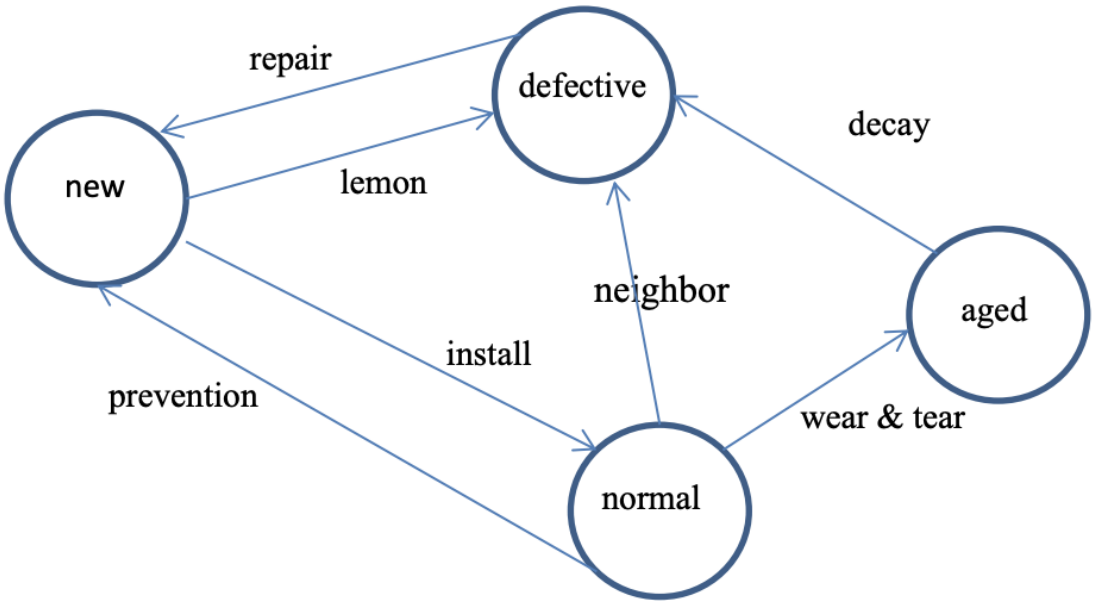
Kwounsu Lee

```
#!/usr/bin/python3.5
print("Start Program")

from tkinter import *
import random

class Simulation(Frame):
    def __init__(self, parent):
        Frame.__init__(self, parent)

        self.phase = 0
        self.defective = 0
        self.new = 0
        self.normal = 0
        self.aged = 0
        self.w, self.h = 21, 21
        self.number_of_tiles = self.w * self.h
        self.rectangle = [[0 for x in range(self.w)] for y in range(self.h)]
        self.memory = [[0 for x in range(self.w)] for y in range(self.h)]
        for i in range(0,self.w):
            for j in range(0,self.h):
                rand = random.randrange(4)
                self.memory[i][j] = rand
                if rand == 0:
                    self.defective += 1
                elif rand == 1:
                    self.new += 1
                elif rand == 2:
                    self.normal += 1
                elif rand == 3:
                    self.aged += 1

        self.parent = parent
        self.PrintMemory()

    def PrintMemory(self):
        self.parent.title("Kwounsu Lee")
        self.pack(fill=BOTH, expand=1)
        self.canvas = Canvas(self)
        for i in range(0,self.w):
            for j in range(0,self.h):
                if self.memory[i][j] == 0: # defective
                    self.rectangle[i][j] = self.canvas.create_rectangle(10+i*20,
 10+j*20, 30+i*20, 30+j*20,
                        outline="#fff", fill="#f50")
                elif self.memory[i][j] == 1: # new
                    self.rectangle[i][j] = self.canvas.create_rectangle(10+i*20,
 10+j*20, 30+i*20, 30+j*20,
                        outline="#fff", fill="#040")
                elif self.memory[i][j] == 2: # normal
                    self.rectangle[i][j] = self.canvas.create_rectangle(10+i*20,
 10+j*20, 30+i*20, 30+j*20,
                        outline="#fff", fill="#190")
                elif self.memory[i][j] == 3: # aged
```

```
                    self.rectangle[i][j] = self.canvas.create_rectangle(10+i*20,
10+j*20, 30+i*20, 30+j*20,
                        outline="#fff", fill="#2f0")
        if self.phase == 0:
            self.canvas.pack(fill=BOTH, expand=1)

        if self.phase == 0:
            self.text = StringVar()
            self.text.set("Phase "+str(self.phase))
            self.stats = StringVar()
            self.stats.set("Normal=> new: "+str(round(self.new/self.number_of_til
es*100,2))+"%, normal :"+str(round(self.normal/self.number_of_tiles*100,2))+"%, a
ged: "+str(round(self.aged/self.number_of_tiles*100,2))+"%, defective: "+str(roun
d(self.defective/self.number_of_tiles*100,2))+"%")
            self.phaseLabel = Label(self, textvariable=self.text)
            self.statsLabel = Label(self, textvariable=self.stats)
            self.phaseLabel.pack()
            self.statsLabel.pack()

    def neighborIsDefective(self, x, y):
        for i in range(-1,2):
            for j in range(-1,2):
                if i == 0 and j == 0:
                    pass
                elif x+i > -1 and x+i < self.w and y+j > -1 and y+j < self.h and
self.memory[x+i][y+j] == 0:
                    return 1
        return 0

    def change(self):
