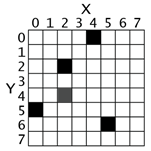
**EX NO: 12 Bouncing Ball simulation using PyGames**

**Date :**

**AIM**

To simulate the bouncing ball using pygame.

**ALGORITHM**

**Step 1:**  Import and initialize pygame with **pygame.init()  
Step 1:** Create a graphical screen (Surface) with pygame.display.set\_mode().  
**Step 2:** Load ball image and define the rectangle around the image.  
**Step 3:** Inside the infinite loop, move the ball by (2,2) pixels (speed) in x,y coordinates  
  
**Step 4:** If the ball goes out of the screen boundary, reverse the direction of speed  
**Step 5:** Fill the screen with black, before displaying the next position of the ball to avoid the trail of the ball visible in the animation.  
**Step 6:** Draw the ball in its next position on the screen using Surface.blit() method.   
**Step 7:** The pygame.display.flip() method makes everything we have drawn on the screen Surface become visible.  
**Step 8:** If user triggers quit() event (close button), the simulation stops.

**Reference**

<http://pygame.org/docs/tut/PygameIntro.html>  
<https://inventwithpython.com/pygame/chapter2.html>

**SOURCE CODE**

import sys, pygame

pygame.init()

size = width, height = 320, 240

speed = [2, 2]

black = 0, 0, 0

screen = pygame.display.set\_mode(size)

clock = pygame.time.Clock()

ball = pygame.image.load("ball.gif")

ballrect = ball.get\_rect()

while 1:

clock.tick(30)

for event in pygame.event.get():

if event.type == pygame.QUIT: sys.exit()

ballrect = ballrect.move(speed)

if ballrect.left < 0 or ballrect.right > width:

speed[0] = -speed[0]

if ballrect.top < 0 or ballrect.bottom > height:

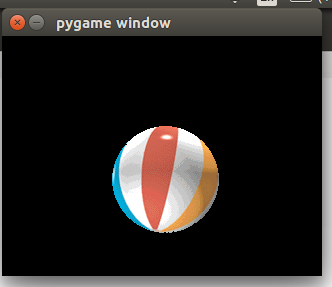
speed[1] = -speed[1]

screen.fill(black)

screen.blit(ball, ballrect)

pygame.display.flip()

**OUTPUT**



**RESULT**

Thus the bouncing ball using pygame was simulated successfully.