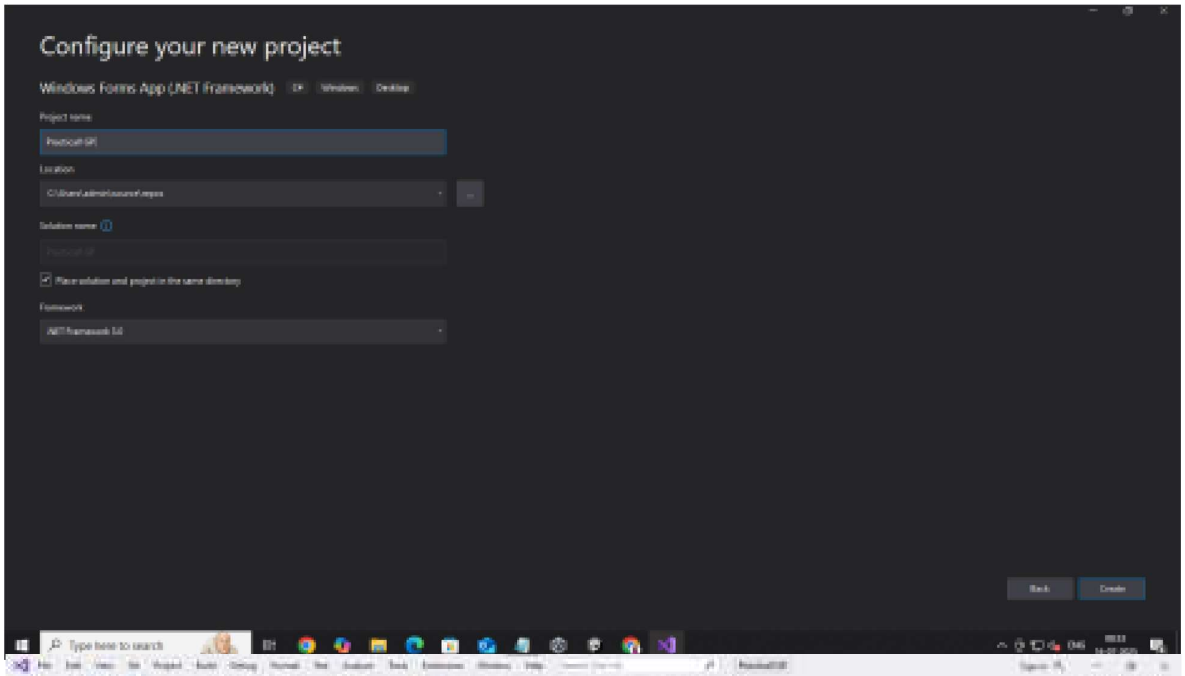
Practical no: 01

AIM : Set up Direct X 11,Window Frame work and Initialize Direct3D Device.

Theory: DirectX is an application program interface (API) for creating and managing graphic images and multimedia effects in applications such as games or active Web pages that will run in Microsoft's Windows operating systems.

Open visual studio: File -> New -> Project -> Visualc# -> Select Windows Forms Application Framework: .Net Framework3.0.



Configure your new project

Windows Forms App (.NET Framework) C# Windows Desktop

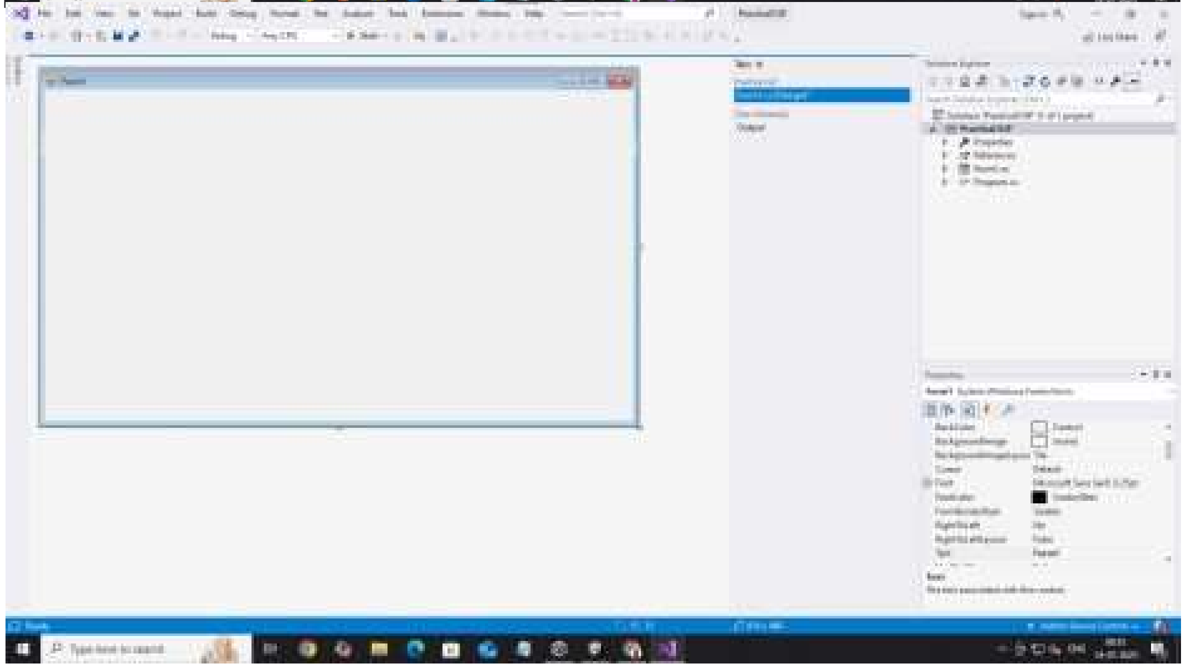
Project name: [illegible]

Location: C:\Users\admin\source\repos

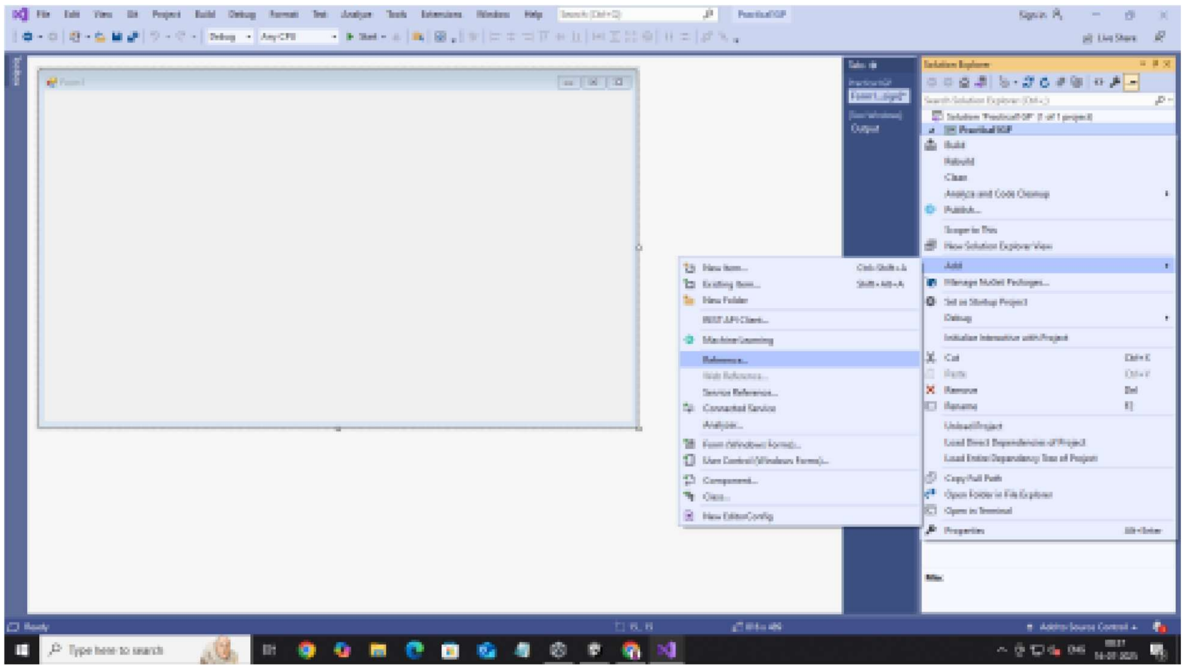
Solution name: [illegible]

