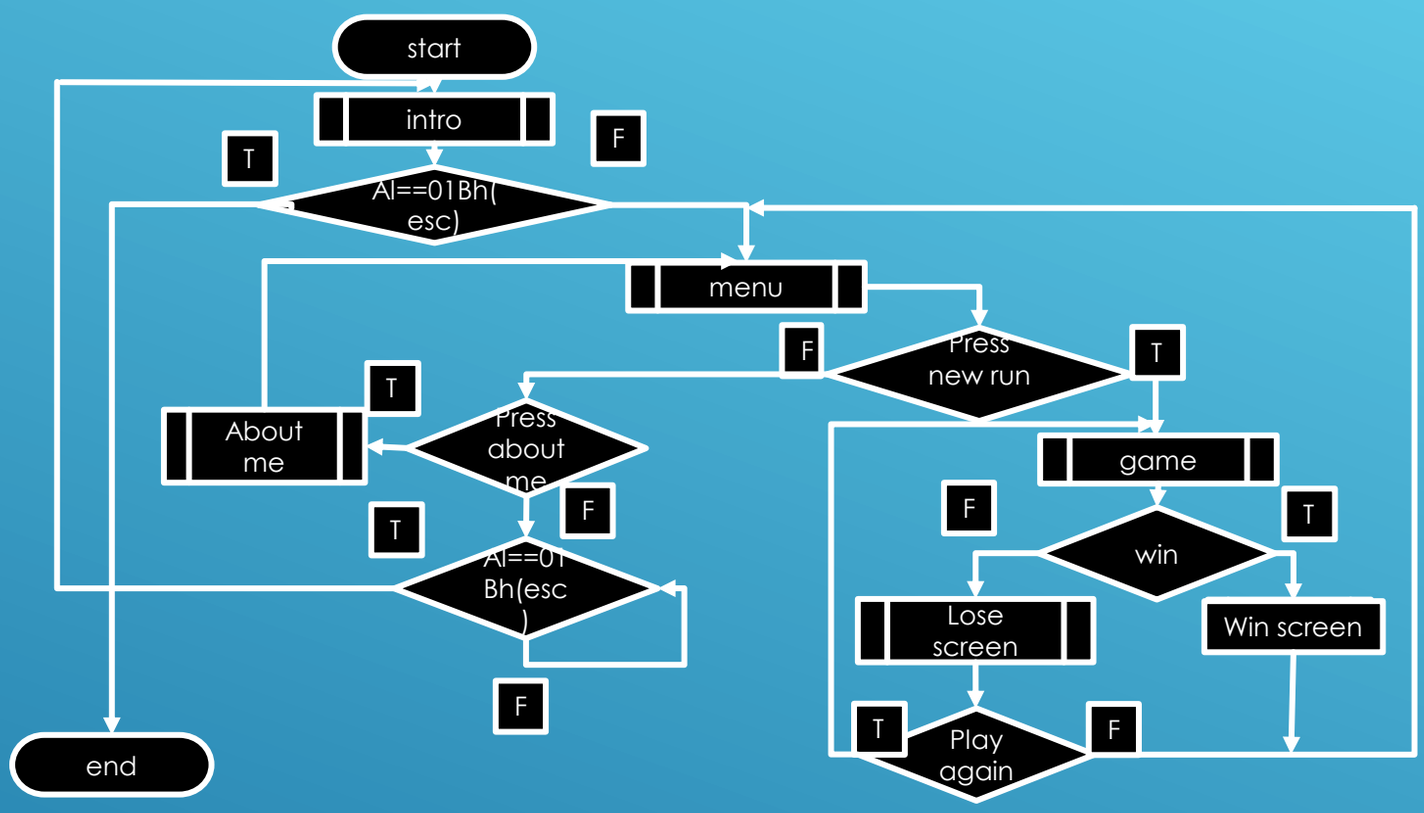
Start/end

commend

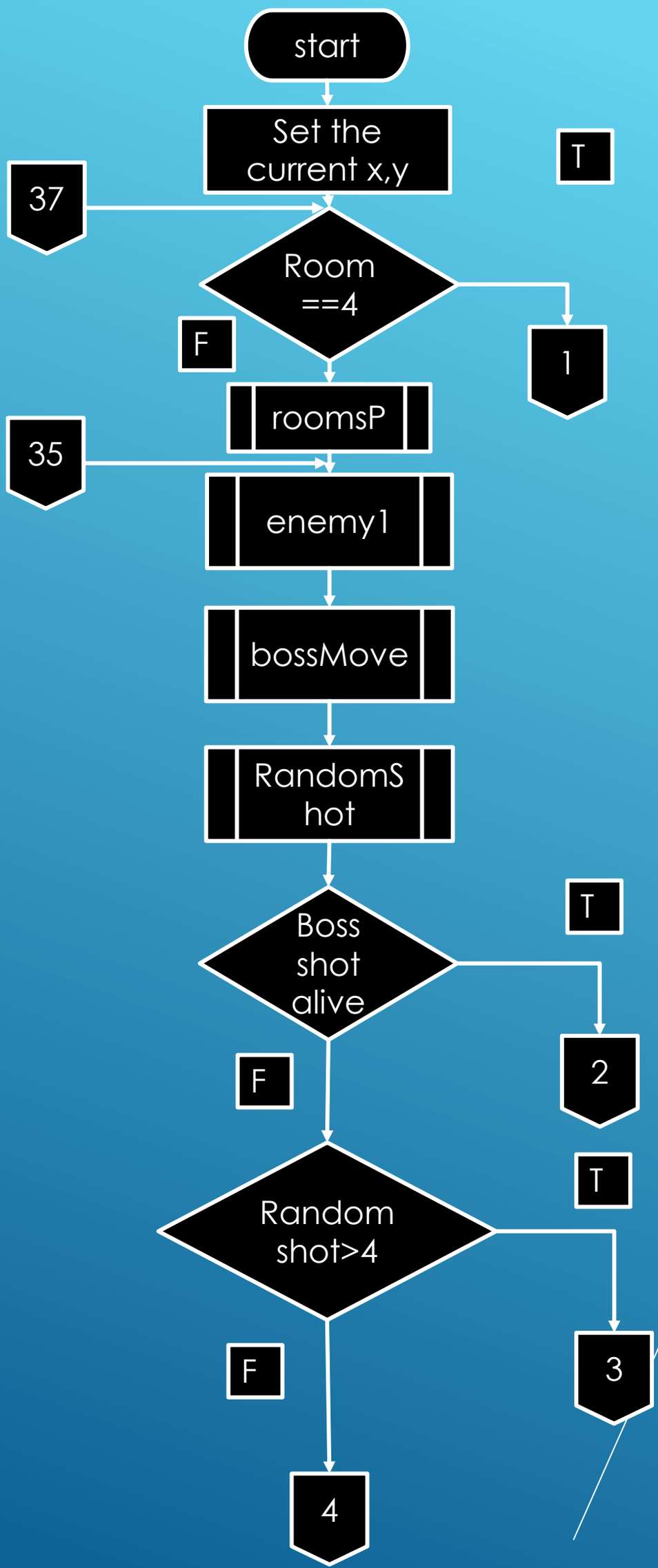
Input/
output

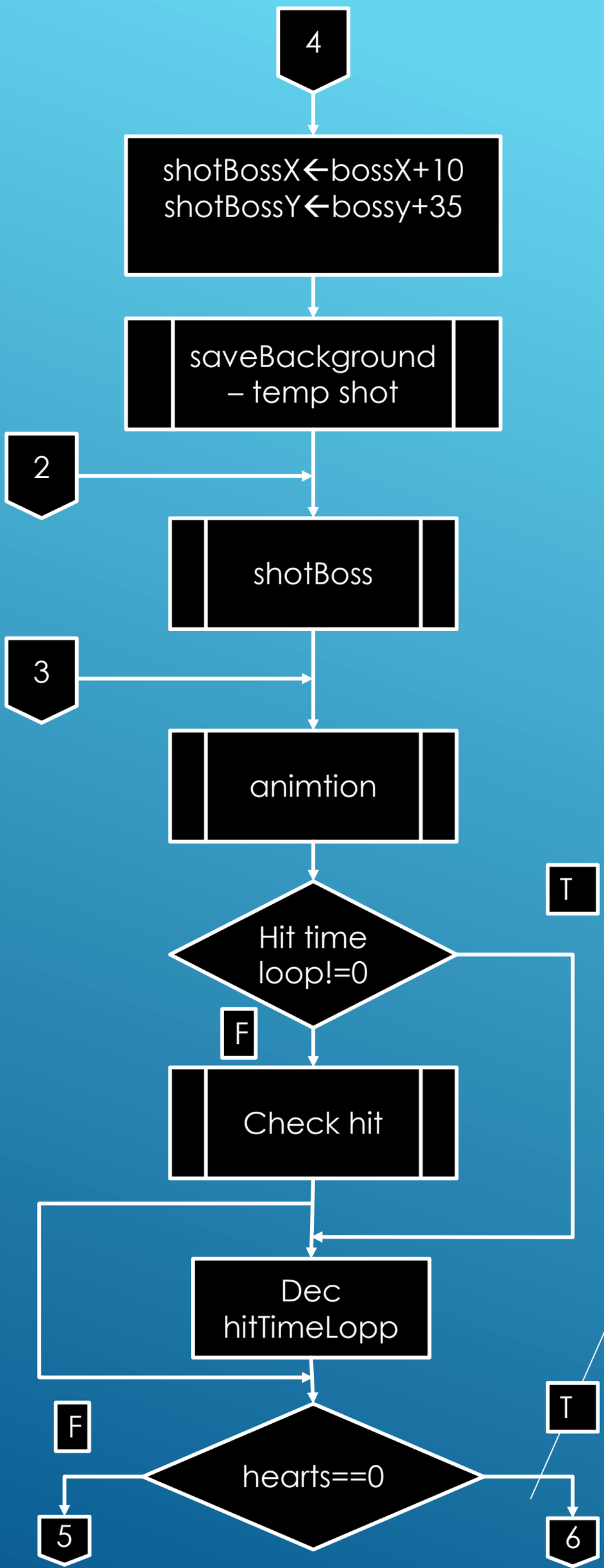
F-false
T-True

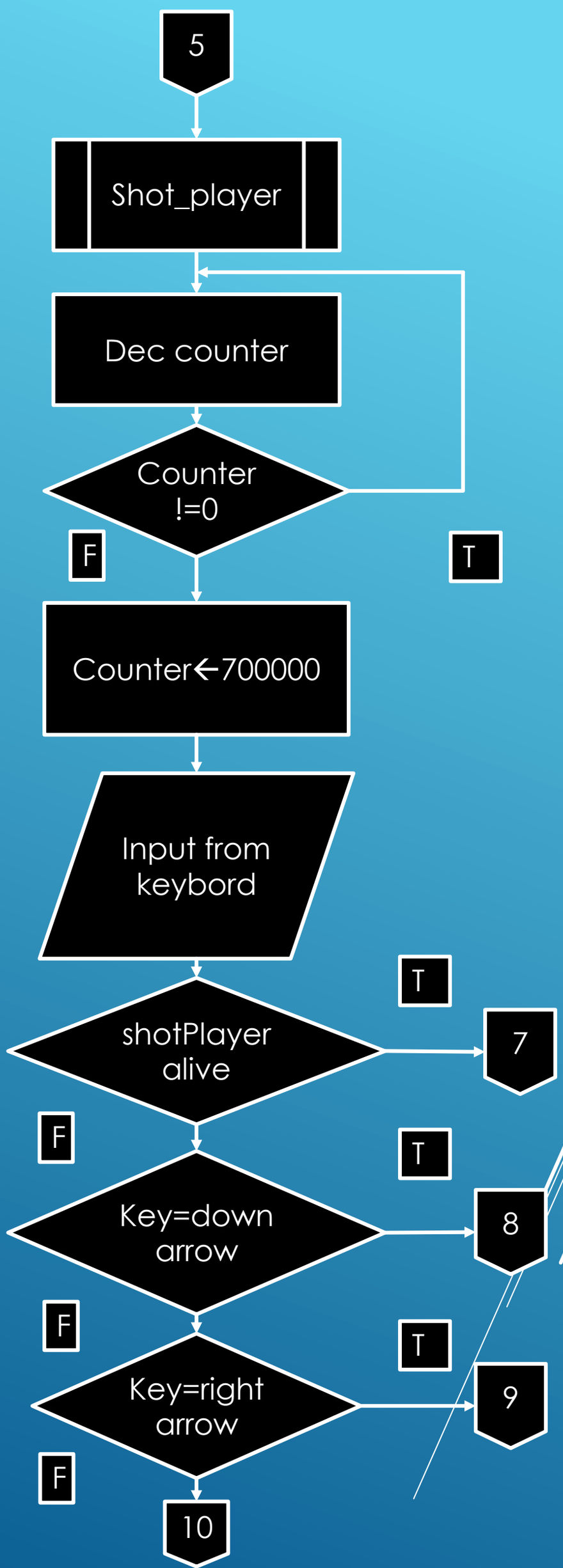
Main locig

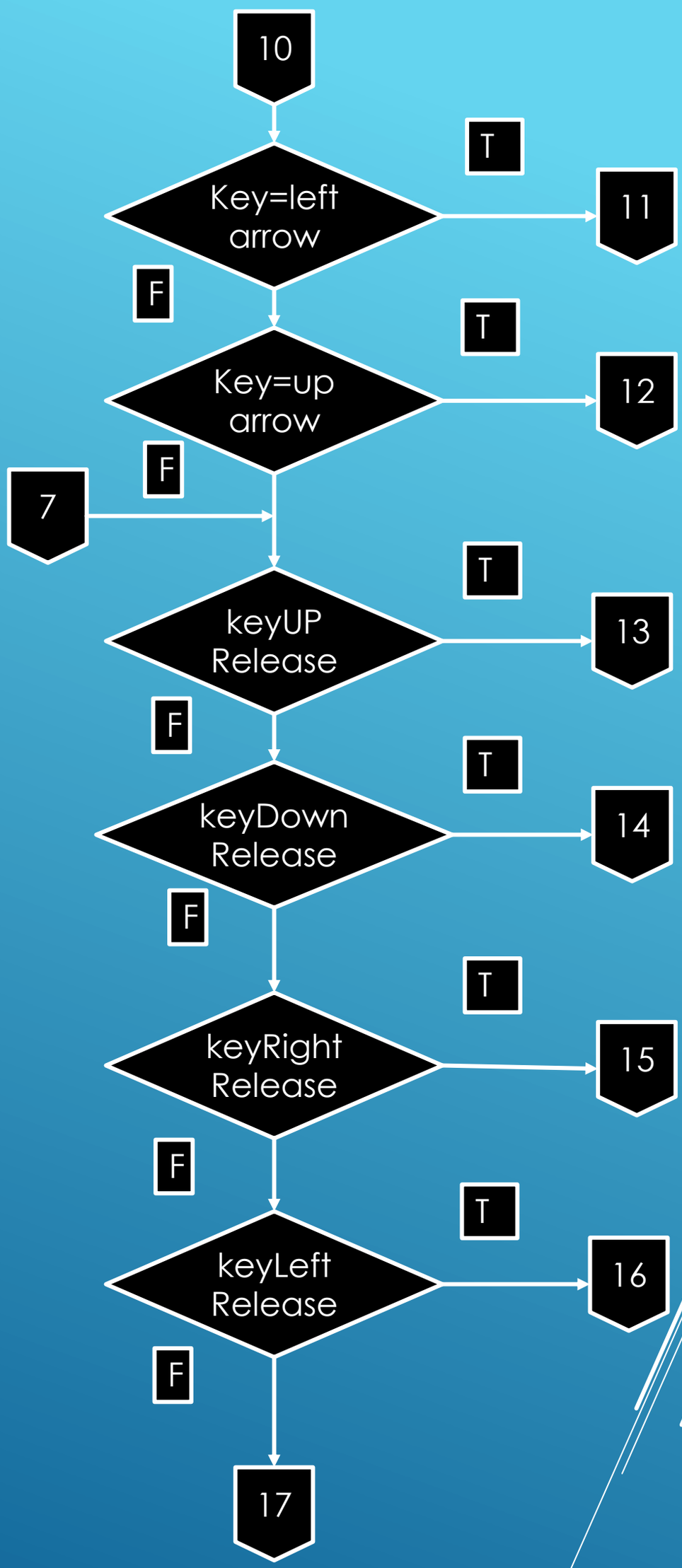

start
intro
T
F
Al==01Bh(esc)
menu
F
Press new run
T
T
About me
Press about me
F
T
Al==01Bh(esc)
F
game
F
T
win
Lose screen
Win screen
T
Play again
F
end

