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

**«СТРЕЛЬБА»**

Студент гр. 924403 К.Д. САНДАРОВИЧ

Проверил С.О. ЛУКАШОВ

Минск 2022

# Общая постановка задачи

## Краткие теоретические сведения

Если в игре используется такая механика как стрельба, то в этом случае важно прописать снаряды как обьекты, на которые воздействует сила тяжести, поскольку дает возможность предсказать место падения снаряда или точно послать снаряд к заданной цели.

## Системные подходы к решению задачи

Этот рецепт несколько отличается от прочих, поскольку в нем не применяется базовый класс AgentBehaviour. Для алгоритмов GetLandingTime, GetLandingPos и GetFireDirection необходимо создать отдельный класс снаряда Projectile и не забыть настроить порядок их выполнения.

## Разбор решения поставленной задачи

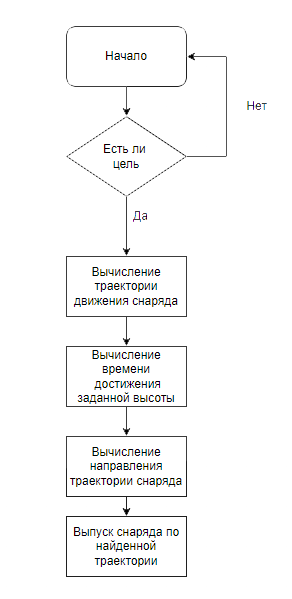
Прежде чем пытаться определить место падения, важно узнать время, оставшееся до падения (или до достижения определенной позиции). Поэтому вместо создания новой модели поведения внесем изменения в класс снаряда Projectile.

* 1. Создадим класс снаряда ProjecHle со свойствами, которые обрабатываются физической системой.
  2. Определим функцию Update.
  3. Добавим функцию Set, реализующую метание игрового объекта (например, после инициализации сцены).
  4. Добавим функцию GetLandingTime для вычисления времени падения**.**
  5. Добавим функцию GetLandingPos для вычисления предположительного места падения.
  6. Определим функцию GetFireDirection.
  7. Решим квадратное уравнение и, если можно послать снаряд в соответствии с заданными параметрами, вернуть ненулевой вектор направления.

**Как это работает**

Модель Projectile вычисляет параболлическую траекторию движения, опираясь на законы физики, изучаемые в средней школе. В GetLandingTime и GetLandingPos решается уравнение из предыдущего рецепта для фиксированной высоты. Зная текущую позицию снаряда и его скорость, можно получить время, за которое снаряд достигнет заданной высоты. Подставляя заданную фиксированную скорость, мы решаем квадратное уравнение и получаем направление (при наличии хотя бы одного значения времени), которое не нуждается в нормализации, поскольку уже было нормализовано при настройке. снаряда

# Блок-схема алгоритма

****

Рисуно**к** 1 — Блок схема алгоритма «стрельба»

# Листинг

usinq UnityEnqine ;

using System.Collections;

public class Projectile : MonoBehaviour

{

private bool set = false;

private VectorЗ firePos;

private VectorЗ direction;

private float speed;

private float timeElapsed;

}

void Update ()

{

if (!set)

return ;

timeElapsed += Time . deltaTime ;

transform.position = firePos + direction \* speed \* timeElapsed;

transform.position += Physics. gravity \*

(timeElapsed \* timeElapsed) / 2.0f;

// дополнительные проверки для очистки сцены

if (transform.position .y < -1 . 0f)

Destroy (this.gameObject) ; // или set = false; и скрыть его

}

public void Set (VectorЗ firePos, VectorЗ direction, float speed)

{

this.firePos = firePos;

this.direction = direction.normalized;

this.speed = speed;

transform.position = firePos;

set = true;

}

public float GetLandingТime (float height = O.Of)

{

VectorЗ position = transform.position;

float time = O.Of;

fioat valueint = (direction.y \* direction.y) \* (speed \* speed) ;

valueint = valueint - (Physics.gravity.y \* 2 \*

(position.y - height) );

valueint = Mathf.Sqrt (valueint) ;

float valueAdd = (-direction.y) \* speed;

fioat valueSub z (-direction.y) \* speed;

valueAdd = (valueAdd + valueint) / Physics.gravity.y;

valueSub • (valueSub - valueint) / Physics. gravity.y;

if (fioat.IsNaN (valueAdd) && ! fioat . IsNaN (valueSub) )

return valueSub;

else if ( ! fioat.IsNaN (valueAdd) && fioat. IsNaN (valueSub) )

return valueAdd;

else if (float. IsNaN (valueAdd) && float. IsNaN (valueSuЬ) )

return -1.0f;

time = мathf.Max (valueAdd, valueSub) ;

return time;

}

public VectorЗ GetLandingPos (float height • O.Of)

{

Vector3 landingPos = VectorЗ.zero;

float time = GetLandingTime( );

if (time < O.Of)

return landingPos;

landingPos.y = height;

landingPos.x = firePos.x + direction.x \* speed \* time ;

landingPos.z = firePos.z + direction.z \* speed \* time ;

return landingPos;

}

public static VectorЗ GetFireDirection (VectorЗ startPos, VectorЗ endPos, fioat speed)

{

// тело функции

}

VectorЗ direction = VectorЗ . zero;

VectorЗ delta = endPos - startPos;

float а = VectorЗ.Dot (Physics.gravity, Physics. gravity) ;

float Ь = -4 \* (VectorЗ. Dot (Physics.gravity, delta) + speed • speed) ;

float с= 4 \* VectorЗ.Dot (delta, delta) ;

if (4 \* а \* с > Ь \* Ь)

return direction ;

float timeO = мathf.Sqrt ( (-b + Mathf.Sqrt (b \* Ь - 4 \*а\* с)) / (2\*а));

float timel = Mathf.Sqrt ( (-b - Mathf.Sqrt (b \* Ь - 4 \*а\* с)) / (2\*а) );

float time;

if (timeO < O.Of)

{

if (time1 < 0)

return direction;

time = timel;

}

Else

{

if (time1 < 0)

time = timeO;

else

time = Мathf.Min (timeO, timel) ;

}

direction = 2 • delta - Physics.gravity • (time • time) ;

direction = direction / (2 \* speed \* time) ;

return direction;

**Вывод**

В процессе выполнения лабораторной работы были реализованы системы стрельбы.