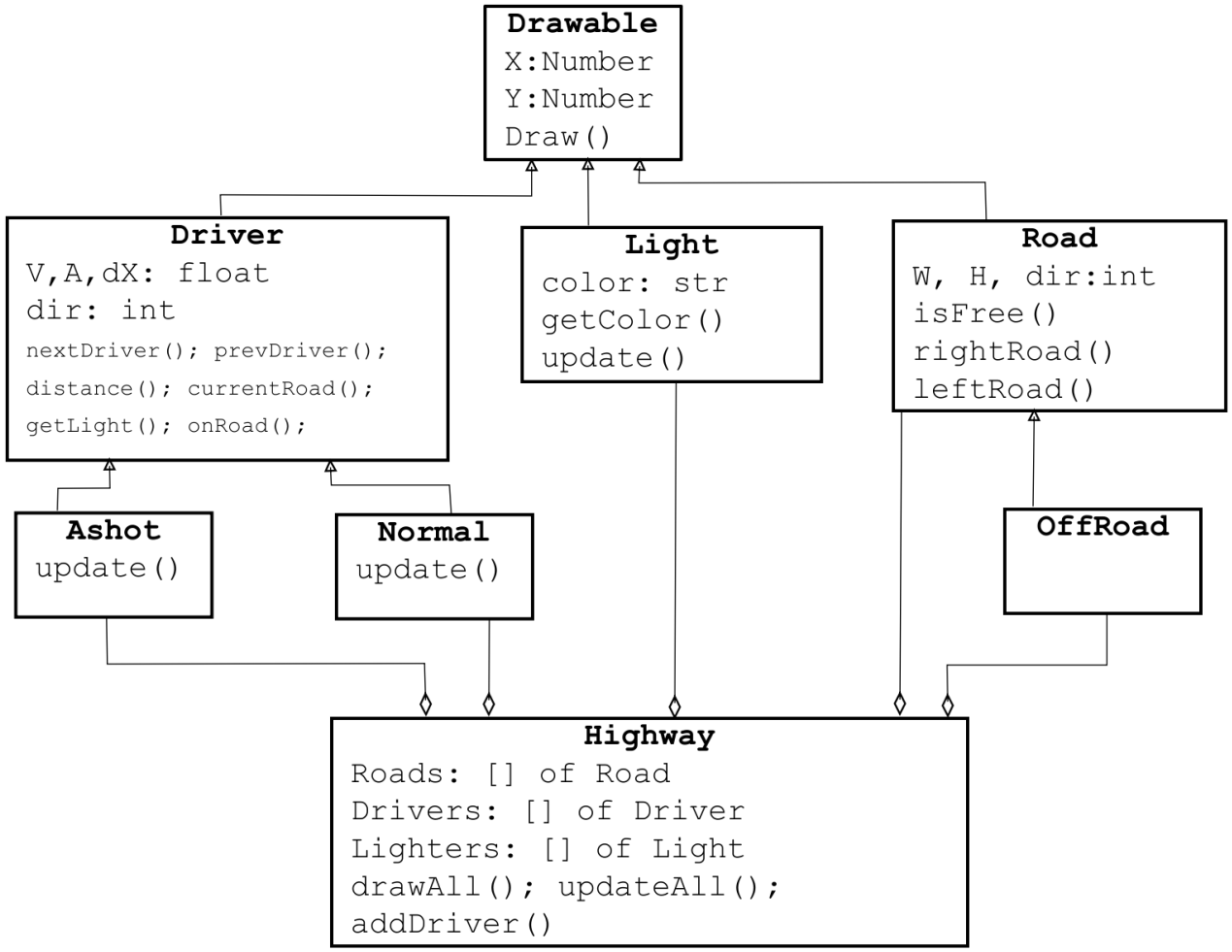
Архитектура симулятора дорожного движения RoadSim



Процедура **main():**
Выполняет инициализацию экземпляра Tkinter, и инициализирует объект класса Highway с необходимой конфигурацией дорог, фонарей, и критериев генерации водителей. По окончании инициализации программа переходит в цикл постоянного обновления представления, что таким образом позволяет пользователю лицезреть динамические свойства модели посредством её анимированности.

