Diseño de Sistemas Interactivos

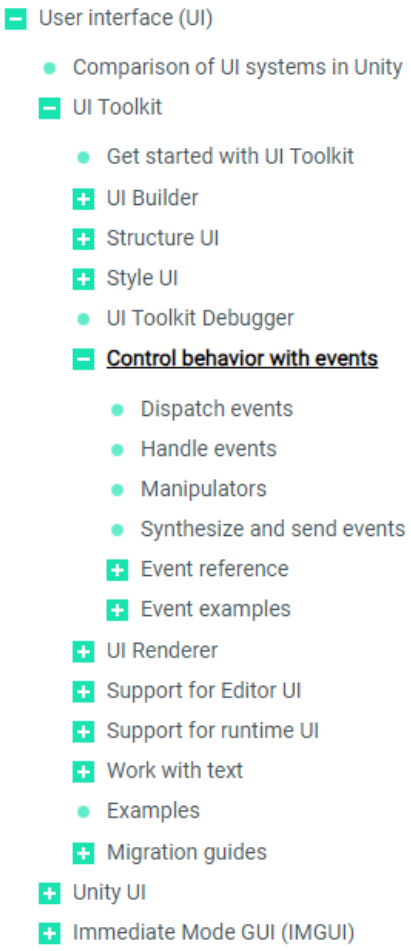
Curso 2024 - 2025

Profesor: José Manuel Velasco

Despacho 309, 3ª planta. Facultad de Informática.

Laboratorio 3: Introducción a Eventos in UI Toolkit.

Unity Manual



Control behavior with events

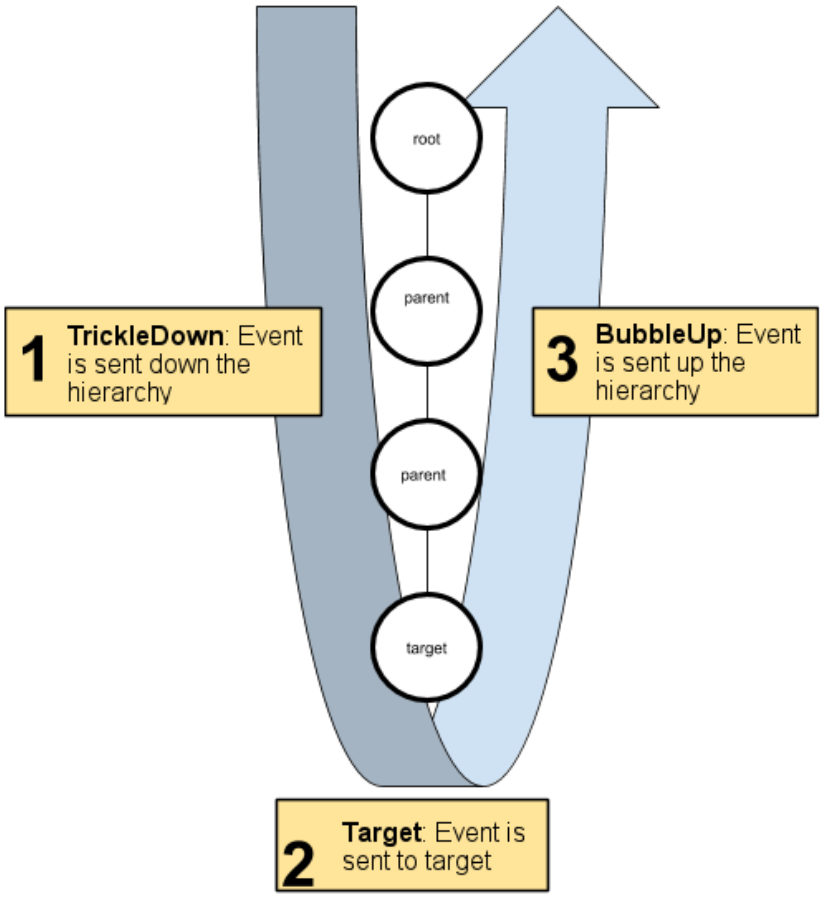
UI Toolkit provides events that communicate user actions or notifications to **visual elements**. The **UI** Toolkit **event system** shares the same terminology and event naming as HTML events.

Topic	Description
Dispatch events	Understand the event dispatch behavior, propagation, and target.
Handle events	Understand event handling with built-in and custom controls.
Use manipulators to handle events	Learn how to use manipulators to handle events from examples.
Synthesize and send events	Learn how to synthesize and send events by an example.
Event reference	Understand behavior and characteristics of each event.

UI Toolkit Event System

- **Control del comportamiento mediante eventos.**
  - **Ruta de propagación**
  - **Manejo de eventos**
  - **Modificación de Propiedades**
- Manipuladores
- Activadores
- Unity manual → Dragger
- Unity manual → Resizer

Ruta de propagación



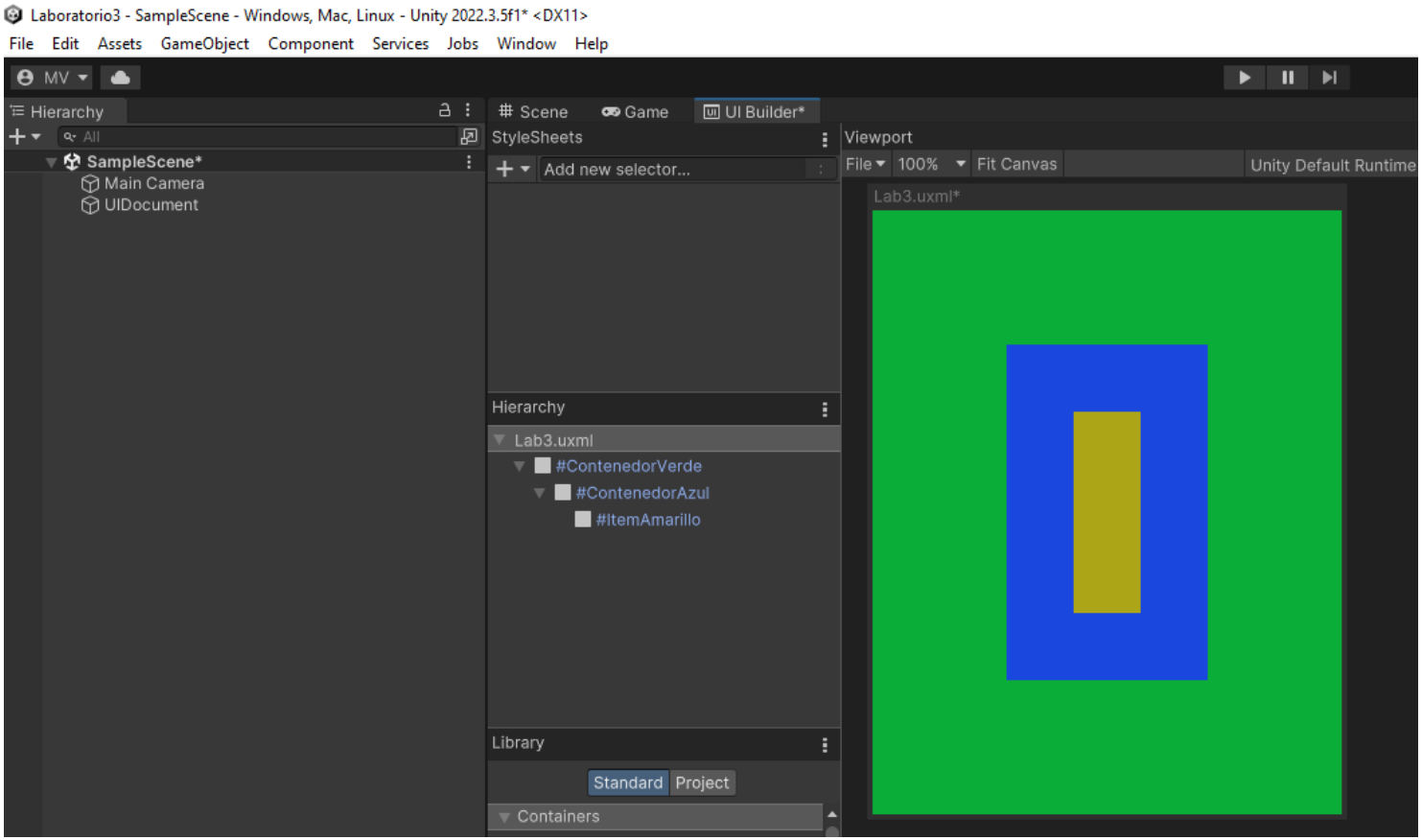
- La ruta de propagación es una lista ordenada de elementos visuales que reciben el evento. La ruta de propagación se produce en el siguiente orden:

1. La ruta comienza en la raíz del árbol de elementos visuales y desciende hacia el objetivo. Esta es la fase de **TrickleDown** (goteo).

2. El **Target** (objetivo) del evento lo recibe.

3. A continuación, el evento asciende por el árbol hacia la raíz. Esta es la fase de **BubbleUp** (burbuja ascendente).

UI Builder

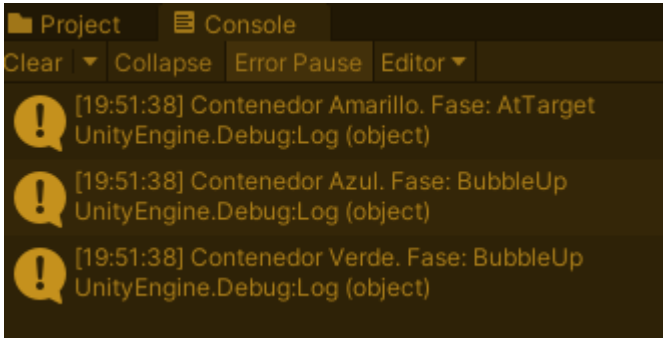

Laboratorio3 - SampleScene - Windows, Mac, Linux - Unity 2022.3.5f1* <DX11>
File Edit Assets GameObject Component Services Jobs Window Help
MV
Hierarchy
All
SampleScene*
Main Camera
UIDocument
Scene
Game
UI Builder*
StyleSheets
Add new selector...
Hierarchy
Lab3.uxml
#ContenedorVerde
#ContenedorAzul
#ItemAmarillo
Library
Standard
Project
Containers
Viewport
File
100%
Fit Canvas
Unity Default Runtime
Lab3.uxml*

Propagación

Lab3.cs

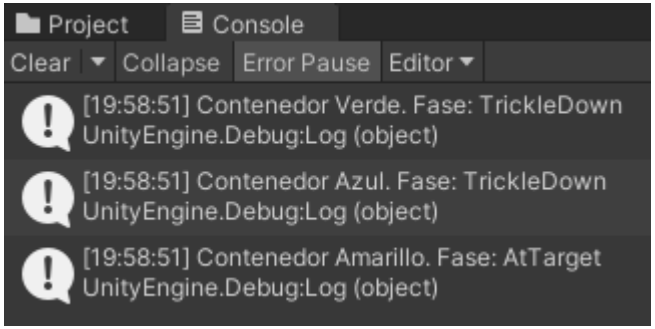
Assets > Lab3.cs > ...

```
1   using UnityEngine;
2   using UnityEngine.UIElements;
3
    0 references
4   public class Lab3 : MonoBehaviour
5   {
        0 references
6       private void OnEnable() {
7           VisualElement root = GetComponent<UIDocument>().rootVisualElement;
8
9           VisualElement verde     = root.Q("ContenedorVerde");
10          VisualElement azul      = root.Q("ContenedorAzul");
11          VisualElement amarillo  = root.Q("ItemAmarillo");
12
13          verde.RegisterCallback<MouseDownEvent>(
14              ev =>
15              {
16                  Debug.Log("Contenedor Verde. Fase: " + ev.propagationPhase);
17              });
18
19          azul.RegisterCallback<MouseDownEvent>(
20              ev =>
21              {
22                  Debug.Log("Contenedor Azul. Fase: " + ev.propagationPhase);
23              });
24          amarillo.RegisterCallback<MouseDownEvent>(
25              ev =>
26              {
27                  Debug.Log("Contenedor Amarillo. Fase: " + ev.propagationPhase);
28              });
29      }
30  }
```



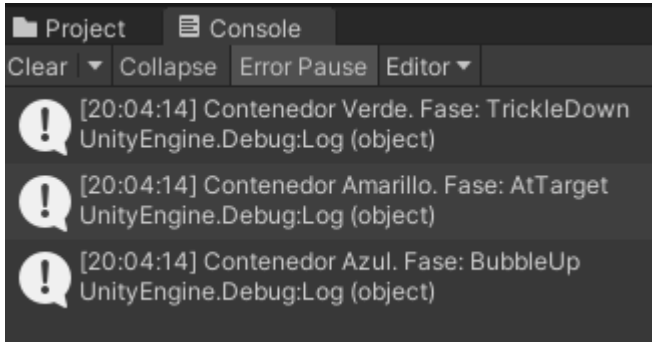
Propagación

```
verde.RegisterCallback<MouseDownEvent>(
    ev =>
    {
        Debug.Log("Contenedor Verde. Fase: " + ev.propagationPhase);
    }, TrickleDown.TrickleDown);

azul.RegisterCallback<MouseDownEvent>(
    ev =>
    {
        Debug.Log("Contenedor Azul. Fase: " + ev.propagationPhase);
    }, TrickleDown.TrickleDown);
amarillo.RegisterCallback<MouseDownEvent>(
    ev =>
    {
        Debug.Log("Contenedor Amarillo. Fase: " + ev.propagationPhase);
    });
```

