Contents

BudInGui Documentation

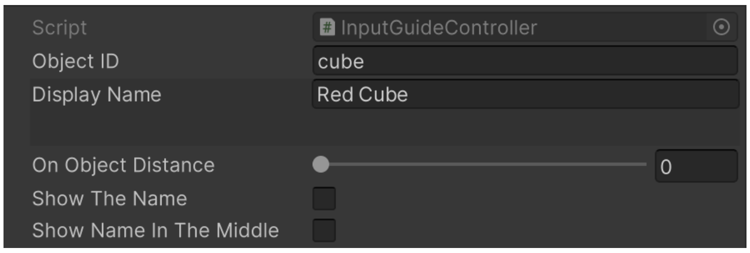
1.Getting Started

The **BudInGui-Inputs, Guides, and Events System** facilitates the assignment of events to inputs, with parameters that are both multiple and editable via the inspector. These events can be executed based on specific conditions, and the parameters for these condition functions are also adjustable from the inspector. If necessary, the inputs for these events can be displayed anywhere within the scene view as a guide UI. Additionally, guide texts can be customized to accommodate various localizations. All of these configurations can be managed exclusively through the inspector.

2.Implementation

First, locate the prefab named **'InputGuideUIDisplay'** in the directory **'Assets/BudInGui/InputGuideSystem/Prefabs/UIElements/'** and drag it into the scene. Subsequently, add the **'InputGuideController'** as a component to the object.

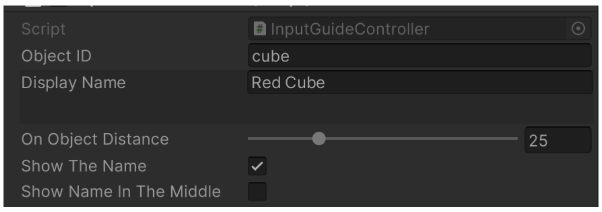
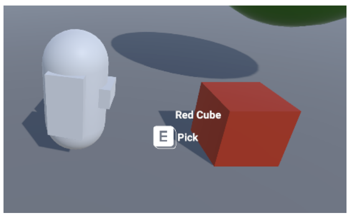
2.1.First Setup

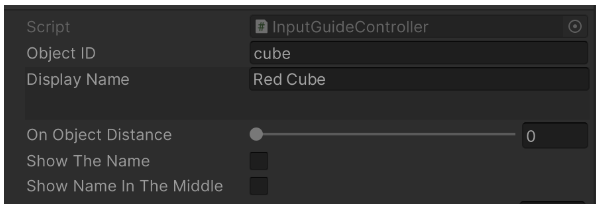
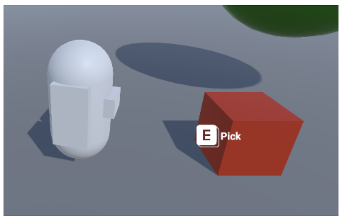
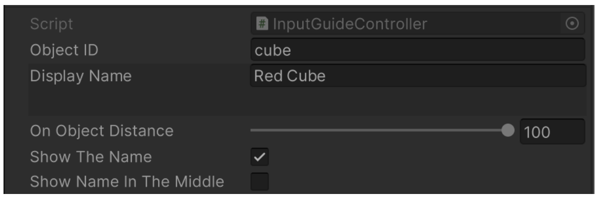
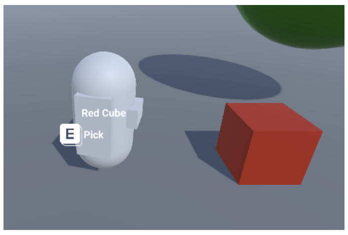


Script	InputGuideController
Object ID	cube
Display Name	Red Cube
On Object Distance	0
Show The Name	☐
Show Name In The Middle	☐

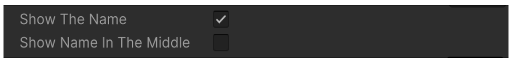
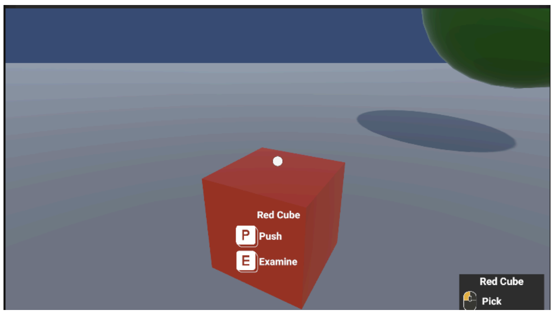
This section enables the identification of objects before assigning inputs, events, and guides.

- **Object ID**: Serves to define the object type and ensure unique recognition during interactions, functioning similarly to a Unity Tag.
- **Display Name**: Specifies the name to be displayed in the UI for associated guides.
- **On Object Distance**: Applicable to guides with the **'On Object'** type selected in the **'Place On Screen'** property. To enable the object to recognize the player, the **'InputGuidePlayerReference'** component must be added to the player object.

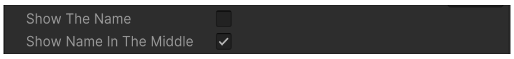
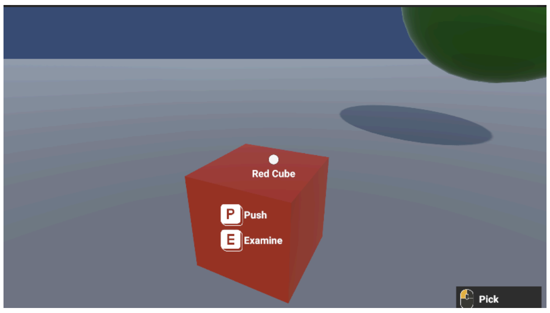
On Object Distance Value	View Of Result
 Script: InputGuideController; Object ID: cube; Display Name: Red Cube; On Object Distance: 25; Show The Name: ✓; Show Name In The Middle: ☐	

 Script InputGuideController Object ID cube Display Name Red Cube On Object Distance 0 Show The Name ☐ Show Name In The Middle ☐	
 Script InputGuideController Object ID cube Display Name Red Cube On Object Distance 100 Show The Name ✓ Show Name In The Middle ☐	

- **Show Name**: Enable this option to display the object's name above the guide elements.

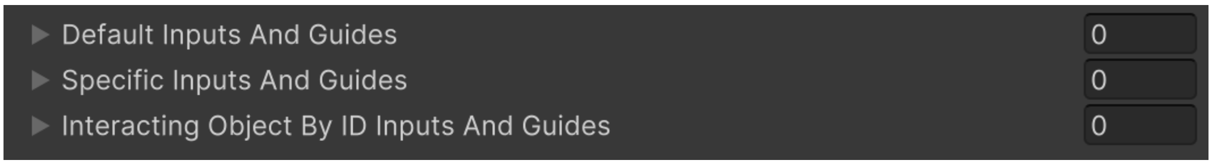
When 'Show The Name' enabled	**View Of Result**
 Show The Name ✓ Show Name In The Middle ☐	

- **Show Name in the Middle**: Enable this option to display the object's name at the center of the screen view.