Класс **Highway:**
Представляет собой саму шоссе, то есть совокупность доступных для движения линий(**Road),** светофоров(**Light**) и машин(**Driver**). В ходе вызова функции обновления *updateAll*, все объекты, причастные к данному экземпляру шоссе, меняют свои характеристики в зависимости от прошедшего с последнего обновления времени(Светофоры могут поменять цвет, водители меняют свои координаты). В задачи функции update также входит удаление водителей, которые выехали за пределы наблюдаемого участка дороги. Далее, пользуясь обновлёнными данными о всех объектах, следует отрисовать все изменения на холсте посредством вызова *drawAll*. Данная функция отрисовывает все объекты согласно их параметрам, а также дополнительные графические элементы, касающиеся непосредственно шоссе: например, дорожную разметку,

пешеходную “зебру”, непреодолимые полосы. Для добавления нового участника к дорожному движению следует использовать функцию *addDriver.*

Класс **Drawable:**

Данный класс является абстрактным классом, потомками которого являются все классы, имеющие графическое представление в данной модели, т.е. **Road, Driver, Light**. Так как данные класы могут отрисовываться по-разному, например, как эллипсы, или как прямоугольники, что влияет на необходимые исходные данные, сам метод отрисовки *draw* не содержит прямых инструкций по функциям отображения фигуры, и данные инструкции должны быть указаны в соответствующем методе классов-потомков. Однако, класс Drawable предоставляет общий доступ к параметрам метосположения X и Y.

Класс **Light:**

Имеет поле c*olor*, обозначающее текущий цвет светофора. Цвет меняется при выполнении функции *update* в зависимости от внутренних временных интервалов. Также, данные интервалы могут быть настраиваемым элементом модели. Рекомендуется графическое представление в виде эллипса соответствующего текущему цвету цвета.

Класс **Driver:**

Персонализация водителя и его транспортного средства. В качестве полей имеет текущую позицию, ускорение(A), скорость(V), направление движения(dir), и фактор смещения dX, указывающий, совершает ли в настоящий водитель перестроение в другой ряд, и если да, то в какую сторону (0 – не соврешает, >1 – перестроение в сторону по увеличению X, <1 – перестроение в сторону по уменьшению X). Экземпляр должен иметь ссылку на объект Highway, к которому данный водитель принадлежит, поскольку этот объект будет необходим для получения информации о ситуации на дороге посредством следующих функций:
nextDriver() - получение объекта водителя, едущего впереди данного водителя
prevDriver() - получение объекта водителя, едущего позади данного водителя
distance() - получение дистанции до заданного в аргументе водителя
getLight() - получение информации о цвете светофора для дорожной линии по которой едет водитель, если этот светофор в поле видимости
currentRoad() - получение объекта дорожной линии, по которой в настоящий момент едет водитель
align() - получение информации о том, едет ли водитель ровно по дорожной линии, или он находится в состоянии смены линии

Представленные функции необходимы водителям, как говорилось, для получения обстановки для дороги, чтобы иметь возможность грамотно реагировать в соответствии с математической моделью, однако, сами реакции водителей описаны в потомках класса Driver, так как стратегии поведения различного рода водителей могут отличаться.
Графически водитель может отражатья также как эллипс, или же как квадрат или прямоугольник.

Класс **Normal:**

Класс **Ashot:**

Данные классы наследуют все поля и методы класса **Driver**, так как в полной мере являются водителями, и различаются они исполнением функции *update*, отвечающей за принятие водителем решений на основании текущей ситуации на дороге. Более подробное различие поведение можно узнать в математической модели.