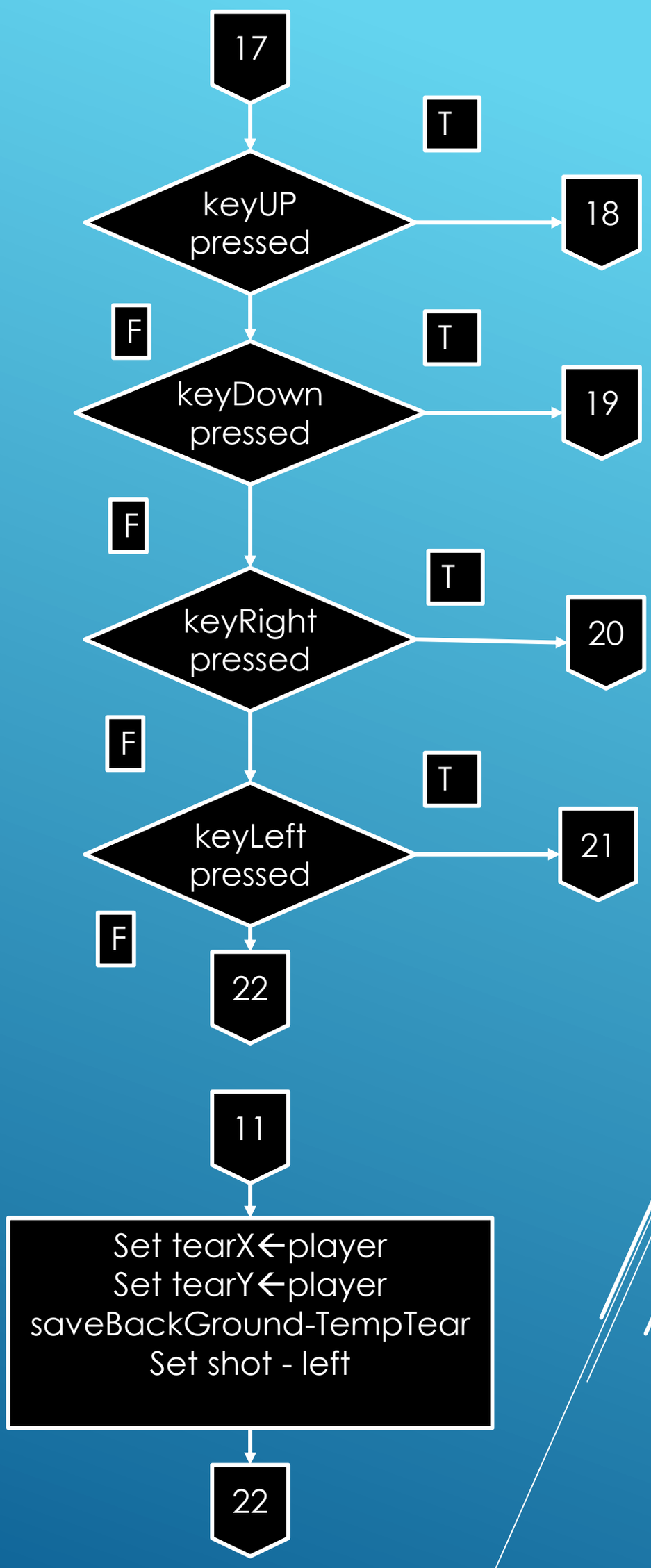
Main function-game

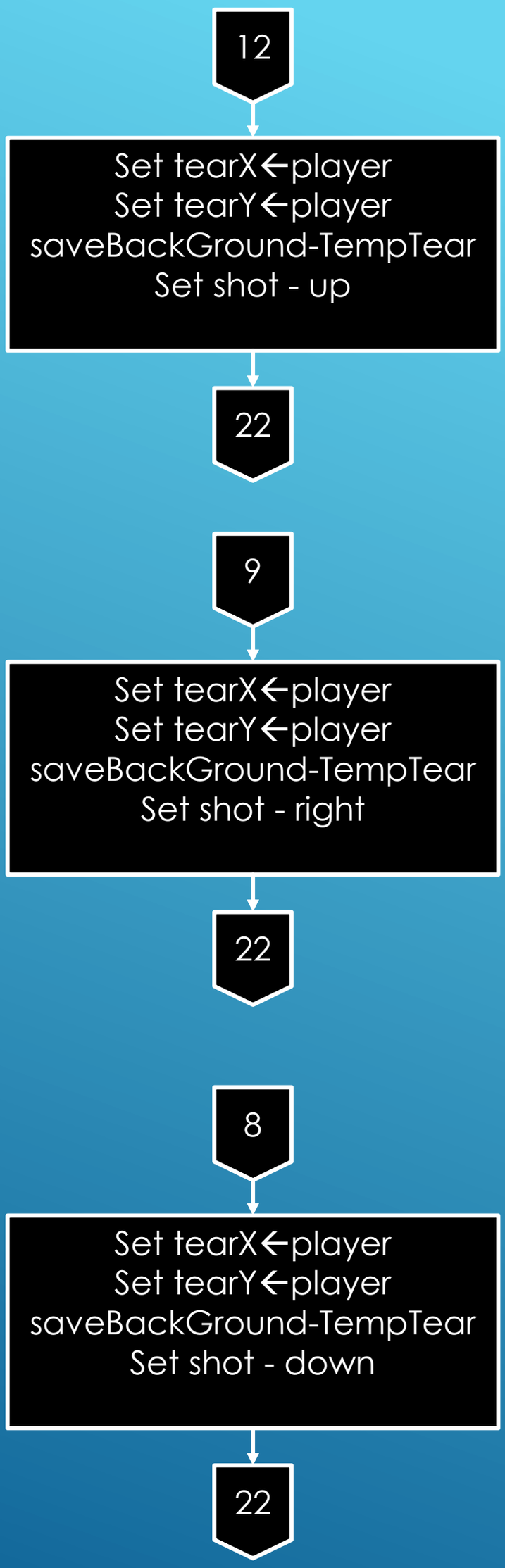


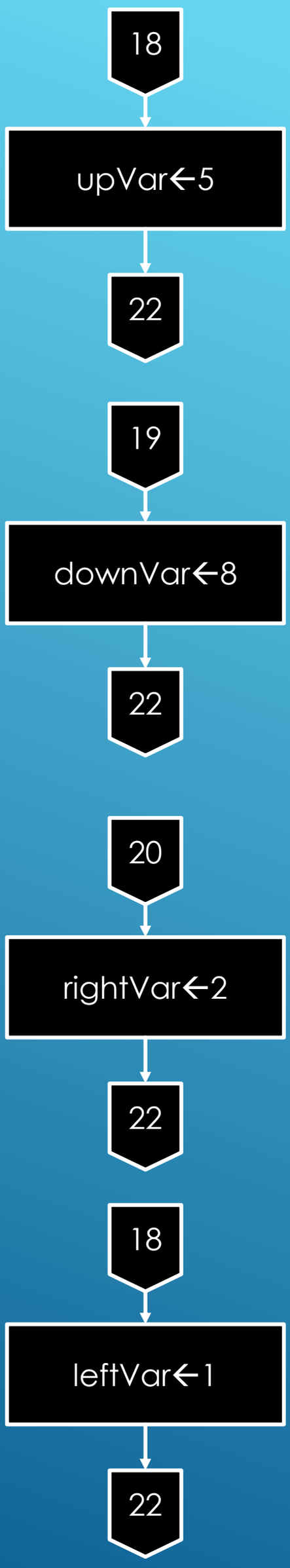

4
shotBossX←bossX+10
shotBossY←bossy+35
saveBackground
– temp shot
2
shotBoss
3
animtion
T
Hit time
loop!=0
F
Check hit
Dec
hitTimeLopp
F
T
hearts==0
5
6

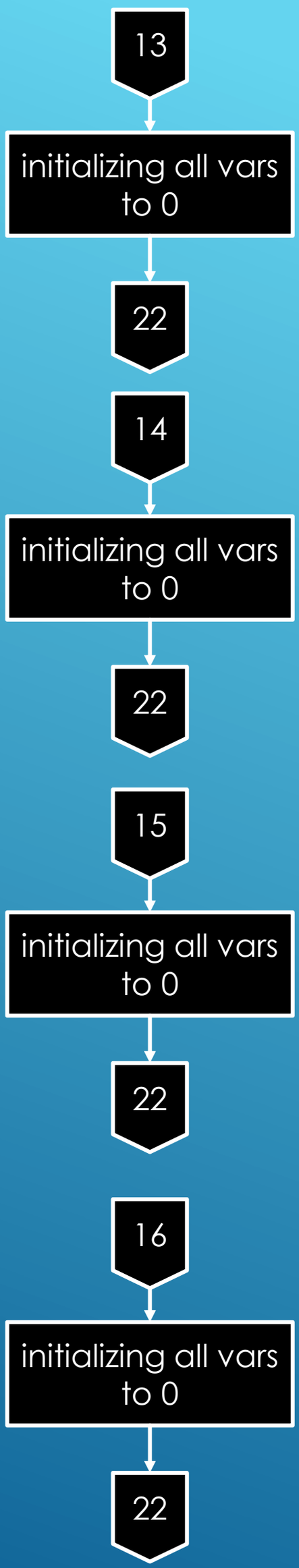

5
Shot_player
Dec counter
Counter
!=0
F
T
Counter←700000
Input from
keybord
T
shotPlayer
alive
7
F
T
Key=down
arrow
8
F
T
Key=right
arrow
9
F
10

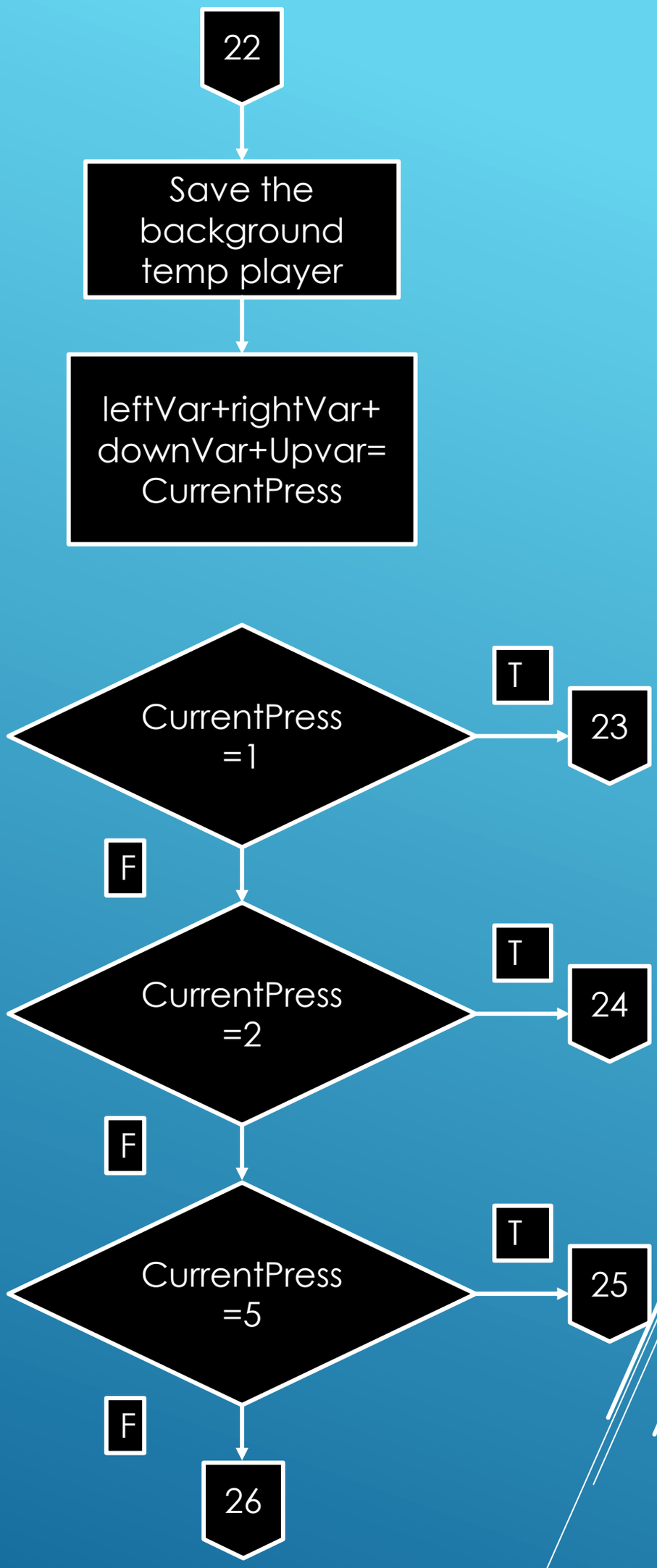

10
Key=left arrow
T
11
F
Key=up arrow
T
12
F
7
keyUP Release
T
13
F
keyDown Release
T
14
F
keyRight Release
T
15
F
keyLeft Release
T
16
F
17

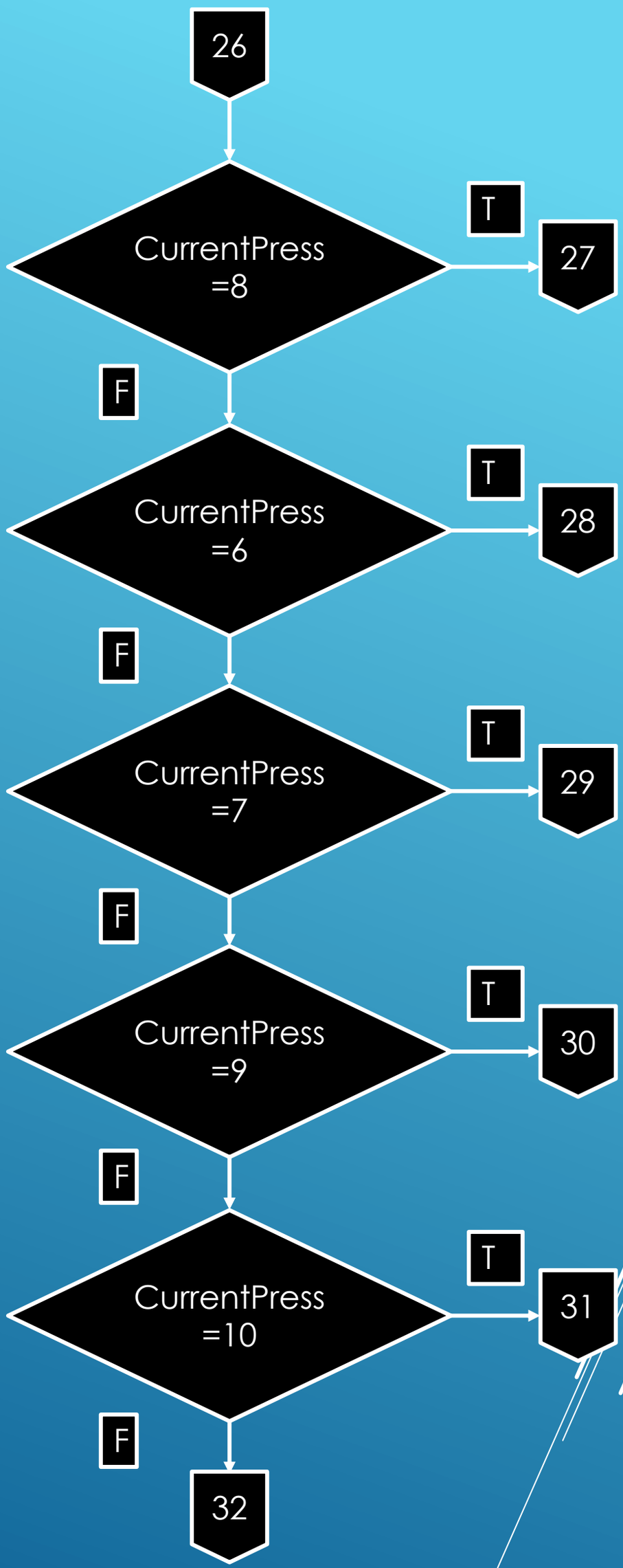

17
T
keyUP pressed
18
F
T
keyDown pressed
19
F
T
keyRight pressed
20
F
T
keyLeft pressed
21
F
22
11
Set tearX←player
Set tearY←player
saveBackGround-TempTear
Set shot - left
22

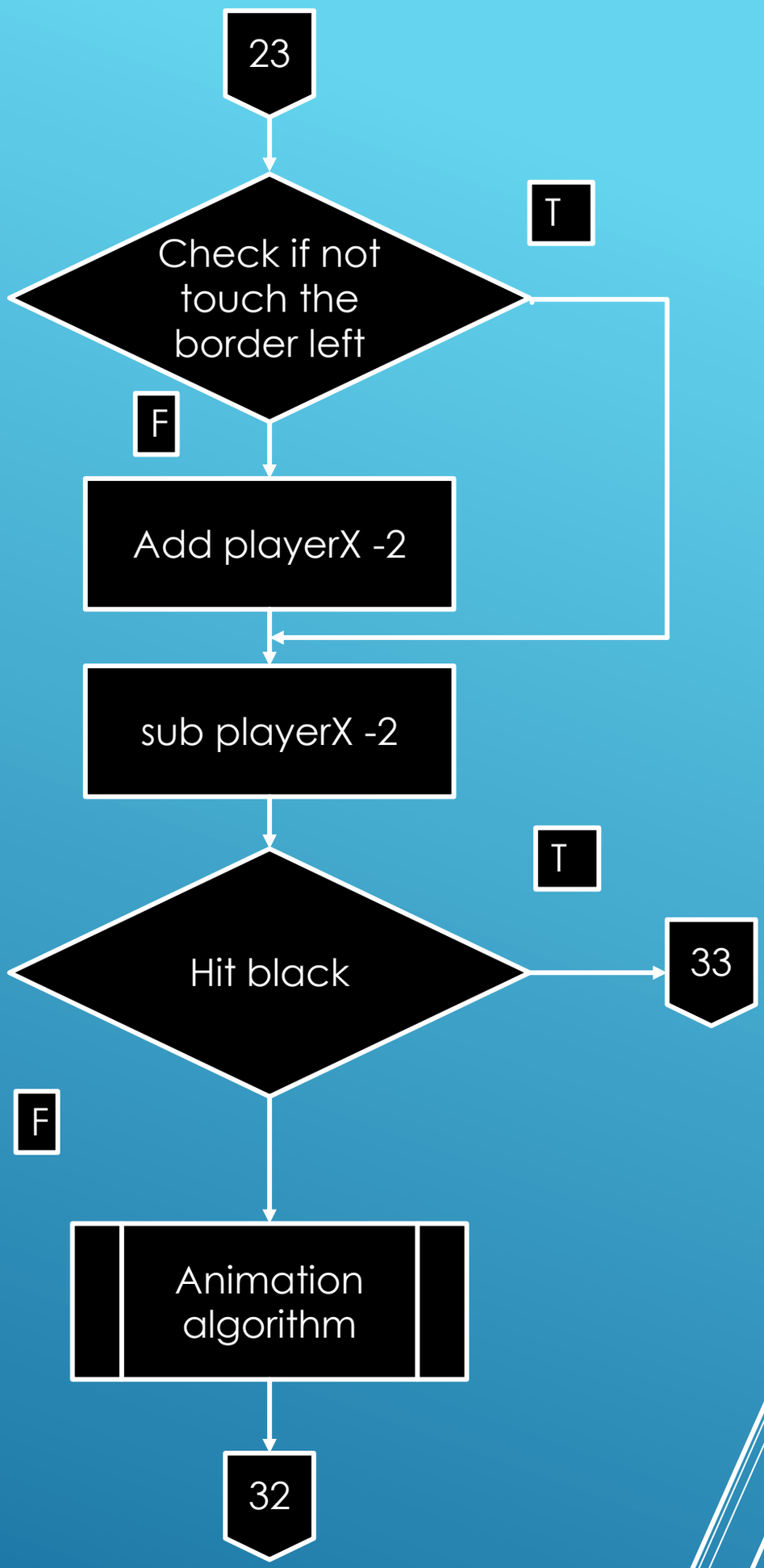

12
Set tearX←player
Set tearY←player
saveBackGround-TempTear
Set shot - up
22
9
Set tearX←player
Set tearY←player
saveBackGround-TempTear
Set shot - right
22
8
Set tearX←player
Set tearY←player
saveBackGround-TempTear
Set shot - down
22

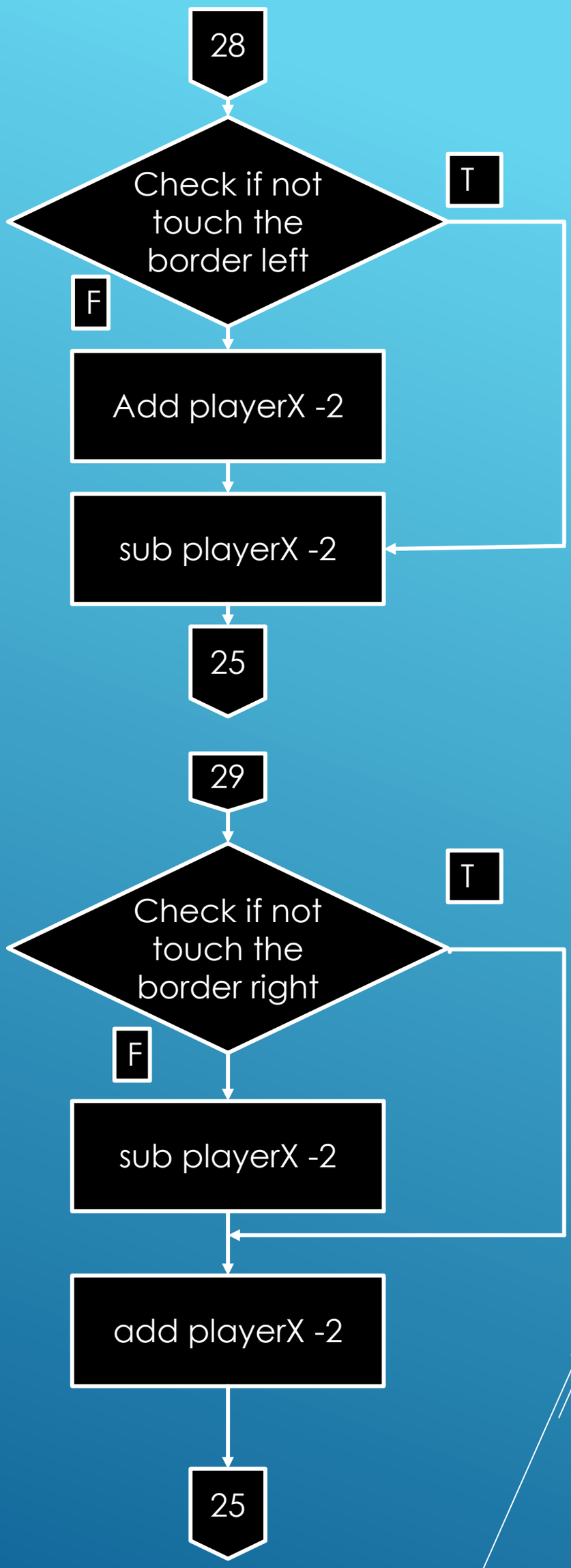

18
upVar←5
22
19
downVar←8
22
20
rightVar←2
22
18
leftVar←1
22

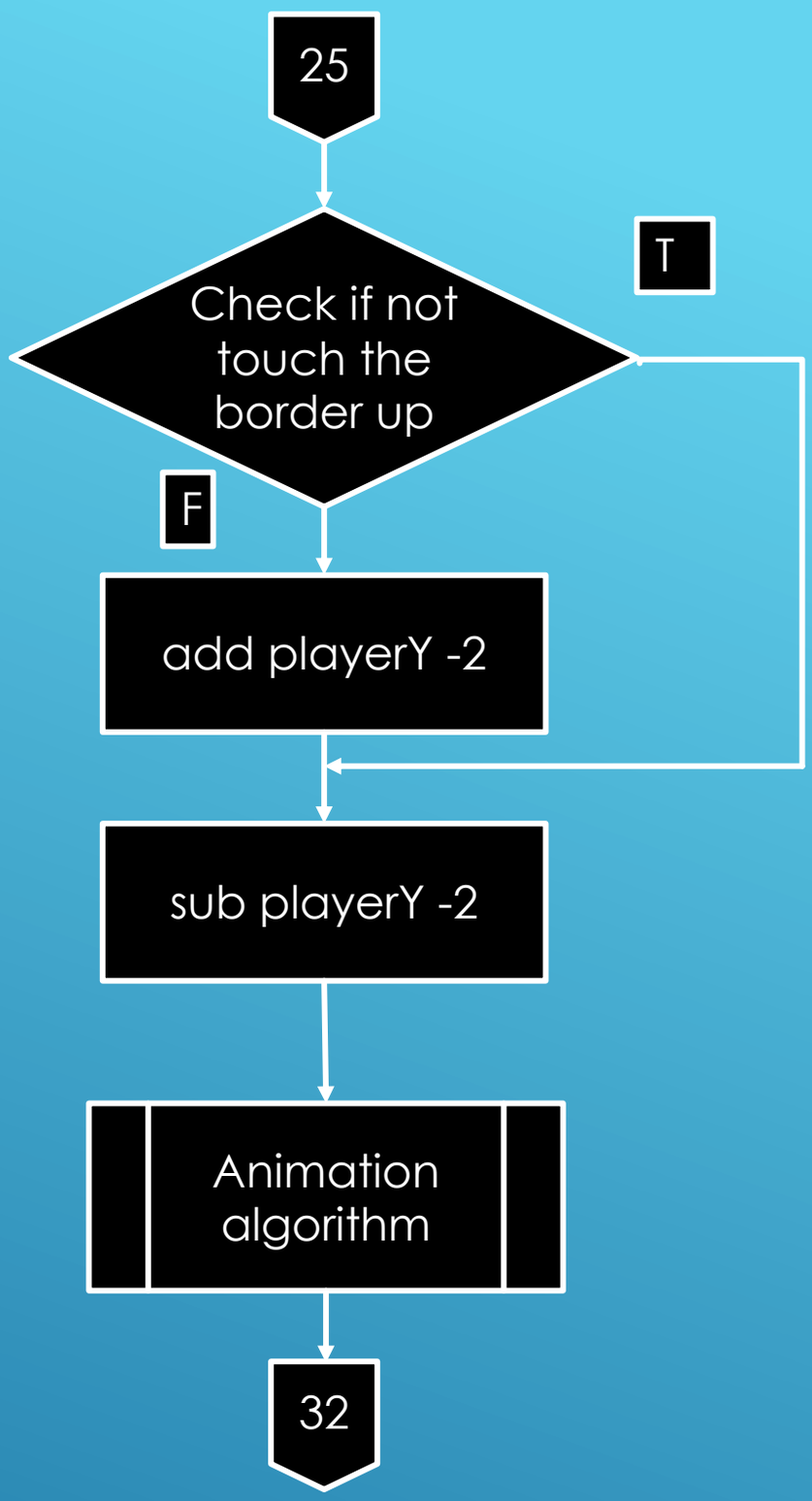

13
initializing all vars to 0
22
14
initializing all vars to 0
22
15
initializing all vars to 0
22
16
initializing all vars to 0
22