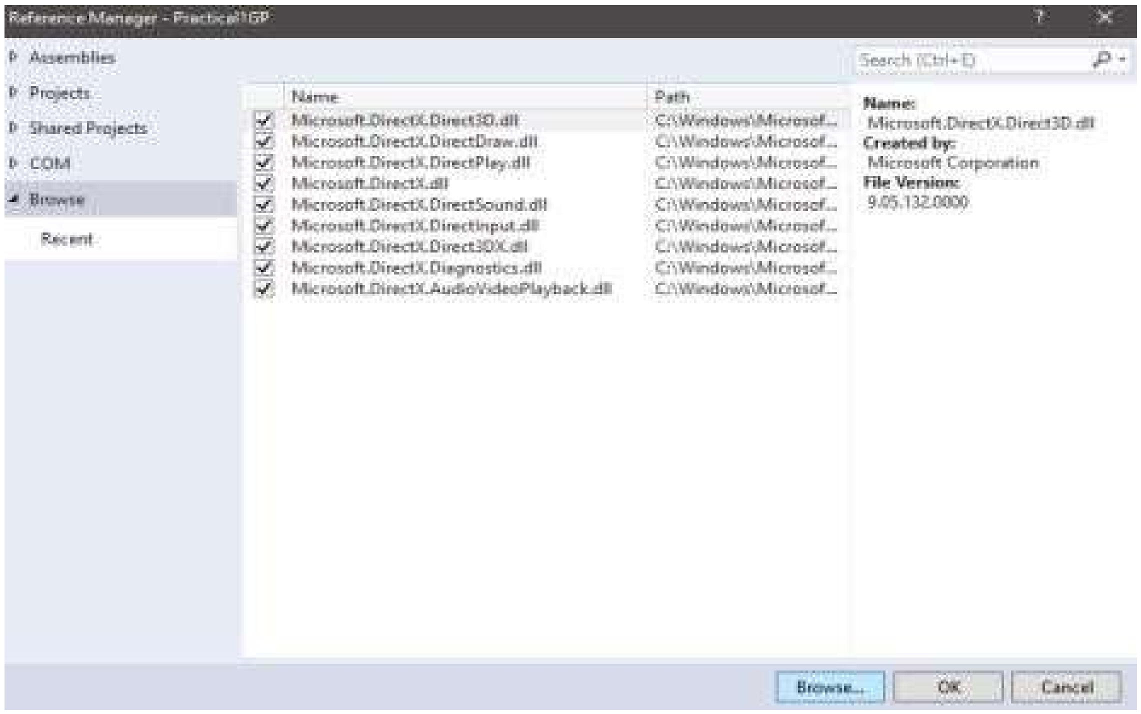
Place solution and project in the same directory

Framework: .NET Framework [illegible]



Add References:
Right Click on References -> Add References -> Browse -> Click on Browse Button Goto C -> Windows -> Microsoft.Net folder ->DirectX for managed code ->folder 1.0.2902.0 Select the all the files ->click on OK




Reference Manager - Practical1GP

Assemblies
Projects
Shared Projects
COM
Browse
Recent

Search (Ctrl+E)

	Name	Path
✓	Microsoft.DirectX.Direct3D.dll	C:\Windows\Microsof...
✓	Microsoft.DirectX.DirectDraw.dll	C:\Windows\Microsof...
✓	Microsoft.DirectX.DirectPlay.dll	C:\Windows\Microsof...
✓	Microsoft.DirectX.dll	C:\Windows\Microsof...
✓	Microsoft.DirectX.DirectSound.dll	C:\Windows\Microsof...
✓	Microsoft.DirectX.DirectInput.dll	C:\Windows\Microsof...
✓	Microsoft.DirectX.Direct3DX.dll	C:\Windows\Microsof...
✓	Microsoft.DirectX.Diagnostics.dll	C:\Windows\Microsof...
✓	Microsoft.DirectX.AudioVideoPlayback.dll	C:\Windows\Microsof...

Name:
Microsoft.DirectX.Direct3D.dll
Created by:
Microsoft Corporation
File Version:
9.05.132.0000

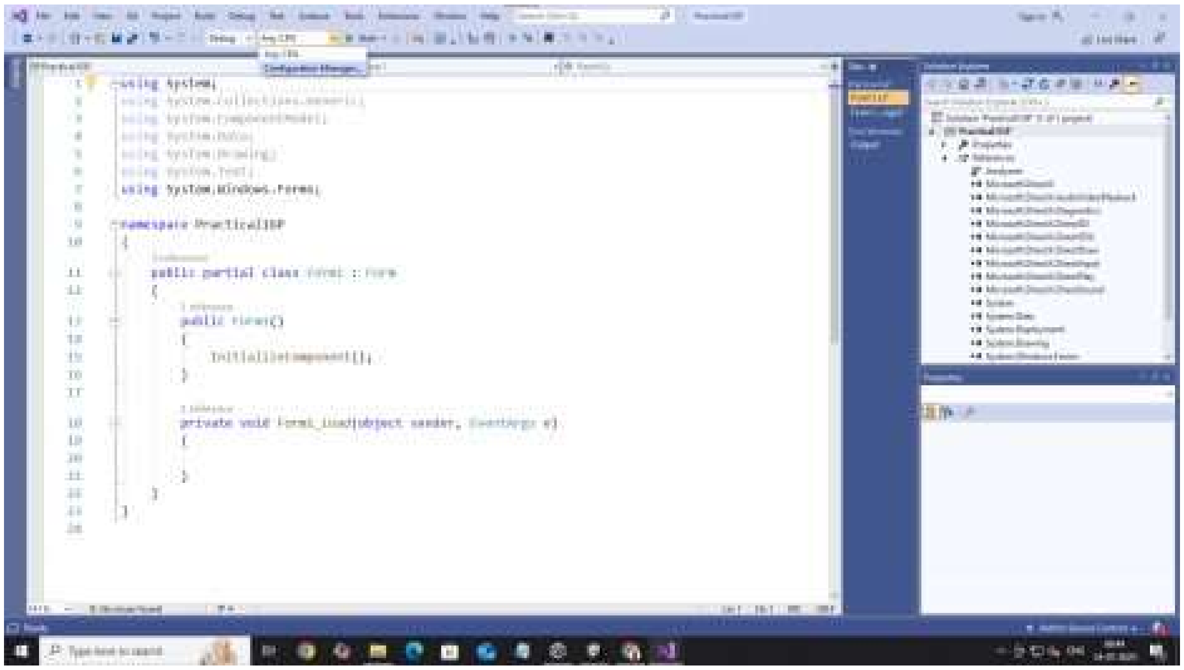
Browse... OK Cancel

Now right click on your Form1.cs design and click on View code. And modify the code as given below.

