ADNAN MENDERES UNIVERSITY

ENGINEERING FACULTY

COMPUTER ENGINEERING

COMPUTER GAMES

JUMPMAN

Prepared by:

Ersu KARPUZ – 171805043

Adem Enes KESİCİ – 171805029

Advisor:

Dr. Lecturer Samsun Mustafa BAŞARICI

Content

1.1 History

Jumpman is a platform game written by Randy Glover and published by Epyx in 1983. It was first developed for the Atari 8-bit family, and versions were also released for the Commodore 64, Apple II, and IBM PC.

The game received very favorable reviews when it was released and was a major hit for its publisher, Automated Simulations. It was so successful that the company renamed itself Epyx, formerly their brand for action titles like Jumpman.

Re-creations on other platforms, and new levels for the original versions, continue to appear.

Jumpman was published on diskette, but a version of the game with 12 new levels instead of 30 was released on cartridge as Jumpman Junior. It was available on the Atari 8-bit computers, Commodore 64, and ColecoVision.

1.2 GamePlay

According to the story, the base on Jupiter has been sabotaged by terrorists who have placed bombs throughout the base's three buildings. The object of the game is to defuse all the bombs in a platform-filled screen. Jumpman defuses a bomb by touching it. Jumpman can jump, climb up and down ladders, and there are two kinds of rope each allowing a single direction of climbing only.

The game map is organized into a series of levels, representing the floors in three buildings. When all of the bombs on a level have been deactivated, the map scrolls vertically to show another floor of the building. When all of the levels in a building are complete, a screen shows the remaining buildings and moves onto the next one. The order of the maps is randomized so players do not end up trapped on a level they cannot complete.

Hazards include falling "smart darts" , fall damage, and other hazards that are unique to a certain level. Upon being hit or falling from a height, Jumpman tumbles down to the bottom of the screen, with a measure from Chopin's Funeral March being played.

Points are awarded for each bomb defused, with bonus points available for completing a level quickly. Jumpman's game run-speed can be chosen by the player, with faster speeds being riskier but providing greater opportunity to earn bonus points.

1.3 Development

Randy Glover was living in Foster City, California and had been experimenting with electronics when he saw his first computer in 1977 when he played Star Trek at a Berkeley university open house. This prompted him to purchase a Commodore PET in 1978,and then upgraded to a TRS-80 due to its support of a hard drive.

Jumpman came about after Glover saw Donkey Kong in a local Pizza Hut.This led him to become interested in making a version for home computers. He visited a local computer store who had the TI-99/4A and Atari 400. He initially purchased the TI-99 due to its better keyboard, but when he learned the graphics were based on character set manipulation, he returned it the next day and purchased the Atari.

The initial version was written using a compiler on the Apple II, moving the software to the Atari. A prototype with 13 levels took four or five months to complete. After looking in the back of a computer magazine for publisher, in early 1983 he approached Broderbund. They were interested but demanded that their programmers be allowed to work on it. The next day he met with Automated Simulations, who were much more excited by the game and agreed to allow Glover to complete it himself.

At the time, the company was in the process of moving from the strategy game market to action titles, which they released under their Epyx brand. Jumpman was the perfect title for the brand, and the company hired him. Aiming the game at the newly enlarged RAM available on the Atari 800 led to the 32 levels of the final design. The Atari release was a huge hit, and the company soon abandoned their strategic games and renamed as Epyx. Glover then moved on to a C64 port, which was not trivial due to a particular feature of the Atari hardware Glover used to ease development.

Other programmers at Epyx ported it to the Apple II, with poor results,and a year later, contracted Mirror Images Software for an IBM PC/PCjr port. The Atari and Commodore versions were released on disk and cassette tape, the Apple and IBM versions only on disk. The Atari version used a classic bad-sector method of preventing copying, but this had little effect on piracy.

After developing the original versions, Glover moved on to Jumpman Junior, a cartridge title with only 12 levels. He stated that it wasn't really a sequel to Jumpman, but more of a "lite" version for Atari and Commodore users who didn't have disk drives. These versions removed the more complex levels and any code needed to run them. Two of its levels were turned into Sreddal and Fire! Fire! on the latter. The C64 version was later ported to the ColecoVision, which used the C64 levels.