        print(str(self.phase)+"=> new: "+str(self.new)+", normal :"+str(self.norm
al)+", aged: "+str(self.aged)+", defective: "+str(self.defective))
        self.phase += 1
        for i in range(0,self.w):
            for j in range(0,self.h):
                if self.memory[i][j] == 0: # Case: defective
                    repairPercent = int(self.defective * 100 / (self.w * self.h))
                    rand = random.randrange(100)
                    if rand < repairPercent: # repair
                        self.memory[i][j] = 1
                        self.canvas.itemconfig(self.rectangle[i][j], fill="#040")
                        self.defective -= 1
                        self.new += 1
                elif self.memory[i][j] == 1: # Case: new
                    rand = random.randrange(100)
                    if rand < 5: # lemon
                        self.memory[i][j] = 0
                        self.canvas.itemconfig(self.rectangle[i][j], fill="#f50")
                        self.new -= 1
                        self.defective += 1
                    elif rand < 45: #install
                        self.canvas.itemconfig(self.rectangle[i][j], fill="#190")
                        self.new -= 1
                        self.normal += 1
                elif self.memory[i][j] == 2: # Case: normal
```

```
                    rand = random.randrange(100)
                    if rand < 15: # prevent
                        self.memory[i][j] = 1
                        self.canvas.itemconfig(self.rectangle[i][j], fill="#040")
                        self.normal -= 1
                        self.new += 1
                    elif rand < 20: # wear & tear
                        self.memory[i][j] = 3
                        self.canvas.itemconfig(self.rectangle[i][j], fill="#2f0")
                        self.normal -= 1
                        self.aged += 1
                    elif rand < 40: # neighbor
                        if self.neighborIsDefective(i, j) == 1:
                            self.memory[i][j] = 0
                            self.canvas.itemconfig(self.rectangle[i][j], fill="#f
50")
                            self.normal -= 1
                            self.defective += 1
                elif self.memory[i][j] == 3: # Case: aged
                    rand = random.randrange(100)
                    if rand < 10: # decay
                        self.memory[i][j] = 0
                        self.canvas.itemconfig(self.rectangle[i][j], fill="#f50")
                        self.aged -= 1
                        self.defective += 1
        if self.phase > 0:
            self.text.set("Phase "+str(self.phase))
            self.stats.set("Normal=> new: "+str(round(self.new/self.number_of_til
es*100,2))+"%, normal :"+str(round(self.normal/self.number_of_tiles*100,2))+"%, a
ged: "+str(round(self.aged/self.number_of_tiles*100,2))+"%, defective: "+str(roun
d(self.defective/self.number_of_tiles*100,2))+"%")

def main():
    root = Tk()
    sim = Simulation(root)
    root.geometry("460x500+100+100")

    change_button = Button(root, text="Next State", command=sim.change)

    change_button.pack()

    root.mainloop()

if __name__ == '__main__':
    main()

print("Terminate Program")
```