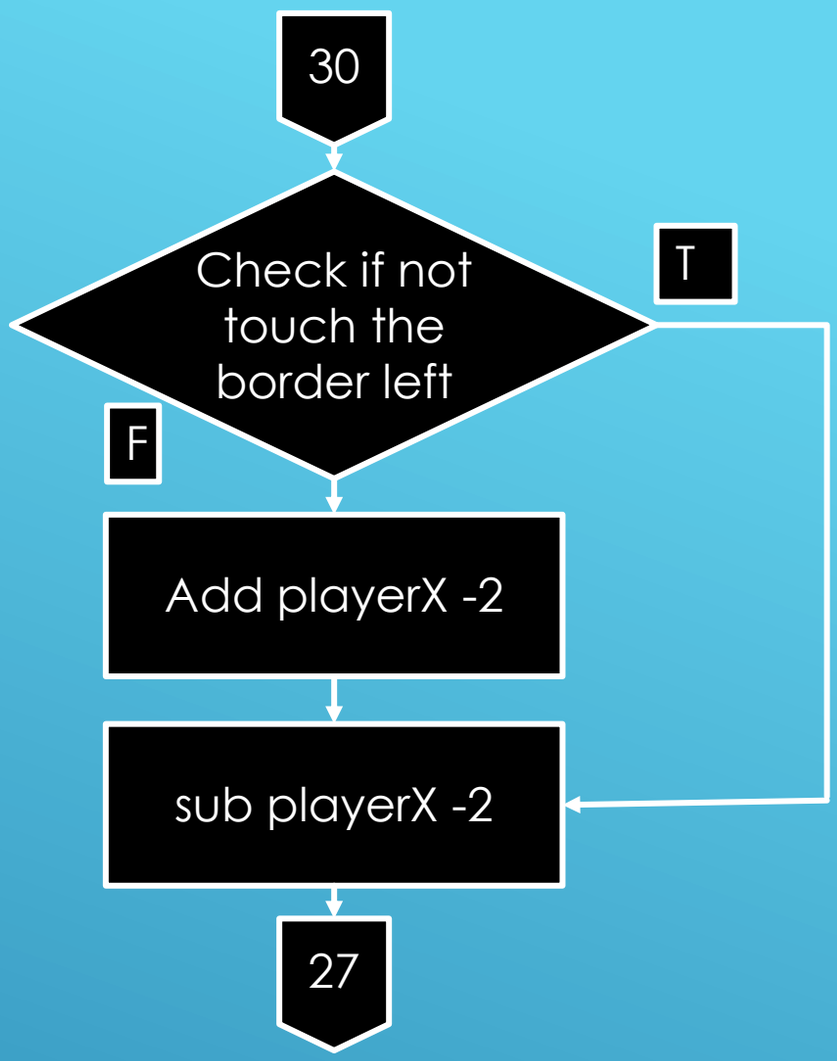

22
Save the background temp player
leftVar+rightVar+downVar+Upvar=CurrentPress
CurrentPress =1
T
23
F
CurrentPress =2
T
24
F
CurrentPress =5
T
25
F
26

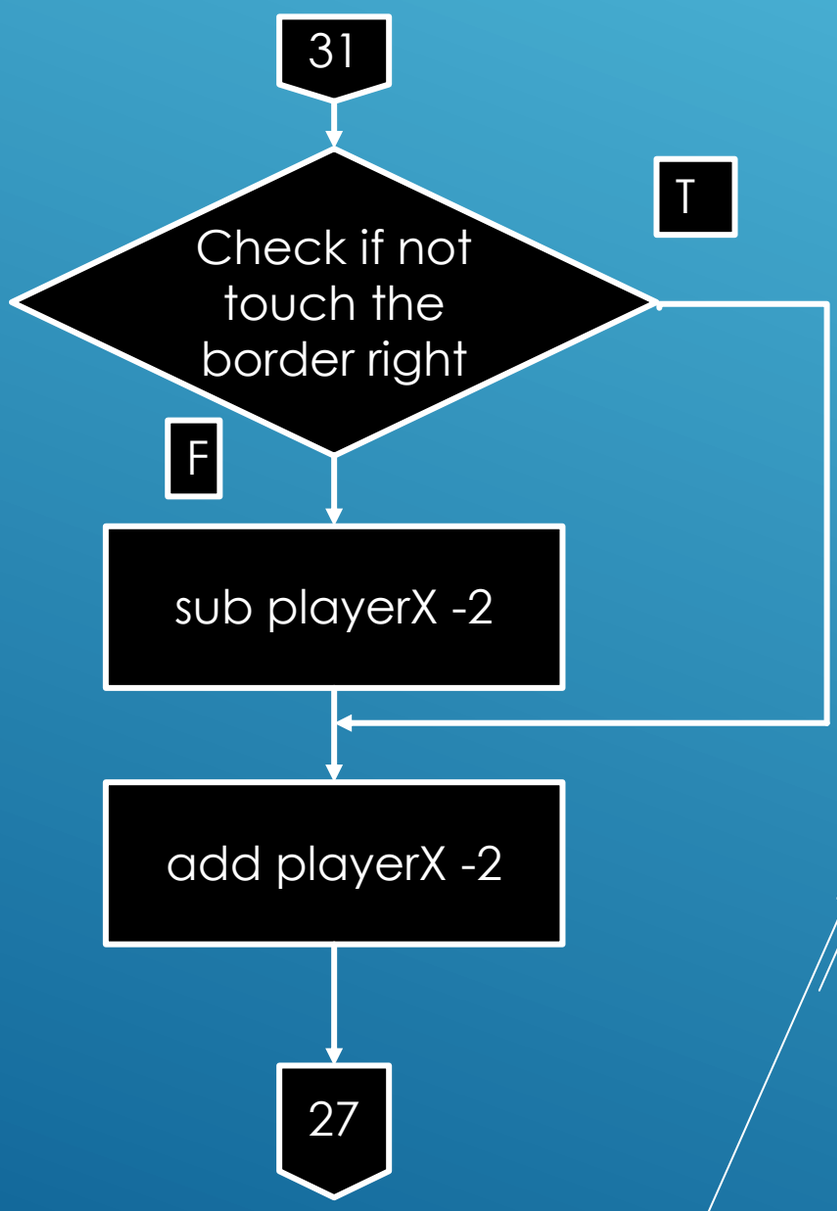

26
CurrentPress =8
T
27
F
CurrentPress =6
T
28
F
CurrentPress =7
T
29
F
CurrentPress =9
T
30
F
CurrentPress =10
T
31
F
32

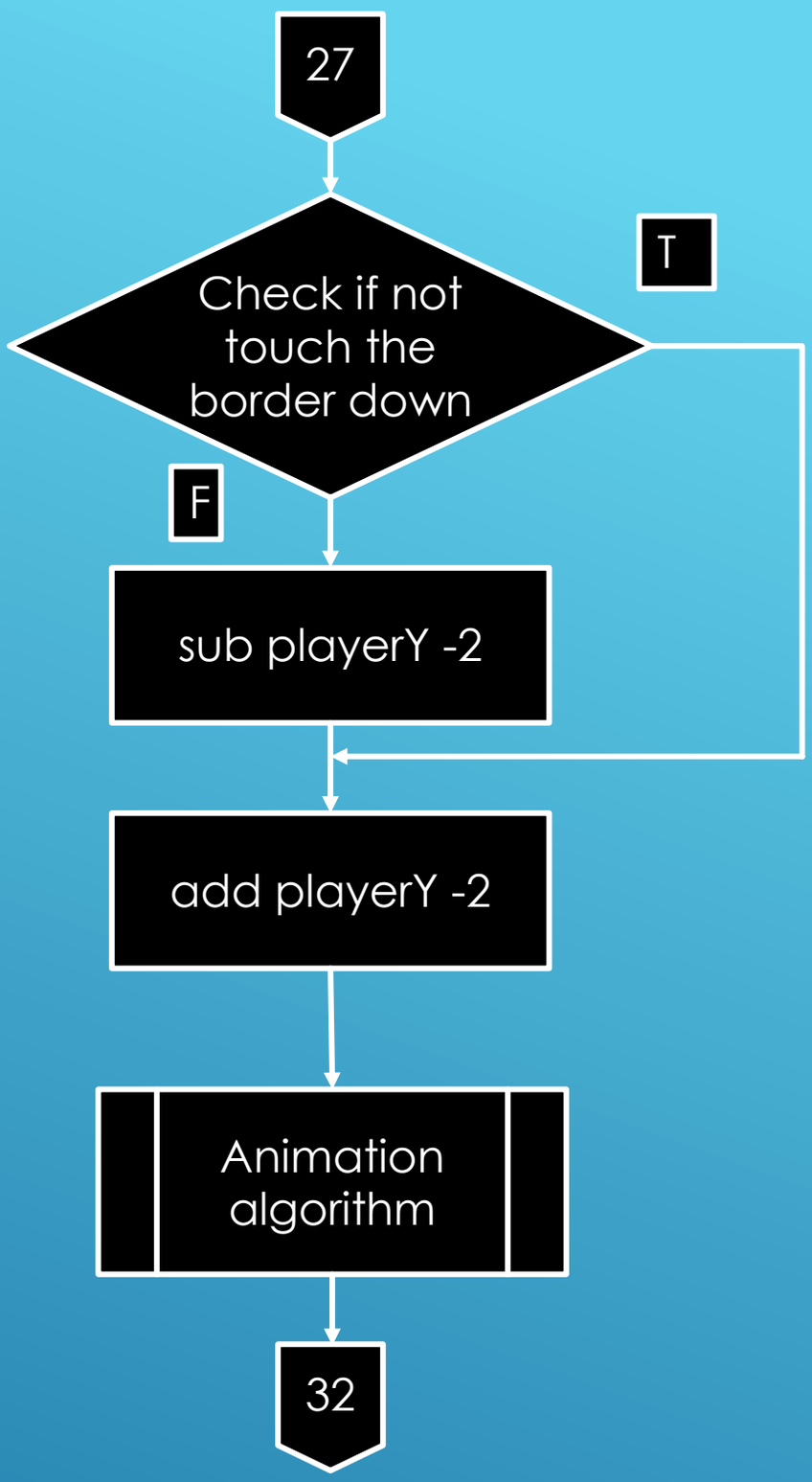

23
Check if not touch the border left
T
F
Add playerX -2
sub playerX -2
T
Hit black
33
F
Animation algorithm
32

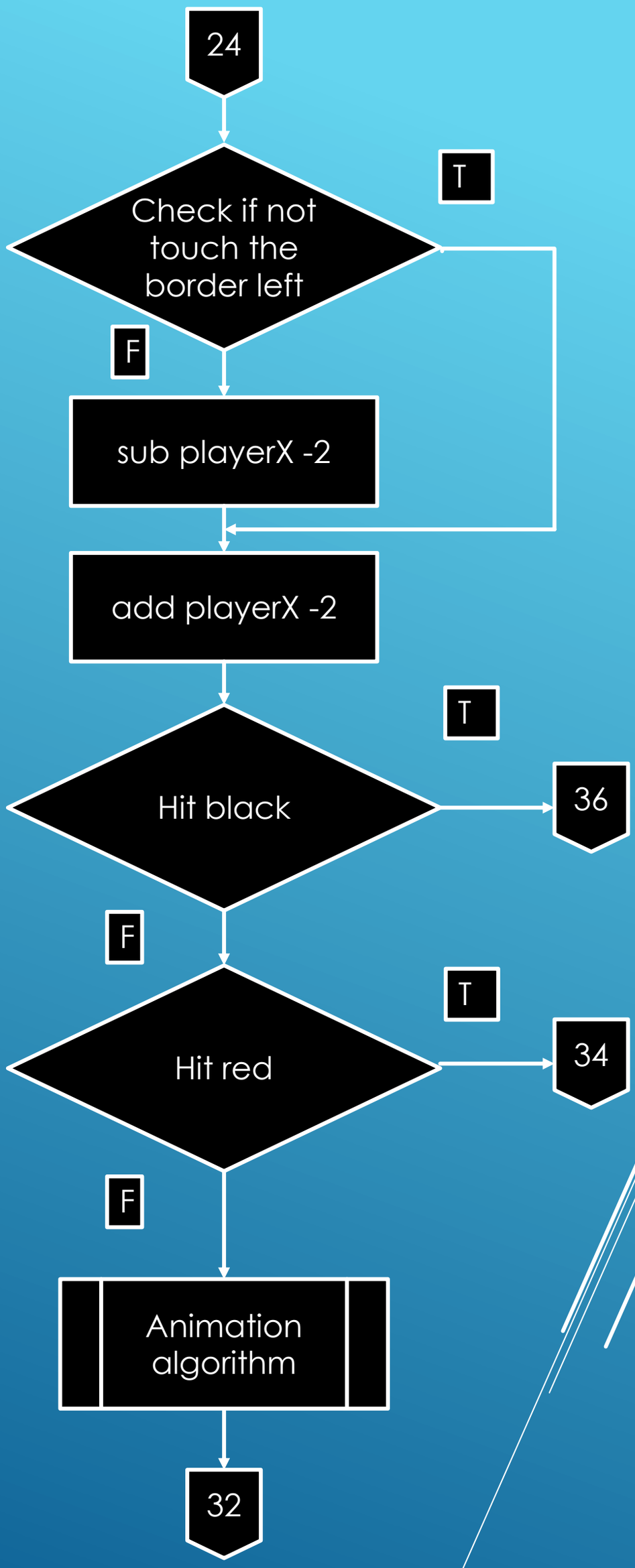

28
Check if not touch the border left
T
F
Add playerX -2
sub playerX -2
25
29
Check if not touch the border right
T
F
sub playerX -2
add playerX -2
25

