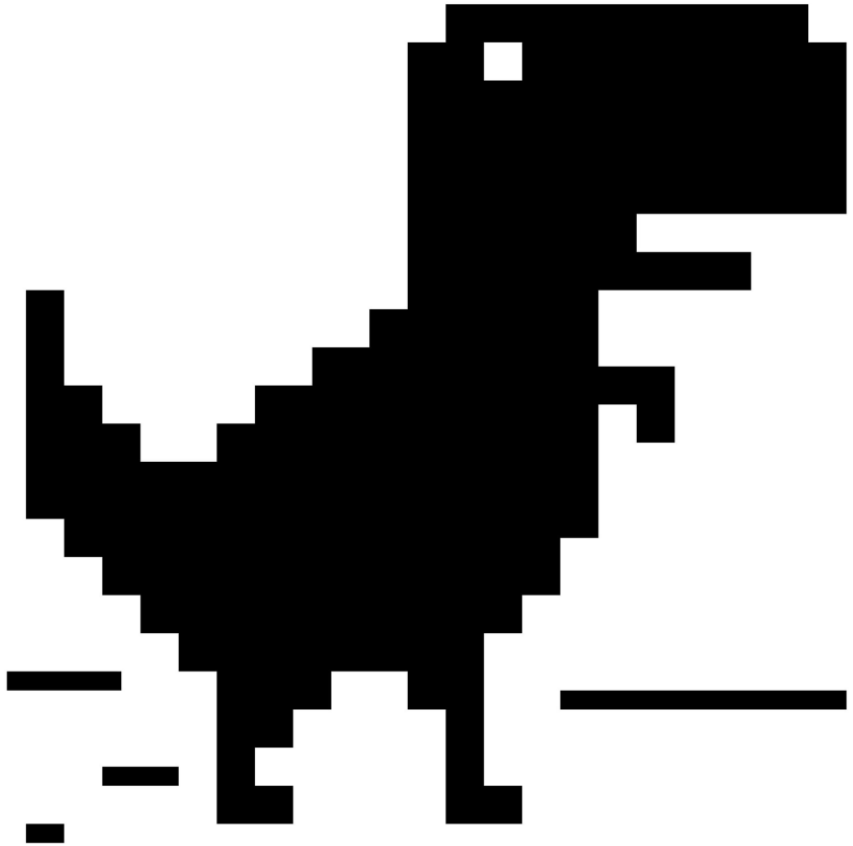
# Проект по дисциплине “Основы алгоритмизации и программирования”

Тема: Разработка игры “Динозаврик из Chrome”

Подготовили студенты группы ИС-24:
Желяк Мария и Скупков Алексей

# Цель проекта

Разработать игру динозавра из Google Chrome



# Задачи проекта:

- Выбор темы проекта и дизайна
- Изучение вспомогательного материала
- Написание кода
- Тестирование и устранение ошибок
- Создание отчета
- Защита проекта

# Выбор темы проекта и дизайна

При выбор темы не возникло трудностей т. к. мы давно хотели сделать динозаврика из Google Chrome. Это интересная и достаточно простая игра в реализации. Дизайн был выбран стандартный как и в оригинальной игре.

# Написание кода

Используем библиотеки pygame, os и random.

Устанавливаем разрешение графического окна и добавляем всем объектам картинки.

```
import pygame #импортируем pygame
import os
import random
pygame.init()

SCREEN_HEIGHT = 600 #высота экрана
SCREEN_WIDTH = 1100 #ширина экрана
SCREEN = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT)) #определяем дисплей на котором будет показана игра
#в переменную running мы положили все картинки с динозавром
RUNNING = [pygame.image.load(os.path.join("Assets/Dino", "DinoRun1.png")), #бег динозавра
           pygame.image.load(os.path.join("Assets/Dino", "DinoRun2.png"))]
JUMPING = pygame.image.load(os.path.join("Assets/Dino", "DinoJump.png")) #прыжок динозавра
DUCKING = [pygame.image.load(os.path.join("Assets/Dino", "DinoDuck1.png")), #ныряние динозавра
          pygame.image.load(os.path.join("Assets/Dino", "DinoDuck2.png"))]
#так же мы добавили обьекты которые будут как препятствие
SMALL_CACTUS = [pygame.image.load(os.path.join("Assets/Cactus", "SmallCactus1.png")), #маленький кактус
                pygame.image.load(os.path.join("Assets/Cactus", "SmallCactus2.png")),
                pygame.image.load(os.path.join("Assets/Cactus", "SmallCactus3.png"))]
LARGE_CACTUS = [pygame.image.load(os.path.join("Assets/Cactus", "LargeCactus1.png")), #большой кактус
                pygame.image.load(os.path.join("Assets/Cactus", "LargeCactus2.png")),
                pygame.image.load(os.path.join("Assets/Cactus", "LargeCactus3.png"))]
BIRD = [pygame.image.load(os.path.join("Assets/Bird", "Bird1.png")), #птицы
        pygame.image.load(os.path.join("Assets/Bird", "Bird2.png"))]
#так же мы добавили сюда фон
CLOUD = pygame.image.load(os.path.join("Assets/Other", "Cloud.png")) #облака
BG = pygame.image.load(os.path.join("Assets/Other", "Track.png")) #а это дорожка по которой будет бежать динозавр
```

