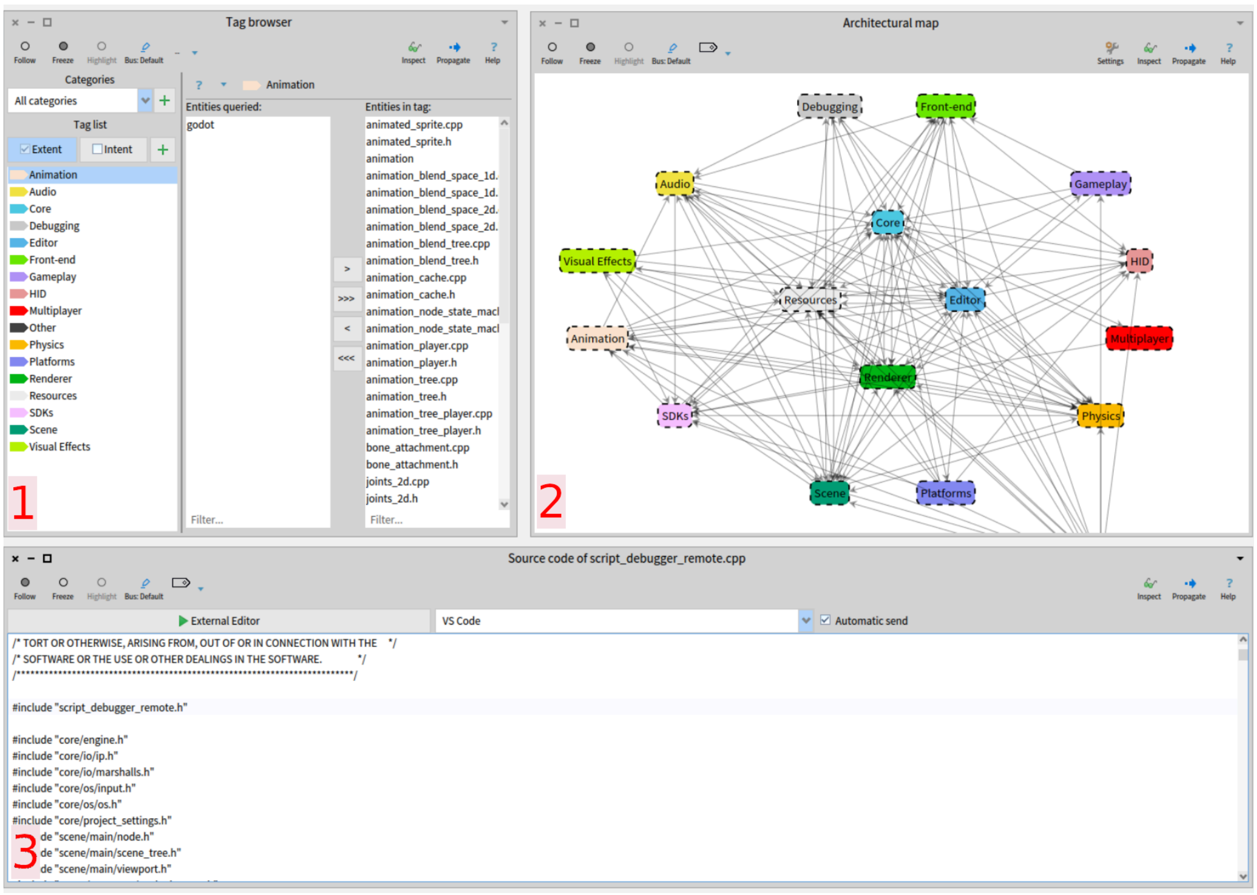
Moose - How to Use

This document is a step-by-step guide on how to use each of Moose's functionalities you will need for this experiment. It should not take longer than 5 minutes to read. We recommend you read this document with Moose open so you can read about the functionalities and try it out at the same time.

