УДК 004

Создание ИИ для противников с помощью Unity 3D и C#

Ульянов Егор Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье рассматривается и описывается создание простого искусственного интеллекта для компьютерных противников. ИИ будет разрабатываться на языке программирования C# с помощью игрового движка Unity 3D. Практическим результатом является разработанный искусственный интеллект.
Ключевые слова: ИИ, C#, Unity 3D

Create AI for opponents with Unity 3D and C#

Ulianov Egor Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article discusses and describes the creation of a simple artificial intelligence for computer opponents. The AI will be developed in the C# programming language using the Unity 3D game engine. The practical result is the developed artificial intelligence.
Keywords: AI, C#, Unity 3D

Во многих играх искусственный интеллект (ИИ) часто необходим для взаимодействия с игроком. В некоторых сценариях ИИ должен помогать, а в других быть противником. ИИ может иметь простое или сложное поведение, в зависимости от требований проекта. Требования могут включать необходимость вести дипломатические переговоры с игроком или просто ходить назад, вперед по платформе. Как бы то ни было, важно создать ИИ, который хорошо выполняет свою работу.

Цель данной статьи рассмотреть возможности игрового движка Unity 3D для создании искусственного интеллекта.

И. А. Савин, О. В. Батенькина рассмотрели процесс написания скриптовых сценариев при разработке виртуального тренажера[1]. В своей работе Э. Р. Гараева, И. И. Бикмуллина, И. А. Барков описали возможности Unity 3D на предмет создания 3D-моделей[2]. С. А. Суродин в своей статье представил сценарий углубленного изучения одного из лучших движков, существующих на данный момент, для создания красивых 2D и 3D игр[3]. В своей работе Р. Ф. Гайнуллин, В. А. Захаров, Е. А. Аксенова изучили

инструмент для разработки двух- и трёхмерных игр – Unity 3D[4]. А. Ю. Субботина, Н. И. Хохлов в результате их работы была предложена трехмерная версия игры Конвея «Жизнь» [5].

Создаем 3D проект и называем его. Далее понадобиться создать структуру папок, а именно папку материалов, и папку для скриптов см. рисунок 1.

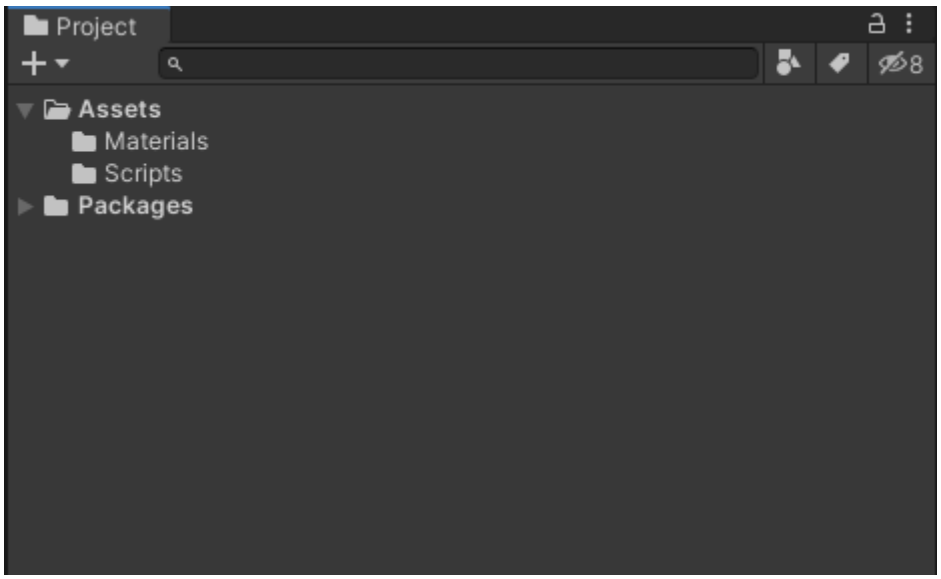

Рис. 1. Создание структуры папок

Затем создаем пол, на котором будут стоять все объекты. Для этого в окне «Hierarchy» выбираем «Create-> 3D Object-> Plane». Называем объект «Floor», где в поле «Scale» X=7, а Z=3 см. рисунок 2.

