МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технологический университет «СТАНКИН»
(ФГБОУ ВО «МГТУ «СТАНКИН»)

Институт информационных систем и технологий

КАФЕДРА ИНФОРМАЦИОННЫХ СИСТЕМ

Программирование специализированных вычислительных устройств

Отчет по лабораторной работе №4

«Индивидуальный творческий проект»

Выполнил студент гр. ИДБ-21-06 Бабурян А.М.

Проверил Лаверычев М.А.

Москва 2022г.

ОГЛАВЛЕНИЕ

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Цель: закрепление навыков работы с графическими библиотеками и фреймворками.

Задачи:

1. Согласовать индивидуальное творческое задание с преподавателем. Индивидуальное задание должно отвечать следующим требованиям:
    - содержать от двух трехмерных объектов, связанных по смыслу;
    - включать элементы анимации, текстуры, освещения.
2. Используя знания, полученные на предыдущих лабораторных занятиях, написать программный код, реализующий согласованное задание.
3. Составить и защитить отчет по работе.

ПРИМЕНЯЕМЫЕ ТЕХНОЛОГИИ

Язык Python и библиотека Ursina, и встроенные в нее функции, так же модели насекомых, скаченные из интернета. Так же в проекте использовались встроенные текстуры в Ursina, например, «grass». Текстуры и 3d-объекты были взяты из интернета.

Ссылка документацию: https://www.ursinaengine.org/documentation.html

ОПИСАНИЕ ПРОЕКТА

Создан макет игры-шутера, где врагами являются насекомые. Был проанализирован код из официальной документации Ursina, а так же из их GitHub, и взяты и в дальнейшем переработаны некоторые детали для выполнения задачи.

Основной класс в библиотеке Ursina является Entity с помощью его могут созданы практически любые объекты, в него входят такие параметры как расположение, ориентация по осям, модель объекта(может быть установлен объект формата obj), текстура, цвет по шкале rgb и hsv, скорость. Выбор был был сделан из-за реализованной работы с 3d-объектами и удобными решениями для создания простых игр.