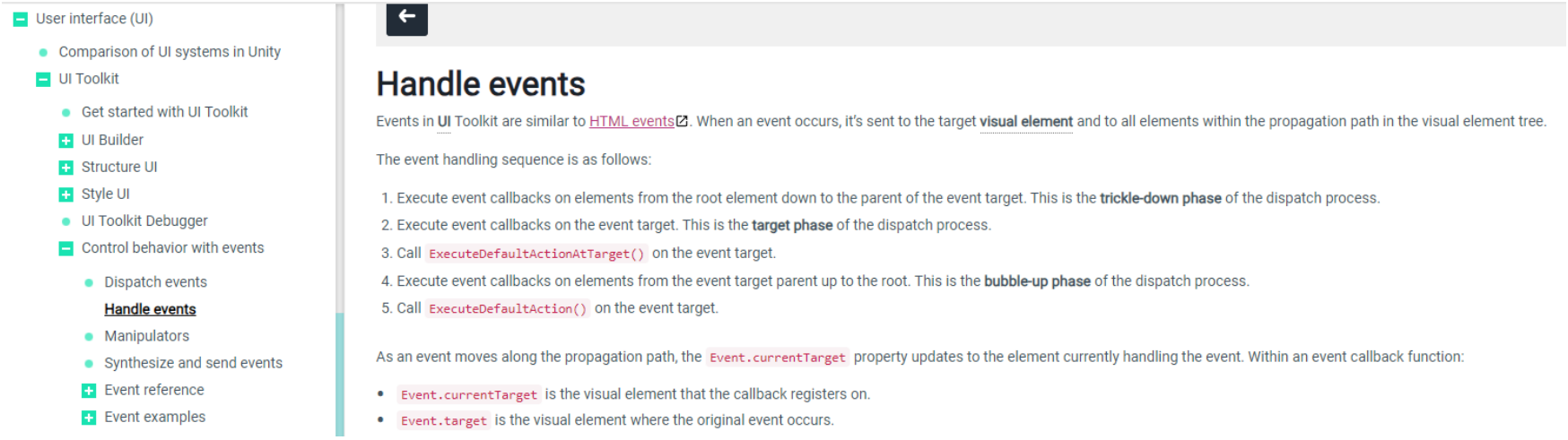


Propagación

```
verde.RegisterCallback<MouseDownEvent>(
    ev =>
    {
        Debug.Log("Contenedor Verde. Fase: " + ev.propagationPhase);
    }, TrickleDown.TrickleDown);

azul.RegisterCallback<MouseDownEvent>(
    ev =>
    {
        Debug.Log("Contenedor Azul. Fase: " + ev.propagationPhase);
    }, TrickleDown.NoTrickleDown);
amarillo.RegisterCallback<MouseDownEvent>(
    ev =>
    {
        Debug.Log("Contenedor Amarillo. Fase: " + ev.propagationPhase);
    });
```



Handle Events

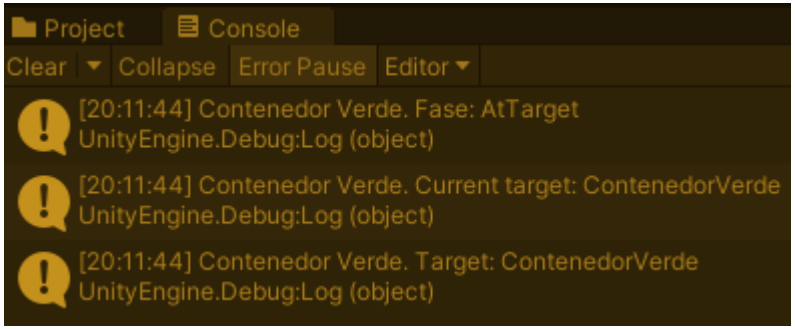


Current Target ←→ Target

```
verde.RegisterCallback<MouseDownEvent>(
    ev =>
    {
        Debug.Log("Contenedor Verde. Fase: " + ev.propagationPhase);
        Debug.Log("Contenedor Verde. Current target: " + (ev.currentTarget as VisualElement).name);
        Debug.Log("Contenedor Verde. Target: " + (ev.target as VisualElement).name);
    }, TrickleDown.TrickleDown);

azul.RegisterCallback<MouseDownEvent>(
    ev =>
    {
        Debug.Log("Contenedor Azul. Fase: " + ev.propagationPhase);
        Debug.Log("Contenedor Verde. Current target: " + (ev.currentTarget as VisualElement).name);
        Debug.Log("Contenedor Verde. Target: " + (ev.target as VisualElement).name);
    }, TrickleDown.TrickleDown);
amarillo.RegisterCallback<MouseDownEvent>(
    ev =>
    {
        Debug.Log("Contenedor Amarillo. Fase: " + ev.propagationPhase);
        Debug.Log("Contenedor Verde. Current target: " + (ev.currentTarget as VisualElement).name);
        Debug.Log("Contenedor Verde. Target: " + (ev.target as VisualElement).name);
    });
```

