МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра «Вычислительные системы и технологии»

ОТЧЁТ

По лабораторной работе №3

по дисциплине «Аппаратное и программное обеспечение   
роботизированных систем»

«Программное обеспечение  
роботизированных систем»

ПРОВЕРИЛ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Гай В.Е.

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Круглов С.В.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Малинин М.Ю.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Попов В.А.

\_\_\_\_\_\_\_\_\_\_\_\_ Дятлов Д.

**17-В-1**

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

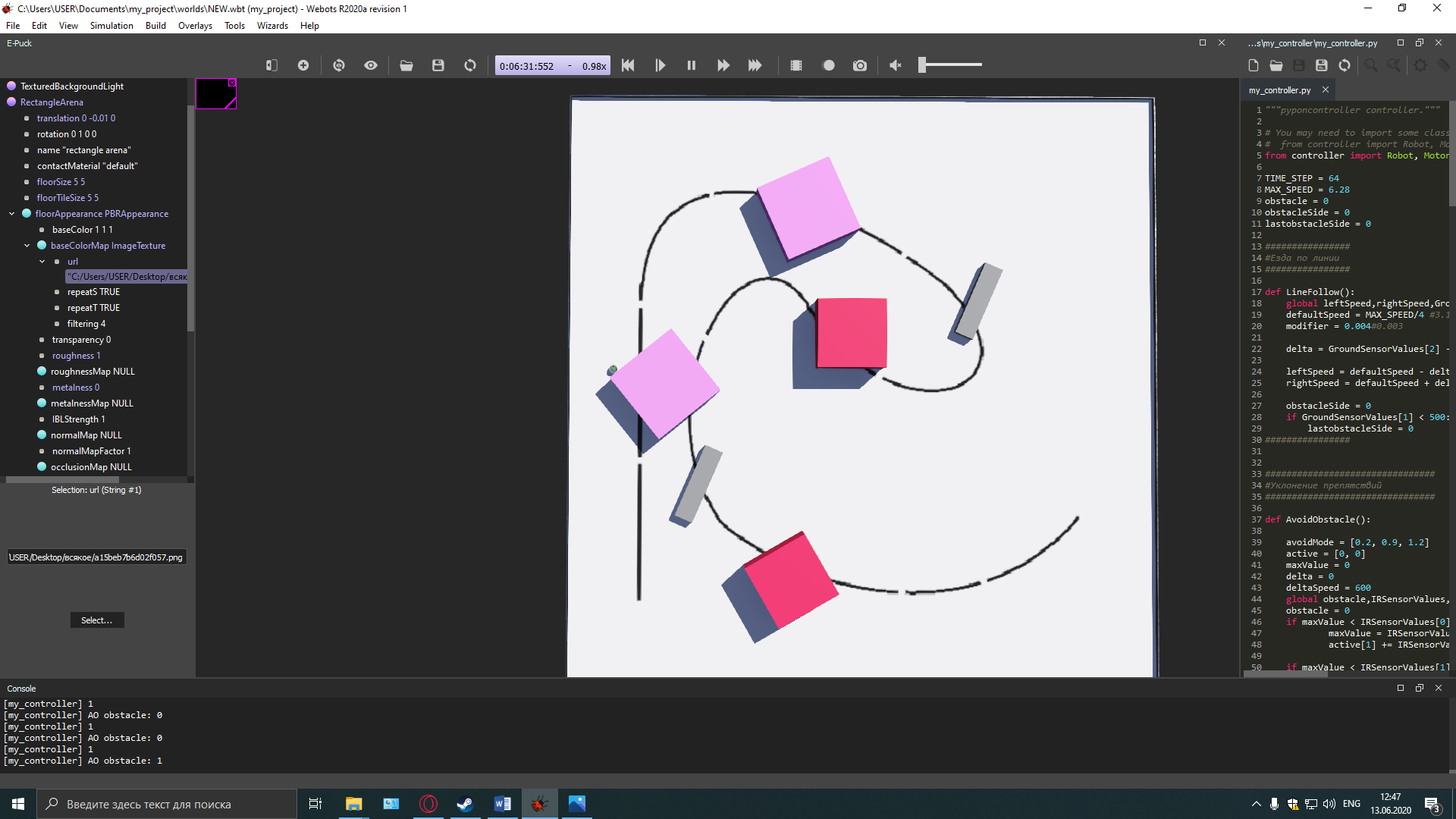
Нижний Новгород

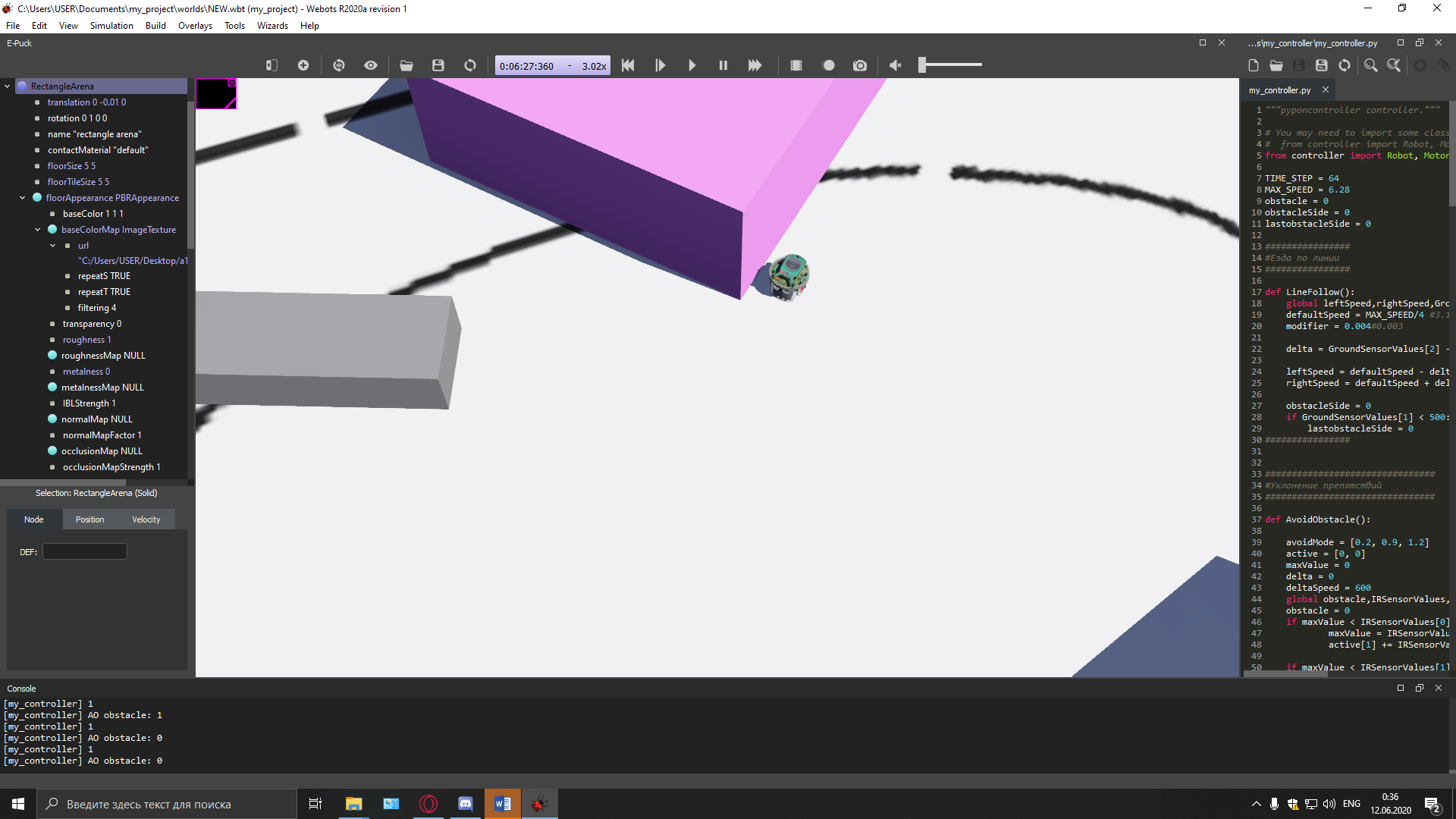
2020

**Цель:** написать программу для робота по вариантам в среде симуляции Webots.

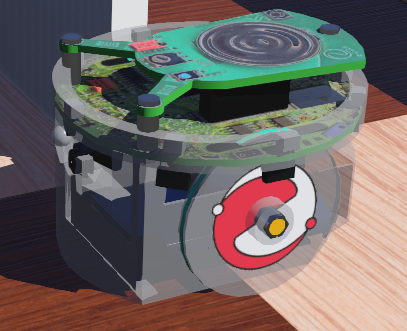
**Вариант:** **обход лабиринта:** робот должен ездить по линии и объезжать препятствия.

**Ход работы:** для начала была создана поле 5х5 в среде Webots и нанесена разметка так же были установлены объекты, преграждающие путь, линия в некоторых местах прерывается.