# Написание кода

Создаем базовые свойства для динозавра: добавляем координаты, делаем так чтобы он по умолчанию бежал, используем все его положения в прыжке и в приседании.

Также добавляем управление.

```
def update(self, userInput): #функция обновления динозавра
        if self.dino_duck: #функции которые проверяют состояние динозавра при определённых действиях
            self.duck()
        if self.dino_run:
            self.run()
        if self.dino_jump:
            self.jump()

        if self.step_index >= 10: #индекс шага который сбрасывается каждые 10 шагов
            self.step_index = 0

        if userInput[pygame.K_UP] and not self.dino_jump: #это операторы которые определяют состояние динозавра
            self.dino_duck = False
            self.dino_run = False
            self.dino_jump = True
        elif userInput[pygame.K_DOWN] and not self.dino_jump:
            self.dino_duck = True
            self.dino_run = False
            self.dino_jump = False
        elif not (self.dino_jump or userInput[pygame.K_DOWN]):
            self.dino_duck = False
            self.dino_run = True
            self.dino_jump = False
```

```
def __init__(self): #мы добавляем динозавра каждый раз когда создаётся объект этого класса
    self.duck_img = DUCKING #включаем все изображения динозавра
    self.run_img = RUNNING
    self.jump_img = JUMPING

    self.dino_duck = False #делаем так чтобы динозавр по умолчанию бежал а не прыгал или нырял
    self.dino_run = True
    self.dino_jump = False

    self.step_index = 0 #индекс шага
    self.jump_vel = self.JUMP_VEL
    self.image = self.run_img[0] #это первое изображение при создании динозавра
    self.dino_rect = self.image.get_rect() #это изображение когда динозавр умирает
    self.dino_rect.x = self.x_pos #подключаем координаты
    self.dino_rect.y = self.y_pos
```

```
def duck(self):
    self.image = self.duck_img[self.step_index // 5]  # изображение ныряющего
    self.dino_rect = self.image.get_rect()
    self.dino_rect.x = self.x_pos  # получаем координаты динозавра
    self.dino_rect.y = self.y_pos_duck
    self.step_index += 1  # увеличиваем индекс шага на единицу (когда значения от 0 до 5 показываются первые изображения динозавра)
def run(self):
    self.image = self.run_img[self.step_index // 5] #изображение бегущего динозавра для анимации
    self.dino_rect = self.image.get_rect()
    self.dino_rect.x = self.x_pos #получаем координаты динозавра
    self.dino_rect.y = self.y_pos
    self.step_index += 1
def jump(self): #здесь мы сделали так чтобы при прыжке дино по оси у значение уменьшалось так как он прыгает, а потом опять увеличив
    self.image = self.jump_img
    if self.dino_jump:
        self.dino_rect.y -= self.jump_vel * 4
        self.jump_vel -= 0.8 #тут мы уменьшили скорость
    if self.jump_vel < - self.JUMP_VEL:
        self.dino_jump = False
        self.jump_vel = self.JUMP_VEL
```

```
class Dinosaur():
    x_pos =80 #это координаты динозавра чтобы он стоял на одном месте
    y_pos =310
    y_pos_duck =340 #координаты динозавра когда он ныряет
    JUMP_VEL =8.5 #скорость динозавра когда он оторвется от земли для прыжка
```