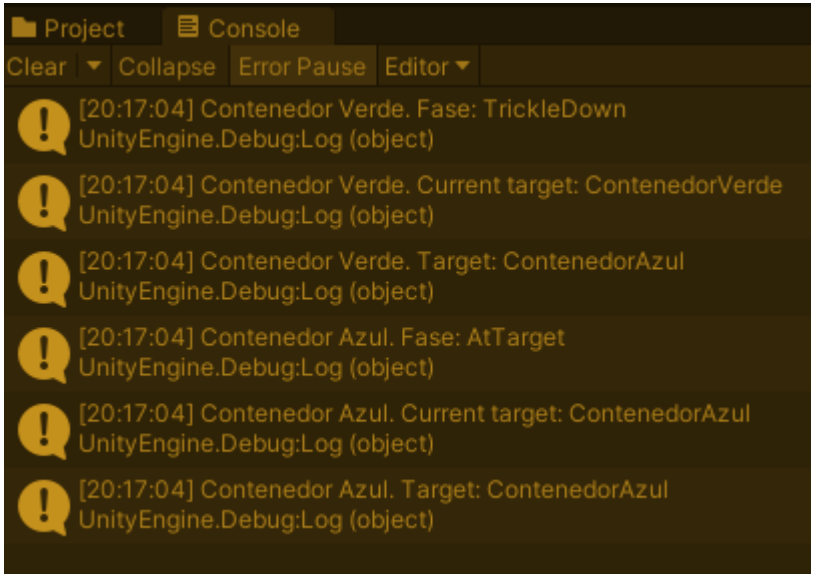
Current Target ←→ Target

- Pulsando en el Verde



Current Target ←→ Target

- Pulsando en el Azul



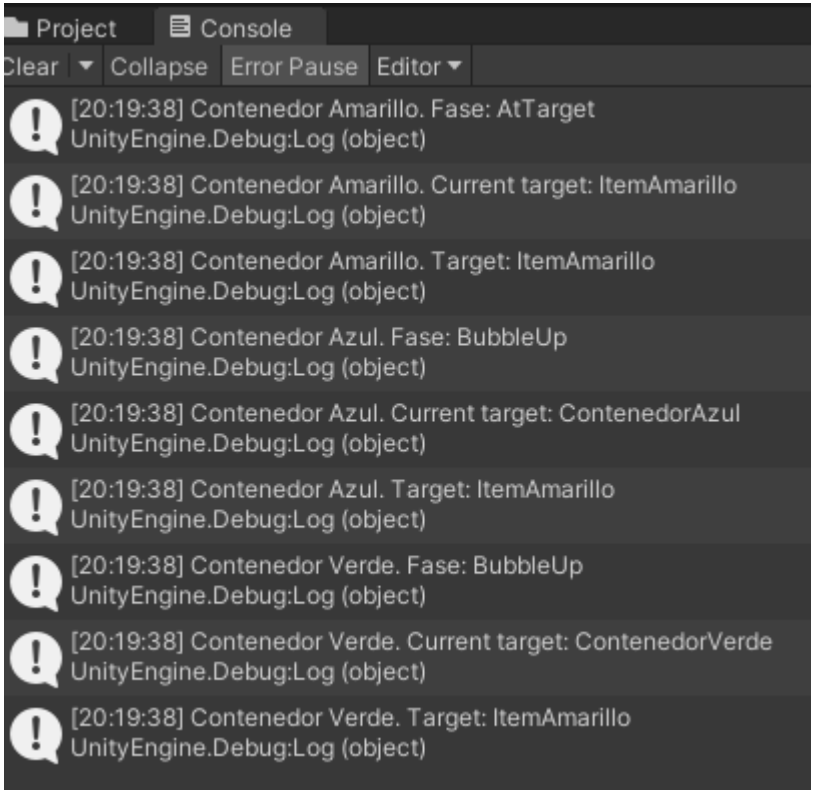
Current Target ←→ Target

- Pulsando en el Amarillo

Current Target ←→ Target

- No Trickledown. Pulsando en el Amarillo

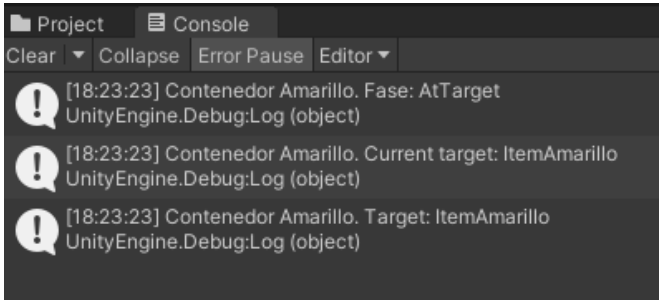
```
verde.RegisterCallback<MouseDownEvent>(
    ev =>
    {
        Debug.Log("Contenedor Verde. Fase: " + ev.propagationPhase);
        Debug.Log("Contenedor Verde. Current target: " + (ev.currentTarget as VisualElement).name);
        Debug.Log("Contenedor Verde. Target: " + (ev.target as VisualElement).name);
    }, TrickleDown.NoTrickleDown);

azul.RegisterCallback<MouseDownEvent>(
    ev =>
    {
        Debug.Log("Contenedor Azul. Fase: " + ev.propagationPhase);
        Debug.Log("Contenedor Azul. Current target: " + (ev.currentTarget as VisualElement).name);
        Debug.Log("Contenedor Azul. Target: " + (ev.target as VisualElement).name);
    }, TrickleDown.NoTrickleDown);
amarillo.RegisterCallback<MouseDownEvent>(
    ev =>
    {
        Debug.Log("Contenedor Amarillo. Fase: " + ev.propagationPhase);
        Debug.Log("Contenedor Amarillo. Current target: " + (ev.currentTarget as VisualElement).name);
        Debug.Log("Contenedor Amarillo. Target: " + (ev.target as VisualElement).name);
    });
```



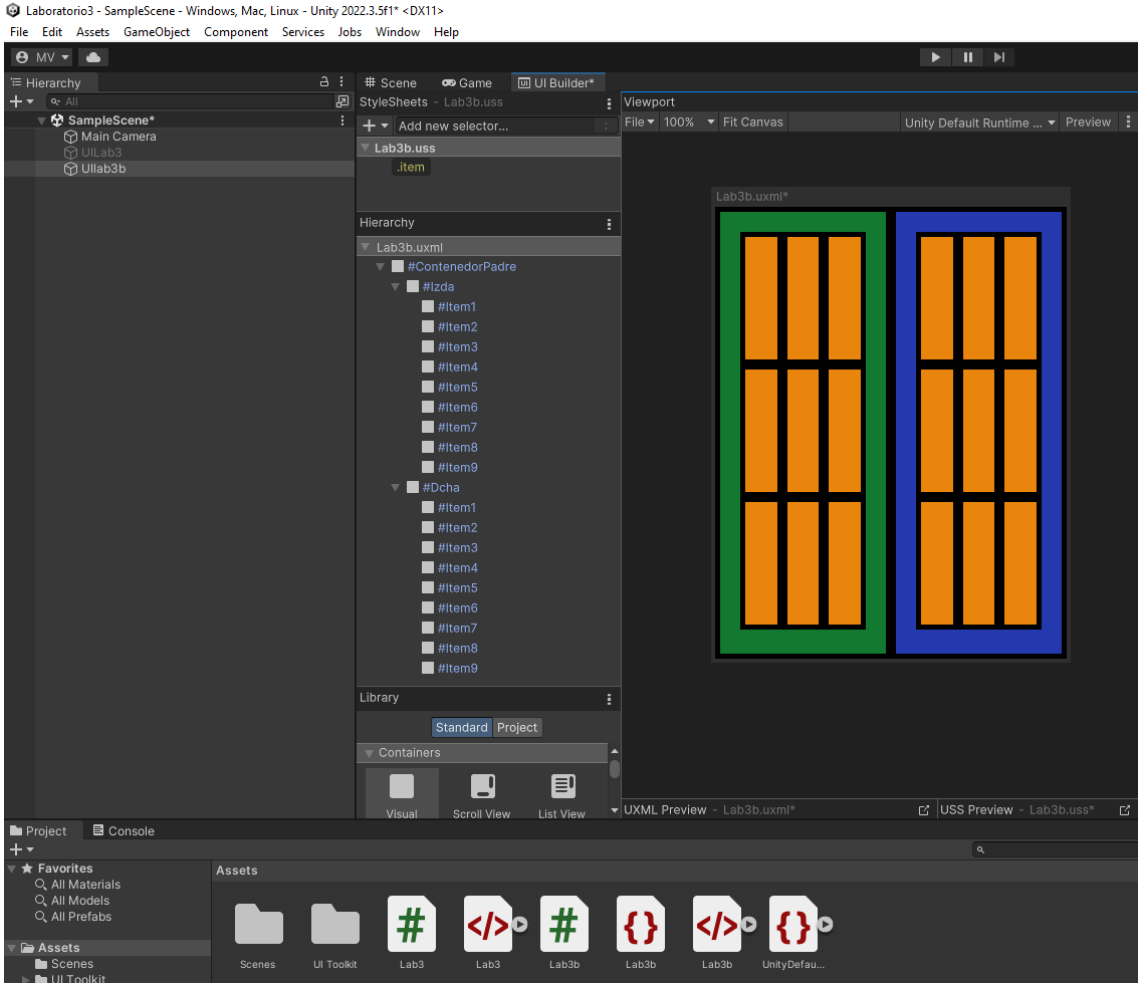
Stop Propagation

- Stop Propagation. Pulsando en el Amarillo

```
verde.RegisterCallback<MouseDownEvent>(
    ev =>
    {
        Debug.Log("Contenedor Verde. Fase: " + ev.propagationPhase);
        Debug.Log("Contenedor Verde. Current target: " + (ev.currentTarget as VisualElement).name);
        Debug.Log("Contenedor Verde. Target: " + (ev.target as VisualElement).name);
    }, TrickleDown.NoTrickleDown);

azul.RegisterCallback<MouseDownEvent>(
    ev =>
    {
        Debug.Log("Contenedor Azul. Fase: " + ev.propagationPhase);
        Debug.Log("Contenedor Azul. Current target: " + (ev.currentTarget as VisualElement).name);
        Debug.Log("Contenedor Azul. Target: " + (ev.target as VisualElement).name);
    }, TrickleDown.NoTrickleDown);
amarillo.RegisterCallback<MouseDownEvent>(
    ev =>
    {
        Debug.Log("Contenedor Amarillo. Fase: " + ev.propagationPhase);
        Debug.Log("Contenedor Amarillo. Current target: " + (ev.currentTarget as VisualElement).name);
        Debug.Log("Contenedor Amarillo. Target: " + (ev.target as VisualElement).name);
        ev.StopPropagation();
    });
```