Overview



```
/* TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE    */
/* SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.               */
/*************************************************************************/

#include "script_debugger_remote.h"

#include "core/engine.h"
#include "core/io/ip.h"
#include "core/io/marshalls.h"
#include "core/os/input.h"
#include "core/os/os.h"
#include "core/project_settings.h"
   de "scene/main/node.h"
   de "scene/main/scene_tree.h"
   de "scene/main/viewport.h"
```

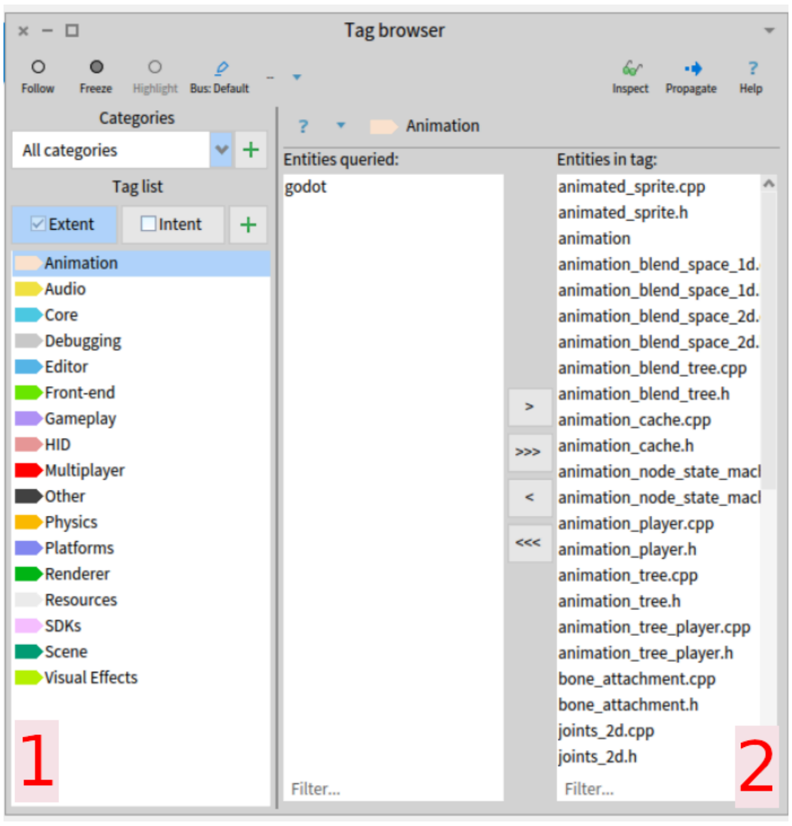
For this experiment, you will use three Moose functionalities, which are highlighted in the image above and described in detail in this document:

1. **Tag Browser**: shows all files/folders categorized by subsystem as a list.
2. **Architectural map**: shows all files/folders categorized by subsystem as an interactive diagram.
3. **Source code**: shows the source code for a file you propagate from the Architectural map. It links with Visual Studio Code and Emacs.

In case you have any issues with Moose, please read Section 5 of this document. If you would like to play around with the tool before starting, you can find some optional "warmup" exercises in Section 6.

Tag Browser

1. Click on the left-side list to select a subsystem.
2. After you click, you can see files/folders related to the subsystem on the right-side list, labelled as “Entities in tag”. You can scroll up/down the “Entities in tag” list to see all files/folders.



Entities queried:

godot

Tag list:

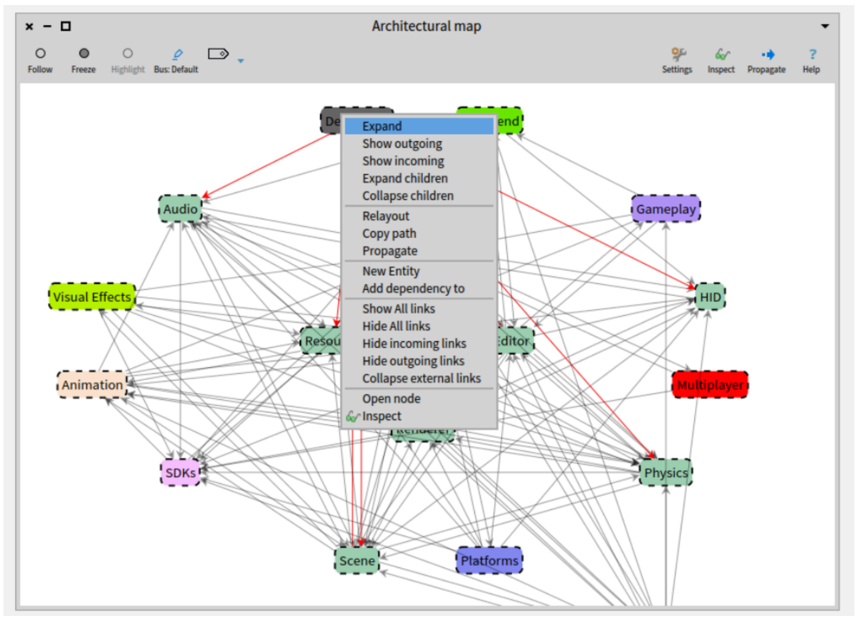
Animation
Audio
Core
Debugging
Editor
Front-end
Gameplay
HID
Multiplayer
Other
Physics
Platforms
Renderer
Resources
SDKs
Scene
Visual Effects