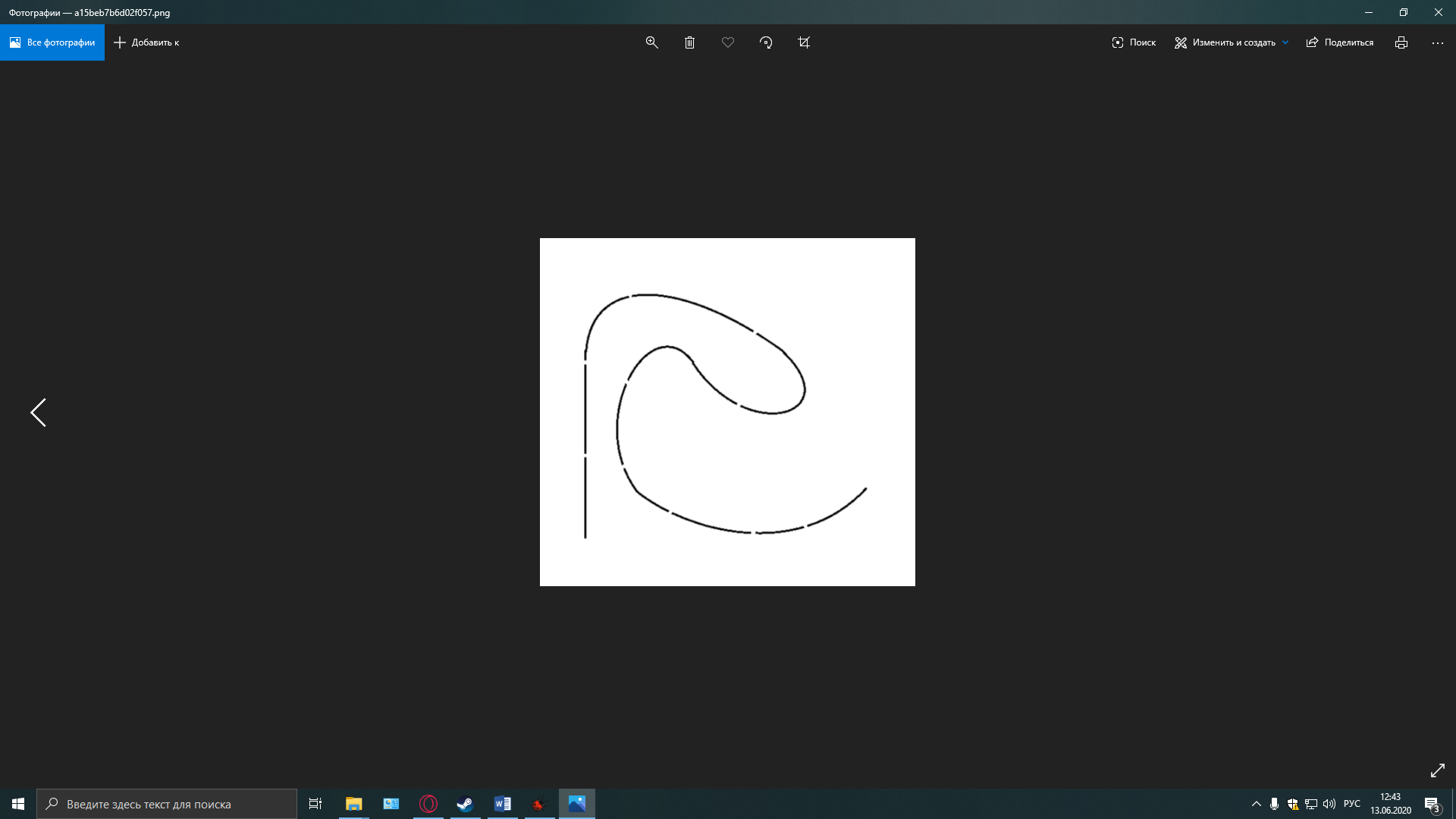


Для выполнения задания был взят робот «e-puck». <https://cyberbotics.com/doc/guide/epuck>

