



USER MANUAL

for Coati v0.3

TABLE OF CONTENTS

TABLE OF CONTENTS

1. QUICK START GUIDE
 - 1.1. Short description
 - 1.2. Starting up Coati
 - 1.3. Create New Project
 - 1.4. Source Analysis
 - 1.5. UI Intro
 - 1.5.1. Search Field
 - 1.5.2. Graph Visualization
 - 1.5.3. Code View
 - 1.6. Start exploring!
2. INTRODUCTION
 - 2.1. About this document
 - 2.2. Supported Languages
 - 2.2.1. C
 - 2.2.2. C++
 - 2.3. Finding System Header Locations
 - 2.3.1. on Windows
 - 2.3.2. on Mac
 - 2.3.3. on Linux
3. INSTALLATION
 - 3.1. Windows
 - 3.2. OS X
 - 3.3. Linux
 - 3.4. Data folder
4. USER INTERFACE
 - 4.1. Start Window
 - 4.2. Path List Box
 - 4.3. Preferences Window
 - 4.4. Project Setup Window
 - 4.5. Main Window
 - 4.5.1. Widget Windows
 - 4.5.2. Statusbar
 - 4.6. Menu Structure
 - 4.6.1. Project
 - 4.6.1.1. New Project

- 4.6.1.2. Open Project
 - 4.6.1.3. Edit Project
 - 4.6.1.4. Save Project
 - 4.6.1.5. Save Project As
 - 4.6.1.6. Recent Projects
 - 4.6.2. Edit
 - 4.6.2.1. Undo
 - 4.6.2.2. Redo
 - 4.6.2.3. Refresh
 - 4.6.2.4. Force Refresh
 - 4.6.2.5. Find
 - 4.6.3. View
 - 4.6.3.1. Search Window
 - 4.6.3.2. Graph Window
 - 4.6.3.3. Code Window
 - 4.6.3.4. Larger Font
 - 4.6.3.5. Smaller Font
 - 4.6.3.6. Switch Color Scheme
 - 4.6.4. Help
 - 4.6.4.1. About
 - 4.6.4.2. Licences
 - 4.6.4.3. Preferences
 - 4.6.5. Close Window
- 4.7. Shortcuts
- 4.8. Graph View
 - 4.8.1. Nodes
 - 4.8.2. Edges
- 4.9. Code View
 - 4.9.1. Files
 - 4.9.2. Snippets
- 4.10. Search View
 - 4.10.1. Undo & Redo
 - 4.10.2. Refresh
 - 4.10.3. Search bar
 - 4.10.4. Autocompletion Popup
- 5. CODE EDITOR PLUGINS
 - 5.1. From Coati
 - 5.2. To Coati
 - 5.3. Sublime Text
 - 5.3.1. Installation
 - 5.3.2. Use
 - 5.4. Visual Studio

- 5.4.1. Installation**
- 5.4.2. Use**

1. QUICK START GUIDE

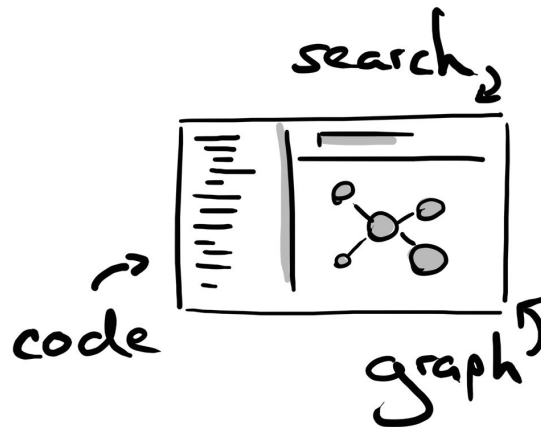
This short introduction will briefly guide you through the project setup and the user interface of Coati. The bullet point lists will tell you what to do:

Tasks:

- Keep on reading

1.1. Short description

Coati is an interactive source explorer that simplifies navigation in existing source code. Coati's aim is to give answers to all your questions about your source code. Coati first analyses your code and gathers data about its structure and then provides you with a simple interface consisting of three linked views, each having a key role in getting information:



- **Search:** Use the search field to quickly find and select analyzed symbols in your source code. The autocomplete box will instantly provide an overview of all matching results throughout your codebase.
- **Graph:** The graph displays the structure of your source code. It focuses on the currently selected symbol and directly shows all incoming and outgoing dependencies to other symbols.
- **Code:** The Code view displays all source locations of the currently selected symbol in a list of code snippets. Clicking on a different source location allows you to change the selection and dig deeper.

Coati currently only supports the language C and C++. Much of the UI design is therefore based on these languages and might change as soon as other languages are supported. For more information have a look at [supported languages](#).

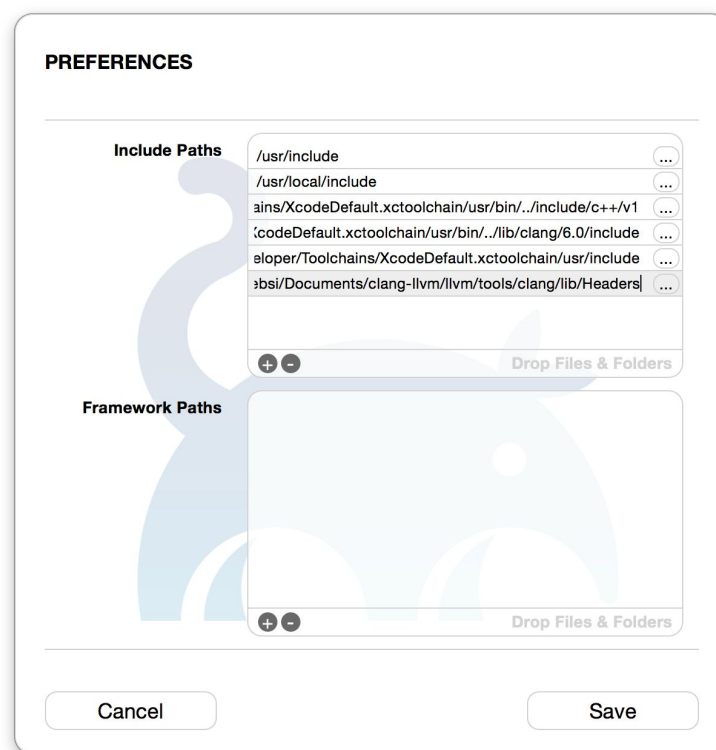
1.2. Starting up Coati

Once you [installed](#) Coati successfully you are ready to run the application.

Tasks:

- Run Coati.

When running Coati for the first time you are greeted with the [preferences window](#), which lets you define the location of the standard libraries on your system, so the analyzer will find them.

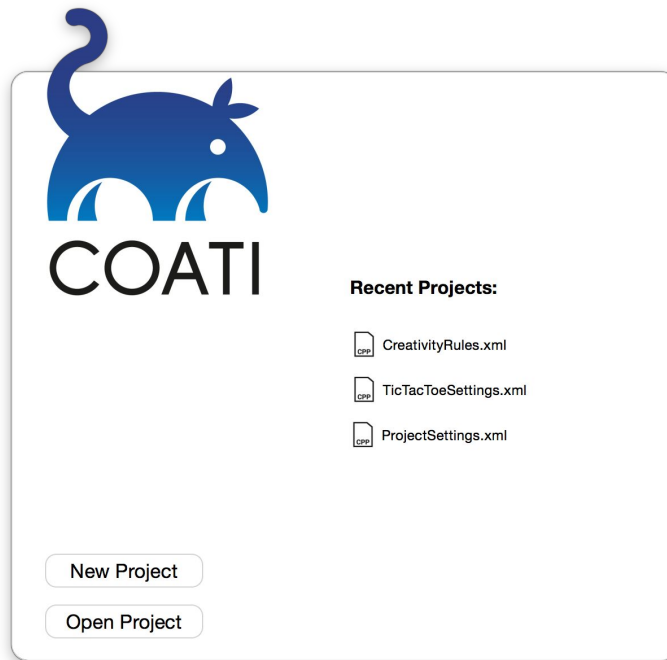


Tasks:

- Set the paths to your standard library headers and click "save" (see [preferences window](#) to find out how to get them on your platform)
- or click "cancel" and set them later through the [menu](#).