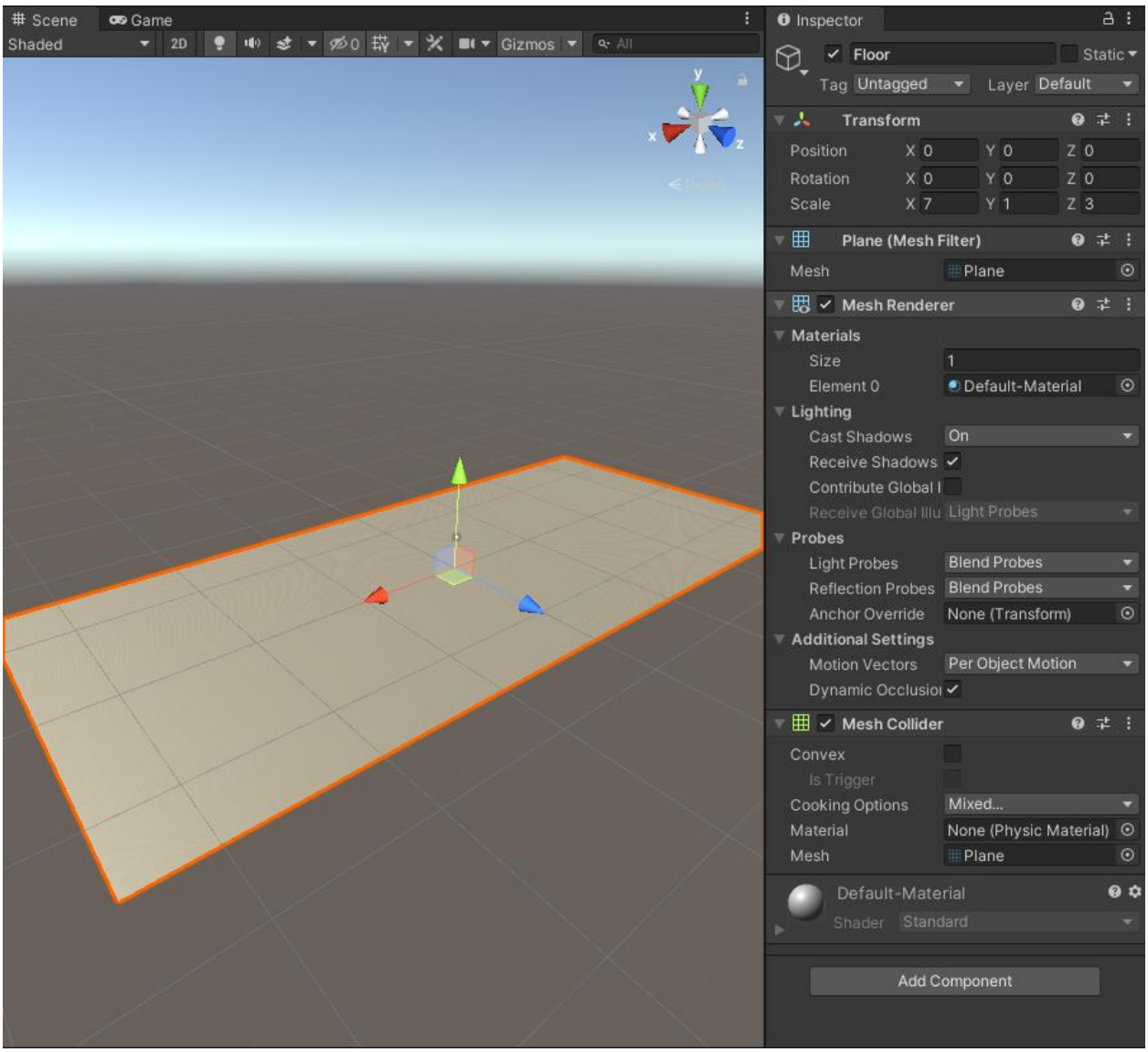

Рис. 2. Создание и настройка пола

Теперь нужно создать новый материал, чтобы отличать пол от других объектов, которые будут размещены на сцене. В папке «Materials» в окне «Assets» создаем новый материал, щелкнув правой кнопкой мыши в окне «Assets» и выбрав «Create-> Material». Ставим любой цвет и средством «Drag and Drop» меняем цвет пола см. рисунок 3.

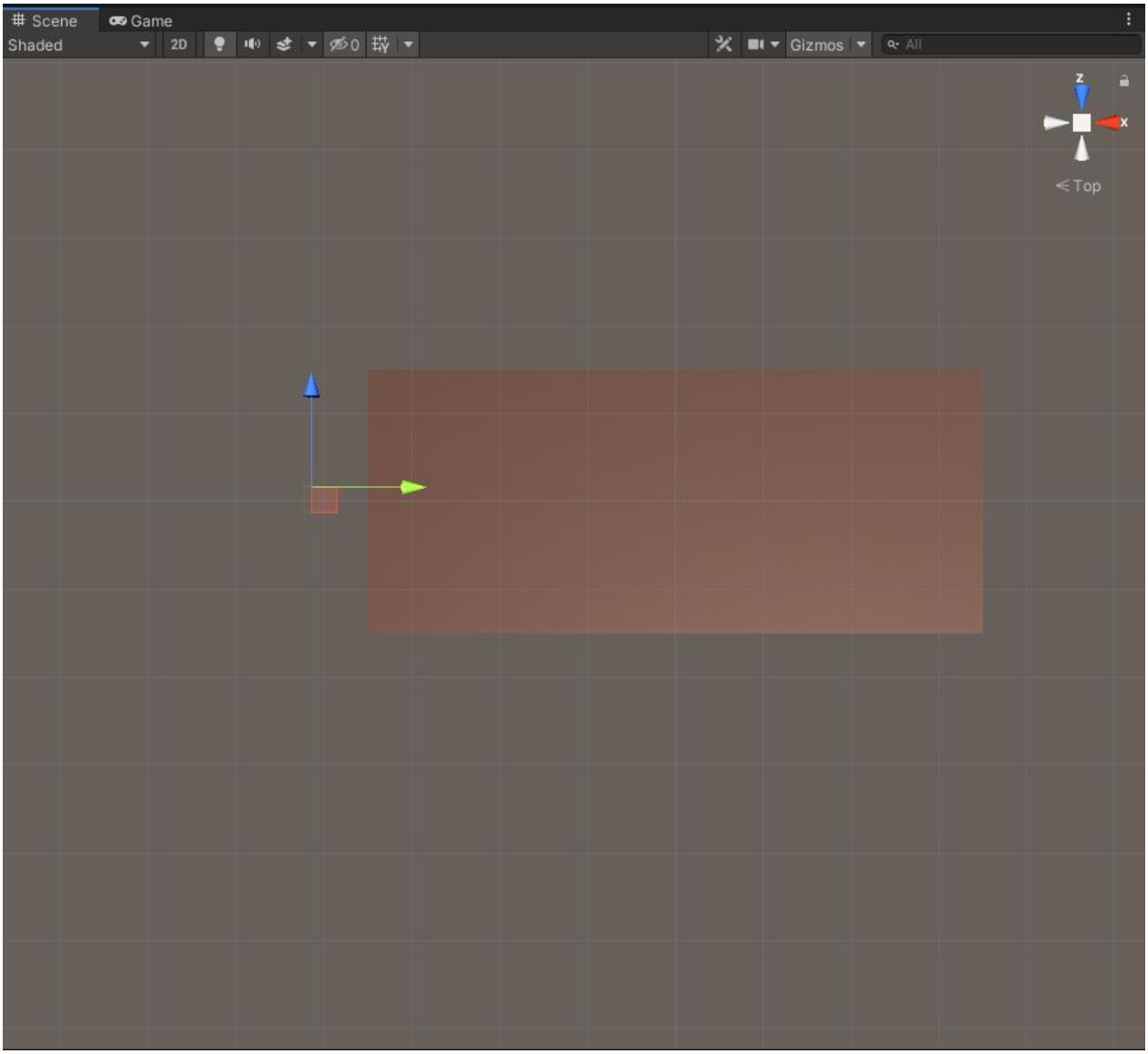

Рис. 3. Настройка цвета пола

Далее необходимо создать 4 стены, чтобы игрок не упал. Создаем объект «Plane», называем «Wall» и ставим аналогичный полу масштаб. Дублируем стену и расставляем см. рисунок 4.

