Assignment 3 Part B

Vasu Bansal, Roll No. 160776, Course - ME766

Path planning with obstacles.
Algorithm used - RRT(Rapidly exploring Random Trees)
Assignment is written in *Python* and implemented on *Jupyter Notebook*
Note - Python is used because Object oriented programming was easier to do and also because there is a library called Pygame which was used. It offered various functionalities like drawing Obstacles, detecting collisions plotting lines easily and also the implementation was a lot faster than in MatLab

Reference used - https://en.wikipedia.org/wiki/Rapidly-exploring_random_tree (https://en.wikipedia.org/wiki/Rapidly-exploring_random_tree)
Pseudo code used :-

```
Algorithm BuildRRT
  Input: Initial configuration q_init, number of vertices in RRT K, incremental distance Δq)
  Output: RRT graph G

  G.init(q_init)
  for k = 1 to K
    q_rand ← RAND_CONF()
    q_near ← NEAREST_VERTEX(q_rand, G)
    q_new ← NEW_CONF(q_near, q_rand, Δq)
    G.add_vertex(q_new)
    G.add_edge(q_near, q_new)
  return G
```

In [1]:
```
# Import necessary libraries
import math, pygame, random, sys
from math import * # So that we can simply use the function
from pygame import *
```

```
pygame 1.9.6
Hello from the pygame community. https://www.pygame.org/contribute.html
```

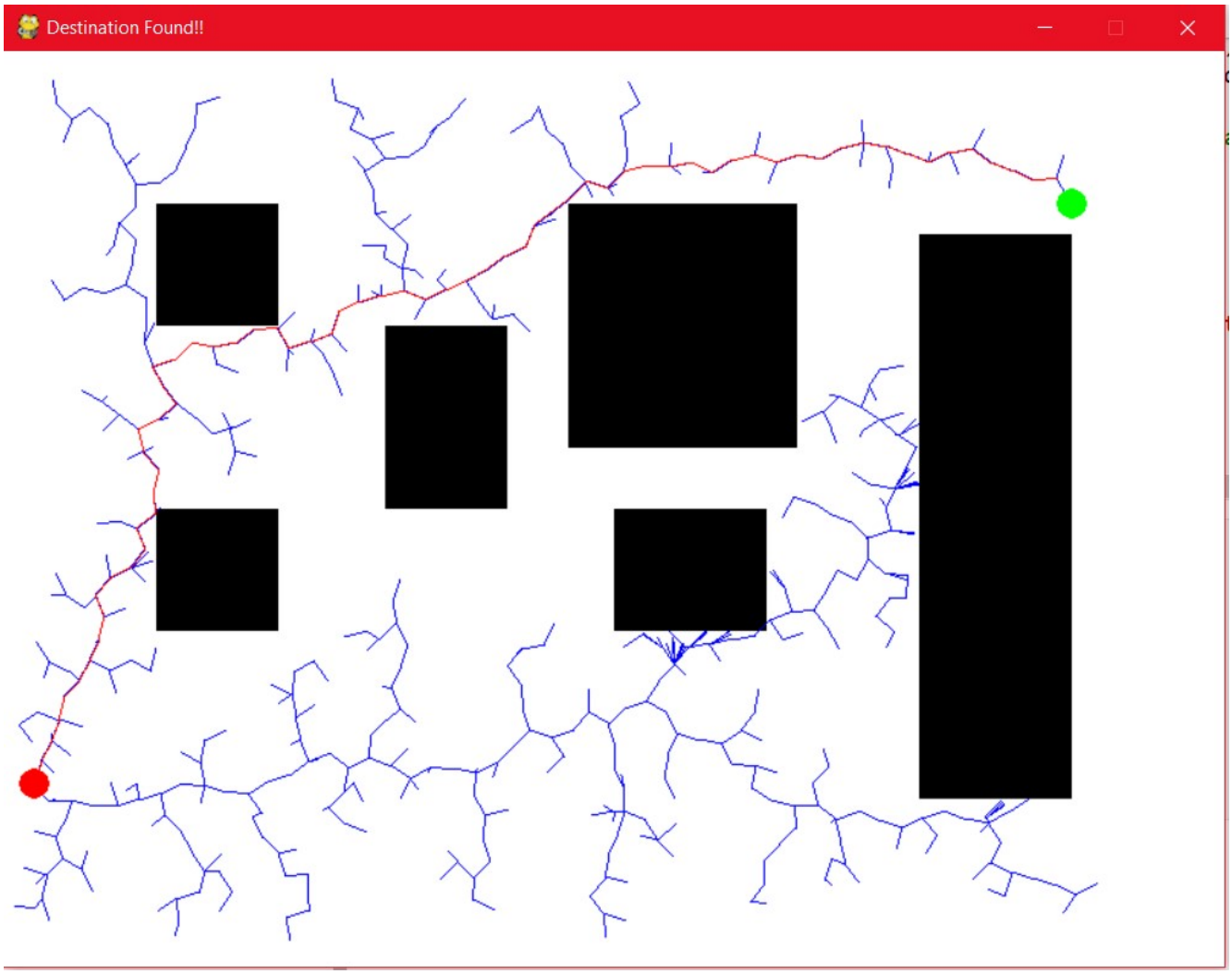
In [2]:
```
# Screen parameters
WIDTH = 800
HEIGHT = 600
FPS = 100 # Frames per second

# Standard colors which will be used
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255)

# Parameters
startingX, startingY = 20, 480
goalX, goalY = 700, 100
thresh = 10 # Maximum distance between new node and current node
range = 10 # Distance upto which Goal can be detected automatically
num_of_nodes = 10000
```