Start Window

On every start of Coati you are shown the [start window](#). It allows for creating new projects or opening existing ones.

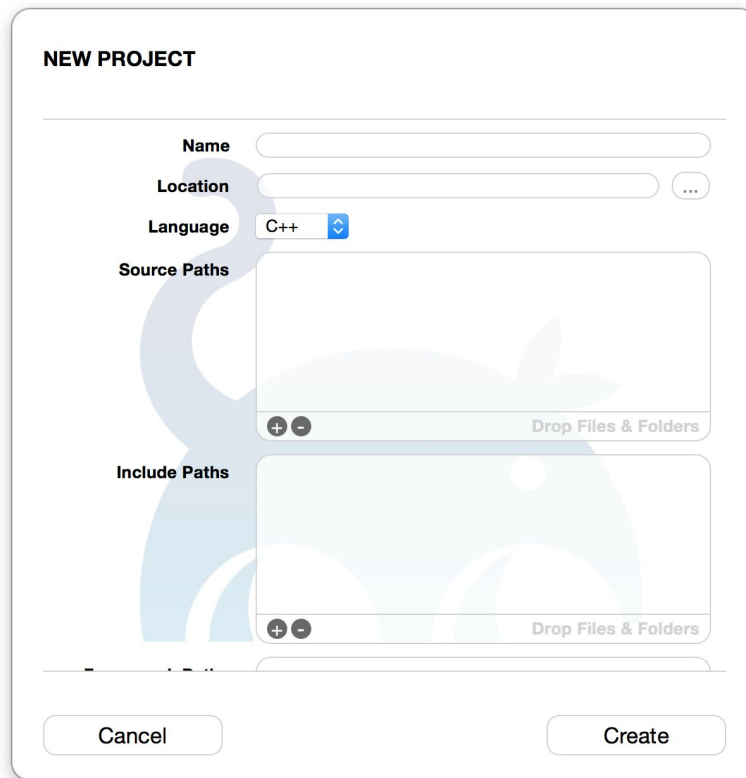


Tasks:

- Press "New Project" to continue creating a new project.
- or select the included recent project "TicTacToe" and continue with the [UI Intro](#).

1.3. Create New Project

Next you will see the [project setup window](#), which lets you create a new Coati project. "Name" and "Location" is used to define all where and how your Coati project file will be saved. The "Source Paths" list is used to define all source and header files that will be analysed by Coati. The "Include Paths" list is used to define all paths to additional header files that your source paths include, but they won't be analysed by Coati (e.g. external framework or libraries).

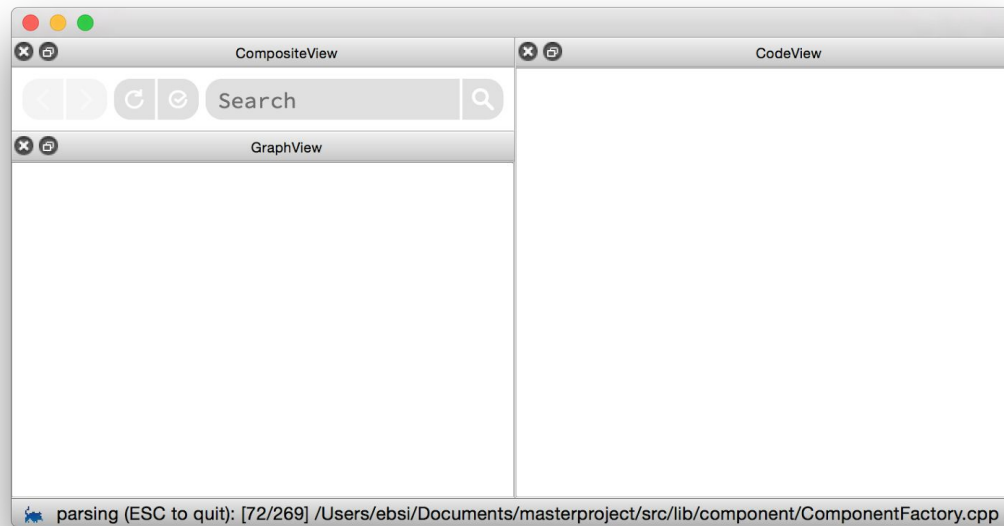
A screenshot of a 'NEW PROJECT' dialog box. It has a title bar 'NEW PROJECT'. Below it are four fields: 'Name' (a text input), 'Location' (a text input with a dropdown arrow), 'Language' (a dropdown menu showing 'C++'), and 'Source Paths' (a list box with a '+ -' button and a 'Drop Files & Folders' label). Below 'Source Paths' is an 'Include Paths' field, also a list box with a '+ -' button and a 'Drop Files & Folders' label. At the bottom are 'Cancel' and 'Create' buttons. A faint Android robot watermark is visible in the background.

Tasks:

- Enter your project name in the text field "Name".
- Select a location for your project file in "Location".
- Define the source files by adding at least one item to the "Source Paths" list. (see [Path List Box](#) on how)
- Similarly add "Include Paths" if the source files depend on headers that you don't want to analyse. (see [Path List Box](#) on how)
- Press "Create" to create your project.

1.4. Source Analysis

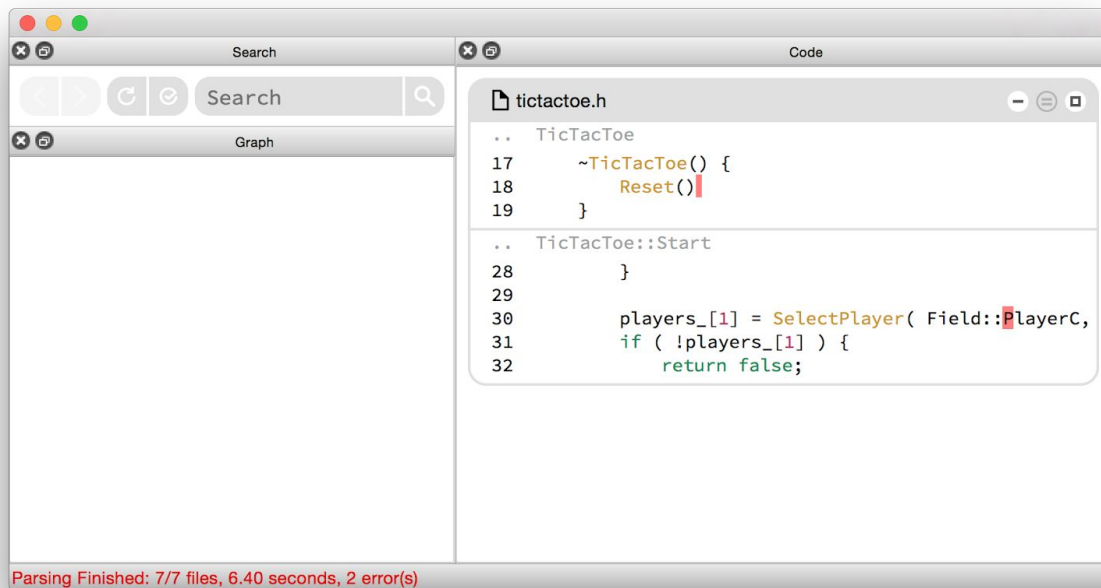
Coati will now start analyzing your source files, which might take a little while. The [status bar](#) will give you information about the progress. Otherwise the UI will be empty. Coati analyses all named symbols and their relationships throughout the provided source files. However, it stops the analysis at the level of local variables and symbols that are only valid within a certain function scope.



