IDEs and Debugging

Arash Bakhtiari
bakhtiar@in.tum.de

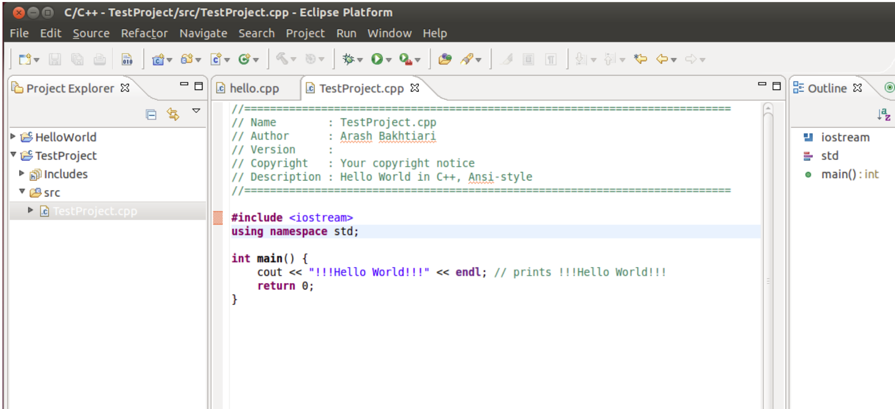
2012-11-06 Tue

Outline

What are IDEs

IDE(Integrated development environment):

- collection of facilities with graphical interface (Text Editor, Compile and Build, Debugging Tools)
- examples: Eclipse, Code::Blocks, NetBeans, Qt Creator, MS Visual Studio



C/C++ - TestProject/src/TestProject.cpp - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
HelloWorld
TestProject
Includes
src
TestProject.cpp
hello.cpp
TestProject.cpp
Outline
iostream
std
main() : int

```
//============================================================================
// Name        : TestProject.cpp
// Author      : Arash Bakhtiari
// Version     :
// Copyright   : Your copyright notice
// Description : Hello World in C++, Ansi-style
//============================================================================

#include <iostream>
using namespace std;

int main() {
    cout << "!!!Hello World!!!" << endl; // prints !!!Hello World!!!
    return 0;
}
```

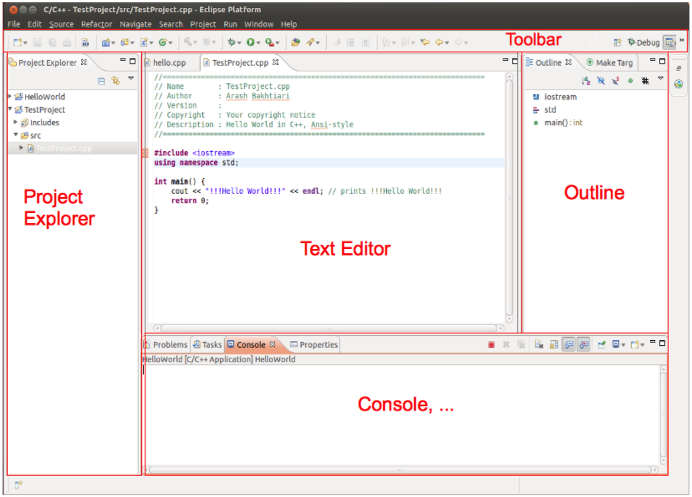
Advantages and Disadvantages

- Advantages:
  - less time and effort: helps you organize resources, prevent mistakes, and provide shortcuts
  - project management: (documentation tools, visual presentation of resources)
- Disadvantage:
  - using graphical interface requires more memory and processing power
  - will not fix bad code, practices, or design(sometimes cause bad code!!!)

Installing Eclipse CDT

- Eclipse CDT: provides a fully functional C and C++ IDE based on the Eclipse platform.
- Installing on Ubuntu:
    - sudo apt-get install eclipse-cdt
- Installing on Windows:
    - go to the link: http://www.eclipse.org/downloads/
    - download “Eclipse IDE for C/C++ Developers”

Eclipse User Interface Overview



```
//============================================================================
// Name        : TestProject.cpp
// Author      : Arash Bakhtiari
// Version     :
// Copyright   : Your copyright notice
// Description : Hello World in C++, Ansi-style
//============================================================================

#include <iostream>
using namespace std;

int main() {
	cout << "!!!Hello World!!!" << endl; // prints !!!Hello World!!!
	return 0;
}
```

