Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____ Информатика и вычислительная техника _____

КАФЕДРА _____Автоматизированные системы обработки информации и управления_____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

Обучение искусственного интеллекта для прохождения игры Dino Runner

Студенты__ИУ5-63Б___ _____ _____**Воронова О.А** __
(Группа) (Подпись, дата) (И.О.Фамилия)

_____ _____**Наказной Н.А** ___
(Подпись, дата) (И.О.Фамилия)

Руководитель _____ ___**Гапанюк Ю.Е.** __
(Подпись, дата) (И.О.Фамилия)

Консультант _____ _____
(Подпись, дата) (И.О.Фамилия)

2022 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой _____
(Индекс)

_____ _____
(И.О.Фамилия)

« _____ » _____ 20 ____ г.

З А Д А Н И Е
на выполнение научно-исследовательской работы

по теме ___ Обучение искусственного интеллекта для прохождения игры Dino Runner __
_____

_____

_____

Студенты группы ___ИУ5-63Б_____

____Воронова Ольга Александровна, Наказной Никита Александрович___
(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)

_Исследовательская_____

Источник тематики (кафедра, предприятие, НИР) _____

График выполнения НИР: 25% к ___ нед., 50% к ___ нед., 75% к __ нед., 100% к ___ нед.

Техническое задание: __ Обучение искусственного интеллекта для прохождения игры Dino Runner _____

Оформление научно-исследовательской работы:

Расчетно-пояснительная записка на _____ листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

_____

_____

Дата выдачи задания « _20__ » _____марта_____ 2022 г.

Руководитель НИР _____ ____Гапанюк Ю.Е ____
(Подпись, дата) (И.О.Фамилия)

Студенты _____**Воронова О.А. __**
(Подпись, дата) (И.О.Фамилия)