Tasks:

- Wait until the analysis of your source files has finished.
- or Press ESC to stop the analysis (Coati will provide all information gathered so far and the analysis can be continued later by [refreshing](#)).

After analysis has finished, Coati will look for the function main and activate it. In case it's not there, the first analyzed file will be activated. If the analysis found errors in your code, the code view will display them after analysis. You can see the error message by hovering the error location.

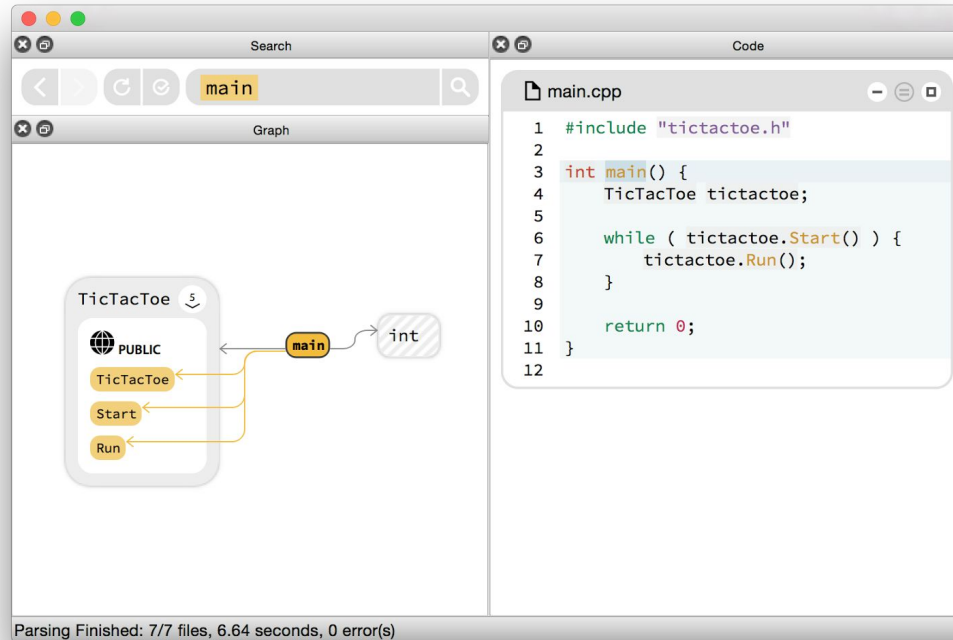


Tasks:

- Fix your errors and [refresh](#) to reanalyze the files with errors (Open Issue: As long as there was no change in the specific file, Coati won't reanalyze it, do a [force refresh](#) until fixed).
- or ignore them and continue with an incomplete analysis.

1.5. UI Intro

As already mentioned Coati's user interface is split into three main views. Their arrangement can be changed however you wish, you can even put them on different screens (see [Window Widgets](#)).



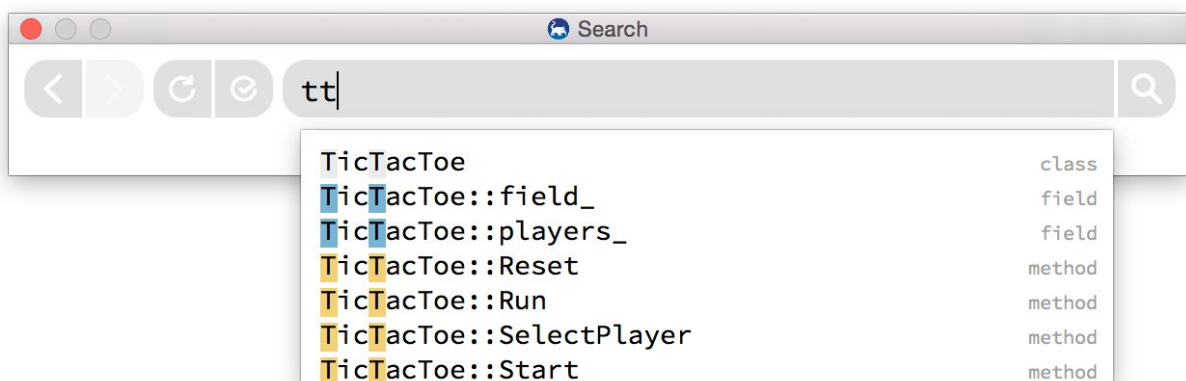
The three views show information about the currently selected symbol. The search field provides the name, the graph visualization shows its relationships to other symbols and the code view displays all locations the symbol gets referenced throughout the codebase.

1.5.1. Search Field

The [search field](#) allows for easy access to all analyzed symbols. Use it to find all the classes and functions you wish to investigate. Apart from that it also holds the UI buttons for [undo & redo](#) as well as [refreshing](#).



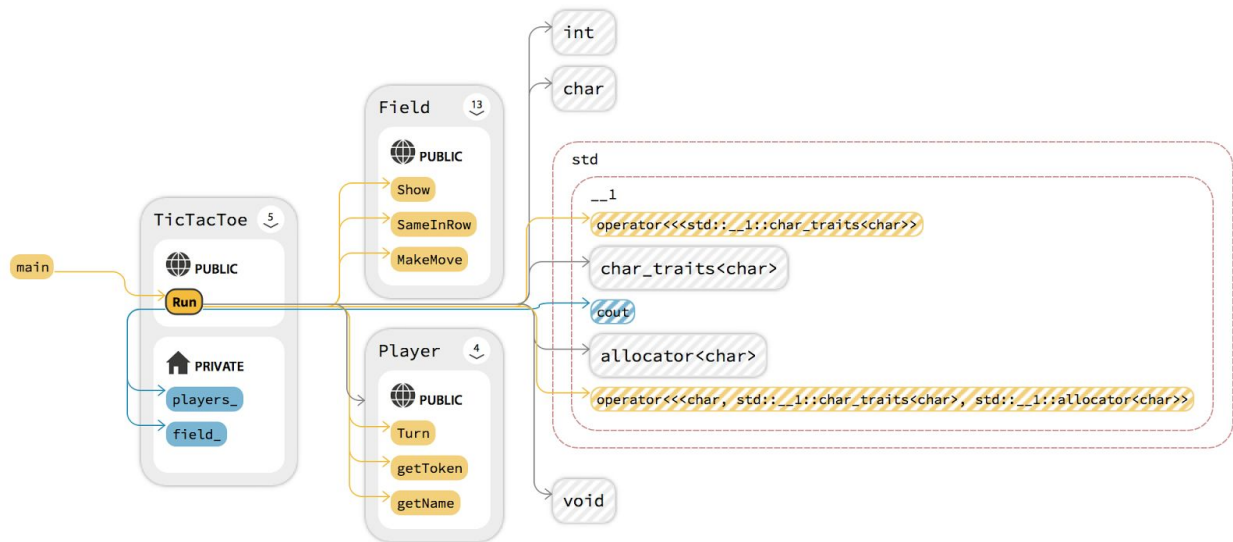
When entering a search query the [autocomplete popup](#) will provide you a concise list of all matching symbols. Note that Coati uses a fuzzy matching algorithm, that allows you to skip characters.



1.5.2. Graph Visualization

The [graph visualization](#) displays the currently selected symbol in an active state and all other symbols it shares relationships with. The visualization is made up of nodes and edges.

- **Nodes:** All named symbols in your source code will be displayed as different [nodes](#), such as functions, classes, files. Nodes with members, like classes, can be expanded to show all their contents, the number at the expansion arrow shows how many members are hidden. Click a node to activate it and update all the views to the new selection. Drag a node to change its position.
- **Edges:** The relationships between the symbols are displayed as different [edges](#), such as type use, function call, file include. Sometimes edges get bundled together and are displayed as an aggregation edge that includes a number of how many edges it contains. Click an edge to highlight its source location in the code view.



