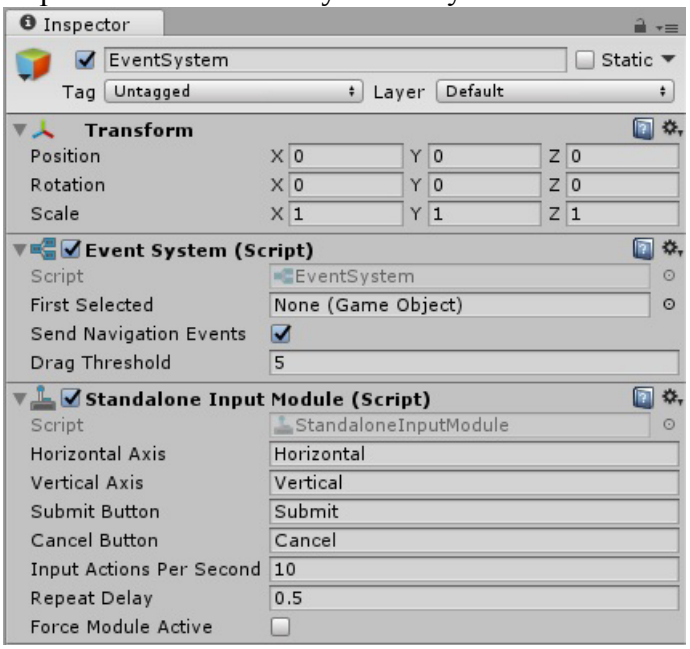
Система обработки сообщений

Для программного доступа к UI-компонентам к коду надо подключить пространство имен UnityEngine.UI.

Объект Event System

С системой обработки сообщений связан особый игровой объект EventSystem, который автоматически добавляется к иерархии при добавлении в нее канвы или любого другого UI-объекта.

Игровой объект EventSystem по умолчанию включает три компонента.



Компонент Event System (Event System Manager)

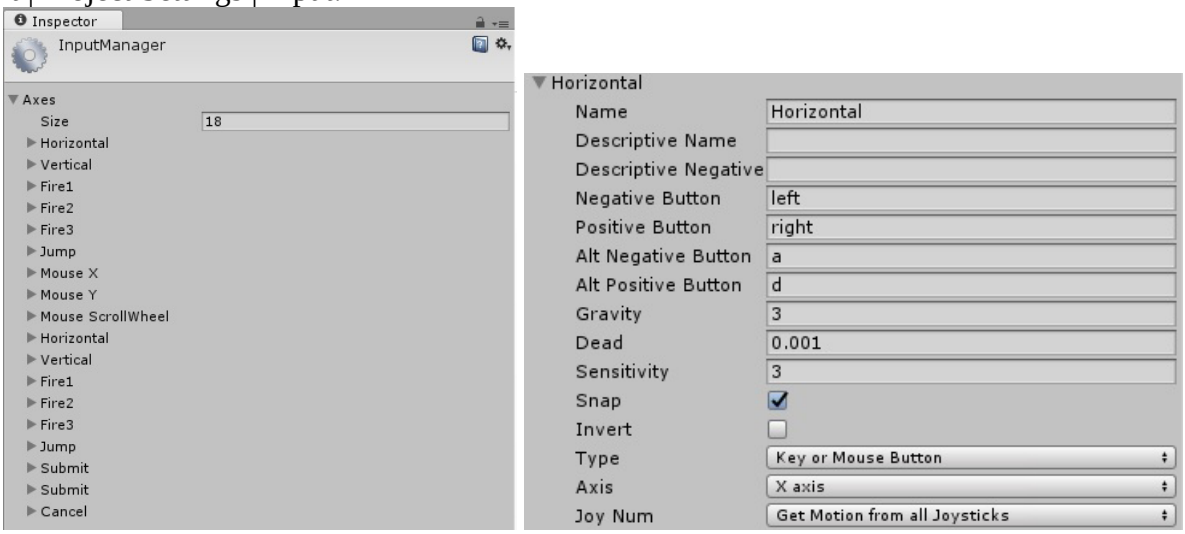
First Selected — какой UI-объект будет выбран при загрузке сцены.

Send Navigation Events — делает доступными события Move (перебор UI-объектов с помощью клавиатуры), Submit и Cancel.

