Control States

Introduction

Controls are components that a user interacts with through direct manipulation to read or edit information about an application. In CGI Studio Controls are templates created for later reuse via control nodes. They are meant to be interactive and some of them even react to user input like touch or focus. But as Controls should also be able to change their state corresponding to a user input the concept of Control States was introduced.

Possible states of a Control are:

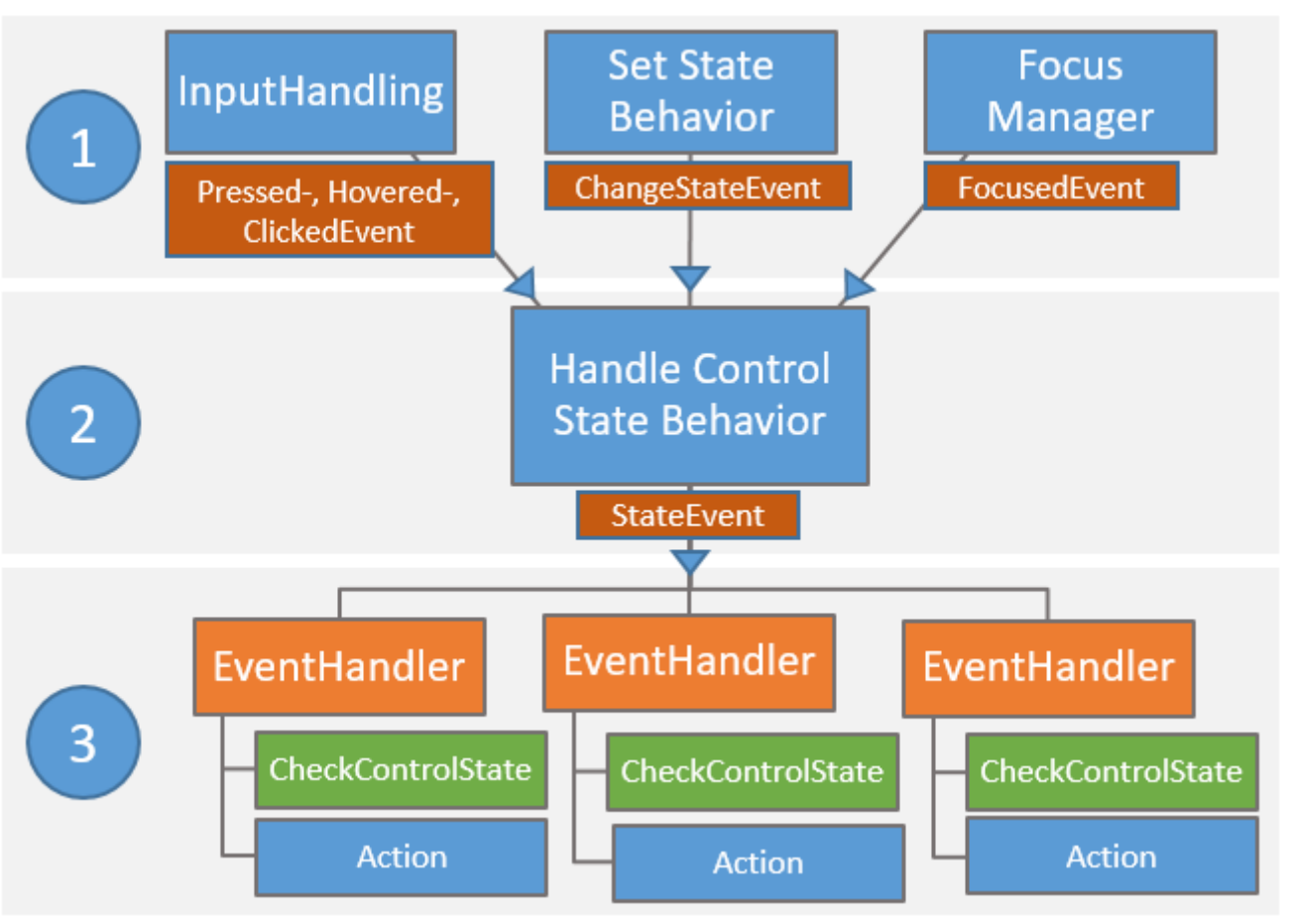
- Selected: Used for Controls which can be toggled between selected and deselected. E.g. CheckBox, RadioButton.
- Pressed: Nearly every touchable Control has a Pressed state to show a visual reaction to a touch.
- Enabled: Enabled state can for example be used to set the Control disabled. E.g. receives no input, grayed out, etc.
- Highlighted: This state can be used for cases when dragging a control, etc.
- KeyboardFocused: Keyboard focus refers to the UI element that is currently receiving input. Only one UI element in an HMI can have keyboard focus at the same time. This is handled by the KeyboardFocusManager.
- ScopeFocused: The scope focus is kept per focus scope. An HMI can have more than one focus scope. When a focus scope becomes active, then the UI element having the scope focus will receive keyboard focus. Yet there is no manager for the scope focus.
- Hover: Used to mark the Control which is currently hovered.