Colors:

The different node and edge types are also displayed using different colors. The default color scheme uses this convention:

Color	Node	Edge
gray	types and classes	type use
yellow	functions and methods	calls
blue	variables and fields	variable access

Hatching:

Nodes displayed with a striped hatching, are nodes that were used within your analyzed source files, but were not analyzed themselves. Clicking them shows you all locations where they are used, without providing where they got defined.



1.5.3. Code View

The [code view](#) displays all locations of the currently active symbol within the analyzed source files. It does not allow for editing the source code. Syntax highlighting is used to increase readability. Source locations with a grey highlight can be clicked to activate the symbol with this name. Active source locations have a blue highlight.



The source locations are displayed as code snippets, containing the line of the occurrence and extra lines added to the top and bottom to give information about its context. Code snippets are then bundled together into files.

Note: A file can be selected as active symbol by clicking its name in the title bar. By clicking the icons in the title bar, the file can switch between 3 different states:

- **Minimized:** The file does not show any content




- **Snippets:** The file displays the snippets containing active locations separated by lines.



```
1 #include "tictactoe.h"
2
3 int main() {
4     TicTacToe tictactoe;
5
6
7
8     }
9
10    int result = 0;
11    return result;
12 }
13
```

- **Maximized:** The whole content of the file is visible.



```
1 #include "tictactoe.h"
2
3 int main() {
4     TicTacToe tictactoe;
5
6     while ( tictactoe.Start() ) {
7         tictactoe.Run();
8     }
9
10    int result = 0;
11    return result;
12 }
13
```

For more information please have a look at [Code View Files](#).

1.6. Start exploring!

After reading this quick guide, you know the basics of Coati's user interface and you are able to start exploring your codebase. Coati will allow you to see your source code from a whole new perspective, by giving you a concise overview of its parts and a faster way of drilling down to its internals.

For more detailed information please have a look at the much more extensive instruction manual below.

If you have feedback for us, please don't hesitate and let us know by writing to support@coati.io, we'd be glad to hear from you.

The Coati team wishes you a good start with our product, lots of saved time, increased productivity and much cleaner code.

Tasks:

- Start exploring & have fun!

2. INTRODUCTION

2.1. About this document

This document is the official documentation of Coati and explains everything you need to know about working with it.

If you have questions not answered by this document, please write us an e-mail to support@coati.io.

2.2. Supported Languages

2.2.1. C

Supported by [Clang 3.6](#).

If you have a problem loading C code, please have a look at [Clang language compatibility](#).

2.2.2. C++

Supported by [Clang 3.6](#).

For more Information please visit [Clang C++ Status](#).

If you have a problem loading C++ code, please have a look at [Clang language compatibility](#).

2.3. Finding System Header Locations

2.3.1. on Windows

When using the Visual Studio IDE the system headers can be found at:
`path_to_visual_studio/VC/include/`

2.3.2. on Mac

Run this command in your terminal:
`gcc -x c++ -v -E /dev/null`

You will find the header search paths your compiler uses in the output section starting with:
#include <...> search starts here:

2.3.3. on Linux

```
gcc -x c++ -v -E /dev/null  
or  
clang -x c++ -v -E /dev/null
```

You will find the header search paths your compiler uses in the output section starting with:
#include <...> search starts here:

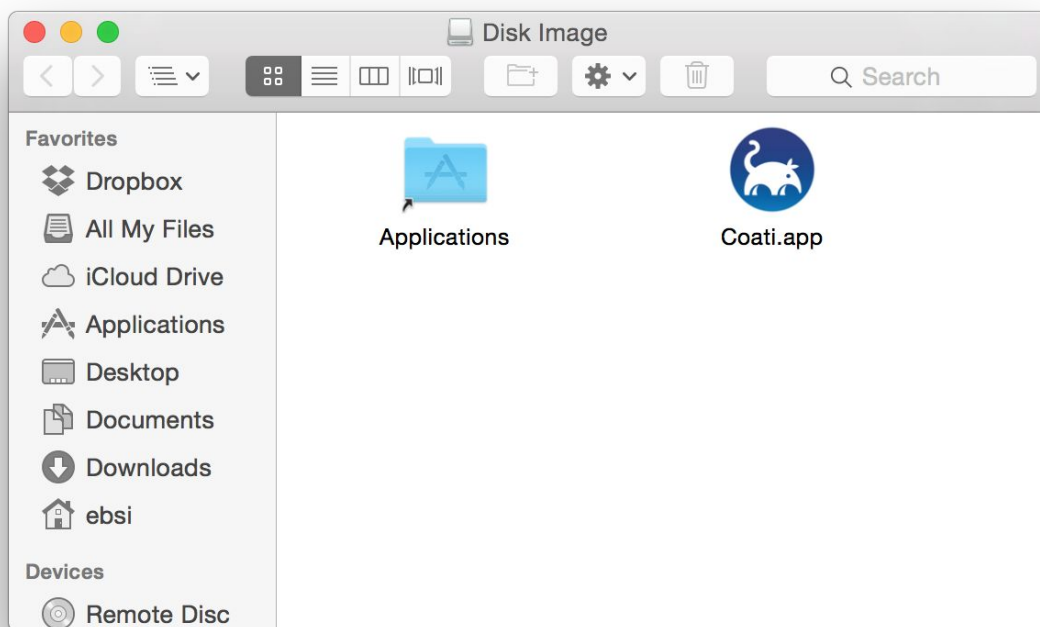
3. INSTALLATION

3.1. Windows

Download and open the Coati.rar file and extract its contents into a folder of your choice. You can now launch Coati by running the Coati.exe file.

3.2. OS X

Download and open the Coati.dmg file and drag Coati.app into the applications folder. You can now launch Coati from your Applications.



3.3. Linux

Download Coati.tar.gz file and extract it. To start Coati run the Coati.sh script. Coati creates a folder ~/.config/coati at the first run, this is the folder for Coati settings.

3.4. Data folder

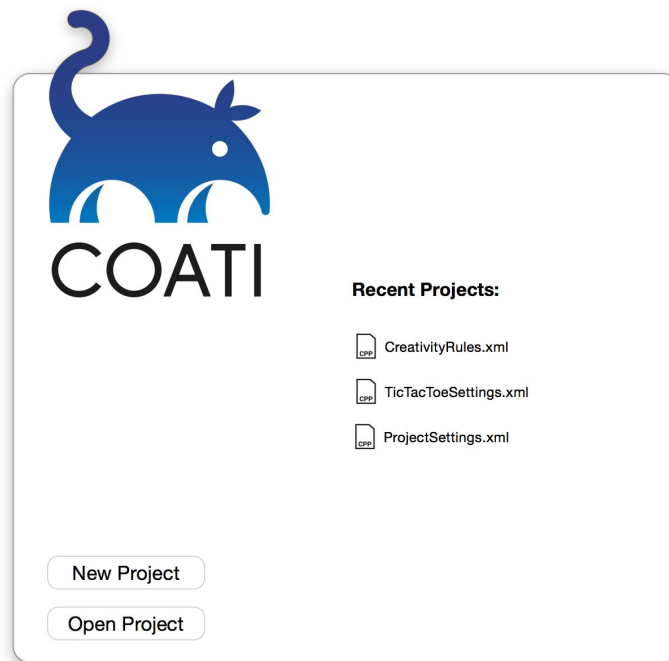
The data folder holds certain files that are used by Coati to run the program. After following the [installation instructions](#) the data folder should be located in the following locations on your platform.