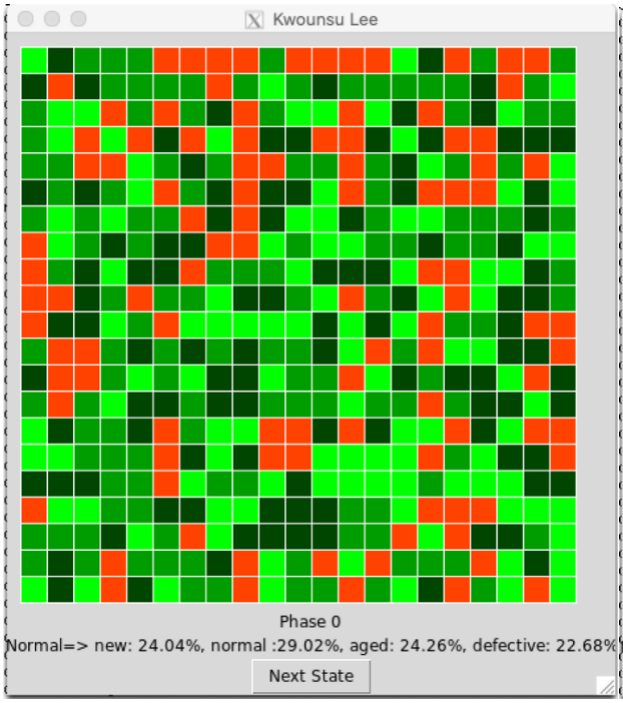
[Visual Concept]