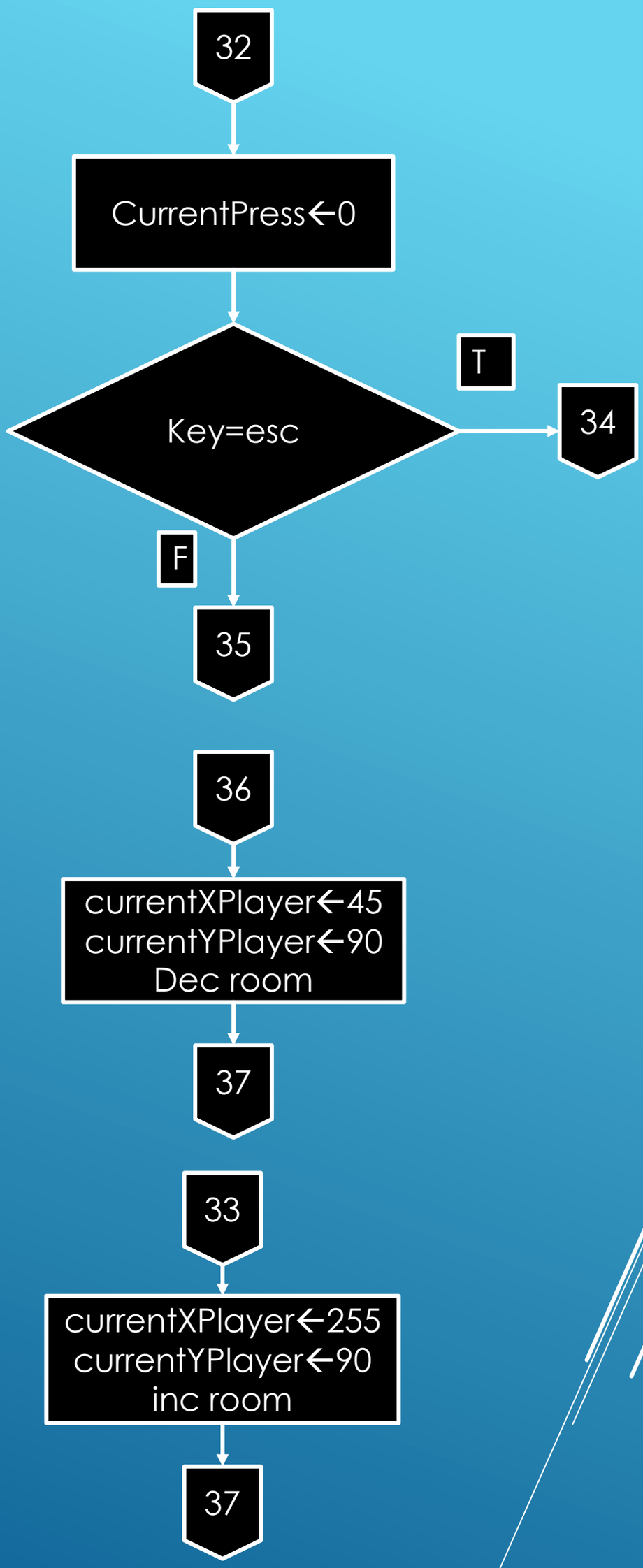

25
Check if not touch the border up
T
F
add playerY -2
sub playerY -2
Animation algorithm
32

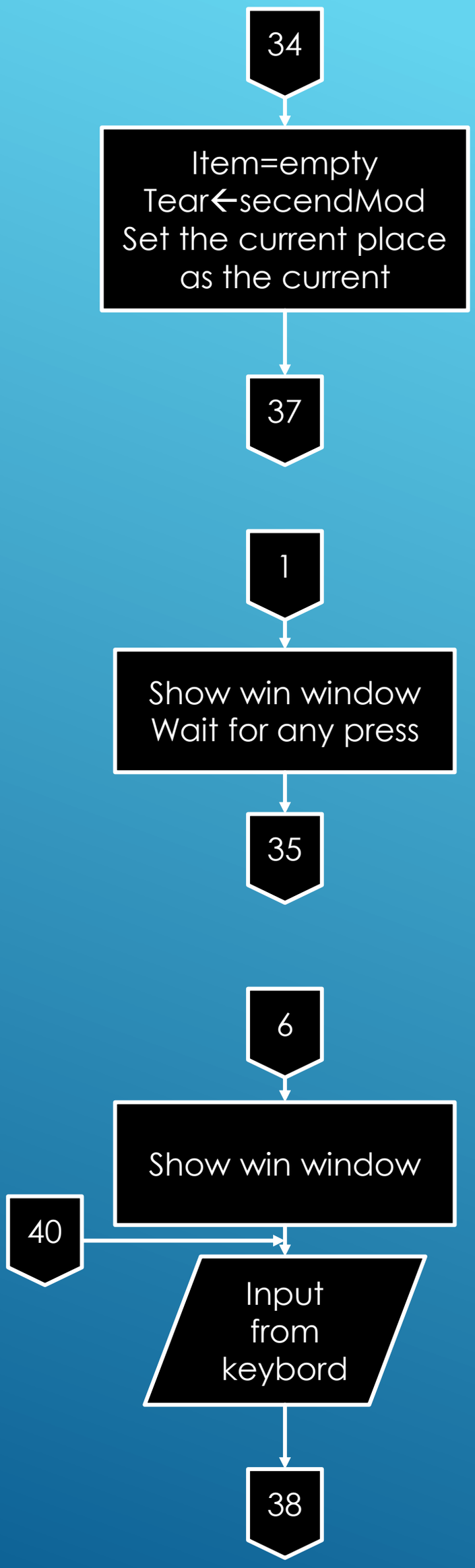

30
Check if not touch the border left
T
F
Add playerX -2
sub playerX -2
27

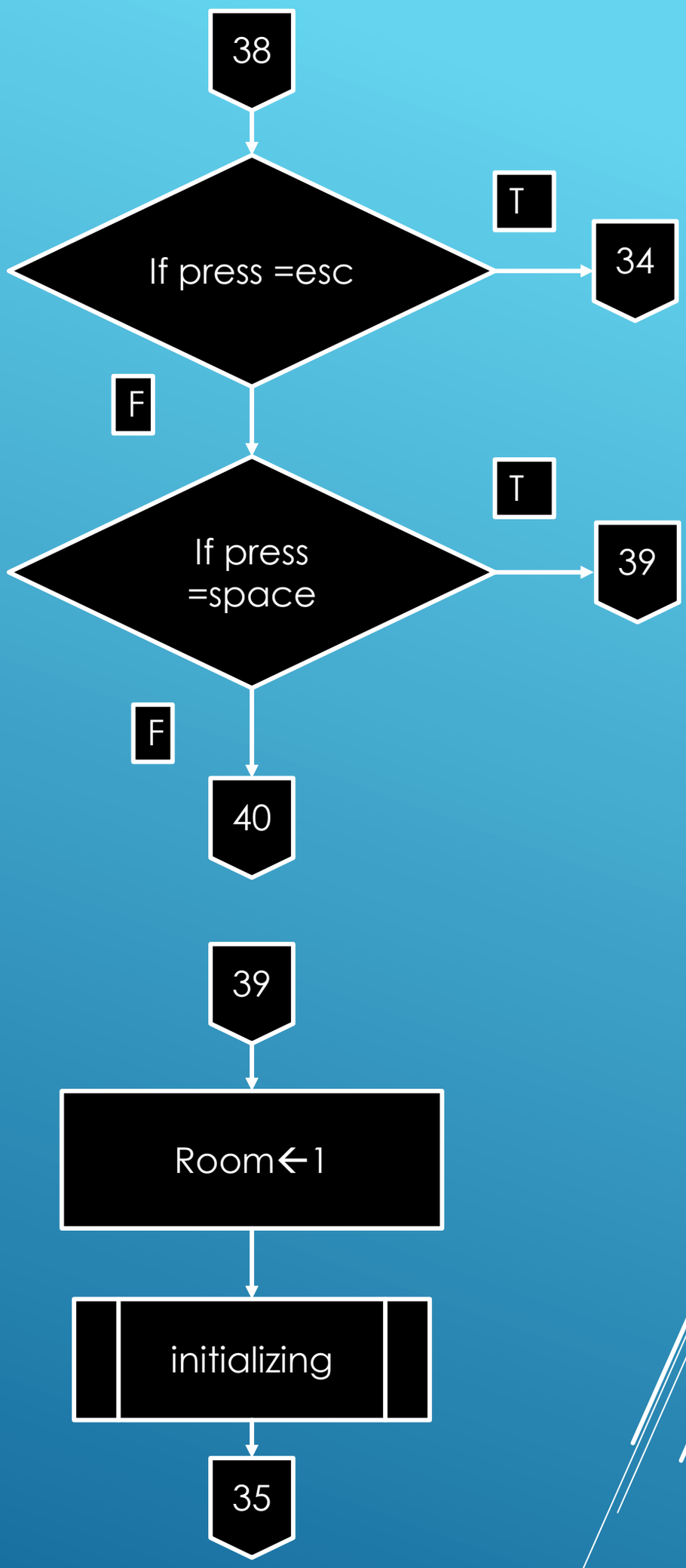

31
Check if not touch the border right
T
F
sub playerX -2
add playerX -2
27

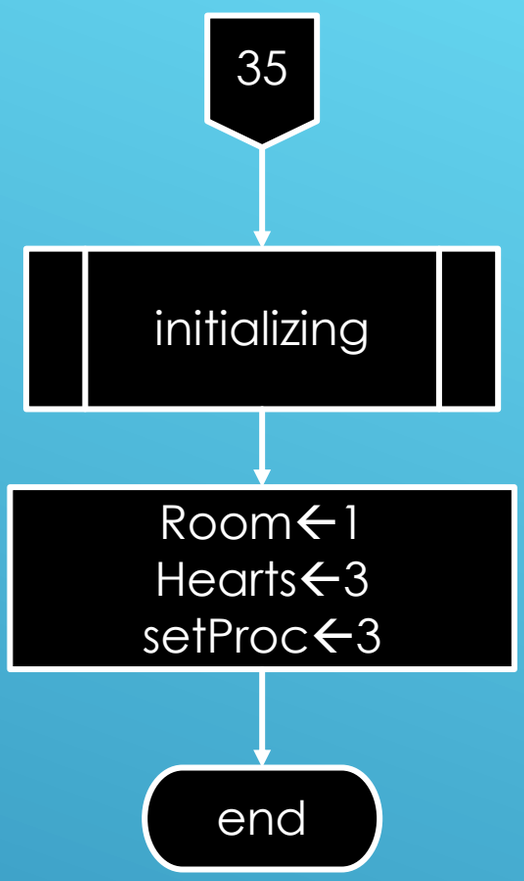

27
Check if not touch the border down
T
F
sub playerY -2
add playerY -2
Animation algorithm
32


24
Check if not touch the border left
T
F
sub playerX -2
add playerX -2
Hit black
T
36
F
Hit red
T
34
F
Animation algorithm
32


32
CurrentPress←0
Key=esc
T
34
F
35
36
currentXPlayer←45
currentYPlayer←90
Dec room
37
33
currentXPlayer←255
currentYPlayer←90
inc room
37


34
Item=empty
Tear←secendMod
Set the current place
as the current
37
1
Show win window
Wait for any press
35
6
Show win window
40
Input
from
keybord
38


38
If press =esc
T
34
F
If press
=space
T
39
F
40
39
Room←1
initializing
35


35
initializing
Room←1
Hearts←3
setProc←3
end

FUNCTIONS AND MACROES

```
;-----------------------------------------------------------
; SHOWPCX – set the photo (macro)
;-----------------------------------------------------------
; Input:
;   StartX, StartY - Image location
;   file name to show as pcx photo
; Output:
;   None
; Registers
;   AX, Dx
;-------------------------------------------------------
;-----------------------------------------------------------
; MDrawImage - set the bitmap (macro)
;-----------------------------------------------------------
; Input:
;   StartX, StartY - bitMap Image location
;   BitMap img name to show the photo
; Output:
;   None
; Registers
;   AX, si
;-------------------------------------------------------
;-----------------------------------------------------------
; MSaveBkGround – set variables to save the background
;-----------------------------------------------------------
; Input:
;   StartX, StartY - Image location
;   temp_array     - the array for save a back ground
; Output:
;   None
; Registers
;   AX, SI
;-------------------------------------------------------
```

```
;-----------------------------------------------------------
; Hearts_view – show the hearts(macro)
;-----------------------------------------------------------
; Input:
;    hearts variable
; Output:
;   output the mod of the Hearts
; Registers
;   AX, Dx
;-------------------------------------------------------
;-------------------------------------------------------
; introP - show intro
;-------------------------------------------------------
; Input:
;      any key/esc
; Output:
;      print the intro screen
; Registers
;       ax
;-------------------------------------------------------
;-------------------------------------------------------
; menuP - MainMenu
;-------------------------------------------------------
; Input:
;      key up/ key down / esc / enter
; Output:
;      print menu screens
; Registers
;       ax
;-------------------------------------------------------
;-----------------------------------------
; gameP – the main function
;-----------------------------------------
; Input:
;      keys, RandomShotB, inshotB,inshotE
; Output:
;      from this function we call and show the diffrents rooms
;      the tears (bullets)
; Registers
;       ax,cx,bx,dx
;-----------------------------------------
```

```
;-----------------------------------------
; DrawRoom1
;-----------------------------------------
; Input:
;      none
; Output:
;      room 1, hearts, map
; Registers
;       none
;-----------------------------------------
;-----------------------------------------
; DrawRoom3C - boos room Close
;-----------------------------------------
; Input:
;      none
; Output:
;      room 3 close, hearts, map
; Registers
;       none
;-----------------------------------------
;-----------------------------------------
; DrawRoom3O - boos room open
;-----------------------------------------
; Input:
;      none
; Output:
;      room 3 open, hearts, map
; Registers
;       none
;-----------------------------------------
;-----------------------------------------
; DrawRoom2 open
;-----------------------------------------
; Input:
;      none
; Output:
;      room 2 open, hearts, map
; Registers
;       none
;-----------------------------------------
```

```
;-----------------------------------------
; DrawRoom2 Colse
;-----------------------------------------
; Input:
;      none
; Output:
;      room 2 close, hearts, map
; Registers
;       none
;-----------------------------------------
;-----------------------------------------
; DrawRoom0_withItem
;-----------------------------------------
; Input:
;      none
; Output:
;      room 0 with item, hearts, map
; Registers
;       none
;-----------------------------------------
;-----------------------------------------
; DrawRoom0_withoutItem
;-----------------------------------------
; Input:
;      none
; Output:
;       room 0 without item, hearts, map
; Registers
;       none
;-----------------------------------------
;-----------------------------------------
; aboutMeP - about me
;-----------------------------------------
; Input:
;      esc
; Output:
;      about me
; Registers
;       AX
;-----------------------------------------
```

```
; drawIMG - draw bit map image
;-----------------------------------------
; Input:
;      none
; Output:
;      bitMap img pixel after pixel
; Registers
;       AX,cx,si,bh
;-----------------------------------------
;-----------------------------------------
; ReadPCXFile - read PCX file into FILEBUF
;-----------------------------------------
; Input:
;      File name
; Output:
;      File into FILEBUF
; Registers
;       AX, BX, CX, DX, DS
;-----------------------------------------
;-----------------------------------------
; PSaveBkGround - save the backround
;-----------------------------------------
; Input:
;      File name
; Output:
;      The file
; Registers
;       AX, BX, CX, DX, DS
;-----------------------------------------
;-----------------------------------------
; ShowPCXFile - show PCX file
;-----------------------------------------
; Input:
;      File name
; Output:
;      The file
; Registers
;       AX, BX, CX, DX, DS
;-----------------------------------------
```

```
;-----------------------------------------
; PutPixel - draw pixel
;-----------------------------------------
; Input:
;      x - Point_x, y - Point_y, Color - color
; Output:
;      The pixel
; Registers
;       AX, BH, CX, DX
;-----------------------------------------
;-----------------------------------------
; PClearScreen
;-----------------------------------------
; Input:
;      none
; Output:
;      The pixel
; Registers
;       ax,es,cx,di
;-----------------------------------------
;-----------------------------------------
; animate
;-----------------------------------------
; Input:
;      pic_aniMode
; Output:
;      the player in diffrent antimtion
; Registers
;       none
;-----------------------------------------
;-----------------------------------------
; roomsP
;-----------------------------------------
; Input:
;      room, current_x_player, current_y_player
; Output:
;      the diffrent rooms
; Registers
;       none
;-----------------------------------------
```

```
;-----------------------------------------
; shot proc
;-----------------------------------------
; Input:
;      shootB - the current shoot
; Output:
;      shot animtion
; Registers
;       ax
;-----------------------------------------
;-----------------------------------------
; TearD
;-----------------------------------------
; Input:
;      tearMode
; Output:
;      draw the tears
; Registers
;       none
;-----------------------------------------
;-----------------------------------------
; enemy
;-----------------------------------------
; Input:
;      enemyExist,room,randomVar
; Output:
;       draw the enemy with anitmion
; Registers
;       none
;-----------------------------------------
;-----------------------------------------
; dash
;-----------------------------------------
; Input:
;      randomVar
; Output:
;      set the new intrupt adress
; Registers
;       ax,ds,dx
;-----------------------------------------
```

```
;-----------------------------------------
; Random - random number 0-3
;-----------------------------------------
; Input:
;       none
; Output:
;       random number for enemy
; Registers
;        ax
;-----------------------------------------
;-----------------------------------------
; RandomShot - random number 0-3
;-----------------------------------------
; Input:
;       none
; Output:
;       random number for boss
; Registers
;        ax
;-----------------------------------------
;-----------------------------------------
; readPixel
;-----------------------------------------
; Input:
;       ReadPixel_X,ReadPixel_Y
; Output:
;       al=the color
; Registers
;        ax,ds,dx
;-----------------------------------------
```

```
;-----------------------------------------
; checkHit
;-----------------------------------------
; Input:
;      StartPictX,StartPictY
; Output:
;      none
; Registers
;       ax,ds,dx
;-----------------------------------------
;-----------------------------------------
; intalazing
;-----------------------------------------
; Input:
;      none
; Output:
; intalazing the variables
; Registers
;       ax,ds,dx
;-----------------------------------------
;-----------------------------------------
; boosMove
;-----------------------------------------
; Input:
;      bossLife,bossExist
; Output:
;      the boss in new postion
; Registers
;       none
;-----------------------------------------
;-----------------------------------------
; shotBoss - show the lazer shot
;-----------------------------------------
; Input:
;      none
; Output:
;      shot of the boss
; Registers
;      none
;-----------------------------------------
```

```
;-----------------------------------------
; clearParticals - clear all the Scraps from the screen by
; draw the pcx photo of the room
;-----------------------------------------
; Input:
;       none
; Output:
;       clear the scraps
; Registers
;       ax
;-----------------------------------------
;-----------------------------------------
; beep_M
;-----------------------------------------
; Input:
;       none
; Output:
;       output beep
; Registers
;       ax
;-----------------------------------------
;-----------------------------------------
; beep_M2
;-----------------------------------------
; Input:
;       none
; Output:
;       output beep
; Registers
;       ax
;-----------------------------------------
```