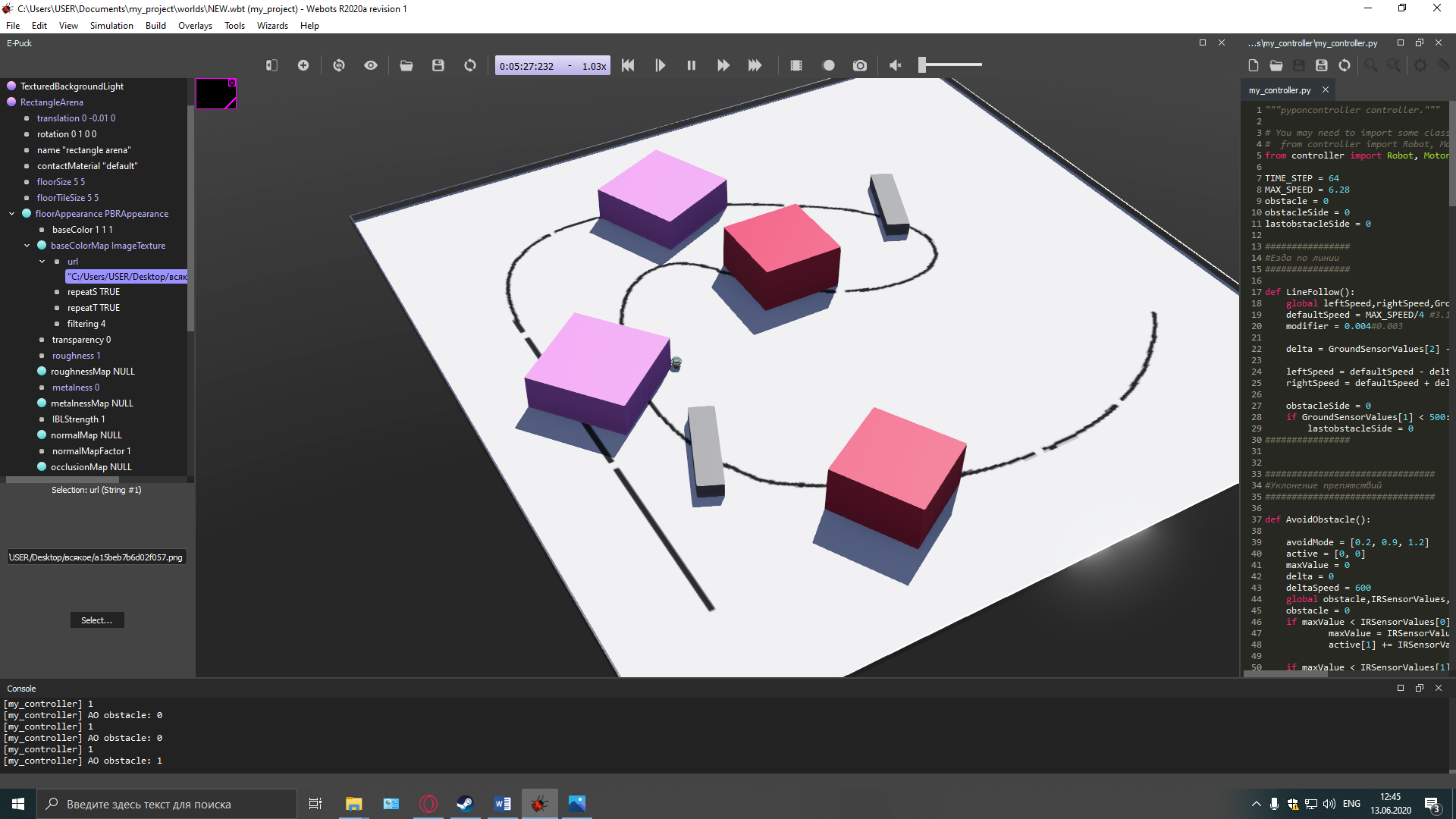


**Добавление линии**

В paint нарисовали трассу с помощью прямых и кривых тонких линий, и тонким ластиком делали дырочки для промежутков



После в Webots мы добавили наш рисунок. Сделали мы это так: зашли в настройки созданной арены (Rectangle Arena) после помели параметры floorsize на 5х5 и также floorTilesize тоже 5х5 для того, чтобы масштаб полученной картинки был 1:1 с ареной. Дальше в floorAppearance мы выбираем PBRAppereance и после в baseColorMap выбираем ImageTexture, где в url вставляем путь к нашей картинке



**Описание алгоритма работы робота:**

Рассматривает 4 передних датчика (2 слева, 2 справа), смотрит на наличие препятствий, суммирует значения 3 передних левых датчика, суммирует значения 3 передних правых датчика и находит их разность на основе этого принимает решение поворота с помощью некоторых коэффициентов и суммирует эту разность с скоростью левого мотора и вычитает эту разность из правого мотора.

**Код контроллера:**

"""pyponcontroller controller."""