This concept offers an easy way to manage the visual representation of different states of Controls.

As the image shows, the Button reacts to the users input in an intuitive way. But therefore it is necessary that the Button has different states like Hovered, Pressed, or Focused (also see chapter Control Examples). The second example shows the CheckBox getting selected by a user input which basically demonstrates the workflow of Control States.

Description

As already shown in the introduction the state handling of Controls basically consists of three layers. For further understanding these three layers can be split up more detailed as shown in the graph below:



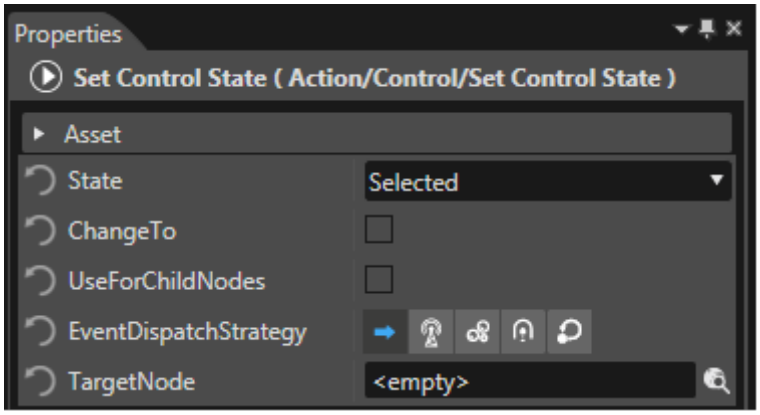
1. Event Emitter

The **first layer** represents the emitter of events which get received by the Handle Control State Behavior. They all have the purpose to change the state of a Control. To give a short overview following three categories will be explained.

- Input Handling : e.g. Touch Input Handling . Therefore the ControlTouchSession receives a TouchEvent and passes the information to the Control. Therefore it uses more specific events in the sense that a combination of touch events result in a new input event (e.g.

PressedEvent, HoveredEvent or ClickEvent).

- **Behaviors**: Also with Behaviors the state of a Control can be changed. e.g. the Set Control State Behavior is an Action Behavior which sends a ChangeStateEvent to the Handle Control State Behavior.