Код:

```
# Импортированы необходимые библиотеки:

from ursina import *
from random import randint
from ursina.prefabs.first_person_controller import FirstPersonController
from ursina.shaders import lit_with_shadows_shader

app = Ursina()

# Создание руки:

class Hand(Entity):
	def __init__(self):
		super().__init__(
			parent = camera.ui,
			model = 'texture/arm',
			texture = load_texture('texture/arm_texture.png'),
			scale = 0.2,
			rotation = Vec3(150,-10,0),
			position = Vec2(0.6,-0.6))


editor_camera = EditorCamera(enabled=False, ignore_paused=True)

Entity.default_shader = lit_with_shadows_shader # использование света и теней

# создание игрока
player = \
	FirstPersonController(
	model='cube',
	z=-10,
	color=color.rgb(0, 0, 0, 0), # невидимый кубик игрока
	origin_y=-.5,
	speed=8)

player.collider = BoxCollider(player, Vec3(0,1,0), Vec3(1,2,1))

h = Hand()

# создание пистолета
gun = \
	Entity(
		model="texture/A180_pistol.obj", # выбор модели
		parent=camera.ui, scale=.003,
		color=color.crgb(0, 0, 0, 200) , # установка цвета по шкале HSV
```

```
            position=(0.5, -0.2), # расположение пистолета
            rotation=(-10, -5, 10), # положение пистолета
            on_cooldown=False)

gun.muzzle_flash = Entity(parent=gun, z=1, world_scale=.5, model='quad',
color=color.yellow, enabled=False, position=(-15 , 60))

shootables_parent = Entity()
mouse.traverse_target = shootables_parent

def shoot():
    if not gun.on_cooldown:

            gun.on_cooldown = False
            gun.muzzle_flash.enabled = True

            from ursina.prefabs.ursfx import ursfx

            ursfx([(0.0, 0.0), (0.1, 0.9), (0.15, 0.75), (0.3, 0.14), (0.6, 0.0)],
                volume=0.5, wave='noise',
                pitch=random.uniform(-13,-12),
                pitch_change=-12, speed=3.0)

            invoke(gun.muzzle_flash.disable, delay=.05)
            invoke(setattr, gun, 'on_cooldown', False, delay=.15)
            if mouse.hovered_entity and hasattr(mouse.hovered_entity, 'hp'):
                mouse.hovered_entity.hp -= 0.5
                mouse.hovered_entity.blink(color.red)

def update():
    if held_keys['left mouse']:
            shoot()


class Sky(Entity):
    def __init__(self):
            super().__init__(
                parent = scene,
                model = 'sphere',
                texture = load_texture('texture/skybox.png'),
                scale = 150,
                double_sided = True)

s = Sky()

ground = Entity(
    model='cube',
```

```
        scale=(100, 1, 100),
        color=color.rgb(255, 128, 0),
        texture=«grass", # тектура травы
        texture_scale=(100, 100),
        collider='box')

# Класс Скорпиона

class Scorpions(Entity):

        def __init__(self, **kwargs):

                super().__init__(
                        parent=shootables_parent,
                        model=‘texture/scorpion.OBJ’,
                        texture = 'texture/Map__0_Noise.png',
                        texture_scale=(10, 10),
                        scale_y=0.1, scale_x=0.1, scale_z=0.1,#  маштабирование 3d объекта
                        origin_y=-3,
                        color=color.rgb(150, 60, 55, 200),
                        collider='box',
                        **kwargs)

# создание шкалы здоровья

                self.health_bar = Entity(
                parent=self,
                y=20,
                model='cube',
                color=color.red,
                world_scale=(1.5,.1,.1))
                self.max_hp = 100
                self.hp = self.max_hp

#передвижение объекта при дистанции от игрока меньше 40
        def update(self):
                dist = distance_xz(player.position, self.position)
                if dist > 40:
                        return

                self.health_bar.alpha = max(0, self.health_bar.alpha - time.dt)

                self.look_at_2d(player.position, ‘y')
                hit_info = raycast(self.world_position + Vec3(0,1,0), self.forward, 30,
ignore=(self,))

                if hit_info.entity == player and dist > 2:
```

```
                    self.position += self.forward * time.dt * 15
# обновленит шкалы здоровья
    @property
    def hp(self):
        return self._hp
#Уничтожение объекта при нулевом здоровье
    @hp.setter
    def hp(self, value):
        self._hp = value
        if value <= 0:
            destroy(self)
            return

        self.health_bar.world_scale_x = self.hp / self.max_hp * 1.5
        self.health_bar.alpha = 1

class Wasp(Entity):
    def __init__(self, **kwargs):
        super().__init__(
            parent=shootables_parent,
            model='texture/Wasp.obj',
            texture = ‘texture/Map__0_Noise.png',
            texture_scale=(10, 10),
            scale_y=0.2, scale_x=0.2, scale_z=0.2,
            origin_y=-15,
            collider='box',
            **kwargs)

        self.health_bar = Entity(parent=self, y=20, model='cube', color=color.red,
world_scale=(1.5,.1,.1))
        self.max_hp = 100
        self.hp = self.max_hp

    def update(self):
        dist = distance_xz(player.position, self.position)
        if dist > 40:
            return

        self.health_bar.alpha = max(0, self.health_bar.alpha - time.dt)

        self.look_at_2d(player.position, 'y')
        hit_info = raycast(self.world_position + Vec3(0,1,0), self.forward, 30,
ignore=(self,))

        if hit_info.entity == player and dist > 2:
                    self.position += self.forward * time.dt * 15
```

```
        @property
        def hp(self):
            return self._hp

        @hp.setter
        def hp(self, value):
            self._hp = value
            if value <= 0:
                destroy(self)
                return

            self.health_bar.world_scale_x = self.hp / self.max_hp * 1.5
            self.health_bar.alpha = 1

def pause(key):
    if key == 'tab':
        editor_camera.enabled = not editor_camera.enabled

        player.visible_self = editor_camera.enabled
        player.cursor.enabled = not editor_camera.enabled
        gun.enabled = not editor_camera.enabled
        mouse.locked = not editor_camera.enabled
        editor_camera.position = (player.position.x - 1, player.position.y + 3,
player.position.z - 1)

        application.paused = editor_camera.enabled


p = Entity(ignore_paused=True, input=pause)

en = [Scorpions(x=randint(0, 50)) for i in range(3)]

e2 = [Wasp(x=randint(30, 70)) for _ in range(3)]

app.run()
```

РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ЗАДАНИЯ

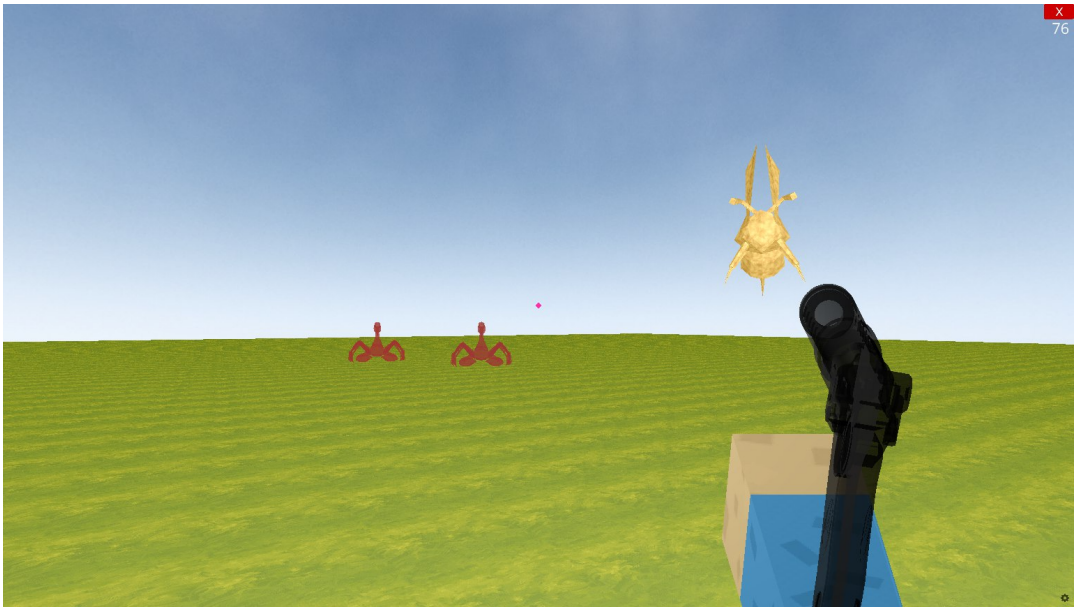

Рис. 2 Результат выполнения задания. Игровой режим

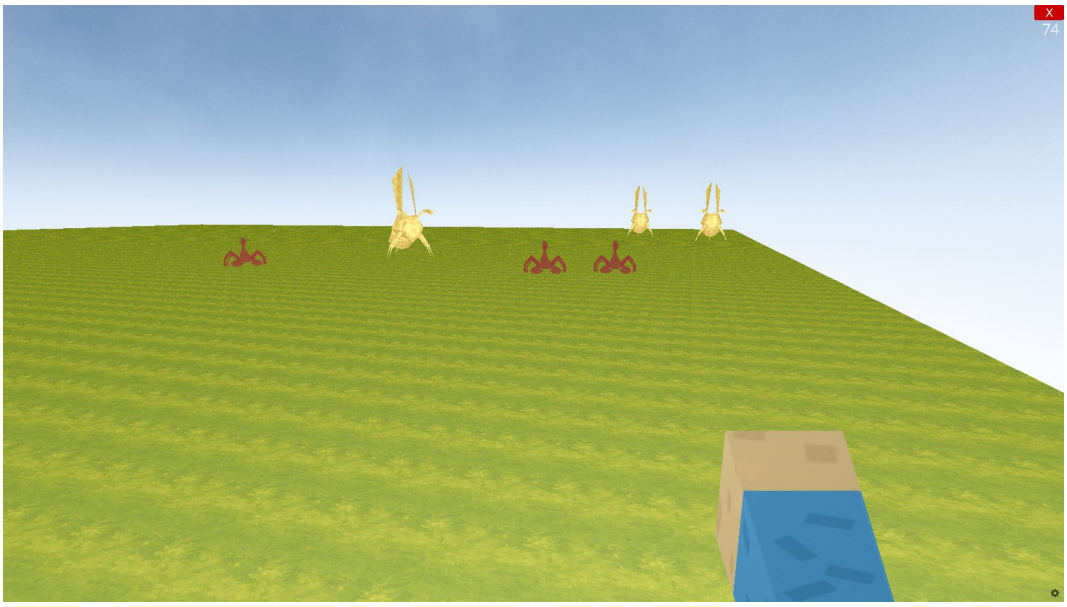

Рис. 1 Результат выполнения задания. Пауза