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

**«Выбор с помошью дерева принятия решений»**

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

Минск 2022

# Общая постановка задачи

## 1. Краткие теоретические сведения

Деревья принятия решений считаются одним из простейших механизмов благодаря наглядности и простоте реализации. Как следствие сегодня это один из наиболее часто используемых методов. Он широко используется в других областях управления персонажами, таких как анимация.

2. Системные подходы к решению задачи

Этот рецепт требует хорошего знания рекурсии и наследования, поскольку в его реализации используются виртуальные функции.

3. Разбор решения поставленной задачи

Реализация этого рецепта требует большой внимательности из-за большого количества обрабатываемых файлов. Здесь мы создадим родительский класс DecisionTreeNode, который наследуют все остальные. Далее рассмотрим реализацию нескольких узлов для стандартных решений:

1.Во-первых, создадим родительский класс DecisionTreeNode.

2.Создадим псевдоабстрактный класс Decision, наследующий родительский класс DecisionTreeNode.

3. Определим псевдоабстрактный класс Action.

4. Реализуем виртуальную функцию LateUpdate.

5. Определим законченный класс DecisionTree.

6. Переопределим функцию MakeDecision.

7. Наконец, реализуем функцию Update.

Выбор пути в узлах дерева решений выполняется вызовом рекурсивной функции MakeDecision. Элементы верхнего уровня дерева являются решениями, а нижнего уровня - действиями. При создании дерева следует проявить особую осторожность, чтобы в нем не образовались циклы.

# Блок-схема алгоритма

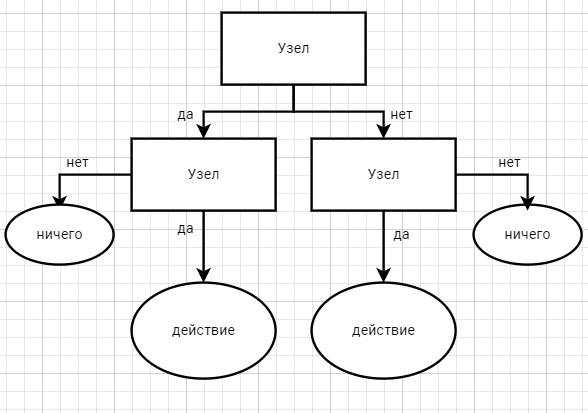


Рисунок 1 – Дерево решений.

# Листинг

using UnityEngine;

using System . Collections;

public class DecisionTreeNode : MonoBehaviour

{

public virtual DecisionTreeNode MakeDecision {)

{

return null;

}

}

using UnityEngine;

using System. Collections;

public class Decision : DecisionTreeNode

{

public Action nodeTrue ;

public Action nodeFalse;

public virtual Action GetBranch {)

{

return null;

}

}

using UnityEngine;

using System . Collections;

public class Action : DecisionTreeNode

{

public bool activated = false;

puЬlic override DecisionTreeNode MakeDecision ()

{

return this;

}

}

public virtual void LateUpdate ()

(

if ( ! activated)

return;

\*\*

}

using UnityEngine ;

using System. Collections;

puЫic class DecisionTree : DecisionTreeNode

{

public DecisionTreeNode root;

private Action actionNew;

private Action actionOld;

}

public override DecisionTreeNode makeDecision ()

(

return root. МakeDecision ( ) ;

void Update ()

(

actionNew . activated = false ;

actionOld = actionNew;

actionNew = root . MakeDecision () as Action ;

if (actionNew == null)

act ionNew = actionOld;

actionNew . act ivated = true ;

# Вывод

В процессе выполнения лабораторной работы были реализованы модели поведения для исскуственного интеллекта на Unity такие как «принятие решений с помощью древа».