Glover continued working at Epyx, working on the little-known Lunar Outpost and the swimming section of Summer Games. He remained at the company for about two years before returning to the cash register business.

1.4 Technical Details

Movement was controlled through the collision detection system of the Atari's player/missile graphics hardware. This system looks for overlap between the sprites and the background, setting registers that indicate which sprite had touched which color.

Glover separated the Jumpman sprite into two parts, the body and the feet. By examining which of these collided, the engine could determine which direction to move. For instance, if both the body and feet collided with the same color, it must be a wall, and the Jumpman should stop moving. If there was no collision with either his feet or body, Jumpman is unsupported and should fall down the screen. Variations on these allowed support for ramps, ropes, and other features.

This not only saved processing time comparing the player location to an in-memory description of the map, but also meant that maps could be created simply by drawing with them and experimenting with the results in the game.

1.5 Legacy

In 1998, Randy Glover became aware of the many fans of Jumpman and started working on Jumpman II, keeping a development diary at the now defunct jumpman2.com site. The last recorded diary entry was made in 2001.

In 2008, the original Jumpman was released on the Wii's Virtual Console.

In 2014, Midnight Ryder Technologies shipped Jumpman Forever for the OUYA micro-console, with planned releases for PC, Mac, iOS, and Android platforms. Originally titled Jumpman: 2049, the game is considered to be an official sequel based on rights given to Midnight Ryder Technologies back in 2000 by Randy Glover.

In 2018 Jumpman was re-released on THEC64 Mini as one of the 64 built in games. A game called Jumpman 2 was also on the machine, but in reality is just Jumpman Junior rebranded.

2. Piece of Codes

Firstly, We added variables and screen size.

```python
pygame.init()

tile_size = 25
game_over = 0
main_menu = True
score = 0
screen_width = 800
screen_height = 600
screen = pygame.display.set_mode((screen_width, screen_height))

pygame.display.set_caption("JUMPMAN")
font = pygame.font.SysFont("Retro Gaming", 20)
titleFont = pygame.font.SysFont("Retro Gaming", 90)
authorFont = pygame.font.Font(None, 18)
background = pygame.Surface(screen.get_size())
player_sprite = pygame.sprite.Group()
enemy_group = pygame.sprite.Group()
coin_group = pygame.sprite.Group()
ladder_group = pygame.sprite.Group()
clock = pygame.time.Clock()
exit_program = False
initial_screen = True
```

2.1 Load Button Images

```python
b_screen = pygame.image.load("images/b_screen.jpg")
restart_img = pygame.image.load("images/restart_btn.png")
level1_img = pygame.image.load("images/Level1_img.png")
level1_img = pygame.transform.scale(level1_img, (240, 80))
level2_img = pygame.image.load("images/Level2_img.png")
level2_img = pygame.transform.scale(level2_img, (240, 80))
level3_img = pygame.image.load("images/Level3_img.png")
level3_img = pygame.transform.scale(level3_img, (240, 80))
exit_img = pygame.image.load("images/exit_btn.png")
main_menu_img = pygame.image.load("images/main_menu_img.png")
```

2.2 Create Button Class

```
class Button():
    def __init__(self, x, y, image):
        self.image = image
        self.rect = self.image.get_rect()
        self.rect.x = x
        self.rect.y = y
        self.clicked = False

    def draw(self):
        action = False

        # get mouse position
        pos = pygame.mouse.get_pos()

        # check mouseover and clicked conditions
        if self.rect.collidepoint(pos):
            if pygame.mouse.get_pressed()[0] == 1 and self.clicked == False:  # 0
is left mouse click
                action = True
                self.clicked = True
        if pygame.mouse.get_pressed()[0] == 0:
            self.clicked = False

        # draw button
        screen.blit(self.image, self.rect)

        return action
```

2.3 Create Player Class

2.3.1 Creat Update Function

We added keypress for player also animatons , gravity and collision with blocks, enemys, ladders.