CODE:

```
using System;
using System.Windows.Forms;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;
namespace WindowsFormsApp2
{
 public partial class Form1 : Form
 {
 Microsoft.DirectX.Direct3D.Device device;
 public Form1()
 {
 InitializeComponent();
 this.Load += new EventHandler(Form1_Load);
 this.Paint += new PaintEventHandler(Form1_Paint);  }
 private void Form1_Load(object sender, EventArgs e)  {
 InitDevice();
 }
 public void InitDevice()
 {
 PresentParameters pp = new PresentParameters();
 pp.Windowed = true;
 pp.SwapEffect = SwapEffect.Discard;
 device = new Device(0, DeviceType.Hardware, this,
CreateFlags.SoftwareVertexProcessing, pp);
 }
 private void Render()
 {
 if (device == null)
 return;
 device.Clear(ClearFlags.Target, System.Drawing.Color.Pink, 1.0f, 0);
device.BeginScene();
 // Draw any 3D objects here if needed
 device.EndScene();
 device.Present();
 }
 private void Form1_Paint(object sender, PaintEventArgs e)  {
 Render();
 }
 }
 }
```

Now, as we are dealing with 3D files we have to change the capacity of our CPU from x64 to x86.for that follow below steps.
Click on Any CPU -> Configuration manager ->New-> select x86 as a new platform ->click on Ok-> click on close.

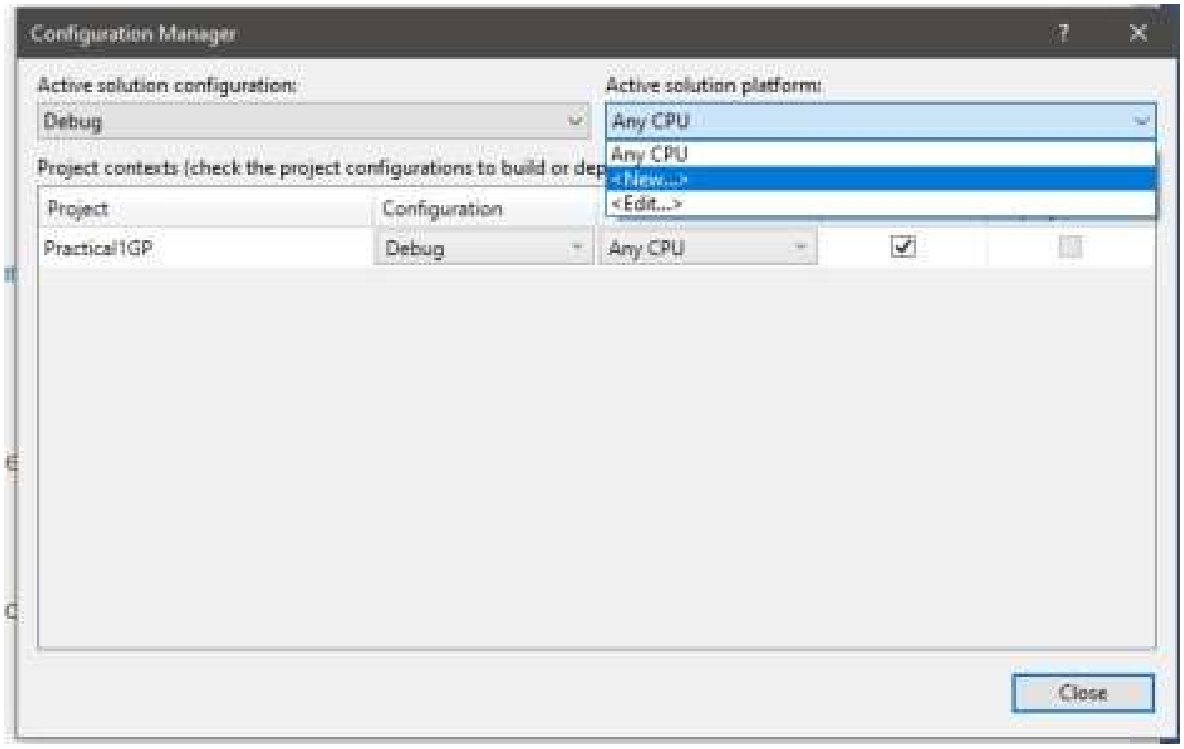


```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace Practical1GP
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }
    }
}
```



