Course Name:	**APPA**	**Semester:**	**V**
Date of Performance:	**01 / 10 / 2024**	**Batch No:**	**APPA 2**
Faculty Name:	**Prof. Deepa Jain**	**Roll No:**	**16014022096**
Faculty Sign & Date:		**Grade/Marks:**	**/25**

Experiment No: 6
Title: Study Pygame Library

Aim and Objective of the Experiment:

Implementation: Create a Python game

COs to be achieved:

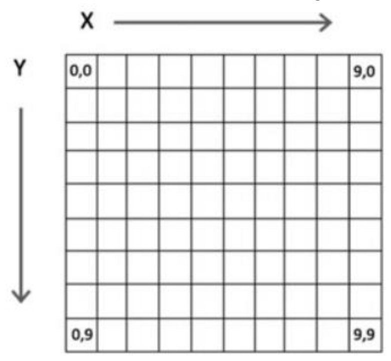
CO4: Develop a brief understanding of game development, game framework and Pythons role

Theory

pygame is a cross-platform, free and Open Source Python library designed to make building multimedia applications such as games easy. pygame is built on top of the SDL library. SDL (or Simple Directmedia Layer) is a cross platform development library designed to provide access to audio, key-boards, mouse, joystick and graphics hardware via OpenGL and Direct3D. To promote portability, pygame also supports a variety of additional backends including WinDIB, X11, Linux Frame Buffer etc.

The Display Surface

The Display Surface (aka the display) is the most important part of a pygame game. It is the main window display of your game and can be of any size, however you can only have one Display Surface. In many ways the Display Surface is like a blank piece of paper on which you can draw. The surface itself is made up of pixels which are numbered from 0,0 in the top left hand corner with the pixel locations being indexed in the x axis and the y axis. This is shown below:



The Display Surface is created by the pygame.display.set_mode() function.

Event Types:
Each event that occurs has associated information such as the type of that event. For example:
• Pressing a key will result in a KEYDOWN type of event, while releasing a key will result in a KEYUP event type.
• Selecting the window close button will generate a QUIT event type etc.
• Using the mouse can generate MOUSEMOTION events as well as MOUSEBUTTONDOWN and MOUSEBUTTONUP event types.

Event Information
Each type of event object provides information associated with that event. For example a Key oriented event object will provide the actual key pressed while a mouse oriented event object will provide information on the position of the mouse, which button was pressed etc. If you try an access an attribute on an event that does not support that attribute, then an error will be generated.
The following lists some of the attributes available for different event types:
• KEYDOWN and KEYUP, the event has a key attribute and a mod attribute
• MOUSEBUTTONUP and MOUSEBUTTONDOWN has an attribute pos that holds a tuple indicating the mouse location in terms of x and y coordinates on the underlying surface. It also has a button attribute indicating which mouse was pressed.
• MOUSEMOTION has pos, rel and buttons attributes. The pos is a tuple indicating the x and y location of mouse cursor. The real attribute indicates the amount of mouse movement and buttons indicates the state of the mouse buttons.