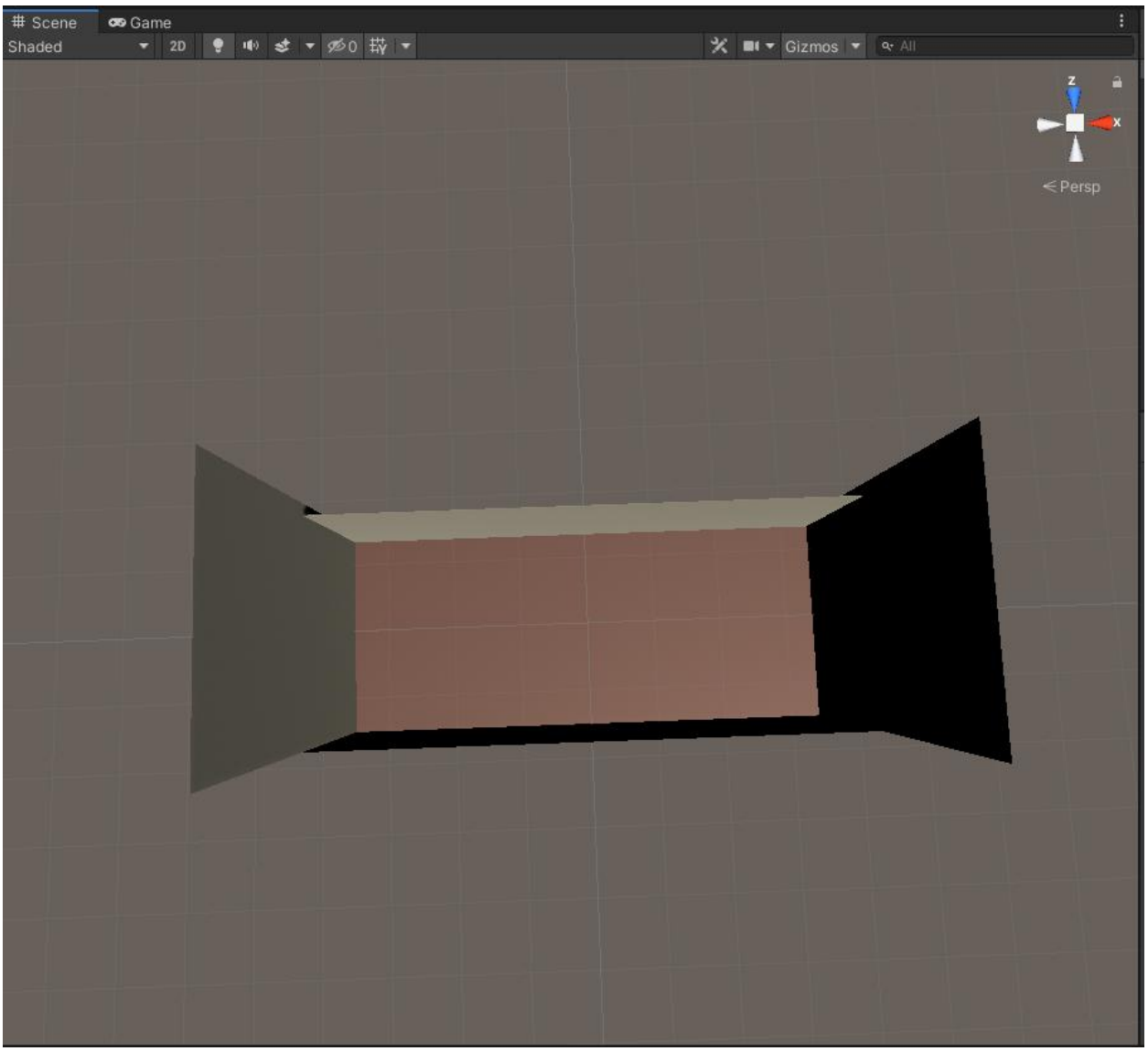

Рис. 4. Расстановка стен

Сцена настроена, значит, создаем персонажа. Для этого создаем объект «*Sphere»*, называем этот объект «*Player*». Для придания персонажу физики добавляем компонент «Rigidbody», также для сферы понадобиться тэг «Player» см. рисунок 5.

