```python
 #########################


from kivy.app import App
from kivy.lang import Builder
from kivy.uix.screenmanager import ScreenManager, Screen
from kivy.uix.gridlayout import GridLayout
from kivy.uix.button import Button
from CustomModules import CustomGraphics
from kivy.uix.screenmanager import SlideTransition
from kivy.uix.boxlayout import BoxLayout
from kivy.config import Config
from kivy.graphics import Ellipse
from kivy.graphics import Triangle
from kivy.graphics import Color as kvColor
import time
from multiprocessing import Process, Value, Array
from threading import Thread, Event
from time import sleep
import math
import json
from espaceDeTuples import espaceDeTuples
from bcolors import bcolors
from agent import *


delai = 15

tabAgent = list()

cartes = [1, 2, 3, 4, 5, 6]

def initialisationAutorisationTuple(ts):
    for batiment in data["batiments"]:
        for badgeuse in batiment['informations']['badgeuses']:
            for carte in badgeuse["cartes"]:
                ts.OUT(("autorisationCarte", badgeuse["entree"], carte['id'],
carte["autorise"]))
                ts.OUT(("autorisationCarte", badgeuse["sortie"], carte['id'],
carte["autorise"]))

def lancementAgents(tab):
    for agent in tab:
        agent.start()

def indexBatiment(string):
    for i in range(len(data)):
        if data["batiments"][i]["name"] == string:
            return i
    return -1

def initialisationAgentBadgeuse(tsBatiment, tsAutorisation, tsPersonne,
tabBadgeuse):
    res = []
    i = 0
    for badgeuse in tabBadgeuse:
        batiment = trouverBatiment(badgeuse["id"], badgeuse["batiment"])
        numBatiment = indexBatiment(batiment)
        agentVerifCarte = Thread(target=verifCarte, args=(tsBatiment,
tsAutorisation, badgeuse["id"]), daemon=True)
```

```python
        agentScanCarte = Thread(target=scanCarte,
                                args=(tsBatiment, badgeuse["id"], "batiment" if badgeuse["batiment"] else "salle"),
                                daemon=True)
        agentLumiereVerte = Thread(target=lumiereVerte, args=(tsBatiment, badgeuse["id"]), daemon=True)
        agentLumiereRouge = Thread(target=lumiereRouge, args=(tsBatiment, badgeuse["id"]), daemon=True)
        agentDetectionPassage = Thread(target=detectionPassage, args=(tsBatiment, tsPersonne, badgeuse["id"]),daemon=True)
        agentAlarme = Thread(target=declencheAlarme, args=[tsBatiment], daemon=True)
        agentIncendie = Thread(target=incendie, args=(tsBatiment,numBatiment), daemon=True)
        if badgeuse["id"] % 2 == 1:
            agentPorte = Thread(target=etatPorte, args=(tsBatiment,batiment,True),daemon=True)
            res.append(agentPorte)
        tsBatiment.OUT(("nbPersonnesPassees", badgeuse["id"], 0))

        res.append(agentVerifCarte)
        res.append(agentScanCarte)
        res.append(agentLumiereVerte)
        res.append(agentLumiereRouge)
        res.append(agentDetectionPassage)
        res.append(agentAlarme)
        res.append(agentIncendie)

    return res
    # agents =    [agentVerifCarte,agentScanCarte,
    #             agentLumiereVerte,agentLumiereRouge,agentDetectionPassage,
    #             ]

def start(tsBatiment, idBadgeuse, carte):
    agentLecteurCarte = Thread(target=lecteurCarte, args=(tsBatiment, idBadgeuse, carte), daemon=True)
    agentLecteurCarte.start()

def test():
    tupleSpaces = list()

    badgeuseTest = data['batiments'][0]['informations']['badgeuses'][0]['entree']
    badgeuseTest2 = data['batiments'][1]['informations']['badgeuses'][0]['entree']
    badgeuseTest3 = data['batiments'][2]['informations']['badgeuses'][0]['entree']
    carteTest1 = cartes[0]
    carteTest2 = cartes[1]


    initialisationAutorisationTuple(tsAutorisation)

    # agentLecteurCarte = Thread(target=lecteurCarte, args=(tsBatiment, badgeuseTest, carteTest1), daemon=True)
    # tsPersonne.OUT(("personnePresente", 1, 11, "batiment"))
    # personnesPresentes(tsPersonne)
    tabBadgeuse = allBadgeuse()
    agents = initialisationAgentBadgeuse(tsBatiment, tsAutorisation, tsPersonne, tabBadgeuse)

    agentLecteurCarte = Thread(target=lecteurCarte, args=(tsBatiment, badgeuseTest, cartes[0]), daemon=True)
    agentLecteurCarte.start()
    lancementAgents(agents)
    # agentAlarme = Thread(target=declencheAlarme, args=[tsBatiment], daemon=True)
    # agentVerifCarte = Thread(target=verifCarte, args=(tsBatiment, tsAutorisation,badgeuseTest), daemon=True)
    # agentScanCarte = Thread(target=scanCarte, args=(tsBatiment,badgeuseTest,"batiment"), daemon=True)
    # agentLumiereVerte = Thread(target=lumiereVerte, args=(tsBatiment,badgeuseTest), daemon=True)
    # agentLumiereRouge = Thread(target=lumiereRouge, args=(tsBatiment,badgeuseTest), daemon=True)
    # agentDetectionPassage = Thread(target=detectionPassage, args=(tsBatiment, tsPersonne,badgeuseTest), daemon=True)
    # agents = [agentLecteurCarte,agentAlarme]


def allBadgeuse():
    badgeuses = []
    for batiment in data['batiments']:
        for badgeuse in batiment['informations']['badgeuses']:
            badgeuses.append({
                "id": badgeuse["entree"],
                "batiment": badgeuse["batiment"]
            })
            badgeuses.append({
                "id": badgeuse["sortie"],
                "batiment": badgeuse["batiment"]
            })
    return badgeuses

tsPersonne = espaceDeTuples()
tsBatiment = espaceDeTuples()
tsAutorisation = espaceDeTuples()


def initialisationAgent(app):
    print("Agent lance")
    initialisationAutorisationTuple(tsAutorisation)

    tabBadgeuse = allBadgeuse()

    agents = initialisationAgentBadgeuse(tsBatiment, tsAutorisation, tsPersonne, tabBadgeuse)

    lancementAgents(agents)

    # TODO : Lancement fenetre Nico

    agentListenGreen = Thread(target = app.mainScreen.listenGreen, args=[tsBatiment], daemon = True)
    agentListenRed = Thread(target = app.mainScreen.listenRed, args=[tsBatiment], daemon = True)
    agentListenFire = Thread(target = app.mainScreen.listenFire, args=[tsBatiment], daemon = True)


    agentListenFire.start()
    agentListenGreen.start()
    agentListenRed.start()

def personnesPresentes(tsPersonne):
    f = open("personnePresente.txt", "w")
    i = 1
    for personne in tsPersonne.listeTuples:
        res = []
        res.append(personne[1])
        res.append(personne[2])
        res.append(personne[3])
        f.write(str(i) + " -  nom : " + str(data["cartes"][str(res[0])] ) +", id badgeuse : " + str(res[1]) + ", type badgeuse : " + str(res[2]))
        i += 1
    f.close()


def videFichiers():
    f1 = open("personnePresente.txt", "w")
    f2 = open("logPassage.txt", "w")
    f1.write('')
    f2.write('')
    f1.close()
    f2.close()

kivyApp = None
def startScreen():
    global kivyApp
    kivyApp = app()
    kivyApp.run()


def main():
    screen = Thread(target = startScreen, daemon = True)
    screen.start()
    sleep(1)



    videFichiers()
    #test()

    initialisationAgent(kivyApp)
    screen.join()



class MainScreen(BoxLayout):
    card = 0
    idBadgeuse = 0
    estBatiment = False
    entree = True

    WHITE =     [1,1,1,1]
    RED =       [1,0,0,1]
    GREEN =     [0,1,0,1]
    FIRE =      [1,0.5,0,1]

    def vraiIdBadgeuse(self):
        return self.idBadgeuse if self.entree else self.idBadgeuse + 1

    def check_card(self):
        print("_________________________________")
        print("badg : " + str(self.vraiIdBadgeuse()))
        print("bat  : " + str(self.estBatiment))
        print("Entrer / sortir : " + str(self.entree))
        print("cart ", self.card)
        global tsBatiment
        lecteurCarte(tsBatiment,self.vraiIdBadgeuse(),self.card)
        for i in tsBatiment.listeTuples:
            print(i)

    def add_person(self):
        global tsBatiment
        tsBatiment.OUT(("capteurPassage", self.vraiIdBadgeuse()))


    def redraw(self, green, red, fire):
        c = self.ids.floatlayout.canvas
        with c:
            c.get_group('a').clear()
            kvColor(green[0], green[1], green[2], green[3])
            c.add(Ellipse(pos=(112, 418), size=(80, 80)))

            kvColor(red[0], red[1], red[2], red[3])
            c.add(Ellipse(pos=(112, 320), size=(80, 80)))

            kvColor(fire[0], fire[1], fire[2], fire[3])
            c.add(Triangle(points=(112,218,152,298,192,218)))


    def change_to_green(self):
        self.redraw(self.GREEN, self.WHITE, self.WHITE)

    def change_to_red(self):
        self.redraw(self.WHITE, self.RED, self.WHITE)

    def change_to_fire(self):
        c = self.FIRE
        self.redraw(self.WHITE, self.WHITE, c)

    def mettreFeu(self):
        global tsBatiment
        tsBatiment.OUT(("incendie",indexBatiment(trouverBatiment(self.idBadgeuse,self.estBatiment))))

    def listenGreen(self, tsBatiment):

        tsBatiment.IN(("turnOnLightGreen",0),[])
        self.change_to_green()
        tsBatiment.IN(("turnOffLightGreen",0),[])
        self.change_to_white()
        self.listenGreen(tsBatiment)

    def listenRed(self, tsBatiment):

        tsBatiment.IN(("turnOnLightRed",0),[])
        self.change_to_red()
        tsBatiment.IN(("turnOffLightRed",0),[])
        self.change_to_white()
        self.listenRed(tsBatiment)

    def listenFire(self, tsBatiment):
        tsBatiment.IN(("turnOnLightFire",0),[])
        self.change_to_fire()

    def change_to_white(self):
        self.redraw(self.WHITE, self.WHITE, self.WHITE)

    def print_logs(self):
        global tsPersonne
        personnesPresentes(tsPersonne)

    def __init__(self):
        super().__init__()
        global tsBatiment



class app(App):

    def build(self):
        Config.set('graphics', 'width', '1280')
        Config.set('graphics', 'height', '720')
        Builder.load_file('./builder.kv')
        self.mainScreen = MainScreen()
        return self.mainScreen





if __name__ == '__main__':
    with open('config.json') as json_file:
        global data
        data = json.load(json_file)
    main()
```