_____**Наказной Н.А. __**
(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

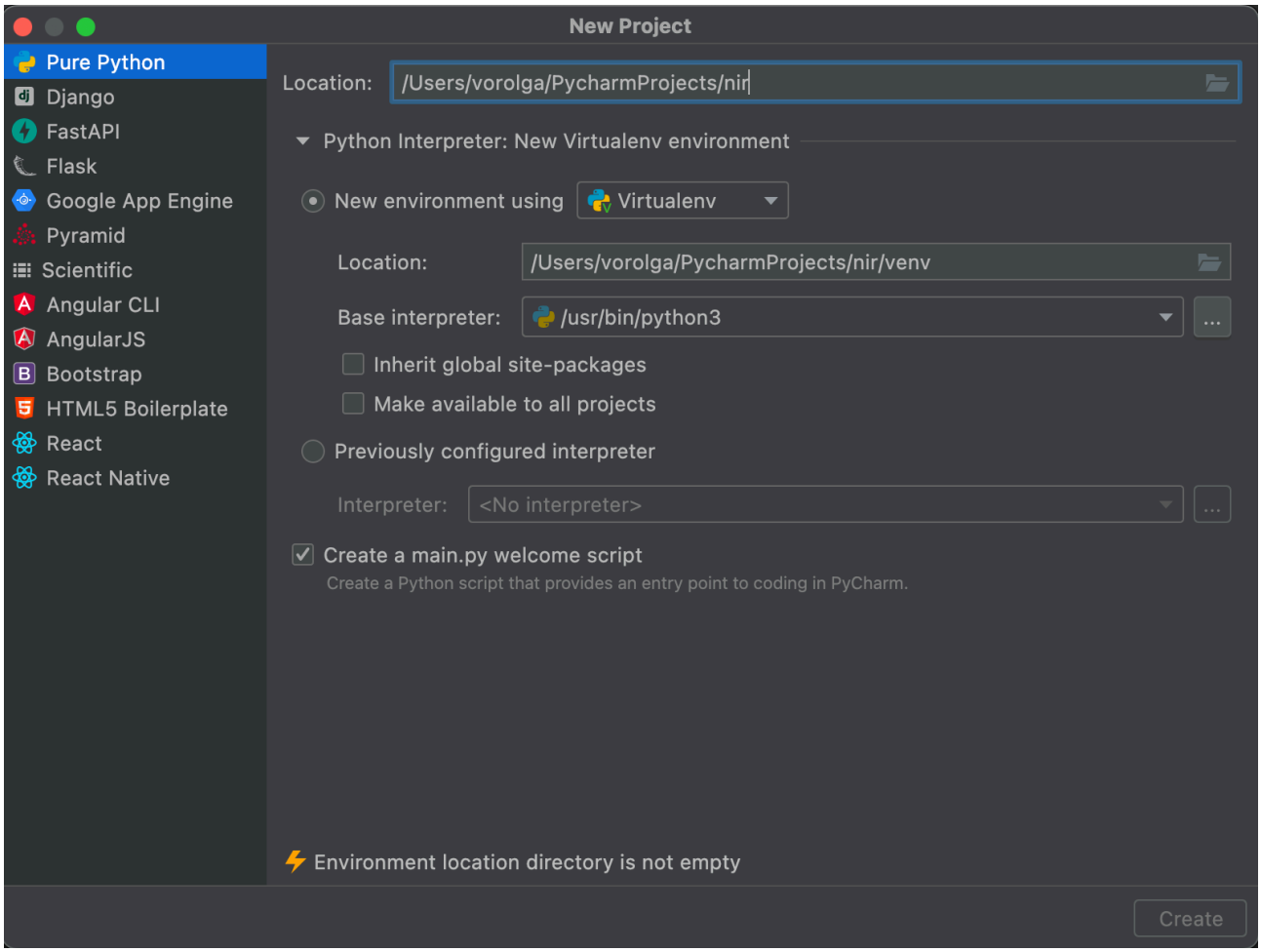
Оглавление

Введение

В современном мире машинное обучение одно из самых популярных направлений. Данное направление нашло реализацию также и в таком языке, как Python. NEAT — это метод, разработанный Кеннетом О. Стэнли для разработки произвольных нейронных сетей. NEAT-Python — это реализация NEAT на чистом Python без каких-либо зависимостей, кроме стандартной библиотеки Python

Ход работы

Для более удобного написания кода мы использовали IDE от JetBrains PyCharm. В начале необходимо создать проект.



Будем использовать библиотеку pygame для создания игры

Код программы

```
import pygame
import random
import sys
import math
from enum import Enum
import neat

width = 1280
height = 720
```

```
bg = (255, 255, 255, 255)

score = 0
score_speedup = 100
game_speed = 8.0
skins = ["default", "aqua", "black", "bloody", "cobalt", "gold", "insta",
         "lime", "magenta", "magma", "navy", "neon", "orange", "pinky",
         "purple", "rgb", "silver", "subaru", "sunny", "toxic"]
names = ["Флафи", "Фалафель", "Ведьмак", "Лютик", "Пучеглазик", "Слайм",
"Шустрый", "Следопыт",
         "Малыш", "Субарик", "Т-Рекс", "Птенец", "Рядовой", "Опытный",
"Ветеран", "Геймер",
         "Самурай", "Странник"]
generation = 0


class DinoState(Enum):
    RUN = 1
    JUMP = 2


class Dino:
    name = "Carl"
    jump_power = 10
    cur_jump_power = jump_power
    color = "default"
    sprites = {
        "run": [],
        "jump": []
    }
    image = None
    run_animation_index = [0, 5]
    hitbox = None
    state = DinoState.RUN

    def __init__(self, x, y, color="default", name=None):
        self.color = color
        self.load_sprites()
        self.hitbox = pygame.Rect(x, y, self.sprites["run"][0].get_width(),
self.sprites["run"][0].get_height())
        self.image = self.sprites["run"][0]

        if name is not None:
            self.name = name

    def load_sprites(self):

self.sprites["jump"].append(pygame.image.load(f"sprites/dino/{self.color}_jump.p
ng"))

self.sprites["run"].append(pygame.image.load(f"sprites/dino/{self.color}_run1.pn
g"))

self.sprites["run"].append(pygame.image.load(f"sprites/dino/{self.color}_run2.pn
g"))

    def update(self):
        if self.state == DinoState.RUN:
            self.run()
        elif self.state == DinoState.JUMP:
            self.jump()

    def run(self):
        self.sprites["run"][0] =
pygame.image.load(f"sprites/dino/{self.color}_run1.png")
        self.sprites["run"][1] =
```

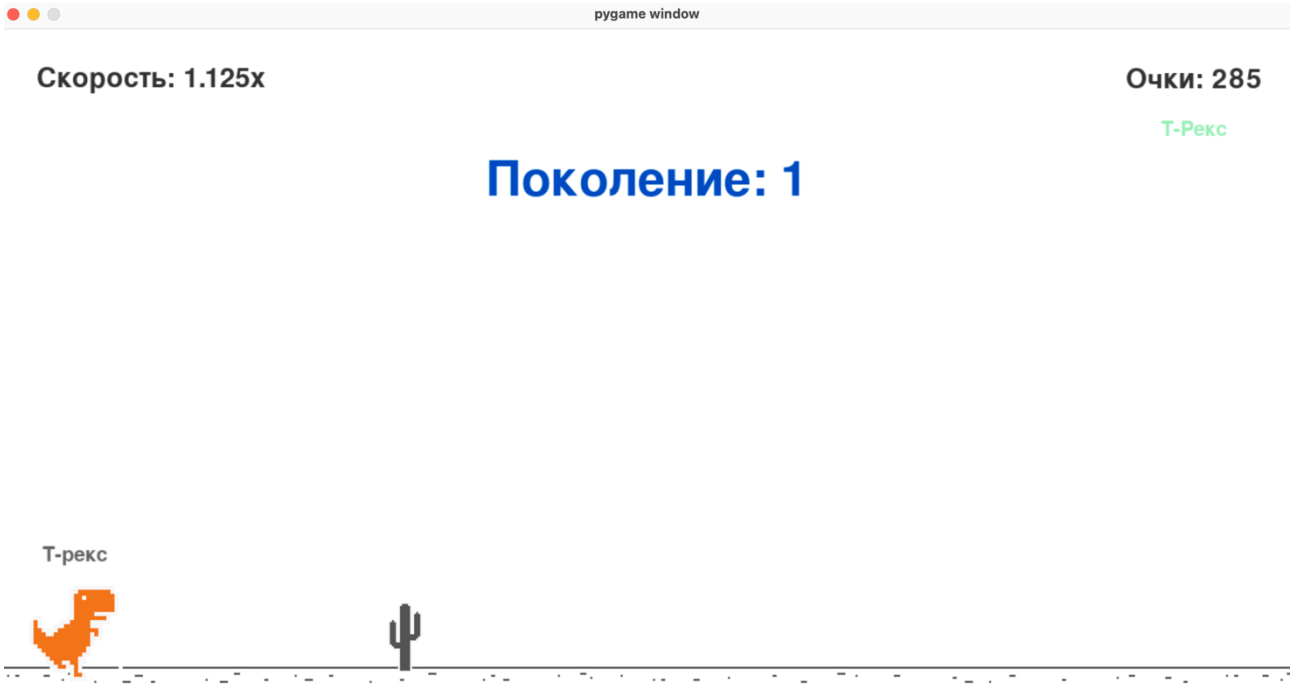
```
pygame.image.load(f"sprites/dino/{self.color}_run2.png")

        self.image = self.sprites["run"][self.run_animation_index[0] //
self.run_animation_index[1]]

        self.run_animation_index[0] += 1
        if self.run_animation_index[0] >= self.run_animation_index[1] * 2:
            self.run_animation_index[0] = 0

    def jump(self):
        if self.state == DinoState.JUMP:
            self.hitbox.y -= self.cur_jump_power * (2 * (game_speed / 8))
            self.cur_jump_power -= 0.5 * (game_speed / 8)

            # if self.cur_jump_power <= -self.jump_power:
            #     self.hitbox.y -= self.cur_jump_power * (2 * (game_speed / 8))
            #     self.state = DinoState.RUN
            #     self.cur_jump_power = self.jump_power
            if self.hitbox.y >= height - 170:
                self.hitbox.y = height - 170
                self.state = DinoState.RUN
                self.cur_jump_power = self.jump_power
        else:
            self.state = DinoState.JUMP
            self.image =
pygame.image.load(f"sprites/dino/{self.color}_jump.png")
            # self.image = self.sprites["jump"][0]

    def draw(self, scr, fnt=None):
        scr.blit(self.image, (self.hitbox.x, self.hitbox.y))

        if fnt is not None:
            c_label = fnt.render(self.name.capitalize(), True, (100, 100, 100))
            c_label_rect = c_label.get_rect()
            c_label_rect.center = (self.hitbox.x + 45, self.hitbox.y - 30)
            scr.blit(c_label, c_label_rect)


class Cactus:
    available_types = ["1", "2", "3", "4", "5", "6"]
    cactus_type = None
    image = None
    hitbox = None
    is_active = True

    def __init__(self, x, y, forced_type=None):
        if forced_type is not None:
            self.cactus_type = forced_type

        self.load_image()
        self.hitbox.x = x
        self.hitbox.y = y - self.hitbox.height  # origin from bottom

    def randomize_cactus(self):
        self.cactus_type = random.choice(self.available_types)

    def load_image(self):
        if self.cactus_type is None:
            self.randomize_cactus()

        self.image = pygame.image.load(f"sprites/cactus/{self.cactus_type}.png")
        self.hitbox = self.image.get_rect()

    def update(self):
        self.hitbox.x -= game_speed
        if self.hitbox.x < -self.hitbox.width:
```

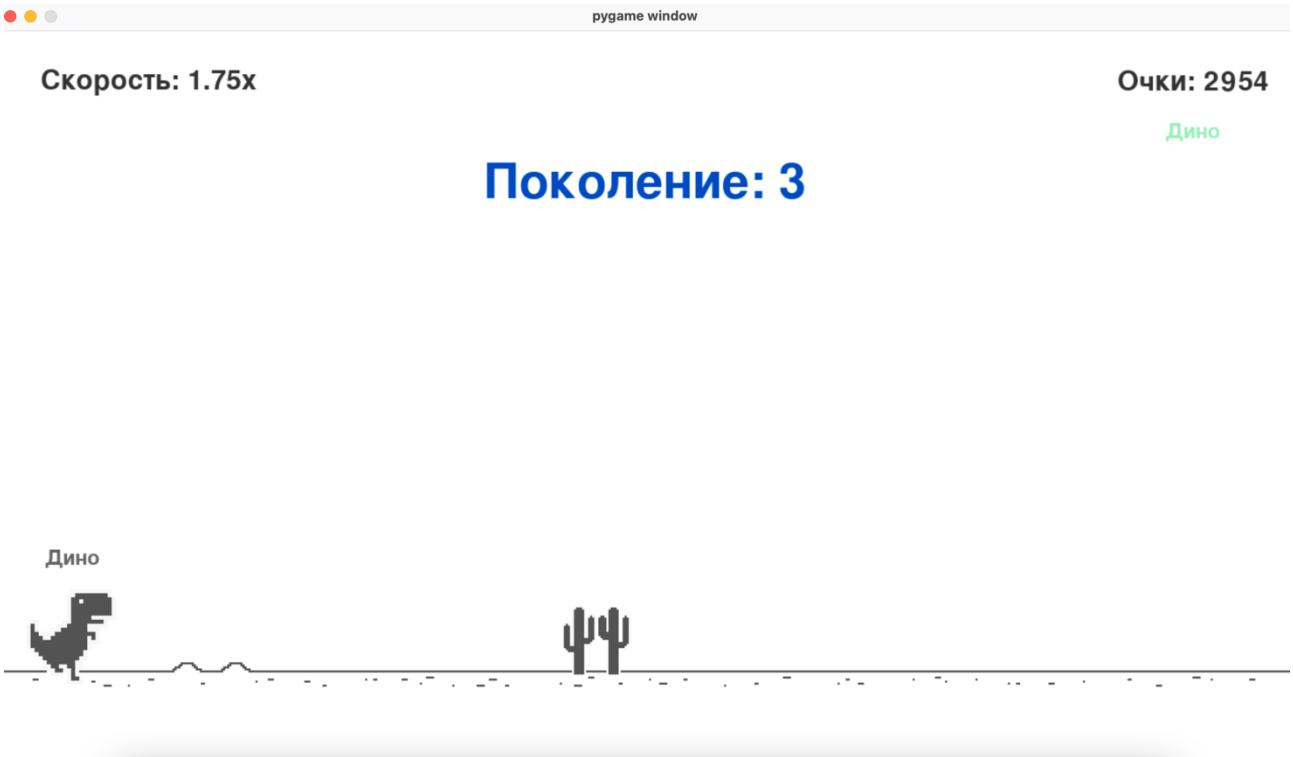
```
            # remove this cactus
            self.is_active = False

    def draw(self, scr):
        scr.blit(self.image, self.hitbox)


def calc_dist(a, b):
    dx = a[0] - b[0]
    dy = a[1] - b[1]

    return math.sqrt(dx ** 2 + dy ** 2)


def run_game(genomes, config):
    global game_speed, score, enemies, dinosaurs, generation, score_speedup

    generation += 1
    game_speed = 8.0
    score = 0
    score_speedup = 100
    enemies = [Cactus(width + 300 / random.uniform(0.8, 3), height - 85),
               Cactus(width * 2 + 200 / random.uniform(0.8, 3), height - 85),
               Cactus(width * 3 + 400 / random.uniform(0.8, 3), height - 85)]
    dinosaurs = []
    nets = []
    skins_copy = skins[:]
    names_copy = names[:]

    # init genomes
    for i, g in genomes:
        net = neat.nn.FeedForwardNetwork.create(g, config)
        nets.append(net)
        g.fitness = 0  # every genome is not successful at the start

        skin = "default"
        if len(skins_copy):
            skin = skins_copy.pop()

        name = "Дино"
        if len(names_copy):
            name = names_copy.pop()

        dinosaurs.append(Dino(30, height - 170, skin, name))

    # init
    pygame.init()
    screen = pygame.display.set_mode((width, height))
    clock = pygame.time.Clock()
    road_chunks = [
        [pygame.image.load('sprites/road.png'), [0, height - 100]],
        [pygame.image.load('sprites/road.png'), [2404, height - 100]]
    ]
    font = pygame.font.SysFont("Roboto Condensed", 30)
    score_font = pygame.font.SysFont("Roboto Condensed", 40)
    dname_font = pygame.font.SysFont("Roboto Condensed", 30)
    heading_font = pygame.font.SysFont("Roboto Condensed", 70)

    # dinosaurs = [Dino(30, height-170, "subaru", "Howdy")]

    # the loop
    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
```

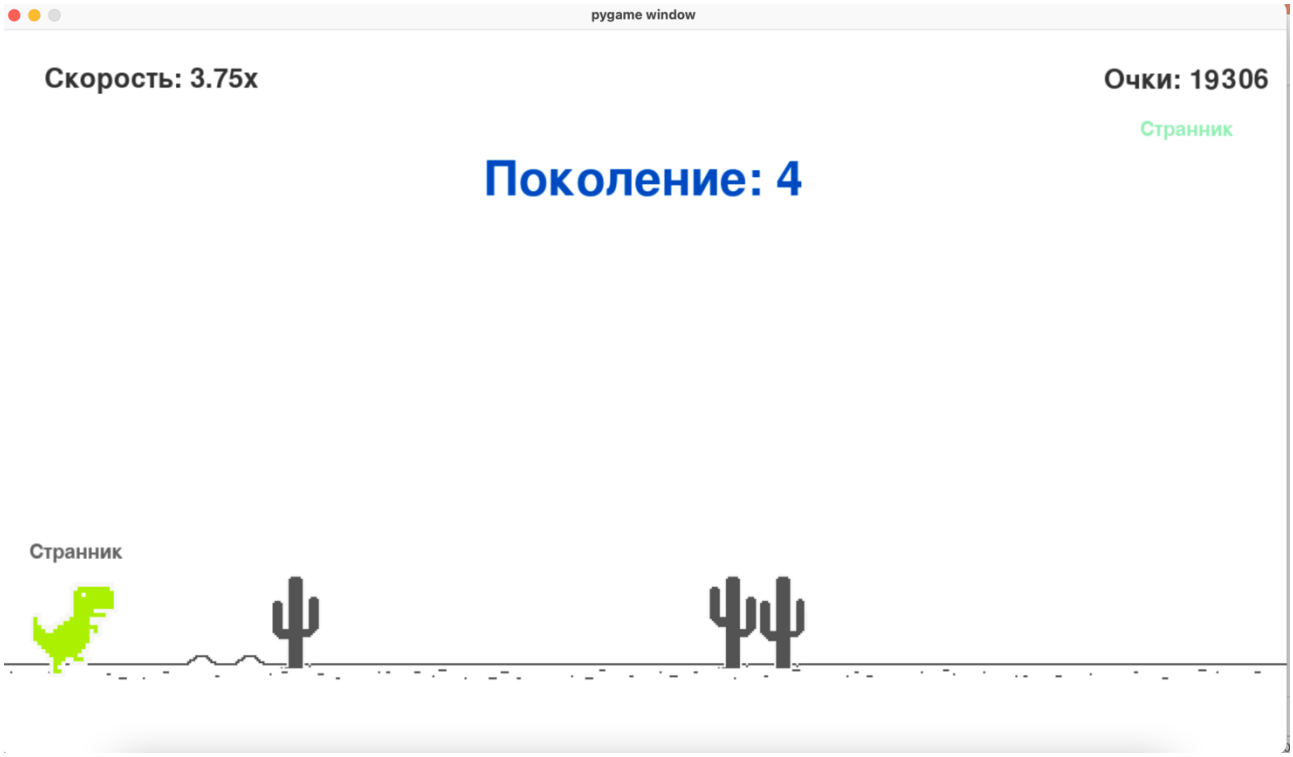
```
        # display bg & road
        screen.fill(bg)
        for road_chunk in road_chunks:
            if road_chunk[1][0] <= -2400:
                road_chunk[1][0] = road_chunks[len(road_chunks) - 1][1][0] +
2400

                road_chunks[0], road_chunks[1] = road_chunks[1], road_chunks[0]
                break

            road_chunk[1][0] -= game_speed
            screen.blit(road_chunk[0], (road_chunk[1][0], road_chunk[1][1]))

        # draw dino
        for dino in dinosaurs:
            dino.update()
            dino.draw(screen, font)

        # quit if there is no dinos left
        if len(dinosaurs) == 0:
            print(f"Max score - {str(math.floor(score))} for generation -
{generation}")
            break

        # generate enemies
        if len(enemies) < 3:
            enemies.append(Cactus(enemies[len(enemies) - 1].hitbox.x + width /
random.uniform(0.8, 3), height - 85))

        # draw enemies
        rem_list = []
        for i, enemy in enumerate(enemies):
            enemy.update()
            enemy.draw(screen)

            if not enemy.is_active:
                rem_list.append(i)
                continue

            for j, dinosaur in enumerate(dinosaurs):
                if dinosaur.hitbox.colliderect(enemy.hitbox):
                    genomes[j][1].fitness -= 10  # lower fitness (failed)
                    dinosaurs.pop(j)

                    genomes.pop(j)
                    nets.pop(j)

        for i in rem_list:
            enemies.pop(i)

            for j, dinosaur in enumerate(dinosaurs):
                genomes[j][1].fitness += 5  # raise fitness (+5 for every enemy)


        # controls
        for i, dinosaur in enumerate(dinosaurs):
            output = nets[i].activate((dinosaur.hitbox.y,
                                       calc_dist((dinosaur.hitbox.x,
dinosaur.hitbox.y), enemies[0].hitbox.midtop),
                                       enemies[0].hitbox.width,
                                       game_speed))

            if output[0] > 0.5 and dinosaur.state is not DinoState.JUMP:
                dinosaur.jump()
                genomes[i][1].fitness -= 1  # every jump lowers the fitness
```

```
(assuming it's false jump)

        # read user input (jump test)
        # user_input = pygame.key.get_pressed()
        # if user_input[pygame.K_SPACE]:
        #     for dino in dinosaurs:
        #         if not dino.state == DinoState.JUMP:
        #             dino.jump()

        # score & game speed
        score += 0.5 * (game_speed / 4)
