**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Львівський національний університет імені Івана Франка Факультет електроніки та комп’ютерних технологій Кафедра**

Допустити до захисту

Завідувач кафедри

доц. Шувар Р. Я.

« » 2022 р.

Кваліфікаційна робота

**Магістр**

(освітній ступінь)

**Дослідження та розробка ботів для автоматизації ігрових процесів під управлінням людини та штучного інтелекту**

Виконав:

студент групи ФЕІм – 22

спеціальності 122 – Комп’ютерні науки

**Товкач Б. М.**

Науковий керівник:

**доц. Демків Л. С.**

**« » 2022 р.**

Рецензент:

**доц. Сидоренко С.С.**

Львів 2022

**АНОТАЦІЯ**

Програми, які допомагають, або надають перевагу, можуть застосовуватися в іграх для оптимізації ігрового процесу. Також, розробка засобів моніторингу та аналізу із подальшою реалізацією у системи управління робототехнікою показує результат на прикладі Boston Dynamics. Програмний бот (робот) - це корисний інструмент, який використовують, зокрема, як умілі гравці, так і ні.

Приклад з грою, всього лиш частковий випадок реалізації деякої ідеї щодо автоматизації процесу, у якому, безпосередньо, рішення приймає людина. У роботі представлена одна з описаного (в курсовій) набору програм (оптимальний варіант, на якому і надалі буде акцентована увага), принцип, якої базується на використанні ШІ для самостійного прийняття рішення. Для розробки програми обрано мову розробки Python, як потужний інструмент та використані технології для роботи із нейронними мережами, обробкою зображень тощо. Розроблено зручний віконний інтерфейс, для моніторингу активності, стану бота та з можливістю передавання керування на людину через голосовий канал.

У розділах описано теоретичні відомості. Розглянуто: методи обробки інформації, описано поняття та значення комп’ютерного зору (комп’ютерне бачення), розпізнавання мовлення людини, розробку програмного забезпечення та автоматизацію процесів. Наведено технології та інструменти для реалізації проєкту. Формується задача та архітектура рішення. Описано: розробка віконного інтерфейсу, способи взаємодії з операційною системою через API, використання бібліотек для роботи з аудіо та відео потоком, їхня обробка та аналіз. Розповідається про реалізацію проєкту, його труднощі розробки та результати. Описано етапи розробки та структура проєкту. Розробка модулів та подальша інтеграція у програму.

В останньому розділі відбувається тестування, аналіз та представлення отриманих результатів, проводиться дослідження.

**ABSTRACT**

Assistive or preferential programs may be used in games to optimize gameplay. Also, the development of monitoring and analysis tools with subsequent implementation in the robotics control system shows the result on the example of Boston Dynamics. A software bot (robot) is a useful tool that is used, in particular, by both skilled players and not.

An example with a game is only a partial case of the implementation of some idea regarding the automation of a process in which, directly, a person makes a decision. The work presents one of the described (in the course) set of programs (the optimal option, which will continue to be the focus of attention), the principle of which is based on the use of AI for independent decision-making. To develop the program, the Python development language was chosen as a powerful tool and used technologies for working with neural networks, image processing, etc. A convenient window interface has been developed for monitoring the activity and status of the bot and with the possibility of transferring control to a person through the voice channel.

Theoretical information is described in the sections. Considered: information processing methods, the concept and meaning of computer vision (computer vision), human speech recognition, software development and process automation are described. Technologies and tools for project implementation are given. The task and solution architecture are being formed. Described: development of a window interface, methods of interaction with the operating system through API, use of libraries for working with audio and video streams, their processing and analysis. It tells about the implementation of the project, its development difficulties and results. The stages of development and the structure of the project are described. Development of modules and further integration into the program.

In the last section, testing, analysis and presentation of the obtained results take place, research is conducted.

**ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ**

ШІ Штучний інтелект  
ПК Персональний комп’ютер  
БД База даних  
ООП Об’єктноорієнтоване програмування  
АП Автоматизація процесу  
ТЗ Технічні засоби

**ЗМІСТ**

[**ВСТУП** 6](#_Toc119422045)

[**РОЗДІЛ I. ОБРОБКА ІНФОРМАЦІЇ. АВТОМАТИЗАЦІЯ ПРОЦЕСІВ. РОЗРОБКА ПРОГРАМНИХ РІШЕНЬ** 7](#_Toc119422046)

[1.1. Методи обробки інформації 7](#_Toc119422047)

[1.2. Комп’ютерний зір 7](#_Toc119422048)

[1.3. Розпізнавання мовлення 7](#_Toc119422049)

[1.4. Задачі автоматизації 7](#_Toc119422050)

[1.5. Комп’ютерна система взаємодій 7](#_Toc119422051)

[1.6. Переваги мови Python 7](#_Toc119422052)

[**РОЗДІЛ II. ТЕХНОЛОГІЇ ПРОГРАМНОЇ РОЗРОБКИ ТА АНАЛІЗУ. СИСТЕМА ПРИЙНЯТТЯ РІШЕНЬ** 8](#_Toc119422053)

[2.1. Постановка задачі та архітектура рішень 8](#_Toc119422054)

[2.2. Розробка віконних програм 8](#_Toc119422055)

[2.3. Програмна взаємодія з операційної системою 8](#_Toc119422056)

[2.4. Бібліотеки для роботи з фото, аудіо матеріалом 8](#_Toc119422057)

[2.5. Бібліотеки для роботи з технологіями розпізнавання 8](#_Toc119422058)

[2.6. Підготовка набору даних та навчання нейронної мережі 8](#_Toc119422059)

[**РОЗДІЛ III. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ. РЕАЛІЗАЦІЯ ПРОЕКТУ** 9](#_Toc119422060)

[3.1. Створення прототипів та тестування 9](#_Toc119422061)

[3.2. Реалізація віконної програми з майбутнім функціоналом 9](#_Toc119422062)

[3.3. Написання модулів для роботи комп’ютерного зору та розпізнавання мовлення 9](#_Toc119422063)

[3.4. Підготовка моделі, тренування нейронної мережі 9](#_Toc119422064)

[3.5. Реалізація програмної взаємодії з грою 9](#_Toc119422065)

[3.6. Інтеграція роботи модулів у програмі та тестування 9](#_Toc119422066)

[**РОЗДІЛ IV. ТЕСТУВАННЯ РОЗРОБКИ. ДОСЛІДЖЕННЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ** 10](#_Toc119422067)

[4.1. Запуск програми в тестовому середовищі 10](#_Toc119422068)

[4.2. Аналіз результатів тестування та корекція програми 10](#_Toc119422069)

[4.3. Збір даних із довготривалого користування в “бойовому” режимі 10](#_Toc119422070)

[4.4. Аналіз результатів дослідження 10](#_Toc119422071)

[**ВИСНОВКИ** 11](#_Toc119422072)

[**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ** 12](#_Toc119422073)

[**ДОДАТОК А** 14](#_Toc119422074)

[**ДОДАТОК Б** 31](#_Toc119422075)

**ВСТУП**

# РОЗДІЛ I. ОБРОБКА ІНФОРМАЦІЇ. АВТОМАТИЗАЦІЯ ПРОЦЕСІВ. РОЗРОБКА ПРОГРАМНИХ РІШЕНЬ

* 1. Методи обробки інформації
  2. Комп’ютерний зір
  3. Розпізнавання мовлення

* 1. Задачі автоматизації

* 1. Комп’ютерна система взаємодій
  2. Переваги мови Python

# РОЗДІЛ II. ТЕХНОЛОГІЇ ПРОГРАМНОЇ РОЗРОБКИ ТА АНАЛІЗУ. СИСТЕМА ПРИЙНЯТТЯ РІШЕНЬ

* 1. Постановка задачі та архітектура рішень
  2. Розробка віконних програм
  3. Програмна взаємодія з операційної системою
  4. Бібліотеки для роботи з фото, аудіо матеріалом
  5. Бібліотеки для роботи з технологіями розпізнавання
  6. Підготовка набору даних та навчання нейронної мережі

# РОЗДІЛ III. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ. РЕАЛІЗАЦІЯ ПРОЕКТУ

* 1. Створення прототипів та тестування
  2. Реалізація віконної програми з майбутнім функціоналом
  3. Написання модулів для роботи комп’ютерного зору та розпізнавання мовлення
  4. Підготовка моделі, тренування нейронної мережі

* 1. Реалізація програмної взаємодії з грою
  2. Інтеграція роботи модулів у програмі та тестування

# РОЗДІЛ IV. ТЕСТУВАННЯ РОЗРОБКИ. ДОСЛІДЖЕННЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ

* 1. Запуск програми в тестовому середовищі
  2. Аналіз результатів тестування та корекція програми
  3. Збір даних із довготривалого користування в “бойовому” режимі
  4. Аналіз результатів дослідження

# ВИСНОВКИ

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Detection Classes [Електронний ресурс] / Режим доступу: www/URL:https://imageai.readthedocs.io/en/latest/ – 09.05.2019 р.
2. Prediction Classes [Електронний ресурс] / Режим доступу: www/URL: https://imageai.readthedocs.io/en/latest/prediction/index.html - 09.05.2019 р.
3. Video and Live-Feed Detection and Analysis [Електронний ресурс] / Режим доступу URL: https://imageai.readthedocs.io/en/latest/video/index.html – 09.05.2019р.
4. Python [Електронний ресурс] / Режим доступу: www/URL: https://www.python.org/ - 09.05.2019 р.
5. Prediction Classes [Електронний ресурс] / Режим доступу: www/URL: https://imageai.readthedocs.io/en/latest/prediction/index.html - 09.05.2019 р.
6. Kivy [Електронний ресурс] / Режим доступу URL: https://kivy.org/#home – 09.05.2019р.
7. Рихтер Д. Head First / Рихтер Д. – Вашингтон: 2017. – 816с.
8. Fluent Python / Лучано Рамалью. – Вашингтон: 2015. - 792с.
9. Р. Динеш. Всі патерни проектування, 2019. – 320 с.
10. Фрейдзон И.Р. Автоматизовані системи / И.Р. Фрейдзон. – Л.: Судостроение, 1988. – 365 с.
11. Крисилов В.А. представлення вихідних даних в задачах нейронного програмування / Одесса: ОНПУ. 2003. С. 7.
12. Леван Д.Н., Феоктистов Н.А. Особливості використання многослойного парцептрона/ Науковведення. віп. 2. 2014. С. 8.
13. Шитиков В.К., Розенберг Г.С., Зинченко Т.Д. Методи системної ідентифікації / Тольятти: ИЭВБ РАН. 2003. 463 с.
14. Шахнов В.А., Власов А.И., Кузнецов А.С. Нейрокомпютери: схемотехніка / М.: Изд-во Машинобудування . 2000. 64 с.
15. McMillan C. The Connectionist Scientist Game: Rule Extraction and Refinement in a Neural Network / C. McMillan, M.C. Mozer, P. Smolensky // Proc. XIII Annual Conf of the Cognitive Science Society, Hillsdale, NJ, USA. – 2001
16. Область застосування штучних нейронних мереж [Електроний ресурс//Основні напрямки використання / URL: http://www.neuropro.ru/ ( дата звернення 20.11.2020).
17. Robert M.HaralickLinda G.Shapiro; Image segmentation techniques [Електронний ресурс] — Режим доступу: https://www.sciencedirect.com/science/ article/ pii/S0734189X85901537.
18. Dilpreet Kaur, Yadwinder Kaur; Various Image Segmentation.Techniques: A Review// IJCSMC, Vol. 3, Issue. 5, May 2014, pg.809 – 814. ISSN 2320–088X.
19. Su Hnin Hlaing, Aung Soe Khaing; Weed and crop segmentation and classification using area thresholding //International Journal of Research in Engineering and Technology eISSN: 2319-1163 pISSN: 2321-7308.
20. Sebastian Haug, Andreas Michaels; Plant classification system for crop /weed discrimination without segmentation// Published in IEEE Winter Conference 2014. DOI:10.1109/WACV.2014.6835733.
21. . Philipp Lottes, Cyrill Stachniss; Semi-Supervised Online Visual Crop and Weed Classification in Precision Farming Exploiting Plant Arrangement. [Електронний ресурс] — Режим доступу: http://flourishproject.eu/fileadmin/user\_upload/publications/lottes17iros.pdf.
22. Yao Wang ; Yisong Chen ; Peng Lu ; Heng Wang Sobel Heuristic Kernel for Aerial Semantic Segmentation// 2018 25th IEEE International Conference on Image Processing (ICIP), Electronic ISSN: 2381-8549.
23. The 2016 Sugar Beets Dataset Recorded at Campus Klein Altendorf in Bonn,Germany [Електронний ресурс] — Режим доступу: http://www.ipb.unibonn.de/data /sugabeets2016.
24. Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel: Backpropagation Applied to Handwritten Zip Code Recognition, Neural Computation, 1(4):541-551, Winter 1989.
25. Le Hou ; Dimitris Samaras ; Tahsin M. Kurc ; Patch-Based Convolutional Neural Network for Whole Slide Tissue Image Classification //2016 IEEE Conference on Computer Vision and Pattern Recognition, ISSN: 1063-6919.
26. Fully Convolutional Networks for Semantic Segmentation [Електронний ресурс] — Режим доступу: https://people.eecs.berkeley.edu/~jonlong/long\_shelhamer\_fcn.pdf.

# ДОДАТОК А

**main.py**

import PySimpleGUI as sg

import tensorflow as tf

import os

import random

from imageai.Detection.Custom import CustomObjectDetection

from wincapture import WindowCapture

from components import layout\_all

# from speech import recognition\_speech

from datetime import datetime

from dashboard import \*

from utils import (

resize\_image,

init\_config,

output\_stream,

get\_titles,

get\_file\_path,

get\_perm

)

os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '3'

os.environ['TF\_XLA\_FLAGS'] = '--tf\_xla\_enable\_xla\_devices'

tf.config.experimental.enable\_mlir\_graph\_optimization()

tf.config.run\_functions\_eagerly(True)

def start\_event(window):

global STREAMING, DETECTING, CHECKED, DETECTOR\_READY

STREAMING = True

DETECTING = not DETECTING

CHECKED = not CHECKED

def pause\_event(window):

global STREAMING, DETECTING, DETECTOR\_READY

STREAMING = not STREAMING

DETECTING = not DETECTING

def targetWindow\_value(window):

try:

target\_folder = CONFIG['TARGET\_WINDOW']

window['-TARGET WINDOW-'].update(target\_folder)

except:

pass

def targetWindow\_event(window, values):

global CONFIG

window\_title = values['-WINDOW TITLES-']

CONFIG['TARGET\_WINDOW'] = window\_title

window['-TARGET WINDOW-'].update(window\_title)

window['-WINDOW TITLES-'].update(['Unused', get\_titles()])

def targetFolder\_value(window):

try:

target\_folder = CONFIG['TARGET\_FOLDER']

window['-TARGET FOLDER-'].update(target\_folder)

except:

pass

def targetFolder\_event(values):

global CONFIG

CONFIG['TARGET\_FOLDER'] = values['-TARGET FOLDER-']

def actions\_value(window):

try:

actions = CONFIG['ACTIONS\_LOGS']

window['-ACTIONS LOGS-'].update(actions)

except:

pass

def speech\_value(window):

try:

speech = CONFIG['SPEECH\_LOGS']

window['-SPEECH LOGS-'].update(speech)

except:

pass

def activate\_game\_bot(values):

if 'R2' not in values['-TARGET WINDOW-'] or '??' not in values['-TARGET WINDOW-']:

sg.Popup('WARNING!\ngame is not ready',

title='Activate ERROR')

def streaming\_event(window, values, detector):

global DETECTIONS

try:

wincap = WindowCapture(values['-TARGET WINDOW-'])

stream = wincap.get\_screenshot()

except:

return

DETECTIONS, stream\_data = output\_stream(

stream, CONFIG['WINDOW\_SIZE'], detector, detecting=DETECTING)

window['-VIDEO STREAM-'].update(data=stream\_data)

def actions\_logs\_event(window, values):

global DETECTOR\_COUNT

if DETECTING and DETECTIONS is not None:

from datetime import datetime

now = datetime.now()

time = now.strftime("%H:%M:%S")

if DETECTIONS != []:

DETECTOR\_COUNT += 1

window["-ACTIONS LOGS-"].update('\n\n', append=True)

window["-ACTIONS LOGS-"].update('Event ' + str(DETECTOR\_COUNT), text\_color\_for\_value='white',

background\_color\_for\_value='blue', append=True)

window["-ACTIONS LOGS-"].update('-'\*18, append=True)

window["-ACTIONS LOGS-"].update(

'--==DETECTED==--', text\_color\_for\_value='yellow', background\_color\_for\_value='black', append=True)

window["-ACTIONS LOGS-"].update('-'\*18, append=True)

window["-ACTIONS LOGS-"].update(

'[' + time + ']', text\_color\_for\_value='white', background\_color\_for\_value='blue', append=True)

for idx, detect in enumerate(DETECTIONS):

window["-ACTIONS LOGS-"].update('\n\n', append=True)

window["-ACTIONS LOGS-"].update(f'{idx+1}. ', text\_color\_for\_value='white',

background\_color\_for\_value='black', append=True)

window["-ACTIONS LOGS-"].update('MONSTER => ' + detect['name'],

text\_color\_for\_value='black', background\_color\_for\_value='yellow', append=True)

window["-ACTIONS LOGS-"].update('\t', append=True)

window["-ACTIONS LOGS-"].update('PROBABILITY => ' + str(round(detect['percentage\_probability'], 3)),

text\_color\_for\_value='black', background\_color\_for\_value='yellow', append=True)

window["-ACTIONS LOGS-"].update('\n', append=True)

window["-ACTIONS LOGS-"].update('POSITION => x = ' + str(detect['box\_points'][0]) + ' y = ' + str(detect['box\_points'][1]),

text\_color\_for\_value='black', background\_color\_for\_value='yellow', append=True)

def speech\_window():

window = sg.Window('Speech Recognition', layout=[

[sg.Button('Hello World')]])

while True:

event, values = window.read()

if event == sg.WIN\_CLOSED:

break

window.close()

def speeching\_logs\_event(window, values):

global DETECTOR\_COUNT

if DETECTING and DETECTIONS is not None:

now = datetime.now()

time = now.strftime("%H:%M:%S")

commands = random.choices(

SLAVE\_MESSAGE['commands'], weights=SLAVE\_MESSAGE['weights'], k=random.randint(1, 3))

# time = '-=== DETECTED ===-'

if DETECTIONS != []:

for command in commands:

window["-SPEECH LOGS-"].update('\n', append=True)

window["-SPEECH LOGS-"].update(command,

text\_color\_for\_value='black', background\_color\_for\_value='yellow', append=True)

def clear\_event(window):

global CONFIG, FIRST\_LOAD

FIRST\_LOAD = False

window['-START-'].update(disabled=True)

window['-TARGET FOLDER-'].update('')

window['-TARGET WINDOW-'].update('')

window['-VIDEO STREAM-'].update(

data=resize\_image(CONFIG['DISCONNECT\_INFO'], CONFIG['WINDOW\_SIZE']))

window['-ACTIONS LOGS-'].update('')

window['-SPEECH LOGS-'].update('')

def program\_close():

global CONFIG

CONFIG['TARGET\_WINDOW'] = VALUES['-TARGET WINDOW-']

CONFIG['TARGET\_FOLDER'] = VALUES['-TARGET FOLDER-']

def get\_window\_settings():

layout\_props = [

get\_titles(),

CONFIG['DISCONNECT\_INFO'],

CONFIG['WINDOW\_SIZE']

]

main\_settings = dict(

layout=layout\_all(\*layout\_props),

size=(1120, 630)

)

icon\_path = get\_file\_path(CONFIG['WINDOW\_ICON'])

if icon\_path is not None:

main\_settings.update(icon=icon\_path)

return main\_settings

def save\_actions\_logs(values):

with open('actions\_logs.txt', 'w') as file:

file.write(values['-ACTIONS LOGS-'])

def check\_event(window, values):

global DETECTING, CHECKED

target\_folder = values['-TARGET FOLDER-']

perm\_file = CONFIG['PERMISSION\_FILE']

if not get\_perm(perm\_file, target\_folder, window, values):

sg.Popup('WARNING!\n"target" or "window" is not ready',

title='CHECK ERROR')

else:

CHECKED = True

def init\_detector(folder\_path):

model\_path = os.path.join(

folder\_path, "models/detection\_model-ex-049--loss-0012.515.h5")

json\_path = os.path.join(folder\_path, "json/detection\_config.json")

detector = CustomObjectDetection()

detector.setModelTypeAsYOLOv3()

detector.setModelPath(model\_path)

detector.setJsonPath(json\_path)

detector.loadModel()

return detector

def psutility(window, net\_graph\_in, net\_graph\_out, disk\_graph\_read, gpu\_usage\_graph, cpu\_usage\_graph, mem\_usage\_graph):

netio = psutil.net\_io\_counters()

write\_bytes = net\_graph\_out.graph\_value(netio.bytes\_sent)

read\_bytes = net\_graph\_in.graph\_value(netio.bytes\_recv)

window['\_NET\_OUT\_TXT\_'].update(

'Net out {}'.format(human\_size(write\_bytes)))

window['\_NET\_IN\_TXT\_'].update(

'Net In {}'.format(human\_size(read\_bytes)))

# ----- Disk Graphs -----

diskio = psutil.disk\_io\_counters()

read\_bytes = disk\_graph\_read.graph\_value(diskio.read\_bytes)

window['\_DISK\_READ\_TXT\_'].update(

'Disk Read {}'.format(human\_size(read\_bytes)))

# ----- GPU Graph -----

gpu = GPUtil.getGPUs()[-1].load\*100

gpu\_usage\_graph.graph\_percentage\_abs(gpu)

window['\_GPU\_TXT\_'].update('{0:2.0f}% GPU Used'.format(gpu))

# ----- CPU Graph -----

cpu = psutil.cpu\_percent(0)

cpu\_usage\_graph.graph\_percentage\_abs(cpu)

window['\_CPU\_TXT\_'].update('{0:2.0f}% CPU Used'.format(cpu))

# ----- Memory Graph -----

mem\_used = psutil.virtual\_memory().percent

mem\_usage\_graph.graph\_percentage\_abs(mem\_used)

window['\_MEM\_TXT\_'].update('{}% Memory Used'.format(mem\_used))

CONFIG = init\_config()

STREAMING = True

DETECTING = False

FIRST\_LOAD = True

VALUES = None

CHECKED = False

DETECTIONS = None

DETECTOR\_COUNT = 0

DETECTOR\_READY = True

SLAVE\_MESSAGE = {

'commands': [

'[ACTION] \tpress',

'[ACTION] \tloot',

'[MOVE] \tv',

'[MOVE] \t^',

'[MOVE] \t<-',

'[MOVE] \t->',

'[TARGET] \tvampire #1',

'[TARGET] \tvampire #2',

'[TARGET] \tvampire #3'

],

'weights': [2, 2, 4, 3, 4, 3, 1, 1, 1]

}

def main():

global VALUES, DETECTOR\_READY

detector = None

main\_settings = get\_window\_settings()

window = sg.Window('AI Bot', \*\*main\_settings)

timeout = int(1000/CONFIG['FPS'])

netio = psutil.net\_io\_counters()

net\_in = window['\_NET\_IN\_GRAPH\_']

net\_graph\_in = DashGraph(net\_in, netio.bytes\_recv, '#23a0a0')

net\_out = window['\_NET\_OUT\_GRAPH\_']

net\_graph\_out = DashGraph(net\_out, netio.bytes\_sent, '#56d856')

diskio = psutil.disk\_io\_counters()

disk\_graph\_read = DashGraph(

window['\_DISK\_READ\_GRAPH\_'], diskio.read\_bytes, '#5681d8')

gpu\_usage\_graph = DashGraph(window['\_GPU\_GRAPH\_'], 0, '#d34545')

cpu\_usage\_graph = DashGraph(window['\_CPU\_GRAPH\_'], 0, '#d34545')

mem\_usage\_graph = DashGraph(window['\_MEM\_GRAPH\_'], 0, '#BE7C29')

while True:

event, values = window.read(timeout=timeout)

if event == sg.WIN\_CLOSED:

program\_close()

init\_config(conf\_file=CONFIG)

break

if FIRST\_LOAD:

if values['-TARGET FOLDER-'] == '':

targetFolder\_value(window)

if values['-TARGET WINDOW-'] == '':

targetWindow\_value(window)

if values['-ACTIONS LOGS-'] == '':

actions\_value(window)

if values['-SPEECH LOGS-'] == '':

speech\_value(window)

if values['-TARGET WINDOW-'] != '':

window['-PAUSE-'].update(disabled=False)

else:

window['-PAUSE-'].update(disabled=True)

if event == '-TARGET FOLDER-':

targetFolder\_event(values)

if event == '-WINDOW TITLES-':

targetWindow\_event(window, values)

if event == '-START-':

start\_event(window)

if event == '-PAUSE-':

pause\_event(window)

if CHECKED and DETECTOR\_READY:

detector = init\_detector(values['-TARGET FOLDER-'])

DETECTOR\_READY = False

if STREAMING:

streaming\_event(window, values, detector)

actions\_logs\_event(window, values)

speeching\_logs\_event(window, values)

if event == '-CLEAR-':

clear\_event(window)

if event == '-CHECK-':

check\_event(window, values)

if event == '-ACTIVATE-':

activate\_game\_bot(values)

if event == '-SPEECH-':

speech\_window()

if event == 'Save':

save\_actions\_logs(values)

psutility(window, net\_graph\_in, net\_graph\_out, disk\_graph\_read,

gpu\_usage\_graph, cpu\_usage\_graph, mem\_usage\_graph)

VALUES = {\*\*values}

window.close()

if \_\_name\_\_ == '\_\_main\_\_':

**components.py**

from utils import resize\_image, get\_file\_path, create\_image

import PySimpleGUI as sg

import os

def get\_video\_banner(image\_path, image\_size):

sources = dict()

path = get\_file\_path(image\_path)

if os.path.exists(path):

sources.update(source=resize\_image(image\_path, image\_size))

else:

sources.update(source=create\_image(image\_size))

return sources

def layout\_all(windows\_list, image\_path, image\_size):

window\_menu = ['Unused', windows\_list]

header\_column\_left = [

[

sg.FolderBrowse(button\_text='Target',

target='-TARGET FOLDER-', size=(7, 1), enable\_events=True),

sg.Input(size=(70, 1), enable\_events=True,

readonly=True, key='-TARGET FOLDER-'),

sg.ButtonMenu('Window', window\_menu, size=(

7, 1), key='-WINDOW TITLES-'),

sg.Input(size=(30, 1), enable\_events=True,

readonly=True, key='-TARGET WINDOW-')

]

]

header\_column\_right = [

[

sg.Button(button\_text='START', enable\_events=True,

key='-START-', disabled=True),

sg.Button(button\_text='PAUSE', enable\_events=True, key='-PAUSE-'),

sg.Button(button\_text='CLEAR', enable\_events=True, key='-CLEAR-')

]

]

header\_layout = [

[

sg.Column(header\_column\_left,

element\_justification='left', expand\_x=True),

sg.Column(header\_column\_right,

element\_justification='right', expand\_x=True)

]

]

center\_column\_left = [

[

sg.Image(\*\*get\_video\_banner(image\_path, image\_size), size=image\_size, enable\_events=True,

key='-VIDEO STREAM-')

]

]

multiline\_menu = ['', ['Save']]

center\_column\_right = [

[

sg.Multiline(size=(55, 22), key='-ACTIONS LOGS-', autoscroll=True, disabled=True,

enable\_events=True, right\_click\_menu=multiline\_menu)

]

]

center\_layout = [

[

sg.Column(center\_column\_left,

element\_justification='left'),

sg.Column(center\_column\_right,

element\_justification='right', expand\_x=True)

]

]

GRAPH\_WIDTH, GRAPH\_HEIGHT = 120, 40

def GraphColumn(name, key):

layout = [

[sg.Text(name, size=(18, 1), font=('Helvetica 8'), key=key+'TXT\_')],

[sg.Graph((GRAPH\_WIDTH, GRAPH\_HEIGHT),

(0, 0),

(GRAPH\_WIDTH, 100),

background\_color='black',

key=key+'GRAPH\_')]]

return sg.Col(layout, pad=(2, 2))

dash\_layout = [

[GraphColumn('Net Out', '\_NET\_OUT\_'),

GraphColumn('Net In', '\_NET\_IN\_'),

GraphColumn('Disk Read', '\_DISK\_READ\_')],

[GraphColumn('GPU Usage', '\_GPU\_'),

GraphColumn('CPU Usage', '\_CPU\_'),

GraphColumn('Memory Usage', '\_MEM\_')],

]

footer\_column\_left = [

[

sg.Column(dash\_layout),

sg.Column([

[

sg.Button(button\_text='CHECK SETTINGS',

enable\_events=True, key='-CHECK-', expand\_x=True),

sg.Button(button\_text='ACTIVATE BOT',

key='-ACTIVATE-', expand\_x=True)

],

[

sg.Button(button\_text='SPEECH INTERVENE',

key='-SPEECH-', expand\_x=True)

]

])

]

]

footer\_column\_right = [

[

sg.Column([

[sg.Multiline(size=(55, 18), key='-SPEECH LOGS-', autoscroll=True, disabled=True,

enable\_events=True)]

])

]

]

footer\_layout = [

[

sg.Column(footer\_column\_left, element\_justification='left',

expand\_x=True, expand\_y=True),

sg.Column(footer\_column\_right,

element\_justification='right', expand\_x=True)

]

]

layout = [

[sg.Frame(layout=header\_layout, title='',

element\_justification='center', expand\_x=True)],

[sg.Frame(layout=center\_layout, title='',

element\_justification='center', expand\_x=True)],

[sg.Frame(layout=footer\_layout, title='',

element\_justification='center', expand\_x=True)]

]

return layout

**speech.py**

import random

import time

import speech\_recognition as sr

def recognize\_speech\_from\_mic(recognizer, microphone):

"""Transcribe speech from recorded from `microphone`.

Returns a dictionary with three keys:

"success": a boolean indicating whether or not the API request was

successful

"error": `None` if no error occured, otherwise a string containing

an error message if the API could not be reached or

speech was unrecognizable

"transcription": `None` if speech could not be transcribed,

otherwise a string containing the transcribed text

"""

# check that recognizer and microphone arguments are appropriate type

if not isinstance(recognizer, sr.Recognizer):

raise TypeError("`recognizer` must be `Recognizer` instance")

if not isinstance(microphone, sr.Microphone):

raise TypeError("`microphone` must be `Microphone` instance")

# adjust the recognizer sensitivity to ambient noise and record audio

# from the microphone

with microphone as source:

recognizer.adjust\_for\_ambient\_noise(source)

audio = recognizer.listen(source)

# set up the response object

response = {

"success": True,

"error": None,

"transcription": None

}

# try recognizing the speech in the recording

# if a RequestError or UnknownValueError exception is caught,

# update the response object accordingly

try:

response["transcription"] = recognizer.recognize\_google(audio)

except sr.RequestError:

# API was unreachable or unresponsive

response["success"] = False

response["error"] = "API unavailable"

except sr.UnknownValueError:

# speech was unintelligible

response["error"] = "Unable to recognize speech"

return response

def recognition\_speech():

# set the list of words, maxnumber of guesses, and prompt limit

WORDS = ["apple", "banana", "grape", "orange", "mango", "lemon"]

NUM\_GUESSES = 3

PROMPT\_LIMIT = 5

# create recognizer and mic instances

recognizer = sr.Recognizer()

microphone = sr.Microphone()

# get a random word from the list

word = random.choice(WORDS)

# format the instructions string

instructions = (

"I'm thinking of one of these words:\n"

"{words}\n"

"You have {n} tries to guess which one.\n"

).format(words=', '.join(WORDS), n=NUM\_GUESSES)

# show instructions and wait 3 seconds before starting the game

print(instructions)

time.sleep(3)

for i in range(NUM\_GUESSES):

# get the guess from the user

# if a transcription is returned, break out of the loop and

# continue

# if no transcription returned and API request failed, break

# loop and continue

# if API request succeeded but no transcription was returned,

# re-prompt the user to say their guess again. Do this up

# to PROMPT\_LIMIT times

for j in range(PROMPT\_LIMIT):

print('Guess {}. Speak!'.format(i+1))

guess = recognize\_speech\_from\_mic(recognizer, microphone)

if guess["transcription"]:

break

if not guess["success"]:

break

print("I didn't catch that. What did you say?\n")

# if there was an error, stop the game

if guess["error"]:

print("ERROR: {}".format(guess["error"]))

break

# show the user the transcription

print("You said: {}".format(guess["transcription"]))

# determine if guess is correct and if any attempts remain

guess\_is\_correct = guess["transcription"].lower() == word.lower()

user\_has\_more\_attempts = i < NUM\_GUESSES - 1

# determine if the user has won the game

# if not, repeat the loop if user has more attempts

# if no attempts left, the user loses the game

if guess\_is\_correct:

print("Correct! You win!".format(word))

break

elif user\_has\_more\_attempts:

print("Incorrect. Try again.\n")

else:

print("Sorry, you lose!\nI was thinking of '{}'.".format(word))

break

if \_\_name\_\_ == "\_\_main\_\_":

recognition\_speech()

**wincapture.py**

import numpy as np

import win32gui

import win32ui

import win32con

class WindowCapture:

# properties

w = 0

h = 0

hwnd = None

cropped\_x = 0

cropped\_y = 0

offset\_x = 0

offset\_y = 0

# constructor

def \_\_init\_\_(self, window\_name):

# find the handle for the window we want to capture

self.hwnd = win32gui.FindWindow(None, window\_name)

if not self.hwnd:

raise Exception('Window not found: {}'.format(window\_name))

# get the window size

window\_rect = win32gui.GetWindowRect(self.hwnd)

self.w = window\_rect[2] - window\_rect[0]

self.h = window\_rect[3] - window\_rect[1]

# account for the window border and titlebar and cut them off

border\_pixels = 8

titlebar\_pixels = 30

self.w = self.w - (border\_pixels \* 2)

self.h = self.h - titlebar\_pixels - border\_pixels

self.cropped\_x = border\_pixels

self.cropped\_y = titlebar\_pixels

# set the cropped coordinates offset so we can translate screenshot

# images into actual screen positions

self.offset\_x = window\_rect[0] + self.cropped\_x

self.offset\_y = window\_rect[1] + self.cropped\_y

def get\_screenshot(self):

# get the window image data

wDC = win32gui.GetWindowDC(self.hwnd)

dcObj = win32ui.CreateDCFromHandle(wDC)

cDC = dcObj.CreateCompatibleDC()

dataBitMap = win32ui.CreateBitmap()

dataBitMap.CreateCompatibleBitmap(dcObj, self.w, self.h)

cDC.SelectObject(dataBitMap)

cDC.BitBlt((0, 0), (self.w, self.h), dcObj,

(self.cropped\_x, self.cropped\_y), win32con.SRCCOPY)

# convert the raw data into a format opencv can read

#dataBitMap.SaveBitmapFile(cDC, 'debug.bmp')

signedIntsArray = dataBitMap.GetBitmapBits(True)

img = np.fromstring(signedIntsArray, dtype='uint8')

img.shape = (self.h, self.w, 4)

# free resources

dcObj.DeleteDC()

cDC.DeleteDC()

win32gui.ReleaseDC(self.hwnd, wDC)

win32gui.DeleteObject(dataBitMap.GetHandle())

# drop the alpha channel, or cv.matchTemplate() will throw an error like:

# error: (-215:Assertion failed) (depth == CV\_8U || depth == CV\_32F) && type == \_templ.type()

# && \_img.dims() <= 2 in function 'cv::matchTemplate'

img = img[..., :3]

# make image C\_CONTIGUOUS to avoid errors that look like:

# File ... in draw\_rectangles

# TypeError: an integer is required (got type tuple)

# see the discussion here:

# https://github.com/opencv/opencv/issues/14866#issuecomment-580207109

img = np.ascontiguousarray(img)

return img

# find the name of the window you're interested in.

# once you have it, update window\_capture()

# https://stackoverflow.com/questions/55547940/how-to-get-a-list-of-the-name-of-every-open-window

def list\_window\_names(self):

def winEnumHandler(hwnd, ctx):

if win32gui.IsWindowVisible(hwnd):

print(hex(hwnd), win32gui.GetWindowText(hwnd))

win32gui.EnumWindows(winEnumHandler, None)

# translate a pixel position on a screenshot image to a pixel position on the screen.

# pos = (x, y)

# WARNING: if you move the window being captured after execution is started, this will

# return incorrect coordinates, because the window position is only calculated in

# the \_\_init\_\_ constructor.

def get\_screen\_position(self, pos):

return (pos[0] + self.offset\_x, pos[1] + self.offset\_y)

**utils.py**

from io import BytesIO

from PIL import Image

import cv2

import os

import base64

import json

import win32gui

import sys

DEFAULT\_CONFIG = {

"FPS": 20,

"WINDOW\_SIZE": [

640,

360

],

"DISCONNECT\_INFO": "disconnect\_info.png",

"WINDOW\_ICON": "window\_icon.ico"

}

def get\_file\_path(filename):

bundle\_dir = getattr(

sys, '\_MEIPASS', os.path.abspath(os.path.dirname(\_\_file\_\_)))

path\_to\_file = os.path.abspath(

os.path.join(bundle\_dir, 'static/' + filename))

return path\_to\_file

def init\_config(conf\_file=None, file\_path='config.json'):

file\_path = get\_file\_path(file\_path)

if conf\_file is None:

mode = 'r'

else:

mode = 'w'

file\_creating = True

while True:

if os.path.exists(file\_path):

with open(file\_path, mode) as file:

if mode == 'r':

conf\_file = json.load(file)

return conf\_file

json.dump(conf\_file, file, indent=4)

break

if file\_creating:

with open(file\_path, 'w') as file:

json.dump(DEFAULT\_CONFIG, file, indent=4)

file\_creating = False

def get\_perm(filename, directory, window, values):

if filename != '' and directory != '':

if filename in os.listdir(directory):

abs\_path = os.path.join(directory, filename)

with open(abs\_path, 'r') as file:

ready\_check = json.load(file)

if ready\_check['READY'] and values['-TARGET WINDOW-'] == ready\_check['WINDOW']:

window['-START-'].update(disabled=False)

return True

window['-START-'].update(disabled=True)

return False

def get\_titles():

WINDOW\_LIST = []

def winEnumHandler(hwnd, ctx):

nonlocal WINDOW\_LIST

if win32gui.IsWindowVisible(hwnd):

window\_title = win32gui.GetWindowText(hwnd)

if window\_title != '':

WINDOW\_LIST.append(window\_title)

win32gui.EnumWindows(winEnumHandler, None)

return WINDOW\_LIST

def create\_image(image\_size):

img = Image.new('RGB', image\_size, color='black')

buffered = BytesIO()

img.save(buffered, format="PNG")

img\_str = base64.b64encode(buffered.getvalue())

return img\_str

def resize\_image(filename, image\_size):

abs\_path = get\_file\_path(filename)

img = Image.open(abs\_path)

resized\_img = img.resize(image\_size)

buffered = BytesIO()

resized\_img.save(buffered, format="PNG")

img\_str = base64.b64encode(buffered.getvalue())

return img\_str

def label\_detecting(stream, detector):

detections = detector.detectObjectsFromImage(

input\_image=stream,

input\_type='array',

output\_type='array',

minimum\_percentage\_probability=40,

extract\_detected\_objects=True,

thread\_safe=False

)

return detections[0], detections[1]

def output\_stream(stream, image\_size, detector, detecting=True):

detections = None

if detecting:

stream, detections = label\_detecting(stream, detector)

resized = cv2.resize(

stream, image\_size, interpolation=cv2.INTER\_AREA)

imgbytes = cv2.imencode('.png', resized)[1].tobytes()

return detections, imgbytes

**dashboard.py**

import PySimpleGUI as sg

import psutil

import GPUtil

# each individual graph size in pixels

GRAPH\_WIDTH, GRAPH\_HEIGHT = 120, 40

ALPHA = .7

class DashGraph(object):

def \_\_init\_\_(self, graph\_elem, starting\_count, color):

self.graph\_current\_item = 0

self.graph\_elem = graph\_elem # type:sg.Graph

self.prev\_value = starting\_count

self.max\_sent = 1

self.color = color

self.graph\_lines = []

def graph\_value(self, current\_value):

delta = current\_value - self.prev\_value

self.prev\_value = current\_value

self.max\_sent = max(self.max\_sent, delta)

percent\_sent = 100 \* delta / self.max\_sent

line\_id = self.graph\_elem.draw\_line(

(self.graph\_current\_item, 0), (self.graph\_current\_item, percent\_sent), color=self.color)

self.graph\_lines.append(line\_id)

if self.graph\_current\_item >= GRAPH\_WIDTH:

self.graph\_elem.delete\_figure(self.graph\_lines.pop(0))

self.graph\_elem.move(-1, 0)

else:

self.graph\_current\_item += 1

return delta

def graph\_percentage\_abs(self, value):

self.graph\_elem.draw\_line(

(self.graph\_current\_item, 0), (self.graph\_current\_item, value), color=self.color)

if self.graph\_current\_item >= GRAPH\_WIDTH:

self.graph\_elem.move(-1, 0)

else:

self.graph\_current\_item += 1

def human\_size(bytes, units=(' bytes', 'KB', 'MB', 'GB', 'TB', 'PB', 'EB')):

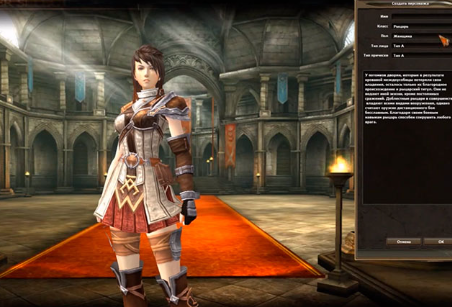
""" Returns a human readable string reprentation of bytes"""

return str(bytes) + units[0] if bytes < 1024 else human\_size(bytes >> 10, units[1:])

**ДОДАТОК Б**

**Опис гри (в зображеннях).**















s