When 'Show Name In The Middle' enabled	**View Of Result**
 Show The Name ☐ Show Name In The Middle ✓	

2.2.Inputs & Guide TABs

Specifies where inputs and guides are defined.



▶ Default Inputs And Guides 0
▶ Specific Inputs And Guides 0
▶ Interacting Object By ID Inputs And Guides 0

2.2.1.Default Inputs And Guides

Refers to guides that are activated without any specific circumstances.

2.2.1.1.Scripting API

-ActivateInputsAndGuides

Declaration

public void ActivateInputsAndGuides();

Description

Provides an option to activate the default inputs and guides.

Usage

GetComponent<InputGuideController>().**ActivateInputsAndGuides();**

-DisableDefaultInputsAndGuides

Declaration

public void DisableDefaultInputsAndGuides(bool removeReferenceObject = false);

Parameters

- removeReferenceObject - If set to true, the reference object will be removed, if there is.

Description

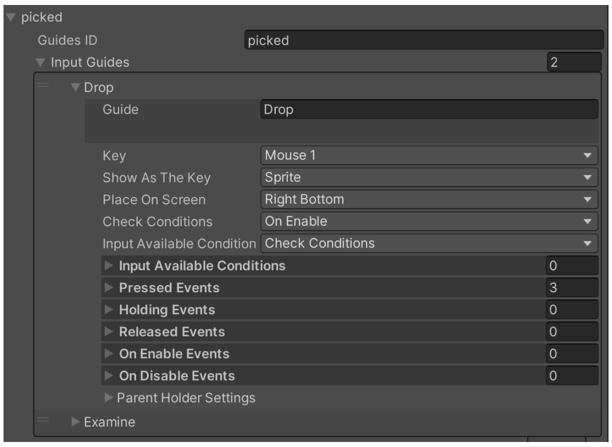
Provides an option to deactivate the default inputs and guides.

Usage

GetComponent<InputGuideController>().**DisableDefaultInputsAndGuides();**
GetComponent<InputGuideController>().**DisableDefaultInputsAndGuides(bool);**

2.2.2.Specific Inputs And Guides

Allows the creation of custom guides based on specific statuses. For instance, if an input is required to drop an object after it is picked up, a **‘Guides ID’** named ‘picked’ should be defined in this section. The necessary inputs can then be added under the ‘picked’ section and activated when the object is picked up.



```
▼ picked
    Guides ID                         picked
  ▼ Input Guides                      2
      ▼ Drop
          Guide                       Drop
          Key                         Mouse 1
          Show As The Key             Sprite
          Place On Screen             Right Bottom
          Check Conditions            On Enable
          Input Available Condition   Check Conditions
          ▶ Input Available Conditions  0
          ▶ Pressed Events              3
          ▶ Holding Events              0
          ▶ Released Events             0
          ▶ On Enable Events            0
          ▶ On Disable Events           0
          ▶ Parent Holder Settings
      ▶ Examine
```

2.2.2.1.Scripting API

-ActivateInputsAndGuides

Declaration

```
public void ActivateInputsAndGuides(string specificGuideID, bool activateAlsoDefaults = false);
```

Parameters

- specificGuideID – Defines the specific inputs and guides associated with this name.
- activateAlsoDefaults – If set to true, default inputs and guides will also be activated.

Description

Enables inputs and guides associated with a designated string identifier.

Usage

```
GetComponent<InputGuideController>().ActivateInputsAndGuides(string);
GetComponent<InputGuideController>().ActivateInputsAndGuides(string,bool);
```

-DisableSpecificInputsAndGuides

Declaration

```
public void DisableSpecificInputsAndGuides(string specificGuideID = null, bool disableAlsoDefaults = false);
```

Parameters

- specificGuideID – Defines the specific inputs and guides associated with this name. If not specified, all specific inputs and guides will be disabled.
- disableAlsoDefaults – If set to true, default inputs and guides will also be disabled.

Description

Disables inputs and guides linked to a designated string identifier.

Usage

```
GetComponent<InputGuideController>().DisableSpecificInputsAndGuides();
GetComponent<InputGuideController>().DisableSpecificInputsAndGuides(string);
GetComponent<InputGuideController>().DisableSpecificInputsAndGuides(string,bool);
```

2.2.3.Interacting Object By ID Inputs And Guides

Establishes connections between the object and others it can interact with. To use this feature, the input guide of the interacting object must be activated with the parameter of the target object.

2.2.3.1.Scripting API

-ActivateInputsAndGuides

Declaration

public void ActivateInputsAndGuides(GameObject interactedObject, bool saveObjAsReference = false, bool activateAlsoDefaults = false);

Parameters

- interactedObject – The object to interact with.
- saveObjAsReference – If set to true, the '**interactedObject'** will be saved as the reference object.
- activateAlsoDefaults – If set to true, default inputs and guides will also be activated.

Description

Activates inputs and guides if the **'interactedObject**' contains an **'InputGuideController'** and its **'ObjectID'** matches one specified in this section.

Usage

GetComponent<InputGuideController>().**ActivateInputsAndGuides(GameObject);**
GetComponent<InputGuideController>().**ActivateInputsAndGuides(GameObject,bool);**
GetComponent<InputGuideController>().**ActivateInputsAndGuides(GameObject,bool,bool
);**

-DisableInteractingInputsAndGuides

Declaration

public void DisableInteractingInputsAndGuides(GameObject interactedObject = null, bool removeReferenceObject = true, bool disableAlsoDefaults = false);

Parameters

- interactedObject – The object to interact with.
- removeReferenceObject – If set to true, the reference object associated with the '**interactedObject'** will be removed, if it saved as a reference object before.
- disableAlsoDefaults – If set to true, default inputs and guides will also be removed.

