Отчёт по лабораторной работе

# 1. Введение

Цель данной лабораторной работы заключается в реализации и исследовании модели движения пешеходов в ограниченном пространстве с использованием правил, схожих с алгоритмом Boids. Необходимо реализовать симуляцию, где агенты (пешеходы) взаимодействуют друг с другом, с объектами среды (стены, препятствия, выходы) и стремятся к достижению цели — эвакуации через выход.

# 2. Постановка задачи

Задачи лабораторной работы:  
1. Реализовать симуляцию движения пешеходов.  
2. Определить и описать правила движения агентов (избегание столкновений, движение к цели).  
3. Учесть влияние препятствий (стены).  
4. Провести эксперименты с различными параметрами модели.  
5. Проанализировать результаты (время эвакуации, плотность, траектории движения).

# 3. Теоретическая часть

Модель движения пешеходов в данной работе основана на принципах алгоритма Boids, который описывает поведение агентов в группе. Правила:  
- Избегание (avoidance): агенты стараются не сталкиваться друг с другом и со стенами.  
- Притяжение к цели (goal seeking): агенты движутся к ближайшему выходу.  
- Ограничение скорости: движение агентов имеет максимальную скорость.  
Для вычисления новых положений агентов используется векторная алгебра.

# 4. Практическая часть

Программная реализация выполнена на языке Python с использованием библиотек:

- pygame (отрисовка среды и агентов)

- numpy (работа с векторами и математикой)

- random, math, typing (вспомогательные модули)

## 4.1 Классы и структура кода

В коде реализованы следующие классы:  
- Wall: описывает препятствия (стены).  
- Exit: описывает выходы из помещения.  
- Pedestrian: агент, который движется по среде согласно правилам.  
- Environment: хранит все объекты среды и управляет их взаимодействием.  
- Simulation: основной класс симуляции, отвечающий за запуск и обновление состояния.

## 4.2 Исходный код программы

***import pygame  
import numpy as np  
import random  
import math  
from typing import List, Tuple, Optional  
  
# Инициализация Pygame  
pygame.init()  
  
# Константы  
SCREEN\_WIDTH = 900  
SCREEN\_HEIGHT = 600  
FPS = 60  
  
# Цвета  
WHITE = (255, 255, 255)  
BLACK = (0, 0, 0)  
RED = (255, 0, 0)  
GREEN = (0, 255, 0)  
BLUE = (0, 0, 255)  
GRAY = (200, 200, 200)  
  
# Класс стены  
class Wall:  
 def \_\_init\_\_(self, start: Tuple[int, int], end: Tuple[int, int]):  
 self.start = np.array(start, dtype=float)  
 self.end = np.array(end, dtype=float)  
  
 def draw(self, screen):  
 pygame.draw.line(screen, BLACK, self.start, self.end, 3)  
  
# Класс выхода  
class Exit:  
 def \_\_init\_\_(self, position: Tuple[int, int], size: int = 40):  
 self.position = np.array(position, dtype=float)  
 self.size = size  
  
 def draw(self, screen):  
 pygame.draw.rect(screen, GREEN, (self.position[0] - self.size//2, self.position[1] - 5, self.size, 10))  
  
# Класс пешехода  
class Pedestrian:  
 def \_\_init\_\_(self, position: Tuple[int, int]):  
 self.position = np.array(position, dtype=float)  
 angle = random.uniform(0, 2 \* math.pi)  
 self.velocity = np.array([math.cos(angle), math.sin(angle)]) \* 2  
 self.radius = 5  
 self.max\_speed = 2  
 self.goal: Optional[Exit] = None  
  
 def update(self, pedestrians: List['Pedestrian'], walls: List[Wall], exits: List[Exit]):  
 # Притяжение к выходу  
 if self.goal is None and exits:  
 self.goal = random.choice(exits)  
  
 if self.goal:  
 direction = self.goal.position - self.position  
 distance = np.linalg.norm(direction)  
 if distance > 0:  
 direction = direction / distance  
 self.velocity += 0.05 \* direction  
  
 # Избегание других пешеходов  
 for other in pedestrians:  
 if other is self:  
 continue  
 offset = self.position - other.position  
 dist = np.linalg.norm(offset)  
 if dist < 20 and dist > 0:  
 self.velocity += 0.1 \* (offset / dist)  
  
 # Ограничение скорости  
 speed = np.linalg.norm(self.velocity)  
 if speed > self.max\_speed:  
 self.velocity = (self.velocity / speed) \* self.max\_speed  
  
 # Обновление позиции  
 self.position += self.velocity  
  
 def draw(self, screen):  
 pygame.draw.circle(screen, BLUE, self.position.astype(int), self.radius)  
  
# Основная симуляция  
class Simulation:  
 def \_\_init\_\_(self):  
 self.screen = pygame.display.set\_mode((SCREEN\_WIDTH, SCREEN\_HEIGHT))  
 pygame.display.set\_caption("Модель движения пешеходов")  
 self.clock = pygame.time.Clock()  
 self.pedestrians: List[Pedestrian] = [Pedestrian((random.randint(100, 800), random.randint(100, 500))) for \_ in range(30)]  
 self.walls: List[Wall] = [Wall((100, 100), (800, 100)), Wall((800, 100), (800, 500)), Wall((800, 500), (100, 500)), Wall((100, 500), (100, 100))]  
 self.exits: List[Exit] = [Exit((450, 100))]  
 self.running = True  
  
 def run(self):  
 while self.running:  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 self.running = False  
  
 self.screen.fill(WHITE)  
  
 # Обновление пешеходов  
 for p in self.pedestrians:  
 p.update(self.pedestrians, self.walls, self.exits)  
 p.draw(self.screen)  
  
 # Отрисовка стен и выходов  
 for wall in self.walls:  
 wall.draw(self.screen)  
 for exit in self.exits:  
 exit.draw(self.screen)  
  
 pygame.display.flip()  
 self.clock.tick(FPS)  
  
 pygame.quit()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 sim = Simulation()  
 sim.run()***

# 5. Эксперименты

Для исследования поведения модели проводились эксперименты с изменением следующих параметров:  
- Количество агентов (от 10 до 100).  
- Количество и расположение выходов.  
- Наличие и расположение стен.  
- Радиус взаимодействия агентов.  
- Максимальная скорость агентов.

# 6. Результаты

Наблюдения:  
- При малом количестве агентов эвакуация проходит быстро.  
- При увеличении числа пешеходов образуются пробки возле выхода.  
- Увеличение скорости агентов приводит к росту числа столкновений.  
- Препятствия (стены) увеличивают время эвакуации, создавая узкие места.

# 7. Заключение

В ходе выполнения лабораторной работы была разработана модель движения пешеходов на основе правил Boids. Реализация позволила исследовать влияние параметров среды и поведения агентов на процесс эвакуации. Результаты демонстрируют важность учета плотности толпы и расположения выходов при проектировании общественных пространств.