Entities in tag:

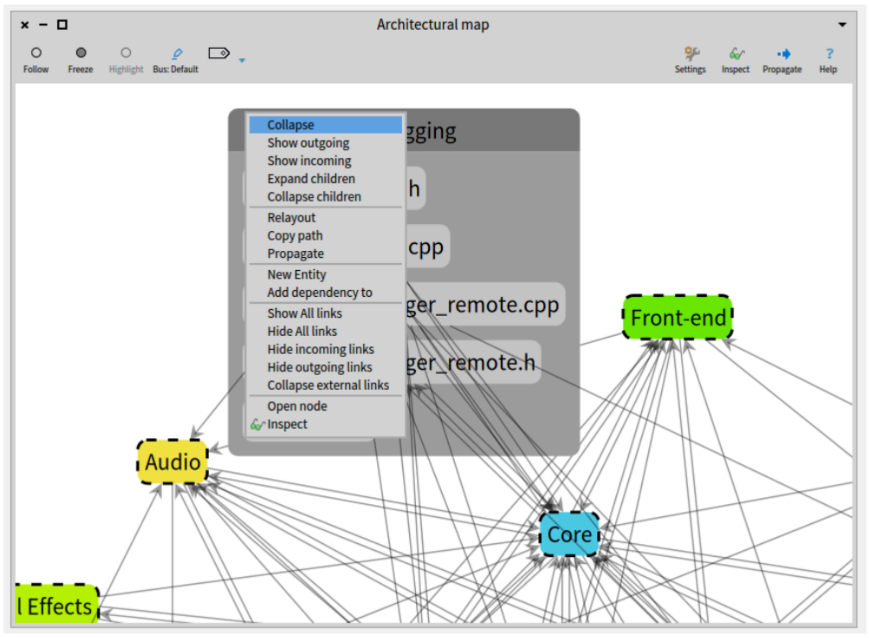
animated_sprite.cpp
animated_sprite.h
animation
animation_blend_space_1d.
animation_blend_space_1d.
animation_blend_space_2d.
animation_blend_space_2d.
animation_blend_tree.cpp
animation_blend_tree.h
animation_cache.cpp
animation_cache.h
animation_node_state_macl
animation_node_state_macl
animation_player.cpp
animation_player.h
animation_tree.cpp
animation_tree.h
animation_tree_player.cpp
animation_tree_player.h
bone_attachment.cpp
bone_attachment.h
joints_2d.cpp
joints_2d.h