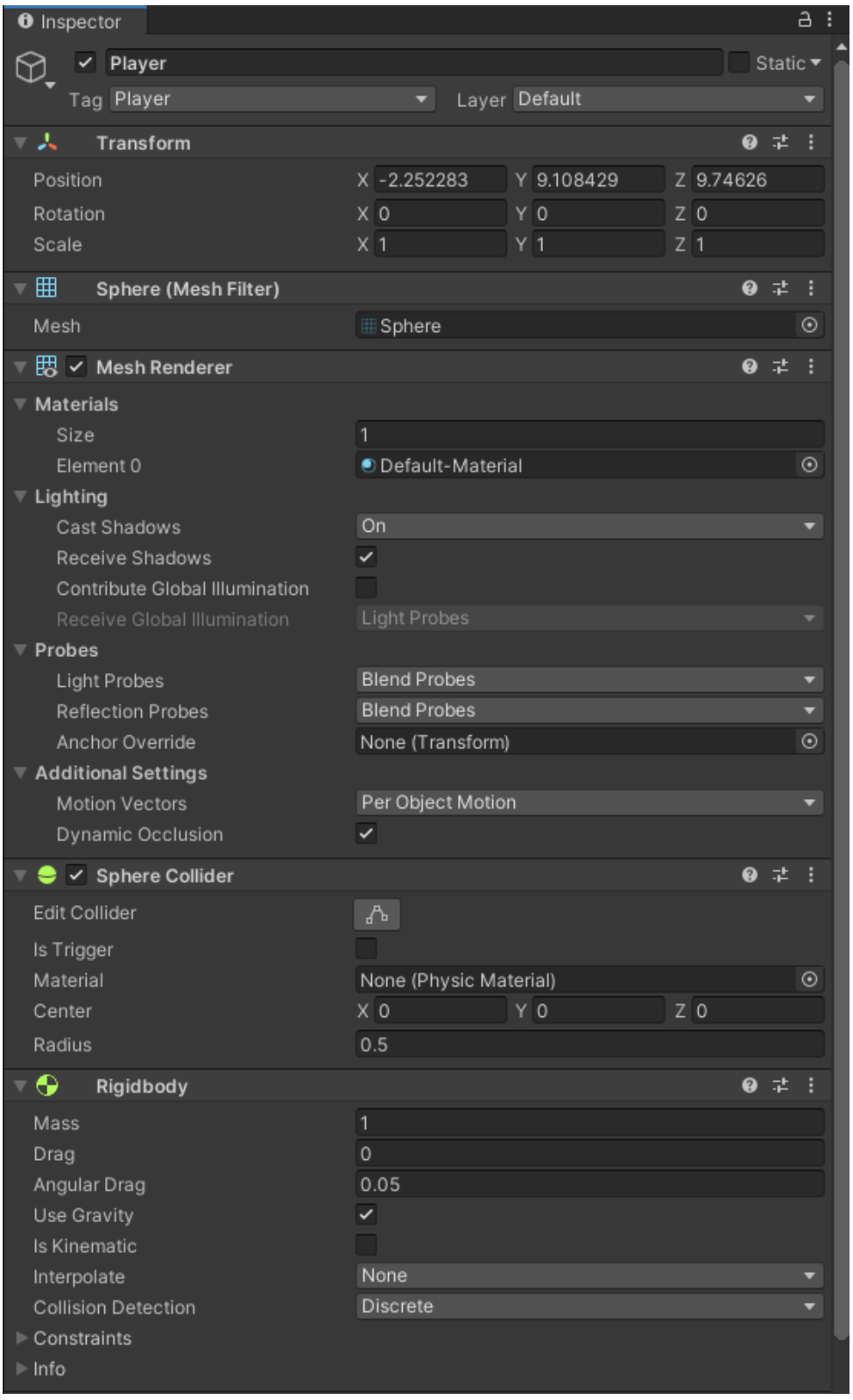

Рис. 5. Настройка персонажа

Теперь необходимо создать стража, для этого создаем объект «Cube» называем «Guard», добавляем физику «Rigidbody» и компонент «NavMesh Agent» см. рисунок 6.

