Eclipse IDE and Debugging

Ravikishore Kommajosyula

What is an IDE?

- Software application that provide several features that assist a computer programmer in efficient software development.
- An IDE typically consists of:
  - Source code editor
  - Build automation tools
  - Debugger
  - Class browser
  - Automatic code completion

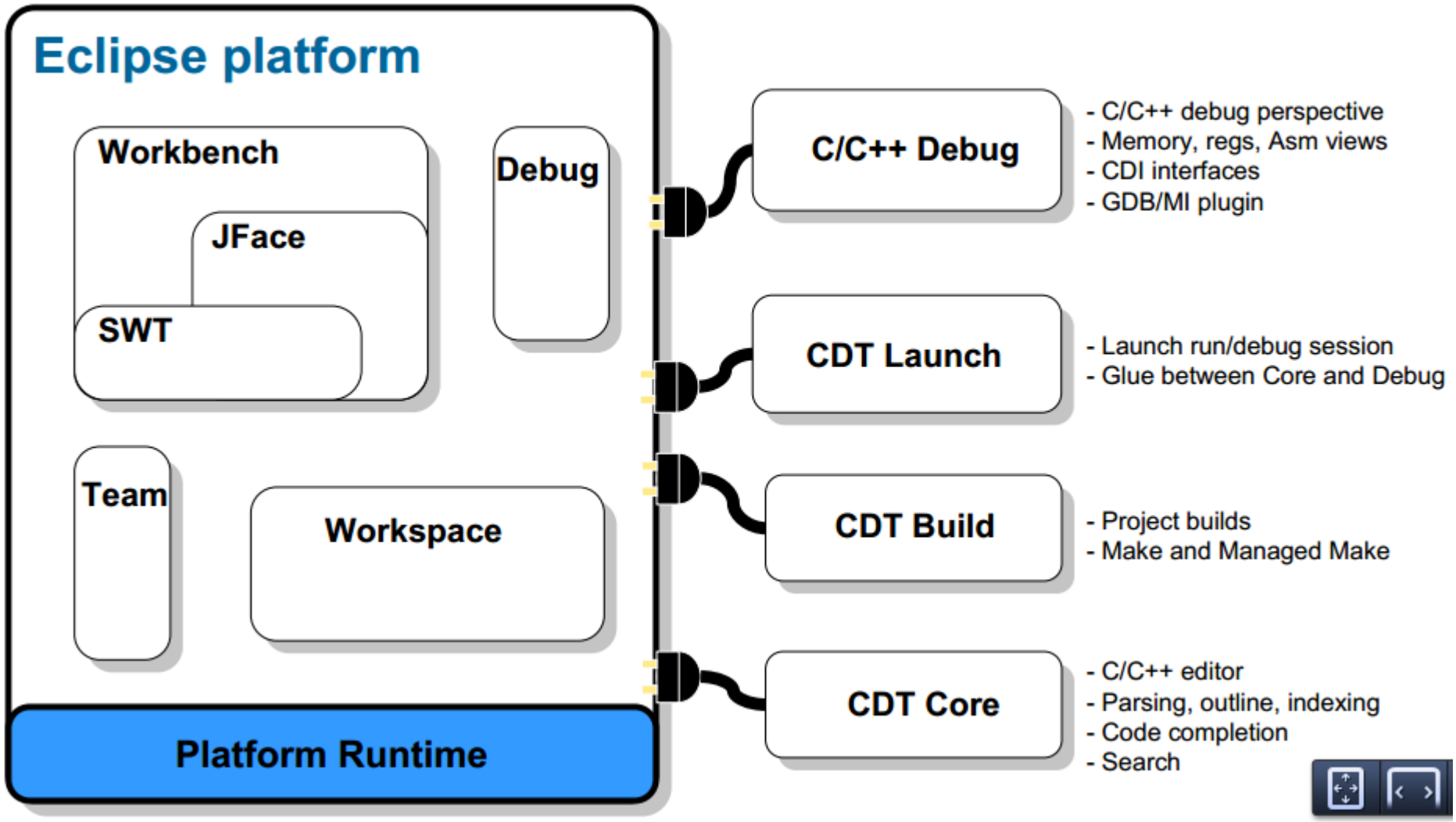
Advantages and Disadvantages

- Advantages:
  - Makes you efficient, helps organize resources, prevent mistakes, provide shortcuts
  - Automates build process
  - Project management tools : Version controlling, Documentation tools etc.
- Disadvantages:
  - Using GUI consumes more resources, and may not be possible always (SSH)
  - Learning to use all features is time consuming