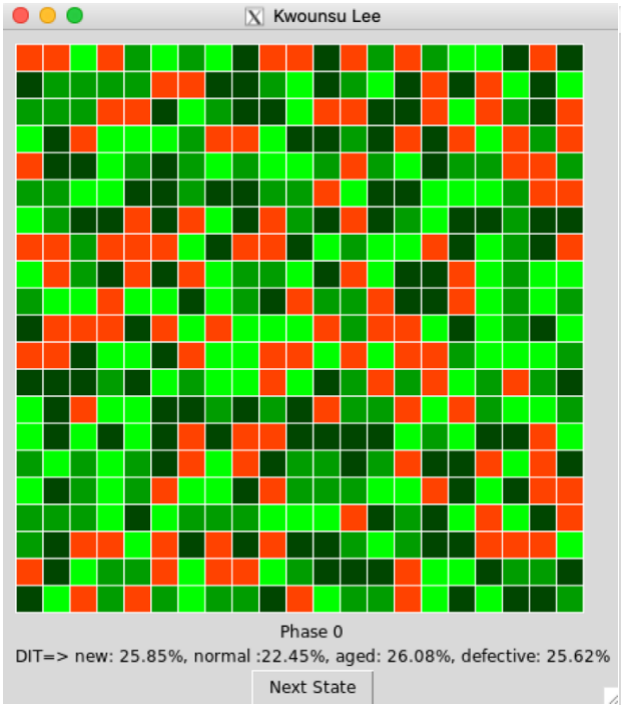


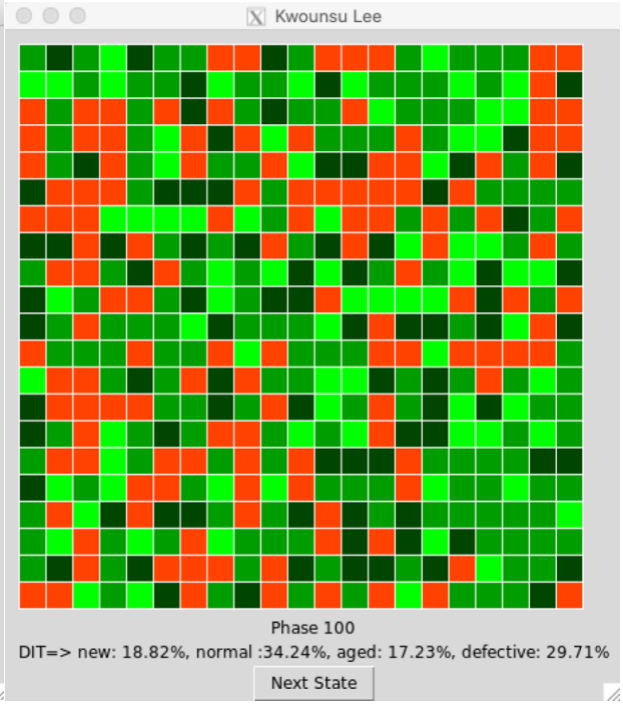
[STATS]

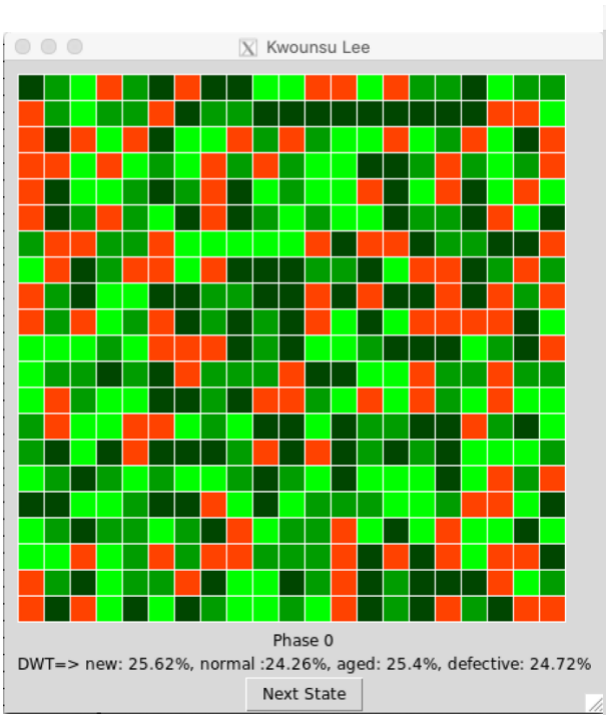
Condition	New %	Normal %	Aged %	Defective %
Base	29.02	30.84	13.15	26.98
Double install	18.82	34.24	17.23	29.71
Double wear & tear	21.09	13.15	38.78	26.98
Double decay	34.47	31.52	7.26	26.76
Double repair	33.56	30.61	15.19	20.63
Double prevent	24.04	21.77	31.75	22.45
Double lemon	15.87	23.58	34.24	26.3
Double neighbor	24.49	20.18	29.48	25.85

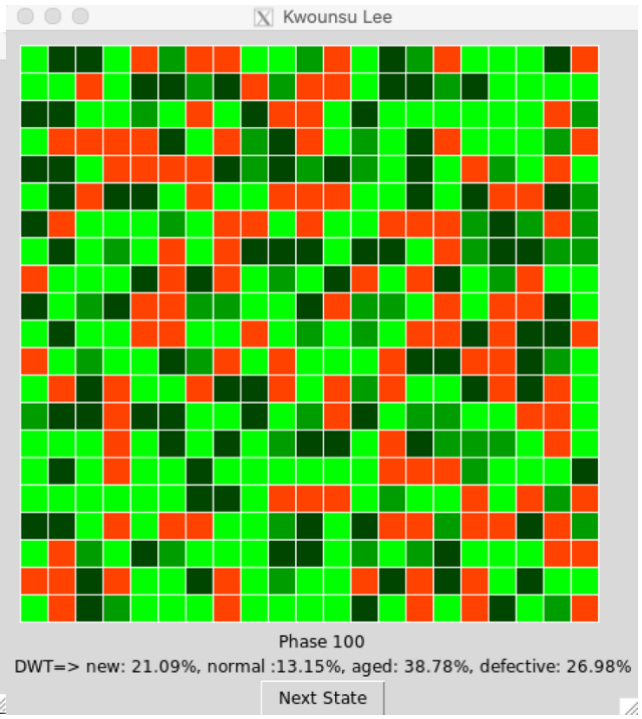
[Screen shots]

