Лабораторная работа № 14

**Знакомство с библиотеками языка Python. PyGame.**

**Цель работы:** приобретение практических навыков создания приложений на языке Python, быстрая разработка 2d игр.

**Задание**:

Для выполнения лабораторной работы требуется установить интерпретатор Python версии 3.5+. Выполнить написание программы-сценария в соответствии с вариантом задания (табл. 1). Провести тестирование. Оформить отчет.

Установка библиотеки

pip install pygame

Таблица 1. – Варианты задания

|  |  |
| --- | --- |
| 1. | 2048 с возможностью выбора размера поля |
| 2. | Лабиринт |
| 3. | Домашний питомец |
| 4. | Тетрис |
| 5. | Space shooter |
| 6. | Клеточный tower-defense |
| 7. | Гонки |
| 8. | Игра «Жизнь» |
| 9. | Змейка |
| 10. | PacMan |

Пример игры для космической стрелялки:

**Основной игровой цикл pygame:**

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

size = width, height = 800, 800

game = Game(size)

running = True

while running:

if pygame.key.get\_pressed()[pygame.K\_RIGHT]:

game.player\_right()

if pygame.key.get\_pressed()[pygame.K\_LEFT]:

game.player\_left()

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

if event.type == pygame.MOUSEBUTTONDOWN:

game.on\_click(event.pos)

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_SPACE:

game.player\_fire()

if event.key == pygame.K\_ESCAPE:

game.create\_menu()

pygame.display.flip()

game.draw()

pygame.quit()

**Основной игровой класс:**

class Game:

def \_\_init\_\_(self, size):

pygame.init()

self.clock = pygame.time.Clock()

self.tick = 0

self.size = size

self.records = Recors()

self.state\_game = 0 # 0 открыто меню

# 1 идет игра

# 2 информация об игре

# 3 конец игры

self.create\_menu()

self.create\_new\_game()

def create\_new\_game(self):

self.screen = pygame.display.set\_mode(self.size)

self.about = About(self.size)

self.end\_game = TheEndGame(self.size, result=0)

self.sp\_player = pygame.sprite.Group()

self.bullets = pygame.sprite.Group()

self.enemy = pygame.sprite.Group()

self.background = BackGround(self.size, 1)

self.player = Player(0, width, width // 2, height - 40)

self.sp\_player.add(self.player)

self.scenario = ScenarioAsteroid(self.size[0], self.size[1])

def create\_new\_player(self):

player = Player(0, self.size[0], self.size[0] // 2, self.size[1] - 20)

self.player = player

def draw\_background(self, tick):

self.screen.fill((0, 0, 0))

self.background.draw(self.screen, tick)

def draw(self):

self.tick = self.clock.tick()

if self.state\_game == 1:

f = pygame.sprite.groupcollide(self.bullets, self.enemy, True, True)

if f:

self.records.add\_hit()

for bullet, enemy in f.items():

self.scenario.remove\_enemy(enemy)

self.draw\_background(tick=self.tick)

self.bullets.update(tick=self.tick)

self.bullets.draw(self.screen)

self.sp\_player.update(tick=self.tick)

self.sp\_player.draw(self.screen)

self.enemy.update(tick=self.tick)

self.enemy.draw(self.screen)

self.records.draw(self.screen)

if self.scenario.is\_end():

self.scenario = self.scenario.next\_scenario()

self.start\_scenario()

elif self.state\_game == 0:

self.draw\_background(tick=self.tick)

self.draw\_menu()

elif self.state\_game == 2:

self.draw\_background(tick=self.tick)

self.draw\_about()

elif self.state\_game == 3:

self.draw\_background(tick=self.tick)

self.draw\_the\_end\_game()

def player\_left(self):

self.player.move\_left(self.tick)

def player\_right(self):

self.player.move\_right(self.tick)

def player\_fire(self):

bullet = self.player.make\_shot()

self.bullets.add(bullet)

def start\_scenario(self):

if self.scenario.is\_last():

self.state\_game = 3

else:

enemys = self.scenario.get\_enemys()

for enemy in enemys:

self.enemy.add(enemy)

def draw\_about(self):

self.about.draw(self.screen)

def draw\_menu(self):

self.menu.draw(self.screen)

def draw\_the\_end\_game(self):

self.end\_game.draw(self.screen, result=self.records.current\_record)

def create\_menu(self):

self.state\_game = 0

self.menu = Menu(self.size)

def on\_click(self, pos):

if self.state\_game == 0:

self.menu.click(pos, self)

**Класс для учета очков игрока**

class Records:

def \_\_init\_\_(self):

self.current\_record = 0

def add\_hit(self):

self.current\_record += 1

def draw(self, screen):

font = pygame.font.Font(None, 100)

textsurface = font.render(str(self.current\_record), True, (255, 0, 0))

screen.blit(textsurface, (20, 20))

**Модуль с формированием игровых сценариев.** Обратите внимание на алгоритм формирования нового сценария и последнего сценария. Какие паттерны проектирования были использованы?

from abc import ABC, abstractmethod

from include.enemy import Asteroid, Chicken

SIZE\_ASTEROID = (50, 50)

SIZE\_CHICKEN = (100, 104)

class ScenarioEnemy(ABC):

def \_\_init\_\_(self, width\_window, height\_window):

self.width\_window = width\_window

self.height\_window = height\_window

self.active = 0

self.army = []

@abstractmethod

def is\_last(self):

pass

@abstractmethod

def get\_enemys(self):

pass

def activate(self):

self.active = 1

def remove\_enemy(self, enemy):

for e in enemy:

self.army.remove(e)

def is\_end(self):

if self.army:

return False

return True

class ScenarioAsteroid(ScenarioEnemy):

def is\_last(self):

return False

def next\_scenario(self):

return ScenarioChiken(self.width\_window, self.height\_window)

def get\_enemys(self, count=None):

count\_x = self.width\_window // (SIZE\_ASTEROID[0] + 3)

count\_y = self.height\_window // 2 // (SIZE\_ASTEROID[1] + 6) - 4

for i in range(count\_x):

for j in range(count\_y):

self.army.append(Asteroid(

i \* (SIZE\_ASTEROID[0] + 3) + 25,

j \* (SIZE\_ASTEROID[1] + 3) + 25,

"src/asteroid.png",

SIZE\_ASTEROID

))

return self.army

class ScenarioEndGame(ScenarioEnemy):

def is\_last(self):

return True

def get\_enemys(self):

return []

class ScenarioChiken(ScenarioEnemy):

def is\_last(self):

return False

def next\_scenario(self):

return ScenarioEndGame(self.width\_window, self.height\_window)

def get\_enemys(self):

count\_x = self.width\_window // (SIZE\_CHICKEN[0] + 3)

count\_y = self.height\_window // 2 // (SIZE\_CHICKEN[1] + 3)

for i in range(count\_x):

for j in range(count\_y):

self.army.append(Chicken(

i \* (SIZE\_CHICKEN[0] + 5),

j \* (SIZE\_CHICKEN[1] + 5),

"src/chicken.png",

SIZE\_CHICKEN

))

return self.army

**Контрольные вопросы:**

1. ООП при реализации игровых сценариев.
2. Особенность отрисовки объектов в pygame.
3. Классы pygame.sprite.Sprite и pygame.sprite.Group
4. Игровой цикл, передача события.