PROJECT CODE

The next pages contains the code of the game.

```
IDEAL

        MODEL large
P386
;-----------------------------------------------------------
; SHOWPCX
;-----------------------------------------------------------
; Input:
;   StartX, StartY - Image location
;   file name to show as pcx photo
; Output:
;   None
; Registers
;   AX, Dx
;-----------------------------------------------------------
MACRO SHOWPCX StartX, StartY, fName
        mov ax, [StartX]
        mov [Point_X], ax
        mov ax, [StartY]
        mov [Point_Y], ax


        mov dx, offset fName

        call ShowPCXFile
ENDM SHOWPCX
;-----------------------------------------------------------
; MDrawImage
;-----------------------------------------------------------
; Input:
;   StartX, StartY - bitMap Image location
;   BitMap img name to show the photo
; Output:
;   None
; Registers
;   AX, si
;-----------------------------------------------------------
MACRO MDrawImage StartX, StartY, ImgName
    mov ax, [StartX]
    mov [current_X], ax
    mov ax, [StartY]
    mov [current_Y], ax


    mov si, offset ImgName
    call PDrawImage
ENDM MDrawImage
```

```
STACK 256

RIGHT_KEY                   equ 20h
LEFT_KEY                    equ 1Eh
UP_KEY                      equ 11h
DOWN_KEY                    equ 1Fh

RIGHT_UP                    equ 10100000b
LEFT_UP                     equ 10011110b
UP_UP                       equ 10010001b
DOWN_UP                     equ 10011111b

RIGHT_KEY_arrow     equ 77
LEFT_KEY_arrow      equ 75
UP_KEY_arrow        equ 72
DOWN_KEY_arrow      equ 80

RIGHT_UP_arrow      equ 11001101b
LEFT_UP_arrow               equ 11001011b
UP_UP_arrow                 equ 11001000b
DOWN_UP_arrow               equ 11010000b

ESC_KEY                     equ 1
VGA_SEGMENT                 equ 0a000h
TRANSPORENT_COLOR           equ 00
enemySpeed                  equ 3


DATASEG
include     'bitMap.dat'

ErrorReadingFile DB 'Can not open file$'

int_file        DB 'intro.pcx',0
menu1_file          DB 'm1_p.pcx',0
        menu2_file          DB 'm2_p.pcx',0
        room1                       db 'room1.pcx',0
        ab_file                     db 'ab.pcx',0 ; about me file
        itemRoomWi                  db 'itemRwi.pcx',0
        itemRoomi               db 'itemRi.pcx',0
        enemyRoomC                  db 'ERC.pcx',0
        enemyRoomO                  db 'ERO.pcx',0
        boosRoomC               db 'BRC.pcx',0
        boosRoomO               db 'BRO.pcx',0
        loseWindow              db 'loseW.pcx',0
        winWindow               db 'win.pcx',0

FileName        DW ?    ; offset file name for current file

    timerSeg    dw ?
timerOfs    dw ?
```

```
;-----------------------------------------------------------
; MSaveBkGround
;-----------------------------------------------------------
; Input:
;   StartX, StartY - Image location
;   temp_array     - the array for save a back ground
; Output:
;   None
; Registers
;   AX, SI
;-----------------------------------------------------------
MACRO MSaveBkGround  StartX, StartY, array
   mov ax, [StartX]
   mov [current_X], ax
   mov ax, [StartY]
   mov [current_Y], ax

   mov si, offset array
   call PSaveBkGround
ENDM MSaveBkGround
;-----------------------------------------------------------
; Hearts_view
;-----------------------------------------------------------
; Input:
;    hearts variable
; Output:
;   output the mode of the Hearts_view
; Registers
;   AX, Dx
;-----------------------------------------------------------
MACRO Hearts_view
        cmp [hearts],3
        je @@heart_3_label
        cmp [hearts],2
        je @@heart_2_label
        cmp [hearts],1
        je @@heart_1_label
        MDrawImage  StartPictX, StartPicty, heart_1
        jmp @@endHearts

@@heart_3_label:
        MDrawImage  StartPictX, StartPicty, heart_3
        jmp @@endHearts
@@heart_2_label:
MDrawImage  StartPictX, StartPicty, heart_2
        jmp @@endHearts
@@heart_1_label:
        MDrawImage  StartPictX, StartPicty, heart_1
        jmp @@endHearts
@@endHearts:
endM Hearts_view
```

```
FileHandle          DW ?

            StartY                          db 0
            StartX                          db 0

      FileSize          DW ?

      ImageSizeInFile         DW ?

      ImageWidth              DW ?
      ImageHeigth             DW ?
      Point_X           DW ?
      Point_Y           DW ?
      Color             DB ?
      StartPictX        DW ?
      StartPictY        DW ?

            current_x_player   dw ?
            current_y_player   dw ?

            w_img                   dw ?
            h_img                   dw ?

            current_x          dw ?
            current_y          dw ?

            enemy1_x                dw 100
            enemy1_y                dw 100

            boss_X                          dw 160
            boss_Y                          dw 35

            shot_x                          dw 170
            shot_y                          dw 65

            set_menu                db 0h
            set_proc                db 0 ;deffult return to menu , 1= game, 2 , about me

            temp_array              db 13*16 dup(0)
            temp_tear0              db 9*9 dup(0)
            temp_tear1              db 9*9 dup(0)

            temp_enemy                      dw 48*28 dup(0)
            temp_boss               dw 33*28 dup(0)
            temp_shot               dw 4*86 dup(0)

            key                     db ?

            pic_aniMode             db 0

            save_Background_var db 1

            room                    db 1

            item                    db 0
```

```
hearts                  db 3

leftVar         dw 0
rightVar        dw 0
upVar           dw 0
downVar                 dw 0
currentPress            dw 0

leftBorderNum           dw ?

counter                 dd 700000
```

```
cycleTime               dw 60000

tearX           dw  ?
tearY           dw  ?

inshot                  db 0

tearMode                db 0

tearLoop                db 30

shootB          db 0

randomVar               dw ?

enemyMove               db 40

dashAlive               db 0

ReadPixel_X             dw ?
ReadPixel_Y             dw ?

enemyLife               db 6

bossLife                db 10

enemyExist              db 1

bossExist               db 1

hitTimeLoop             dw 0

bossVar                 db 1

shotB_counter           dw 40

randomShotB             dw ?

inshotB                 db 0
```

```
SEGMENT FILEBUF para public
        DB 52142 DUP(?)
ENDS

        CODESEG

Start: ;----------------start------------------------
        mov ax, @data
        mov ds, ax
```

```
                call random
start_intro:
                call introP ;intro
start_menu:
                call menuP ;menu



chek_whereToJump:
                cmp [set_proc],0 ;back to the intro
                je start_intro
                cmp [set_proc],1 ; call game
                je call_game_proc
                cmp [set_proc],2 ; call about me
                je call_aboutMe_proc
                cmp [set_proc],3 ; start menu
                je start_menu

call_game_proc:
                call gameP              ;main function
                jmp chek_whereToJump
call_aboutMe_proc:
                call aboutMeP           ;aboutMe
                jmp chek_whereToJump


Exit:

        mov ax,04c00h
        int 21h
```

```
---------------------------------------------------------;
 ;introP - show intro
---------------------------------------------------------;
 ;Input:
 ;      any key/esc
 ;Output:
 ;      print the intro screen
 ;Registers
;        ax
---------------------------------------------------------;
        proc introP Near
        mov ax, 0013h
    int 10h
         =----------------- ;Show intro----------------=
    mov [StartPictX], 0d
    mov [StartPictY], 0d

   SHOWPCX StartPictX, StartPictY, int_file
        mov ah, 00h
        int 16h
        cmp al, 01Bh
        jne @@end_proc
        mov ah, edom txet tes ;0
        mov al, 2
        int 10h
        jmp exit
@@end_proc:
        ret
        endp introP
```

```
-------------------------------------------------------;
 ;menuP - MainMenu
-------------------------------------------------------;
 ;Input:
 ;       key up/ key down / esc / enter
 ;Output:
 ;       print menu screens
 ;Registers
;         ax
-------------------------------------------------------;
proc MenuP Near
        mov [StartPictX], 0d
        mov [StartPictY], 0d
       SHOWPCX StartPictX, StartPictY, menu elfi_1
                         -----;Clear keyboard buffer
                         mov ah,0ch
                         mov al,07h
                         int 21h
L:1
                         -----;Read SCAN code from keyboard port
                         in al,060h
                         push ax
                         -----;Checking the pressed key deffult up key
                         cmp al, DOWN_KEY
                         je Down
                         cmp al,1ch
                         je enterL
                         cmp al,01h
                         je introL

Up:
                         mov [set_menu],1 cip wohs ;0
                         SHOWPCX StartPictX, StartPictY, menu elfi_1
                         jmp contM
Down:
                         mov [set_menu],2 cip wohs;1
                         SHOWPCX StartPictX, StartPictY, menuelfi_2
                         jmp contM
introL:
                         mov [set_proc],ortni;0

                         jmp end_menuL
```

```
enterL:
        cmp [set_menu],0
        je gameL
        cmp[set_menu],1
        je aboutMeL
        jmp L1

ContM:
                pop ax
                mov ah, 00h
                int 16h
                jmp L1
gameL:
                mov [set_proc],emag llac ;1
                jmp end_menuL

aboutMeL:
                mov [set_proc],em tuoba wohs ;2

end_menuL:
        pop ax
        ret
        endp menuP
```

```
-----------------------------------------;
 ;gameP - here is the its the main function
-----------------------------------------;
 ;Input:
 ;       keys, RandomShotB, inshotB,inshotE
 ;Output:
 ;       from this function we call and show the diffrents rooms
;        the tears (bullets)
 ;Registers
       ;ax,cx,bx,dx
-----------------------------------------;
PROC gameP Near
;set the first player loction
mov [current_x_player],320/2
mov [current_y_player],100
rooms:
                cmp [room],4
                je win
                call roomsP ;call to the rooms and check what is the right room no

@@L :1

 ---;Save back ground
                MSaveBkGround StartPictX, StartPictY, temp_array ; save the background

        --;show the antmtion--


        call enemy1
        call boosMove

        call RandomShot
        cmp [inshotB],1
        je shotL
        cmp [RandomShotB],4
        ja contShotBmain
        mov [inshotB],1
        mov ax,[boss_X]
        add ax,10
        mov [shot_x],ax
        mov ax,[boss_Y]
        add ax,35
        mov [shot_Y],ax
        mov [w_img],4
        mov [h_img],86
        MSaveBkGround shot_x, shot_y , temp_shot
shotL:
        call shotBoss
contShotBmain:
```

```
        call animate

        cmp [hitTimeLoop],0
        jne   checkHitL
        call checkHit
        jmp hitCont
checkHitL:
        dec [hitTimeLoop]
hitCont:
        cmp [hearts],0
        je loseL
        --;the shot proc--
        call shot




@@contine:
        ---;delay
                dec [counter]
                jnz @@contine
                mov [counter], 700000

         -----;Read SCAN code from keyboard port
                in al,060h
                mov [key], al
                mov al,0
        ----;check if there is a shot on the screen
                cmp [inshot],1
                je shotCONT
```

```
-----;Checking the arrow key pressed

        cmp [key], UP_KEY_arrow
        je @@UpArrow

        cmp [key], DOWN_KEY_arrow
        je @@DownArrow

        cmp [key], RIGHT_KEY_arrow
        je @@RightArrow

        cmp [key], LEFT_KEY_arrow
        je @@leftArrow
```

```
shotCONT:

-----;Checking key is released
    cmp [key], UP_UP
    je @@UpUp

    cmp [key], DOWN_UP
    je @@DownUp


    cmp [key], RIGHT_UP
    je @@RightUp

    cmp [key], LEFT_UP
    je @@LeftUp

-----;Checking the pressed key

        cmp [key], UP_KEY
        je @@Up

        cmp [key], DOWN_KEY
        je @@Down

        cmp [key], RIGHT_KEY
        je @@Right

        cmp [key], LEFT_KEY
        je @@left
        jmp @@calculat



@@leftArrow:
    ---;set the tear loction
        push [StartPictX]
```

```
                pop [tearX]
                sub [tearX],10
                push [StartPictY]
                pop [tearY]
        -------;save Background--------
                MSaveBkGround tearX, tearY, temp_tear0
                mov [shootB],1
                jmp @@calculat
@@upArrow:
        ---;set the tear loction
                push [StartPictX]
                pop [tearX]
                push [StartPictY]
                pop [tearY]
                sub [tearY],13
        -------;save Background--------
                MSaveBkGround tearX, tearY, temp_tear0
                mov [shootB],2
                jmp @@calculat
@@downArrow :
        ---;set the tear loction
                push [StartPictX]
                pop [tearX]
                push [StartPictY]
                pop [tearY]
                add [tearY],20
        -------;save Background--------
                MSaveBkGround tearX, tearY, temp_tear0
                mov [shootB],3
                jmp @@calculat
@@rightArrow :
---;set the tear loction
                push [StartPictX]
                pop [tearX]
                add [tearX],17
                push [StartPictY]
                pop [tearY]
-------;save Background--------
                MSaveBkGround tearX, tearY, temp_tear0
                mov [shootB],4
                jmp @@calculat



@@left:
                mov [leftVar],1      rav evom tes;
                jmp @@calculat
@@up:           mov [upVar],5                ;set move var
                jmp @@calculat
```

```
@@down: mov [downVar],8            ;set move var
                jmp @@calculat
@@right: mov [rightVar],2      ;set move var
                 jmp @@calculat




@@UpUp:   mov [upVar],         0
                 mov [downVar],       0
                 mov [rightVar],      0
                 mov [leftVar],       0
       jmp @@calculat

@@DownUp: mov [upVar],         0
                 mov [downVar],       0
                 mov [rightVar],      0
                 mov [leftVar],       0
       jmp @@calculat

@@RightUp:mov [rightVar],      0
                 mov [downVar],       0
                 mov [upVar],         0
                 mov [leftVar],       0
       jmp @@calculat

@@LeftUp:mov [leftVar],        0
        mov [downVar],         0
                 mov [rightVar],      0
                 mov [upVar],         0

;in this label i connect between all the move variables
;and because of that i know what the current dirctione
@@calculat:
                mov [w_img],13
                mov [h_img],16
        ---;draw the temp background
                mov [save_Background_var],0
                MDrawImage StartPictX, StartPictY, temp_array
                mov [save_Background_var],1
        --;connect between all the move variables
                mov bx,[upVar]
                add [currentPress],bx
                mov bx,[downVar]
                add [currentPress],bx
                mov bx,[rightVar]
                add [currentPress],bx
```

```
                mov bx,[leftVar]
                add [currentPress],bx

        ;check where to move
                        cmp [currentPress],1
                je @@leftMove
                        cmp [currentPress],2
                je @@RightMove
                        cmp [currentPress],5
                je @@UpMove
                        cmp [currentPress],8
                je @@DownMove
                        cmp [currentPress],6
                je @@UpLeftMove
                        cmp [currentPress],7
                je @@UpRightMove
                        cmp [currentPress],9
                je @@DownLeftMove
                        cmp [currentPress],10
                je @@DownRightMove
                jmp @@cont

------;Left------
@@leftMove:
-------;chek the left border
                mov ax,[leftBorderNum]
                cmp [StartPictX],ax
                jge leftBorder
                add [StartPictX],2
leftBorder:
                sub [StartPictX],2
-------------------;
---;check if the touch in black pixel on left pressed
                mov ah,0dh
                mov cx,[StartPictX]
                mov dx,[StartPictY]
                add dx,8
                mov bh,0
                int 10h
                cmp al,0
                je ChangeRoomPlus

-----;antimtion algorit
                cmp [pic_aniMode],6
                jne set_picLeft
                inc [pic_aniMode]
                jmp @@cont
set_picLeft:
                cmp [pic_aniMode],7
                jne set_picLeft1
```

```
                inc [pic_aniMode]
                jmp @@cont

set_picLeft:1
                cmp [pic_aniMode],8
                jne set_picLeft2

set_picLeft:2
                mov [pic_aniMode],6
                jmp @@cont

-------;UP---------
--;upleft
@@UpLeftMove:
        ---;border
                mov ax,[leftBorderNum]
                cmp [StartPictX],ax
                jge leftBorderU
                add [StartPictX],2
leftBorderU:
                sub [StartPictX],2
                jmp @@upMove

--;UpRight
@@UpRightMove:
        ---;border
                cmp [StartPictX], 230
                jle rightBorderD2
                sub [StartPictX],2
rightBorderD:2
                mov [currentPress],0
                add [StartPictX],2
--;Up
        @@UpMove :
                ;border
                mov [currentPress],0
                cmp [StartPicty], 35
                jge upBorder
                add [StartPicty],2
upBorder:
                sub [StartPictY],2

--;antimtion algorit
                cmp [pic_aniMode],0
                jne set_picup
                inc [pic_aniMode]
                jmp @@cont
set_picup:
                cmp [pic_aniMode],1
                jne set_picup1
```

```
                inc [pic_aniMode]
                jmp @@cont
set_picup:1
                cmp [pic_aniMode],2
                jne set_picup2

set_picup:2
                mov [pic_aniMode],0
                jmp @@cont


----;Down
--;DownLeft
@@DownLeftMove:
        ---;border
                mov ax,[leftBorderNum]
                cmp [StartPictX],ax
                jge leftBorderU2
                add [StartPictX],2
leftBorderU:2
                sub [StartPictX],2
                jmp @@DownMove

@@DownRightMove:
        ---;border
                cmp [StartPictX], 264
                jle rightBorderD
                sub [StartPictX],2
rightBorderD:
                mov [currentPress],0
                add [StartPictX],2
---;down
        @@DownMove:
        --;border
                mov [currentPress],0
                cmp [StartPicty], 153
                jle downBorder
                sub [StartPicty],2
downBorder:
                add [StartPictY],2

        --;animtion algorit
                cmp [pic_aniMode],3
                jne set_picDown
                inc [pic_aniMode]
                jmp @@cont
set_picDown:
                cmp [pic_aniMode],4
                jne set_picDown1
                inc [pic_aniMode]
```

```
                jmp @@cont
set_picDown:1
                cmp [pic_aniMode],5
                jne set_picDown2

set_picDown:2
                mov [pic_aniMode],3
                jmp @@cont



--;Right
        @@RightMove:
                cmp [StartPictX], 264
                jle rightBorder
                sub [StartPictX],2
rightBorder:
                add [StartPictX],2
--;check if the player hit red/black pixel
                mov ah,0dh
                mov cx,[StartPictX]
                add cx,16
                mov dx, [StartPictY]
                add dx, 5
                mov bh,0
                int 10h
                cmp al,der ;4
                je changeItem
                cmp al,kcalb;0
                je ChangeRoomMinus

                --;animtion algorit
                cmp [pic_aniMode],9
                jne set_picRight
                inc [pic_aniMode]
                jmp @@cont
set_picRight:
                cmp [pic_aniMode],10
                jne set_picRight1
                inc [pic_aniMode]
                jmp @@cont
set_picRight:1
                cmp [pic_aniMode],11
                jne set_picRight2

set_picRight:2
                mov [pic_aniMode],9

;contine
        @@Cont:
```

```
                ;check if esc pressed
                mov [currentPress],0
                cmp [key], ESC_KEY
                je @@quit
                jmp @@L1
;change the rooms and set new start place in the rooms
ChangeRoomMinus:
                mov [shootB],5
                mov [current_x_player],45
                mov [current_y_player],90
                dec [room]
                jmp rooms
ChangeRoomPlus:
                mov [shootB],5
                inc [room]
                mov [current_x_player],320-65
                mov [current_y_player],90
                jmp rooms
changeItem:
        mov [item],1
        mov [tearMode],1
        mov [enemyLife],4
        mov [bossLife],7
        push [StartPictX]
        pop [current_x_player]
        push [StartPictY]
        pop [current_y_player]
        jmp rooms
win:
        mov [StartPictX], 0d
    mov [StartPictY], 0d
        mov [hearts],3
   SHOWPCX StartPictX, StartPictY, winWindow
        ;clear the keybord buffer
        mov ah,0ch
        mov al,07h
        int 21h
        jmp @@quit
loseL:
        mov [StartPictX], 0d
    mov [StartPictY], 0d
        mov [hearts],3
   SHOWPCX StartPictX, StartPictY, loseWindow
        ;clear the keybord buffer
        mov ah,0ch
        mov al,07h
        int 21h
loseCHose:
        mov ah, 00h
        int 16h
```

```
        cmp al, 01Bh
        je @@quit
        cmp al,32
        je space
        jmp loseCHose
space:
        mov [current_x_player],320/2
        mov [current_y_player],100
        mov [room],1
        call intalazing
        jmp rooms

@@quit:
        call intalazing
        mov [hearts],3
        mov [room],1
        mov [set_proc],3


        ret
endp gameP


-----------------------------------------;
 ;DrawRoom1
-----------------------------------------;
 ;Input:
 ;      none
 ;Output:
 ;      room 1
 ;Registers
      ;none
-----------------------------------------;
proc DrawRoom1 Near
        mov [StartPictX],0
        mov [StartPictY],0
        SHOWPCX StartPictX, StartPictY, room1
        mov [w_img],36
        mov [h_img],9
        Hearts_view ;draw the hearts
        mov [w_img],41
        mov [h_img],10
        mov [StartPictX],320-45
        MDrawImage  StartPictX, StartPicty, map_1
        mov [w_img],13
        mov [h_img],16
        mov [leftBorderNum],26
        ret
endp DrawRoom1
```

```
-----------------------------------------;
 ;DrawRoom3C - boos room Close
-----------------------------------------;
 ;Input:
 ;       none
 ;Output:
 ;       room pam,straeh ,1
 ;Registers
       ;none
-----------------------------------------;
proc DrawRoom3C Near
        mov [StartPictX],0
        mov [StartPictY],0
        SHOWPCX StartPictX, StartPictY, boosRoomC
        mov [w_img],33
        mov [h_img],28
        MSaveBkGround boss_X, boss_Y, temp_boss
        mov [w_img],4
        mov [h_img],86
        MSaveBkGround shot_x, shot_y , temp_shot
        mov [w_img],36
        mov [h_img],9
        Hearts_view ;draw the hearts
        mov [w_img],41
        mov [h_img],10
        mov [StartPictX],320-45
        MDrawImage  StartPictX, StartPicty, map_3
        mov [w_img],13
        mov [h_img],16
        mov [leftBorderNum],44
        ret
endp DrawRoom3C
-----------------------------------------;
 ;DrawRoom3C - boos room Close
-----------------------------------------;
 ;Input:
 ;       none
 ;Output:
 ;       room 1
 ;Registers
       ;none
-----------------------------------------;
proc DrawRoom3O Near
        mov [StartPictX],0
        mov [StartPictY],0
        SHOWPCX StartPictX, StartPictY, boosRoomO
        mov [w_img],33
        mov [h_img],28
        mov [w_img],36
        mov [h_img],9
        Hearts_view ;draw the hearts
```

```
        mov [w_img],41
        mov [h_img],10
        mov [StartPictX],320-45
        MDrawImage  StartPictX, StartPicty, map_3
        mov [w_img],13
        mov [h_img],16
        mov [leftBorderNum],44
        ret
endp DrawRoom3O
-------------------------------------------;
 ;DrawRoom2 open
-------------------------------------------;
 ;Input:
 ;      none
 ;Output:
 ;      room 1
 ;Registers
      ;none
-------------------------------------------;
proc DrawRoom2O Near
        mov [StartPictX],0
        mov [StartPictY],0
        SHOWPCX StartPictX, StartPictY, enemyRoomO
        mov [w_img],36
        mov [h_img],9
        Hearts_view ;draw the hearts
        mov [w_img],41
        mov [h_img],10
        mov [StartPictX],320-45
        MDrawImage  StartPictX, StartPicty, map_2
        mov [w_img],13
        mov [h_img],16
        mov [leftBorderNum],44
        ret
endp DrawRoom2O
-------------------------------------------;
 ;DrawRoom2 Colse
-------------------------------------------;
 ;Input:
 ;      none
 ;Output:
 ;      room 1 with colse doors
 ;Registers
      ;none
-------------------------------------------;
proc DrawRoom2C Near

        mov [StartPictX],0
        mov [StartPictY],0
        SHOWPCX StartPictX, StartPictY, enemyRoomC
```

```
        mov [w_img],48
        mov [h_img],28
        MSaveBkGround enemyymene_pmet ,y_1ymene ,x_1
        mov [w_img],36
        mov [h_img],9
        Hearts_view ;draw the hearts
        mov [w_img],41
        mov [h_img],10
        mov [StartPictX],320-45
        MDrawImage  StartPictX, StartPicty, map_2
        mov [w_img],13
        mov [h_img],16
        mov [leftBorderNum],44
```

```
        ret
endp DrawRoom2C

-----------------------------------------;
 ;DrawRoommetIhtiw_0
-----------------------------------------;
 ;Input:
 ;      none
 ;Output:
 ;      item room with the item
 ;Registers
      ;none
-----------------------------------------;
proc DrawRoomraeN  metIhtiw_0
        mov [StartPictX],0
        mov [StartPictY],0
        SHOWPCX StartPictX, StartPictY, itemRoomi
        mov [w_img],36
        mov [h_img],9
        Hearts_view ;draw the hearts
        mov [w_img],41
        mov [h_img],10
        mov [StartPictX],320-45
        MDrawImage  StartPictX, StartPicty, map_0
        mov [w_img],13
        mov [h_img],16
        mov [leftBorderNum],44
        ret
endp DrawRoommetIhtiw_0
```

```
-----------------------------------------;
 ;DrawRoommetItuohtiw_0
-----------------------------------------;
 ;Input:
 ;       none
 ;Output:
 ;       item room without the item
 ;Registers
       ;none
-----------------------------------------;
proc DrawRoomraeN  metItuohtiw_0

        mov [StartPictX],0
        mov [StartPictY],0
        SHOWPCX StartPictX, StartPictY, itemRoomwi
        mov [w_img],36
        mov [h_img],9
        Hearts_view ;draw the hearts
        mov [w_img],41
        mov [h_img],10
        mov [StartPictX],320-45
        MDrawImage  StartPictX, StartPicty, map_0
        mov [w_img],13
        mov [h_img],16
        mov [leftBorderNum],44
        ret
endp DrawRoommetItuohtiw_0
-----------------------------------------;
 ;aboutMeP - about me
-----------------------------------------;
 ;Input:
 ;       esc
 ;Output:
 ;       about me
 ;Registers
     ;AX
-----------------------------------------;
PROC aboutMeP Near

        call PClearScreen
        SHOWPCX StartPictX, StartPictY, ab_file
        @@L060,la ni :1h ; al <- scan cod ah <- ASCII
        dec al ; scan cod ESC - 1
        jnz @@L1
        mov [set_menu],0
        mov [set_proc],3

        ret
endp aboutMeP
```

```
-------------------------------------------;
 ;drawIMG - draw bit map image
-------------------------------------------;
 ;Input:
 ;       none
 ;Output:
 ;       bitMap img pixel after pixel
 ;Registers
       ;AX,cx,si,bh
-------------------------------------------;
PROC PDrawImage Near
    mov cx, [h_img]
vertical_loop:
    push cx
    mov cx, [w_img]
horizontal_loop:
    push cx
    mov bh, 0
    mov cx, [current_x]
    mov dx, [current_y]
    mov al, [si]
--;fix the black problem, when the the temp draw its ignore from the black
          cmp [save_Background_var],0
          je save_Background
    cmp al, TRANSPORENT_COLOR
    je not_draw
save_Background:
    mov ah, 0ch
    int 10h
not_draw:
    inc [current_x]
    inc si
    pop cx
    loop horizontal_loop
    inc [current_y]
    mov ax, [current_x]
    sub ax, [w_img]
    mov [current_x], ax

    pop cx
    loop vertical_loop
    ret
ENDP PDrawImage
```

```
-----------------------------------------;
 ;ReadPCXFile - read PCX file into FILEBUF
-----------------------------------------;
 ;Input:
 ;       File name
 ;Output:
 ;       File into FILEBUF
 ;Registers
     ;AX, BX, CX, DX, DS
-----------------------------------------;
PROC ReadPCXFile Near
        pusha

  -----;Initialize variables
        mov     [FileHandle],0
        mov     [FileSize],0

  -----;Open file for reading
        mov     ah, 3Dh
        mov     al, 0
 ;        mov DX,offset FileName
        int    21h
        jc      @@Err
        mov     [FileHandle],AX   ; save Handle

  -----;Get the length of a file by setting a pointer to its end
        mov     ah, 42h
        mov     al ,2
        mov     bx, [FileHandle]
        xor     cx, cx
        xor     dx, dx
        int    21h
        jc                rrE@@
        cmp    dx,0
        jne     @@Err  ;file size exceeds 64K

  -----;Save size of file
        mov     [FileSize], ax