# Написание кода

```
class Cloud: #облака
    def __init__(self): #функция инициализации
        self.x = SCREEN_WIDTH + random.randint(800, 1000) #координаты облака при его создании
        self.y = random.randint(50, 100)
        self.image = CLOUD #задаём изображение облака и его ширину
        self.width = self.image.get_width()

    def update(self): #функция обновления
        self.x -= game_speed #мы делаем так чтобы облако перемещалось справа на лево ( вычитаем скорость игры )
        if self.x < -self.width: #и когда облако исчезает с экрана мы сбрасываем координаты облака чтобы оно появилось снова
            self.x = SCREEN_WIDTH + random.randint(1000, 1500)
            self.y = random.randint(50, 100)
```

Добавляем облака.

По факту имея одно облако мы будем его запускать с определенной периодичностью. Облако пролетает и обновляет свои координаты на начальные.

# Написание кода

Добавляем препятствия: маленькие и большие кактусы и птичек. Устанавливаем координаты по y и добавляем анимацию птички.

```
class Obstacle: #препятствия
    def __init__(self, image, type):
        self.image = image #изображение препятствия
        self.type = type #тип препятствия
        self.rect = self.image[self.type].get_rect()
        self.rect.x = SCREEN_WIDTH #каждый раз когда появляется препятствие

    def update(self):
        self.rect.x -= game_speed
        if self.rect.x < -self.rect.width: #это поможет убрать перпятствие
            obstacles.pop()

    def draw(self, SCREEN):
        SCREEN.blit(self.image[self.type], self.rect) #выводит изображение
```

```
class Bird(Obstacle):
    def __init__(self, image):
        self.type = 0 #устанавливаем тип обьекта
        super().__init__(image, self.type)
        self.rect.y = 250 #координаты
        self.index = 0
#анимация птички
    def draw(self, SCREEN):
        if self.index >= 9: #оператор сбрасывает значение на ноль
            self.index = 0
        SCREEN.blit(self.image[self.index//5], self.rect) #выводи
        self.index += 1 #увеличиваем индекс на 1
```

```
class SmallCactus(Obstacle): #маленький кактус
    def __init__(self, image): #инициализируем изображение в качестве
        self.type = random.randint(0, 2) #устанавливаем тип
        super().__init__(image, self.type)
        self.rect.y = 325 #устанавливаем координаты

#тоже самое мы сделаем с классом для большого кактуса только координат

class LargeCactus(Obstacle):
    def __init__(self, image):
        self.type = random.randint(0, 2)
        super().__init__(image, self.type)
        self.rect.y = 300
```

# Написание кода

```
def main():
    global game_speed, x_pos_bg, y_pos_bg, points, obstacles
    run = True #run это как переключатель для цикла while
    clock = pygame.time.Clock() #добавили часы
    player = Dinosaur() #добавили игрока
    cloud = Cloud() #экземпляр класса облака
    game_speed = 14 #эта переменная отслеживает скорость всего что находится на экране
    x_pos_bg = 0 #координаты фона
    y_pos_bg = 380
    points = 0 #параменная баллов, в начале игры очков будет 0
    font = pygame.font.Font('freesansbold.ttf', 20) #добавляем шрифт
    obstacles = []
    death_count = 0 #коллмчество смертей

    def score(): #функция счета
        global points, game_speed
        points += 1
        if points % 100 == 0: #тут каждые 100 очков увеличивается скорость игры на 1
            game_speed += 1

        text = font.render("Points: " + str(points), True, (0, 0, 0)) #добавляем текст очков
        textRect = text.get_rect() #координаты прямоугольника в котором отображаются очки
        textRect.center = (1000, 40) #центр этого прямоугольника устанавливаем в правый верхний угол
        SCREEN.blit(text, textRect) #выводим на экран
```

Добавляем функцию main благодаря которой игра запускается и работает

Добавляем функцию счета: выводим количество очков и каждые 100 очков скорость будет увеличиваться.

# Написание кода

Добавляем фон, изображение фона и двигаем их также как и облака и делаем дорожки.

Также теперь при нажатии на крестик во время игры она остановится, а при нажатии на крестик после поражения - закроется.