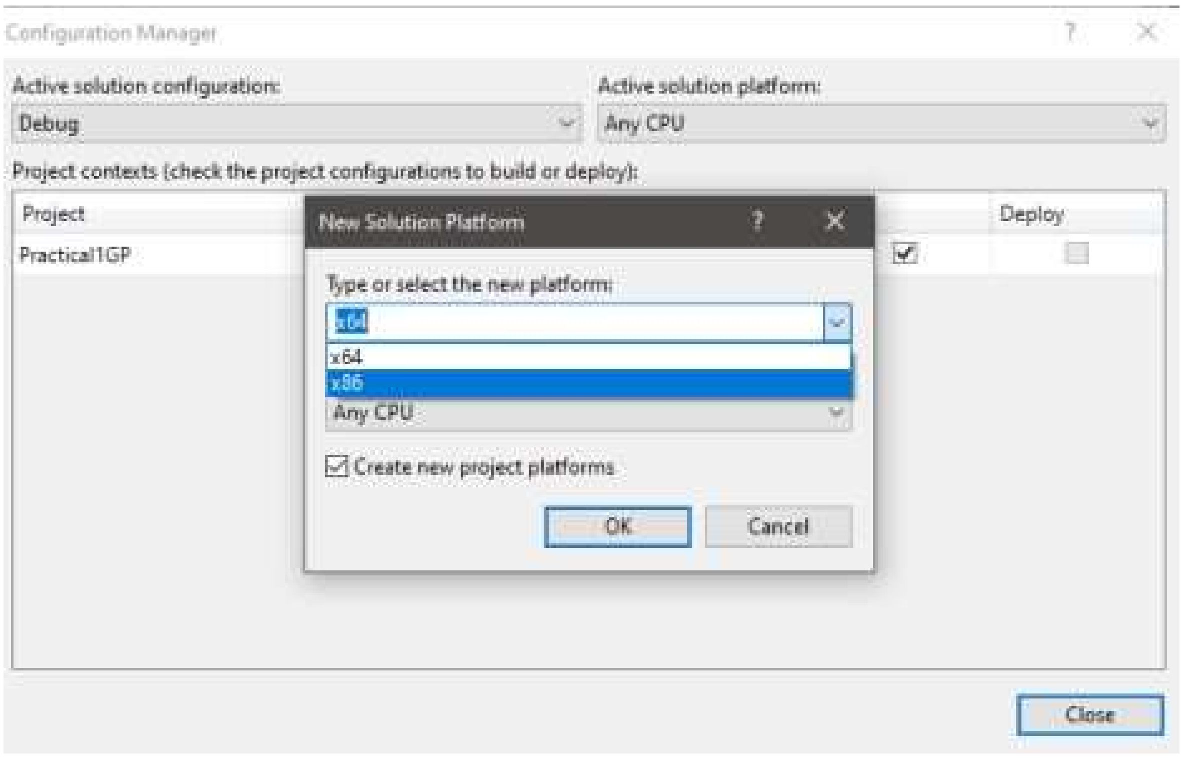
Configuration Manager

Active solution configuration: Debug

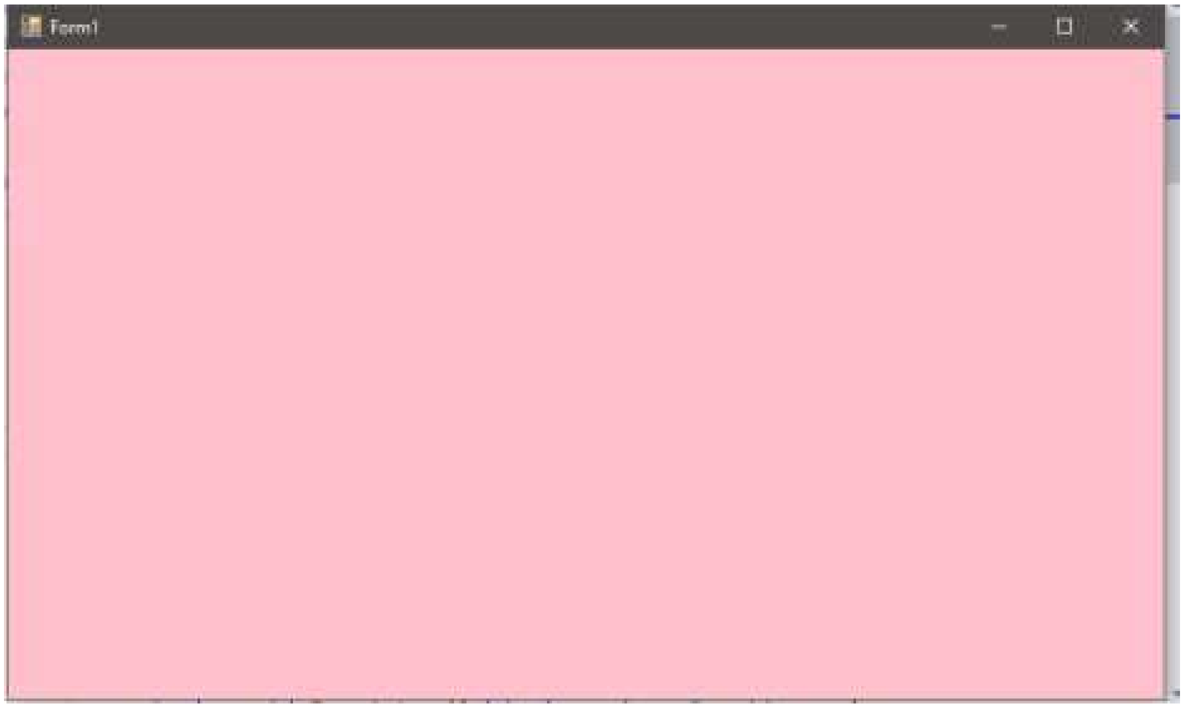
Active solution platform: Any CPU

Project contexts (check the project configurations to build or dep

Project	Configuration			
Practical1GP	Debug	Any CPU	✓	


Configuration Manager
Active solution configuration:
Debug
Active solution platform:
Any CPU
Project contexts (check the project configurations to build or deploy):
Project
Deploy
Practical1GP
New Solution Platform
Type or select the new platform:
x64
x64
x86
Any CPU
Create new project platforms
OK
Cancel
Close

Now start debugging your project it will successfully change the background colour of form using the paint method.



Practical no:02

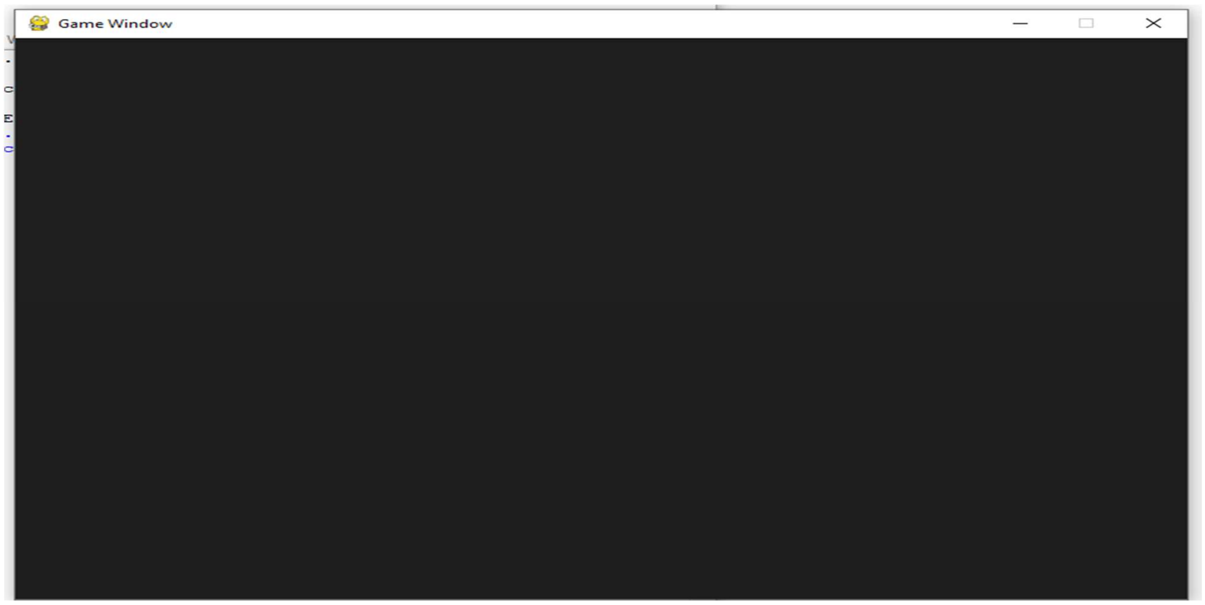
Aim: Learn Basic Game Designing Techniques with pygame.
2A]Create a gaming window using pygame.

Theory: o Pygame is a cross-platform set of Python modules which is used to create video games. o It consists of computer graphics and sound libraries designed to be used with the Python programming language.

- o Pygame was officially written by Pete Shinners to replace PySDL.
- o Pygame is suitable to create client-side applications that can be potentially wrapped in a standalone executable.

```
import pygame  import
sys  pygame.init()
print("YourName_rollno
")
# Set up window  screen =
pygame.display.set_mode((800, 600))
pygame.display.set_caption("Game Window")
# Game loop  while True:   for
event in pygame.event.get():
if event.type ==
pygame.QUIT:
pygame.quit()   sys.exit()

 screen.fill((30, 30, 30)) # Dark background
pygame.display.flip()
```



2B] Write a python program to draw different shapes using pygame Pygame Draw.

• Pygame provides geometry functions to draw simple shapes to the surface.

• These functions will work for rendering to any format to surfaces.

• Most of the functions accept a width argument to signify the size of • If the width is passed 0, then the shape will be solid(filled).

• All the drawing function takes the color argument that can be one of the following formats:

o A pygame.Color objects o An (RGB) triplet(tuple/list) o An (RGBA) quadruplet(tuple/list)

o An integer value that has been mapped to the surface's pixel format

• Draw a rectangle

The following functions are used to draw a rectangle on the given surface.

1. pygame.draw.rect(surface, color, rect)
2. pygame.draw.rect(surface, color, rect, width=0) Parameters:

· surface - Screen to draw on.

· color- This argument is used to color the given shape. The alpha value is optional if we are using a tuple.

· rect(Rect)- Draw rectangle, position, and dimensions.

· Width (int)- This is optional to use the line thickness or to indicate that the rectangle is filled.

· Draw a straight line

This method is used to draw a straight line on the given surface.
There are no endcaps.

1.
pygame.draw.line(surface,color,start_pos,end_pos,wi
dth) 2.
pygame.draw.line(surface,color,start_pos,end_pos,wi
dth=1) Parameters:

· surface - Screen to draw on.

· color- This argument is used to color the given shape. The alpha value is optional if we are using a tuple.