 -----;Return a pointer to the beginning of the file
        mov     ah, 42h
        mov     al, 0
        mov     bx, [FileHandle]
        xor     cx, cx
        xor dx, dx
        int 21h
        jc  @@Err

  -----;Read file into FILEBUF
        mov     bx, [FileHandle]
        pusha
        push    ds
```

```
mov     ax,FILEBUF
        mov     ds, ax
        xor     dx, dx
        mov     cx, 52142
        mov     ah, 3Fh
        int     21H
        pop     ds
        popa
        jc      @@Err
```

```
-----;Close the file
        mov     ah, 3Eh
        mov     bx,[FileHandle]
        int     21H
        jc      @@Err
        popa
        ret

-----;Exit - error reading file
@@Err:  ; Set text mode
        mov     ax, 3
        int     10h

        mov     dx, offset ErrorReadingFile
        mov     ah, 09h
        int     21h
        jmp     Exit
ENDP ReadPCXFile
```

```
-----------------------------------------;
 ;PSaveBkGround - save the backround
-----------------------------------------;
 ;Input:
 ;      File name
 ;Output:
 ;      The file
 ;Registers
;        AX, BX, CX, DX, DS
-----------------------------------------;
PROC PSaveBkGround Near
    mov cx, [h_img]
@@vertical_loop:
    push cx
    mov cx, [w_img]
@@horizontal_loop:
    push cx
    mov bh, 0
    mov cx, [current_x]
    mov dx, [current_y]
    mov ah, 0dh
    int 10h
    mov [si], al
    inc [current_x]
    inc si
    pop cx
    loop @@horizontal_loop
    inc [current_y]
    mov ax, [current_x]
    sub ax, [w_img]
    mov [current_x], ax

    pop cx
    loop @@vertical_loop
    ret
ENDP PSaveBkGround
```

```
-------------------------------------------;
 ;ShowPCXFile - show PCX file
-------------------------------------------;
 ;Input:
 ;       File name
 ;Output:
 ;       The file
 ;Registers
;         AX, BX, CX, DX, DS
-------------------------------------------;
PROC ShowPCXFile Near
        pusha
```

```
        call    ReadPCXFile

         mov   ax, FILEBUF
        mov     es, ax

  -----;Set ES:SI on the image
        mov     si, 128

  -----;Calculate the width and height of the image
        mov     ax, [es:42h]
        mov     [ImageWidth], ax
        dec     [ImageWidth]

        mov     ax, [es:0Ah]
        sub     ax, [es:[6
        inc     ax
        mov     [ImageHeigth], ax

  -----;Calculate the offset from the beginning of the palette file
        mov     ax, [FileSize]
        sub     ax, 768
        mov     ax, [FileSize]
        sub     ax, 128+768
        mov     [ImageSizeInFile], ax

        xor  ch, ch             ; Clear high part of CX for string copies
        push [StartPictX]       ; Set start position
        pop  [Point_x]
        push [StartPictY]
        pop  [Point_y]
```

```
NextByte:
        mov     cl, [es:si]     ; Get next byte
        cmp     cl, 0C0h        ; Is it a length byte?
        jb      normal          ;  No, just copy it
        and     cl, 3Fh         ; Strip upper two bits from length byte
        inc     si              ; Advance to next byte - color byte

         mov     al, [es:si]
         mov   la ,[roloC]
NextPixel:
        call      PutPixel
        cmp     cx, 1
                  je       CheckEndOfLine

        inc     [Point_X]

                  loop   NextPixel
        jmp     CheckEndOfLine
Normal:
         mov   lc ,[roloC]
        call      PutPixel

CheckEndOfLine:
        mov     ax, [Point_X]
        sub     ax, [StartPictX]
        cmp     ax, [ImageWidth]
 [tciPhtdiW] =< [XtciPtratS] - [X_tnioP]  -----;
        jae     LineFeed
        inc     [Point_x]
        jmp     cont
LineFeed:
        push    [StartPictX]
        pop     [Point_x]
        inc     [Point_y]
cont:
        inc     si
        cmp     si, [ImageSizeInFile]     ; End of file? (written 320x200 bytes)
        jb      nextbyte
        popa
        ret
ENDP ShowPCXFile
```

```
;-----------------------------------------
; PutPixel - draw pixel
;-----------------------------------------
; Input:
;       x - Point_x, y - Point_y, Color - color
; Output:
;       The pixel
; Registers
;        AX, BH, CX, DX
;-----------------------------------------
PROC PutPixel near
      pusha
      mov    bh, 0h
      mov    cx, [Point_x]
      mov    dx, [Point_Y]
      mov    al, [color]
      mov    ah, 0ch
      int    10h
      popa
      ret
ENDP PutPixel

;-----------------------------------------
; PClearScreen
;-----------------------------------------
; Input:
;       none
; Output:
;       The pixel
; Registers
;        ax,es,cx,di
;-----------------------------------------
PROC PClearScreen near
        pusha
   mov ax, VGA_SEGMENT
   mov es,ax         ; es:di - video memory
   xor di,di
   mov cx,320*200/2
   mov al, 0       ; color
   mov ah, 0       ; color
   rep stosw        ; mov es:[di],ax   add di,2
        popa
ret
ENDP PClearScreen
```

```
-----------------------------------------;
 ;animate
-----------------------------------------;
 ;Input:
 ;       pic_aniMode
 ;Output:
 ;       the player in diffrent antimtion
 ;Registers
;          none
-----------------------------------------;
proc animate near
         ---;Show Picture
                  mov [w_img],13
                  mov [h_img],16
                  cmp [pic_aniMode],1  - ;0  dirctiones
                  JE Dfor_0                    ;
                  cmp [pic_aniMode],1  - ;1  dirctiones
                  JE Dfor_     ;            1
                  cmp [pic_aniMode],1  - ;2  dirctiones
                  JE Dfor_   - ;            2
                  cmp [pic_aniMode],2  - ;3  dirctiones
                  JE Dback_   - ;           0
                  cmp [pic_aniMode],2  - ;4  dirctiones
                  JE Dback_   - ;           1
                  cmp [pic_aniMode],2  - ;5  dirctiones
                  JE Dback_  - ;           2
                  cmp [pic_aniMode],3  - ;6  dirctiones
                  JE Dleft_   - ;           0
                  cmp [pic_aniMode],3  - ;7  dirctiones
                  JE Dleft_  - ;           1
                  cmp [pic_aniMode],3  - ;8  dirctiones
                  JE Dleft_   - ;           2
                  cmp [pic_aniMode],4  - ;9  dirctiones
                  JE Dright_  - ;          0
                  cmp [pic_aniMode],4  - ;10 dirctiones
                  JE Dright_   - ;          1
                  cmp [pic_aniMode],4  - ;11 dirctiones
                  JE Dright_2
```

```
Dfor_:0
        MDrawImage StartPictX, StartPictY, back_notimina;0
        jmp @@contine                       ;animtion
Dfor_:1         notimina;
        MDrawImage StartPictX, StartPictY, back_notimina;1
        jmp @@contine                       ;animtion
Dfor_:2         notimina;
        MDrawImage StartPictX, StartPictY, back_notimina;2
        jmp @@contine                       ;animtion
Dback_:0    notimina;
        MDrawImage StartPictX, StartPictY, for_notimina;0
        jmp @@contine                       ;animtion
Dback_notimina;                         :1
        MDrawImage StartPictX, StartPictY, for_notimina;1
        jmp @@contine                       ;animtion
Dback_notimina;                         :2
        MDrawImage StartPictX, StartPictY, for_notimina;2
        jmp @@contine                       ;animtion
Dleft_notimina;                         :0
        MDrawImage StartPictX, StartPictY, left_notimina;0
        jmp @@contine                       ;animtion
Dleft_notimina;                         :1
        MDrawImage StartPictX, StartPictY, left_notimina;1
        jmp @@contine
```

```
;animtion
Dleft_notimina;                                    :2
            MDrawImage StartPictX, StartPictY, left_notimina;2
            jmp @@contine                              ;animtion
Dright_notimina;                                   :0
            MDrawImage StartPictX, StartPictY, right_notimina;0
            jmp @@contine
Dright_:1
            MDrawImage StartPictX, StartPictY, right_1
            jmp @@contine
Dright_:2
            MDrawImage StartPictX, StartPictY, right_2


@@contine:
ret
endp animate
```

```
-----------------------------------------;
 ;roomsP
-----------------------------------------;
 ;Input:
 ;      room, current_x_player, current_y_player
 ;Output:
 ;      the diffrent rooms
 ;Registers
;        none
-----------------------------------------;
proc roomsP near
        cmp [room],3
        je roomwon_3
        cmp [room],2
        je roomwon_2
        cmp [room],1
        je roomwon_1
        cmp [room],0
        je roomwon_0

room:won_3
        cmp [bossExist],1
        jne roomlebaL_3
        call DrawRoom3C
        push [current_x_player]
        pop [StartPictX]                reyalp eht fo ecalp trats eht tes;
        push [current_y_player]
        pop [StartPictY]
        jmp @@L1
room:lebaL_3
        call DrawRoom3O
        push [current_x_player]
        pop [StartPictX]                reyalp eht fo ecalp trats eht tes;
        push [current_y_player]
        pop [StartPictY]                reyalp eht fo ecalp trats eht tes;
        jmp @@L1
roommoor tsrfi;    :won_2
```

```
        cmp [enemyExist],1
        jne roomlebaL_2
        call DrawRoom2C
        push [current_x_player]
        pop [StartPictX]                reyalp eht fo ecalp trats eht tes;
        push [current_y_player]
        pop [StartPictY]                        reyalp eht fo ecalp trats eht tes;
        jmp @@L1
room:lebaL_2
        call DrawRoom2O
        push [current_x_player]
        pop [StartPictX]                reyalp eht fo ecalp trats eht tes;
        push [current_y_player]
        pop [StartPictY]                        reyalp eht fo ecalp trats eht tes;
        jmp @@L1

roommoor tsrfi;    :won_1
        call DrawRoom1
        push [current_x_player]
        pop [StartPictX]                reyalp eht fo ecalp trats eht tes;
        push [current_y_player]
        pop [StartPictY]                        reyalp eht fo ecalp trats eht tes;

        jmp @@L1
roommoor meti;    :won_0
        cmp [item],1
        je ItemLablel
        call DrawRoommetIhtiw_0
        push [current_x_player]
        pop [StartPictX]                reyalp eht fo ecalp trats eht tes;
        push [current_y_player]
        pop [StartPictY]                        reyalp eht fo ecalp trats eht tes;
        jmp @@L1
ItemLablel:        meti tuohtiw moor meti;
        call DrawRoommetItuohtiw_0
        push [current_x_player]
        pop [StartPictX]
        push [current_y_player]
        pop [StartPictY]
@@L:1
ret
endp roomsP
```

```
-----------------------------------------;
 ;shot proc
-----------------------------------------;
 ;Input:
 ;      shootB - the current shoot
 ;Output:
 ;      shot animtion
 ;Registers
;        ax
-----------------------------------------;
proc shot near
        mov [w_img],9
        mov [h_img],9
        cmp [shootB],0
        je contT
        cmp [shootB],5
        je endShotRelese2
        mov [save_Background_var],0
        MDrawImage tearX, tearY, temp_tear0
        mov [save_Background_var],1


contT:

        cmp [shootB],0
        je endShot
        cmp [shootB], 1
        je LTear
        cmp [shootB], 2
        je UTear
        cmp [shootB], 3
        je DTear
        cmp [shootB], 4
        je RTear
        jmp endShot
Ltear:
;border for the tear
        mov ax,[leftBorderNum]
        cmp [tearX],ax
        jge TearBorderLeft
        add [tearX],4
```

```
TearBorderLeft:
;chek hit
        mov ax,[tearX]
        sub ax,17
        mov [ReadPixel_X],ax
        mov ax,[tearY]
        add ax,5
        mov [ReadPixel_Y],ax
        call ReadPixel
        cmp al,030
        je hitE
        cmp al,16
        je     hitB

        mov [inshot],1
        sub [tearX],3
        call TearD ;draw tear
        dec [tearLoop]
        jz endShotRelese
        jmp endShot

UTear:
;border for the tear
        cmp [tearY],35
        jge TearBorderUP
        cmp [room],3
        je endShotRelese
        add [tearY],4
TearBorderUP:
;chek hit
        mov ax,[tearX]
        sub ax,12
        mov [ReadPixel_X],ax
        mov ax,[tearY]
        sub ax,2
        mov [ReadPixel_Y],ax
        call ReadPixel
        cmp al,030
        je hitE
        cmp al,16
        je     hitB
```

```
        sub [tearY],3
        mov [inshot],1
        call TearD ;draw tear
        dec [tearLoop]
        jz endShotRelese
        jmp endShot

DTear:
;border for the tear
        cmp [tearY],153
        jle TearBorderDown
        sub [tearY],4
TearBorderDown:
;chek hit
        mov ax,[tearX]
        sub ax,12
        mov [ReadPixel_X],ax
        mov ax,[tearY]
        add ax,11
        mov [ReadPixel_Y],ax
        call ReadPixel ;draw tear
        cmp al,030
        je hitE

        add [tearY],3
        mov [inshot],1
        call TearD
        dec [tearLoop]
        jz endShotRelese
        jmp endShot
RTear:
;border for the tear
        cmp [tearX],264
        jle TearBorderRight
        sub [tearX],4
TearBorderRight:
```

```
;chek hit
        mov ax,[tearX]
        sub ax,5
        mov [ReadPixel_X],ax
        mov ax,[tearY]
        add ax,5
        mov [ReadPixel_Y],ax
        call ReadPixel ;draw tear
        cmp al,030
        je hitE
        cmp al,16
        je      hitB
```

```
        add [tearX],3
        mov [inshot],1
        call TearD
        dec [tearLoop]
        jz endShotRelese
        jmp endShot

hitE: ; hit enemy
        call beep_M
        mov [ReadPixel_X],0
        mov [ReadPixel_Y],0
        dec [enemyLife]

        jmp endShotRelese
hitB: ; hit boss
        call beep_M
        mov [ReadPixel_X],0
        mov [ReadPixel_Y],0
        dec [bossLife]
endShotRelese:2
MSaveBkGround tearX, tearY, temp_tear0
endShotRelese: ; reselse shot
        mov [shootB],0
        mov [inshot],0
        mov [tearLoop],30
        mov [save_Background_var],0
        MDrawImage tearX, tearY, temp_tear0
        mov [save_Background_var],1
        call clearParticals

endShot:

ret
endp shot
```

```
-----------------------------------------;
 ;TearD
-----------------------------------------;
 ;Input:
 ;      tearMode
 ;Output:
 ;      draw the tears
 ;Registers
;        none
-----------------------------------------;
proc TearD near

        MSaveBkGround tearX, tearY,
temp_tear0
        cmp [tearMode],1
        je tearL
        MDrawImage tearX, tearY,
tear_size0
        jmp endTL
tearL:
        MDrawImage tearX, tearY,
tear_size1
endTL:
ret
endp TearD
```

```
-----------------------------------------;
 ;enemy
-----------------------------------------;
 ;Input:
 ;       enemyExist,room,randomVar
 ;Output:
 ;       draw the enemy with anitmion
 ;Registers
;         none
-----------------------------------------;
proc enemy1 near
        mov [w_img],48
        mov [h_img],28
```

```
        cmp [enemyExist],1
        jne next
        cmp [room],2
        jne next
        cmp [enemyLife],0
        je killE1
        cmp [dashAlive],0
        je contD
        jmp enemtChekCont
contd:
        mov bx,[StartPictY]
        mov ax,[StartPictX]

        cmp ax,[enemy[x_1
        je dash_vertical
        cmp bx,[enemy[y_1
        je dash_horizental
        jmp enemtChekCont
dash_horizental:
        cmp ax,[enemy[x_1
        ja setRightDash
        mov [randomVar],hsad ftel;4
        mov [dashAlive],1
        jmp enemtChekCont
setRightDash:
        mov [randomVar],hsad thgir ;5
        mov [dashAlive],1
        jmp enemtChekCont
dash_vertical:
        cmp bx,[enemy[y_1
        ja setDownDash
        mov [randomVar],hsad pu;7
        mov [dashAlive],1
        jmp enemtChekCont
setDownDash:
        mov [randomVar],hsad nwod ;6
        mov [dashAlive],1
```

```
enemtChekCont:
        call dash
        ;chack where to move
        cmp [randomVar],0
        je rightE
        cmp [randomVar],1
        je LeftE
        cmp [randomVar],2
        je UPE
        cmp [randomVar],3
        jne next
downE:
        mov [save_Background_var],0
        MDrawImage enemyymene_pmet ,y_1ymene ,x_1
        mov [save_Background_var],1
        inc [enemy[y_1
        MSaveBkGround enemyymene_pmet ,y_1ymene ,x_1
        MDrawImage enemynwod_ymene ,y_1ymene ,x_1
        cmp [enemy153,[y_1
        je DownBorderE
        dec [enemyMove]
        jz enemy1Cont
        jmp next
rightE:

        mov [save_Background_var],0
        MDrawImage enemyymene_pmet ,y_1ymene ,x_1
        mov [save_Background_var],1
        inc [enemy[x_1
        MSaveBkGround enemyymene_pmet ,y_1ymene ,x_1
        MDrawImage enemyN_thgiR_ymene ,y_1ymene ,x_1
        cmp [enemy225,[x_1
        je RightBorderE
        dec [enemyMove]
        jz enemy1Cont
        jmp next
LeftE:
```

```
        mov [save_Background_var],0
        MDrawImage enemyymene_pmet ,y_1ymene ,x_1
        mov [save_Background_var],1
        dec [enemy[x_1
        MSaveBkGround enemyymene_pmet ,y_1ymene ,x_1
        MDrawImage enemyN_fteL_ymene ,y_1ymene ,x_1
        mov ax,[leftBorderNum]
        cmp [enemyxa,[x_1
        je LeftBorderE
        dec [enemyMove]
        jz enemy1Cont
        jmp next
UPE:

        mov [save_Background_var],0
        MDrawImage enemyymene_pmet ,y_1ymene ,x_1
        mov [save_Background_var],1
        dec [enemy[y_1
        MSaveBkGround enemyymene_pmet ,y_1ymene ,x_1
        MDrawImage enemyPU_ymene ,y_1ymene ,x_1
        cmp [enemy35,[y_1
        je UPBorderE
        dec [enemyMove]
        jz enemy1Cont
        jmp next

killE:1
        mov [enemyExist],0
        MDrawImage enemyymene_pmet ,y_1ymene ,x_1
        ;set the start place of the player
        push [StartPictX]
        pop [current_x_player]
        push [StartPictY]
        pop [current_y_player]
        call roomsP

        jmp next
```

```
enemy1Cont:
        call Random ;set a new dirction
        mov [enemyMove],elcyc eht teser;40
        jmp next
;if touch the border going the the oopiste dirction
UPBorderE:
        mov [randomVar],3
        mov [enemyMove],40
        jmp next
```

```
DownBorderE:
        mov [randomVar],2
        mov [enemyMove],40
        jmp next

RightBorderE:
        mov [randomVar],1
        mov [enemyMove],40
        jmp next

LeftBorderE:
        mov [randomVar],0
        mov [enemyMove],40

next:
        ret
endp enemy1
```

```
-----------------------------------------;
 ;dash
-----------------------------------------;
 ;Input:
 ;      randomVar
 ;Output:
 ;      set the new intrupt adress
 ;Registers
;        ax,ds,dx
-----------------------------------------;
proc dash near
        cmp [randomVar],5
        je rightEdash
        cmp [randomVar],4
        je LeftEdash
        cmp [randomVar],7
        je UPEdash
        cmp [randomVar],6
        jne nextdash

downEdash:
        mov [save_Background_var],0
        MDrawImage enemyymene_pmet ,y_1ymene ,x_1
        mov [save_Background_var],1
        add [enemydeepSymene,[y_1
        MSaveBkGround enemyymene_pmet ,y_1ymene ,x_1
        MDrawImage enemynwod_ymene ,y_1ymene ,x_1
        cmp [enemy153,[y_1
        jae downDash
        jmp nextdash
rightEdash:

        mov [save_Background_var],0
        MDrawImage enemyymene_pmet ,y_1ymene ,x_1
        mov [save_Background_var],1
        add [enemydeepSymene,[x_1
        MSaveBkGround enemyymene_pmet ,y_1ymene ,x_1
        MDrawImage enemyO_thgiR_ymene ,y_1ymene ,x_1
        cmp [enemy225,[x_1
        jae rightDash
        jmp nextdash
```

```
LeftEdash:
        mov [save_Background_var],0
        MDrawImage enemyymene_pmet ,y_1ymene ,x_1
        mov [save_Background_var],1
        sub [enemydeepSymene,[x_1
        MSaveBkGround enemyymene_pmet ,y_1ymene ,x_1
        MDrawImage enemy ,x_1enemyO_fteL_ymene ,y_1
        mov ax,[leftBorderNum]
        cmp [enemyxa,[x_1
        jbe leftdash
        jmp nextdash
UPEdash:

        mov [save_Background_var],0
        MDrawImage enemyymene_pmet ,y_1ymene ,x_1
        mov [save_Background_var],1
        sub [enemydeepSymene,[y_1
        MSaveBkGround enemyymene_pmet ,y_1ymene ,x_1
        MDrawImage enemyPU_ymene ,y_1ymene ,x_1
        cmp [enemy35,[y_1
        jbe upDash
        jmp nextdash
rightDash:
mov [randomVar],1
mov [dashAlive],0
jmp nextdash
downDash:
mov [randomVar],2
mov [dashAlive],0
jmp nextdash
upDash:
mov [randomVar],3
mov [dashAlive],0
jmp nextdash
leftdash:
mov [randomVar],0
mov [dashAlive],0
jmp nextdash
nextdash:
ret
endp dash
```

```
-----------------------------------------;
 ;Random - random number 0-3
-----------------------------------------;
 ;Input:
 ;       none
 ;Output:
 ;       random number for enemy
 ;Registers
;         ax
-----------------------------------------;
proc Random near
push ax
in al,40h ; Port of timer 0  255
and ax,11b
mov [randomVar],ax
pop ax
ret
endp Random
-----------------------------------------;
 ;Random - random number 0-3
-----------------------------------------;
 ;Input:
 ;       none
 ;Output:
 ;       random number for boss
 ;Registers
;         ax
-----------------------------------------;
proc RandomShot near
push ax
in al,40h ; Port of timer 0  255
mov [RandomShotB],ax
pop ax
ret
endp RandomShot
 ;-----------------------------------------
; readPixel
;-----------------------------------------
; Input:
;       ReadPixel_X,ReadPixel_Y
; Output:
;       al=the color
; Registers
;         ax,ds,dx
;-----------------------------------------
proc ReadPixel near
        mov ah,0dh
        mov cx,[ReadPixel_X]
        add cx,16
        mov dx, [ReadPixel_Y]
        mov bh,0
        int 10h
        ret
endp ReadPixel
```

```
-----------------------------------------;
 ;checkHit
-----------------------------------------;
 ;Input:
 ;      StartPictX,StartPictY
 ;Output:
 ;      none
 ;Registers
;        ax,ds,dx
-----------------------------------------;
proc checkHit near
        ;up
        mov ax,[StartPictX]
        sub ax,8
        mov [ReadPixel_X],ax
        mov ax,[StartPictY]
        dec ax
        mov [ReadPixel_Y],ax
        call ReadPixel
        cmp al,30
        je DLife
        cmp al,16
        je DLife
        ;down
        mov ax,[StartPictX]
        sub ax,8
        mov [ReadPixel_X],ax
        mov ax,[StartPictY]
        add ax,14
        mov [ReadPixel_Y],ax
        call ReadPixel
        cmp al,30
        je DLife
        cmp al,40
        je DLife
        cmp al,16
        je DLife
```

```
        ;right
        mov ax,[StartPictX]
        inc ax
        mov [ReadPixel_X],ax
        mov ax,[StartPictY]
        add ax,6
        mov [ReadPixel_Y],ax
        call ReadPixel
        cmp al,16
        je DLife
        cmp al,30
        je DLife
        cmp al,40
        je DLife
        ;left
        mov ax,[StartPictX]
        sub ax,17
        mov [ReadPixel_X],ax
        mov ax,[StartPictY]
        add    ax,6
        mov [ReadPixel_Y],ax
        call ReadPixel
        cmp al,30
        je DLife
        cmp al,16
        je DLife
        cmp al,40
        je DLife
        jmp contHIT
DLife:
        call beep_M2
        call clearParticals
        mov [hitTimeLoop],30
        dec [hearts]
        mov [w_img],36
        mov [h_img],9
        push [StartPictX]
        push [StartPictY]
        mov [StartPictX],0
        mov [StartPictY],0
        Hearts_view ;draw the hearts
        pop [StartPictY]
        pop [StartPictX]
        mov [w_img],13
        mov [h_img],16
contHIT:

        ret
endp checkHit
```

```
-----------------------------------------;
 ;intalazing
-----------------------------------------;
 ;Input:
 ;      none
 ;Output:
 ;intalazing the variables
 ;Registers
;        ax,ds,dx
-----------------------------------------;
proc intalazing near
        mov [enemyExist],1
        mov  [tearMode],0
        mov  [item],0
        mov  [enemy100,[x_1
        mov  [enemy100,[y_1
        mov  [dashAlive],0
        mov  [leftVar],0
        mov  [rightVar],0
        mov  [upVar],0
        mov  [downVar],0
        mov  [enemyLife],6
        mov [bossLife],5
        mov [bossExist],1
        ret
endp intalazing
```

```
-----------------------------------------;
 ;boosMove
-----------------------------------------;
 ;Input:
 ;      bossLife,bossExist
 ;Output:
 ;      the boss in new postion
 ;Registers
;        none
-----------------------------------------;
proc boosMove near
        mov [w_img],33
        mov [h_img],28
cmp [bossExist],1
        jne nextB
cmp [room],3
        jne nextB
cmp [bossLife],0
        je killB1
cmp [bossVar],0
        je rightB
cmp [bossVar],1
        je LeftB
        jmp nextB
LeftB:
        mov [save_Background_var],0
        MDrawImage boss_X, boss_Y, temp_boss
        mov [save_Background_var],1
        sub [boss_X],3
        MSaveBkGround boss_X, boss_Y, temp_boss
        MDrawImage boss_X, boss_Y, boos
        cmp [boss_X],40
        je LeftBorderB
        jmp nextB
```

```
rightB:
        mov [save_Background_var],0
        MDrawImage boss_X, boss_Y, temp_boss
        mov [save_Background_var],1
        add [boss_X],3
        MSaveBkGround boss_X, boss_Y, temp_boss
        MDrawImage boss_X, boss_Y, boos
        cmp [boss_X],250
        je RightBorderB
        jmp nextB
        ------------------------------------------------------------------------------------------

RightBorderB:
        mov [bossVar],1
        jmp nextB

LeftBorderB:
        mov [bossVar],0
        jmp nextB
killB:1
        mov [bossExist],0
        MDrawImage enemyssob_pmet ,y_1ymene ,x_1
        push 160
        pop [current_x_player]
        push 100
        pop [current_y_player]
        ;MSaveBkGround StartPictX, StartPictY, temp_array
 ;save the background
        call roomsP
nextB:

        ret
endp boosMove
```

```
-----------------------------------------;
 ;shotBoss - show the lazer shot
-----------------------------------------;
 ;Input:
 ;      none
 ;Output:
 ;      shot of the boss
 ;Registers
;       none
-----------------------------------------;
proc shotBoss near
        mov [w_img],4
        mov [h_img],86
        cmp [room],3
        jne contShotB
        cmp [bossExist],1
        jne contShotB
        cmp [shotB_counter],0
        je endShotB
        dec [shotB_counter]
        cmp [bossVar],0
        je rightBshot
        cmp [bossVar],1
        je LeftBshot
        jmp contShotB
rightBshot:
        mov [save_Background_var],0
        MDrawImage shot_x, shot_y, temp_shot
        mov [save_Background_var],1
        add [shot_x],3
        MSaveBkGround shot_x, shot_y, temp_shot
        MDrawImage shot_x, shot_y, BossShot
        jmp contShotB
LeftBshot:
        mov [save_Background_var],0
        MDrawImage shot_x, shot_y, temp_shot
        mov [save_Background_var],1
        sub [shot_x],3
        MSaveBkGround shot_x, shot_y, temp_shot
        MDrawImage shot_x, shot_y, BossShot
        jmp contShotB
```

```
endShotB:
        mov [inshotB],0
        mov [shotB_counter],40
        mov [save_Background_var],0
        MDrawImage shot_x, shot_y, temp_shot
        mov [save_Background_var],1
contShotB:
        mov [w_img],33
        mov [h_img],28
ret
endp shotBoss
;-----------------------------------------
; clearParticals - clear all the Scraps from the screen by
; draw the pcx photo of the room
;-----------------------------------------
; Input:
;       none
; Output:
;       clear the scraps
; Registers
;       ax
;-----------------------------------------
proc clearParticals near
        push [StartPictX]
        pop [current_x_player]
        push [StartPictY]
        pop [current_y_player]

        cmp [room],3
        je room3_nowClear
        cmp [room],2
        je room2_nowClear
        jmp @@clearCont
room3_nowClear:
        cmp [bossExist],1
        jne room3_LabelC
        call DrawRoom3C
        push [current_x_player]
        pop [StartPictX]                ;set the start place of the player
        push [current_y_player]
        pop [StartPictY]
        jmp @@clearCont
```

```
room:ClebaL_3
        call DrawRoom3O
        push [current_x_player]
        pop [StartPictX]                reyalp eht fo ecalp trats eht tes;
        push [current_y_player]
        pop [StartPictY]                reyalp eht fo ecalp trats eht tes;
        jmp @@clearCont
roommoor tsrfi;    :raelCwon_2
        cmp [enemyExist],1
        jne roomClebaL_2
        call DrawRoom2C
        push [current_x_player]
        pop [StartPictX]                reyalp eht fo ecalp trats eht tes;
        push [current_y_player]
        pop [StartPictY]                reyalp eht fo ecalp trats eht tes;
        jmp @@clearCont
room:ClebaL_2
        call DrawRoom2O
        push [current_x_player]
        pop [StartPictX]                reyalp eht fo ecalp trats eht tes;
        push [current_y_player]
        pop [StartPictY]                reyalp eht fo ecalp trats eht tes;
                        ;set the start place of the player
@@clearCont:
        ret
endp clearParticals
```

```
-----------------------------------------
 ;beep_M
-----------------------------------------;
 ;Input:
 ;      none
 ;Output:
 ;      output beep
 ;Registers
;       ax
-----------------------------------------;
proc beep_M  near
      mov al,10110110b
      out 43h,al ;set chanel to 3
      mov al,0cah

      out 42h,al  ;set frequency
      mov al,11h
      out 42h,al
      in al,61h

      or al,00000011b  ;turn on speaker
      out 61h,al

;wait
    mov cx,001h      tiaw ot dnoces orcim  | ;
    mov dx,0A120h | ;
    mov ah,86h
    int 15h  ;extended service

;close speaker
      in al,61h
      and al,11111100b
      out 61h,al
      ret
endp beep_M
```

```
-----------------------------------------;
 ;beep_M
-----------------------------------------;
 ;Input:
 ;      none
 ;Output:
 ;      output beep
 ;Registers
;       ax
-----------------------------------------;
proc beep_M2  near
     mov al,10110110b
     out 43h,al ;set chanel to 3

     mov al,004h
     out 42h,al  ;set frequency
     mov al,10h
     out 42h,al
     in al,61h

     or al,00000011b  ;turn on speaker
     out 61h,al

;wait
    mov cx,001h      tiaw ot dnoces orcim  | ;
    mov dx,0A120h | ;
    mov ah,86h
    int 15h  ;extended service

;close speaker
     in al,61h
     and al,11111100b
     out 61h,al
     ret
endp beep_M 2

               End Start
```