Platform	Location
Windows	path_to_coati/data
Mac	/Applications/Coati.app/Contents/Resources/data
Linux	~/.config/coati

4. USER INTERFACE

4.1. Start Window

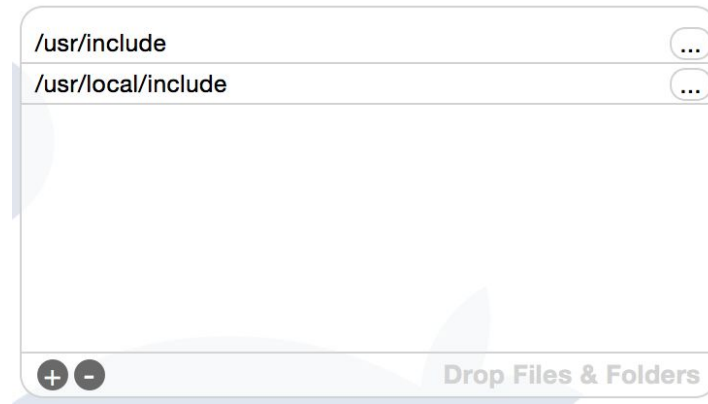
On every start of Coati you are shown the start window. It allows for creating new projects or opening existing ones.



- Clicking "New Project" will lead you to [Project Setup](#).
- Clicking "Open Project" will let you open an existing Coati project by choosing from a file dialog.
- Clicking on one of the "Recent Projects" will open this project. The list shows a maximum of 7 items ordered by recent first.
- Pressing ESC will close the window.

4.2. Path List Box

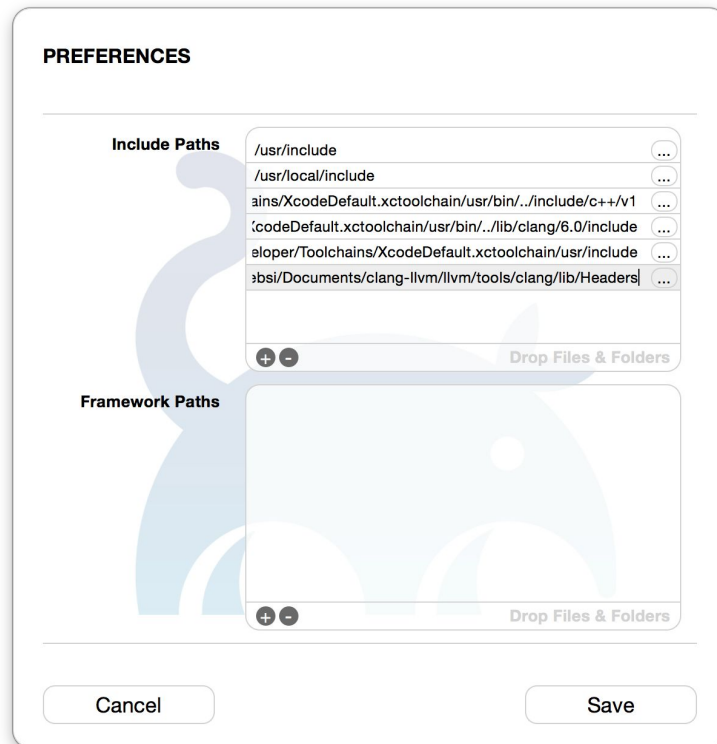
The Path List Box is a user interface element that is used within the [Preferences Window](#) and the [Project Setup Window](#). It allows for entering a list of file and directory paths.

**Interactions:**

- Click the "+" icon to add a new path line.
- Click the "-" icon to remove a selected path line.
- Click a path line to select it.
- Enter the path by typing on your keyboard
- Click "..." within the path line to open a file dialog for choosing a file or directory path.
- Directly add multiple paths into the box by dropping elements from your filesystem.

4.3. Preferences Window

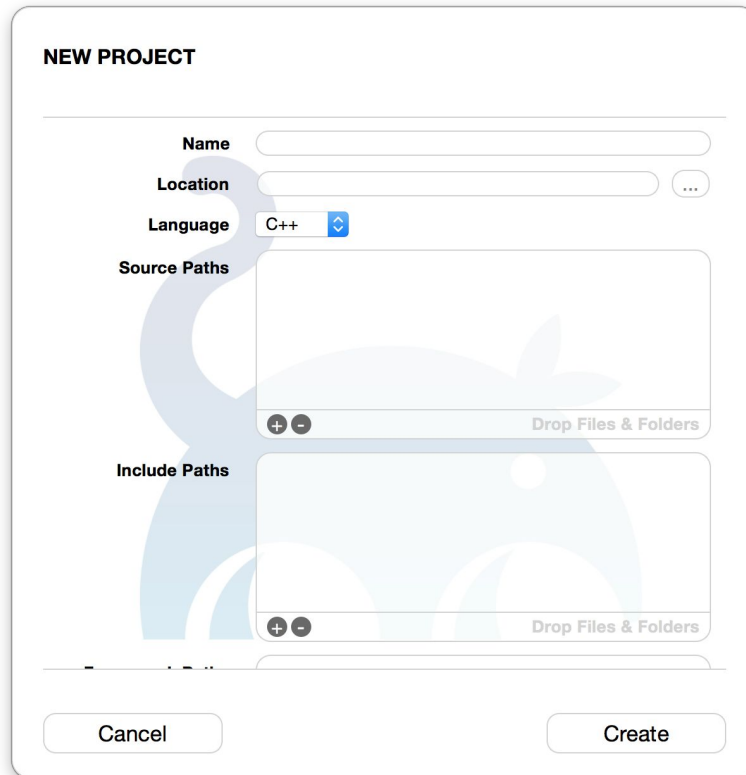
The Preferences window lets you define settings for all projects. The window automatically shows up when running Coati for the first time. You can open the Preferences from the menu via [Help/Preferences](#).



Setting	Description
Include Paths	Set header search paths for all of your projects (e.g. std headers). These files won't be analyzed by Coati. (For instructions on how to add paths see Path List Box . For instructions on how to find the system header paths see Finding System Header Locations)
Framework Paths	Mac only. Define the search paths for .framework files for all of your projects. (For instructions on how to add paths see Path List Box .)

4.4. Project Setup Window

The Project Setup Window lets you create a new Coati project.



NEW PROJECT

Name

Location ...

Language C++ ▾

Source Paths

+ - Drop Files & Folders

Include Paths

+ - Drop Files & Folders

Cancel Create

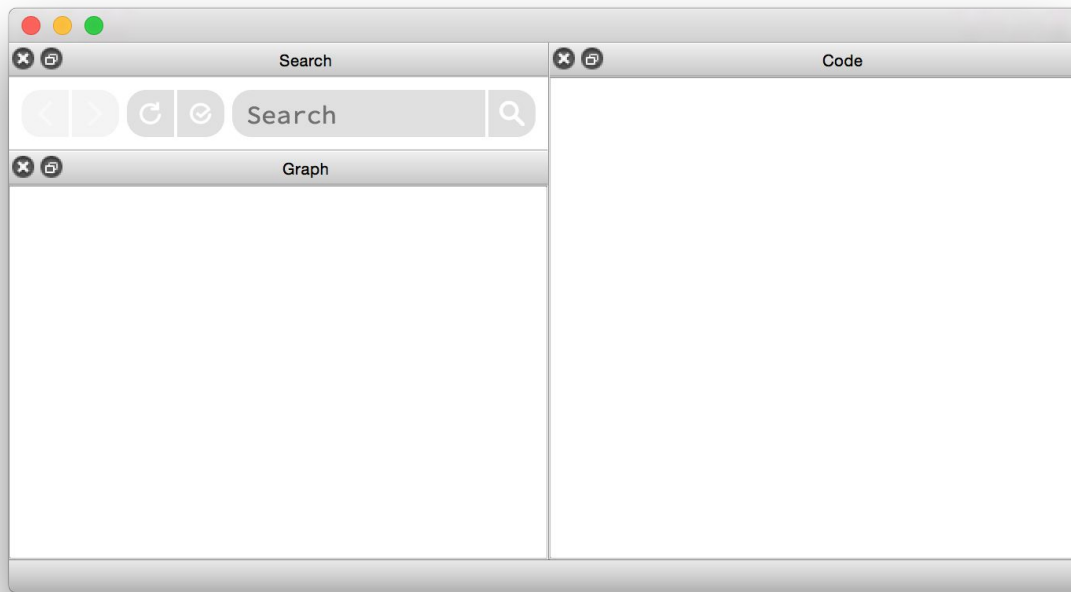
- Clicking "Cancel" will close the window.
- Clicking "Create" will check your inputs, save the new project file and start analyzing the source files.
- Pressing ESC will close the window.