Description

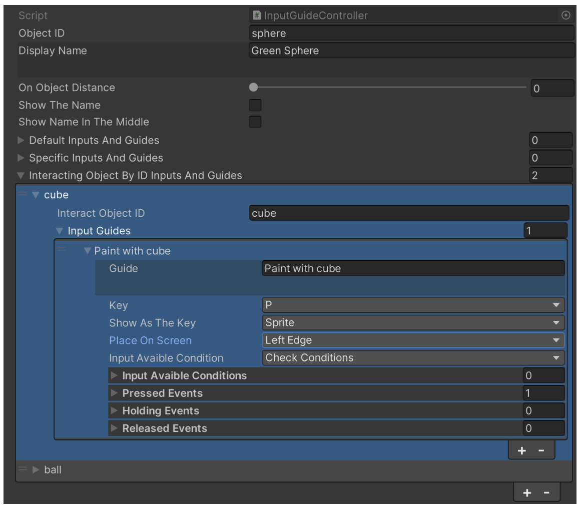
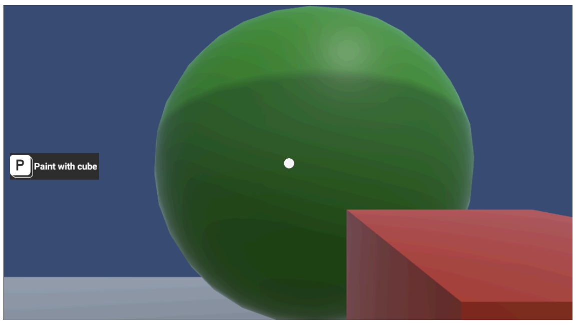
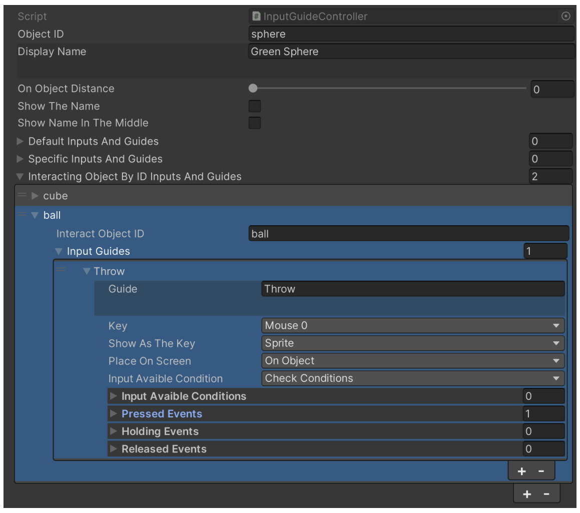
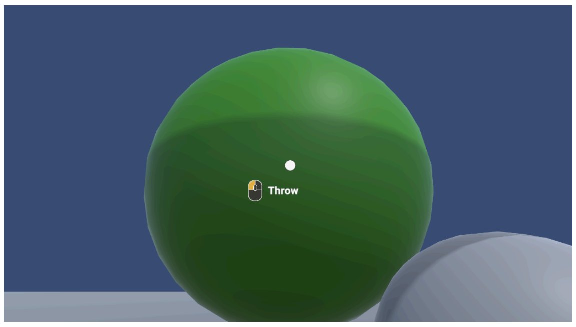
Disables inputs and guides associated with the defined **'interactedObject'**.

Usage

GetComponent<InputGuideController>().**DisableInteractingInputsAndGuides(GameObject
);**
GetComponent<InputGuideController>().**DisableInteractingInputsAndGuides(GameObject
,bool);**
GetComponent<InputGuideController>().**DisableInteractingInputsAndGuides(GameObject
,bool,bool);**

2.2.3.2.Example Scenario

If a player holds a cube and approaches a sphere, they may want to paint the sphere. If they hold a ball, they may want to throw it at the sphere. To achieve this, the IDs of the cube and ball should be specified in the sphere's ‘InputGuideController’ under the ‘*Interacting Object by ID Inputs and Guides’* section. After defining the desired actions, calling the activate method with one of these game objects will suffice. If the game object contains an ‘InputGuideController’ and its ID matches one defined in the sphere's ‘InputGuideController’, the inputs and guides in this section will activate, and the associated events will begin to listen.

Guide	**View Of Result**
 Script InputGuideController Object ID sphere Display Name Green Sphere On Object Distance 0 Show The Name Show Name In The Middle Default Inputs And Guides 0 Specific Inputs And Guides 0 Interacting Object By ID Inputs And Guides 2 cube Interact Object ID cube Input Guides 1 Paint with cube Guide Paint with cube Key P Show As The Key Sprite Place On Screen Left Edge Input Avaible Condition Check Conditions Input Avaible Conditions 0 Pressed Events 1 Holding Events 0 Released Events 0 ball	
 Script InputGuideController Object ID sphere Display Name Green Sphere On Object Distance 0 Show The Name Show Name In The Middle Default Inputs And Guides 0 Specific Inputs And Guides 0 Interacting Object By ID Inputs And Guides 2 cube ball Interact Object ID ball Input Guides 1 Throw Guide Throw Key Mouse 0 Show As The Key Sprite Place On Screen On Object Input Avaible Condition Check Conditions Input Avaible Conditions 0 Pressed Events 1 Holding Events 0 Released Events 0	

2.2.4.Other Scripting APIs

-DisableAllInputsAndGuides

Declaration

`public void DisableAllInputsAndGuides(bool removeReferenceObject = false);`

Parameters

- removeReferenceObject - If set to true, the reference object will be removed, if there is.

Description

Disables all inputs and guides associated with this object.

Usage

```
GetComponent<InputGuideController>().DisableAllInputsAndGuides();
GetComponent<InputGuideController>().DisableAllInputsAndGuides(bool);
```

-SetReferenceObject

Declaration

`public void SetReferenceObject(GameObject refObj);`

Parameters

- refObj - The object will be designated as a reference for this purpose.

Description

Specifies an object that can be used as a reference for events and functions.

Usage

```
GetComponent<InputGuideController>().SetReferenceObject(GameObject);
```

-RemoveReferenceObject

Declaration

`public void RemoveReferenceObject();`

Description

Removes the defined reference object, making it unavailable for use in events and functions.

Usage

```
GetComponent<InputGuideController>().RemoveReferenceObject();
```

-SetMiddleDot

Declaration

`public void SetMiddleDot(bool setActive);`

Parameters

- setActive – Toggles the visibility of the dot at the center of the screen.

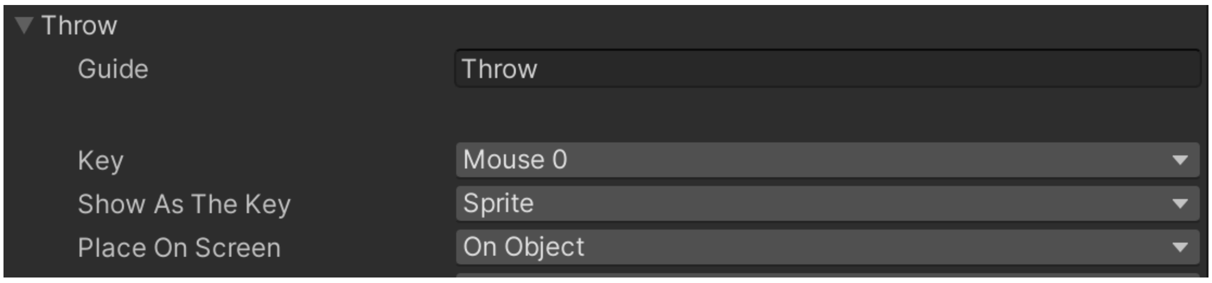
Description

Activates a dot at the center of the screen, commonly used in first-person games. This dot can also be customized via the **'UIDisplay'** component of the **'InputGuideUIDisplay'** prefab.