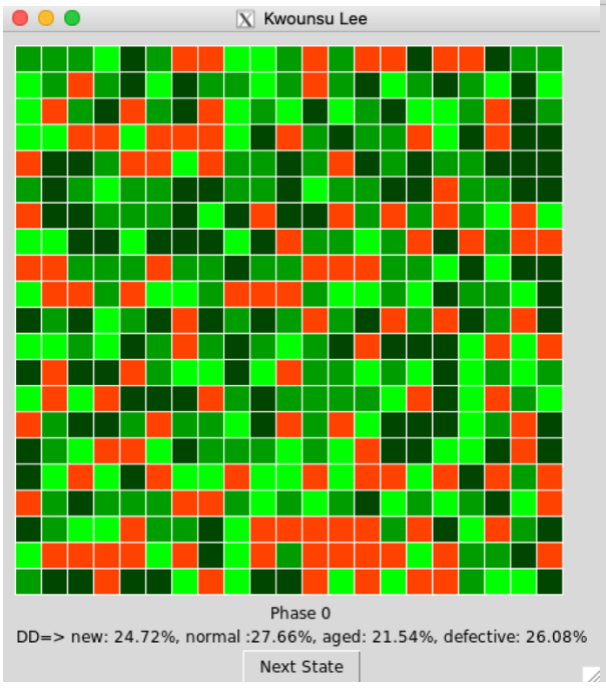


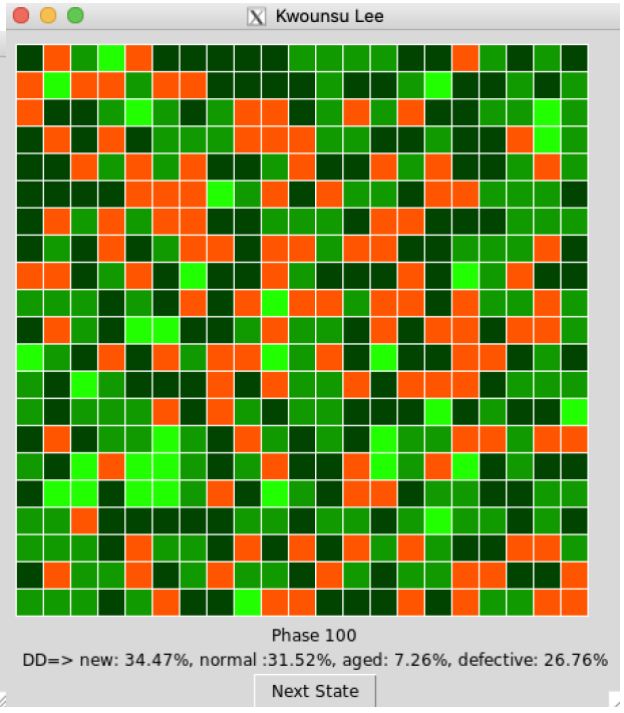


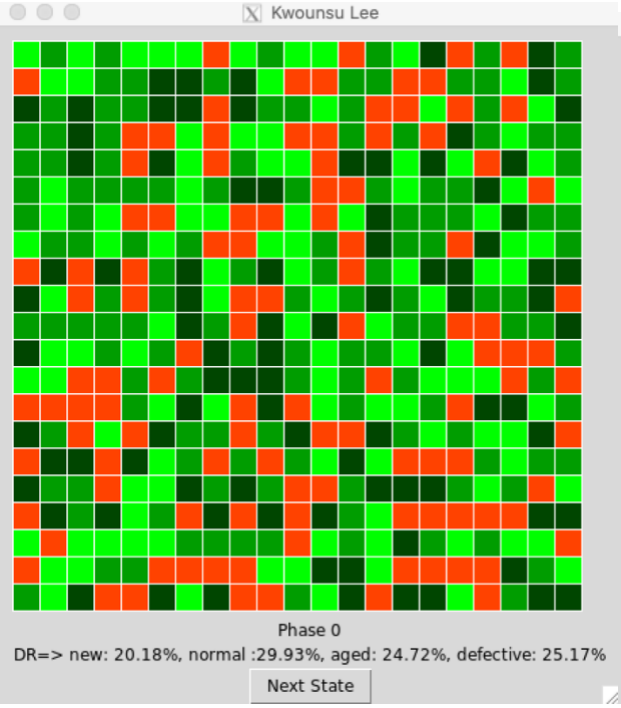


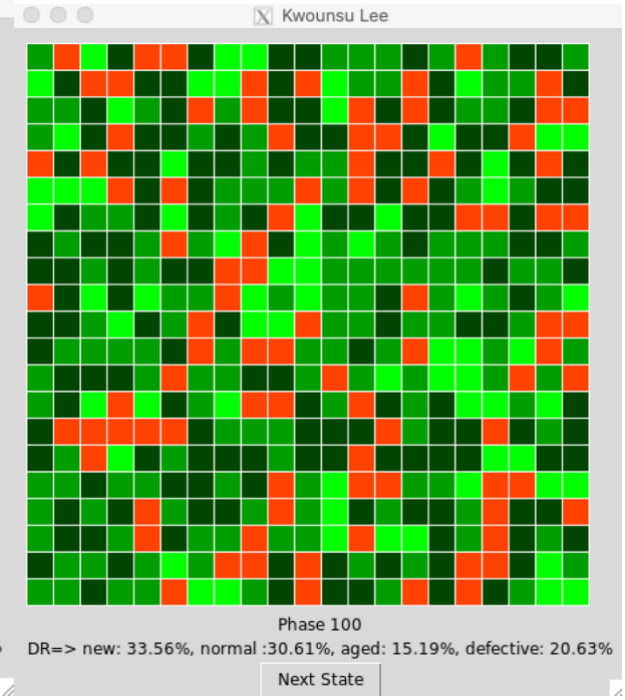

Kwounsu Lee
Phase 0
DWT=> new: 25.62%, normal :24.26%, aged: 25.4%, defective: 24.72%
