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

**«УКЛОНЕНИЕ ОТ АГЕНТОВ И СТЕН»**

Студент гр. 924403 К.Д. САНДАРОВИЧ

Проверил С.О. ЛУКАШОВ

Минск 2022

**Общая постановка задачи**

**1 Краткие теоретические сведения**

При моделировании толпы в играх поведение агентов, аналогичное поведению физических частиц, выглядит неестественно. Цель этого рецепта – реализовать уклонение агента от столкновений с себе подобным во время движения.

Аналогичная ситуация с рецептом уклонения от стен: необходимо реализовать модель уклонения от столкновений со стенами, которая учитывает безопасное расстояние до них и обеспечивает удаление от поверхностей при чрезмерном приближении.

**2 Системные подходы к решению задачи**

Для алгоритмов AvoidAgent и AvoidWall необходимо создать отдельные файлы и не забыть настроить порядок их выполнения.

**3 Разбор решения поставленной задачи**

При реализации этих моделей поведения используется один и тот же подход, но они содержат разные свойства и производят разные расчёты в начальной части функции GetSteering.

Уклонение от агентов:

- необходимо создать модель AvoidAgent с радиусом, определяющим область столкновения, и список агентов для уклонения;

- далее создаётся функция Start, заполняющую список агентов по тегу, созданному ранее;

- определяем функции GetSteering;

- добавляем переменные для вычисления расстояния и скорости агентов, находящихся поблизости;

- находим ближайшего агента, с которым может столкнуться текущий;

- если такой агент имеется, пытаемся его обойти.

Уклонение от стен:

- создаём модель AvoidWall, наследующую Seek;

- добавляем свойства для определения безопасного удаления и длины отбрасываемого луча;

- добавим свойства для определения безопасного удаления и длины отбрасываемого луча;

- определяем функцию Awake для настройки целевого объекта;

- определяем функцию GetSteering;

- внутри функции объявляем и инициализируем переменные, необходимые для отбрасывания луча;

- отбрасываем луч и выполняем необходимые расчёты, если он пересекается со стеной.

**Как это работает**

**Уклонение от агентов**:из списка агентов извлекается ближайший, и если он находится достаточно близко, текущий агент покидает маршрут, чтобы избежать столкновения, при этом учитывается текущая скорость встречного агента.

**Уклонение от стен**: по направлению движения агента отбрасывается луч. Если луч пересекается со стеной, целевой объект перемещается в новую позицию, с учётом расстояния до стены и заданного безопасного удаления, при этом выполнение самого перемещения передаётся по модели Seek. В результате создаётся видимость уклонения от стен.

Модель поведения можно улучшить, добавив дополнительные лучи и расположив их в виде усов. Кроме того, эта модель поведения обычно используется вместе с другими моделями, такими как Pursue, путём смешивания.

**Блок-схема алгоритма**

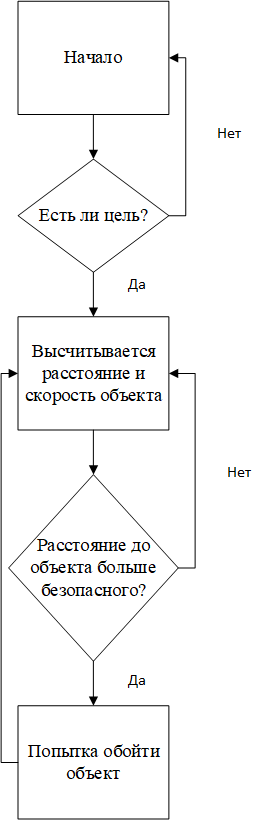
****

Рисунок 1 – Блок-схема алгоритма «уклонение от агентов»

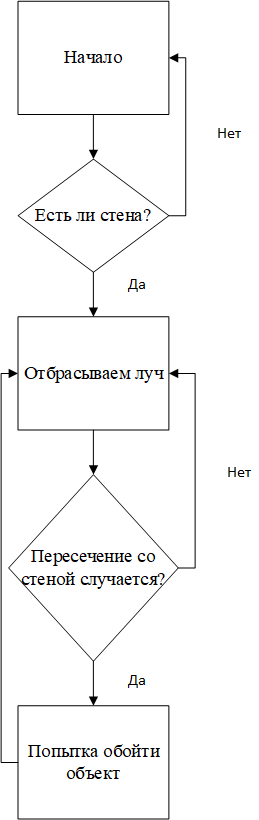


Рисунок 2 – Блок-схема алгоритма «уклонение от стен»

**Листинг кода**

Модель поведения «уклонение от агентов»

public class AvoidAgent : AgentBehaviour

{

public float collisionRadius = 0.4f;

GameObject[] targets;

void Start()

{

targets = GameObject.FindGameObjectsWithTag(“Agent”);

}

public override Steering GetSteering()

{

Steering steering = new Steering();

float shortestTime = Mathf.Infinity;

GameObject firstTarget = null;

float firstMinSeparation = 0.0f;

float firstDistance = 0.0f;

VectorЗ firstRelativePos = VectorЗ. zero;

VectorЗ firstRelativeVel = VectorЗ. zero;

foreach (GameObject t in targets)

{

Vector3 relativePos;

Agent targetAgent = t.GetComponent<Agent>();

relativePos = t.transform.position – transform.position;

VectorЗ relativeVel = tarqetAqent. velocity - aqent. velocity;

float relativeSpeed = relativeVel. maqnitude ;

float timeToCollision = VectorЗ. Dot (relativePos, relativeVel);

timeToCollision /= relativeSpeed \* relativeSpeed \* - 1;

float distance = relativePos. maqnitude;

float minSeparation = distance - relativeSpeed \* timeToCollision;

if (minSeparation > 2 \* collisionRadius)

continue;

if (timeToCollision > O.Of && timeToCollision < shortestTime)

{

shortestTime = timeToCollision ;

firstTarqet = t;

firstMinSeparation = minSeparation;

firstRelativePos = relativePos;

firstRelativeVel = relativeVel;

}

}

if (firstTarqet == null)

return steering;

if (firstMinSeparation <= О. Of 11 firstDistance < 2 \* collisionRadius)

firstRelativePos = firstTarget. transform. position;

else

firstRelativePos += firstRelativeVel \* shortestTime;

firstRelativePos. Normalize( );

steering.linear = -firstRelativePos \* agent.maxAccel;

return steering;

}

Модель поведения «уход от погони»

public class AvoidWall : Seek

{

public float avoidDistance;

public float lookAhead;

public override void Awake()

{

base.Awake();

target = new GameObject();

}

Public override Steering GetSteering()

{

Steering steering = new Steering();

Vector3 position = transform.position;

Vector3 rayVector = agent.velocity.normalized \* lookAhead;

Vector3 direction = rayVector;

RaycastHit hit;

if (Physics.Raycast (position, direction, out hit, lookAhead))

{

position = hit.point + hit.normal \* avoidDistance;

target.transform.position = position;

steering = base.GetSteering();

}

return steering;

}

}

**Выводы**

В процессе выполнения лабораторной работы были реализованы модели поведения для искусственного интеллекта на Unity такие как «уклонение от агентов» и «уклонение от стен».