Drag Threshold — число пикселей, при сдвиге на которое действие пользователя расценивается как перетаскивание.

Компонент Standalone Input Module

Данный компонент связан с системой Input Manager, отобразить настройки которой можно командой Edit | Project Settings | Input.



Изменяя значение Size, можно удалять или добавлять «оси». Кроме того, любую ось можно удалить или создать ее копию с помощью контекстного меню этой оси.

Оси с одинаковыми именами связаны с одинаковыми действиями, которые планируется реализовывать с помощью различных устройств (например, два варианта оси Horizontal связаны с клавиатурой/мышью и джойстиком).

Свойство Gravity определяет размер «трения» при движении (при Gravity = 0 трение отсутствует). Включенное свойство Snap обеспечивает немедленное изменение направления движения, если нажата противоположная клавиша (если отключено, то направление движения сохраняется, но уменьшается его значение). Изменить эти значения из скрипта нельзя.

Компонент Standalone Input Module представляет собой стандартную реализацию интерфейса между программой и внешними устройствами ввода: клавиатурой, мышью, джойстиком и сенсорным экраном. Имеются и другие варианты подобных компонентов (Hololens Input Module, Base Input Module, Pointer Input Module). Они либо предоставляют интерфейс для специализированных устройств ввода (Hololens), либо являются основой для разработки собственных модулей ввода (Base и Pointer).

Событие Submit по умолчанию связано с клавишами Enter и Space (но не с щелчком мыши!). Событие Cancel по умолчанию связано с клавишей Escape. Далее, при описании объекта Button, будет приведен пример использования этих событий.

Функции класса Input для проверки возникших событий

Все эти функции обычно вызываются в методе Update.

GetButton(axisName), GetButtonDown(axisName), GetButtonUp(axisName)

Функция GetButton возвращает true, пока клавиша/кнопка остается нажатой, функция GetButtonDown возвращает true только в кадре, в котором клавиша/кнопка была нажата, функция GetButtonUp возвращает true только в кадре, в котором клавиша/кнопка была отпущена.

Пример:

```
void Update()
{
  if (Input.GetButtonDown("Submit"))
  {
    Debug.Log("You pressed a submit key/button!");
  }
}
```

Реагирует и на клавиши, и на кнопки мыши, если они определены для данной оси.

GetAxis(axisName)

Возвращает вещественное значение в диапазоне от –1 до 1, если нажата клавиша/кнопка, связанная с данной осью.

Пример (реализация перемещения UI-объекта, при котором ось Vertical отвечает за движение «вперед» и «назад», а ось Horizontal отвечает за поворот):

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;

public class Keys : MonoBehaviour {

    Image image;
    public float speed = 100.0f;
    public float rotationSpeed = 100.0f;

