```
1. ##############################################################################
2.
3. #                      Reversing_Cellular_Automata.py                        #
4.
5. ##############################################################################
6.
7. from graphics import *
8. import sys, time
9. from random import randint
10.
11.
12. #custom info, the board width and height, the background color, the block color, and
    the wait time between each loop
13. dimension = 10
14. bg = "grey"
15. color = "yellow"
16. wait = 1.5
17. generations = 100
18.
19. #opens the window
20. win = GraphWin(width = 400, height = 400)
21. win.setBackground(bg)
22. win.setCoords(0, 0, dimension*10, dimension*10)
23.
24.
25. #creates empty displaySquares array
26. displaySquares = [[0 for j in range(dimension)] for i in range(dimension)]
27. x1=0
28. x2=10
29. y1=0
30. y2=10
31. #creates and displays all the blocks
32.
33. for i in range(dimension):
34.     for x in range(dimension):
35.         mySquare = Rectangle(Point(x1, y1), Point(x2, y2))
36.         mySquare.setFill(bg)
37.         mySquare.setOutline(bg)
38.         mySquare.draw(win)
39.         x1 = x1+10
40.         x2 = x2+10
41.         displaySquares[i][x] = mySquare
42.
43.     x1=0
44.     x2=10
45.     y1=y1+10
46.     y2=y2+10
47.
48. #creates two empty 2D Arrays
49. squareArray = [[0 for i in range (dimension)] for j in range (dimension)]
50. squareArray2 = [[0 for i in range (dimension)] for j in range (dimension)]
51.
52.
53. #set initial state as a random pattern
54. for i in range(len(squareArray)):
55.     for j in range(len(squareArray[1])):
56.         squareArray[i][j] = randint(0, 1)
```

```
        if squareArray[i][j] == 1:
            displaySquares[i][j].setFill(bg)
        else:
            displaySquares[i][j].setFill(color)

#run x generations
test = Text(Point(10,10),str(0))
test.draw(win)

for k in range(generations):
    test.setText(str(k))
    #sleep so we can actually see what's going on
    time.sleep(wait)
    #switches between two different 2D arrays, this makes sure we don't change the current generation
    if(k % 2 == 0):
        #resets the board
        squareArray2 = [[0 for i in range (dimension)] for j in range (dimension)]
        arr = squareArray2
        notarr = squareArray
    else:
        #resets the board
        squareArray = [[0 for i in range (dimension)] for j in range (dimension)]
        arr = squareArray
        notarr = squareArray2

    #iterate through the board
    for i in range (dimension):
        for j in range (dimension):
            #reset the count of neighbors
            count = 0

            #determine the immediate range of neighbors
            pos1 = i
            pos2 = j
            pos1_start = i - 1
            pos1_end = i + 1

            pos2_start = j - 1
            pos2_end = j + 1
            #check bounds of neighbors
            #hedge
            if(pos1 == 0):
                pos1_start = pos1
            if(pos1 == dimension-1):
                pos1_end = pos1
            if(pos2 == 0):
                pos2_start = pos2
            if(pos2 == dimension-1):
                pos2_end = pos2


            #check actual alive nieghbors
            for x in range(pos1_start,pos1_end+1):
                for y in range(pos2_start,pos2_end+1):
                    #do not check the square i'm in
                    if(not(x == pos1) or not(y == pos2)):
```

```
                        if(notarr[x][y] == 1):
                            count += 1

            #Rules for game of life
            if(notarr[i][j] == 1):
                if(count < 2):
                    arr[i][j] = 1
                    displaySquares[i][j].setFill(color)
                elif(count > 3):
                    arr[i][j] = 1
                    displaySquares[i][j].setFill(color)
                elif(count == 2 or count == 3):
                    arr[i][j] = 0
                    displaySquares[i][j].setFill(bg)
            elif(notarr[i][j] == 0):
                if(count == 3):
                    arr[i][j] = 0
                    displaySquares[i][j].setFill(bg)


win.close()
```

```
################################################################################
#                    Cellular_Automata_With_Pattern.py                         #
################################################################################


from graphics import *
