

Homework 2

Simulation Engineering

Mateusz Turzyniecki

547299

Simulation of a traffic light system at a pedestrian crosswalk

Car traffic lights class can be in states:

- sigG – green light
- sigY – yellow light
- sigR – red light

It also has timers for all lights:

- greenTime
- yellowTime
- redTime

It also can remember what was it's previous state.

- PreviousState

```
class CarTrafficLights:
    def __init__(self, redTime, yellowTime, greenTime, state, previousState, button):
        self.redTime = redTime
        self.yellowTime = yellowTime
        self.greenTime = greenTime
        self.state = state
        self.previousState = previousState
        self.button = button
```

Init function of car traffic lights class.

It can change lights. It also has logic implemented for a button for pedestrian to change light.

```
def redLight(self):
    if self.state == "sigR":
        self.redTime += 1
    if self.redTime == 61:
        self.previousState = self.state
        self.state = "sigY"
        self.redTime = 0
```

This function changes light from red to yellow.

```
def yellowLight(self):
    if self.previousState == "sigR":
        if self.state == "sigY":
            self.yellowTime += 1
        if self.yellowTime == 6:
            self.previousState = self.state
            self.state = "sigG"
            self.yellowTime = 0
    if self.previousState == "sigG":
        if self.state == "sigY":
            self.yellowTime += 1
        if self.yellowTime == 5:
            self.previousState = self.state
            self.state = "sigR"
            self.yellowTime = 0
```

This function changes light from yellow to red or green based on previous state.

```
def greenLight(self):
    if self.state == "sigG":
        self.greenTime += 1
    if self.button == True:
        if self.greenTime == 60:
            self.button = False
            self.previousState = self.state
            self.state = "sigY"
            self.greenTime = 0
```

This function changes from green light to yellow based on a button state.

Pedestrian crosswalk lights class can be in states:

- sigG – green light
- sigY – yellow light
- sigR – red light

```
class PedestrianCrosswalkLights:
    def __init__(self, state):
        self.state = state
```

Init function of pedestrian crosswalk lights class

```

from car import CarTrafficLights
from pedestrian import PedestrianCrosswalkLights

simulationTime = 1
CTL = CarTrafficLights(0, 0, 0, "sigR", "none", False)
PCL = PedestrianCrosswalkLights("sigG")
while simulationTime < 201:
    CTL.redLight()
    CTL.yellowLight()
    CTL.greenLight()
    if CTL.state == "sigR":
        PCL.state = "sigG"
    if CTL.state == "sigY":
        PCL.state = "sigR"
    if CTL.state == "sigG":
        PCL.state = "sigR"
    print("simulation time:", simulationTime, "current light for cars:", CTL.state, "current light for pedestrians:",
PCL.state)
    if simulationTime == 70:
        CTL.button = True
        print("Button pressed!")
    simulationTime += 1

```

Main function of simulation

```

simulationTime = 1
CTL = CarTrafficLights(0, 0, 0, "sigR", "none", False)
PCL = PedestrianCrosswalkLights("sigG")

```

Here the starting time of simulation and its objects are determined.

```

while simulationTime < 201:

```

Simulation loop

```

CTL.redLight()
CTL.yellowLight()
CTL.greenLight()

```

Calling of functions in object car traffic lights

```

if CTL.state == "sigR":
    PCL.state = "sigG"
if CTL.state == "sigY":
    PCL.state = "sigR"
if CTL.state == "sigG":
    PCL.state = "sigR"

```

Binding the state of car traffic lights with the state of pedestrian crosswalk lights

```

print("simulation time:", simulationTime, "current light for cars:", CTL.state, "current light for pedestrians:", PCL.state)

```

Output of the simulation state to the console.

```

if simulationTime == 70:
    CTL.button = True
    print("Button pressed!")

```

Simulating button press by pedestrian in the 70th second of the simulation.

```

simulationTime += 1

```

Increment of the simulation time.

[illegible]

[illegible]

[illegible]

Console output from the simulation.