Usage

```
UIDisplay.Instance.SetMiddleDot(bool);
```

2.3.Input Guide

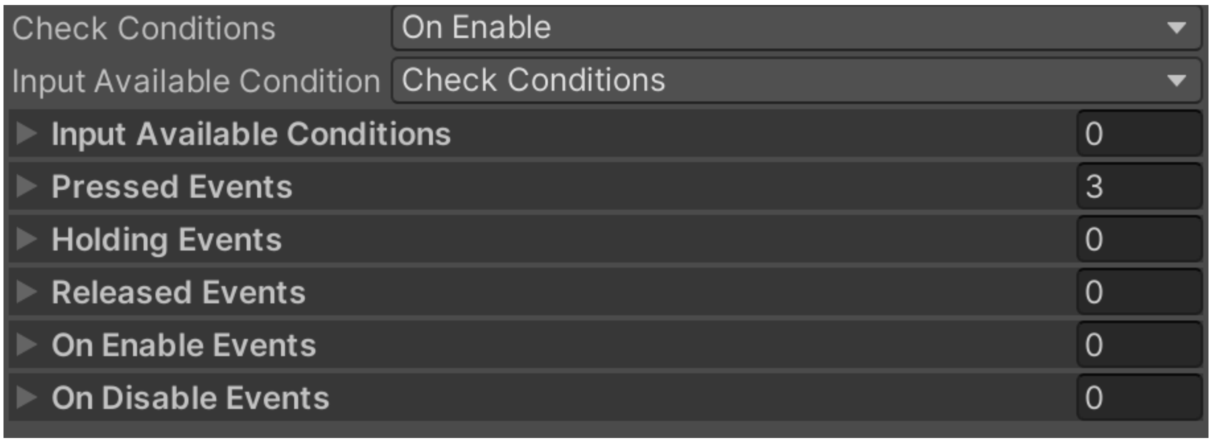

Throw
Guide: Throw
Key: Mouse 0
Show As The Key: Sprite
Place On Screen: On Object

Defines the actions an object will perform in response to a specified key.

- Begin by writing a brief description to serve as a guide on the interface.
- Specify the key that will trigger the associated events.
- Choose how the key will be displayed in the guide: as text or as a sprite.
- Determine where the guide will appear on the screen by selecting an option from the '***Place on Screen'*** section:
    - **OnObject**
    - **Middle**
    - **RightBottom**
    - **LeftBottom**
    - **RightTop**
    - **LeftTop**
    - **RightEdge**
    - **LeftEdge**
    - **BottomEdge**
    - **TopEdge**
    - **DontDisplayUI**

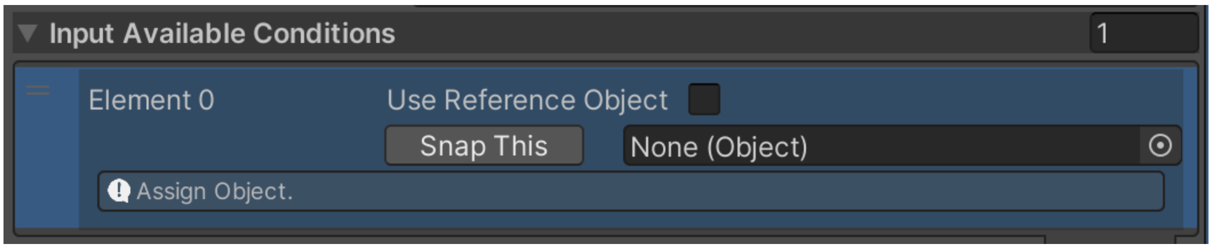
If the '***DontDisplayUI'*** option is selected, the events will continue to be monitored in the background, but the guide for this input will not be displayed in the scene.

2.3.1.Conditions And Events

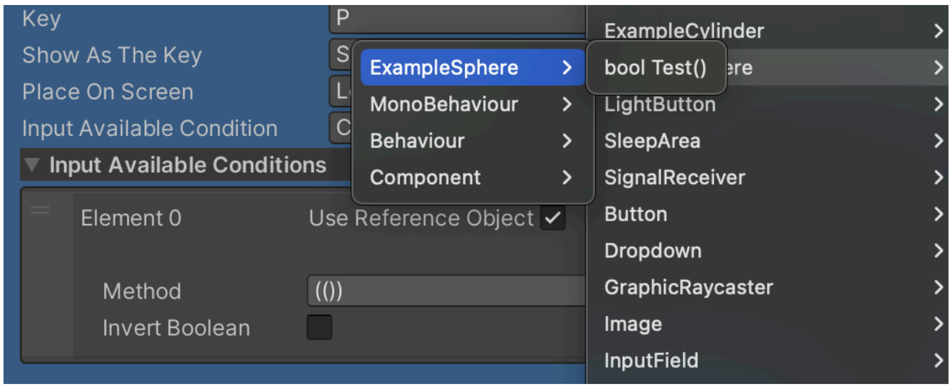

Check Conditions On Enable
Input Available Condition Check Conditions
Input Available Conditions 0
Pressed Events 3
Holding Events 0
Released Events 0
On Enable Events 0
On Disable Events 0

- **Check Conditions:** This option allows for greater control over when input availability conditions are evaluated. It includes two modes:
  - **On Enable**:
    In this mode, the conditions are evaluated only once, at the moment the guide is enabled.
  - **On Enable And Every Update**:
    In this mode, the conditions are evaluated both when the guide is enabled and continuously on every update. This ensures that the guide and events respond dynamically to changes in the conditions. If the conditions switch between **true** and **false**, the guide will appear or disappear accordingly, and the associated events will start or stop listening in real time.
- **Input Available Conditions:**
  Only functions returning boolean values can be defined here. If the '***Check Conditions'*** option is selected in the '***Input Available Conditions'*** section, the defined events will be triggered only if all specified conditions return **true**. If any condition returns **false**, the guide will not appear, and the events will not be executed.
  - If no function is defined, but the '***Check Conditions'*** option is enabled, the condition automatically defaults to **true**, causing the guide to display and the events to be executed.
- **Pressed Events:**
  Executes the assigned events when the specified key is pressed.
- **Holding Events:**
  Executes the assigned events on every frame while the specified key is held down.
- **Released Events:**
  Executes the assigned events when the specified key is released.
- **On Enable Events:**
  Executes the assigned events when the guide is enabled.
- **On Disable Events:**
  Executes the assigned events when the guide is disabled.