Modificar propiedades de los elementos



Modificar elementos

Lab3.cs | Lab3b.cs

Assets > Lab3b.cs > ...

```
using UnityEngine;
using UnityEngine.UIElements;

0 references
public class Lab3b : MonoBehaviour
{
    0 references
    private void OnEnable() {
        VisualElement root = GetComponent<UIDocument>().rootVisualElement;

        VisualElement izda  = root.Q("Izda");
        VisualElement dcha  = root.Q("Dcha");


        izda.RegisterCallback<MouseDownEvent>(
            ev =>
            {
                Debug.Log("Contenedor Izquierda. Fase: " + ev.propagationPhase);
                Debug.Log("Contenedor Izquierda. Target: " + (ev.target as VisualElement).name);
            }, TrickleDown.TrickleDown);

        dcha.RegisterCallback<MouseDownEvent>(
            ev =>
            {
                Debug.Log("Contenedor Derecha. Fase: " + ev.propagationPhase);
                Debug.Log("Contenedor Derecha. Target: " + (ev.target as VisualElement).name);
            }, TrickleDown.TrickleDown);
    }
}
```

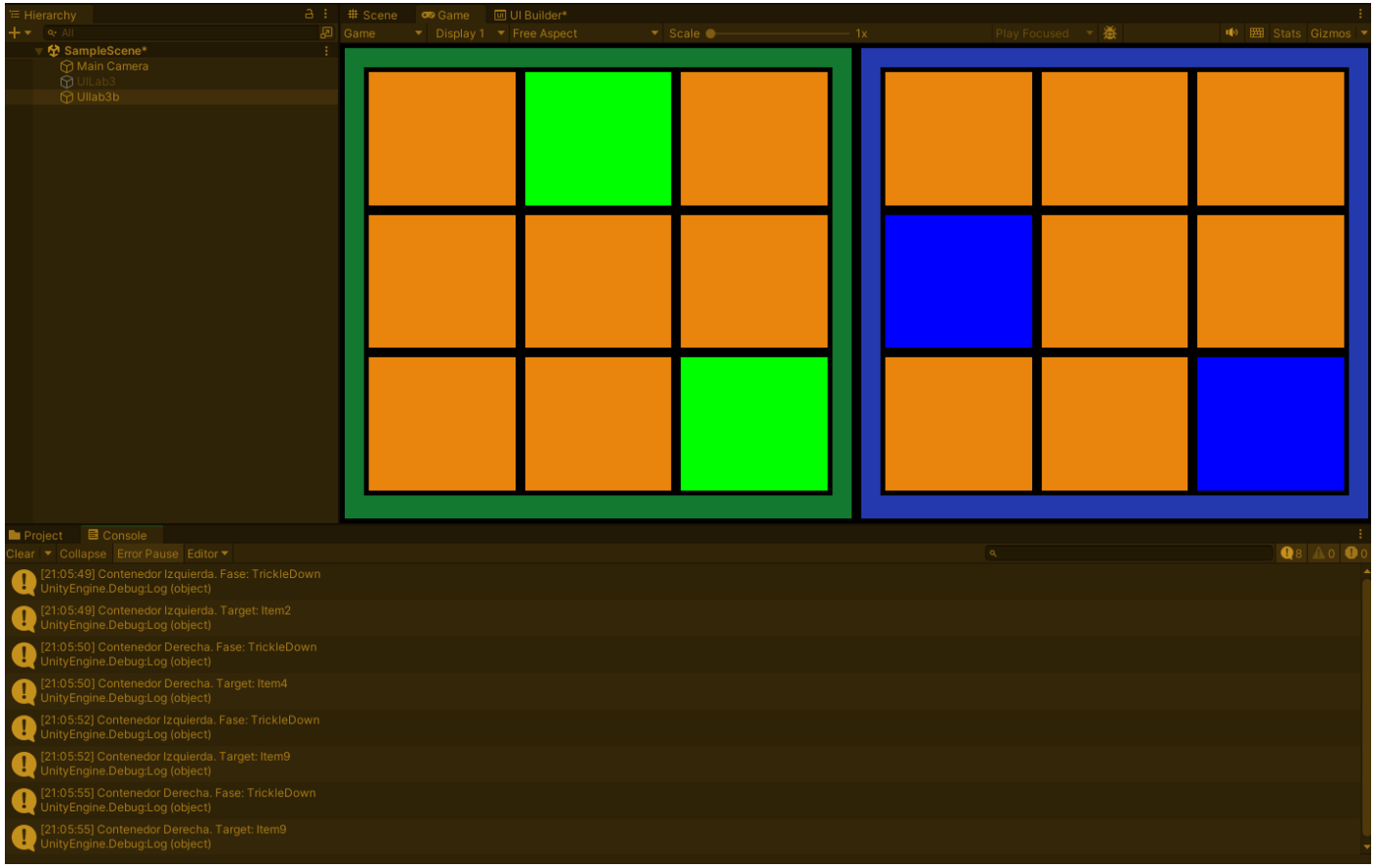
Modificar elementos

Modificar elementos

Assets > Lab3b.cs > ...

```
using UnityEngine;
using UnityEngine.UIElements;

0 references
public class Lab3b : MonoBehaviour
{
    0 references
    private void OnEnable() {
        VisualElement root = GetComponent<UIDocument>().rootVisualElement;

        VisualElement izda  = root.Q("Izda");
        VisualElement dcha  = root.Q("Dcha");

        izda.RegisterCallback<ClickEvent>(
            ev =>
            {
                Debug.Log("Contenedor Izquierda. Fase: " + ev.propagationPhase);
                Debug.Log("Contenedor Izquierda. Target: " + (ev.target as VisualElement).name);
                (ev.target as VisualElement).style.backgroundColor = Color.green;
            }, TrickleDown.TrickleDown);

        dcha.RegisterCallback<ClickEvent>(
            ev =>
            {
                Debug.Log("Contenedor Derecha. Fase: " + ev.propagationPhase);
                Debug.Log("Contenedor Derecha. Target: " + (ev.target as VisualElement).name);
                (ev.target as VisualElement).style.backgroundColor = Color.blue;
            }, TrickleDown.TrickleDown);
    }
}
```

Modificar elementos



UI Toolkit Event System

- Control del comportamiento mediante eventos.
  - Ruta de propagación
  - Manejo de eventos
  - Modificación de Propiedades

- **Manipuladores**

- Activadores

- Unity manual → Dragger

- Unity manual → Resizer

Manipulator

Se utilizan para separar la funcionalidad lógica de los eventos del resto del código de la UI.