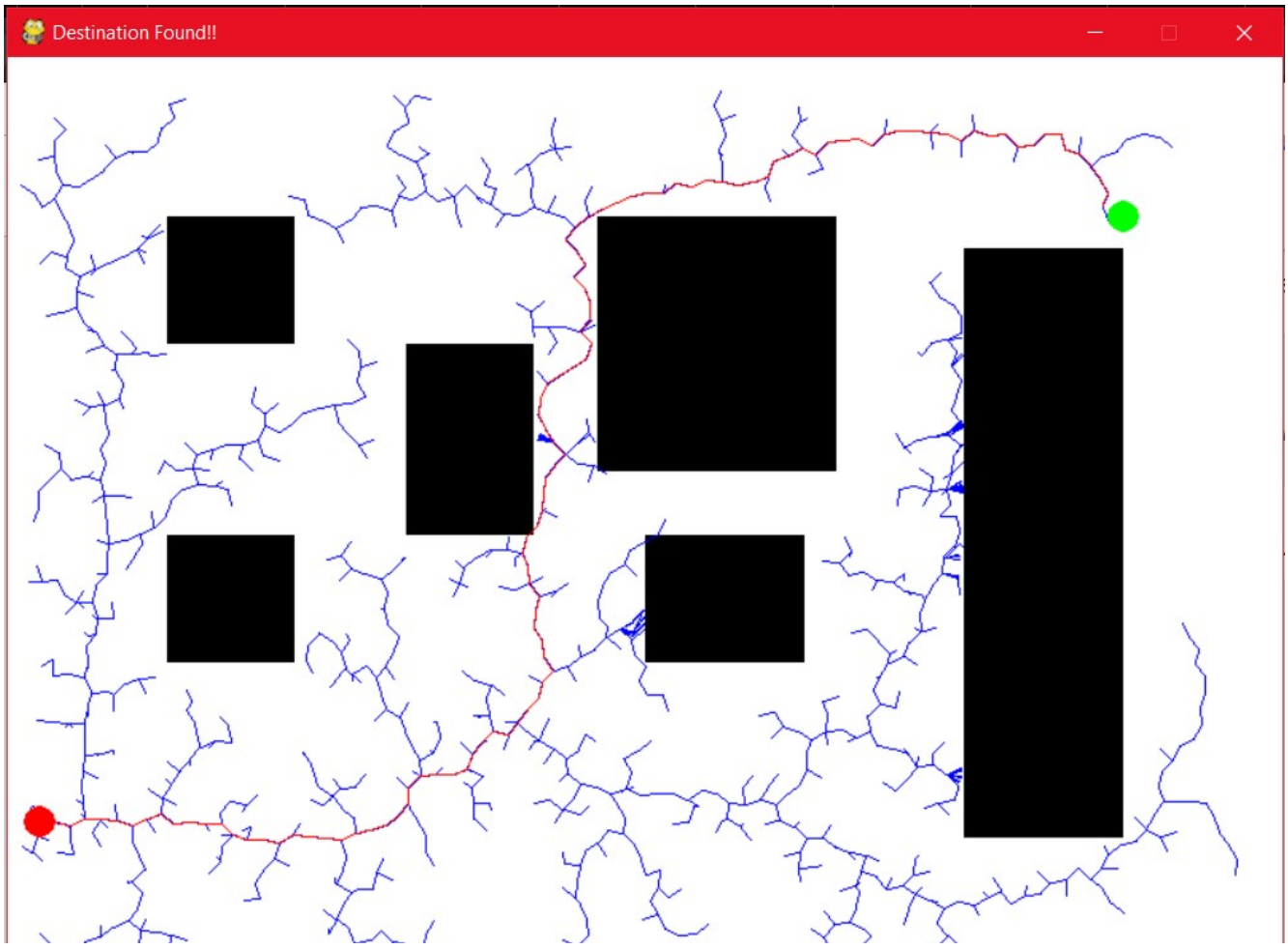
In [3]:
```
# Initialization
pygame.init()
clock = pygame.time.Clock() # Start the clock
screen = pygame.display.set_mode((WIDTH, HEIGHT)) # Create the screen

# Defined node object
class Node(object):
    def __init__(self, point, parent):
        self.point = point # Holds the node coordinates
        self.parent = parent # Holds the parent coordinates

# Function to calculate distance between two points
def dist(p1,p2):
    return sqrt((p1[0]-p2[0])*(p1[0]-p2[0])+(p1[1]-p2[1])*(p1[1]-p2[1]))

# Function to check if the points are in range of each other
def inRange(p1, p2, radius):
    distance = dist(p1,p2)
    if (distance <= radius): return True
    return False

# Clips the point so that the distance between two points is less than threshold
distance
def clip_point(p1,p2):
    if dist(p1,p2) < thresh: return p2
    else:
        theta = atan2(p2[1]-p1[1],p2[0]-p1[0])
        # Using distance form of line, the coordinates of point on the line join
ing the two are obtained
        return p1[0] + thresh*cos(theta), p1[1] + thresh*sin(theta)

# Returns True if the given coordinates are colliding with an obstacle
def isCollision(p):
    for obstacle in Obstacles:
        if obstacle.collidepoint(p) == True: return True
    return False

# This function returns a random point such that it does not collide with the ob
stacles
def get_random_point():
    # This loop returns a random in the workspace. If the generated point collid
es with an obstacle, it will generate again
    while True:
        p = random.random()*WIDTH, random.random()*HEIGHT # Generating a random
point in the workspace
        if not(isCollision(p)): return p

# This function draws the obstacles on the screen
def draw_obstacles():
    global Obstacles
    Obstacles = []
    Obstacles.append(pygame.Rect((WIDTH / 2.0 + 200, HEIGHT / 2.0 - 180),(100,37
0)))
    Obstacles.append(pygame.Rect((370,100),(150,160)))
    Obstacles.append(pygame.Rect((400,300),(100,80)))
    Obstacles.append(pygame.Rect((250,180),(80,120)))
    Obstacles.append(pygame.Rect((100,100),(80,80)))
    Obstacles.append(pygame.Rect((100,300),(80,80)))

    for obstacle in Obstacles:
        pygame.draw.rect(screen, BLACK, obstacle)


# This function resets the screen
def reset():
    global count
    screen.fill(WHITE)
    draw_obstacles()
```

In [4]:
```
rrt_algo()

# Waiting for user to give close window command
while True:
    for event in pygame.event.get():
    # Check for closing the window
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit
```

```
---------------------------------------------------------------------------
error                                     Traceback (most recent call last)
<ipython-input-4-cf4dfa90d521> in <module>
      3 # Waiting for user to give close window command
      4 while True:
----> 5     for event in pygame.event.get():
      6     # Check for closing the window
      7         if event.type == pygame.QUIT:

error: video system not initialized
```

In first image,threshold distance was 15. Second image has threshold distance of 10.


Destination Found!!


Destination Found!!