Searching for Nodes with SearchTreeTraverser

Searching for Nodes

With Candera::SearchTreeTraverser it is possible to find nodes within a given scene tree by certain criteria. The search scene graph traverser is a Candera::TreeTraverserBase implementation that evaluates each node with a given Candera::SearchCriterion.

Following code snippets illustrate how to use the traverser.

Creating a plugin that registers a configuration and uses the default configuration editor.

First create a Candera::SearchTreeTraverser and then set its Candera::SearchCriterion. The search criterion validates nodes based on the implemented criterion.

In the example below the criterion is a Candera::ScopeMaskSearchCriterion.

```
    // Create criterion
    ScopeMaskSearchCriterion<Node> scopeMaskCriterion;
    scopeMaskCriterion.SetScopeMask(scopeA);

    // Create SearchTreeTraverser
    SearchTreeTraverser<Node> scopeSearchTraverser;
    scopeSearchTraverser.SetSearchCriterion(&scopeMaskCriterion);
```

Tree Traversing

To start a search within a node tree, call Candera::SearchTreeTraverser::Traverse(). With the Candera::SearchTreeTraverser::SetOnFoundAction you can define if the search should stop after the first node that fulfills the search criterion is found, or if it should continue to gather all valid nodes.

Possible TraverserActions are:

- *ProceedTraversing*: Find all nodes matching the given search criterion (traversing is stopped at the end of the tree)
- *StopTraversingForDescendants*: Find only nodes on the top level of the tree (traversing stops at child nodes and proceeds at siblings, until the end of the tree)
- *StopTraversing*: Find only the first node matching the search criterion (traversing is stopped immediately).

The nodes fulfilling the search criterion are retained and are available to be later retrieved. Get the nodes one by one with Candera::SearchTreeTraverser::GetSearchResult. The order of the nodes is not defined and each node can be retrieved only once. When no more nodes are available, the method returns 0.

Following are examples for the different TraverserActions.

ProceedTraversing

This action is used if the traversing should continue with the next child (if the current node has any) or with the next sibling.

First define the action and then call the Traverse() method. All nodes that fulfill the search criterion are found and stored internally in a list.

```
    // Set on found action and traverse
    scopeSearchTraverser.SetOnFoundAction(Candera::TreeTraverserBase<Node>::ProceedTraversing);
    scopeSearchTraverser.Traverse(*groupRoot);

    // Get results
    UInt32 nodeCount = 0;
    for (Node* node = scopeSearchTraverser.GetSearchResult(); node != 0; node = scopeSearchTrave
        nodeCount++;
    }
```

StopTraversingForDescendants

Used if the traversing should stop for all child nodes and continue with the next sibling.

```
    // Set on found action and traverse
    scopeSearchTraverser.SetOnFoundAction(
Candera::TreeTraverserBase<Node>::StopTraversingForDescendants);
    scopeSearchTraverser.Traverse(*groupRoot);
```

StopTraversing

If you search for only one node which fulfills your criteria, you can use StopTraversing. The traversing will stop immediately after the first node is found.

```
// Set on found action and traverse
scopeSearchTraverser.SetOnFoundAction(Candera::TreeTraverserBase<Node>::StopTraversing);
scopeSearchTraverser.Traverse(*groupRoot);
Node* resultNode = scopeSearchTraverser.GetSearchResult();
```

> Another call of Candera::SearchTreeTraverser::GetSearchResult would return 0, because we use StopTraversing here. This leads to one result only.

Revision #5
Created 2023-02-21 06:22:22 UTC by Tsuyoshi.Kato
Updated 2026-08-24 07:30:55 UTC by Kanai Tomoaki