Lets Make A Retro Game

Episode 12 – Player Collisions and Lives

In this episode we are going to cover two things, detecting whether any enemy objects hit the player ship as well as display the number of lives the players has remaining.

This will also include detecting when the player is out of lives and ending the game. We will enhance this some more in a future episode.

Player Collisions

First we need to see if any of the enemy objects have hit the player ship, the best place to do this is in our existing enemy object loop, located in the MOVE_ENEMIES subroutine.

Once little fix to start in this routine, as the amount of code has increased some of our previous relative jumps will be out of range i.e. the jump will be too far. Find the label ME1 and just after replace the JR Z,ME2 with the following:

```
JP Z,ME2
```

Now find the code where we detect whether the enemy object has reached the bottom of the screen i.e.

```
; fixed increase for now
INC A
CP 150
JR C,ME3
; enemy has reached the bottom of the screen
```

Now after the CP 150 statement we want to add a section of code to see if the enemy object has hit the player as follows:

```
; enemy has reached the bottom of the screen
; check whether the enemy has hit the player ship
LD A,(SPRTBL+1)
SUB A,14 ; fixed width of enemy at the moment
CP (IX+1)
JR NC, ME7
; x + it's width is larger than the players X
ADD A,28
CP (IX+1)
JR C, ME7
; we should be hitting the player
; decrease the players life counter
LD A,(LIVES)
DEC A
; check we don't overflow
JR NC,ME9
XOR A
```

```
ME9:
    LD (LIVES),A
    ; at the moment we won't do any animation effect
    ; TODO: Animate players death

    ; continue on so that we finish our enemy loop and subroutine
    JR ME8
ME7:
    ; have not hit the player decrease score
```

And we need to add our label ME8 just after our decrease score section as follows:

```
    ; decrease score
    LD A,1
    CALL SCORESUB
    ; explosion?

ME8:
```

One more range fix, go to the bottom of our function and look for the DJNZ ME1 command just after a INC HL, we need to replace that with a normal jump as follows:

```
ME2:
    INC HL
    DEC B
    JP NZ,ME1
```

At the moment, we only have one type of enemy object so we will hardwire the width to 14 pixels, and as we only need to test one direction i.e. x position we do the test here in line rather than calling our collision testing function.

Display The Players Lives

Next we need to display the current number of players remaining lives on the screen, rather than display them with a number, I thought we would be a little bit more creative and display a graphic symbol of the players ship for each life.

We need to do a couple of bit of setup, 1st a new bit of memory. Go to the end of the main assembly file and find our memory declaration section as follows:

```
ORG RAMSTART

LEVEL:      DS 1
LIVES:      DS 1
```

Add one more line after LIVE as follows:

```
LASTLIVES:  DS 1
```

This is our indicator of what the number of lives were last time we wrote them to the screen, so we don't have to write them all the time.

As we have added a new variable we need to make sure that we set it to a known value, find our INITRAM function and update it as follows:

```
; Init Ram for a new game
INITRAM:
    LD A,3
    LD (LIVES),A
    LD HL,0
    LD (SCORE),HL
    LD (SCORE+1),HL
    LD A,1
    LD (LASTSCORE),A
    XOR A
    LD (LASTLIVES),A
    LD (LEVEL),A
    LD (ANIMATE),A
```

Next we need to display our lives on screen 1st with a DISPLAYLIVES function, there is an existing one there, from another one of my programs, replace it with the following code:

```
; Display the current player lives (max 7)
DISPLAYLIVES:
    LD HL,LIVES
    LD A,(LASTLIVES)
    CP (HL)
    RET Z
    ; clear current lives display
    LD HL,VRAM_NAME+688
    LD BC,14
    XOR A
    CALL FILVRM
    LD HL,VRAM_NAME+720
    LD BC,14
    XOR A
    CALL FILVRM
    ; now show the current lives
    ; - first write the top characters
    LD D,44
    LD HL,VRAM_NAME+688
    CALL SETWRT
    ; current # of lives
    LD A,(LIVES)
    ; max 7 to be displayed
    AND %111
    LD B,A
DL1:
    LD A,D
    OUT (DATA_PORT),A
    INC A
```

```
    OUT (DATA_PORT),A
    DJNZ DL1
    ; - now write the bottom characters
    LD D,42
    LD HL,VRAM_NAME+720
    CALL SETWRT
    ; current # of lives
    LD A,(LIVES)
    ; max 7 to be displayed
    AND %111
    LD B,A
DL2:
    LD A,D
    OUT (DATA_PORT),A
    INC A
    OUT (DATA_PORT),A
    DJNZ DL2
    LD A,(LIVES)
    LD (LASTLIVES),A
    RET
```

At the start of the routine, we check whether the current lives counter is different from the value in our LASTLIVES value, if it is different we know to update the lives display.

This routine basically draws two rows, forming the top and bottom tiles of the lives counter, with two tiles per player life. It makes sure it stops at a maximum of seven life counters or we would run out of screen space.

And of course at the end we store the current number of lives in LASTLIVES so we know we have written the lives to the screen.

One final thing we need to do is call our DISPLAYLIVES routine, and the best place to do that is during our vertical blank routine, just after our DISPLAYSCORE call as follows:

```
; This is our routine called every VDP interrupt during normal game
play
; - Do all VDP writes here to avoid corruption
VDU_WRITES:
    CALL DISPLAYSCORE
    CALL DISPLAYLIVES
    RET
```

Game Over

One more final touch is to end the game if the players lives reach zero, for this time we will simply exit back to the title screen. We will enhance this some more in a later episode.

Find our MLOOP section and just after the call to MOVE_ENEMIES we add some code to detect whether the player life counter has reached zero as follows:

```
    CALL MOVE_ENEMIES

    ; test to see if we have run out of lives
    ; TODO: Display G A M E  O V E R message, wait and new game
    LD A,(LIVES)
    CP 0
    JP Z,TITLESCREEN
```

One more thing before we finish, a little fix for when we do get a game over, we need to erase any sprites on screen when we display the title screen, as when we jump back there after running out of lives there will probably be sprites still on the screen. So just after our TITLESCREEN label add the following lines to clear any sprites from the screen as follows:

```
TITLESCREEN:
    ; display our title screen
    CALL DISABLE_NMI
    ; Clear the screen
    CALL CLEARPAT

    ; Clean up in case the game left anything on screen
    CALL CLEARSPRITES
    CALL SPRWRT
```