Architectural Map

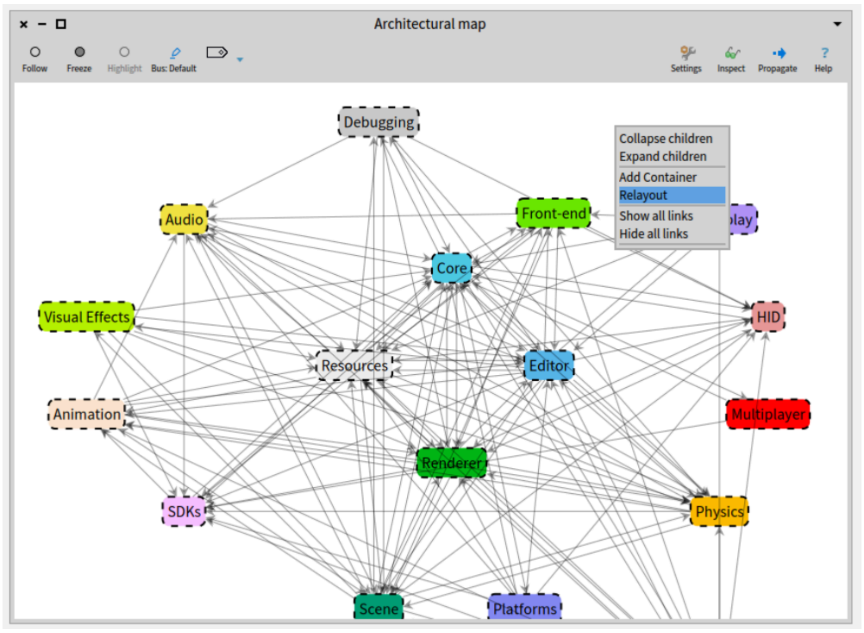
The architectural map is a graph with several nodes, represented as colored rectangles. The node is linked by arrows, which represent dependencies, also called “includes” in the context of C++. For example, X→Y means “X depends on Y”. You can right-click any node to open its context menu. You can then click “Expand” to see the files and folders inside of a node. If a node contains folders, you can also right-click a folder, then click “Expand” to see its contents.



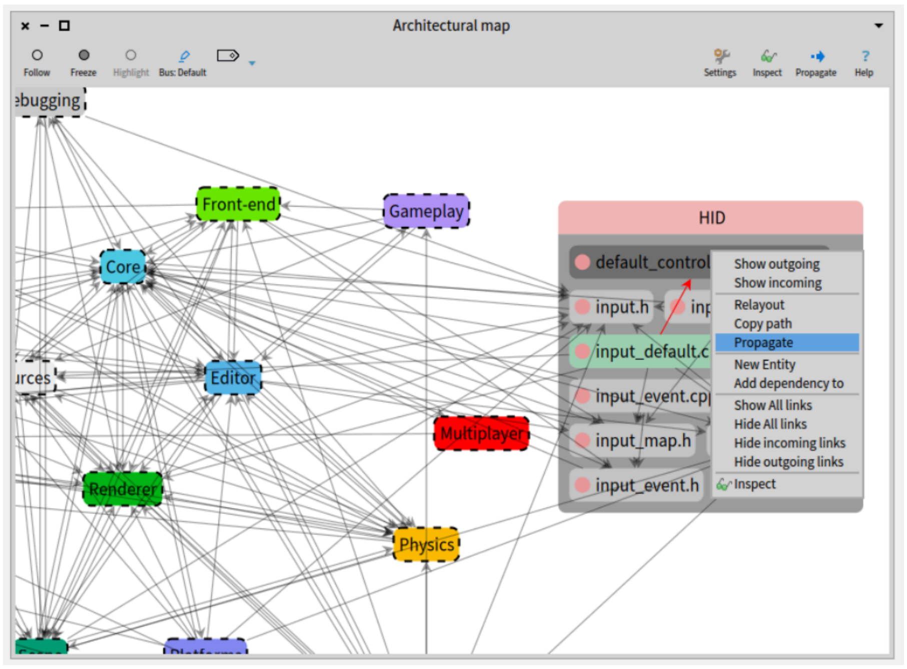
If you no longer want to see the contents of a node, you can right-click the node and then click “Collapse” to hide them. You can expand and collapse nodes at any time. The context menu also contains other options which might be useful, such as Collapse Children, Show/Hide All Links, Show/Hide Incoming/Outgoing Links and Copy Path. You are welcome to use them as much as you want.



The architectural map also allows you to drag and drop nodes as you wish. However, this may cause nodes to sometimes overlap and become hard to read. In such cases, you can automatically adjust the layout of a single node or all nodes by right-clicking the node or the white background and then clicking on “Relayout”.