Setting	Description
Name	The name of the project. This will also be the name of the project file.
Location	Choose the location of the project file from the dialog.
Language	Select the language of your project. (See Language Support)
Source Paths	Set the source paths of your projects that will be analyzed by Coati. (For instructions on how to add paths see Path List Box .)
Include Paths	Set additional header search paths that your source paths depend on. These won't be analyzed by Coati. (For instructions on how to add paths see Path List Box .)
Framework Paths	Mac only. Define the search paths for .framework files for your project. (For instructions on how to add paths see Path List Box .)

4.5. Main Window

4.5.1. Widget Windows

Coati's three views are organized into Widgets Windows, which can be freely arranged within the Main Window or detached from it. Each Widget Window has a title bar displaying the name of the view and 2 buttons for closing the Widget Window and for detaching it from the Main Window.



Interactions:

- Drag the Widget Window at the title bar to rearrange it within the Main Window, detach it or attach it again.
- Press the "x" icon to close the Widget Window. They can be reopened from the [View Menu](#).
- Press the "□" icon to detach the Widget Window from the Main Window.

4.5.2. Statusbar

The Statusbar is located on the bottom of the [Main Window](#) and is used to convey information about Coati's status and running processes to the user.

2 results with 4 references in 1 file

4.6. Menu Structure

4.6.1. Project

4.6.1.1. New Project

Shortcut: [New Project](#)

Opens the [New Project Dialog](#) to define a new project and loads it after creation.

4.6.1.2. Open Project

Shortcut: [Open Project](#)

Opens a file dialog to choose an existing Coati project file from your system's hard drive.

4.6.1.3. Edit Project

Opens the [New Project Dialog](#) prefilled with your project settings and allows for changing them.

4.6.1.4. Save Project

Shortcut: [Save Project](#)

Save the project file.

4.6.1.5. Save Project As

Shortcut: [Save Project As](#)

Save the project file to another location. The project file will be duplicated.

4.6.1.6. Recent Projects

Opens a submenu to choose recent opened Coati projects.

4.6.2. Edit

4.6.2.1. Undo

Shortcut: [Undo](#)

Undoes the last user action. Most of the interactions within Coati can be undone, such as searching, node selection, file selection and so on.

4.6.2.2. Redo

Shortcut: [Redo](#)

Redoes an undone action.

4.6.2.3. Refresh

Shortcut: [Refresh](#)

Refresh will check all analyzed source files for updates and reanalyze the ones that changed and their depending ones.

4.6.2.4. Force Refresh

Shortcut: [Force Refresh](#)

Force Refresh will reanalyze the whole project.

4.6.2.5. Find

Shortcut: [Find](#)

This option will put the focus into the search field, so you can start typing your search query. Alternatively you can click the search field.

4.6.3. View

4.6.3.1. Search Window

Toggle the visibility of the Search Window. This can also be done by closing the Search Window on clicking the "x" icon in it's title bar. (See [Window Widgets](#))

4.6.3.2. Graph Window

Toggle the visibility of the Graph Window. This can also be done by closing the Graph Window on clicking the "x" icon in it's title bar. (See [Window Widgets](#))

4.6.3.3. Code Window

Toggle the visibility of the Code Window. This can also be done by closing the Code Window on clicking the "x" icon in it's title bar. (See [Window Widgets](#))

4.6.3.4. Larger Font

Shortcut: [Larger Font](#)

Increase the font size within the Main Window's user interface.

4.6.3.5. Smaller Font

Shortcut: [Smaller Font](#)

Decrease the font size within the Main Window's user interface.

4.6.3.6. Switch Color Scheme

Opens a file dialog to choose another color scheme XML file. The color schemes are located in [data/color_schemes/](#).

4.6.4. Help

4.6.4.1. About

Shows copyright information about Coati.

4.6.4.2. Licences

Documents which third party licences Coati is using.

4.6.4.3. Preferences

Shortcut: [Preferences](#)

Opens the [Preferences Window](#).

4.6.5. Close Window

Shortcut: [Close Window](#)

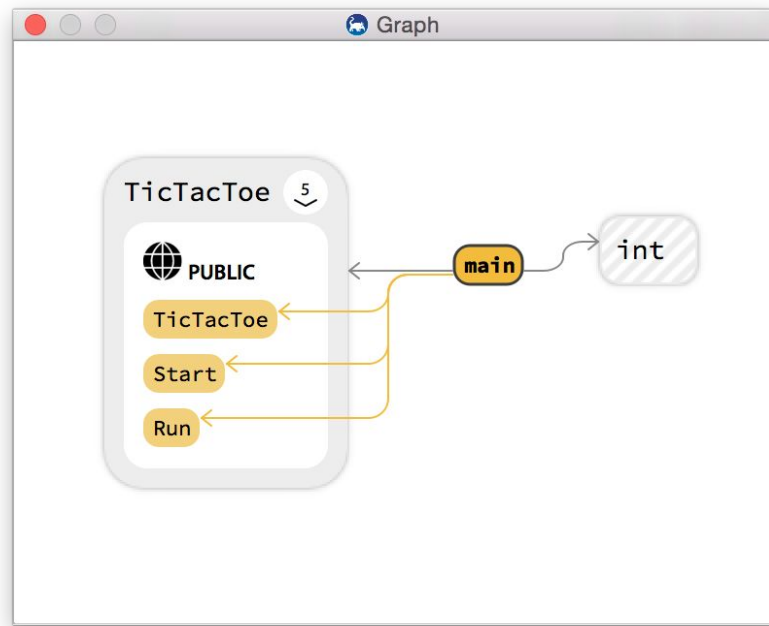
Closes the Coati window.

4.7. Shortcuts

Shortcut	Windows	Mac OS X	Linux
Preferences		Cmd + ,	
New Project	Ctrl + N	Cmd + N	Ctrl + N
Open Project	Ctrl + O	Cmd + O	Ctrl + O
Save Project	Ctrl + S	Cmd + S	Ctrl + S
Save Project As	Ctrl + Shift + S	Cmd + Shift + S	Ctrl + Shift + S
Close Window	Alt + F4	Cmd + W	Ctrl + W
Hide Window		Cmd + H	
Refresh	F5	Cmd + R	F5
Force Refresh	Shift + F5	Cmd + Shift + R	Shift + F5
Undo	Ctrl + Z	Cmd + Z	Ctrl + Z
Redo	Ctrl + Shift + Z	Cmd + Shift + Z	Ctrl + Shift + Z
Find	Ctrl + F	Cmd + F	Ctrl + F
Larger Font	Ctrl + '+'	Cmd + '+'	Ctrl + '+'
Smaller Font	Ctrl + '-'	Cmd + '-'	Ctrl + '-'

4.8. Graph View

The graph view visualizes the currently selected symbol and all its relationships to other symbols as an interactive graph visualization.


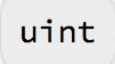


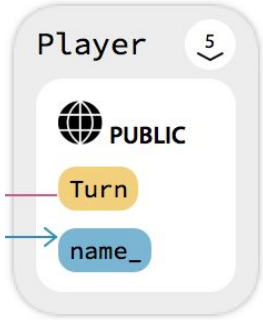
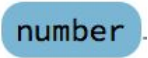
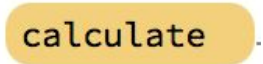

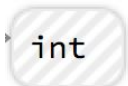

Interactions:

- Scroll left-right and up-down for big graphs.
- Drag the background area to move the whole graph for big graphs.

4.8.1. Nodes

Colors are corresponding to the default color scheme.