Los manipuladores son máquinas de estado que manejan la interacción del usuario con los elementos de la interfaz de usuario. Almacenan, registran y anulan las llamadas a eventos.

Para gestionar eventos, utiliza o hereda de uno de los manipuladores compatibles con UI Toolkit.

Manipulator	Inherits from	Description
`Manipulator`		Base class for all provided manipulators.
`KeyboardNavigationManipulator`	`Manipulator`	Handles translation of device-specific input events to higher-level navigation operations with a keyboard.
`MouseManipulator`	`Manipulator`	Handles mouse input. Has a list of activation filters.
`ContextualMenuManipulator`	`MouseManipulator`	Displays a contextual menu when the user clicks the right mouse button or presses the menu key on the keyboard.
`PointerManipulator`	`MouseManipulator`	Handles pointer input. Has a list of activation filters.
`Clickable`	`PointerManipulator`	Tracks mouse events on an element, and identifies when a click occurs, which is both a pointer press and release on the same element.

Manipulator

Para crear y utilizar un manipulador:

- Define una clase dedicada que herede de una clase ***Manipulator*** soportada por UI Toolkit. Esta clase encapsula la lógica de gestión de eventos adaptada a la interacción específica que se desea gestionar.

- Dentro de la clase, se implementa métodos para responder a interacciones relevantes, como clics del ratón o arrastres.

- La clase manipuladora se adjunta al elemento de UI de destino. Este adjunto permite al manipulador interceptar y gestionar los eventos especificados (interacciones del usuario) mientras mantiene una clara separación con el resto del código de la UI.

Manipulator

Assets > Lab3Manipulator.cs > ...

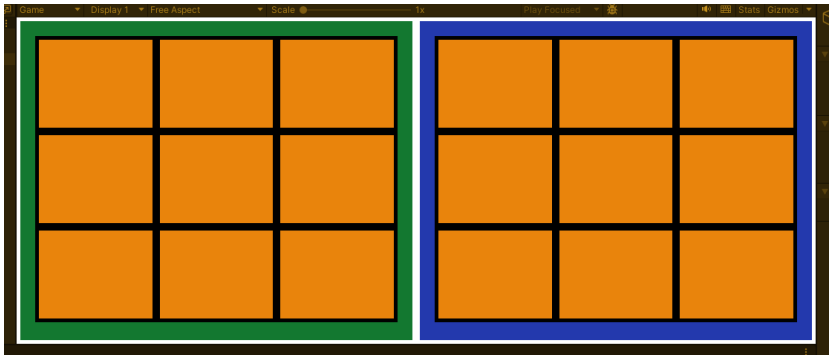
```
1   using UnityEngine;
2   using UnityEngine.UIElements;
3
    0 references
4   public class Lab3Manipulator : Manipulator
5   {
        0 references
6       protected override void RegisterCallbacksOnTarget()
7       {
8
9       }
10
        0 references
11      protected override void UnregisterCallbacksFromTarget()
12      {
13
14      }
15  }
```

Manipulator

Assets > Lab3Manipulator.cs > ...

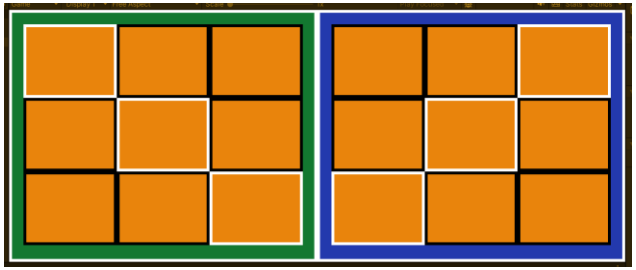
```
1   using UnityEngine;
2   using UnityEngine.UIElements;
3
    2 references
4   public class Lab3Manipulator : Manipulator
5   {
        0 references
6       protected override void RegisterCallbacksOnTarget()
7       {
8           target.RegisterCallback<MouseDownEvent>(OnMouseDown);
9       }
10
        0 references
11      protected override void UnregisterCallbacksFromTarget()
12      {
13          target.UnregisterCallback<MouseDownEvent>(OnMouseDown);
14      }
15
        2 references
16      private void OnMouseDown(MouseDownEvent mev)
17      {
18          target.style.borderBottomColor = Color.white;
19          target.style.borderLeftColor   = Color.white;
20          target.style.borderRightColor  = Color.white;
21          target.style.borderTopColor    = Color.white;
22          mev.StopPropagation();
23      }
24  }
25
```

```
3
    0 references
4   public class Lab3b : MonoBehaviour
5   {
        0 references
6       private void OnEnable() {
7           VisualElement root = GetComponent<UIDocument>().rootVisualElement;
8
9           VisualElement izda  = root.Q("Izda");
10          VisualElement dcha  = root.Q("Dcha");
11
12          izda.AddManipulator(new Lab3Manipulator());
13          dcha.AddManipulator(new Lab3Manipulator());
14
```



Manipulator

```
VisualElement izda  = root.Q("Izda");
VisualElement dcha  = root.Q("Dcha");

izda.AddManipulator(new Lab3Manipulator());
dcha.AddManipulator(new Lab3Manipulator());

List<VisualElement> lveizda = izda.Children().ToList();
List<VisualElement> lvedcha = dcha.Children().ToList();

lveizda.ForEach(elem => elem.AddManipulator(new Lab3Manipulator()));
lvedcha.ForEach(elem => elem.AddManipulator(new Lab3Manipulator()));
```