· start_pos- start position of the line(x,y)

· end_pos- End position of the line

· Draw a Circle

Below are the functions, which are used to draw a circle on the given surface. · circle(surface, color, center, radius)

· circle(surface, color, center, radius, width=0)

Parameters:

· surface - Screen to draw on.

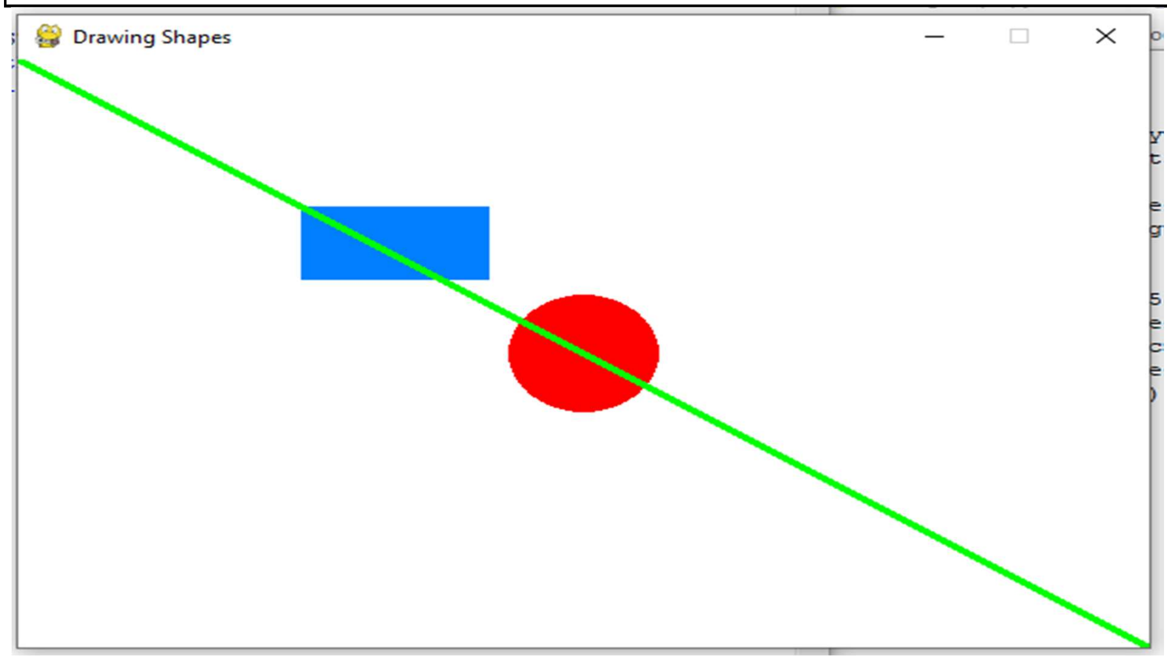
· color- This argument is used to color the given shape. The alpha value is optional if we are using a tuple.

· center - The center point of the circle as a sequence of two int/float, e.g.(x,y) · radius(int or float)- radius of the circle, measured from the center parameter, if the radius is zero, then it will only draw the center pixel.

.

CODE:

```
import pygame
import sys
pygame.init()
screen = pygame.display.set_mode((600, 400))
pygame.display.set_caption("DrawinShap")
while True:for event in pygame.event.get():
   if event.type == pygame.QUIT:
      pygame.quit()
      sys.exit()
    screen.fill((255, 255, 255))
  pygame.draw.rect(screen, (0, 128,
255),(150,100,100,50))
 pygame.draw.circle(screen, (255, 0, 0), (300, 200),
40)
 pygame.draw.line(screen, (0, 255, 0), (0, 0), (600,
400), 5)
pygame.display.flip()
```

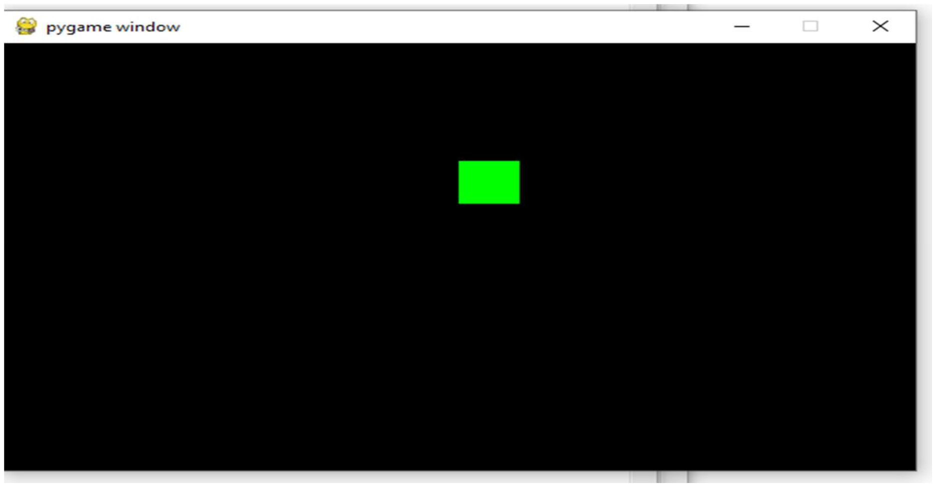


2C] Write a python code to apply/change position of an object using key events with pygame.

Code:

```
import pygame
import sys
pygame.init()
screen =
pygame.display.set_m
ode((600, 400))
clock =
pygame.time.Clock()
player_pos = [300,
200]
while True:
 for event in
pygame.event.get():
    if event.type ==
pygame.QUIT:
      pygame.quit()
      sys.exit()
 keys =
pygame.key.get_press
ed()
 if
keys[pygame.K_LEF
T]: player_pos[0] -= 5
 if
keys[pygame.K_RIG
HT]: player_pos[0] +=
5
 if
keys[pygame.K_UP]:
player_pos[1] -= 5
 if
keys[pygame.K_DO
WN]: player_pos[1]
+= 5
 screen.fill((0, 0, 0))

pygame.draw.rect(scre
en, (0, 255, 0),
(*player_pos, 40, 40))
 pygame.display.flip()
 clock.tick(60)
```



2D] Write a python program for Collision Detection.

Pygame Sprite and Collision detection

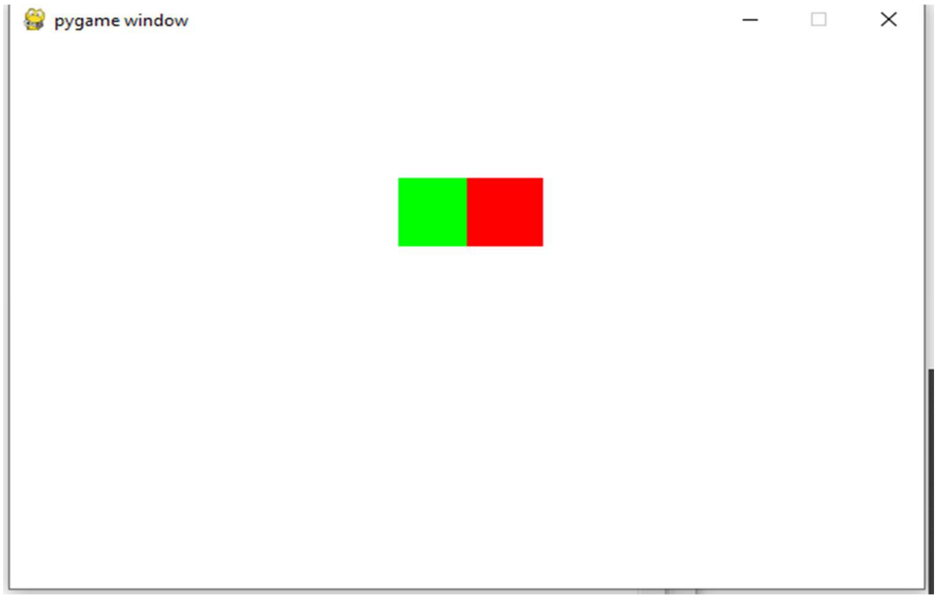
· A pygame sprite is a two-dimensional image that is part of the large graphical scene. Usually, a sprite will be some object in the scene.