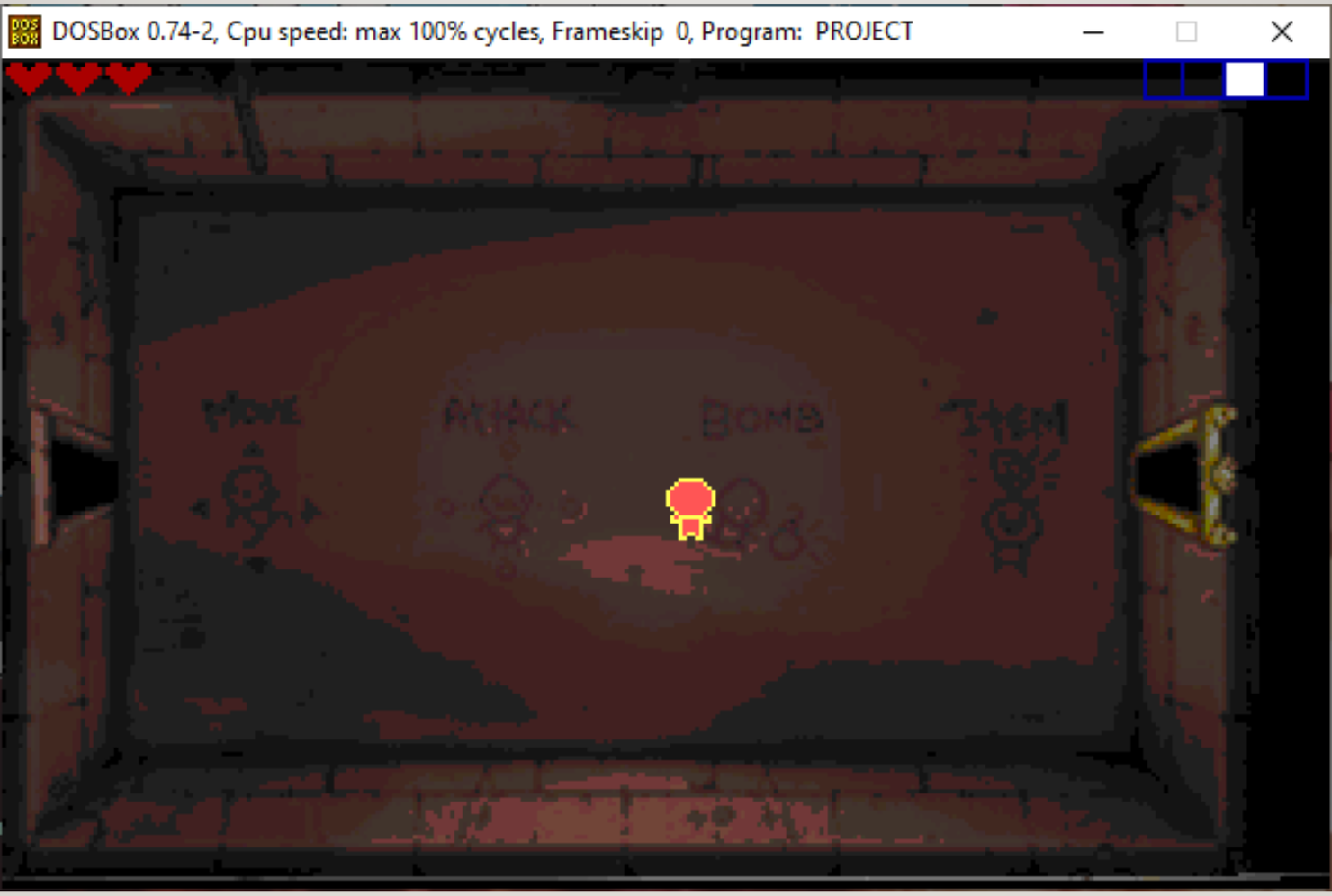
EXAMPLES FOR RUNNING

intro

MENU


DOSBox 0.74-2, Cpu speed: max 100% cycles, Frameskip 0, Program: PROJECT
NEW RUN
ABOUT ME
BACK
Esc
SELECT!
ENTER

FIRST ROOM

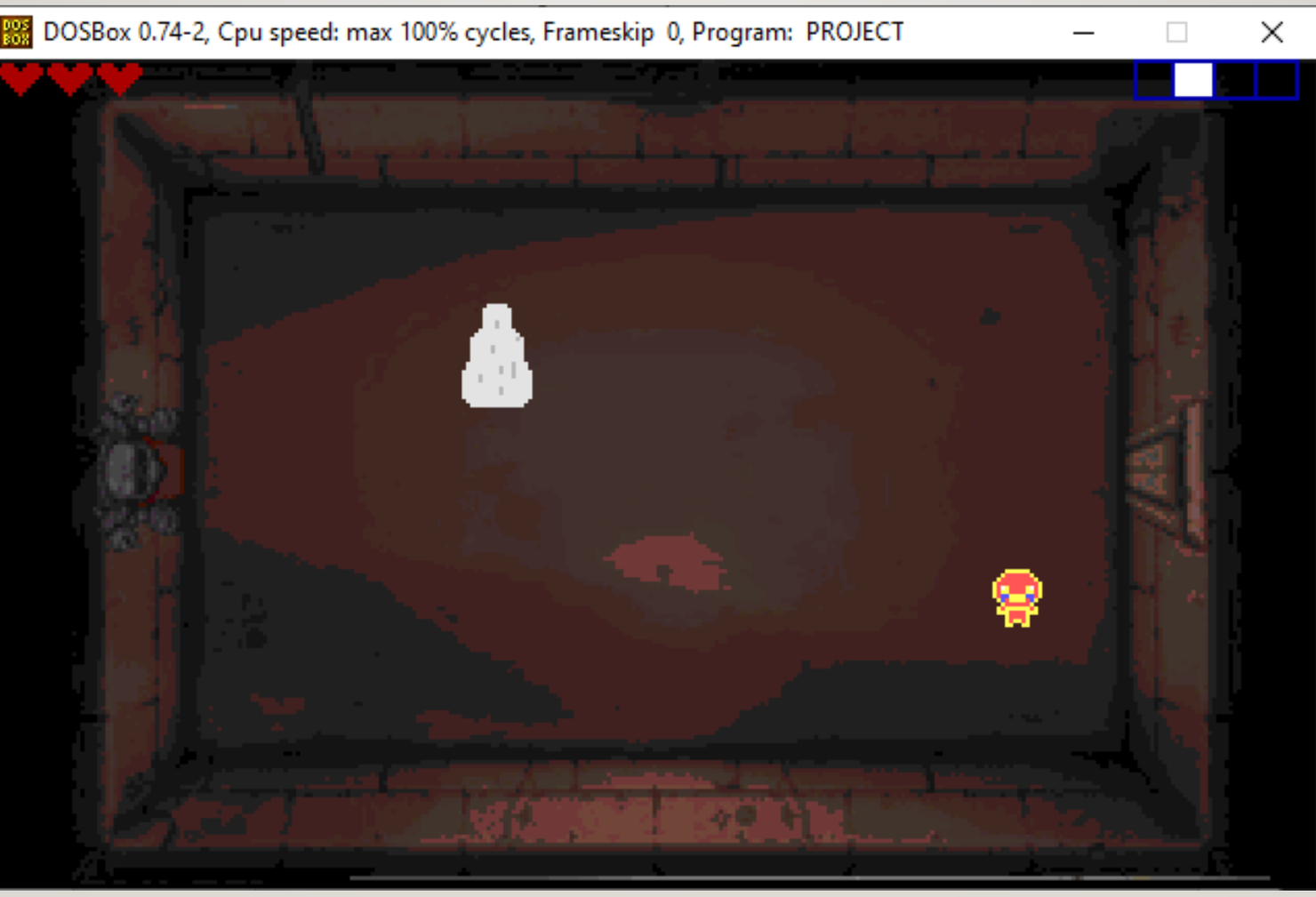

DOSBox 0.74-2, Cpu speed: max 100% cycles, Frameskip 0, Program: PROJECT
MOVE
ATTACK
BOMB
ITEM

ITEM ROOM

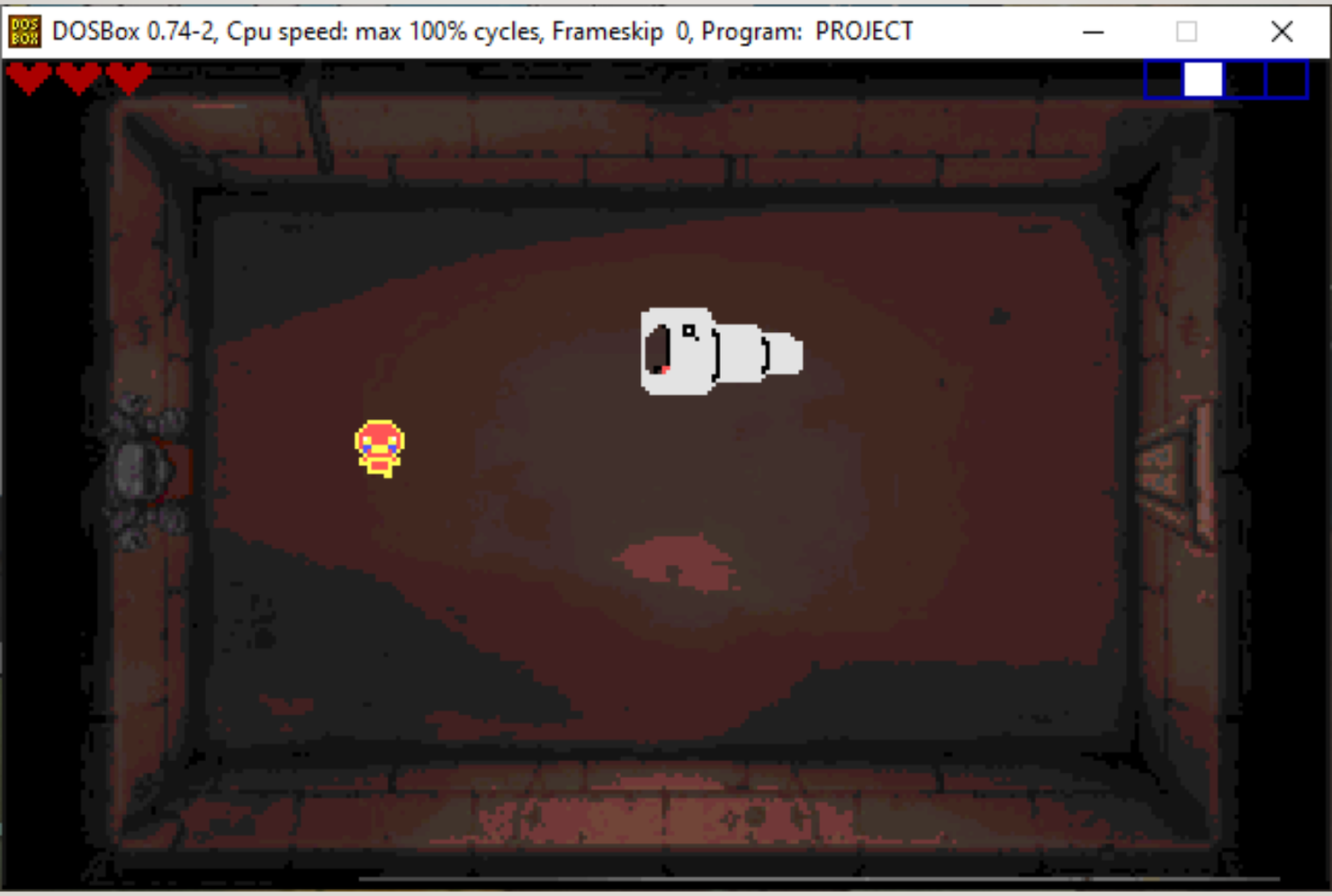

DOSBox 0.74-2, Cpu speed: max 100% cycles, Frameskip 0, Program: PROJECT

ITEM ROOM WITHOUT

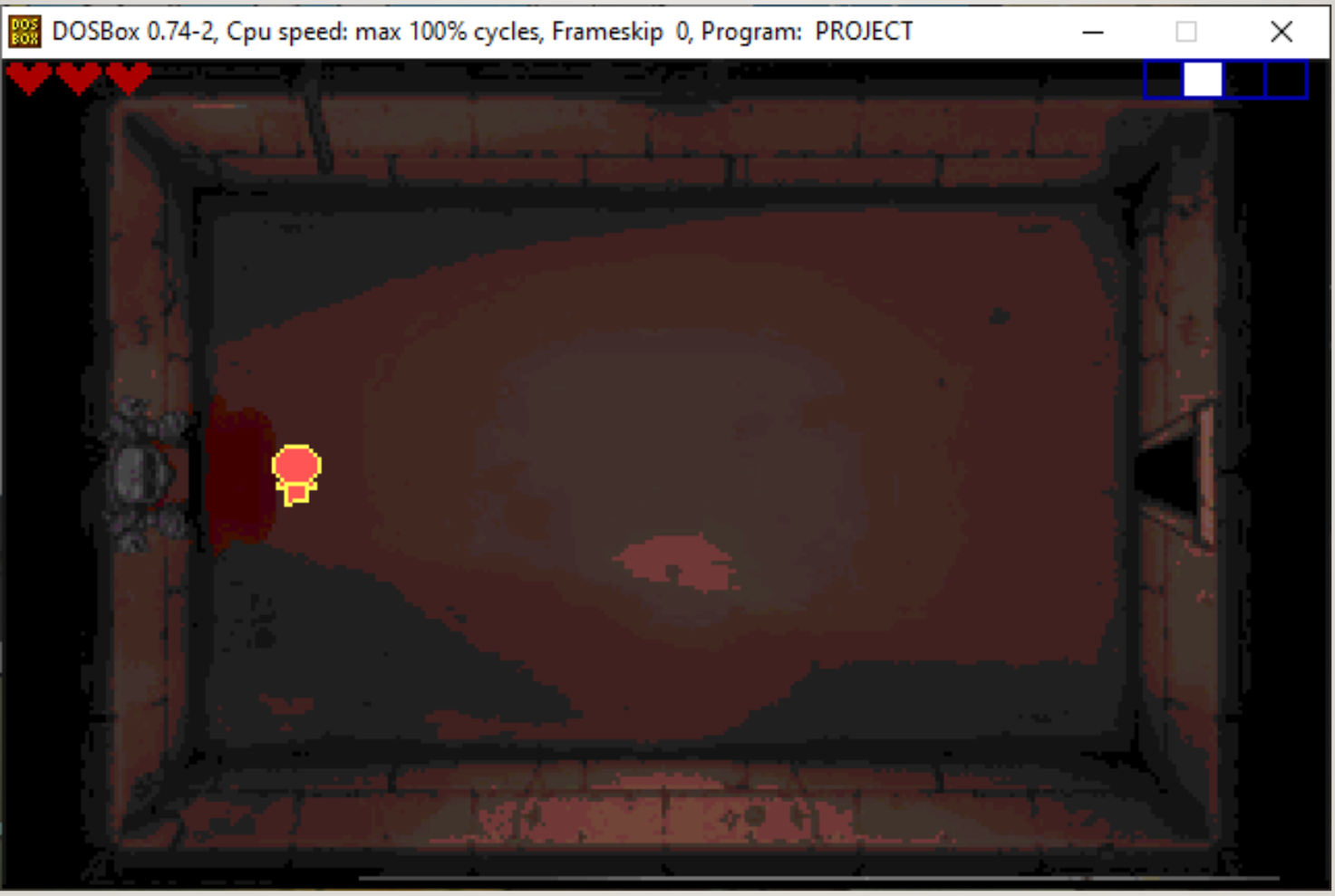
ENEMY ROOM


DOSBox 0.74-2, Cpu speed: max 100% cycles, Frameskip 0, Program: PROJECT

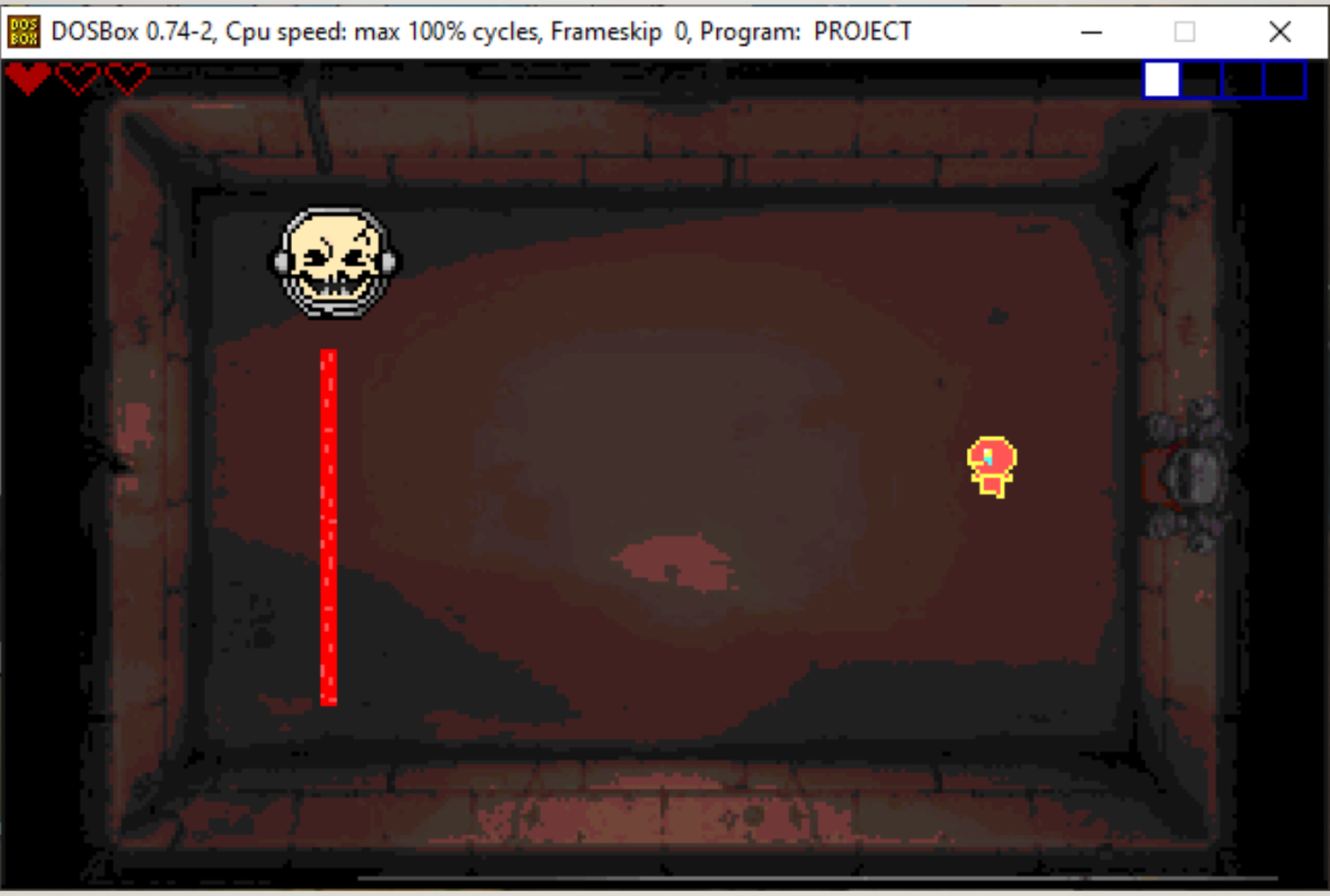
ENEMY ROOM DASH



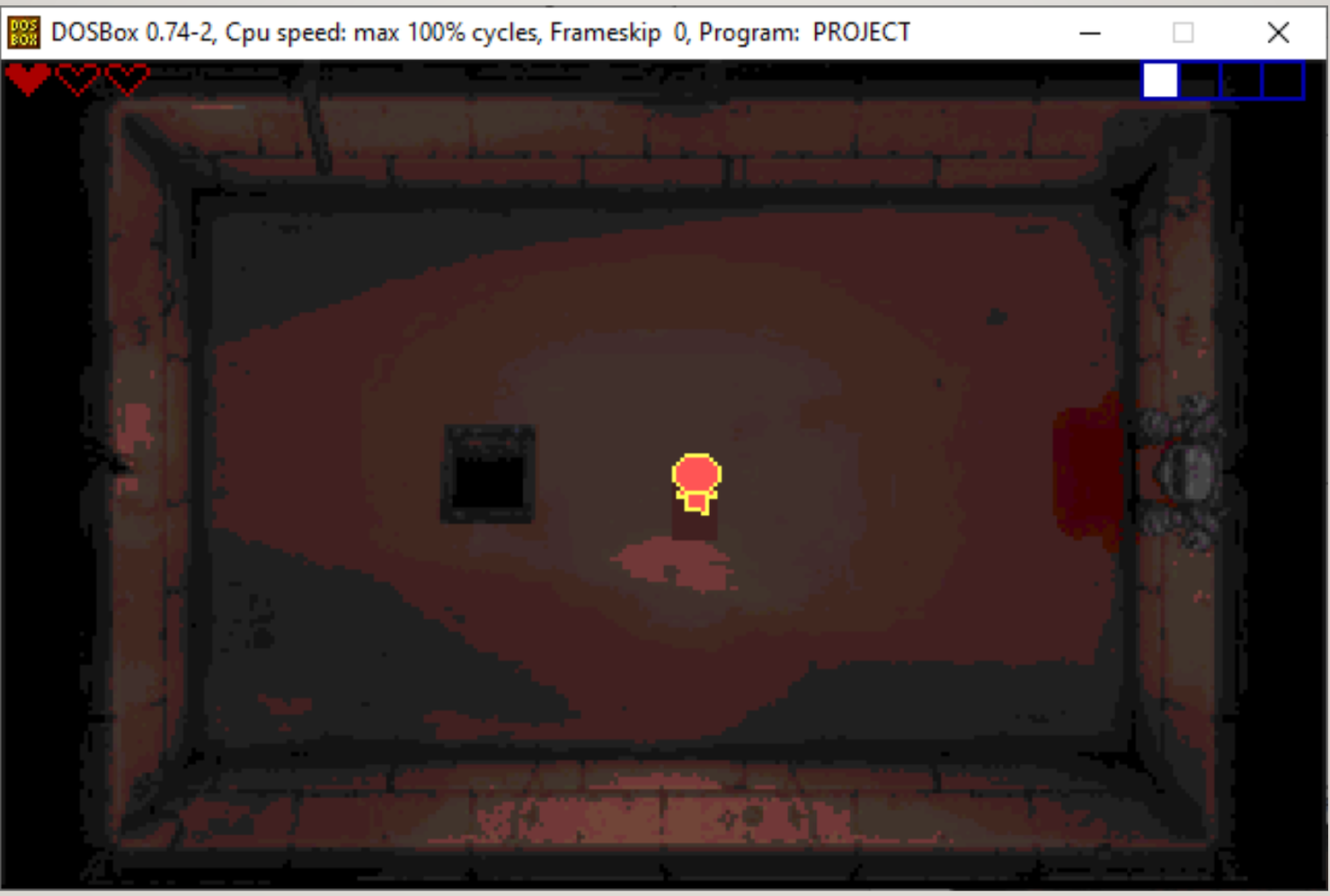
ENEMY ROOM WITHOUT ENEMY



BOOS ROOM


DOSBox 0.74-2, Cpu speed: max 100% cycles, Frameskip 0, Program: PROJECT

BOOS ROOM WITHOUT BOOS

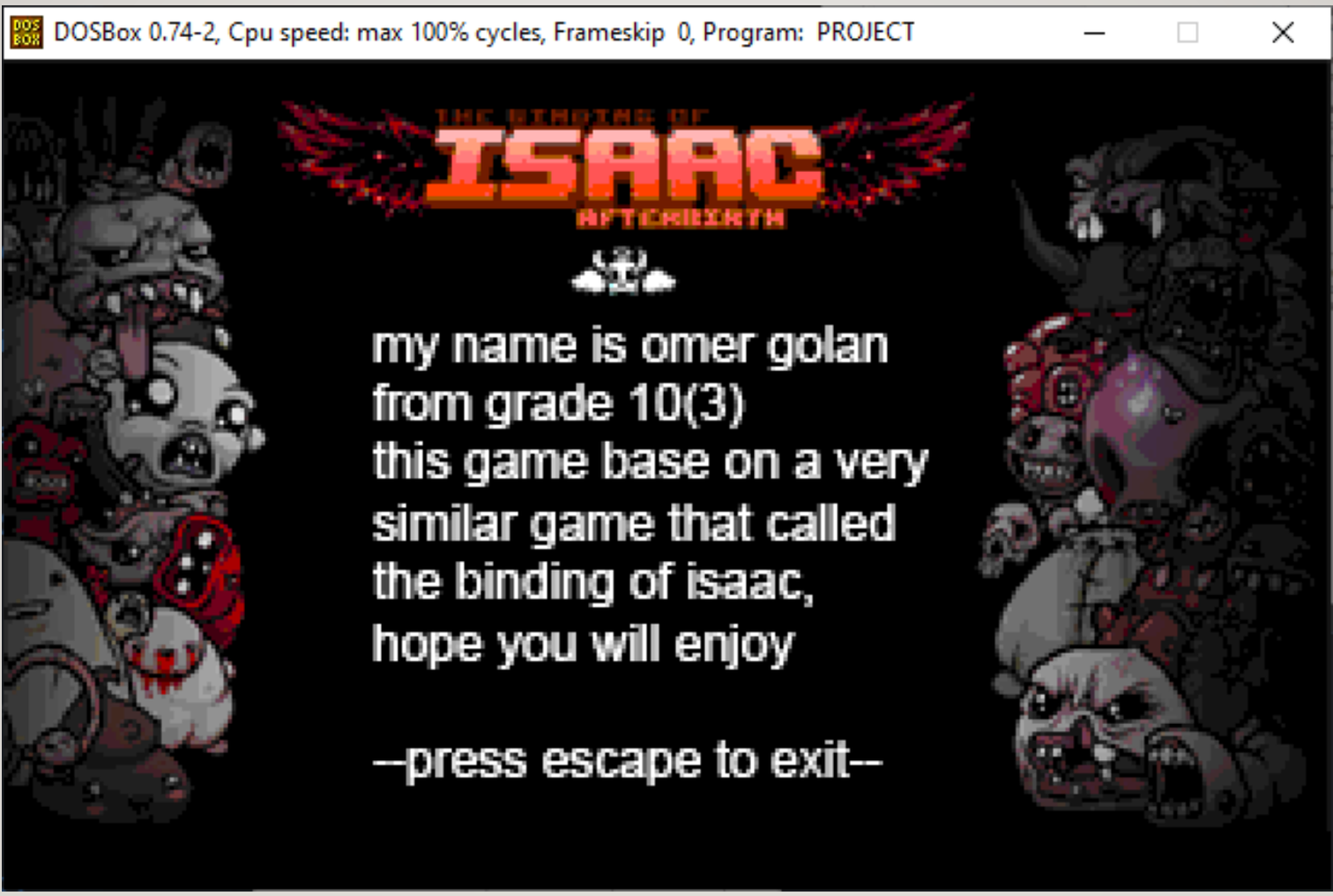


LOSE SCREEN


DOSBox 0.74-2, Cpu speed: max 100% cycles, Frameskip 0, Program: PROJECT
DEAR DIARY,
TODAY I DIED.
I WAS KILLED BY THIS THING →
IN SOME BASEMENT
I LEAVE ALL THAT I OWN TO MY CAT GUPPY
GOODBYE CRUEL WORLD
XOXO APOLLYON
EXIT
Esc
RESTART!
SPACE

VICTORY SCREEN

About me


DOSBox 0.74-2, Cpu speed: max 100% cycles, Frameskip 0, Program: PROJECT
THE BINDING OF
ISAAC
AFTERBIRTH
my name is omer golan
from grade 10(3)
this game base on a very
similar game that called
the binding of isaac,
hope you will enjoy
--press escape to exit--

Personal summary

- First of all, I would like to thanks to my teacher Anatoly and also to my friends and all the people that helped me in the process
- I enjoyed it a lot. and that's was very fun to make the game in my own
- We learn a lot about computers, thinking and how to work Ongoing work about the project
- I hope that more students will learn this so important and interesting language