    void Start()
    {
        image = GetComponent<Image>();
    }

    void Update()
```

```
3
    {
        float translation = Input.GetAxis("Vertical") * speed * Time.deltaTime;
        float rotation = Input.GetAxis("Horizontal") * rotationSpeed * Time.deltaTime;
        transform.Translate(0, translation, 0);
        transform.Rotate(0, 0, -rotation);
    }
}
```

Интересно проанализировать изменение поведения объекта при изменении свойства Gravity оси Vertical (данное изменение можно выполнить только в редакторе).

GetKey(keyCode), GetKeyDown(keyCode), GetKeyUp(keyCode)

Возвращает true, если была нажата/отпущена клавиша, указанная в качестве параметра (особенности аналогичны особенностям функций группы GetButton). Параметр имеет тип перечисления KeyCode с набором значений, соответствующих различным клавишам: `KeyCode.A` – `KeyCode.Z`, `KeyCode.Alpha0` – `KeyCode.Alpha9` (цифровые клавиши на основной клавиатуре), `KeyCode.Keypad0` – `KeyCode.Keypad9` (цифровые клавиши на дополнительной клавиатуре), `KeyCode.UpArrow`, `KeyCode.DownArrow`, `KeyCode.LeftArrow`, `KeyCode.RightArrow`, `KeyCode.PageUp`, `DownArrow`,`PageDown`, `KeyCode.Home`, `KeyCode.End`, `KeyCode.F1` – `KeyCode.F12`, `KeyCode.Return`, `KeyCode.KeypadEnter`, `KeyCode.LeftShift`, `KeyCode.RightShift`, `KeyCode.LeftControl`, `KeyCode.RightControl`, `KeyCode.LeftAlt`, `KeyCode.RightAlt`.

GetMouseButton(bNum), GetMouseButtonDown(bNum), GetMouseButtonUp(bNum)

Возвращают значение true, если нажата соответствующая кнопка мыши. Параметр bNum является целым числом (0 — левая кнопка мыши, 1 — правая, 2 — средняя). Особенности данных функций аналогичны особенностям функций группы GetButton.

GetTouch(index)

Возвращает структуру Touch, содержащую информацию о касании сенсорного экрана. Параметр index определяет номер касания в данном кадре (от 0). Число касаний можно определить с помощью свойства Input.touchCount.

Среди свойств структуры Touch можно отметить phase (фаза нажатия, перечисление типа TouchPhase), position (позиция нажатия типа Vector2, отсчитывается от левого нижнего угла экрана, координаты правого верхнего угла можно определить с помощью свойств Screen.width и Screen.height), tapCount (число касаний типа int). Перечисление TouchPhase имеет значения Began (начало касания экрана), Moved (движение по экрану), Stationary (продолжение неподвижного касания), Ended (обычное завершение касания), Canceled (особое завершение касания, «this might happen if, for example, the user puts the device to their face or simultaneously applies more touches than the system can track»).

Дополнительные свойства класса Input

bool anyKey — нажата и удерживается какая-либо клавиша или кнопка мыши;

bool anyKeyDown — в данном кадре нажата какая-либо клавиша или кнопка мыши;

Vector3 mousePosition — позиция мыши (отсчитывается от левого нижнего угла в пикселях, третья координата равна нулю).

Компонент Event Trigger

Add Component | Event | Event Trigger

