Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет информационных технологий и управления

Кафедра вычислительных методов и программирования

ОТЧЕТ

по лабораторной работе

**«Прыжки»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 924403 |  | К.Д. САНДАРОВИЧ |
| Проверил |  | С.О. ЛУКАШОВ |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Минск 2022

# Общая постановка задачи

## 1. Краткие теоретические сведения

Если в разрабатываемой игре убегающий игрок способен перепрыгивать через препятствия, то в этом случае противники должны обладать достаточным умом, чтобы прыгать и рассчитывать параметры своего прыжка.

2. Системные подходы к решению задачи

Определим базовый алгоритм выбора скорости и введём понятие площадок для прыжка и приземления, чтобы вычислить скорость, необходимую для перемещения между ними.

3. Разбор решения поставленной задачи

Агенты должны иметь тег Agent, а главный объект должен иметь компонент Collider, настроенный как триггер. В зависимости от особенностей игры к агенту или к площадкам потребуется присоединить компонент Rigidbody.

1. Реализуем модель поведения VelocityMatch.

2. Определим класс прыжка Jump, наследующий класс VelocityMatch и добавляющий свои свойства.

3. Реализуем метод Isolate. Он отключает все модели поведения объекта, за исключением компонента Jump.

4. Определим функцию, реализующую прыжок с использованием модели снаряда, рассмотренной ранее.

5. Реализуем метод для выбора скорости в зависимости от цели.

6. Реализуем функцию вычисления времени.

7. Переопределим метод Awake. Самым важным здесь является кэширование ссылок на другие присоединённые модели поведения, что придаёт смысл функции Isolate.

8. Переопределим метод GetSteering.

Алгоритм, основываясь на скорости агента, определяет, сможет ли агент допрыгнуть до площадки приземления. Задаваемая цель определяет, будет ли выполнен прыжок, и если агент способен его совершить, будет сделана попытка подбора вертикальной скорости для приземления на целевую площадку.

# Блок-схема алгоритма

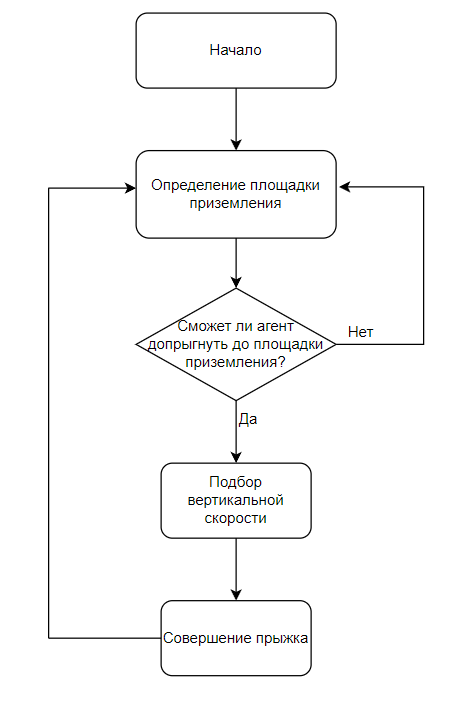


Рисунок 1 – блок-схема алгоритма прыжка

# Листинг

using UnityEngine;

using System.Collections.Generic;

public class Jump:VelocityMatch

(

public JumpPoint jumpPoint;

//Признaк допустимости прыжка

bool canAchieve = false;

/ /Максимальная вертикальная скорость при прыжке

public float maxYVelocity;

public VectorЗ gravi ty = new Vector3 {0,-9.Bf,0);

private Projectile projectile;

private List<AgentBehaviour> behaviours;

}

Public void Isolate(bool state)

{

Foreach (AgentBehaviour b in behaviours)

b.enabled = !state;

this.enabled = state;

}

Public void DoJump()

{

Projectile.enabled = true;

Vector3 direction;

Direction = Projectile.GetFireDirection(jumpPoint.jumpLocation,

Jump.Point.landingLocation, agent.maxSpeed);

Projectile.Set(jumpPoint.jumoLocation, direction,

Agent.maxSpeed, false);

}

Public override void Awake()

{

Base.Awake();

This.enabled = false;

Projectile = gameObject.AddComponent<Projectile>();

Behaviours = new List<AgentBehaviour>();

AgentBehaviour() abs;

Abs = gameObject.GetComponents<AgentBehaviour>();

Foreach (AgentBehaviour b in abs)

{

If (b == this)

Continue;

Behaviours.Add(b);

}

}

# Вывод

В процессе выполнения лабораторной работы были реализована системы прыжков с реализацией площадок для прыжка и приземления.