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-262 Манукян А.В.

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.

2. 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.

3. Описание процесса разработки и результатов тестирования

Как уже было сказано ранее, конечный автомат реализован одним классом, который вызывает функцию того состояние, которое является 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:

HealthStats - основная функция регулирование состояний агента, таких как голод, жажда и здоровье, а также взаимодействие основных состояний агента, их взаимодействие с другими вспомогательными классами, например:

```
private bool ChangeHealth(int i)
    {
        if(i!=0)
        Health -= (Mathf.Exp(Health/25))/(Health*10);

        if(Health < MinHealth)
        return true;

        else return false;
    }
```

Функция ChangeHealth принимает аргумент i, который доставляеться из функции ChangeStats, и если наш агент начинает испытывать голод или жажду (параметры Hungry или Thirst равны 0), у него начинает уменшаться количество здоровья (Health).

```
private int ChangeStats()
    {
        if(Hungry > 30f)
        {

            SM.SetHungry(false);
        }
        else
        {
            SM.SetHungry(true);
        }
        if(Hungry > 0f){
           Hungry -= 0.005f;
        }
        else return 1;
        if(Thirst > 30f)
        {

            SM.SetThirst(false);
        }
        else
        {
            SM.SetThirst(true);
        }
        if(Thirst > 0f){
           Thirst -= 0.01f;
        }
        else return 2;
    return 0;
    }
```

NavigationSystem- основная функция регулирование передвижения агента, а именно: задание случайных точек для передвижения агента, измерение дистанции от агента до точки, а также использование NavMeshAgent-специального класса Unity для передвижения агента.

StateMachine - основной класс, являющийся конечным автоматом. Был разработан в группе.

В результате на движке Unity готова игра, в которой интелектуальные агенты собирают очки, еду и воду, испытвают голод, жажду, теряют здоровье.

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.

Дальнейшее развитие возможно, имея конечный автомат модели поведения и опыт работы с Unity, можно создать более сложную игру, например RTS- Real Time Strategy, которая будет полностью или полу-автономной игрой.

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