Next State


Kwounsu Lee
Phase 100
DWT=> new: 21.09%, normal :13.15%, aged: 38.78%, defective: 26.98%
Next State

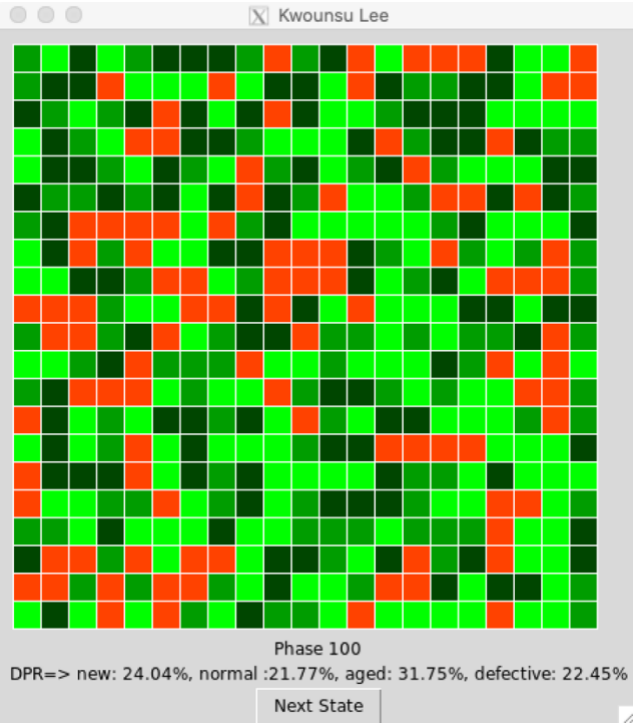

Kwounsu Lee
Phase 0
DD=> new: 24.72%, normal :27.66%, aged: 21.54%, defective: 26.08%
Next State