import sys, time
from random import randint


#custom info, the board width and height, the background color, the block color, and the wait time between each loop
dimension = 10
bg = "grey"
color = "yellow"
wait = 1.5
generations = 100

#opens the window
win = GraphWin(width = 400, height = 400)
win.setBackground(bg)
win.setCoords(0, 0, dimension*10, dimension*10)


#creates empty displaySquares array
displaySquares = [[0 for j in range(dimension)] for i in range(dimension)]
x1=0
x2=10
y1=0
y2=10
#creates and displays all the blocks

for i in range(dimension):
    for x in range(dimension):
        mySquare = Rectangle(Point(x1, y1), Point(x2, y2))
        mySquare.setFill(bg)
        mySquare.setOutline(bg)
        mySquare.draw(win)
        x1 = x1+10
        x2 = x2+10
        displaySquares[i][x] = mySquare

    x1=0
    x2=10
    y1=y1+10
    y2=y2+10

#creates two empty 2D Arrays
squareArray = [[0 for i in range (dimension)] for j in range (dimension)]
squareArray2 = [[0 for i in range (dimension)] for j in range (dimension)]


#set initial state
def live(x, y):
    global squareArray
```

```
    global display_Squares
    squareArray[x][y] = 1
    displaySquares[x][y].setFill(color)

live(0, 0)
live(0, 1)
live(0, 2)
live(0, 7)
live(0, 9)
live(1, 0)
live(1, 3)
live(1, 2)
live(1, 6)
live(1, 8)
live(2, 0)
live(2, 1)
live(2, 4)
live(2, 6)
live(2, 7)
live(2, 8)
live(3, 0)
live(3, 1)
live(3, 2)
live(3, 3)
live(3, 4)
live(3, 5)
live(3, 6)
live(4, 0)
live(4, 6)
live(5, 0)
live(5, 2)
live(5, 3)
live(5, 5)
live(5, 6)
live(6, 2)
live(6, 4)
live(6, 6)
live(7, 0)
live(7, 1)
live(7, 2)
live(7, 3)
live(7, 4)
live(7, 6)
live(8, 0)
live(8, 1)
live(8, 2)
live(8, 3)
live(8, 5)
live(8, 6)
live(9, 0)
live(9, 1)
live(9, 2)
live(9, 3)

#run x generations
test = Text(Point(10,10),str(0))
test.draw(win)
```

```
114.
115. for k in range(generations):
116.     test.setText(str(k))
117.     #sleep so we can actually see what's going on
118.     time.sleep(wait)
119.     #switches between two different 2D arrays, this makes sure we don't change the
     current generation
120.     if(k % 2 == 0):
121.         #resets the board
122.         squareArray2 = [[0 for i in range (dimension)] for j in range (dimension)]
123.         arr = squareArray2
124.         notarr = squareArray
125.     else:
126.         #resets the board
127.         squareArray = [[0 for i in range (dimension)] for j in range (dimension)]
128.         arr = squareArray
129.         notarr = squareArray2
130.
131.     #iterate through the board
132.     for i in range (dimension):
133.         for j in range (dimension):
134.             #reset the count of neighbors
135.             count = 0
136.
137.             #determine the immediate range of neighbors
138.             pos1 = i
139.             pos2 = j
140.             pos1_start = i - 1
141.             pos1_end = i + 1
142.
143.             pos2_start = j - 1
144.             pos2_end = j + 1
145.             #check bounds of neighbors
146.             #hedge
147.             if(pos1 == 0):
148.                 pos1_start = pos1
149.             if(pos1 == dimension-1):
150.                 pos1_end = pos1
151.             if(pos2 == 0):
152.                 pos2_start = pos2
153.             if(pos2 == dimension-1):
154.                 pos2_end = pos2
155.
156.
157.             #check actual alive nieghbors
158.             for x in range(pos1_start,pos1_end+1):
159.                 for y in range(pos2_start,pos2_end+1):
160.                     #do not check the square i'm in
161.                     if(not(x == pos1) or not(y == pos2)):
162.                         if(notarr[x][y] == 1):
163.                             count += 1
164.
165.             #Rules for game of life
166.             if(notarr[i][j] == 1):
167.                 if(count < 2):
168.                     arr[i][j] = 0
169.                     displaySquares[i][j].setFill(bg)
```

```
170.                 elif(count > 3):
171.                     arr[i][j] = 0
172.                     displaySquares[i][j].setFill(bg)
173.                 elif(count == 2 or count == 3):
174.                     arr[i][j] = 1
175.                     displaySquares[i][j].setFill(color)
176.             elif(notarr[i][j] == 0):
177.                 if(count == 3):
178.                     arr[i][j] = 1
179.                     displaySquares[i][j].setFill(color)
180.
181.
182. win.close()
```