Может быть связан с любым объектом (в том числе UI-объектом), позволяет настроить обработчики событий для этого объекта в режиме дизайна.

Будучи присоединен к игровому объекту, данный компонент получает все события системы событий (даже если для них не указаны обработчики), что может отрицательно сказаться на производительности. Альтернативным вариантом является разработка скрипта, явно реализующего интерфейсы, связанные с требуемыми событиями (см. далее).

Если данный компонент присоединен к игровому объекту, не являющемуся UI-объектом, то этот объект должен быть снабжен компонентом-коллайдером из набора (Physics 2D или Physics), кроме того, с камерой должен быть связан соответствующий источник лучей из набора Event (Physics 2D Raycaster или Physics Raycaster). Для UI-объектов все требуемые компоненты уже содержатся в канве.

Типы событий, поддерживаемые компонентом Event Trigger

Большинство событий связано с областью, ограничивающей UI-объект (или с коллайдером игрового объекта).

События, связанные с указателем (pointer events)

Связываются с мышью, сенсорным экраном или тачпадом:
PointerEnter — появление указателя над объектом,
PointerExit — перемещение указателя за границы объекта,
PointerDown — нажатие указателя над объектом,
PointerUp — отпускание указателя над объектом,
PointerClick — нажатие и затем отпускание указателя над объектом.
В случае событий PointerDown и PointerUp не обязательно, чтобы оба действия выполнялись над объектом.

События перетаскивания (drag and drop events)

Все эти события, кроме Drop, происходят на стороне перетаскиваемого объекта (объекта-источника).
InitializePotentialDrag — наступает непосредственно перед началом перетаскивания, когда уже определен потенциальный источник,
BeginDrag — наступает в момент начала перетаскивания,
Drag — возникает в процессе перетаскивания,
EndDrag — наступает при завершении перетаскивания,
Drop — возникает для объекта-приемника, над которым было завершено перетаскивание.

События выбора (selection events)

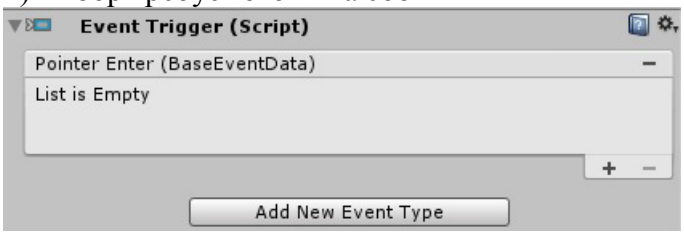
Select — объект становится выбранным (однократное событие),
Deselect — объект перестает быть выбранным (однократное событие),
UpdateSelected — объект является выбранным (возникает в каждом кадре, пока объект выбран).

Другие события

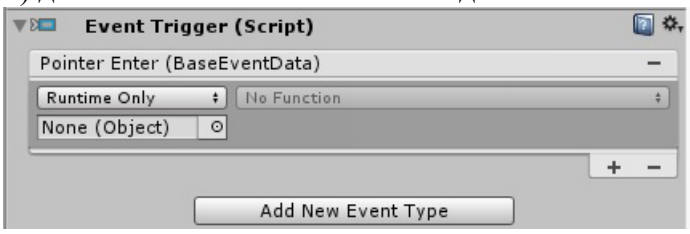
Scroll — прокрутка колесика мыши,
Move, Submit, Cancel — события, связанные с соответствующими осями.

Связывание с событиями требуемых действий

1) выбор требуемого типа события

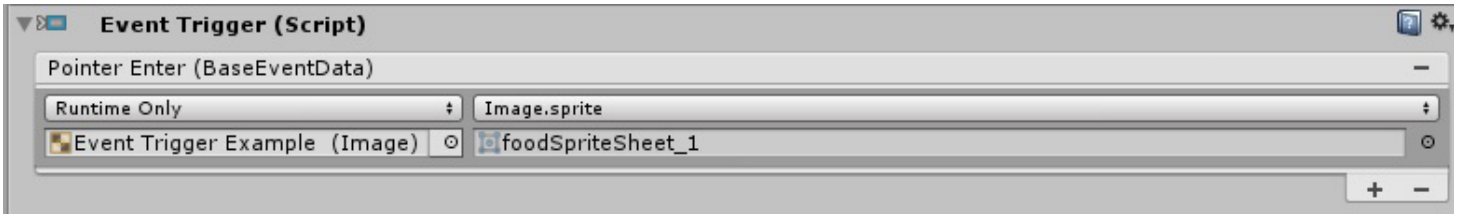


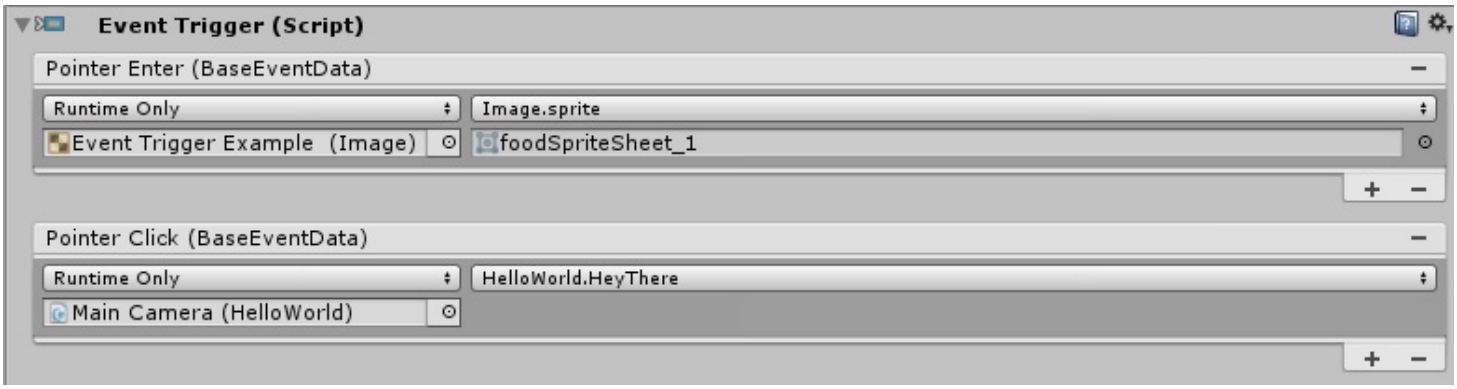
2) добавление к событию нового действия



3) определение параметров нового действия:

- Runtime Only, Editor and Runtime, Off;
- требуемый игровой объект;
- поле его компонента или public void функция его скрипта, имеющая не более одного параметра;
- значение, которое будет присвоено полю, или параметр функции.





Программная настройка событий для компонента EventTrigger

