EX1 and EX2:The code below is the implementation of A star algorithm, after finding the route I save in png file the actual shortest path in BLACK and the search front areas in light color. Actually it can be seen from the picture that the algorithm goes throuh the pink line. It can be happened because in diagonal the line can be passed because there exist some gap. The shortest path is 1458. Overall time for the algorithm to complete the path is 107.98048543930054.

In [50]:
```
from PIL import Image
from matplotlib.pyplot import imshow
import copy
import numpy as np
from queue import Queue
from queue import PriorityQueue
from math import sqrt
import time

e = {}
v = {}
costs = [(173, 216, 230),(144, 238,144),(0, 0, 255),(255, 0, 0), (255, 255, 25
5), (0, 0, 0)]
for i in range (len(costs)):
    e[costs[i]] = True
    if i == 0:
        v[costs[i]] = 4
    elif i == 1:
        v[costs[i]] = 2
    else:
        v[costs[i]] = 1
visitedColor = [145, 145, 145]
```

In [51]:
```
img = Image.open('pixe_world.png')
data = np.array(img)
output = copy.deepcopy(data)
```

In [56]:
```
direction = [(0, 1), (1, 1), (-1, 1),  (1, -1), (0, -1), (-1, -1), (1, 0), (-1
, 0)]
from_vertex = {}
cost = {}
queue = PriorityQueue()
found = None

def euclid(u, v):
    return sqrt((u[0]-v[0])**2 + (u[1]-v[1])**2)

def a_star(start, dest):
    queue.put((0, start))
    cost[start] = 0
    best = None
    while not queue.empty():
        u = queue.get()[1]
        if u[0] == dest[0] and u[1] == dest[1]:
            found = u
            break
        cc = data[u[0]][u[1]]
        output[u[0]][u[1]] = [int((cc[i] + visitedColor[i])/2.0) for i in rang
e(len(cc))]
        for dir in direction:
            i, j = u[0] + dir[0], u[1] + dir[1]
            next_color = data[i][j]
            r, g, b = next_color
            if (r, g, b) not in e:
                continue
            incr = 1
            if abs(dir[0]) == abs(dir[1]):
                incr = sqrt(2)
            new_cost = cost[u] + incr*v[(r, g, b)]
            if (i, j) not in cost or new_cost < cost[(i, j)]:
                cost[(i, j)] = new_cost
                priority = new_cost + euclid((i,j), dest)
                queue.put((priority, (i,j)))
                from_vertex[(i, j)] = u
    path = [found]
    current = found
    while not current == start:
        current = from_vertex[current]
        path.append(current)
    path.reverse()
    return path
```

In [57]:
```
begin = time.time()
path = a_star((180, 280), (1021, 900))
end = time.time()
result_time = end-begin
print(result_time)
```

```
107.98048543930054
```

In [28]:
```
for vertex in path:
    output[vertex[0]][vertex[1]] = [int(([0, 0, 0][i] + visitedColor[i])/2.0)
for i in range(3)]
img = Image.fromarray(output.astype('uint8'))
img.save("3step_netice.png")
```

In [8]:
```
print(cost[path[len(path)-1]])
```

```
1458.976838871641
```

2018: Your Name, method, etc. Distance: xxxx.yyyyy