Лабораторная работа №2:
Навигация и взаимодействие объектов

Цель:

Целью данной лабораторной работы является знакомство с методом организации навигации и взаимодействия анимированных объектов в среде Unity.

Задание:

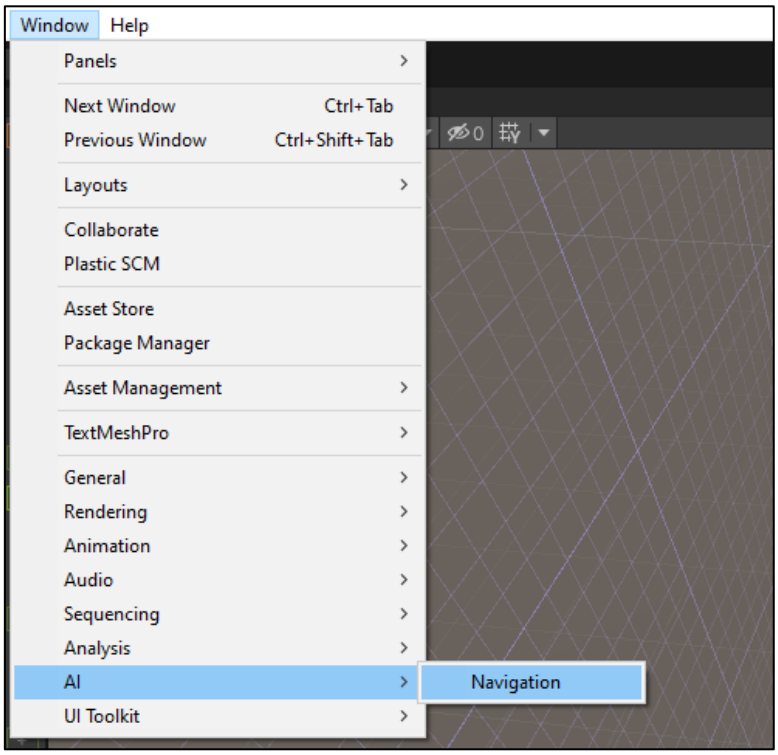
Создать интерактивное приложение трёхмерной графики содержащее:

1. Персонажа, имеющего набор анимаций: простоя, перемещения и взаимодействия. Пользователь должен иметь возможность управлять персонажем при помощи курсора мыши. В зависимости от того, куда попадает клик мыши, персонаж должен: либо перемещаться к указанной точке или объекту, либо осуществлять взаимодействие с указанным объектом (если находится достаточно близко).
2. Объект дверь, имеющую анимации: простоя, открытия и закрытия. Дверь должна проигрывать анимации по клику мыши, если персонаж находится достаточно близко.
3. Объект противник, имеющий анимации: простоя, перемещения, атаки и смерти. Противник должен перемещаться к позиции игрока если находится дальше дистанции атаки и проигрывать анимацию атаки если находится в пределах дистанции атаки. Анимация смерти должна проигрываться при получении повреждений от персонажа.

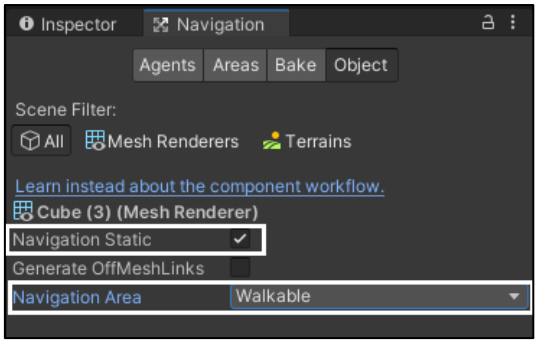
Пример выполнения первой и второй частей задания: https://h-anim.github.io/DMaster/