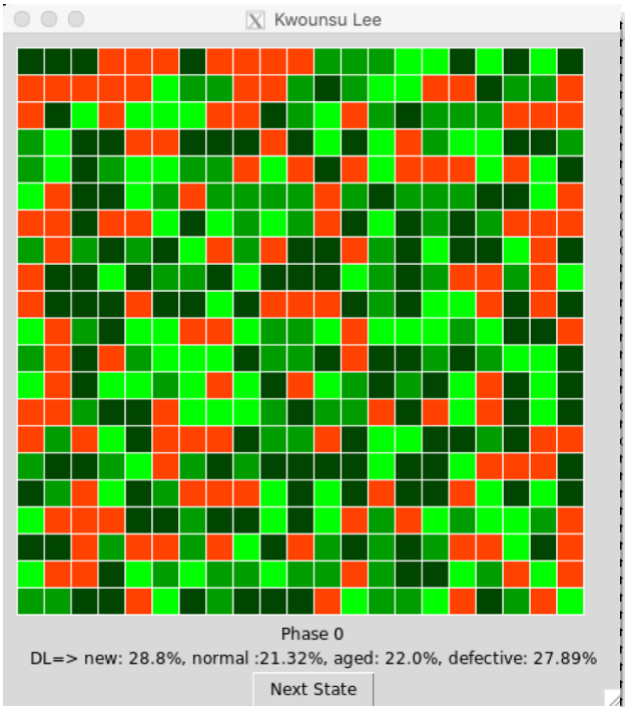

Kwounsu Lee
Phase 100
DD=> new: 34.47%, normal :31.52%, aged: 7.26%, defective: 26.76%
Next State

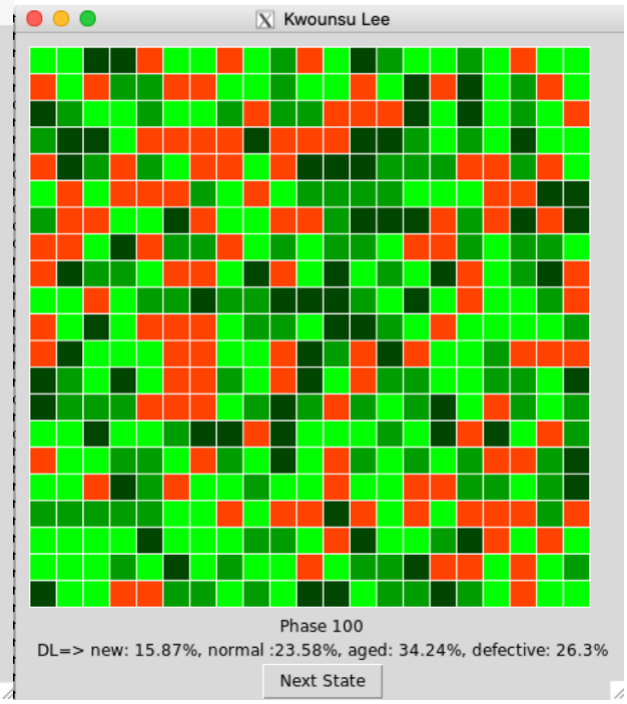

Kwounsu Lee
Phase 0
DR=> new: 20.18%, normal :29.93%, aged: 24.72%, defective: 25.17%
Next State