Node Type	Image
File	 tictactoe.h
Type & Typedef & Template Parameter	 uint





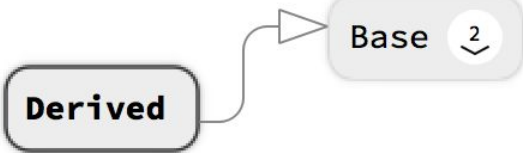
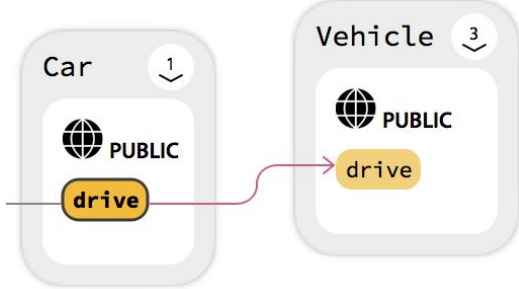
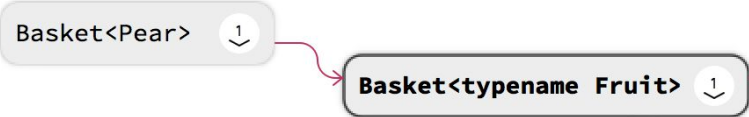
<p>Class & Struct: Display their members nested, and separated by access type: public, protected, private. By default only members with edges are shown. The arrow icon allows to expand and collapse them. The number tells how many nodes are hidden.</p>	
<p>Variable & Field</p>	
<p>Function & Method</p>	
<p>Enum</p>	
<p>Undefined: Nodes that were not defined within the analyzed files are shown with hatched background.</p>	
<p>Bundle: A bundle node combines multiple nodes to reduce the size of the graph visualization. The name describes what kind of nodes are bundled. The number tells how many nodes are bundled.</p>	

Interactions:

- Click a node to activate it.
- Drag a node to change its position.
- Click the arrow icon in class nodes to expand and collapse it.
- Click a bundle node to expand it.
- Hover a node to see a tooltip that displays the node's type.

4.8.2. Edges

Colors are corresponding to the default color scheme.

Edge Type	Image
File Include	
Type Use	
Variable Use	
Function Call	
Inheritance	
Method Override	
Template Parameter Use & Template Argument Use	

Aggregation:
Bundles multiple edges between the child nodes of the 2 nodes. The number tells how many edges are bundled. The arrow tells whether all edges point in a certain direction or not.



Interactions:

- Click an edge to see its location in the [Code View](#).
- Click an aggregation edge to activate all its corresponding edges.
- Hover an edge to see a tooltip that displays the edge's type.

4.9. Code View

The code view displays the corresponding source code of the currently selected symbols. The code view contains a list of one or more files.

```
Code
main.cpp
1  #include "tictactoe.h"
2
3  int main() {
4      TicTacToe tictactoe;
5
6      while ( tictactoe.Start() ) {
7          tictactoe.Run();
8      }
9
10     return 0;
11 }
12
```

The screenshot shows a window titled 'Code' with a file named 'main.cpp'. The code is displayed with line numbers from 1 to 12. The code includes a header file, defines a TicTacToe struct, and contains a main function that calls Start() and Run() methods in a while loop.

Interactions:

- Scroll up and down to see the different source files.

4.9.1. Files

Each file has a title bar with the file's name and buttons to change its display state. There are 3 different states:

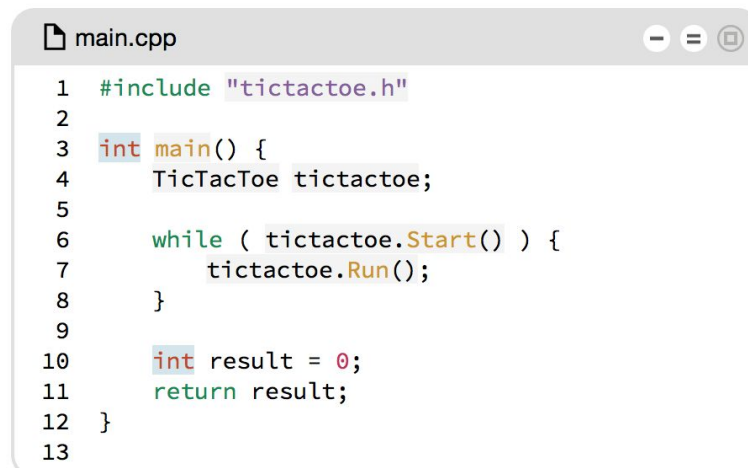
- **Minimized:** The file does not show its content.



- **Snippets:** The file displays the snippets containing active locations separated by lines.



- **Maximized:** The whole content of the file is visible.



Interactions:

- Hover the title to see the full file path.
- Click the title to activate the file's corresponding node.
- Click one of the three icons on the right to change the display state of the file.

4.9.2. Snippets

A code snippet contains the line with the currently active symbols highlighted in blue. Other symbols that were analyzed by Coati are highlighted in gray. In case the snippet is part of a class, function or namespace, an additional line at the top of the snippet provides information about the snippet's context (e.g. the surrounding scope).

```
.. Field::Show
55
56     for ( int col = 0; col < 3; col++ ) {
57         if ( grid_[row][col] == PlayerA ) {
58             io::stringOut(" X ");
59         } else if ( grid_[row][col] == PlayerB ) {
60             io::stringOut(" O ");
61         } else {
```

Interactions:

- Click the top line to show the whole scope around the snippet.
- Click the gray highlights to activate the responding symbol.

4.10. Search View

The Search View contains the search field for searching symbols and some other related user interface elements.



4.10.1. Undo & Redo

The left undo button lets you undo your last actions (see [Undo](#)) and the right redo buttons lets you redo your undone actions again (see [Redo](#)). Both buttons are only enabled when the respective actions are available at the moment.



Interactions:

- Hover the buttons to see a tool tip.
- Press the buttons to execute its action.

4.10.2. Refresh

The left refresh button allows you to refresh the current project and reanalyze all updated files (see [Refresh](#)). With the right auto refresh toggle button you can activate automatic refresh, which will reanalyze all updated files whenever you put window focus on Coati's [Main Window](#).



Interactions:

- Hover the buttons to see a tooltip.
- Press the left refresh button to refresh the project.
- Press the right auto refresh button to toggle automatic refreshing.

4.10.3. Search bar

The search bar allows you to enter search requests to find one of Coati's analyzed symbols. It doesn't allow for full text searching across all files so far. The search field allows for most text editing interactions common to text fields. When typing your request the [Autocompletion Popup](#) will show you search results matching to your entered string.

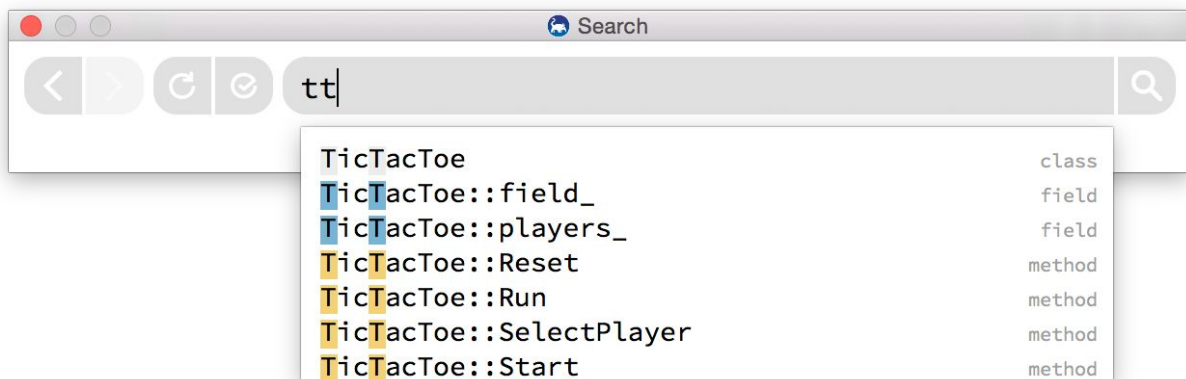


Interactions:

- Focus the search field by clicking it or using the [Find Action](#).
- Enter your search request by typing on your keyboard.
- By pressing enter or clicking on the search icon on the right you send your request.
- The search field allows for most interactions known from other text fields such as moving the cursor, copy&paste and text selection.