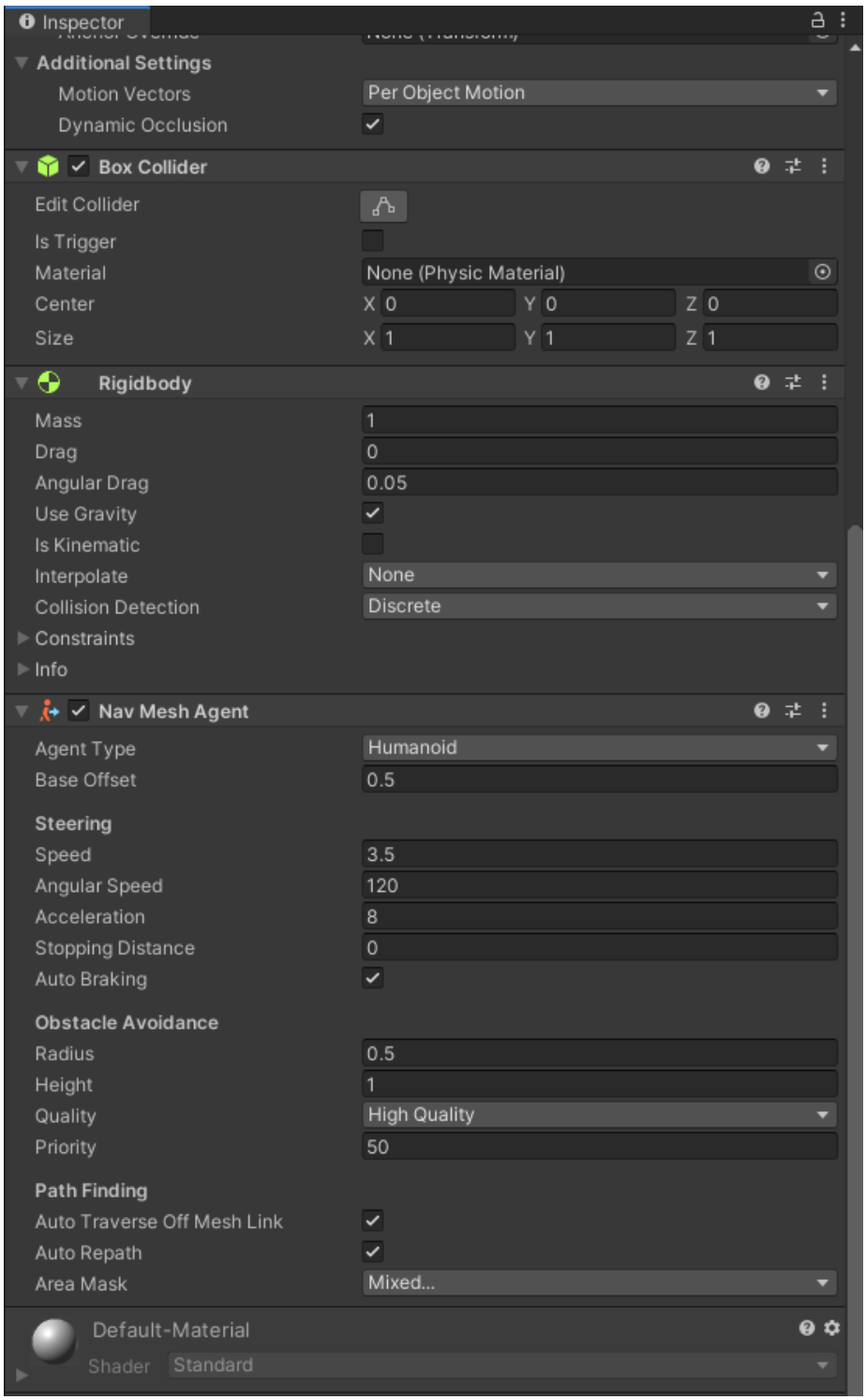

Рис. 6. Добавление стража

Далее, понадобится триггер, который будет действовать как «глаза» «Guard», объект будет предупреждать «Guard» всякий раз, когда игрок касается его. Создаем объект «Sphere», называем его «Looker». Изменяем размер сферы на X = 9; Y = 0,5; Z = 9. Далее помещаем объект «Looker» так, чтобы он находился в средней части пола, также можно окрасить сферу в красный цвет, чтобы придать опасности данному участку см. рисунок 7.

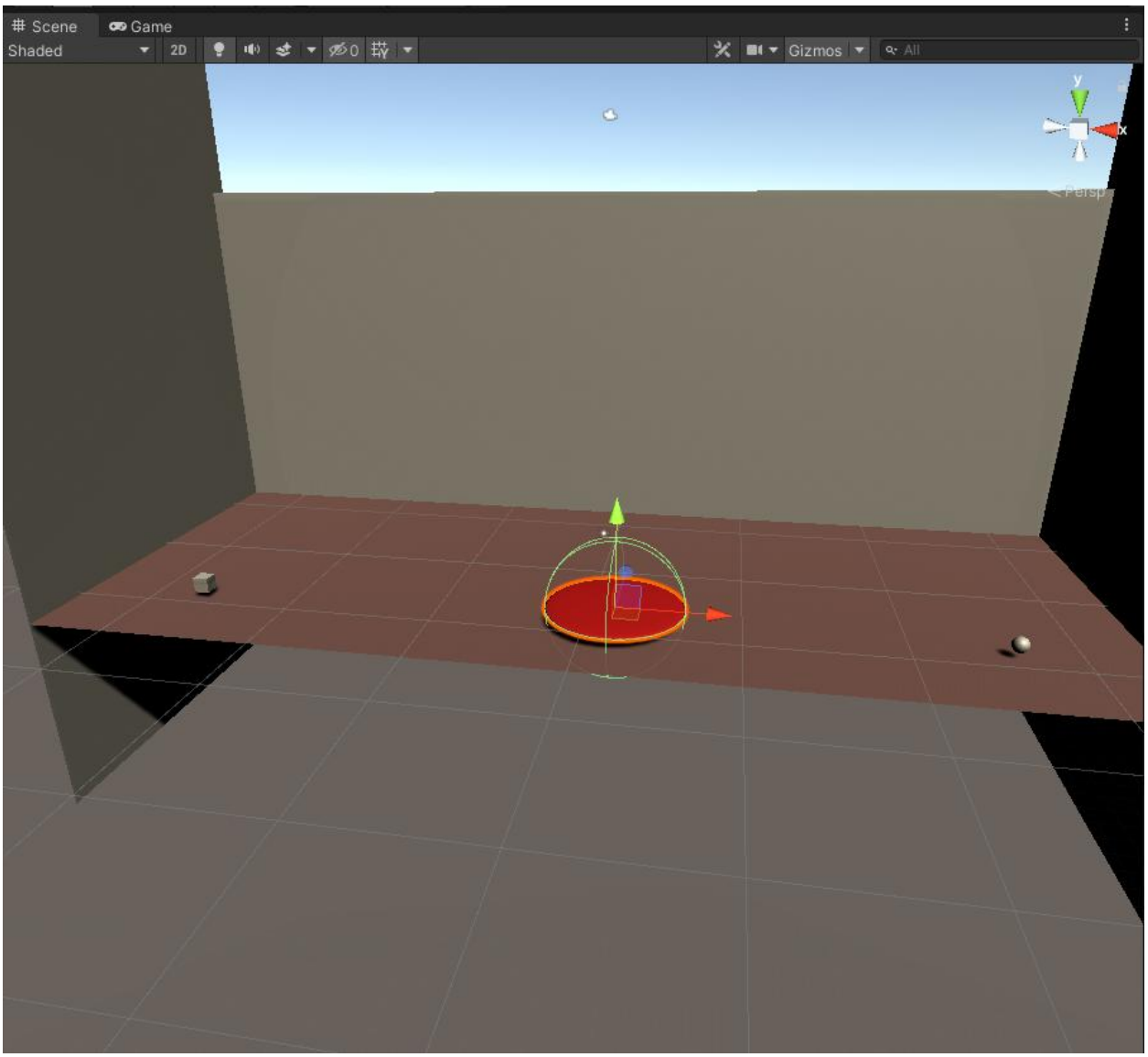

Рис. 7. «Глаза» стражника

Теперь осталась единственная задача - создать навигационную сетку, по которой Страж сможет перемещаться. Открываем окно «Navigation», ставим галочку на «Navigation Static». Затем выбираем опцию «Bake» и подтверждаем.

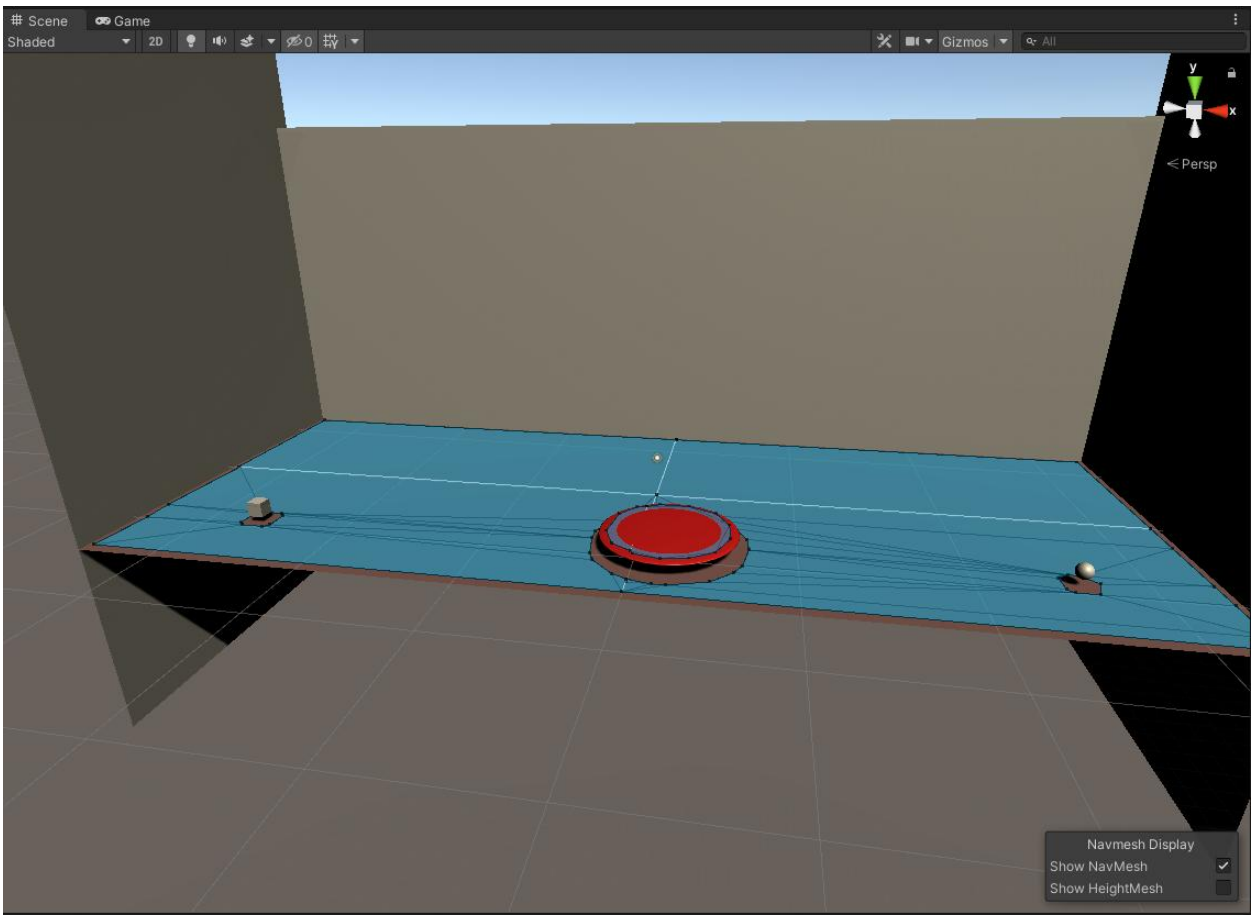

Рис. 8. Настройка навигационной сетки

Приступаем к написанию кода, создаем три скрипта «Player», «Guard» и «Looker». Открываем первый и программируем движения игрока, на стрелочки см. рисунок 9.

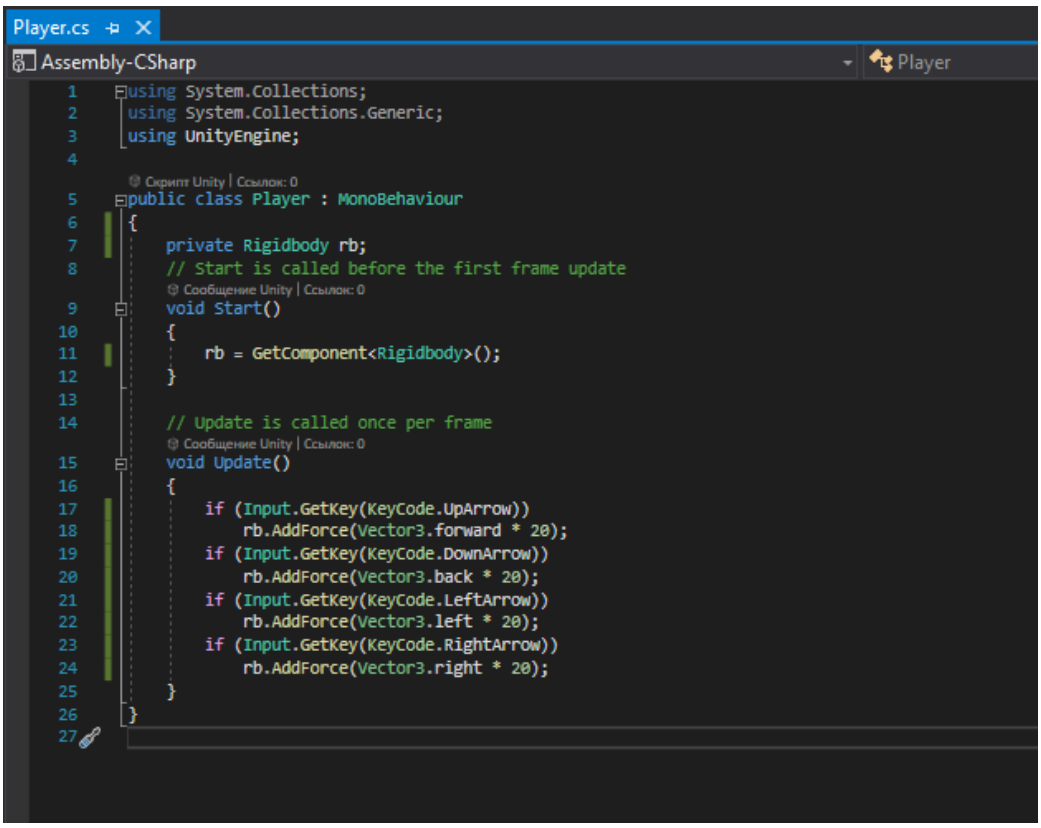

Рис. 9. Скрипт движения игрока

Напишем код стража чтобы, когда игрок попадался на «глаза», стражник начинал преследование см. рисунок 10.

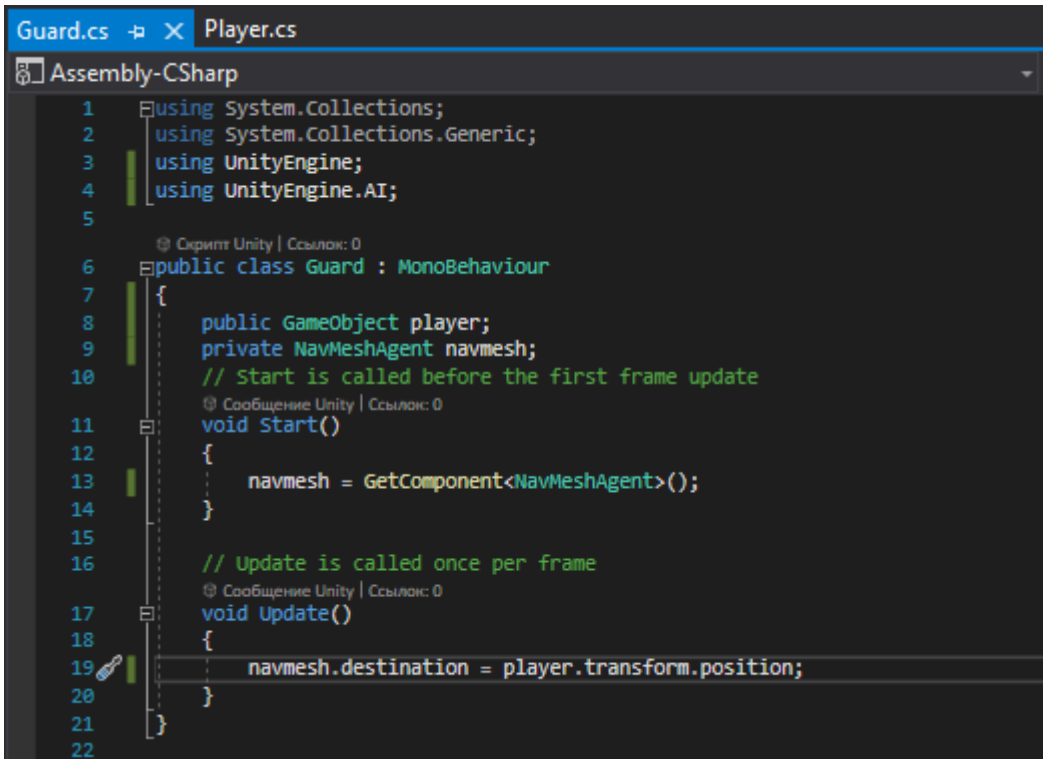

Рис. 10. Скрипт стража

Наконец дописываем код «глаза» см. рисунок 11.

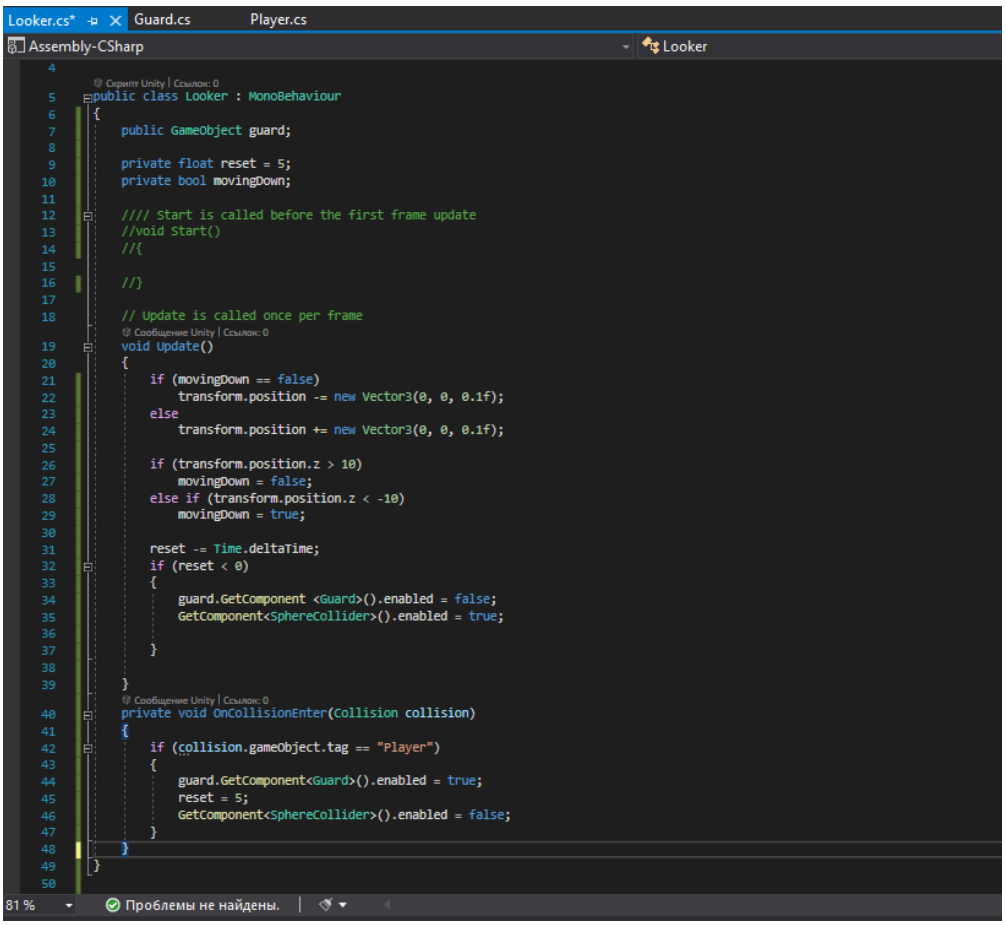

Рис. 11. Скрипт «глаз» стражника

Далее прикрепляем скрипты к объектам и проверяем работу ИИ см. рисунок 12-13.

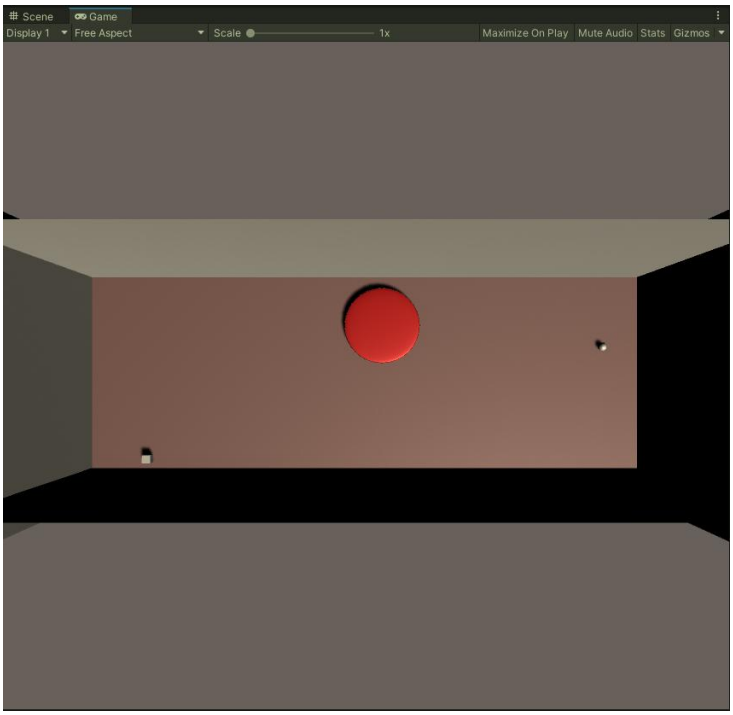

Рис. 12. Действия стража до попадания игрока в поле зрения

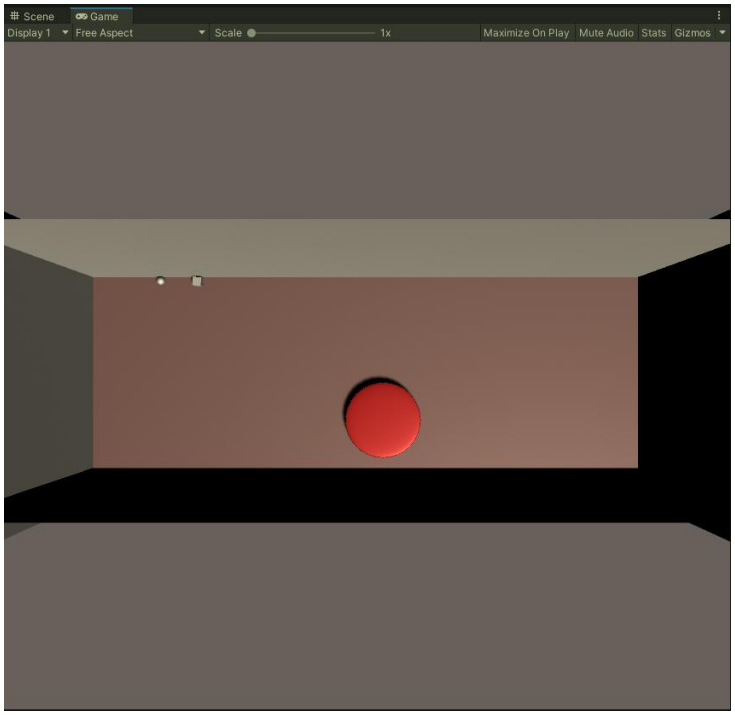

Рис. 13. Преследование игрока

Были проанализированы существующие аналоги и методы разработки, а также выбрана среда разработки. Для реализации поставленной задачи отлично подошла разработка с помощью Unity 3D и языка программирования C#. В итоге был разработан простой искусственный интеллект противника.

Библиографический список

1. Савин И. А., Батенькина О. В. Написание скриптов для трехмерного графического движка // Визуальная культура: дизайн, реклама, информационные технологии. 2018. № 12-7 (28). С. 7-15.
2. Долженко А. И., Глушенко С. А. Особенности подготовки 3d-объектов, смоделированных в Blender, для импорта в Unity 3D // Прикаспийский журнал: управление и высокие технологии. 2014. №6. С. 92-96.
3. Суродин С. А. Unity 3D. разработка сценария проектирования в среде Unity 3D// Информатика и вычислительная техника. 2015. №3. С. 504-511.
4. Гайнуллин Р. Ф., Захаров В. А., Аксенова Е. А. Создание 2d игры на Unity 3D 5.4 // Вестник современных исследований. 2018. №4. С. 78-82.
5. Субботина А. Ю., Хохлов Н. И. Реализация клеточных автоматов "игра жизнь" и клеточного автомата Кохомото-ооно с применением технологии MPI // Компьютерные исследования и моделирование. 2010. №17. С. 319-322.