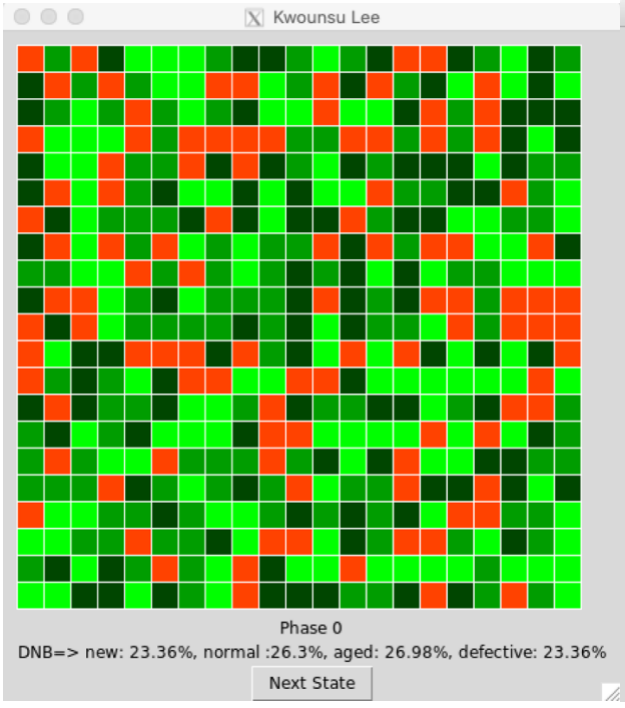

Kwounsu Lee
Phase 100
DR=> new: 33.56%, normal :30.61%, aged: 15.19%, defective: 20.63%
Next State

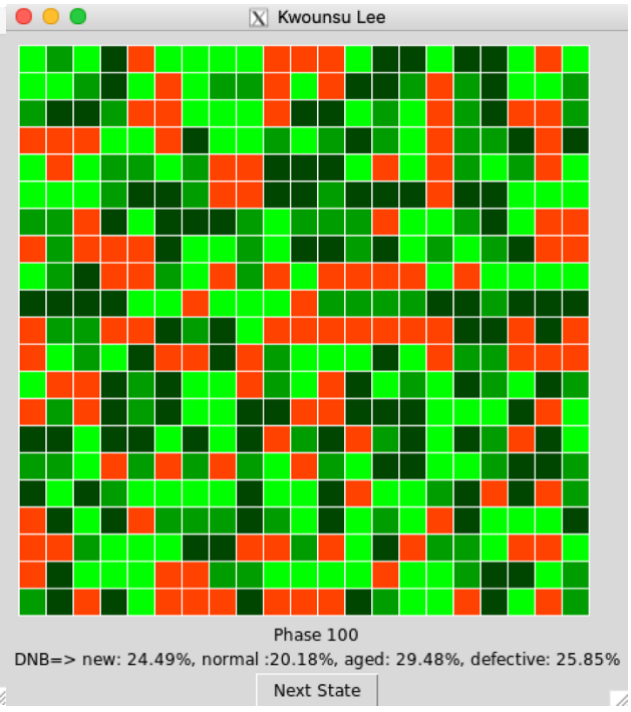

Kwounsu Lee
Phase 0
DPR=> new: 26.08%, normal :23.36%, aged: 26.08%, defective: 24.49%
Next State


Kwounsu Lee
Phase 100
DPR=> new: 24.04%, normal :21.77%, aged: 31.75%, defective: 22.45%
Next State


Kwounsu Lee
Phase 0
DL=> new: 28.8%, normal :21.32%, aged: 22.0%, defective: 27.89%
Next State


Kwounsu Lee
Phase 100
DL=> new: 15.87%, normal :23.58%, aged: 34.24%, defective: 26.3%
Next State


Kwounsu Lee
Phase 0
DNB=> new: 23.36%, normal :26.3%, aged: 26.98%, defective: 23.36%
Next State


Kwounsu Lee
Phase 100
DNB=> new: 24.49%, normal :20.18%, aged: 29.48%, defective: 25.85%
Next State