Навигация

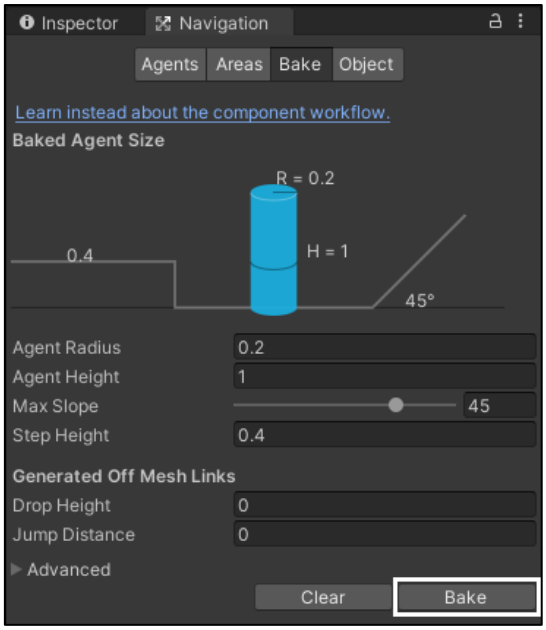
Помимо прочего, в состав Unity входит инструмент навигации объектов, называемый NavMesh. Получить доступ к инструменту можно кликнув Window, выбрав AI и Navigation:



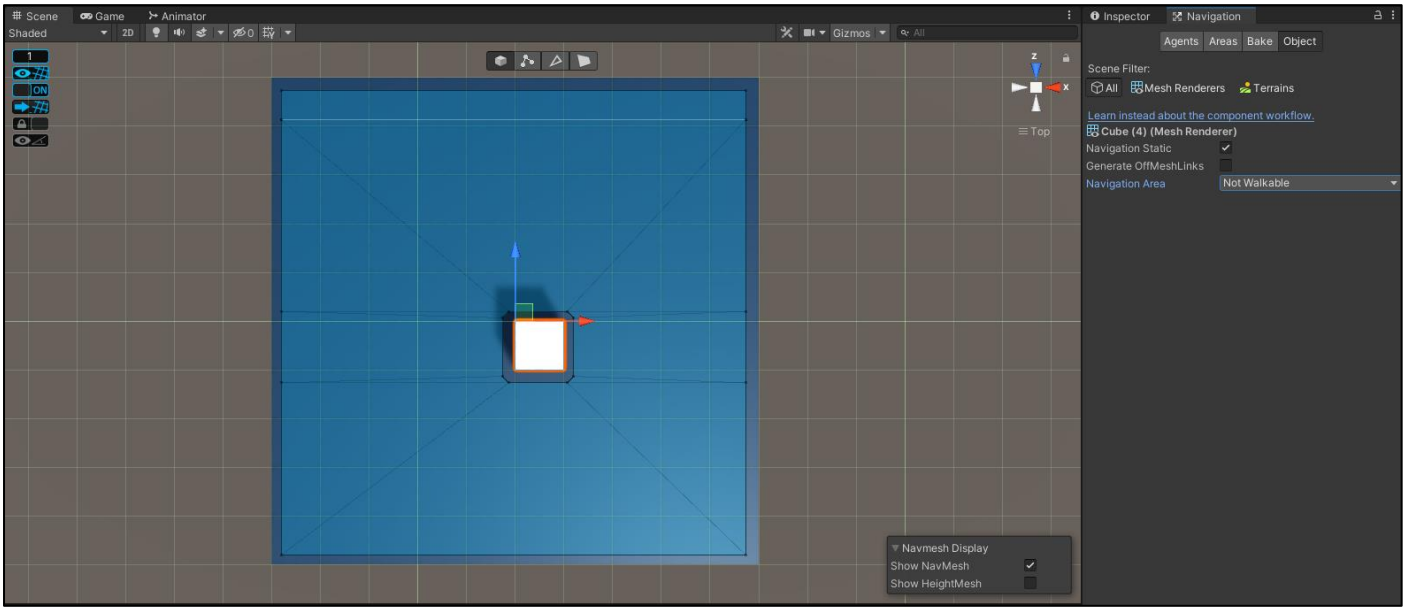
Обозначить область как доступную для перемещения, можно выбрав соответствующие объекты в сцене, или в иерархии, перейдя в раздел Navigation, выбрав вкладку Object, отметив поле Navigation Static и выбрав Walkable:



Что бы создать поверхность доступную для навигации, перейдите на вкладку Bake и нажмите Bake:

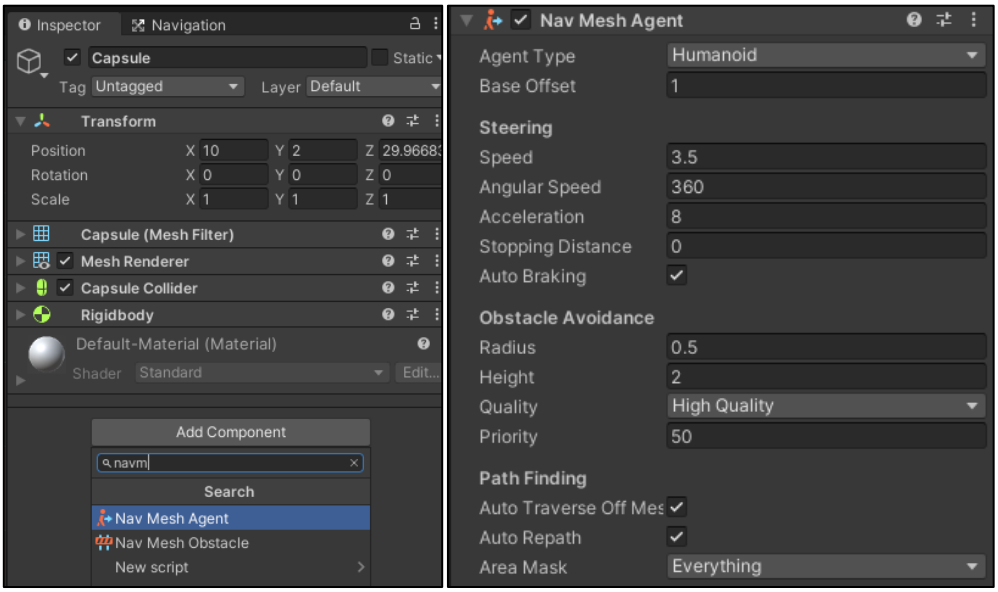


Добавить препятствие можно добавив объект в сцену, отметив его как Not Walkable и нажав Bake:



Обратите внимание, что размер областей, не доступных для перемещения, зависит от Baked Agent Size.

Использовать NavMesh для перемещения объекта можно добавив к этому объекту компонент Nav Mesh Agent:



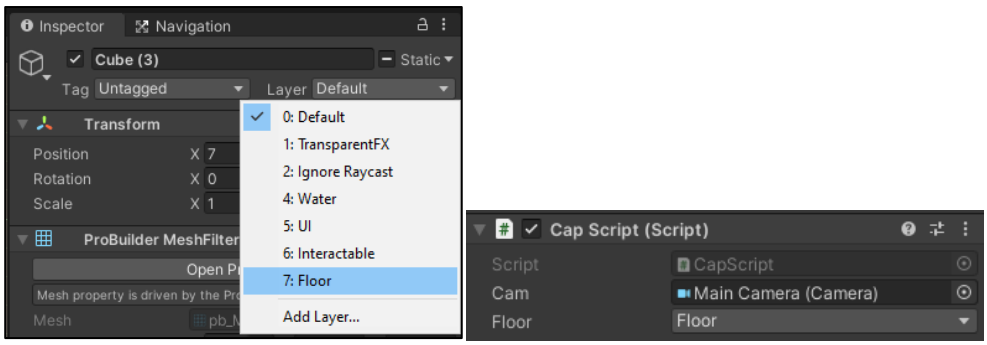
Обратите внимание, что Nav Mesh Agent содержит настройки скорости перемещения, скорость поворотов, параметры огибания препятствий и т.д.

Для использования навигации в скриптах, необходимо подключить пространство имён AI. После этого, перемещение объекта осуществляется заданием точки назначения через сылку на Nav Mesh Agent:

```
using UnityEngine;
using UnityEngine.AI;

Ссылок: 0
public class CapScript : MonoBehaviour
{
    NavMeshAgent agent; //ссылка на компонент навигации

    public Camera cam;

    public LayerMask floor; //слой, определённый для группы объектов "пол"

    Ссылок: 0
    void Start()
    {
        agent = GetComponent<NavMeshAgent>();   //получение ссылки на объект навигации
    }

    Ссылок: 0
    void LateUpdate()
    {
        if (Input.GetAxis("Fire1") > 0) //при клике левой кнопкой мыши
        {
            RaycastHit hit;

            Ray ray = cam.ScreenPointToRay(Input.mousePosition);

            if (Physics.Raycast(ray, out hit, 100, floor))  //если клик попал в "пол"
            {
                agent.SetDestination(hit.point);    //установка точки, к которой должен переместиться игрок
            }
        }
    }
}
```

В приведённом выше примере, точка назначения указывается кликом мыши по области, доступной для перемещения. Помимо этого, в качестве точки назначения можно использовать позицию объекта или персонажа.