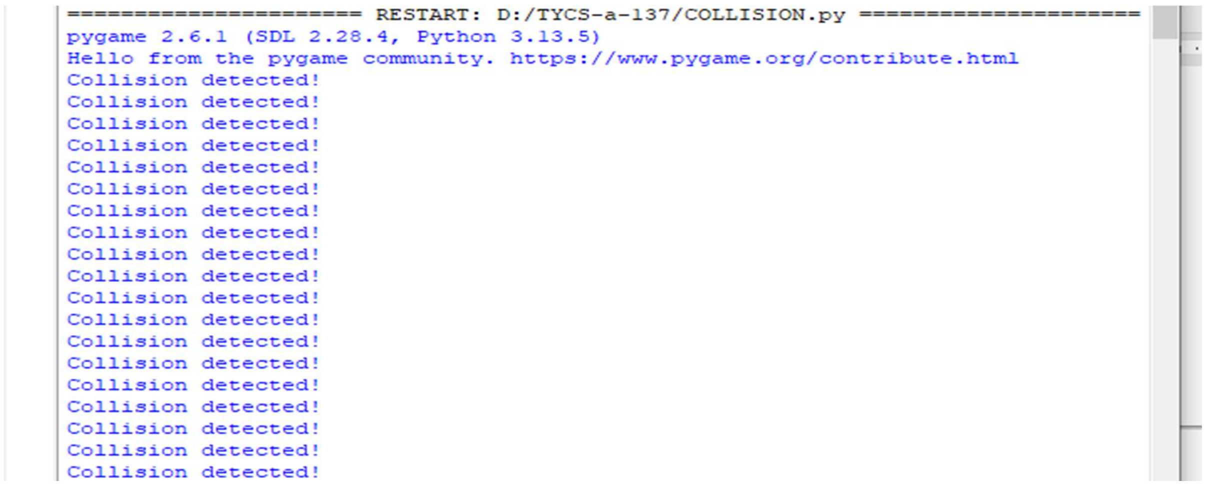
· One of the most advantages of working with sprites is the ability to work with them in groups. We can easily move and draw all the sprites with the one command if they are in the group.

· The Sprite module contains the various simple classes to be used within the games. It is optional to use Sprite classes and different group classes when using pygame.

· Pygame provides sprites and sprite groups that help for collision detection. Collision detection is the process when two objects on the screen collide each other. For example, if a player is hit by the enemy's bullet, then it may lose a life or, the program need to know when the player touches a coin so that they automatically picked up.

```
import pygame
import sys
pygame.init()
screen = pygame.display.set_mode((600, 400))
player = pygame.Rect(100, 100, 50, 50)
enemy = pygame.Rect(300, 100, 50, 50)
clock = pygame.time.C lock()
while True:
 for event in pygame.event.get():
 if event.type == pygame.QUIT:
 pygame.quit()
 sys.exit()
 keys = pygame.key.get_pressed()
 if keys[pygame.K_RIGHT]: player.x += 5
 if keys[py game.K_LEFT]: player.x  -= 5
 screen.fill((255, 255, 255))
 pygame.draw.rect(screen, (0, 255, 0), player)
 pygame.draw.rect(screen, (255, 0, 0), enemy)
 if player.colliderect(enemy):
 print("Collision detected!")
 pygame.display.flip()
 clock.tick(60 )
```





```
===================== RESTART: D:/TYCS-a-137/COLLISION.py =====================
pygame 2.6.1 (SDL 2.28.4, Python 3.13.5)
Hello from the pygame community. https://www.pygame.org/contribute.html
Collision detected!
Collision detected!
Collision detected!
Collision detected!
Collision detected!
Collision detected!
Collision detected!
Collision detected!
Collision detected!
Collision detected!
Collision detected!
Collision detected!
Collision detected!
Collision detected!
Collision detected!
Collision detected!
Collision detected!
Collision detected!
Collision detected!
```

2E] Write a python program to design a score system with pygame..

· Pygame Surface

- The pygame Surface is used to display any image.
- The Surface has a pre-defined resolution and pixel format.
- The Surface color is by default black.
- Its size is defined by passing the size argument.
- Surfaces can have the number of extra attributes like alpha planes, color keys, source rectangle clipping, etc.
- The blit routines will attempt to use hardware acceleration when possible; otherwise, they will use highly enhanced software blitting methods.

· Pygame Clock

- Times are represented in millisecond (1/1000 seconds) in pygame.
- Pygame clock is used to track the time.
- The time is essential to create motion, play a sound, or, react to any event. • In general, we don't count time in seconds. We count it in milliseconds. • The clock also provides various functions to help in controlling the game'sframe rate. The few functions are the following: · tick()

This function is used to update the clock. The syntax is the following: · tick(framerate=0) • This method should be called once per frame.

- It will calculate how many milliseconds have passed since the previous call.
- The framerate argument is optional to pass in the function, and if it is passed as an argument then the function will delay to keep the game running slower than the given ticks per second. · tick_busy_loop()

- The tick_busy_loop() is same as the tick().
- By calling the Clock.tick_busy_loop(20) once per frame, the program will never run at more than 20 frames per second.
- The syntax is the following:
  tick_busy_loop()

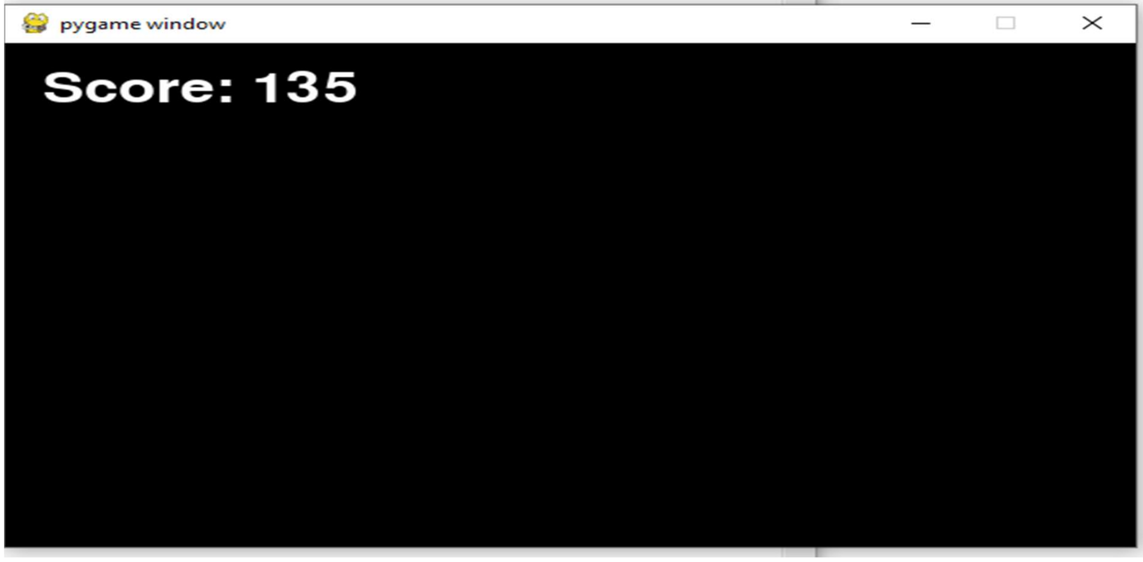
· get_time()