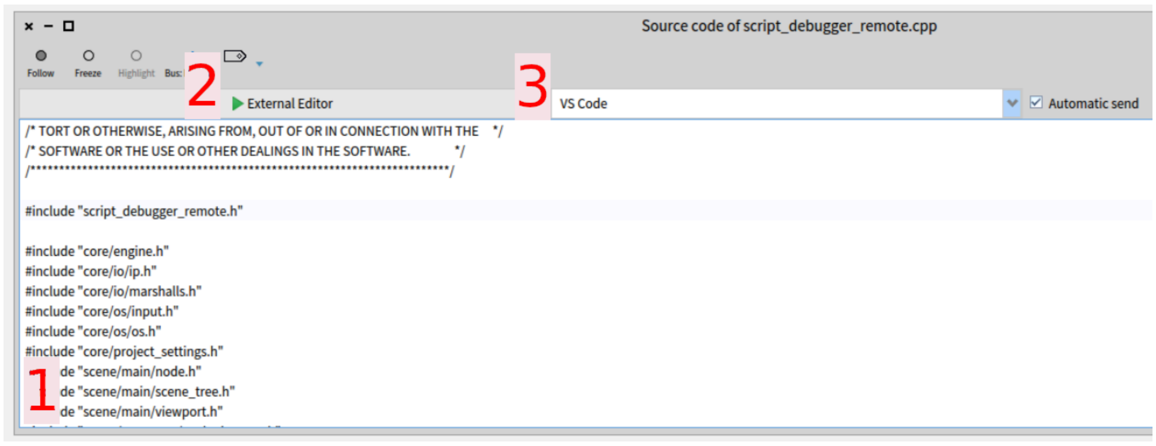


If you want to inspect the source code of a file inside of a node, you can right-click it, and click “Propagate”. The source code of the file will open in the source code browser. **Important:** the “Propagate” functionality shows source code for files only. If you propagate an entire folder or subsystem, no source code will be shown in the code browser.



Source Code Browser

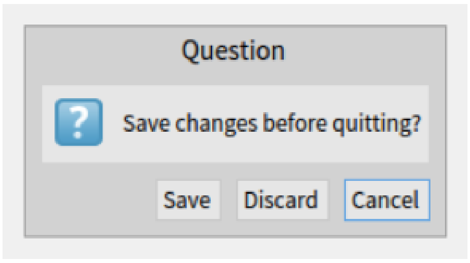
1. The source code browser will show you the source code for the currently propagated file, one at a time. You can scroll up/down to read the entire file.
2. You can click the “External Editor” button on the top-left side of the window to open the code on Visual Studio Code.
3. You can click the combo box and switch between Visual Studio Code and Emacs.



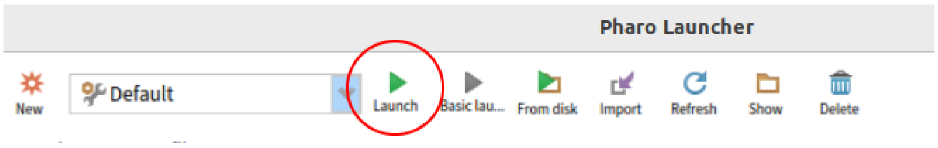
```
/* TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE    */
/* SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.               */
/*************************************************************************/

#include "script_debugger_remote.h"

#include "core/engine.h"
#include "core/io/ip.h"
#include "core/io/marshalls.h"
#include "core/os/input.h"
#include "core/os/os.h"
#include "core/project_settings.h"
    de "scene/main/node.h"
    de "scene/main/scene_tree.h"
    de "scene/main/viewport.h"
```

Issues

1. In case you run into issues, you can always close the Moose image by clicking the “x” button on the top-right side.
2. You will be asked if you want to save changes. Click “Discard”.



3. Open the image again by selecting it on the list and clicking “Launch”.



4. You should be back at the initial state as shown on the first page of this document. You can resume your tasks normally.

Warmup exercises (optional)

1. Go to the Tag Browser and view the Debugging subsystem. What is the first file/folder on the list?
2. Go to the Architectural Map and expand the Debugging system. What is the first file/folder you see, from top to bottom?
3. Considering the file you found in exercise 2, how many other files depend on it?

You can check the answers in the footnote at the bottom of the page.[1]

[1] **Answers:** 1) the folder “debugger”, 2) The file “performance.h”, 3) Yes, the file “performance.h” is included by three other files, as indicated by the three incoming arrows pointing towards the file in the Architectural Map.