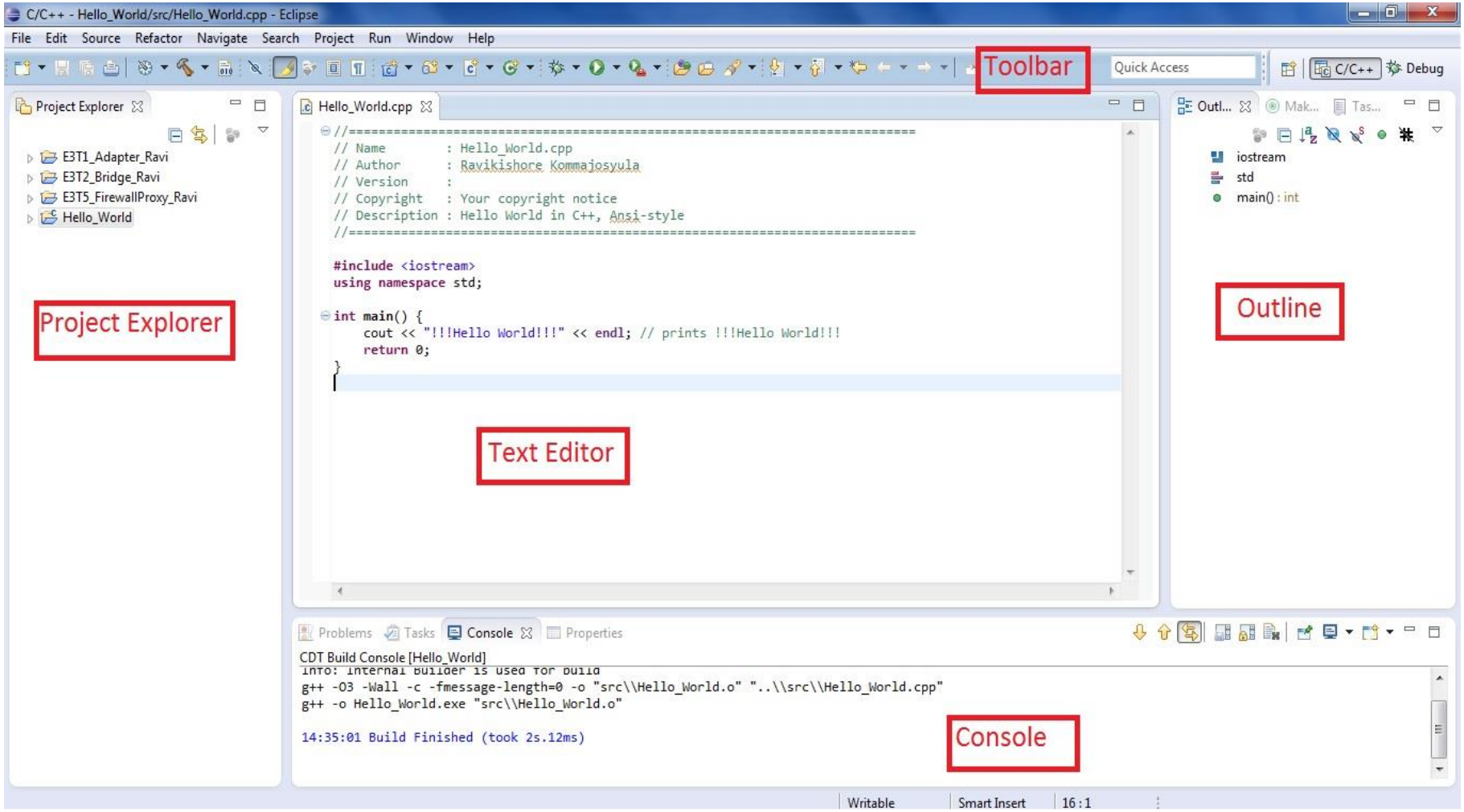
Eclipse CDT

- Provides fully functional C and C++ IDE based on the Eclipse platform
- Automates build process
- Installation:
  - http://www.eclipse.org/cdt/downloads.php
  - Unzip the package and open the eclipse binary