Finally, We did update player coordinates for this function.

```python
def update(self, game_over):
    dx = 0
    dy = 0
    walk_cooldown = 10  # walk cooldown

    if game_over == 0:

        # get keypresses
        key = pygame.key.get_pressed()
        if key[pygame.K_SPACE] and self.jumping == False and self.in_air == False:
            self.vel_y = -10
            self.jumping = True
        if not key[pygame.K_SPACE]:
            self.jumping = False
        if key[pygame.K_LEFT]:
            dx -= 5
            self.direction = -1
            self.counter += 1
        if key[pygame.K_RIGHT]:
            dx += 5
            self.counter += 1
            self.direction = 1

        if key[pygame.K_LEFT] == False and key[pygame.K_RIGHT] == False:
            self.counter = 0
            self.index = 0
            if self.direction == 1:
                self.image = self.images_right[self.index]
            if self.direction == -1:
                self.image = self.images_left[self.index]

        # animation
        if self.counter > walk_cooldown:
            self.counter = 0
            self.index += 1
            if self.index >= len(self.images_left):
                self.index = 0
            if self.direction == 1:
                self.image = self.images_right[self.index]
            if self.direction == -1:
                self.image = self.images_left[self.index]

        # add gravity
        self.vel_y += 1
        if self.vel_y > 10:
           self.vel_y = 10
        dy += self.vel_y
        # check for collision
        self.in_air = True
        for tile in world.tile_list:
            # check for collision in x direction
            if tile[1].colliderect(self.rect.x + dx, self.rect.y, self.width,
self.height):
                dx = 0
            # check for collision in y direction
            if tile[1].colliderect(self.rect.x, self.rect.y + dy, self.width,
self.height):
                # check if below the ground i.e. jumping
                if self.vel_y < 0:
                    dy = tile[1].bottom - self.rect.top
                    self.vel_y = 0
                # check if above the ground i.e. falling
                elif self.vel_y >= 0:
                    dy = tile[1].top - self.rect.bottom
                    self.vel_y = 0
                    self.in_air = False

        # check for collision with enemies
        if pygame.sprite.spritecollide(self, enemy_group, False):
            game_over = -1

        # check for collision and able to climb to ladder
        if pygame.sprite.spritecollide(self, ladder_group, False):
            player.vel_y = 0
            player.in_air = True
            key = pygame.key.get_pressed()
            if key[pygame.K_UP]:
                self.image = self.climbing_img
                dy -= 3
                self.vel_y = 0
            if key[pygame.K_DOWN]:
                self.image = self.climbing_img
                dy += 5
                self.vel_y = 0

        # update player coordinates
        self.rect.x += dx
        self.rect.y += dy

    elif game_over == -1:
        self.image = self.dead_image


    return game_over
```

