1. Создание игрового поля

class MyGame(arcade.Window):  
 # Главный класс игры  
  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_(SCREEN\_WIDTH, SCREEN\_HEIGHT, "School project")  
  
 file\_path = os.path.dirname(os.path.abspath(\_\_file\_\_))  
 os.chdir(file\_path)  
  
 self.frame\_count = 0  
  
 self.game\_over = False  
  
 # Список всех спрайтов  
 self.all\_sprites\_list = None  
 self.ufo\_list = None  
 self.bullet\_list = None  
 self.enemy\_bullet\_list = None  
 self.tank\_life\_list = None  
 self.car\_list = None  
  
 # Очки игрока  
 self.score = 0  
 self.player\_sprite = None  
 self.lives = 3  
  
 self.shape\_list = None  
  
 arcade.set\_background\_color(arcade.color.DARK\_GREEN)

2) Создание не игровых объектов

# растановка домиков в случайном порядке  
for i in range(HOUSE\_COUNT):  
 house\_sprite = arcade.Sprite("images/house.png")  
 house\_sprite.center\_x = random.randrange((SCREEN\_WIDTH // HOUSE\_COUNT) \* i + 60  
 , (SCREEN\_WIDTH // HOUSE\_COUNT) \* (i + 1) - 60)  
 house\_sprite.center\_y = random.randrange(160, SCREEN\_HEIGHT - 280)  
 self.all\_sprites\_list.append(house\_sprite)



# растановка ёлочек в случайном порядке  
for i in range(PAIN\_COUNT):  
 pain\_sprite = arcade.Sprite("images/pine.png")  
 pain\_sprite.center\_x = random.randrange((SCREEN\_WIDTH // PAIN\_COUNT) \* i + 30  
 , (SCREEN\_WIDTH // HOUSE\_COUNT) \* (i + 1) - 30)  
 pain\_sprite.center\_y = random.randrange(120, SCREEN\_HEIGHT - 300)  
 self.all\_sprites\_list.append(pain\_sprite)



3) Создание игровых объектов

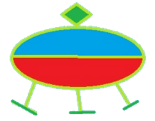
# растановка машинок в случайном порядке  
for i in range(CAR\_COUNT):  
 car\_sprite = CarSprite("images/auto\_00.png", SCALE)  
 car\_sprite.center\_x = random.randrange((SCREEN\_WIDTH // CAR\_COUNT) \* i + 120  
 , (SCREEN\_WIDTH // CAR\_COUNT) \* (i + 1) - 120)  
 car\_sprite.center\_y = random.randrange(65, SCREEN\_HEIGHT - 265)  
 self.all\_sprites\_list.append(car\_sprite)  
 self.car\_list.append(car\_sprite)



# ставим танк игрока  
self.score = 0  
self.player\_sprite = TankSprite("images/tank.png", SCALE)  
self.all\_sprites\_list.append(self.player\_sprite)  
self.lives = 3



# создание вражеских тарелок  
for i in range(STARTING\_UFO\_COUNT):  
 enemy\_sprite = UfoSprite("images/ufo.png", SCALE)  
 enemy\_sprite.guid = "Ufo"  
  
 enemy\_sprite.center\_y = TOP\_LIMIT  
 enemy\_sprite.center\_x = random.randrange(LEFT\_LIMIT, RIGHT\_LIMIT)  
  
 enemy\_sprite.change\_x = random.random() \* 2 - 1  
 enemy\_sprite.change\_y = - random.randrange(1, 4)  
  
 self.all\_sprites\_list.append(enemy\_sprite)  
 self.ufo\_list.append(enemy\_sprite)



4. создание информационных объектов

# иконки жизней игрока  
cur\_pos = 10  
for i in range(self.lives):  
 life = arcade.Sprite("images/tank\_mini.png", SCALE)  
 life.center\_x = cur\_pos + life.width  
 life.center\_y = life.height  
 cur\_pos += life.width  
 self.all\_sprites\_list.append(life)  
 self.tank\_life\_list.append(life)

# написаьт очки игрока на экране  
output = f"Score: {self.score}"  
arcade.draw\_text(output, 10, 70, arcade.color.WHITE, 13)  
  
output = f"Ufo Count: {len(self.ufo\_list)}"  
arcade.draw\_text(output, 10, 50, arcade.color.WHITE, 13)



5. Игровой цикл

а. Расчёт новых позиций игровых объектов

для танка:

def update(self):  
 # Обновление позиции танка  
 self.center\_x += self.change\_x  
 self.center\_y += self.change\_y  
  
 super().update()

для нло:

def update(self):  
 super().update()  
 # Если тарелка выходит за пределы экрана, то появится с другой стороны  
 if self.center\_x < LEFT\_LIMIT:  
 self.center\_x = RIGHT\_LIMIT  
 if self.center\_x > RIGHT\_LIMIT:  
 self.center\_x = LEFT\_LIMIT  
 if self.center\_y > TOP\_LIMIT:  
 self.center\_y = BOTTOM\_LIMIT  
 if self.center\_y < BOTTOM\_LIMIT:  
 self.center\_y = TOP\_LIMIT

для бластера танка:

bullet\_sprite = BulletSprite("images/laserBlue01.png", SCALE)  
bullet\_sprite.guid = "Bullet"  
  
bullet\_speed = 13  
bullet\_sprite.change\_y = math.cos(math.radians(self.player\_sprite.angle)) \* bullet\_speed  
bullet\_sprite.change\_x = -math.sin(math.radians(self.player\_sprite.angle)) \* bullet\_speed  
  
bullet\_sprite.center\_x = self.player\_sprite.center\_x  
bullet\_sprite.center\_y = self.player\_sprite.center\_y  
bullet\_sprite.update()  
  
self.all\_sprites\_list.append(bullet\_sprite)  
self.bullet\_list.append(bullet\_sprite)