UI Toolkit Event System

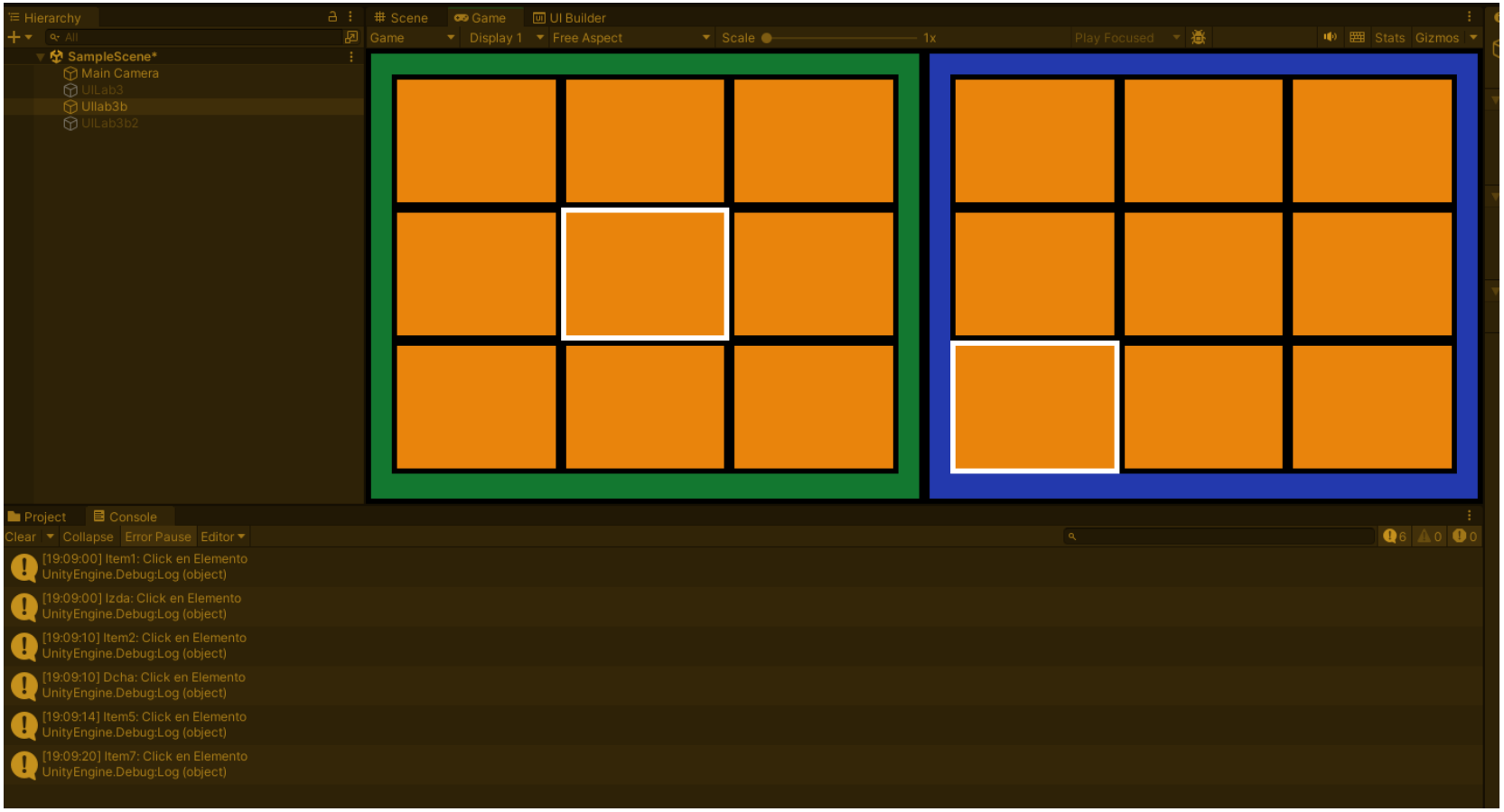
- Control del comportamiento mediante eventos.
  - Ruta de propagación
  - Manejo de eventos
  - Modificación de Propiedades
- Manipuladores
- **Activadores**
- Unity manual → Dragger
- Unity manual → Resizer

Activators

Assets > Lab3Manipulator.cs > ...

```
1   using UnityEngine;
2   using UnityEngine.UIElements;
3
    5 references
4   public class Lab3Manipulator : MouseManipulator
5   {
        4 references
6       public Lab3Manipulator()
7       {
8           activators.Add(new ManipulatorActivationFilter{button = MouseButton.RightMouse});
9       }
        0 references
10      protected override void RegisterCallbacksOnTarget()
11      {
12          target.RegisterCallback<MouseDownEvent>(OnMouseDown);
13      }
14
        0 references
15      protected override void UnregisterCallbacksFromTarget()
16      {
17          target.UnregisterCallback<MouseDownEvent>(OnMouseDown);
18      }
19
        2 references
20      private void OnMouseDown(MouseDownEvent mev)
21      {
22          Debug.Log(target.name + ": Click en Elemento");
23          if(CanStartManipulation(mev))
24          {
25              target.style.borderBottomColor = Color.white;
26              target.style.borderLeftColor   = Color.white;
27              target.style.borderRightColor  = Color.white;
28              target.style.borderTopColor    = Color.white;
29              mev.StopPropagation();
30          }
31
32      }
33  }
```

Activators


Hierarchy
SampleScene*
Main Camera
UILab3
Ullab3b
UILab3b2
Scene
Game
UI Builder
Game
Display 1
Free Aspect
Scale
1x
Play Focused
Stats
Gizmos
Project
Console
Clear
Collapse
Error Pause
Editor
[19:09:00] Item1: Click en Elemento
UnityEngine.Debug:Log (object)
[19:09:00] Izda: Click en Elemento
UnityEngine.Debug:Log (object)
[19:09:10] Item2: Click en Elemento
UnityEngine.Debug:Log (object)
[19:09:10] Dcha: Click en Elemento
UnityEngine.Debug:Log (object)
[19:09:14] Item5: Click en Elemento
UnityEngine.Debug:Log (object)
[19:09:20] Item7: Click en Elemento
UnityEngine.Debug:Log (object)

UI Toolkit Event System

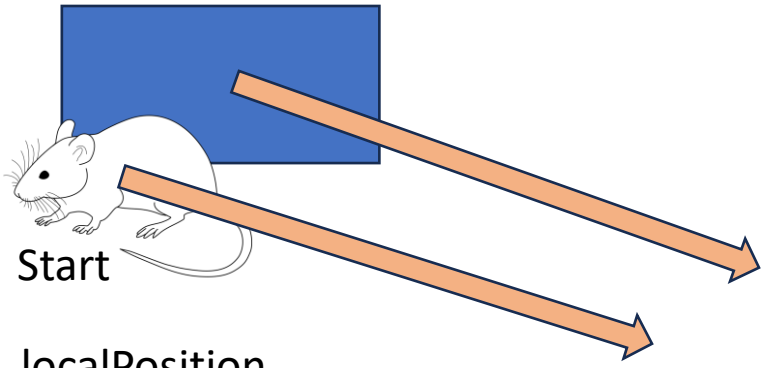
- Control del comportamiento mediante eventos.
  - Ruta de propagación
  - Manejo de eventos
  - Modificación de Propiedades
- Manipuladores
- Activadores
- **Unity manual → Dragger**
- Unity manual → Resizer

Unity Manual → ExampleDragger

```
using UnityEngine;
using UnityEngine.UIElements;

public class ExampleDragger : PointerManipulator
{
    private Vector3 m_Start;
    protected bool m_Active;
    private int m_PointerId;
    private Vector2 m_StartSize;

    public ExampleDragger()
    {
        m_PointerId = -1;
        activators.Add(new ManipulatorActivationFilter { button = MouseButton.LeftMouse });
        m_Active = false;
    }

    protected override void RegisterCallbacksOnTarget()
    {
        target.RegisterCallback<PointerDownEvent>(OnPointerDown);
        target.RegisterCallback<PointerMoveEvent>(OnPointerMove);
        target.RegisterCallback<PointerUpEvent>(OnPointerUp);
    }

    protected override void UnregisterCallbacksFromTarget()
    {
        target.UnregisterCallback<PointerDownEvent>(OnPointerDown);
        target.UnregisterCallback<PointerMoveEvent>(OnPointerMove);
        target.UnregisterCallback<PointerUpEvent>(OnPointerUp);
    }
```

Unity Manual → ExampleDragger

```
protected void OnPointerDown(PointerDownEvent e)
{
    if (m_Active)
    {
        e.StopImmediatePropagation();
        return;
    }

    if (CanStartManipulation(e))
    {
        m_Start = e.localPosition;
        m_PointerId = e.pointerId;

        m_Active = true;
        target.CapturePointer(m_PointerId);
        e.StopPropagation();
    }
}
```