2.3.2 Create Reset Function

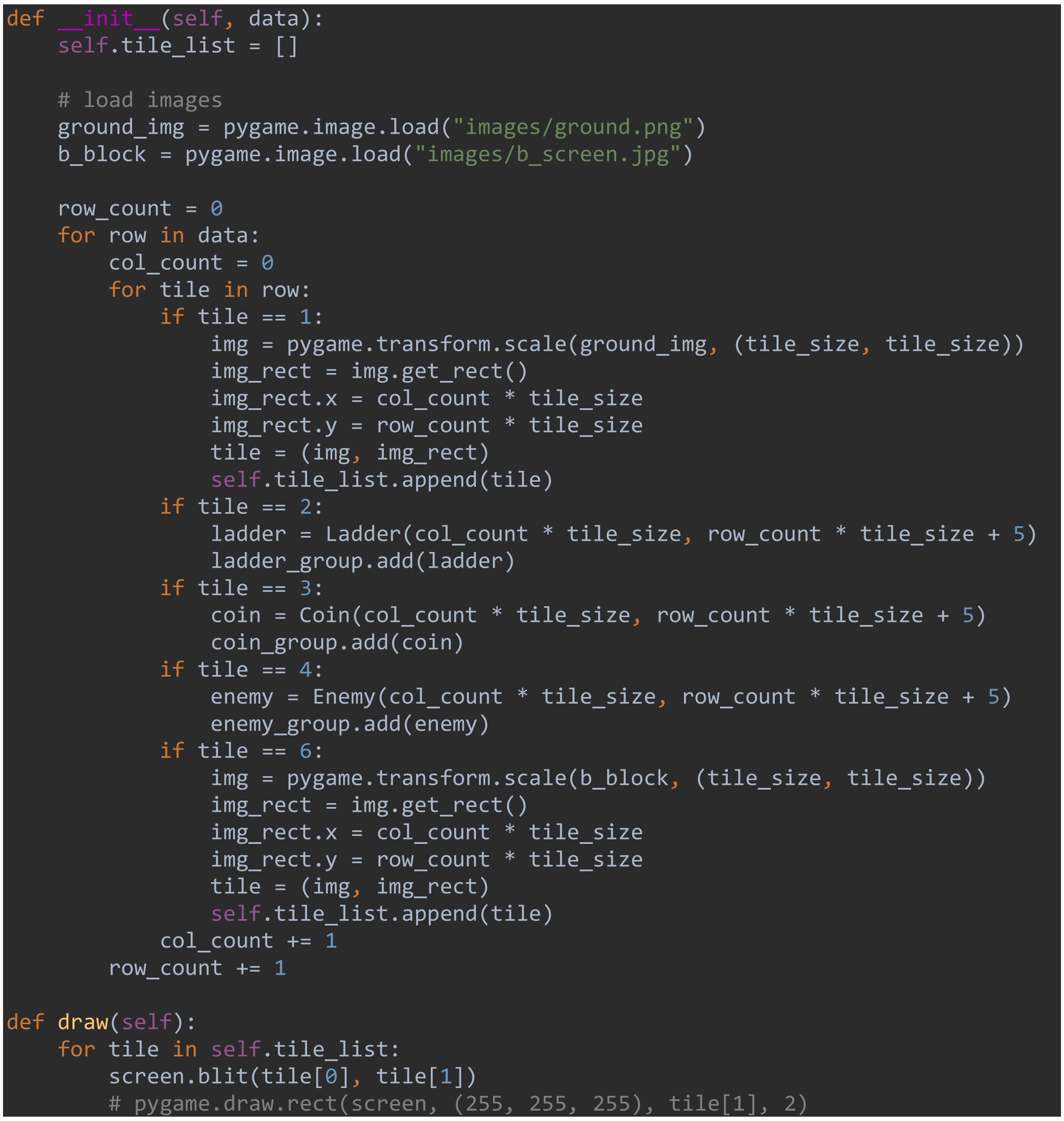
We made the character's animation pictures by creating a value called char num and looping the num values and making the character move in this way.

```
def reset(self, x, y):
    Sprite.__init__(self)
    self.images_left = []
    self.images_right = []
    self.index = 0
    self.counter = 0
    for num in range(1, 4):
        img_left = pygame.image.load(f"images/char{num}.png")
        img_left = pygame.transform.scale(img_left, (tile_size, tile_size * 2))
        img_right = pygame.transform.flip(img_left, True, False)
        self.images_left.append(img_left)
        self.images_right.append(img_right)
    self.dead_image = pygame.image.load("images/deadchar.png")
    self.dead_image = pygame.transform.scale(self.dead_image, (tile_size,
tile_size * 2))
    self.climbing_img = pygame.image.load("images/charOnLadder.png")
    self.climbing_img = pygame.transform.scale(self.climbing_img, (tile_size,
tile_size * 2))
    self.image = self.images_left[self.index]
    self.rect = self.image.get_rect()
    self.rect.x = x
    self.rect.y = y
    self.width = self.image.get_width()
    self.height = self.image.get_height()
    self.screenheight = pygame.display.get_surface().get_height()
    self.screenwidth = pygame.display.get_surface().get_width()
    self.jumping = False
    self.vel_x = 0
    self.vel_y = 0
    self.direction = 0
    self.in_air = True
```

2.4 Create World Class

We added the rows and columns in the for loop and draw the map with the tile method.