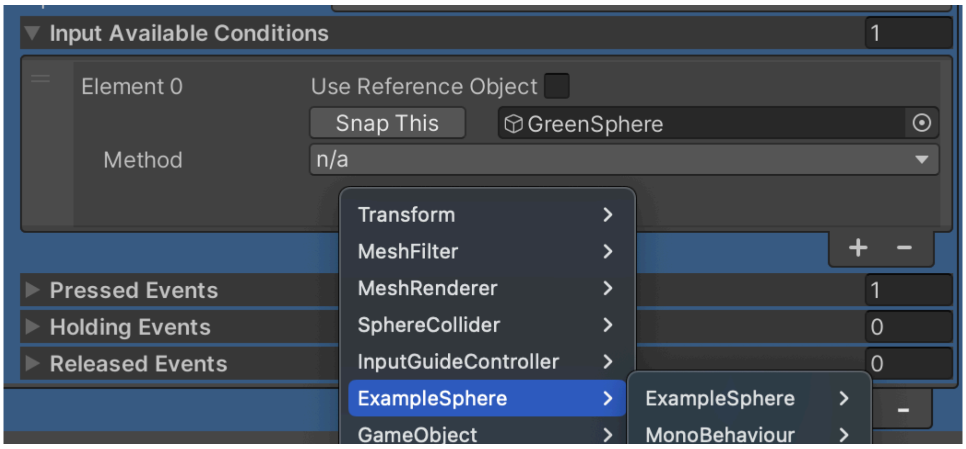
2.3.1.1.Defining Conditions And Events



- If '***Use Reference Object***' is checked, the defined reference object will be utilized.



- Select a boolean-returning function from the available functions associated with the reference object. If the reference object is null or lacks the specified component, the condition is bypassed.
- A GameObject can either be manually dragged and dropped, or selected directly by clicking the **'Snap This'** button, which automatically selects the GameObject to which this component is bound .A selection can then be made from the Boolean-returning functions within the listed components of the selected GameObject.



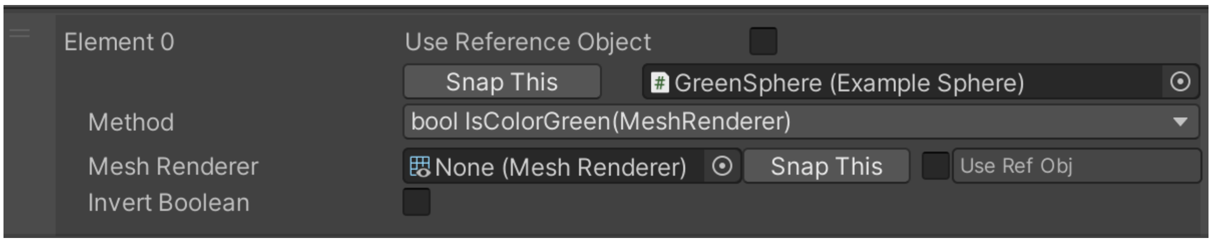
2.3.1.1.1.Example Scenario

Continuing from the earlier example:

- A player holds a red cube while looking at a green sphere. The player wants to paint the sphere red when pressing a specific key.
- To achieve this, the system must check if the sphere's color is green before allowing the input to trigger. If the sphere is green, the guide will be displayed, and the events will be executed.

Here's a sample function to implement the condition:

```
public bool IsColorGreen(MeshRenderer meshRenderer)
{
   // Checks if the material color of the MeshRenderer is green
   return meshRenderer.material.color.Equals(Color.green);}
```



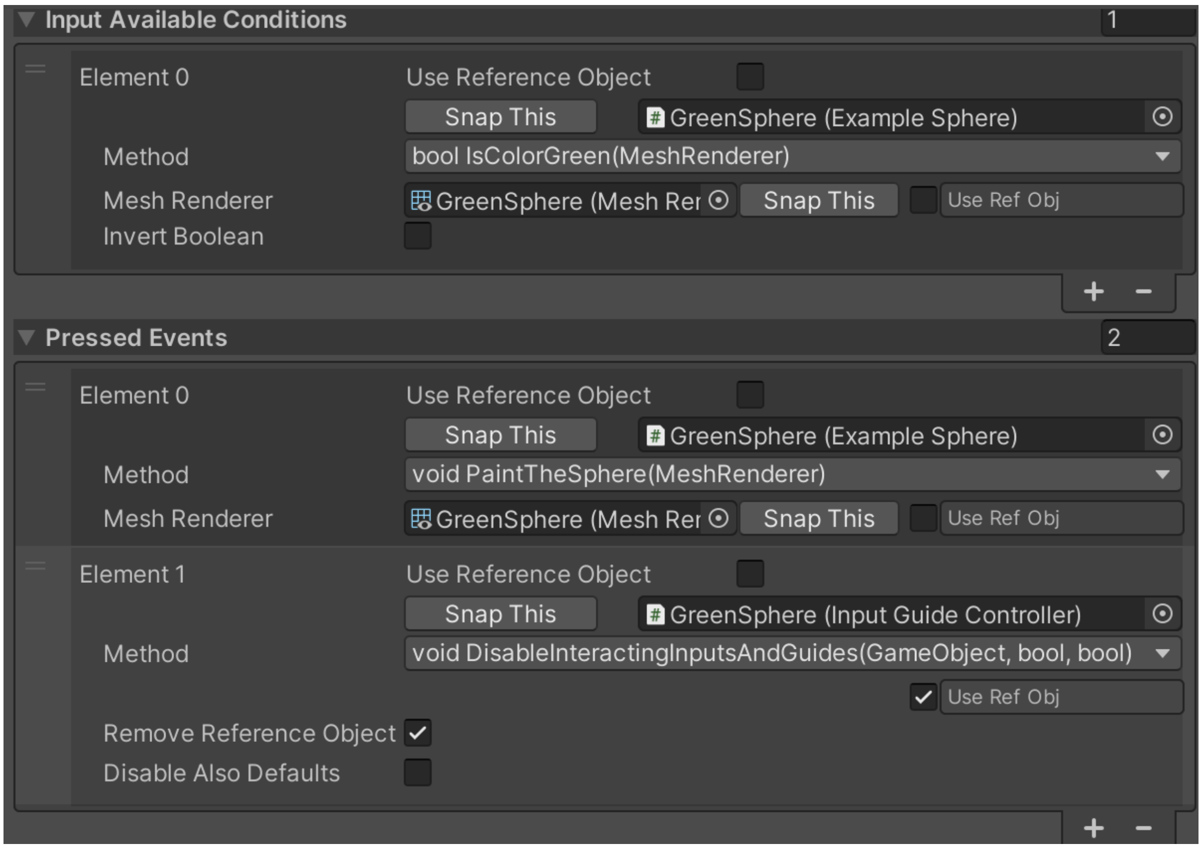
Any GameObject can be assigned manually by dragging it into the designated field or automatically by clicking the '***Snap This'*** button. The button binds the current GameObject to the component for quick selection.