# You may need to import some classes of the controller module. Ex:

# from controller import Robot, Motor, DistanceSensor

from controller import Robot, Motor, DistanceSensor

TIME\_STEP = 64

MAX\_SPEED = 6.28

obstacle = 0

obstacleSide = 0

lastobstacleSide = 0

################

#Езда по линии

################

def LineFollow():

global leftSpeed,rightSpeed,GroundSensorValues,obstacleSide,lastobstacleSide

defaultSpeed = MAX\_SPEED/4 #3.14

modifier = 0.004#0.003

delta = GroundSensorValues[2] - GroundSensorValues[0]

leftSpeed = defaultSpeed - delta \* modifier

rightSpeed = defaultSpeed + delta \* modifier

obstacleSide = 0

if GroundSensorValues[1] < 500:

lastobstacleSide = 0

################

################################

#Уклонение препятствий

################################

def AvoidObstacle():

avoidMode = [0.2, 0.9, 1.2]

active = [0, 0]

maxValue = 0

delta = 0

deltaSpeed = 600

global obstacle,IRSensorValues,obstacleSide,leftSpeed,rightSpeed,lastobstacleSide

obstacle = 0

if maxValue < IRSensorValues[0]:

maxValue = IRSensorValues[0]

active[1] += IRSensorValues[0]

if maxValue < IRSensorValues[1]:

maxValue = IRSensorValues[1]

active[1] += IRSensorValues[1]

# if maxValue < IRSensorValues[2]:

# maxValue = IRSensorValues[2]

# active[1] += IRSensorValues[2]

# if maxValue < IRSensorValues[5]:

# maxValue = IRSensorValues[5]

# active[0] += IRSensorValues[5]

if maxValue < IRSensorValues[6]:

maxValue = IRSensorValues[6]

active[0] += IRSensorValues[6]

if maxValue < IRSensorValues[7]:

maxValue = IRSensorValues[7]

active[0] += IRSensorValues[7]

if maxValue > 100:

obstacle = 1

# else:

# obstacle = 0

if obstacleSide == 0 and obstacle == 1:

if active[1] > active[0]:

obstacleSide = 1

lastobstacleSide = 1

else:

lastobstacleSide = -1

print (lastobstacleSide)

if obstacle == 1:

leftSpeed = 6.28 / 8

rightSpeed = 6.28 / 8

if obstacleSide == -1:

for i in range(3):

delta -= avoidMode[i] \* IRSensorValues[i]

else:

for i in range(3):

delta += avoidMode[i] \* IRSensorValues[i + 5]

if delta > deltaSpeed:

delta = deltaSpeed

if delta < -deltaSpeed:

delta = -deltaSpeed

leftSpeed -= delta/100

rightSpeed += delta/100

################################

################################

#ОбЪезд препятствий - подправляет робота в сторону препятствия, компенсируя поворот от уклонения

################################

def ObstacleTrack():

global leftSpeed,rightSpeed, obstacleSide, lastobstacleSide

defaultSpeed = MAX\_SPEED/8

if lastobstacleSide == -1: #препятствие слева

leftSpeed -= defaultSpeed

rightSpeed += defaultSpeed

if lastobstacleSide == 1: #препятствие справа

leftSpeed += defaultSpeed

rightSpeed -= defaultSpeed

################################

################################

#Возвращение на линию

################################

################################

robot = Robot()

# get the time step of the current world.

timestep = int(robot.getBasicTimeStep())

# You should insert a getDevice-like function in order to get the

# instance of a device of the robot. Something like:

# motor = robot.getMotor('motorname')

# ds = robot.getDistanceSensor('dsname')

# ds.enable(timestep)

leftMotor = robot.getMotor('left wheel motor')

rightMotor = robot.getMotor('right wheel motor')

#Инициализация ИК датчиков

IRSensorNames = ['ps0','ps1','ps2','ps3',

'ps4','ps5','ps6','ps7']

IRSensors = []

for sensor in IRSensorNames:

IRSensors.append(robot.getDistanceSensor(sensor))

#----------

#Инициализация датчиков линии

GroundSensorNames = ['gs0','gs1','gs2']

GroundSensors = []

for sensor in GroundSensorNames:

GroundSensors.append(robot.getDistanceSensor(sensor))

#----------

#включаем все сенсоры

for sensor in GroundSensors:

sensor.enable(TIME\_STEP)

for sensor in IRSensors:

sensor.enable(TIME\_STEP)

#----------

#Инициализация моторов

leftMotor.setPosition(float('inf'))

rightMotor.setPosition(float('inf'))

leftMotor.setVelocity(0.0)

rightMotor.setVelocity(0.0)

leftSpeed = 0

rightSpeed = 0

# Main loop:

# - perform simulation steps until Webots is stopping the controller

while robot.step(timestep) != -1:

GroundSensorValues = []

IRSensorValues = []

# Process sensor data here.

for sensor in GroundSensors:

GroundSensorValues.append(sensor.getValue())

for sensor in IRSensors:

IRSensorValues.append(sensor.getValue())

# for sensor in GroundSensorValues:

# print(str(sensor))

LineFollow()

AvoidObstacle()

ObstacleTrack()

# if leftSpeed>6.28: leftSpeed = 6.28

# if rightSpeed>6.28: rightSpeed = 6.28

leftMotor.setVelocity(leftSpeed)

rightMotor.setVelocity(rightSpeed)

# print(str(leftSpeed) + " " + str(rightSpeed))

# Enter here functions to send actuator commands, like:

# motor.setPosition(10.0)

pass

# Enter here exit cleanup code.

**Блок-схема**

ObstacleTrack()

avoidMode = [0.2, 0.9, 1.2]

active = [0, 0]

maxValue = 0

delta = 0

deltaSpeed = 600

global obstacle,IRSensorValues,obstacleSide,leftSpeed,rightSpeed,lastobstacleSide

obstacle = 0

LineFollow()

leftSpeed -= delta/100

rightSpeed += delta/100

ObstacleTrack()

Нет

Нетт

Нет

Нет

Нет

Нет

Нет

Нет

Нет

Нет

Нет

Нет

Нет

Нет

Да

Да

Да

Да

Да

Да

Да

Да

Да

Да

Да

Да

Нет

Да

global leftSpeed,rightSpeed, obstacleSide, lastobstacleSide

defaultSpeed = MAX\_SPEED/8

leftSpeed += defaultSpeed

rightSpeed -= defaultSpeed

lastobstacleSide == -1

leftSpeed -= defaultSpeed

rightSpeed += defaultSpeed

lastobstacleSide == 1

delta += avoidMode[i] \* IRSensorValues[i + 5]

delta += avoidMode[i] \* IRSensorValues[i]

delta = -deltaSpeed

delta < -deltaSpeed

delta = deltaSpeed

delta > deltaSpeed

Конец

Конец

Для i от 1 до 3

Для i от 1 до 3

obstacleSide == -1

obstacle == 1

leftSpeed = 6.28 / 8

rightSpeed = 6.28 / 8

print (lastobstacleSide)

lastobstacleSide = -1

obstacleSide = 1

lastobstacleSide = 1

active[1] > active[0]

obstacleSide == 0 and obstacle == 1

Maxvalue > 100

Obstacle = 1

maxValue < IRSensorValues[7]:

maxValue = IRSensorValues[7]

active[1] += IRSensorValues[7]

maxValue < IRSensorValues[6]:

maxValue = IRSensorValues[6]

active[1] += IRSensorValues[6]

maxValue < IRSensorValues[5]:

maxValue = IRSensorValues[5]

active[1] += IRSensorValues[5-]

maxValue = IRSensorValues[2]

active[1] += IRSensorValues[2]

maxValue < IRSensorValues[2]:

maxValue = IRSensorValues[1]

active[1] += IRSensorValues[1]

maxValue < IRSensorValues[1]:

maxValue = IRSensorValues[0]

active[1] += IRSensorValues[0]

maxValue < IRSensorValues[0]:

Да

Нет

lastobstacleSide = 0

GroundSensorValues[1] < 500

global leftSpeed,rightSpeed,GroundSensorValues,obstacleSide,lastobstacleSide

defaultSpeed = MAX\_SPEED/4 #3.14

modifier = 0.004#0.003

delta = GroundSensorValues[2] - GroundSensorValues[0]

leftSpeed = defaultSpeed - delta \* modifier

rightSpeed = defaultSpeed + delta \* modifier

obstacleSide = 0

Main

Нет

Да

LineFollow()

AvoidObstacle()

ObstacleTrack()

leftMotor.setVelocity(leftSpeed)

rightMotor.setVelocity(rightSpeed)

robot.step(timestep) != -1

GroundSensorValues = []

IRSensorValues = []

Конец

Конец

IRSensorValues.append(sensor.getValue())

Для sensors из IRSensors

GroundSensorValues.append(sensor.getValue())

Для sensors из GroundSensors

leftMotor.setPosition(float('inf'))

rightMotor.setPosition(float('inf'))

leftMotor.setVelocity(0.0)

rightMotor.setVelocity(0.0)

leftSpeed = 0

rightSpeed = 0

robot = Robot()

timestep = int(robot.getBasicTimeStep())

leftMotor = robot.getMotor('left wheel motor')

rightMotor = robot.getMotor('right wheel motor')

IRSensorNames = ['ps0','ps1','ps2','ps3','ps4','ps5','ps6','ps7']

IRSensors = []

Для sensors из IRSensorNames

IRSensors.append(robot.getDistanceSensor(sensor))

Конец

GroundSensorNames = ['gs0','gs1','gs2']

Для sensors из GroundSensorNames

GroundSensors.append(robot.getDistanceSensor(sensor))

Конец

Для sensors из GroundSensors

sensor.enable(TIME\_STEP) 

Конецv

Для sensors из IRSensors

sensor.enable(TIME\_STEP) 

Конецv