В качестве оптимизации работы приложения, можно вынести область доступную для перемещения на отдельный слой сцены. Для этого выберите поверхности доступные для перемещения, а затем в окне инспектора, укажите в поле Layer желаемое значение:

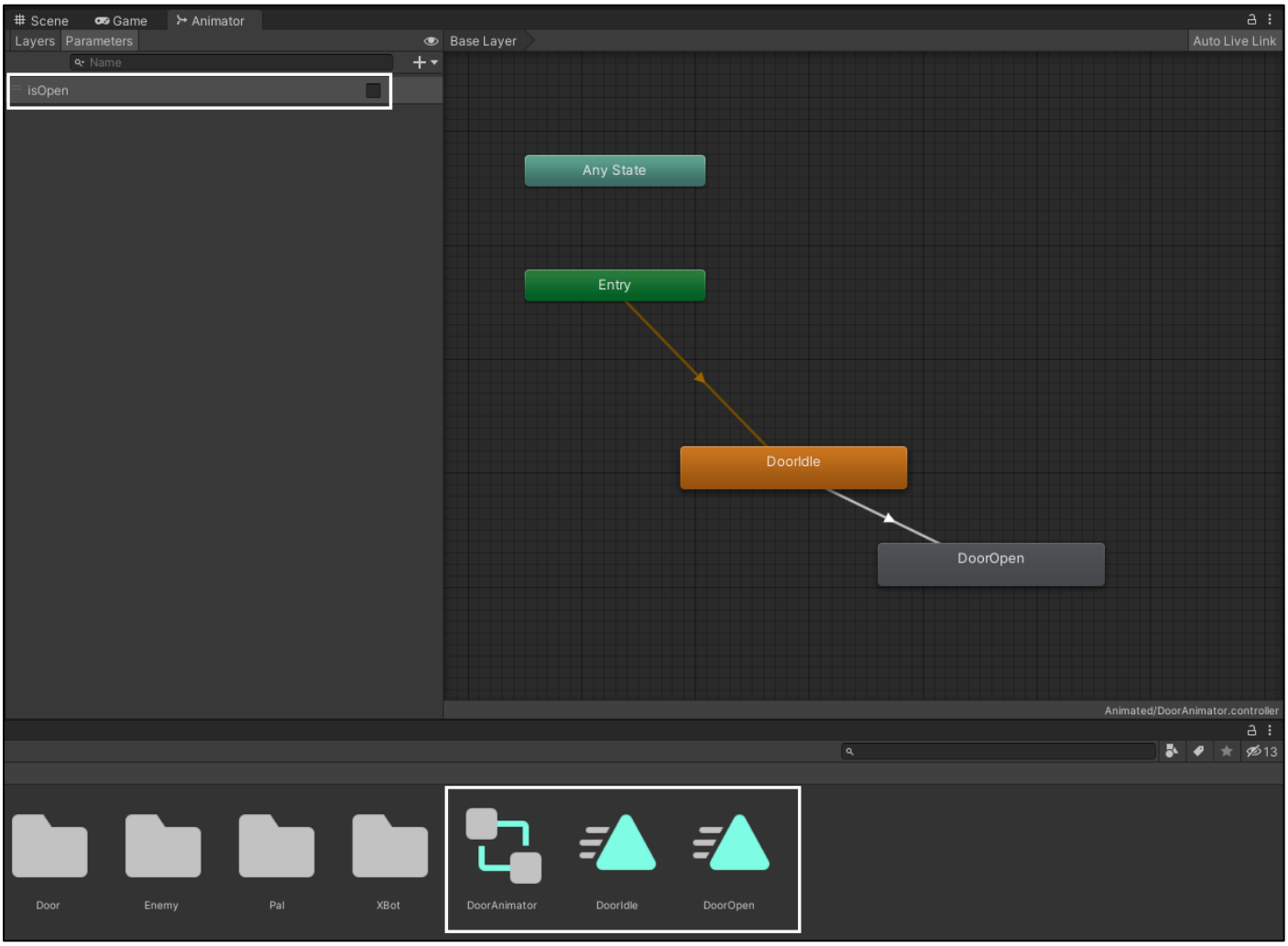


Добавить слой Floor, или любой другой, можно кликнув по Add Layer…

Полученный скрипт можно прикрепить к объекту, для которого необходимо реализовать перемещение. Если всё было сделано правильно, то при клике по области доступной для перемещения, объект будет перемещаться в указанную точку.

