

Iteration 1:

Algoritme:

Start

Define 3 points with random X and Y value

Draw triangle from points

Make a object with random X and Y value inside the triangle

Draw the object

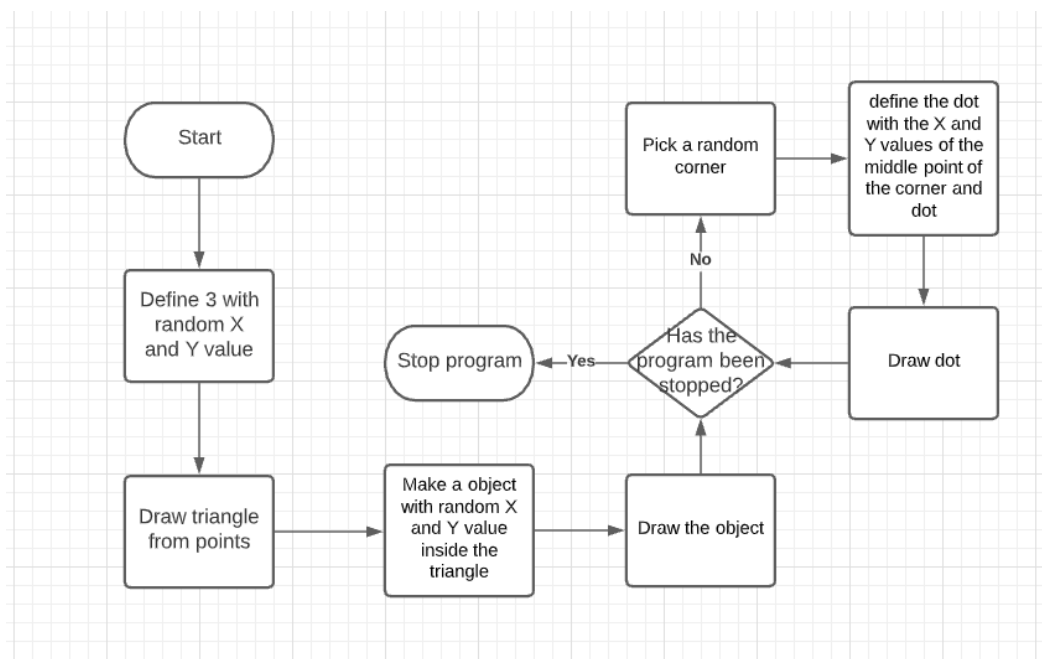
While the game hasnt been stopped

 Pick a random corner

 define the dot with the X and Y values of the middle point of the corner and dot

 Draw dot

Flowchart:



Kodeskelet:

```
main.py x
1
2
3 Housekeeping = 0
4 i = 0
5 def GenerateStartDots():
6     pass
7
8 def GenerateStartObj():
9     pass
10
11 def CycleOfDrawing():
12     pass
13 def ChangeObj():
14     pass
15 def DrawObj():
16     pass
17 GenerateStartDots()
18 GenerateStartObj()
19 while i < 100000:
20     ChangeObj()
21     CycleOfDrawing()
22     DrawObj()
23     i += 1
24
```

Iteration 2:

Jeg startede med at fylde klassen for Objects on screen ud. Der skulle ikke meget til da der kun skulle være tilfældige koordinater og en tom draw funktion for nu.

Derefter udfyldte jeg alle de funktioner der skulle køres, alle ud over change har ikke ændret sig siden dette skridt.

```
24 def GenerateStartDots():
25     global amountOfCorners
26     for x in range(amountOfCorners):
27         listOfCorners.append(ObjectOnScreen())
28
29 def GenerateStartObj():
30     global StartObj
31     StartObj = ObjectOnScreen()
32
33 def CycleOfDrawing():
34     StartObj.DrawObj()
```

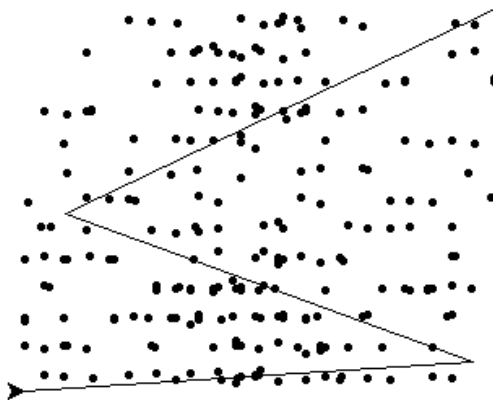
Derefter blev Turtle funktionen tilføjet. Det var ret simpelt da jeg fulgte tutorialen på <https://realpython.com/beginners-guide-python-turtle/>. Vi starter koden med at oprette turtle, og viduet den tegner i

```
4 s = turtle.getscreen()
5 t = turtle.Turtle()
6 t.hideturtle()
7 t.speed(100000)
8 turtle.screensize(0,800)
```

Så blev der tilføjet draw funktionen for både prikker og trekanten. Den tager to runder rundt i trekanten så trekanten bliver lukket af og der ikke sidder en side åben.

```
14 class ObejctOnScreen:
15
16     def __init__(self):
17         self.x = random.randint(0,400)
18         self.y = random.randint(0,400)
19     def DrawObj(self):
20         t.goto(self.x,self.y)
21         t.dot(3)
44 c = 0
45 for corner in listOfCorners * 2:
46     if c == 0:
47         t.penup()
48     else:
49         t.pendown()
50         t.goto(corner.x,corner.y)
51     c += 1
52     t.penup()
```

Derefter blev change funktionen lavet. Der var mange variationer der ikke virket. Jeg viser kun den mest spændene. Den første variation som tegnede noget på skærmen, fik det til at ligne den tegnede tilfældigt. Dette var fordi den tog det absolute midtpunkt af de to punkter. Det skabte problemer både fordi det var nu floats og ikke ints. Og fordi midtpunktet af skærmen er 0,0. Gjorde det at prikkenu fik de forkerte koordinater.



Change funktionen var det sidste der manglede og virke og endte med at se sådan ud.

```
35 def ChangeObj():
36     RandomCorner = random.randint(0,2)
37     ChosenCorner = listOfCorners[RandomCorner]
38     StartObj.x = (StartObj.x + ChosenCorner.x) / 2
39     StartObj.y = (StartObj.y + ChosenCorner.y) / 2
40
```