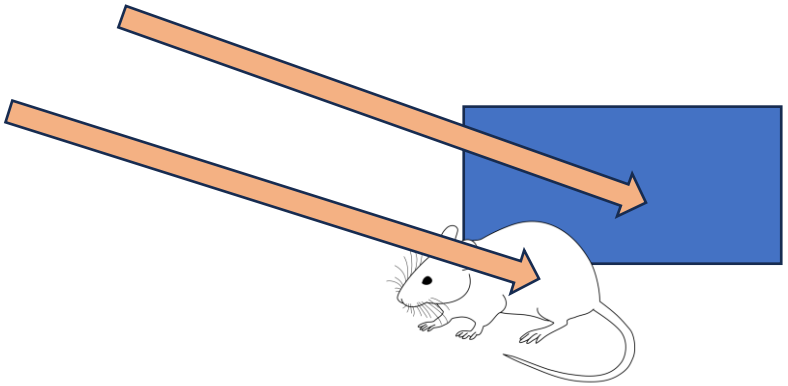
localPosition

localMousePosition

Para evitar parar si el ratón sale del visual element.
Target.CaptureMouse();

Unity Manual → ExampleDragger

```
    protected void OnPointerMove(PointerMoveEvent e)
    {
        if (!m_Active || !target.HasPointerCapture(m_PointerId))
            return;

        Vector2 diff = e.localPosition - m_Start;

        target.style.top = target.layout.y + diff.y;
        target.style.left = target.layout.x + diff.x;

        e.StopPropagation();
    }

    protected void OnPointerUp(PointerUpEvent e)
    {
        if (!m_Active || !target.HasPointerCapture(m_PointerId) || !CanStopManipulation(e))
            return;

        m_Active = false;
        target.ReleaseMouse();
        e.StopPropagation();
    }
}
```



UI Toolkit Event System

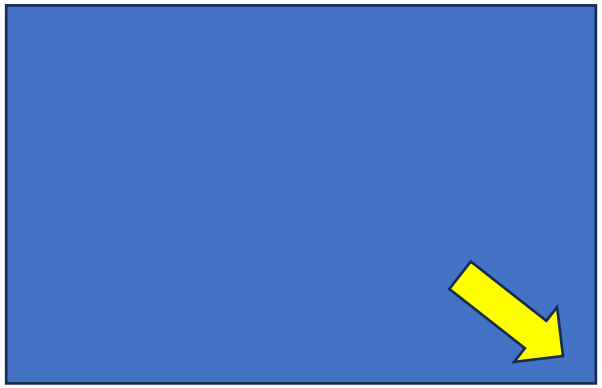
- Control del comportamiento mediante eventos.
  - Ruta de propagación
  - Manejo de eventos
  - Modificación de Propiedades
- Manipuladores
- Activadores
- Unity manual → Dragger
- **Unity manual → Resizer**

Unity Manual → Example Resizer

```
using UnityEngine;
using UnityEngine.UIElements;

public class ExampleResizer : PointerManipulator
{
    private Vector3 m_Start;
    protected bool m_Active;
    private int m_PointerId;
    private Vector2 m_StartSize;
    public ExampleResizer()
    {
        m_PointerId = -1;
        activators.Add(new ManipulatorActivationFilter { button = MouseButton.LeftMouse });
        m_Active = false;
    }

    protected override void RegisterCallbacksOnTarget()
    {
        target.RegisterCallback<PointerDownEvent>(OnPointerDown);
        target.RegisterCallback<PointerMoveEvent>(OnPointerMove);
        target.RegisterCallback<PointerUpEvent>(OnPointerUp);
    }

    protected override void UnregisterCallbacksFromTarget()
    {
        target.UnregisterCallback<PointerDownEvent>(OnPointerDown);
        target.UnregisterCallback<PointerMoveEvent>(OnPointerMove);
        target.UnregisterCallback<PointerUpEvent>(OnPointerUp);
    }
```

Unity Manual → Example Resizer

```
protected void OnPointerDown(PointerDownEvent e)
{
    if (m_Active)
    {
        e.StopImmediatePropagation();
        return;
    }

    if (CanStartManipulation(e))
    {
        m_Start = e.localPosition;
        m_StartSize = target.layout.size;
        m_PointerId = e.pointerId;
        m_Active = true;
        target.CapturePointer(m_PointerId);
        e.StopPropagation();
    }
}
```

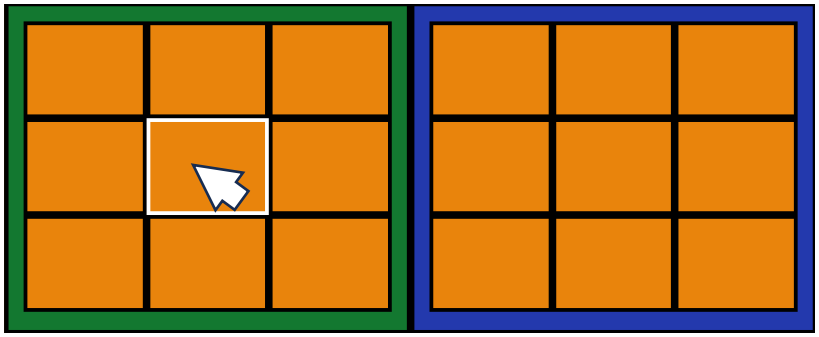


Unity Manual → Example Resizer

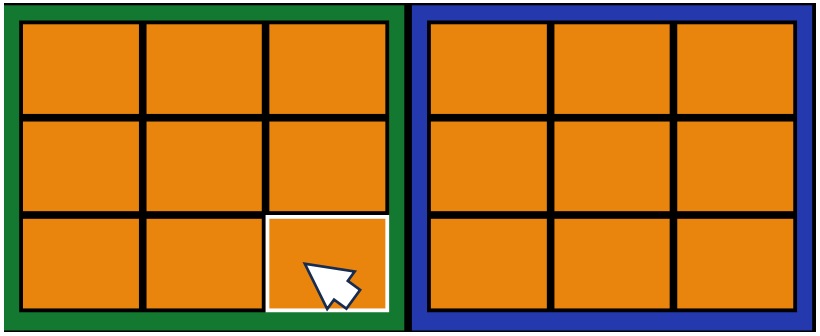
```
    protected void OnPointerMove(PointerMoveEvent e)
    {
        if (!m_Active || !target.HasPointerCapture(m_PointerId))
            return;

        Vector2 diff = e.localPosition - m_Start;

        target.style.height = m_StartSize.y + diff.y;
        target.style.width = m_StartSize.x + diff.x;

        e.StopPropagation();
    }

    protected void OnPointerUp(PointerUpEvent e)
    {
        if (!m_Active || !target.HasPointerCapture(m_PointerId) || !CanStopManipulation(e))
            return;

        m_Active = false;
        target.ReleasePointer(m_PointerId);
        m_PointerId = -1;
        e.StopPropagation();
    }
}
```

Ejercicios

- 1. Realizar los ejercicios del tutorial.



- 2. Selección de un elemento al pasar el ratón por él (deseleccionando todos los demás).



- 3. Modificar el resizer para que funcione con la rueda del ratón. Se selecciona manteniendo pulsado el botón izquierdo. El tamaño cambia en todas las direcciones.