**Building a Game Using Pygame**

**Introduction to Programming**

**Final Semester Project**

**Bakhtawar Baloch**

**SP19-BSSE-0043**

**Spring - CM**

**Abstract**

This report is written in support to the orientation of the python game using Pygame. It has a properly formatted code which have a proper display using blocks. The block of the user is supposed to watchout the enemies and find a way out of their way.

**Introduction**

This is a final semester project and was coded with quite a lot of interest, especially in learning the usage of new libraries such as Pygame.

**Methodology**

A few libraries were installed including Pygame, random and sys. Then using tuple proper color scheme was laid out which would form the display of the game. Tyhen according to different scores the game would either continue or a new game would start if the user fails in his attempt. Different classes were introduced such as:

def set\_level(score, SPEED):

    if score < 20:

        SPEED = 5

    elif score < 40:

        SPEED = 8

    elif score < 60:

        SPEED = 12

    else:

        SPEED = 15

    return SPEED

    # SPEED = score/5 + 1

def drop\_enemies(enemy\_list):

    delay = random.random()

    if len(enemy\_list) < 10 and delay < 0.1:

        x\_pos = random.randint(0,WIDTH-enemy\_size)

        y\_pos = 0

        enemy\_list.append([x\_pos, y\_pos])

def draw\_enemies(enemy\_list):

    for enemy\_pos in enemy\_list:

        pygame.draw.rect(screen, BLUE, (enemy\_pos[0], enemy\_pos[1], enemy\_size, enemy\_size))

def update\_enemy\_positions(enemy\_list, score):

    for idx, enemy\_pos in enumerate(enemy\_list):

        if enemy\_pos[1] >= 0 and enemy\_pos[1] < HEIGHT:

            enemy\_pos[1] += SPEED

        else:

            enemy\_list.pop(idx)

            score += 1

    return score

def collision\_check(enemy\_list, player\_pos):

    for enemy\_pos in enemy\_list:

        if detect\_collision(enemy\_pos, player\_pos):

            return True

    return False

def detect\_collision(player\_pos, enemy\_pos):

    p\_x = player\_pos[0]

    p\_y = player\_pos[1]

    e\_x = enemy\_pos[0]

    e\_y = enemy\_pos[1]

    if (e\_x >= p\_x and e\_x < (p\_x + player\_size)) or (p\_x >= e\_x and p\_x < (e\_x+enemy\_size)):

        if (e\_y >= p\_y and e\_y < (p\_y + player\_size)) or (p\_y >= e\_y and p\_y < (e\_y+enemy\_size)):

            return True

    return False

According to which the speed, the conditions for enemies blocks and the collision with them, was all drafted out.

**Conclusion**

In conclusion this was a very exciting project and while using simple syntax and algorithms of python, this creative game can be crafted easily.