Overview

This chapter gives an overview of Behaviors.

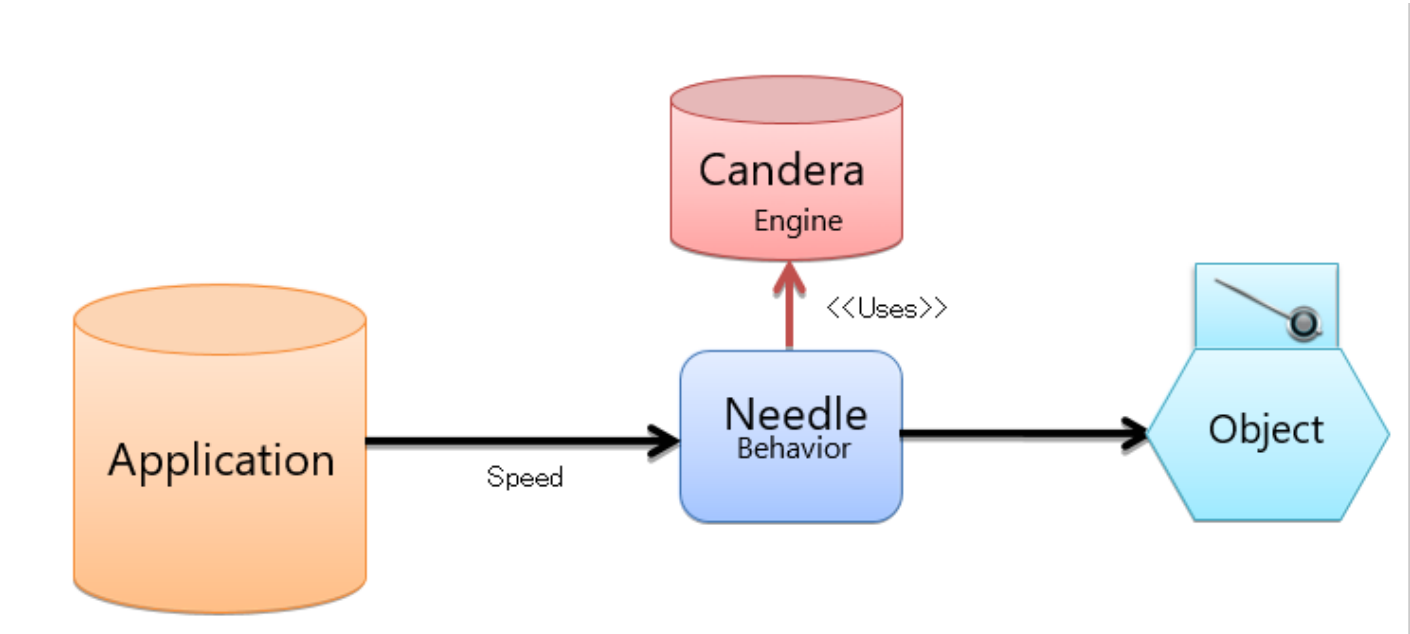
What are Behaviors?

The purpose of Behaviors is to perform modifications to the content presented to the user.

While it would be possible for the application to directly perform any required change directly to the nodes of the scene tree this would require quite complex application code. The application would need to be specifically tailored to a single solution and every change to the scene tree might require a change in the application code. To avoid this strong dependency, Behaviors are used to perform the actual modification on the content.

> The following impact has to be considered: Cross-references (nodes or behaviors) to different scenes may occur. For details see Cross-Reference Implications below.

Their advantage is that they after they are created they can be easily used in Scene-Composer by people without any coding knowledge. By adding configurable properties they can be adapted to apply in multiple different scenarios without the need to create new ones. Further the application code can be simplified as it only needs to provide an "input" for the Behaviors. It is no longer responsible to decide how the content needs to be modified based on the input.



Since the introduction of Behaviors in CGI-Studio 3.4.2 they are a replacement for Widgets, which are now being considered deprecated. Therefore instead of Widgets, Behaviors should be used to add functionality.