Класс **Road:**

Дорожная полоса, по которой происходит движение. Характеризуется позиционными параметрами X, Y, а также шириной(W) и высотой(H). Также, у дороги есть направление (dir), в котором по данной дороге принято ехать.
Дорожная линия имеет методы *leftRoad()* и *rightRoad()*, которые возвращают объект соответствующей соседней дорожной линии или None если таковой нет. Метод *isFree()* позволяет получить информацию о свободности данной линии на обозримом интервале для того, чтобы водители могли обозревать соседние дорожные линии и перестраиваться на них по необходимости, обусловленной математической моделью.

Класс **OffRoad:**

Являет собой обочину и в текущей реализации ничего дополнительного к классу дорожной линии не добавляет, однако теоретически данный класс можно использовать для того чтобы, например, уменьшать скорость движения водителя по обочине, что сильно может поменять динамику модели.

Получить информацию об организации архитеуктуры также можно посредством интерпретатора python: для этого следует выполнить операции

```
>>>  import arc
>>>  help(arc)
```

где arc.py – это файл с нижеприлагаемым шаблоном описания архитектуры.

Шаблон-пример реализации данной архитектуры на языке Python:

```
#!/usr/bin/python3
#coding=utf-8

from tkinter import *

class Highway(object):
  """
Highway have links on all roads, lights and drivers, an can
iterately loop throuph them to update or to draw.
  """

  def update(self, time):
    """Updates coordinates of all drivers and lights, removing outdrived drivers"""
    pass

  def addDriver(self, driver,road):
    """Generates new driver on this road"""
    pass

  def draw(self, can):
    """Draws all the objects on provided Canvas object"""
    pass

class Drawable(object):
  """The object, that can be drawed!"""

  def draw(self, can):
    """Draws this"""
    pass

  def getXY(self):
    """returns x and y of object"""
    pass

class Road(Drawable):
  """
```

```
    The road has:
    X position
    Y position
    Width
    Height
    Direction of move(Up, Down)
  """

  def getRight(self):
    """Get right to this road, or None"""
    pass

  def getLeft(self):
    """Get left to this road, or None"""
    pass

  def isFree(self,x0,x1):
    """Is it able to change current line to this line?"""
    pass

  def draw(self, can):
    """Draws this road"""
    pass

class OffRoad(Road):
  """Also a road, but not for all. OffRoad has another draw attributes(color, example)"""
  pass

class Light(Drawable):
  """The light. Can be yellow, red, or blue."""

  def update(self,time):
    """Updates the color of light"""
    pass

  def draw(self, can):
    """Draws the light on current location"""
    pass

  def getColor(self):
    """Current color of light"""
    pass

class Driver(Drawable):
  """
    Usuall driver abstract class. Must have link to current highway,
    to get information about drivers, roads, lights e.t.c
    Basic class attributes:
    X position
    Y position
    Speed
    Acceleration
    Direction(Up or Down?)
    dX - variable, detecting change of line, and which side to.
  """

  def update(self,time):
    """Updates logic of driver; Need to be overriden"""
    pass

  def isAligned(self):
    """Is the driver at center of current road?"""
    pass

  def currentRoad(self):
    """Road, on which the driver is currently riding."""
    pass

  def nextDriver(self):
    """Returns the next to you river on this road"""
    pass

  def prevDriver(self):
    """Returns the previos driver on this road"""
```

```
  def distance(self, obj):
    """Measures distance from driver to obj."""

  def draw(self,can):
    """Draws a driver"""
    pass

  def getAcc(self):
    """Return Acceleration of this driver"""
    pass

class NormalDriver(Driver):
  def update(self,time):
    """Drive Logic of normal Driver"""
    pass

class AshotDriver(Driver):
  def update(self,time):
    """Drive Logic of Ashot"""
    pass

def main():
  """Main loop: Creating TKinter instance, running update cycle"""
  pass


if __name__ == "__main__":
  main()
```