- The get_time() is used to get the previous tick.
- The number of a millisecond that isdra passed between the last two calls in Clock.tick(). get_time()

· Pygame Blit

- The pygame blit is the process to render the game object onto the surface,and this process is called blitting.
- When we create the game object, we need to render it.
- If we don't render the game objects and run the program, then it will give the black window as an output.
- Blitting is one of the slowest operations in any game so, we need to be careful to not to blit much onto the screen in every frame.
- The primary function used in blitting is blit(), which is: blit() blit(source,dest,area=None,special_flags=0)

```
import pygame
import sys
pygame.init()
screen =
pygame.display.set_m
ode((600, 400))
font =
pygame.font.SysFont(
None, 48)
score = 0
clock =
pygame.time.Clock()
while True:
 for event in
pygame.event.get():
  if event.type ==
pygame.QUIT:
   pygame.quit()
   sys.exit()
 screen.fill((0, 0, 0))
 score += 1
 score_text =
font.render(f"Score:
{score}", True, (255,
255, 255))
 screen.blit(score_text,
(20, 20))
 pygame.display.flip()
 clock.tick(60)
```



```
Score: 135
```

Practical no. 3

AIM : Develop a snake game using pygame. ● User-defined functions in Python.

- Python pygame.Color() function
- Python pygame.time.Clock() function
- Python random.randrange(start, stop, step) method
- Python pygame.display.set_caption() method
- Python pygame.display.set_mode() function
- Pygame pygame.render() method
- Pygame .get_rect() function
- flip() method in Pygame
- Pygame .blit() method ● time.sleep() in Python ● pygame.quit() in Pygame.
- pygame .midtop() function ● for loop in Python
- if statements in Python. ● if- else loop in Python
- .insert() in Pygame
- .fill() in Pygame

Theory:

· Pygame Text and Font • Pygame also provides facilities to render the font and text.

- We can load fonts from the system by using the pygame.font.SysFont() function. • Pygame comes with the built-in default font which can be accessed by passing the font name or None.
- There are many functions to help to work with the font.
- The font objects are created with pygame.font.Font().
- The actual font objects do most of the works done with fonts.
- Font objects are generally used to render the text into new Surface objects. • Few important font functions are the following: · render()
- This function is used to draw text on a new Surface.
- Pygame has no facility to draw text on the existing Surface.
- This creates a new Surface with the specified text render on it.
- The syntax is the following: render(text, antialias, color, background=None) · size()
- This function is used to determine the number of space or positioning needed to render text.
- It can also be used for word-wrapping and other layout effects.
- The syntax is the following: size(bool) · set_bold()

· This function is used for bold rending of text. The syntax is following: · set_bold(bool)

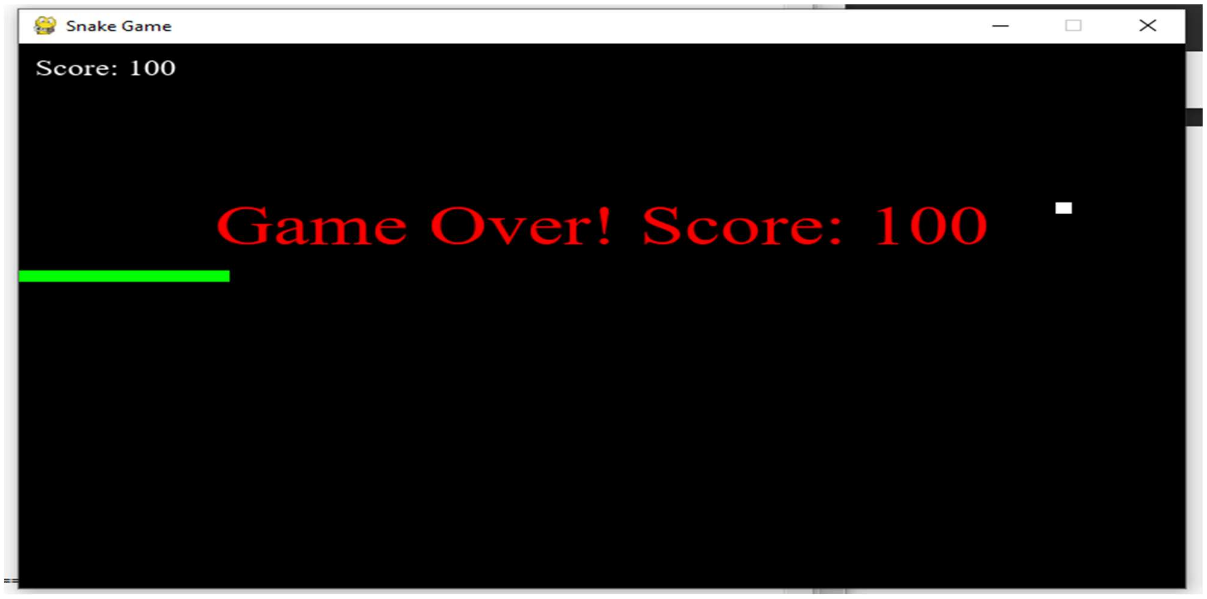
· Pygame Keydown

Pygame KEYDOWN and KEYUP detect the event if a key is physically pressed and released. KEYDOWN detects the key press and, KEYUP detects the key release. Both events (Key press and Key release) have two attributes which are the following: · key: Key is an integer id which represents every key on the keyword. · mod: This is a bitmask of all the modifier keys that were in the pressed state when the event occurred.

Code:

```
import pygame, time, random
pygame.init()
w, h = 720, 480
win = pygame.display.set_mode((w, h))
pygame.display.set_caption("Snake Game")
clock = pygame.time.Clock()
# Colors
white = (255,255,255)
green = (0,255,0)
red = (255,0,0)
black = (0,0,0)
snake = [[100, 50], [90, 50], [80, 50]]
fruit = [random.randrange(1, w//10)*10,
random.randrange(1, h//10)*10]
direction = 'RIGHT'
score = 0
font = pygame.font.SysFont('times new roman', 20)
def game_over():
 msg = pygame.font.SysFont('times new roman',
50).render(f'Game Over! Score: {score}',
True, red)
 win.blit(msg, msg.get_rect(center=(w//2, h//3)))
 pygame.display.flip()
 time.sleep(2)
 pygame.quit()
 quit()
while True:
 for e in pygame.event.get():
  if e.type == pygame.QUIT:
   game_over()
  if e.type == pygame.KEYDOWN:
     if e.key == pygame.K_UP and direction !=
'DOWN': direction = 'UP'
```

```
    elif e.key == pygame.K_DOWN and direction
!= 'UP': direction = 'DOWN'
    elif e.key == pygame.K_LEFT and direction !=
'RIGHT': direction = 'LEFT'
    elif e.key == pygame.K_RIGHT and direction
!= 'LEFT': direction = 'RIGHT'
 head = snake[0][:]
 if direction == 'UP': head[1] -= 10
 elif direction == 'DOWN': head[1] += 10
 elif direction == 'LEFT': head[0] -= 10
 elif direction == 'RIGHT': head[0] += 10
 snake.insert(0, head)
 if head == fruit:
   score += 10
   fruit = [random.randrange(1, w//10)*10,
random.randrange(1, h//10)*10]
 else:
   snake.pop()
 if (head[0] < 0 or head[0] >= w or head[1] < 0 or
head[1] >= h or head in snake[1:]):
   game_over()
 win.fill(black)
 for s in snake:
   pygame.draw.rect(win, green, (*s, 10, 10))
 pygame.draw.rect(win, white, (*fruit, 10, 10))
 win.blit(font.render(f'Score: {score}', True, white),
(10, 10))
 pygame.display.update()
 clock.tick(10)
```



```
Score: 100
Game Over! Score: 100
```

Practical no: 04

Aim: Create a 2D target shooting game using pygame.