```
using UnityEngine;
using UnityEngine.EventSystems;

public class CustomListenersExample : MonoBehaviour
{
    void Start( )
    {
        EventTrigger eventTrigger = GetComponent<EventTrigger>();
        EventTrigger.Entry entry = new EventTrigger.Entry();
        entry.eventID = EventTriggerType.PointerDown;
        entry.callback.AddListener(OnPointerDown1);
        eventTrigger.triggers.Add(entry);
    }

    public void OnPointerDown1(PointerEventData data)
    {
        Debug.Log("OnPointerDown1 called.");
    }
}
```

Определение событий с помощью интерфейсов

Вместо использования компонента Event Trigger можно явно определить скрипт, реализующий требуемые события. Для этого класс скрипта должен реализовывать соответствующий интерфейс.

Интерфейс	**Имя обработчика события**	**Тип параметра обработчика**
IPointerEnterHandler	OnPointerEnter	PointerEventData
IPointerExitHandler	OnPointerExit	PointerEventData
IPointerDownHandler	OnPointerDown	PointerEventData
IPointerUpHandler	OnPointerUp	PointerEventData
IPointerClickHandler	OnPointerClick	PointerEventData
IInitializePotentialDragHandler	OnInitializePotentialDrag	PointerEventData
IBeginDragHandler	OnBeginDrag	PointerEventData
IDragHandler	OnDrag	PointerEventData
IEndDragHandler	OnEndDrag	PointerEventData
IDropHandler	OnDrop	PointerEventData
ISelectHandler	OnSelect	BaseEventData

IDeselectHandler	OnDeselect	BaseEventData
IUpdateSelectedHandler	OnUpdateSelected	BaseEventData
IScrollHandler	OnScroll	PointerEventData
IMoveHandler	OnMove	AxisEventData
ISubmitHandler	OnSubmit	BaseEventData
ICancelHandler	OnCancel	BaseEventData

Шаблон определения нового события для скрипта

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;

public class ClassName : MonoBehaviour, InterfaceName
{
  public void EventHandlerName(EventDataTypeName eventData)
  {
    //what happens after event triggers
  }
}
```

Вариант реализации интерфейсов, обеспечивающий динамическое подключение и отключение обработчиков

```
using UnityEngine;
using UnityEngine.EventSystems;

public class CustomListenersExample : MonoBehaviour, IPointerDownHandler
{
    public event Action<BaseEventData> onPointerDown;

    void Start()
    {
        onPointerDown += OnPointerDown1;
        onPointerDown += OnPointerDown2;
    }

    public void OnPointerDown(PointerEventData data)
    {
        if (onPointerDown != null)
            onPointerDown(data);
    }

    public void OnPointerDown1(PointerEventData data)
    {
        Debug.Log("OnPointerDown1 called.");
    }

    public void OnPointerDown2(PointerEventData data)
    {
        Debug.Log("OnPointerDown1 called.");
    }
}
```

Типы параметров для обработчиков событий

BaseEventData

BaseInputModule currentInputModule — модуль ввода, вызвавший данное событие (read-only),
GameObject selectedObject — игровой объект, для которого произошло это событие.

AxisEventData (потомок BaseEventData)

MoveDirection moveDir — направление, связанное с данным событием (Up, Down, Left, Right),
Vector2 moveVector — значение смещения для данного события.

PointerEventData (потомок BaseEventData)

Приведена только часть свойств.
InputButton button — кнопка, с которой связано событие (Left, Right, Middle),
int clickCount — количество щелчков,
float clickTime — время последнего щелчка,
Vector2 delta — смещение с момента последнего обновления,
bool dragging — выполняется ли в данный момент перетаскивание,
List<GameObject> hovered — список объектов, расположенных под указателем,
GameObject pointerDrag — объект, для которого произошло событие OnDrag,
GameObject pointerEnter — объект, для которого произошло событие OnPointerEnter,
GameObject pointerPress — объект, для которого произошло событие OnPointerDown,
Vector2 position — текущая позиция указателя,
Vector2 pressPosition — позиция нажатия,
Vector2 scrollDelta — изменение значения прокрутки с момента последнего обновления.