4.10.4. Autocompletion Popup

The Autocompletion Popup displays all [Nodes](#) matching your search request within all analyzed symbols. The match results are determined by a fuzzy matching algorithm, that allows you to skip characters. The popup shows which characters in the words are matching and displays their corresponding node color. The node type is displayed on the right.



Interactions:

- Use the up and down arrow keys to switch between search results.
- Pressing tab or clicking on the search result will insert it into the search field.
- Pressing enter will select the search result and send the search request.

5. CODE EDITOR PLUGINS

In order to make Coati the perfect partner for your development workflow you can connect Coati with different code editors. You can find the plugins in Coati's download package located in the folder [ide_plugins](#). Have a look at the following list of supported code editors to find out what editors are currently supported. If you can't find a plugin for the code editor you are using, please let us know by writing to support@coati.io.

Supported Editors:

- Sublime Text 2
- Sublime Text 3
- Visual Studio 2012
- Visual Studio 2013
- Visual Studio 2015

The communication between Coati and the code editor is achieved with a TCP connection. Coati uses the port 6667 to listen for incoming messages. Outgoing messages will be sent to the port 6666.

Outgoing messages are in the form:

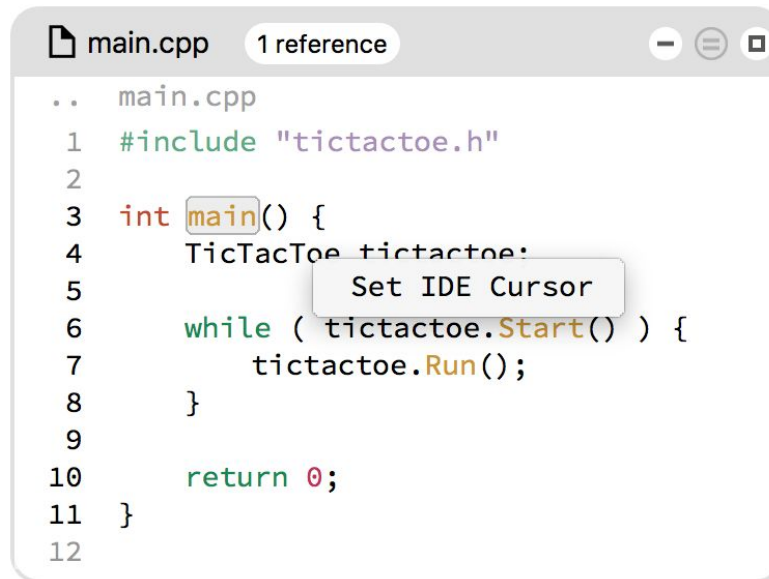
```
moveCursor>>absolute/file_path>>line_number>>column_number<EOM>
```

Incoming messages are in the form:

```
setActiveToken>>absolute/file_path>>line_number>>column_number<EOM>
```

5.1. From Coati

If you want your editor to open a file at a specific location from within Coati, you can achieve this by either selecting the option "Set IDE Cursor" from the right-click menu in the [Code View](#) or by simply clicking into a line in the [Code View](#) while holding down the CTRL or CMD key.



```
.. main.cpp
1  #include "tictactoe.h"
2
3  int main() {
4      TicTacToe tictactoe;
5      while ( tictactoe.Start() ) {
6          tictactoe.Run();
7      }
8
9      return 0;
10 }
11
12
```

5.2. To Coati

By using a Coati plugin for your code editor, you can select a location within a source file and Coati will show you all symbols found at this location. Please have look at the list below to see which plugins are currently available and how they are used.

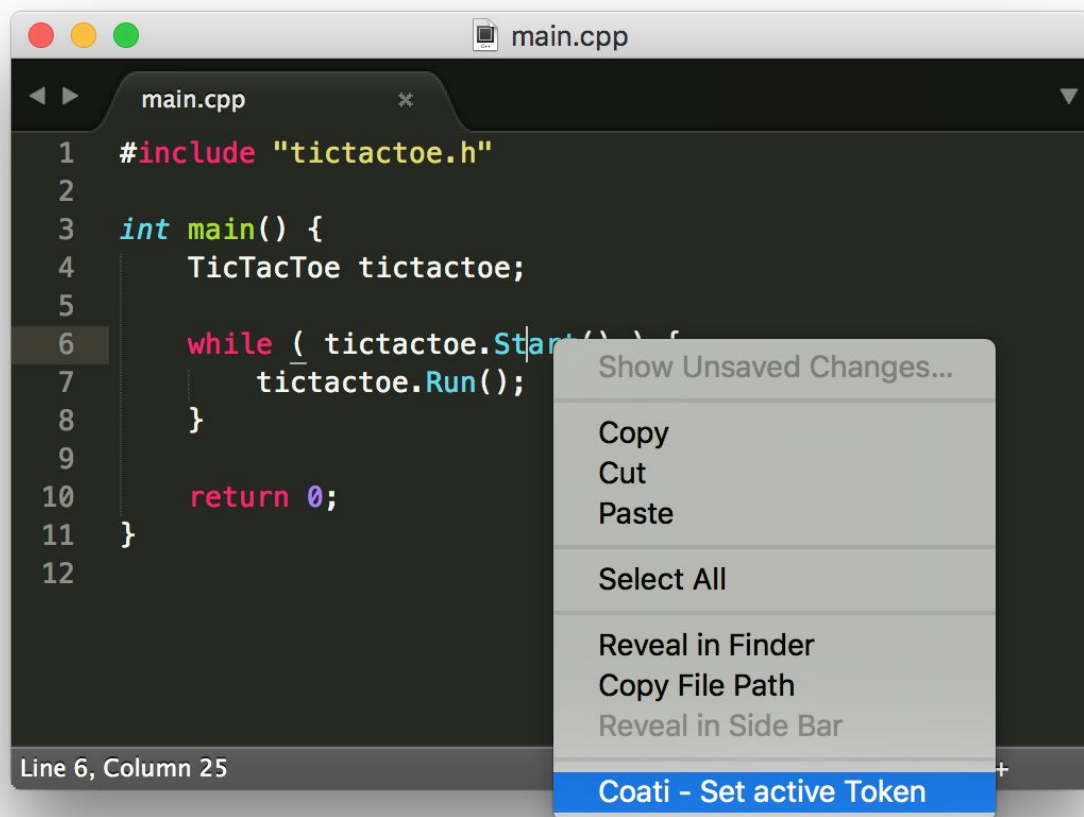
5.2.1. Sublime Text

5.2.1.1. Installation

To install the Coati plugin for Sublime Text copy the CoatiPlugin folder located in your install_directory/plugins/sublime_text to your SublimeText/Packages folder and restart Sublime.

5.2.1.2. Use

If you want Sublime to activate a certain element in Coati, click a location to place the cursor , right-click to bring up the context menu and choose the “Coati - Set active Token” option. Please note that the position of the cursor will be sent to Coati and not the position you opened the context menu at.



5.3. Visual Studio

5.3.1. Installation

To install the Coati plugin for any version of Visual Studio, just execute the corresponding vsix file located in your `install_directory/plugins/visual_studio`.

5.3.2. Use

If you want Visual Studio to activate a certain element in Coati, right-click that element to bring up the context menu and choose the “Set active Token” option.

main.cpp

```
1  #include "tictactoe.h"
2
3  int main() {
4      TicTacToe tictactoe;
5
6      while ( tictactoe.Start() ) {
7          tictactoe.Run();
8      }
9
10     return 0;
11 }
12
```

- Insert Snippet... Ctrl+K, Ctrl+X
- Paste Ctrl+V
- Outlining
- Guidelines
- Set active Token
- Source Control