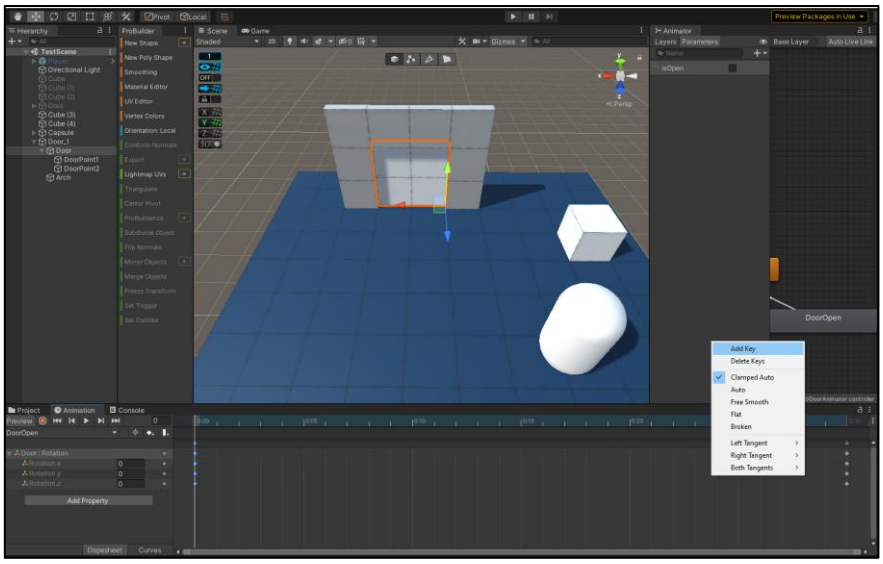
Создание анимации по ключевым кадрам

Помимо прочего, Unity имеет встроенные инструменты для создания и редактирования анимаций. Для реализации анимации открытия/закрытия двери, создайте Animation Controller и 2 Animation:

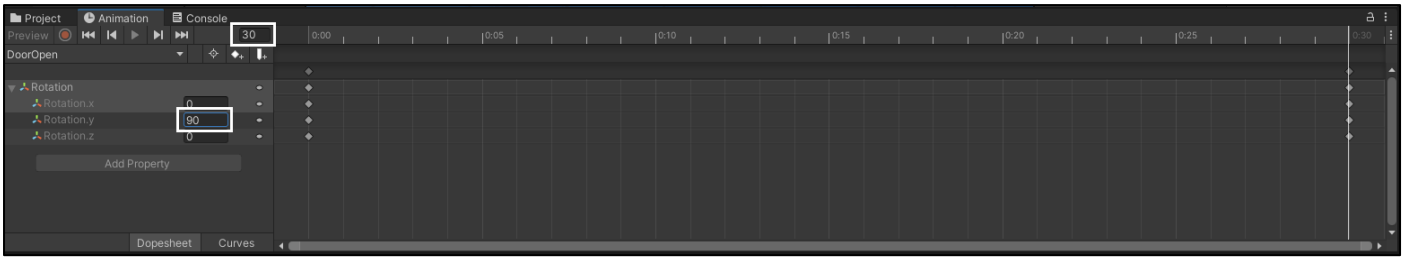


При двойном клике по созданной анимации, будет открыт редактор анимаций. Выбрав объект, который необходимо анимировать, можно, при помощи Add Property, выбрать параметры объекта, которые будут анимированы (в данном случае – поворот по оси Y):