        if score > score_speedup:
            score_speedup += 100 * (game_speed / 2)
            game_speed += 1
            print(f"Game speed increased - {game_speed}")

        score_label = score_font.render("Очки: " + str(math.floor(score)), True,
(50, 50, 50))
        score_label_rect = score_label.get_rect()
        score_label_rect.center = (width - 100, 50)
        screen.blit(score_label, score_label_rect)

        # display dinosaurs names
        for i, dinosaur in enumerate(dinosaurs):
            dname_label = dname_font.render(dinosaur.name, True, (170, 238,
187))
            dname_label_rect = dname_label.get_rect()
            dname_label_rect.center = (width - 100, 100 + (i * 25))
            screen.blit(dname_label, dname_label_rect)

        # display generation
        label = heading_font.render("Поколение: " + str(generation), True, (0,
72, 186))
        label_rect = label.get_rect()
        label_rect.center = (width / 2, 150)
        screen.blit(label, label_rect)

        # display game speed
        score_label = score_font.render("Скорость: " + str(game_speed / 8) +
"x", True, (50, 50, 50))
        score_label_rect = score_label.get_rect()
        score_label_rect.center = (150, 50)
        screen.blit(score_label, score_label_rect)

        # flip & tick
        pygame.display.flip()
        clock.tick(60)  # fixed 60 fps


if __name__ == "__main__":
    # setup config
    config_path = "config-feedforward.txt"
    config = neat.config.Config(neat.DefaultGenome, neat.DefaultReproduction,
neat.DefaultSpeciesSet,
                                neat.DefaultStagnation, config_path)

    # init NEAT
    p = neat.Population(config)

    # run NEAT
    p.run(run_game, 1000)
```

Результаты


pygame window
Скорость: 1.125x
Очки: 285
T-Рекс
Поколение: 1
T-рекс


pygame window
Скорость: 1.75x
Очки: 2954
Дино
Поколение: 3
Дино




```
Game speed increased - 9.0
Max score - 166 for generation - 1
Game speed increased - 9.0
Max score - 176 for generation - 2
Game speed increased - 9.0
Game speed increased - 10.0
Game speed increased - 11.0
Game speed increased - 12.0
Game speed increased - 13.0
Game speed increased - 14.0
Max score - 3215 for generation - 3
Game speed increased - 9.0
Game speed increased - 10.0
Game speed increased - 11.0
```

Выводы.

Из результатов работы видно, что спустя несколько поколений ИИ на основе лучших результатов предыдущих поколений стал проходить игру дальше. На основе этого можно сделать вывод, что ИИ эволюционирует.

Литература

NEAT-Python
https://neat-python.readthedocs.io/en/latest/
Pygame
https://www.pygame.org/news