для бластера нло:

if self.frame\_count % 20 == 0:  
 for ufo in self.ufo\_list:  
 bullet\_sprite = BulletSprite("images/laserBlue01.png", SCALE)  
 bullet\_sprite.guid = "Bullet UFO"  
  
 bullet\_speed = 8  
 bullet\_sprite.change\_y = -math.sin(math.radians(random.randrange(45, 135))) \* bullet\_speed  
 bullet\_sprite.change\_x = math.cos(math.radians(random.randrange(45, 135))) \* bullet\_speed  
  
 bullet\_sprite.center\_x = ufo.center\_x  
 bullet\_sprite.center\_y = ufo.center\_y  
 bullet\_sprite.update()  
  
 self.all\_sprites\_list.append(bullet\_sprite)  
 self.enemy\_bullet\_list.append(bullet\_sprite)

б. расчёт столкновений объектов

проверка столкновения бластера танка с тарелками

for bullet in self.bullet\_list:  
 self.ufo\_list.use\_spatial\_hash = False  
 ufos\_plain = arcade.check\_for\_collision\_with\_list(bullet, self.ufo\_list)  
 self.ufo\_list.use\_spatial\_hash = True  
 ufos\_spatial = arcade.check\_for\_collision\_with\_list(bullet, self.ufo\_list)  
 if len(ufos\_plain) != len(ufos\_spatial):  
 print("ERROR")  
  
 ufos = ufos\_spatial  
  
 for ufo in ufos:  
 ufo.kill()  
 bullet.kill()

проверка столкновения бластеров тарелок с танком

for bullet in self.enemy\_bullet\_list:  
 player\_shoot = arcade.check\_for\_collision(bullet, self.player\_sprite)  
 if player\_shoot:  
 bullet.kill()  
 if self.lives > 0:  
 self.lives -= 1  
 self.player\_sprite.respawn()  
 self.tank\_life\_list.pop().kill()  
 print("Crash")  
 else:  
 self.game\_over = True  
 print("Game over")

проверка столкновения бластера тарелки с машинами горожан

for bullet in self.enemy\_bullet\_list:  
 cars\_spatial = arcade.check\_for\_collision\_with\_list(bullet, self.car\_list)  
  
 for car in cars\_spatial:  
 bullet.kill()  
 if car.max\_life > 0:  
 buf = "images/auto\_0%d.png" % (4 - car.max\_life)  
 new\_car = CarSprite(buf, SCALE)  
 new\_car.center\_x = car.center\_x  
 new\_car.center\_y = car.center\_y  
 new\_car.max\_life = car.max\_life - 1  
 self.all\_sprites\_list.append(new\_car)  
 self.car\_list.append(new\_car)  
 car.kill()  
 else:  
 car.kill()

проверка столкновения танка игрока с тарелками

if not self.player\_sprite.respawning:  
 ufos = arcade.check\_for\_collision\_with\_list(self.player\_sprite, self.ufo\_list)  
 if len(ufos) > 0:  
 if self.lives > 0:  
 self.lives -= 1  
 self.player\_sprite.respawn()  
 ufos[0].kill()  
 self.tank\_life\_list.pop().kill()  
 print("Crash")  
 else:  
 self.game\_over = True  
 print("Game over")

в. Реагирование на нажатия клавишь

def on\_key\_press(self, symbol, modifiers):  
 # обработчики нажатий клавиш  
  
 # Игрок нажал на пробел - выстрел из танка  
 if not self.player\_sprite.respawning and symbol == arcade.key.SPACE:  
 bullet\_sprite = BulletSprite("images/laserBlue01.png", SCALE)  
 bullet\_sprite.guid = "Bullet"  
  
 bullet\_speed = 13  
 bullet\_sprite.change\_y = math.cos(math.radians(self.player\_sprite.angle)) \* bullet\_speed  
 bullet\_sprite.change\_x = -math.sin(math.radians(self.player\_sprite.angle)) \* bullet\_speed  
  
 bullet\_sprite.center\_x = self.player\_sprite.center\_x  
 bullet\_sprite.center\_y = self.player\_sprite.center\_y  
 bullet\_sprite.update()  
  
 self.all\_sprites\_list.append(bullet\_sprite)  
 self.bullet\_list.append(bullet\_sprite)  
  
 # Игрок нажал влево  
 if symbol == arcade.key.LEFT:  
 self.player\_sprite.change\_x -= 2  
 # игрок нажал вправо  
 elif symbol == arcade.key.RIGHT:  
 self.player\_sprite.change\_x = 2  
  
def on\_key\_release(self, symbol, modifiers):  
 # игрок отжал клавишу влево или вправо - танк остановился  
 if symbol == arcade.key.LEFT:  
 self.player\_sprite.change\_x = 0  
 elif symbol == arcade.key.RIGHT:  
 self.player\_sprite.change\_x = 0

г. Отрисовка игровых объектов и информационных надписей

def on\_draw(self):  
 # отрисовка экрана  
  
 arcade.start\_render()  
  
 # отрисовка всех спрайтов  
 self.all\_sprites\_list.draw()  
  
 # написаьт очки игрока на экране  
 output = f"Score: {self.score}"  
 arcade.draw\_text(output, 10, 70, arcade.color.WHITE, 13)  
  
 output = f"Ufo Count: {len(self.ufo\_list)}"  
 arcade.draw\_text(output, 10, 50, arcade.color.WHITE, 13)