Theory:

Game Overview

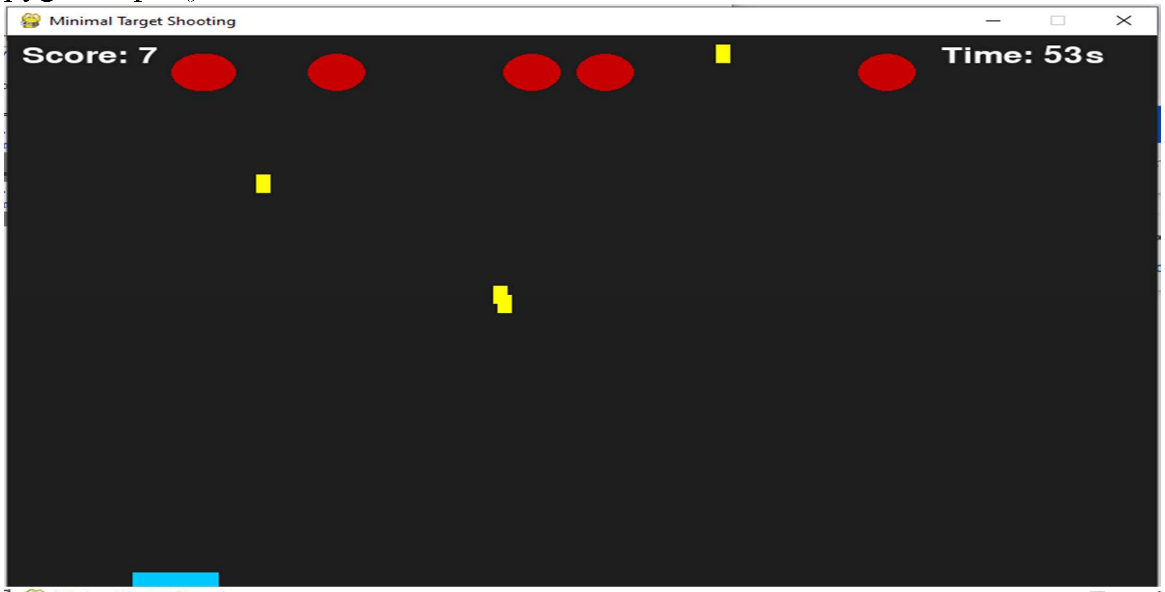
· The player controls a paddle at the bottom of the screen.
· The player can shoot bullets upward by clicking the mouse.
· Targets (red ellipses) move horizontally near the top of the screen.
· The goal is to shoot as many targets as possible within **60 seconds**.
· The game ends when the timer reaches zero, displaying the final score.

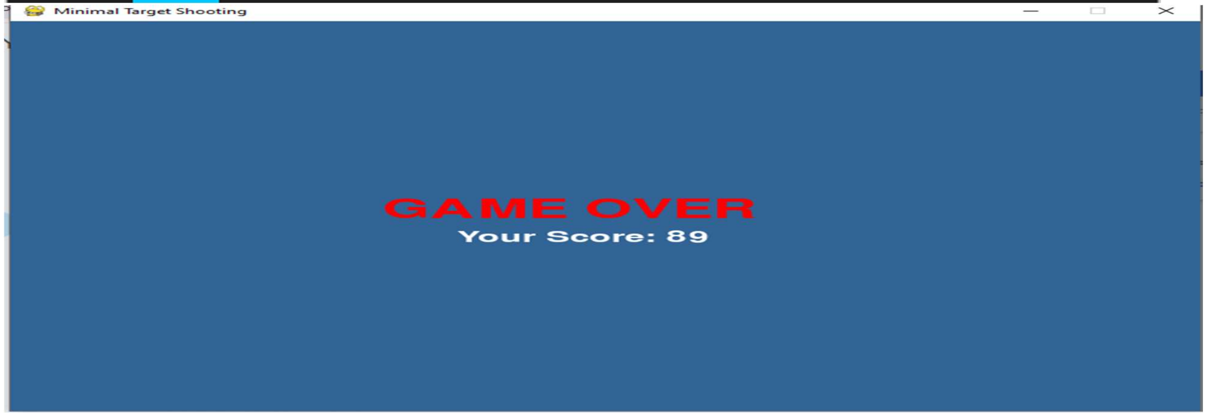
Code:

```
import pygame, random, sys
pygame.init()
w, h = 800, 600
win =
pygame.display.set_mode((w, h))
pygame.display.set_caption("Mini
mal Target Shooting")
clock = pygame.time.Clock()
font = pygame.font.SysFont(None,
36)
big_font =
pygame.font.SysFont(None, 60)
player = pygame.Rect(w//2 - 30, h
- 20, 60, 20)
bullets, targets = [], []
score, time_limit = 00, 60
start = pygame.time.get_ticks()
class Bullet:
 def __init__(self, x, y): self.r =
pygame.Rect(x-5, y-20, 10, 20)
 def move(self): self.r.y -= 10
class Target:
 def __init__(self):
  self.r =
pygame.Rect(random.randint(0,
w-40), 20, 40, 40)
  self.s = random.choice([-3, -2, 2,
3])
 def move(self):
  self.r.x += self.s
  if not (0 <= self.r.x <= w -
self.r.w): self.s *= -1
running = True
while running:
 win.fill((30, 30, 30))
 secs = time_limit -
(pygame.time.get_ticks() -
start)//1000
 if secs <= 0: break
 for e in pygame.event.get():
  if e.type == pygame.QUIT:
pygame.quit(); sys.exit()
 player.centerx =
pygame.mouse.get_pos()[0]
 if pygame.mouse.get_pressed()[0]
and len(bullets) < 5:

bullets.append(Bullet(player.cente
rx, player.top))
 for b in bullets[:]:
  b.move()
  pygame.draw.rect(win, (255,
255, 0), b.r)
  if b.r.bottom < 0:
bullets.remove(b)
 if random.randint(1, 40) == 1:
targets.append(Target())
 for t in targets[:]:
  t.move()
  pygame.draw.ellipse(win, (200,
0, 0), t.r)
 for t in targets[:]:
  for b in bullets[:]:
   if t.r.colliderect(b.r):
    targets.remove(t)
    bullets.remove(b)
    score += 1
    break
 pygame.draw.rect(win, (0, 200,
255), player)
 win.blit(font.render(f"Score:
{score}", True, (255, 255, 255)),
(10, 10))
```

```
 win.blit(font.render(f"Time:
{secs}s", True, (255, 255, 255)),
(w - 150, 10))
 pygame.display.update()
 clock.tick(60)
# Game Over
win.fill((50, 100, 150))
win.blit(big_font.render("GAME
OVER", True, (255, 0, 0)), (w//2 -
150, h//2 - 30))
win.blit(font.render(f"Your Score:
{score}", True, (255, 255, 255)),
(w//2 - 100, h//2 + 20))
pygame.display.update()
pygame.time.delay(4000)
pygame.quit()
```


Minimal Target Shooting
Score: 7
Time: 53s


Minimal Target Shooting
GAME OVER
Your Score: 89