Устанавливаем задержку после проигрыша.

```
def background():
    global x_pos_bg, y_pos_bg #глобальные координаты фона
    image_width = BG.get_width() #изображение фона
    SCREEN.blit(BG, (x_pos_bg, y_pos_bg)) #выводим изображение на экран
    SCREEN.blit(BG, (image_width + x_pos_bg, y_pos_bg))
    if x_pos_bg <= -image_width: #мы сделали тоже самое с фоном что и с облаком, вычитаем скорость и каждый
        SCREEN.blit(BG, (image_width + x_pos_bg, y_pos_bg))
        x_pos_bg = 0
    x_pos_bg -= game_speed

while run: #мы сделали так чтобы игрок нажимая на крестик заканчивал игру
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False
```

```
for event in pygame.event.get(): #добавляем опцию безопасного выхода из игры
    if event.type == pygame.QUIT:
        exit()
```

```
for obstacle in obstacles:
    obstacle.draw(SCREEN)
    obstacle.update()
    if player.dino_rect.colliderect(obstacle.rect):
        pygame.time.delay(500) #задержка времени чтобы сначала увидеть что ты проиграл а потом перейти в гоавное меню
        death_count += 1 #увеличение смертей
        menu(death_count)
```

# Написание кода

Создаем генерацию объектов

Выпадает число от 0 до 3:

Если 0 то появляется маленький кактус

Если 1 - большой кактус

Если 2 - птичка

```
if len(obstacles) == 0:
    if random.randint(0, 2) == 0:
        obstacles.append(SmallCactus(SMALL_CACTUS))
    elif random.randint(0, 2) == 1:
        obstacles.append(LargeCactus(LARGE_CACTUS))
    elif random.randint(0, 2) == 2:
        obstacles.append(Bird(BIRD))
```

```python
def menu(death_count):
    global points
    run = True
    while run:
        SCREEN.fill((255, 255, 255)) #создаём фон и определяем шрифт текста для меню
        font = pygame.font.Font('freesansbold.ttf', 30)

        if death_count == 0: #если смертей 0 то выводим это
            text = font.render("Нажмите любую клавишу для старта", True, (0, 0, 0))
        elif death_count > 0: #если смертей больше нуля то выводим и текст и результат
            text = font.render("Game Over", True, (0, 0, 0))
            score = font.render("Ваш рекорд: " + str(points), True, (0, 0, 0))
            scoreRect = score.get_rect() #определяем где это будет находится на экране
            scoreRect.center = (SCREEN_WIDTH // 2, SCREEN_HEIGHT // 2 + 50)
            SCREEN.blit(score, scoreRect) #выводим на экран
        textRect = text.get_rect() #положение текста
        textRect.center = (SCREEN_WIDTH // 2, SCREEN_HEIGHT // 2)
        SCREEN.blit(text, textRect) #выводим текст на экран
        SCREEN.blit(RUNNING[0], (SCREEN_WIDTH // 2 - 20, SCREEN_HEIGHT // 2 - 140))#там же где и т
        pygame.display.update() #обновляем дисплей
        for event in pygame.event.get(): #добавляем опцию безопасного выхода из игры
            if event.type == pygame.QUIT:
                run = False
            if event.type == pygame.KEYDOWN: #делаем так чтобы игрок при сталкновении с препятстви
```

# Написание кода

Добавляем меню при начале игры и после игры. Определяем фон, надписи в меню, а также в конце игры надпись и количество очков. И делаем возможность перезапустить игру.

# Тестирование и устранение ошибок

В процессе выполнения у нас возникли ошибки. Мы не добавили скорость динозавра во время прыжка, поэтому вылетала ошибка. Возникла ошибка с координатами облака, из-за чего облако попросту не появлялось. По итогу все ошибки были исправлены. Также нашелся один баг который позволял продолжить игру после поражения.

# Вывод

По итогу проделанной работы мы реализовали нашу задумку динозаврика из Google Chrome на Python. Цель работы и поставленные задачи были выполнены, результат работы выгружен на GitHub.

# Вспомогательный материал

https://myrusakov.ru/pygame-animation-loop.html

https://waksoft.susu.ru/2019/04/24/pygame-shpargalka-dlja-ispolzovanija/

https://proglib.io/p/samouchitel-po-python-dlya-nachinayushchih-chast-21-osnovy-razrabotki-igr-na-pygame-2023-05-29