tile1 = ground , tile2 = ladder , tile3 = coin , tile4 = enemy, tile6 = barrierblock



```python
def __init__(self, data):
    self.tile_list = []

    # load images
    ground_img = pygame.image.load("images/ground.png")
    b_block = pygame.image.load("images/b_screen.jpg")

    row_count = 0
    for row in data:
        col_count = 0
        for tile in row:
            if tile == 1:
                img = pygame.transform.scale(ground_img, (tile_size, tile_size))
                img_rect = img.get_rect()
                img_rect.x = col_count * tile_size
                img_rect.y = row_count * tile_size
                tile = (img, img_rect)
                self.tile_list.append(tile)
            if tile == 2:
                ladder = Ladder(col_count * tile_size, row_count * tile_size + 5)
                ladder_group.add(ladder)
            if tile == 3:
                coin = Coin(col_count * tile_size, row_count * tile_size + 5)
                coin_group.add(coin)
            if tile == 4:
                enemy = Enemy(col_count * tile_size, row_count * tile_size + 5)
                enemy_group.add(enemy)
            if tile == 6:
                img = pygame.transform.scale(b_block, (tile_size, tile_size))
                img_rect = img.get_rect()
                img_rect.x = col_count * tile_size
                img_rect.y = row_count * tile_size
                tile = (img, img_rect)
                self.tile_list.append(tile)
            col_count += 1
        row_count += 1

def draw(self):
    for tile in self.tile_list:
        screen.blit(tile[0], tile[1])
        # pygame.draw.rect(screen, (255, 255, 255), tile[1], 2)
```

2.5 Create Ladder Class

We added ladder image.

```
class Ladder(Sprite):
    def __init__(self, x, y):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load("images/ladder.png")
        self.image = pygame.transform.scale(self.image, (tile_size, tile_size))
        self.rect = self.image.get_rect()
        self.rect.x = x
        self.rect.y = y
```

2.6 Create Coin Class

We added coin and the icon that represents the score.

```
class Coin(Sprite):
    def __init__(self, x, y):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load("images/coin.png")
        self.image = pygame.transform.scale(self.image, (tile_size , tile_size ))
        self.rect = self.image.get_rect()
        self.rect.x = x
        self.rect.y = y


# create dummy coin for the score
score_coin = Coin(tile_size // 2 - 3, tile_size // 2 - 9)
coin_group.add(score_coin)
```