- **Parameter Selection**:
  Parameters can be assigned in several ways:
  - By clicking the '***Snap This'*** button, which automatically selects a component from the bound object if it has a required type of the parameter.
  - By enabling the '***Use Ref Obj'*** option, which utilizes the previously defined reference object.
  - By manually dragging and dropping objects of the appropriate parameter type into the designated field.
- **Invert Boolean**:
  If the opposite of the returned boolean value is required, enable the '***Invert Boolean'*** toggle.

Once the conditions are met and the key is pressed, a function can be written and defined under the **'Pressed Events'** section to execute the desired action, such as painting the sphere red. For example:

```
public void PaintSphere(MeshRenderer meshRenderer)
{
   // Sets the material color of the MeshRenderer to red
   meshRenderer.material.color = Color.red;}
```

After the sphere is painted red, the guide should no longer be displayed, and the associated events should stop being listened. This ensures that the input guide dynamically reflects the current state and avoids redundant actions.



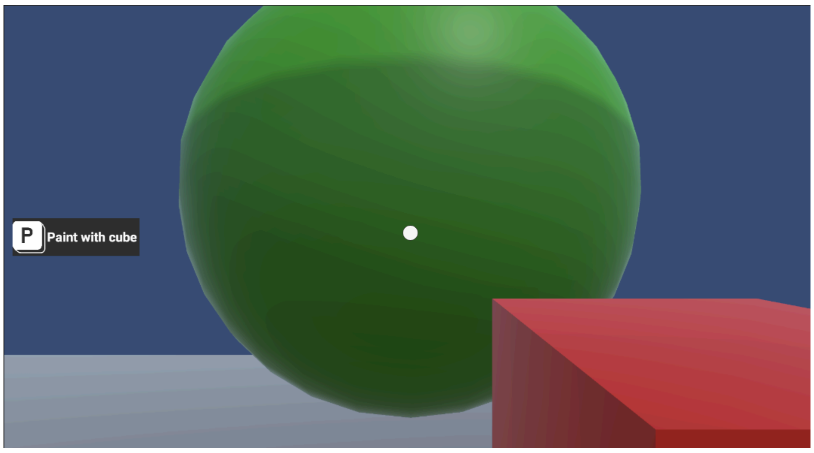
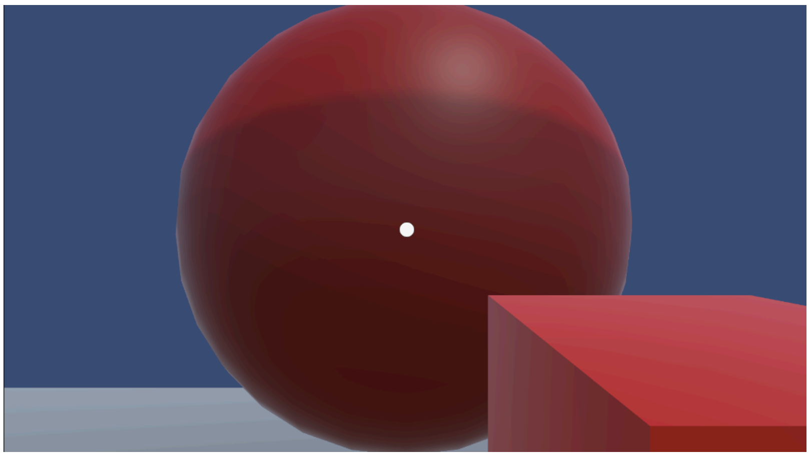
Input Available Conditions 1

Element 0 | Use Reference Object ☐
Snap This | GreenSphere (Example Sphere)
Method | bool IsColorGreen(MeshRenderer)
Mesh Renderer | GreenSphere (Mesh Rer | Snap This | ☐ Use Ref Obj
Invert Boolean ☐

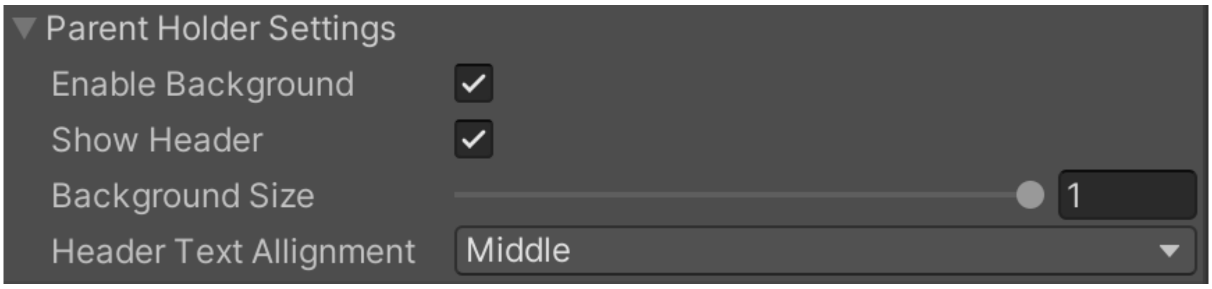
Pressed Events 2

Element 0 | Use Reference Object ☐
Snap This | GreenSphere (Example Sphere)
Method | void PaintTheSphere(MeshRenderer)
Mesh Renderer | GreenSphere (Mesh Rer | Snap This | ☐ Use Ref Obj

Element 1 | Use Reference Object ☐
Snap This | GreenSphere (Input Guide Controller)
Method | void DisableInteractingInputsAndGuides(GameObject, bool, bool)
✓ Use Ref Obj
Remove Reference Object ✓
Disable Also Defaults ☐

Before pressed 'P' key	
After pressed 'P' key	

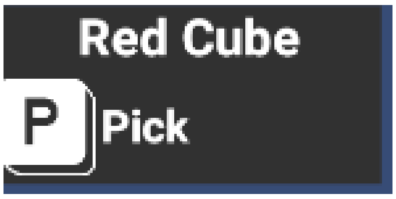
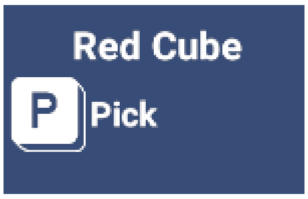
2.3.2.Parent Holder Settings



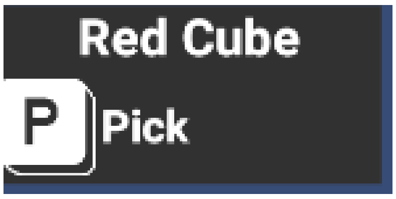
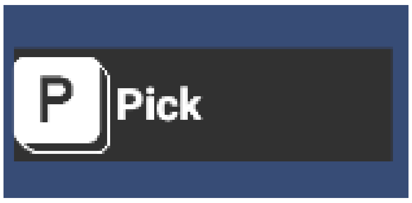
Parent Holder Settings

Setting	Value
Enable Background	✓
Show Header	✓
Background Size	1
Header Text Allignment	Middle

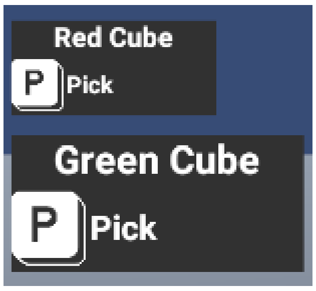
Enable Background: Closes the background of the holder relative to its parent.