Кликнув правой кнопкой мыши ниже шкалы времени анимации, можно добавить ключевой кадр:



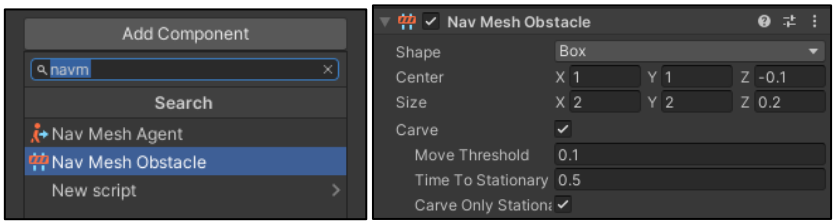
Для каждого ключевого кадра можно указать значение поворота по соответствующей оси:



Для реализации неподвижного состояния двери можно использовать анимацию со значениями по умолчанию. Для анимации открытия и закрытия, достаточно двух анимаций, содержащих по 2 ключевых кадра. Первый – исходное положение и второй – конечное положение.

Поскольку дверь может блокировать часть области, доступной для перемещения, она должна являться препятствием. Однако, дверь не является статичным препятствием, поэтому вариант с обозначением её в качестве такового в окне Navigation не подходит.

Для обозначения динамических препятствий (которые могут перемещаться по сцене), используется компонент Nav Mesh Obstacle:



Взаимодействие с объектами

Поскольку в сцене может быть множество различных объектов, взаимодействие с которыми может отличаться, имеет смысл определить интерфейс, описывающий такие объекты:

```
public abstract class IObject : MonoBehaviour //интерфейс объекта, с которым можено осуществлять взаимодействие
{
    Ссылок: 4
    public abstract void interact();    //метод взаимодействия с объектом
    Ссылок: 3
    public abstract Vector3 getPoint(Vector3 position); //метод получения ближайшей к объекту точки
}
```

Такой интерфейс гарантирует что все объекты сцены, с которыми может осуществляться взаимодействие, будут предоставлять метод, возвращающий точку, с которой будет происходить взаимодействие, а также метод, позволяющий с ними взаимодействовать.

Реализация такого интерфейса для объекта “Дверь” может выглядеть следующим образом:

```
public class DoorScript : IObject //реализация интерфейса взаимодействия для объекта Дверь
{
    Animator an;    //ссылка на аниматор

    public Transform point1;    //точки по обе стороны двери, к которым должен подойти персонаж
    public Transform point2;    //для начала взаимодействия

    Ссылок: 0
    void Start()
    {
        an = GetComponent<Animator>();
    }

    Ссылок: 2
    public override States getAction()
    {
        return States.Open; //при взаимодействии с Дверью, игрок должен воспроизвести анимацию открытия
    }

    Ссылок: 3
    public override void interact() //реализация метода взаимодействия
    {
        an.SetBool("isOpen", true); //проигрывание анимации открытия двери
    }

    Ссылок: 2
    public override Vector3 getPoint(Vector3 position)  //получение ближайшей к игроку точки
    {
        var dist1 = Vector3.Distance(position, point1.position);
        var dist2 = Vector3.Distance(position, point2.position);

        if (dist1 < dist2)
            return point1.position;
        else
            return point2.position;
    }
}
```

Точки создаются как EmptyObject, размещаются рядом с объектом “Дверь” и передаются в качестве параметра в скрипт.

Взаимодействие с объектами со стороны управляемого персонажа может выглядеть следующим образом. Глобальные переменные:

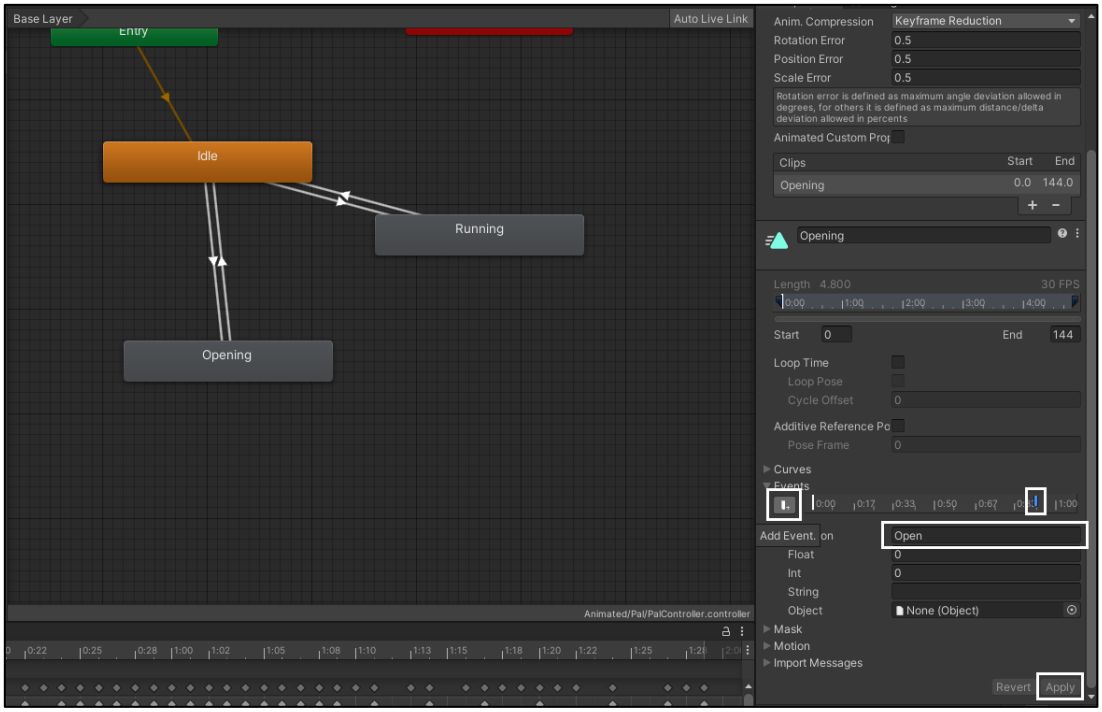
```
NavMeshAgent agent; //ссылка на компонент навигации

public Camera cam;

public LayerMask floor; //слой, определённый для группы объектов "пол"
public LayerMask inter; //слой, определённый для группы объектов, с которыми может осуществляться взаимодействие

public float interactionRange = 2.0f;   //дистанция, в пределах которой может начаться взаимодействие с объектом
```

Реакция на нажатие левой кнопки мыши:

```
void LateUpdate()
{
    if (Input.GetAxis("Fire1") > 0) //при клике левой кнопкой мыши
    {
        RaycastHit hit;

        Ray ray = cam.ScreenPointToRay(Input.mousePosition);

        if (Physics.Raycast(ray, out hit, 100, inter))  //если клик попал в объект, с которым может осуществляться взаимодействие
        {
            Vector3 objPosition = hit.transform.GetComponent<IObject>().getPoint(transform.position);   //получение ближайшей к объекту точки

            double dist = Vector3.Distance(objPosition, transform.position);    //вычисление дистанции между игроком и ближайшей к объекту точки

            if (dist < interactionRange)    //если объект находится в пределах дистанции начала взаимодействия
            {
                hit.transform.GetComponent<IObject>().interact();   //получение номера анимации, связанного с взаимодействием
            }
            else
            {
                agent.SetDestination(objPosition); //установка точки, к которой должен переместиться игрок
            }
        }
        else
        if (Physics.Raycast(ray, out hit, 100, floor))  //если клик попал в "пол"
        {
            agent.SetDestination(hit.point);    //установка точки, к которой должен переместиться игрок
        }
    }
}
```

События анимации

В ряде случаев, некоторые действия следуют выполнять только в определённый момент воспроизведения анимации. Реализовать это можно при помощи событий анимации. Что бы добавить событие, кликните дважды по анимации, разверните список Events, нажмите Add Event, укажите кадр анимации, на котором будет срабатывать событие, введите название события и нажмите Apply:



Обработать событие в скрипте можно создав метод с название события:

```
public void Open()  //событие, связанное с анимацией открытия
{
    hit.transform.GetComponent<IObject>().interact();   //вызов метода взаимодействия объекта, в который попал клик
    anim.SetInteger("State", (int)States.Idle);         //установка анимации простоя
}
```