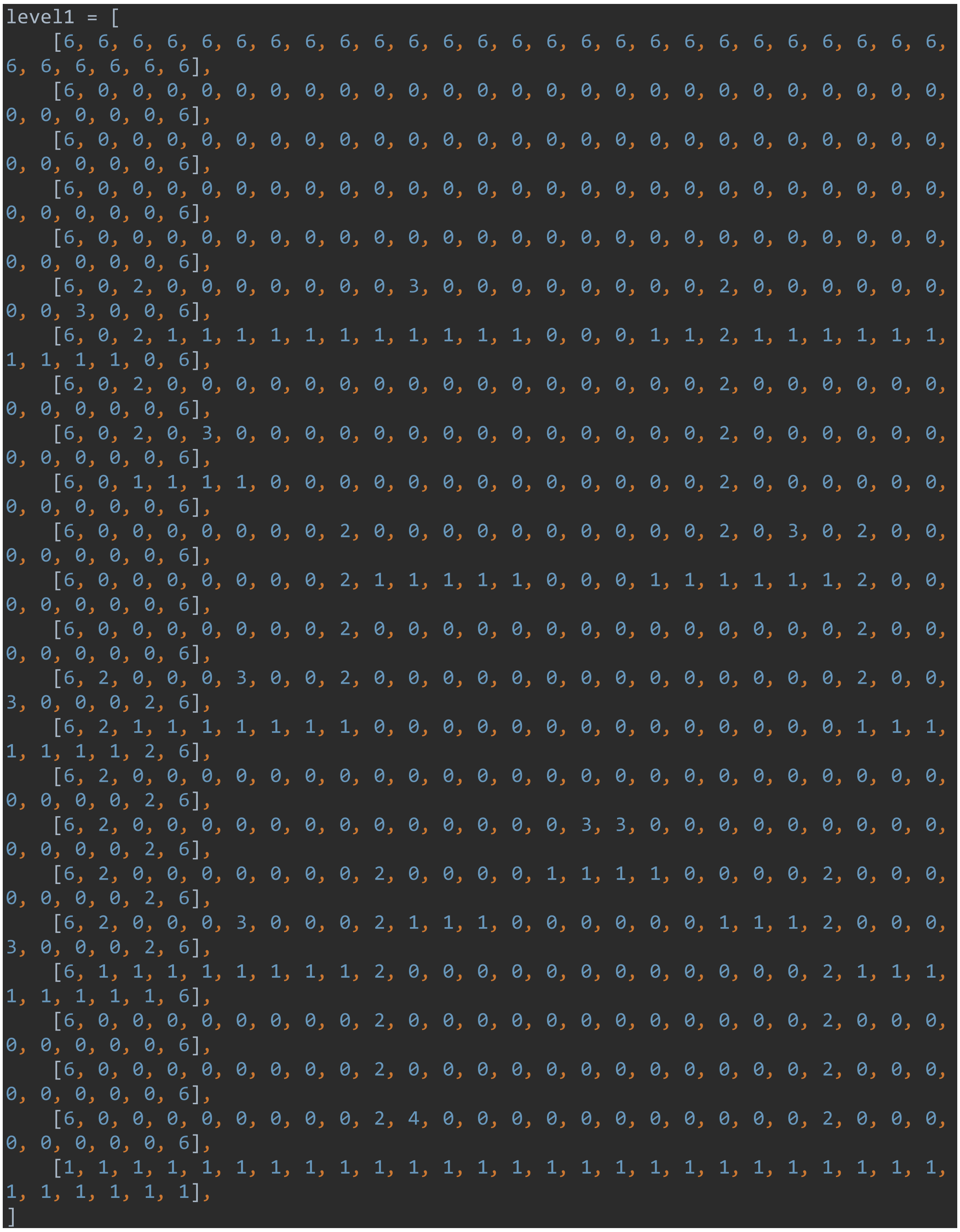
2.7 Create Enemy Class

We added enemy and update function. In this function, we gave the enemy the ability to move in the x-axis.

```
class Enemy(Sprite):
    def __init__(self, x, y):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load("images/enemy.png")
        self.image = pygame.transform.scale(self.image, (25, 19))
        self.rect = self.image.get_rect()
        self.rect.x = x
        self.rect.y = y
        self.move_direction = 1
        self.move_counter = 0

    def update(self):
        self.rect.x += self.move_direction
        self.move_counter += 1
        if abs(self.move_counter) > 50:
            self.move_direction *= -1
            self.move_counter *= -1
```

We created levels on the world. These are the matrices of the levels we created. Each number corresponds to the number in the tile method that we specified in the world class and also each number corresponds to certain picture.



```
level1 = [
    [6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
6, 6, 6, 6, 6, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 2, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0,
0, 0, 3, 0, 0, 6],
    [6, 0, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 2, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 0, 6],
    [6, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 2, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 3, 0, 2, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 2, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 2, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 2, 0, 0, 0, 3, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0,
3, 0, 0, 0, 2, 6],
    [6, 2, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
1, 1, 1, 1, 2, 6],
    [6, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 2, 6],
    [6, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 2, 6],
    [6, 2, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 2, 0, 0, 0,
0, 0, 0, 0, 2, 6],
    [6, 2, 0, 0, 0, 3, 0, 0, 0, 2, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 2, 0, 0, 0,
3, 0, 0, 0, 2, 6],
    [6, 1, 1, 1, 1, 1, 1, 1, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 1, 1,
1, 1, 1, 1, 1, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 0, 2, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1],
]
```

```
level2 = [
    [6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
6, 6, 6, 6, 6, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 2, 0, 3, 0, 0, 0, 0, 4, 0, 0, 3, 0, 0, 0, 0, 0, 3, 2, 0, 0, 4, 0, 0, 0,
0, 0, 0, 3, 0, 6],
    [6, 0, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 2, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 0, 6],
    [6, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 2, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 2, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 2, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 2, 0, 0, 0, 3, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0,
3, 0, 0, 0, 2, 6],
    [6, 2, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
1, 1, 1, 1, 2, 6],
    [6, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 2, 6],
    [6, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 2, 6],
    [6, 2, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 2, 0, 0, 0,
0, 0, 0, 0, 2, 6],
    [6, 2, 0, 0, 0, 3, 0, 0, 0, 2, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 2, 0, 0, 0,
3, 0, 0, 0, 2, 6],
    [6, 1, 1, 1, 1, 1, 1, 1, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 1, 1,
1, 1, 1, 1, 1, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0,
0, 0, 0, 3, 0, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0,
0, 4, 0, 0, 0, 6],
    [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1],
]
```

```
level3 = [
    [6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
6, 6, 6, 6, 6, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 0, 0, 0, 0, 3, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 3, 0, 0, 0, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 2, 0, 0, 0, 4, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 4, 0, 0, 0, 0,
4, 0, 0, 0, 0, 6],
    [6, 0, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 2, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 0, 6],
    [6, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 4, 0, 0, 2, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 2, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 2, 0, 0,
0, 3, 0, 0, 0, 6],
    [6, 0, 0, 0, 3, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 2, 0, 0, 0, 4, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0,
4, 0, 0, 0, 2, 6],
    [6, 2, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
1, 1, 1, 1, 2, 6],
    [6, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 2, 6],
    [6, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 2, 6],
    [6, 2, 0, 0, 0, 0, 0, 3, 0, 2, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 2, 0, 0, 0,
0, 3, 0, 0, 2, 6],
    [6, 2, 0, 0, 0, 4, 0, 0, 0, 2, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 2, 0, 0, 0,
0, 0, 0, 0, 2, 6],
    [6, 1, 1, 1, 1, 1, 1, 1, 1, 2, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 2, 1, 1, 1,
1, 1, 1, 1, 1, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [6, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 4, 0, 0, 0, 0, 0, 4, 0, 0, 2, 0, 0, 0,
0, 0, 0, 0, 0, 6],
    [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1],
]
```