Eclipse IDE

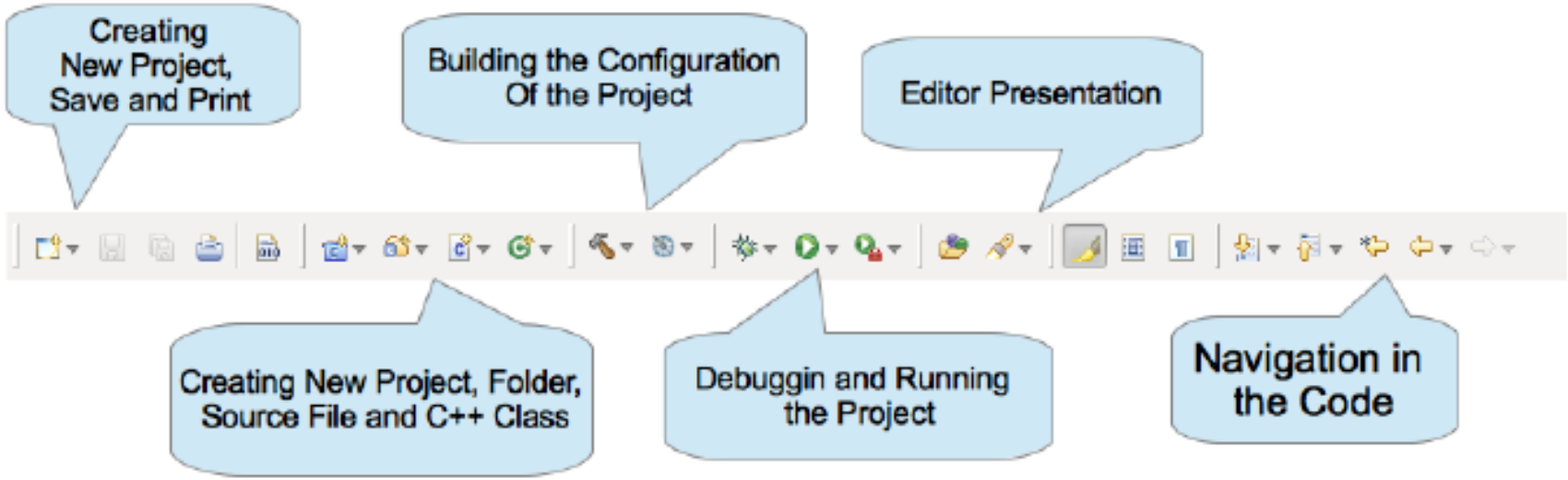

Eclipse platform
Workbench
JFace
SWT
Debug
Team
Workspace
Platform Runtime
C/C++ Debug
- C/C++ debug perspective
- Memory, regs, Asm views
- CDI interfaces
- GDB/MI plugin
CDT Launch
- Launch run/debug session
- Glue between Core and Debug
CDT Build
- Project builds
- Make and Managed Make
CDT Core
- C/C++ editor
- Parsing, outline, indexing
- Code completion
- Search

Eclipse CDT User Interface



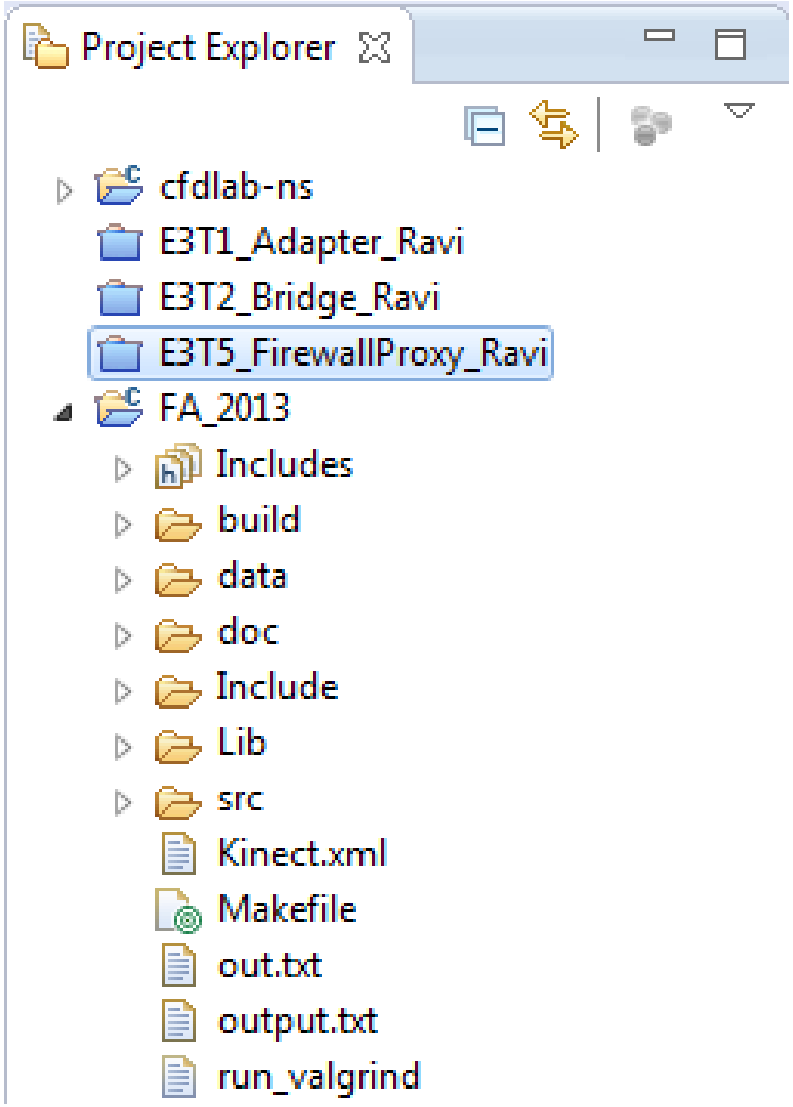
Eclipse Toolbar

- Quick access to several frequently used commands / functionalities

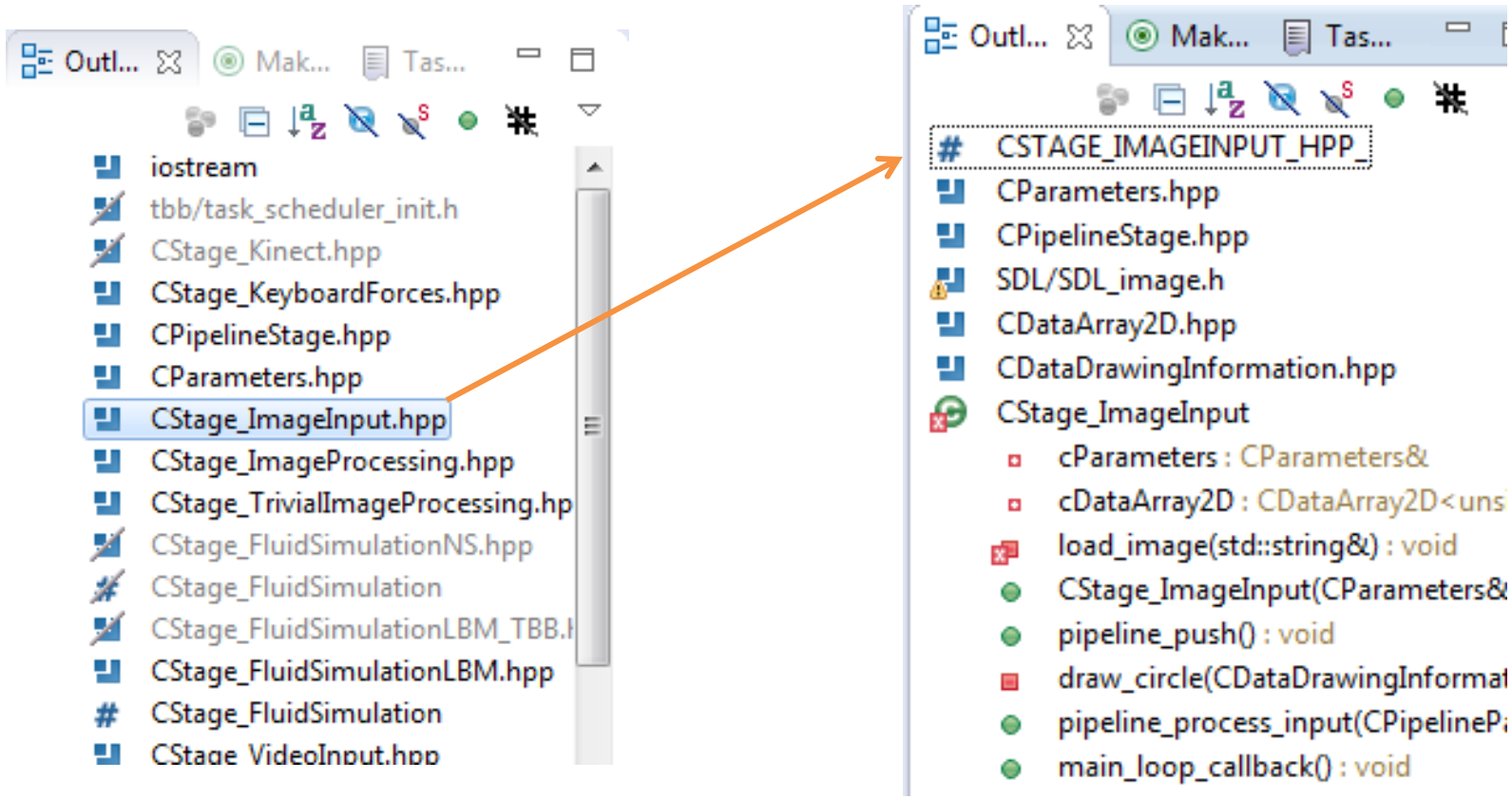


Project Explorer

- Provides a handy way of exploring files in one project / open and close different projects in the workspace



Outline



- Gives a quick view of the contents of the file. Very useful to understand a new code / debug

Other key features

- Integration with Git and SVN
- Code auto completion – *Ctrl + Space*
- Navigate to function declaration / header – *Ctrl + Left-Click or F3*
- Code auto formatting – *Ctrl + Shift + F*
- Code refactoring– *Alt + Shift + R, Alt+ Shift + M*
- Commenting block of code – *Ctrl + /*
- And many many more…

Debugging

Debugging is the process of locating and fixing errors (bugs) in the code

- Classical approach to debugging – printf()….not the smartest way to go!!
- Debugging information can be generated from code by adding the –g flag
- Compiler optimization flags might alter code
- Examples – gdb, valgrind, visual studio etc.

Why we need Debuggers?

- Seeing the source code during the execution of a statement
- Pause the execution at any point in the source code
- Read the current state of the program when execution is paused
- See the call stack, set break points, step into / out of a function etc.

Call Stack

- Program execution starts by calling main function
- Several functions are called within the program
- All function calls are stored (along with local variables and return address) on the Call Stack
- Call stack gives very helpful information for debugging
- Especially helpful for recursive algorithms

Breakpoints

Program execution stops and control is handed over to the user once the execution hits a Breakpoint

- Very powerful tool when used in the right way
- Special types of breakpoints give advanced features:
  - Watch point : Program execution is stopped whenever a particular data value is changed (track a variable)
  - Conditional breakpoint: Program execution is stopped whenever a condition is met (dt < 0)

Stepping

Stepping is the process of stepping into the code and running it one step at a time

- Step Into: Executes current statement. If the statement is a function, it steps into the function
- Step Over: Executes current statement. If the statement if the function, it executes the whole function (step over) and stops at next statement
- Step Out: Steps out of current function
- Continue: Execution till the end / next breakpoint