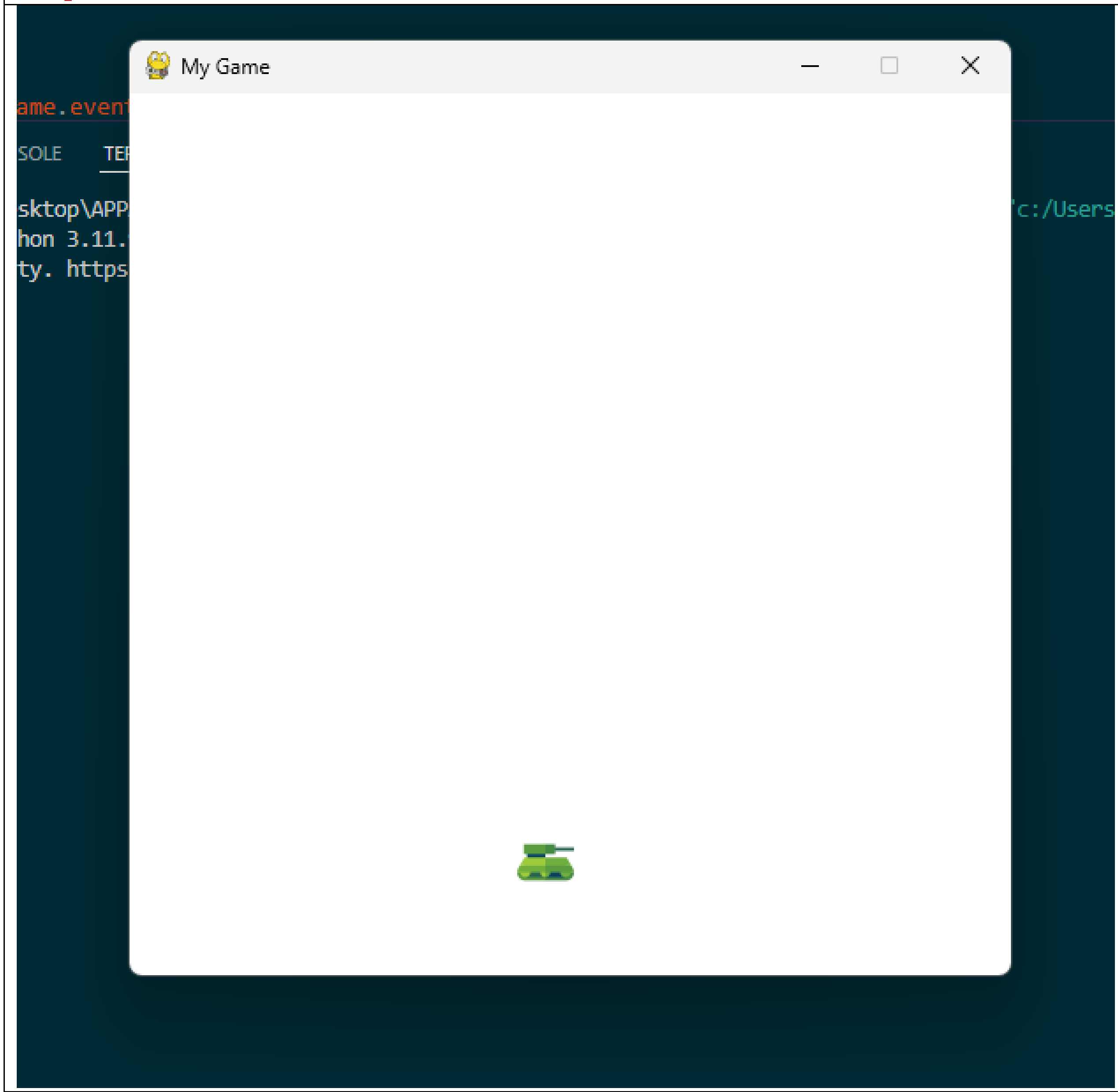
Tools required:
Any python editor tool

Code:

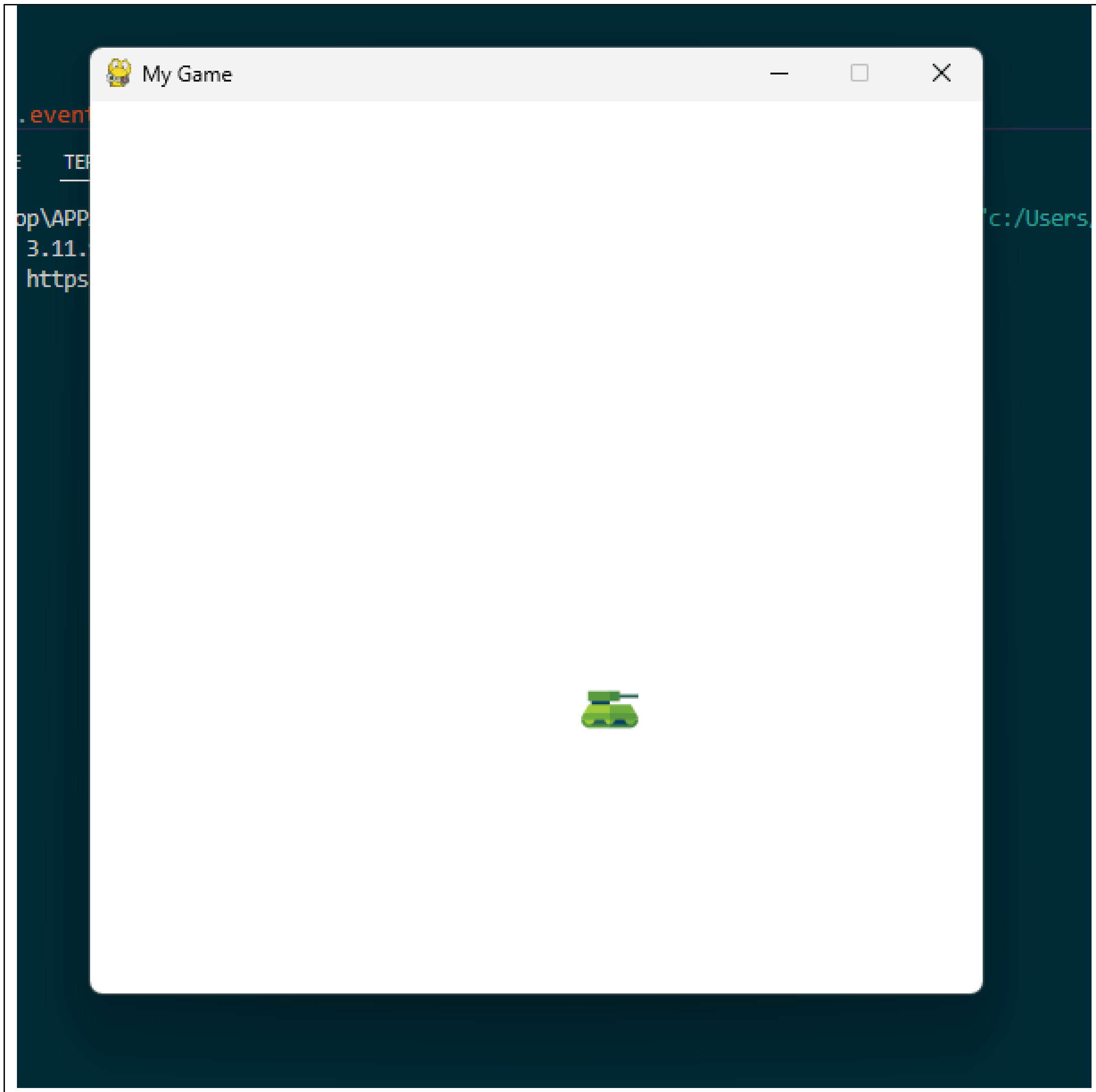
- Write a Python Program for moving a block Up, down, Left, Right and Jump by using various keys.

```
import pygame
import sys
from pygame.locals import *

pygame.init()
pygameSurface = pygame.display.set_mode((500, 500))
pygame.display.set_caption("My Game")
white = pygame.Color(255, 255, 255)


tank = pygame.image.load("EXPERIMENT 07/military-parade.png")
xt = 220
yt = 420

is_jumping = False
velocity_y = 0
gravity = 0.5
jump_strength = 10
initial_y = 0

def mov_left():
    global xt
    xt -= 5
    if xt < 30:
        xt = 30

def mov_right():
    global xt
    xt += 5
    if xt > 470:
        xt = 470

def mov_up():
    global yt
    yt -= 5
    if yt < 30:
        yt = 30

def mov_down():
    global yt
    yt += 5
    if yt > 470:
        yt = 470

def start_jump():
    global is_jumping, velocity_y, initial_y
    is_jumping = True
    initial_y = yt
    velocity_y = -jump_strength


while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()

        if event.type == KEYDOWN:
            if event.key == K_LEFT:
                mov_left()
            if event.key == K_RIGHT:
                mov_right()
            if event.key == K_UP:
                mov_up()
            if event.key == K_DOWN:
                mov_down()
            if event.key == K_SPACE and not is_jumping:
                start_jump()


    if is_jumping:
        yt = yt + velocity_y
        velocity_y += gravity


        if yt >= initial_y:
            yt = initial_y
            is_jumping = False
            velocity_y = 0


    pygameSurface.fill(white)
    pygameSurface.blit(tank, (xt, yt))
    pygame.display.update()
```

Output:



```
ame.event
SOLE    TE
sktop\APP                                         'c:/Users
hon 3.11.
ty. https
```



Conclusion:

We have successfully learned how to implement basic movement and jumping mechanics with boundary checks on pygames.

Signature of faculty in-charge with Date: