Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение   
высшего образования

«Волгоградский государственный технический университет»

Факультет Электроники и вычислительной техники

Кафедра «Системы автоматизированного проектирования и поискового конструирования»

**ОТЧЕТ**

Об учебной практике на кафедре САПР и ПК ВолгГТУ

Руководитель практики от университета \_\_\_\_\_\_\_\_\_\_\_\_ Шабалина О.А.

Студент гр. ИВТ-263 Поляков А.С.

Отчет защищен с оценкой\_\_\_\_\_\_\_\_\_\_\_\_

Волгоград 2019 г

**Введение**

В ходе прохождения учебной практики и в её рамках требуется изучение моделей поведения интеллектуального агента, возможностей движка Unity, а также принципов работы с GitHub. С помощью этих средств необходимо разработать конечно-автономную модель поведения.

1. **Что представляет из себя конечный автомат.**

Конечный автомат (или попросту FSM — Finite-state machine) это модель вычислений, основанная на гипотетической машине состояний. В один момент времени только одно состояние может быть активным. Следовательно, для выполнения каких-либо действий машина должна менять свое состояние. Конечные автоматы обычно используются для организации и представления потока выполнения чего-либо. Это особенно полезно при реализации ИИ в играх. Например, для написания «мозга» врага: каждое состояние представляет собой какое-то действие (напасть, уклониться и т. д.). Конечный автомат можно представить в виде графа, вершины которого являются состояниями, а ребра — переходы между ними. Каждое ребро имеет метку, информирующую о том, когда должен произойти переход.

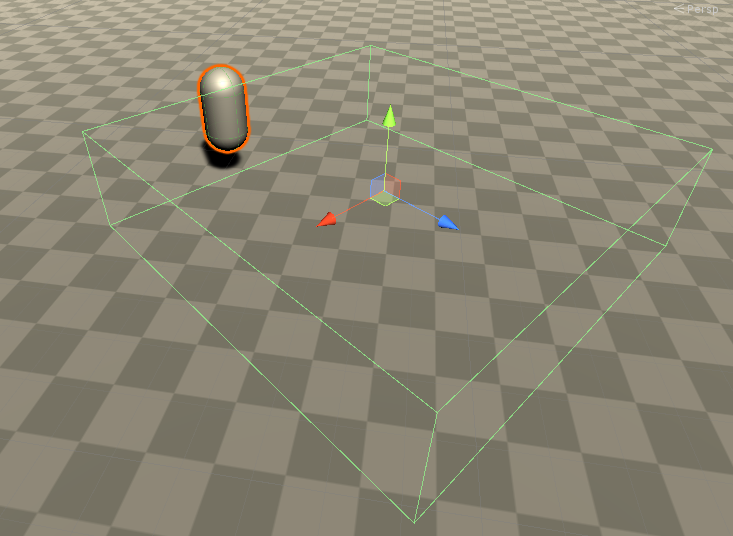
1. **Методы решения задачи**

Нашей группой был выбран метод реализации конечного автомата при помощи одного класса -StateMachine. Идея состоит в том, чтобы реализовать каждое состояние как метод или функцию.Всякое состояние есть функция. Причем такая, что она будет вызываться при каждом обновлении кадра игры. Метод update() класса StateMachine должен вызываться каждый кадр игры. А он, в свою очередь, будет вызывать функцию того состояния, которое в данный момент является активным.Более того, каждая функция, определяющая какое-то состояние автомата, не обязательно должна принадлежать классу StateMachine — это делает наш класс более универсальным.

1. **Описание процесса разработки и результатов тестирования**

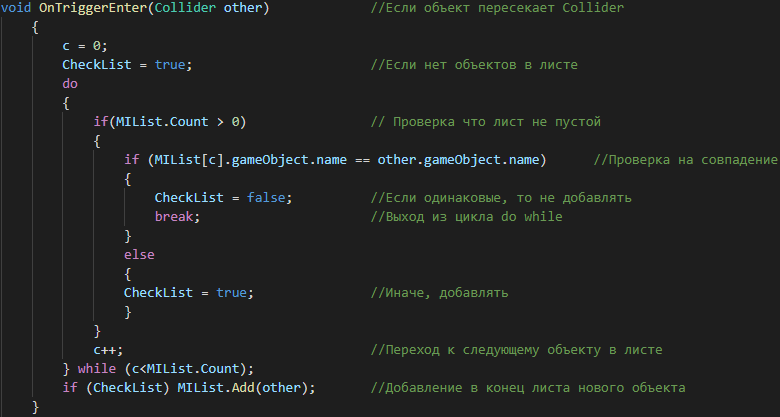
Как уже было сказано ранее, конечный автомат реализован одним классом, который вызывает функцию того состояние, которое являеться активным. Для этого, очевидно, нужны эти вспомогательные классы состояний. В ходе командной работы было разработано несколько вспомогательных классов, и хоть мы и помогали друг другу и совещались по поводу их взаимодействия, мною были разработаны вспомогательные классы, такие как:

InteractSystem – Класс (Система) реализующая «зрение» искусственного интеллекта. Система взаимодействия определяет, что он видит перед собой и нужно ли ему это. Например, система попадания предмета в поле зрения AI:

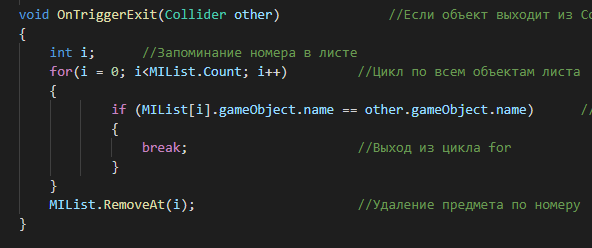


Зеленый короб олицетворяет «зрение» ИИ, если в него что то попадает, то система взаимодействия регистрирует и заносит в список этот объект.

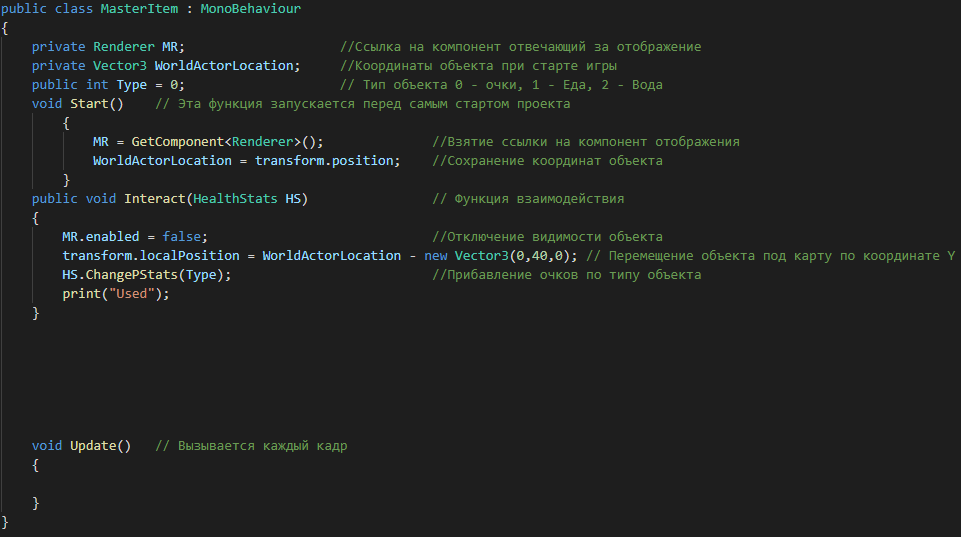
Функция OnTriggerEnterрегистрирует столкновение объектов и добавляет их в список видимых объектов перед ним.



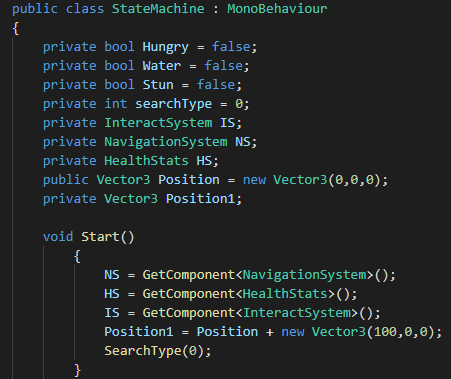
Есть и обратная, в случае если объект перестает взаимодействовать с AIиспользуеться функция OnTriggerEnter.



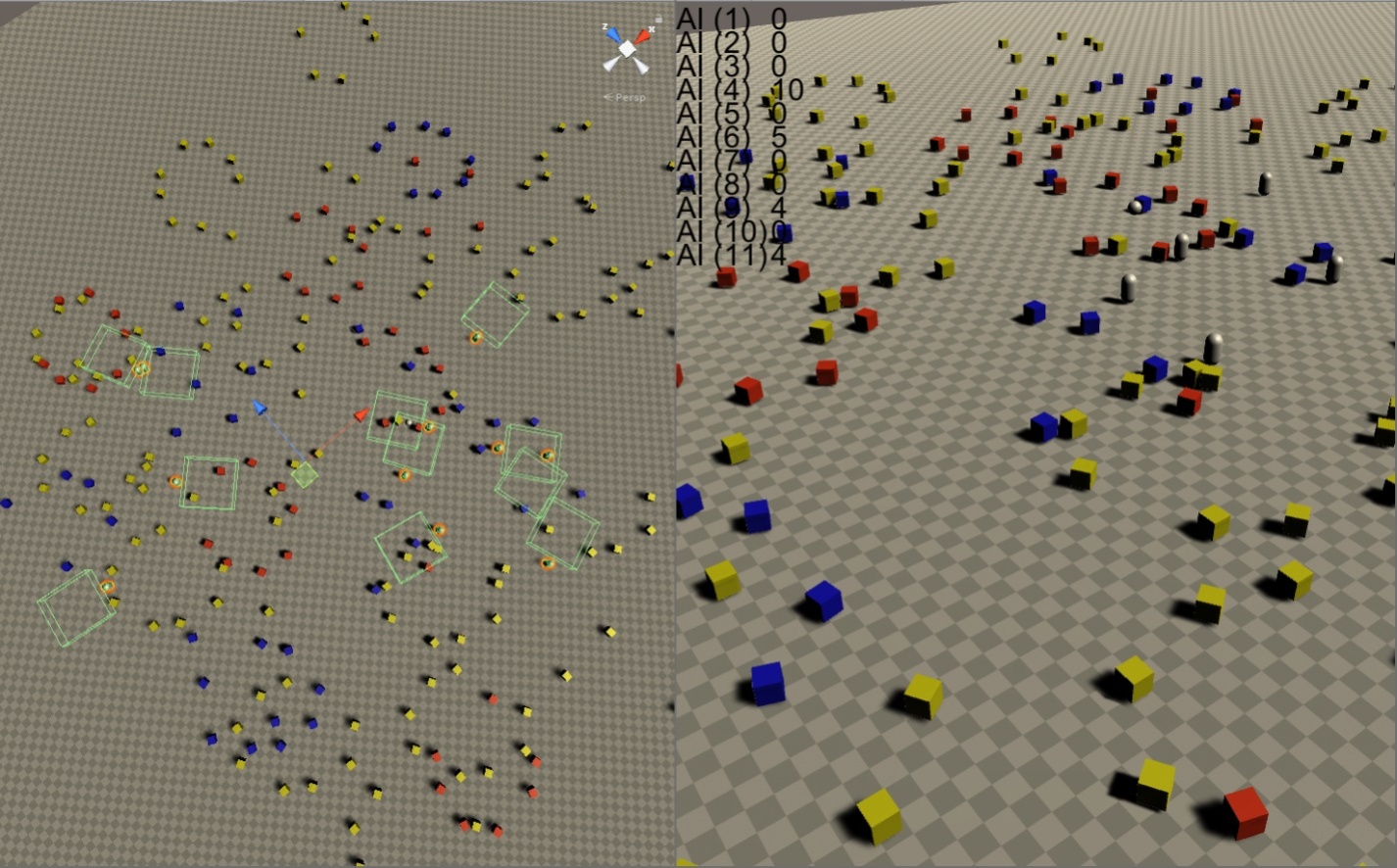
MasterItem – Класс, реализующий систему подбираемых предметов, напрямую связан с системой InteractSystem.



StateMachine – конечный автомат, функция которого описана в пункте 2.



Скрин работающей программы на движке Unity:



**Заключение**

В ходе работы были изучены принципы работы с GitHub, умение делать ветки и коммиты. Изучены модели поведения интеллектуального агента. Реализована конечно-автономная модель поведения интелектуально агента, а также вспомогательные классы.

**Список использованной литературы**

1. Конечный автомат: теория и реализация[Электронный ресурс]. -https://tproger.ru/translations/finite-state-machines-theory-and-implementation/
2. Создание и использование скриптов [Электронный ресурс]. - https://docs.unity3d.com/ru/530/Manual/CreatingAndUsingScripts.html
3. Проверка нахождения точки в коллайдере [Электронный ресурс] -http://www.unity3d.ru/distribution/viewtopic.php?f=18
4. NavMeshAgent[Электронныйресурс]. - https://github.com/Unity-Technologies/NavMeshComponents