If ‘**Enable Background**’ is enabled.	If ‘**Enable Background**’ is disabled.
	

Show Header: If the '**Show The Name**' feature is enabled but the header is not intended to be displayed within its parent holder, it can be disabled here.

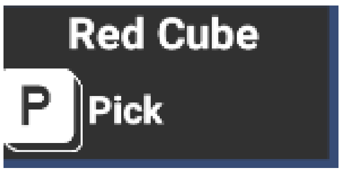
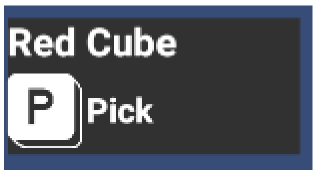
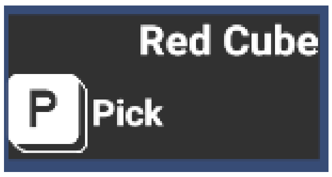
If both ‘**Show The Name**’ and ‘**Show Header**’ are enabled.	If ‘**Show The Name**’ is enabled and ‘**Show Header**’ is disabled.
	

Background Size: Sets the size of its parent holder.



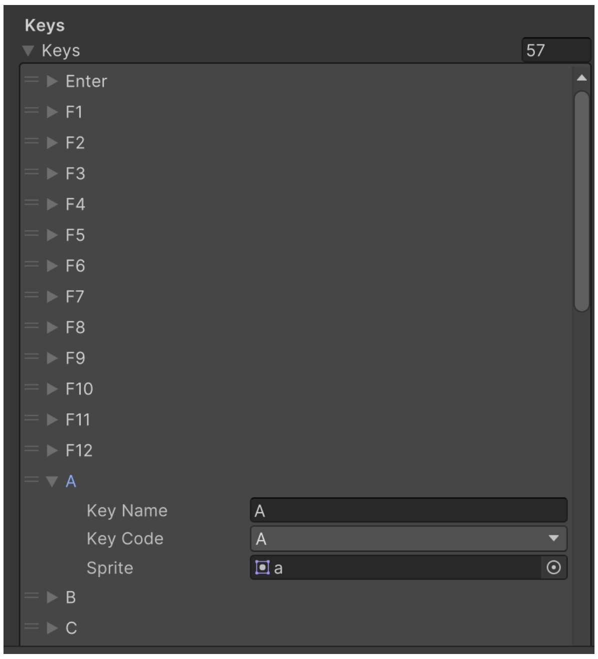
The holder of ‘Red Cube’ size is 0.7.
The holder of ‘Green Cube’ size is 1.

Header Text Allignment: Specifies the alignment of the header within its parent holder.

Chosen **'Middle'** for '**Header Text Allignment'.**	Chosen '**Left'** for '**Header Text Allignment'.**	Chosen '**Right'** for '**Header Text Allignment'.**
		

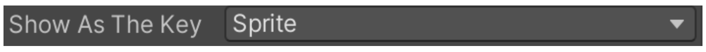
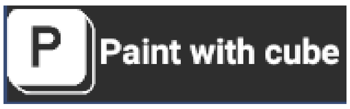
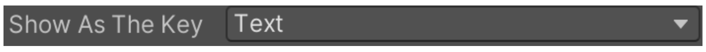
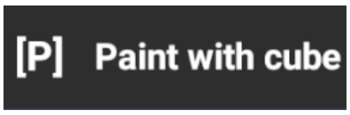
***Tip Note:** The background image can be customized via the **'Image'** component of the prefab named **'InputGuideEdgeHolder'**, located in the directory: '**`/Assets/BudInGui/InputGuideSystem/Prefabs/UIElements/GuideUIs/HolderReferences/`**'.

3.Input & Guide Data


Keys
Keys 57
Enter
F1
F2
F3
F4
F5
F6
F7
F8
F9
F10
F11
F12
A
Key Name A
Key Code A
Sprite a
B
C

A keys data file, named ‘***Input & Guide Data’***, is located in the **/Assets/BudInGui/InputGuideSystem/Data/** path.

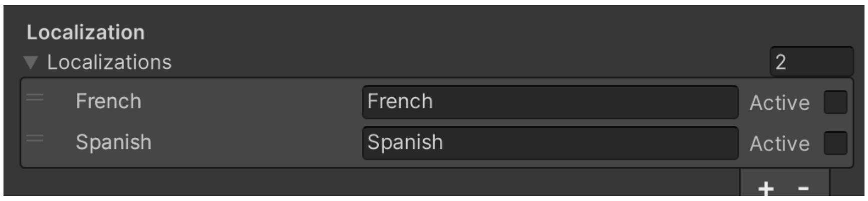
- This file contains Unity keys along with their associated names and sprites.
- If ‘***Sprite’*** is selected in the ‘***Show As The Key’*** section, the sprite will be displayed. If ‘***Text’*** is selected, the key's name will be displayed.

Show Option	**View Of Result**
 Show As The Key: Sprite	
 Show As The Key: Text	 [P] Paint with cube

- You can perform the following actions:
    - Add new Unity key data.
    - Edit or change the sprites and names of existing data.
    - Remove existing data.

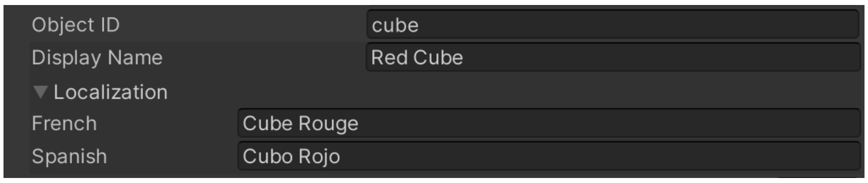
4.Localizations

Localization settings can also be managed within the ‘***Input & Guide Data’*** file in the **/Assets/BudInGui/InputGuideSystem/Data/** path.