We created the level button in the main menu.

```
# create buttons
restart_button = Button(screen_width // 2 - 150, screen_height // 2 + 100,
restart_img)
level1_button = Button(screen_width // 2 - 300, screen_height - 360, level1_img)
level2_button = Button(screen_width // 2 - 300, screen_height - 280, level2_img)
level3_button = Button(screen_width // 2 - 300, screen_height - 200, level3_img)
exit_button = Button(screen_width // 2 + 50, screen_height // 2, exit_img)
main_menu_button = Button(screen_width // 2, screen_height // 2 + 100,
main_menu_img)
```

2.8 Main Loop

We limited the fps.

We did Main Menu. We added levels and exit buttons in main menu

```
if main_menu == True:
    title = titleFont.render("JUMPMAN", True, (77, 166, 48))
    titlePos = title.get_rect(centerx=background.get_width() / 2, centery=200)
    screen.blit(title, titlePos)

    author = authorFont.render("This game completely codded by Ersu KARPUZ and
Adem Enes KESİCİ", True, white)
    authorPos = author.get_rect(centerx=background.get_width() -
author.get_width() / 2 - 10,
                                centery=background.get_height() -
author.get_height() / 2 - 10)
    screen.blit(author, authorPos)
    if exit_button.draw():
        exit_program = True
    if level1_button.draw():
        game_over = 0
        world = World(level1)
        main_menu = False
    if level2_button.draw():
        game_over = 0
        world = World(level2)
        main_menu = False
    if level3_button.draw():
        game_over = 0
        world = World(level3)
        main_menu = False
else:

    world.draw()
```

We have added the win condition and score counting (by collision) in this section also We draw all spriters.

```
if game_over == 0:
    enemy_group.update()

    # draw sprites
    coin_group.draw(screen)
    ladder_group.draw(screen)
    enemy_group.draw(screen)

    # update score
    if pygame.sprite.spritecollide(player, coin_group, True):
        score += 1
    img = font.render("X " + str(score), True, white)
    screen.blit(img, (tile_size * 2 - 10, 10))
    # if player collect 10 coin player will win that level
    if score == 10:
        img = font.render(" YOU WİN!! ", True, white)
        screen.blit(img, (screen_width // 2 - 100, screen_height // 2))
        if main_menu_button.draw():
            main_menu = True
            game_over = 1
            score = 0
        if restart_button.draw():
            player.reset(50, 550)
            game_over = 0
            score = 0


game_over = player.update(game_over)
```

We added 2 buttons that appear when the character dies.

```
# if player is dead
if game_over == -1:
    if main_menu_button.draw():
        main_menu = True
        game_over = 0
    if restart_button.draw():
        coin_group.draw(screen)
        player.reset(50, 550)
        game_over = 0
        score = 0

player_sprite.draw(screen)
```