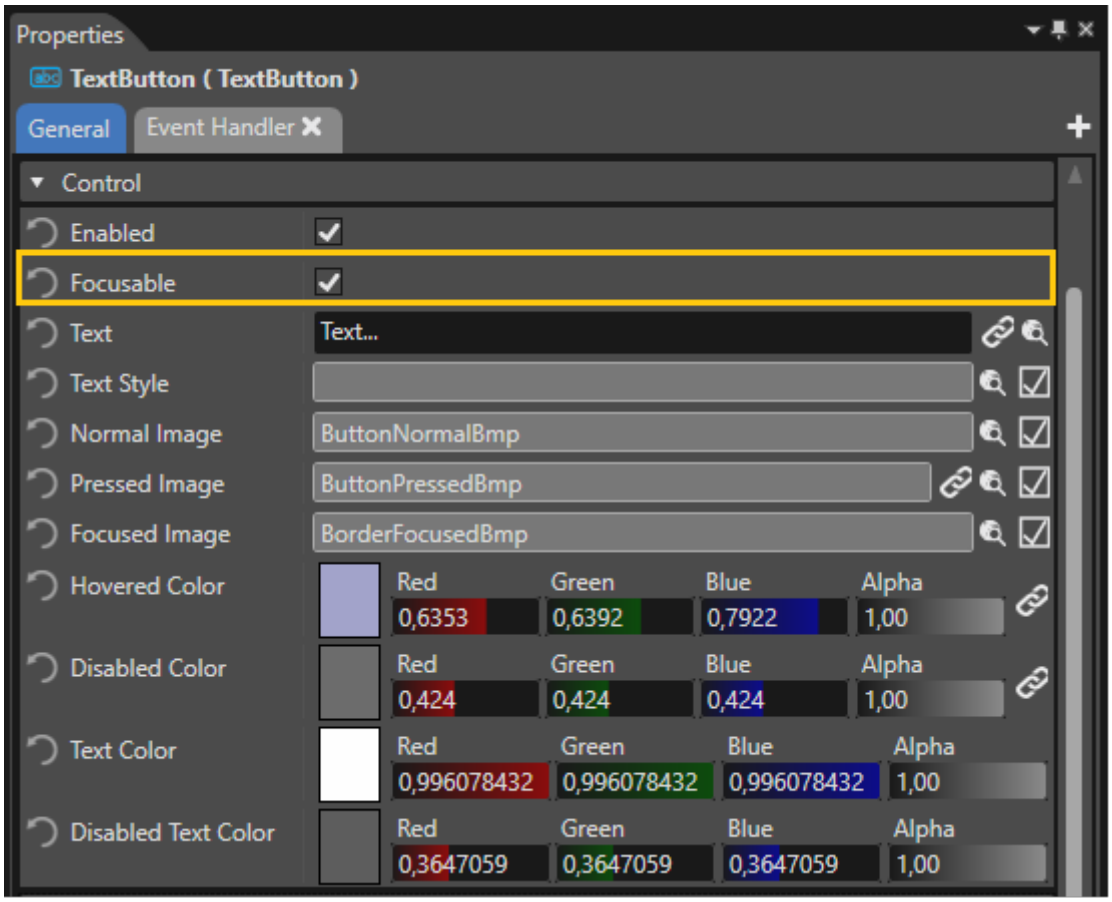


The first two properties offer to select the state which has to be changed and the status (true/false). The other properties define to which node the ChangeStateEvent should get sent to.

- **Focus Manager**: The focus refers to the UI element that is currently receiving input. Only one UI element in an HMI can have focus at the same time. This is handled by the KeyboardFocusManager.

TextButtons which are all focusable. Clicking on the second button to set the focus to it would cause the first button to lose the focus.

It sends a FocusEvent to the Control which got the new focus and to the Control which lost the focus. As for the other Events also the FocusEvent is basically meant to be received by the Handle Control State Behavior. If the Handle Control State Behavior receives a FocusEvent with the LostKeyboardFocus information it sets the focused state to false and vice versa.
Though the Control (Handle Control State Behavior) has to be set to Focusable as shown in the image below.



2. Handle Control State Behavior

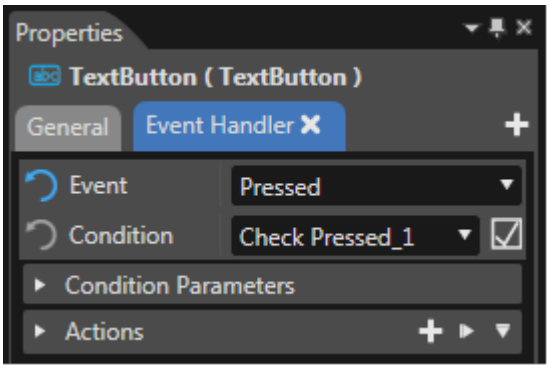
The **second layer** represents the Handle Control State Behavior, the basic Behavior which handles the states of a control.

Therefore it receives Events from the ControlTouchSession (HoverEvent, PressedEvent, ClickEvent), the KeyboardFocusManager (FocusEvent) or from the Set Control State Behavior (ChangeStateEvent).

Corresponding to this Events the Handle Control State Behavior sets the state and furthermore sends a StateEvent (mainly to EventHandlers with StateConditions as explained in the next paragraph). This event always consists of two variables. The first one contains the states as they were before the change and the second one contains the states after the change. This makes it possible to define more specific conditions as explained in the next paragraph.

3. Event Handler

The **third layer** represents the EventHandler which receives the State Event from the Handle Control State Behavior and changes the visual representation of the Control. For checking the State Event the State Condition should be used.



The two properties define which state should be checked and how the state changed:

- ChangedToTrue: state changed from false to true
- ChangedToFalse: state changed from true to false
- IsTrue: state is true (previous state doesn't matter)
- IsFalse: state is false (previous state doesn't matter)

If the condition is fulfilled the Event Handler triggers an Action Behavior which leads to the change of the visual representation. (Predefined Controls often use Set Image Behavior and Set Color Behavior.)

> to learn more about the concept of Event Handlers also see chapter EventHandler Tab .

Revision #3
Created 2023-03-01 03:02:32 UTC by Kanai Tomoaki
Updated 2024-02-21 06:21:56 UTC by Tsuyoshi.Kato