Behaviors can be considered Widgets 2.0. They can perform any task that widgets can perform but have some added features which greatly enhances their usability. While it is in most cases easy to convert an existing Widget into a Behavior a simple conversion might not make usage of the new functionality introduced negating the actual benefit from using them.

Comparison to Widgets

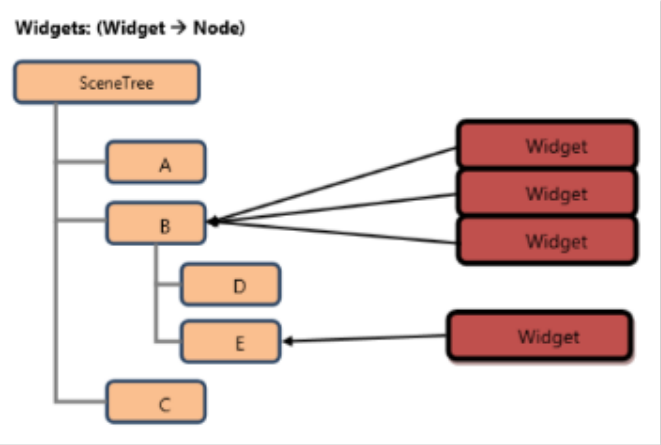
As mentioned behaviors are essentially an enhanced version of widgets. Compared to widgets they have 2 major new features to help with implementing complex functionality.

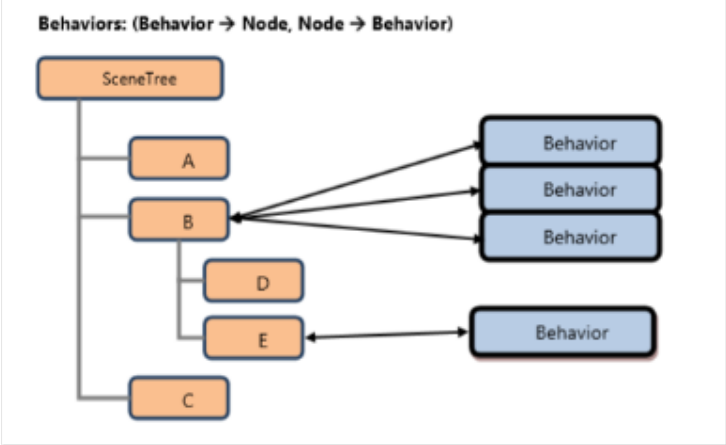
OWNERSHIP

While (most) widgets are attached to a node they are considered part of the scene itself. A widget does know about the node it is attached to but the node does not know about the widget.

This can cause problems if multiple widgets should collaborate as by default they can not know which other widgets they should interact with. To overcome this problem it was required to either specify "other" widgets a property (type is Candera::WidgetBase*) directly in the widget or store custom information on the node about other widgets that are also attached.

Behaviors have simplified this by now longer considering them as part of the Scene but rather part of the node. Each node has a Behavior-Container which hold all attached behaviors for this node. Via this container it is now easily possible to communicate and collaborate between different behaviors.





EVENTS

The second major addition is the usage of events. Events allow to send "information" from one behavior to another behavior without them knowing any details about each other.

Events will be routed thought the scene tree. This is made possible by the change in ownership mentioned before. Once an event is dispatched to a node, all attached behaviors will receive the event unless the event routing is stopped.

For more information about Events see chapter Behaviors and Events .

Cross-Reference Implications

Behaviors can have properties of type Node or Behavior. If those properties refer to parts of other scenes, those scenes will be automatically loaded by the AssetProvider.

If unloading is required, it is necessary to unload the entire cross-referencing network. Partially unloading cross-referenced scenes of such a cross-referenced network will result in dangling pointers and a will **end up in unpredictable and incorrect behavior!**

Unloading the complete cross-reference network is safe.

The same also has to be considered for Controls and State Machines.

Since the Global State Machine cannot be unloaded, it is not allowed to unload any referenced Nodes or Behaviors.

Revision #5
Created 2023-02-22 08:55:13 UTC by Kanai Tomoaki
Updated 2024-04-26 13:05:15 UTC by Elisabeth Zauner