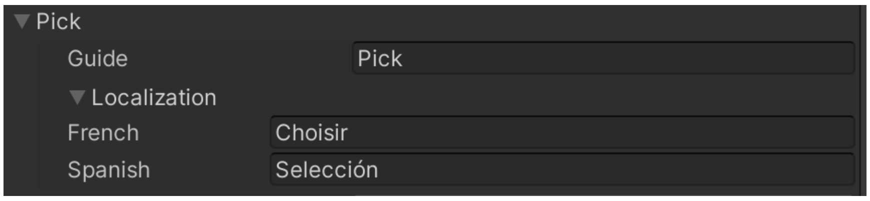
Localization		
▼ Localizations		2
French	French	Active ☐
Spanish	Spanish	Active ☐

- New locales can be added and will be listed as subheadings in two places:
  - The ‘***Display Name’*** field of the object within the ‘**InputGuideController’** component.



Object ID	cube
Display Name	Red Cube
▼ Localization	
French	Cube Rouge
Spanish	Cubo Rojo

  - The ‘***Guide’*** section of inputs and guides.



▼ Pick	
Guide	Pick
▼ Localization	
French	Choisir
Spanish	Selección

- You can switch the current locale to the desired one by updating the relevant settings.

4.1.Scripting API

-ChangeLocalization

Declaration

public void ChangeLocalization(string localName=null);

Parameters

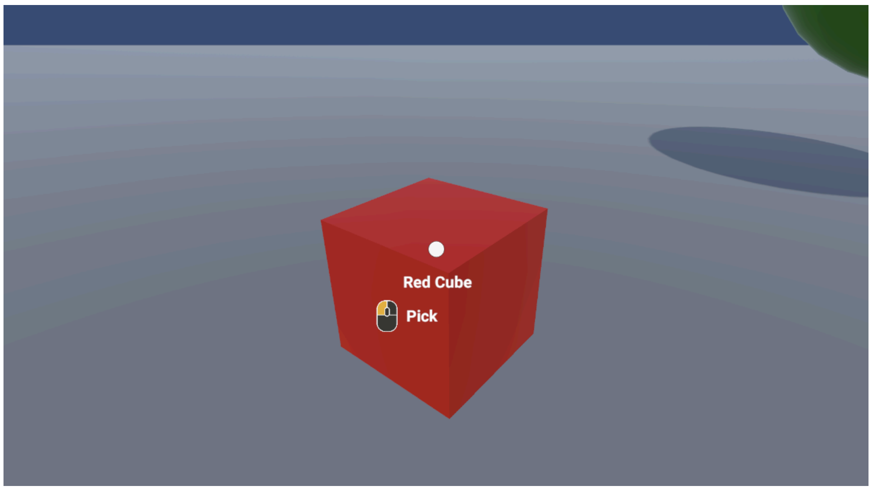
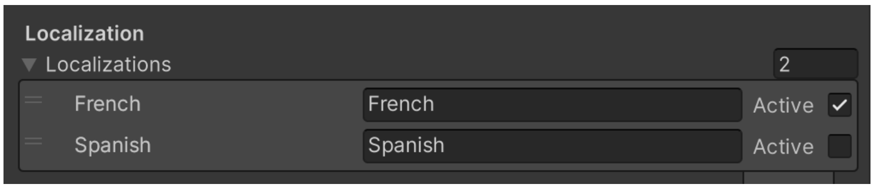
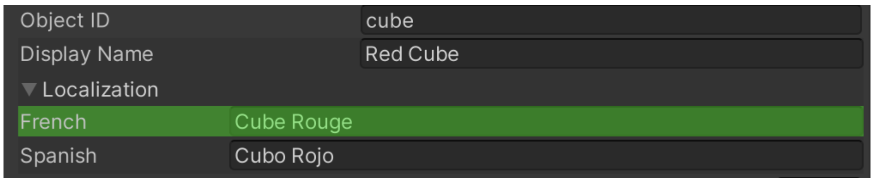
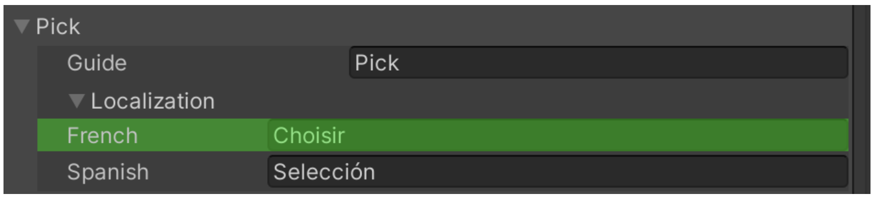
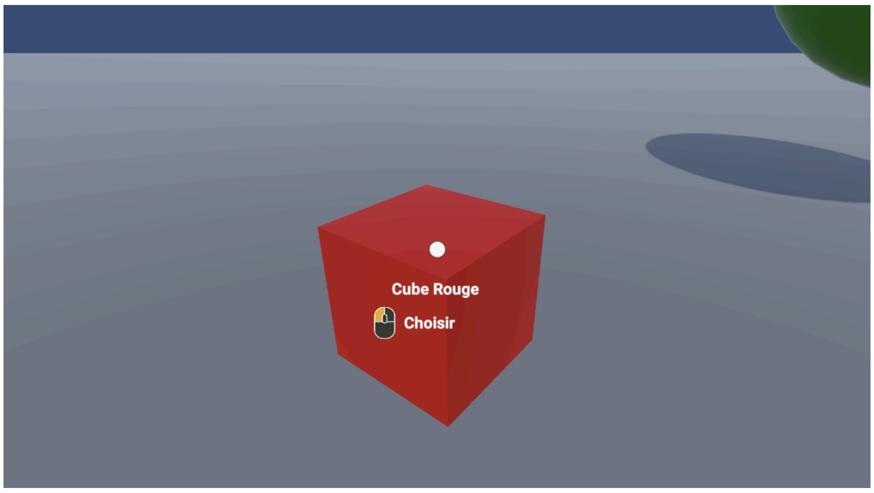
- localName **–** The name of the local. If this is null or undefined within the locals, it will revert to the default local.

Description

Convert the current local to the desired local.

Usage

```
UIDisplay.Instance.inputGuideData.ChangeLocalization();
InputGuideData.Instance.ChangeLocalization();
UIDisplay.Instance.inputGuideData.ChangeLocalization(string);
InputGuideData.Instance.ChangeLocalization(string);
```

No selection	
Selection any ● When the '***Active***' property of any locale is enabled in the data, the subtexts associated with that locale are automatically activated across all '**InputGuideController**' components. ● This ensures that localized content is displayed consistently for each component.	 Localization ▼ Localizations 2 French French Active ✓ Spanish Spanish Active  Object ID cube Display Name Red Cube ▼ Localization French Cube Rouge Spanish Cubo Rojo  ▼ Pick Guide Pick ▼ Localization French Choisir Spanish Selección  Cube Rouge Choisir