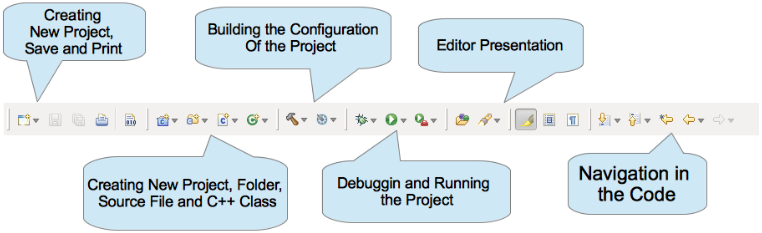
```
HelloWorld [C/C++ Application] HelloWorld
```

Perspective

- Perspective: a visual container for a set of window parts.
- eclipse uses perspectives to arrange windows parts for different development tasks
- switch Perspectives via the Window $\rightarrow$ Open Perspective
- change the layout and content within a Perspective

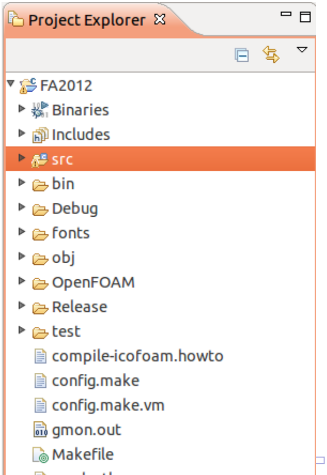
Toolbar

- contains actions which you typically perform


Creating New Project, Save and Print
Building the Configuration Of the Project
Editor Presentation
Creating New Project, Folder, Source File and C++ Class
Debuggin and Running the Project
Navigation in the Code

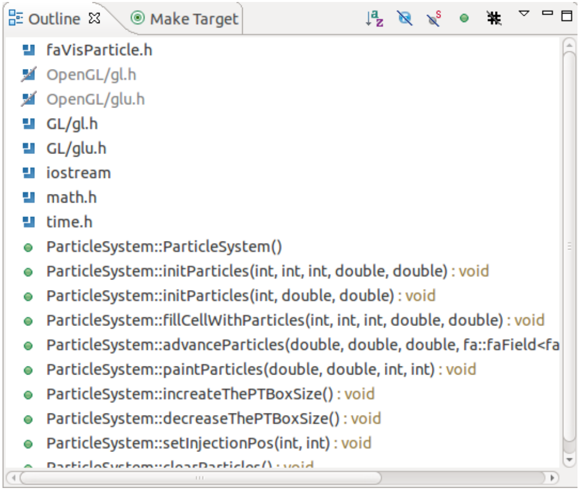
Project Explorer

- browse the structure of your projects and to open files via double-click
- change the structure of your project, e.g. you can rename files or move files or folder via drag and drop.



Outline View

- shows the structure of the currently selected source file



faVisParticle.h
OpenGL/gl.h
OpenGL/glu.h
GL/gl.h
GL/glu.h
iostream
math.h
time.h
ParticleSystem::ParticleSystem()
ParticleSystem::initParticles(int, int, int, double, double) : void
ParticleSystem::initParticles(int, double, double) : void
ParticleSystem::fillCellWithParticles(int, int, int, double, double) : void
ParticleSystem::advanceParticles(double, double, double, fa::faField<fa
ParticleSystem::paintParticles(double, double, int, int) : void
ParticleSystem::increaseThePTBoxSize() : void
ParticleSystem::decreaseThePTBoxSize() : void
ParticleSystem::setInjectionPos(int, int) : void
ParticleSystem::clearParticles() : void

Some Cool Features of Eclipse

- Code Auto-completion: Ctrl - Space
- Code Formatting: Ctrl - Shift - f
- Code Refactoring (Renaming): Shift - Alt - r
- Commenting a block of code: Ctrl - /
- and many others ...

Create Your First C++ Program in Eclipse

DEMO

What is Debugging?

- debugging is the process of locating and fixing bugs (errors)
- normally no way to see the source code of a program while the program is running!!!!
- by using debuggers we can look under the covers while the program is running

Why we need debuggers?

A debugger enables you:

- seeing the source code during the execution of a statement
- pause the execution at any place in the source code
- see and change the internal state of the program while the program is paused
- continue the execution
- Some of the basic debugging concepts: call stack, (conditional) breakpoint, stepping

Call Stack

- programs generally call functions
- one function can call another, or a function can call itself (recursion)
- chain of called functions as a stack of executing functions (call stack)
- currently running function is the topmost one on the stack
- For each function on the call stack, the system maintains a data area $\rightarrow$ stack frame
    - contains the function's parameters, local variables, return value and information needed to manage the call stack

Breakpoint

- a breakpoint is where the execution of the program should break off (stop)
- you can take over control of the program's execution after the program reached the breakpoint
- you can add and remove as many breakpoints as you like
- in addition to normal breakpoints: conitional breakpoints and watchpoints

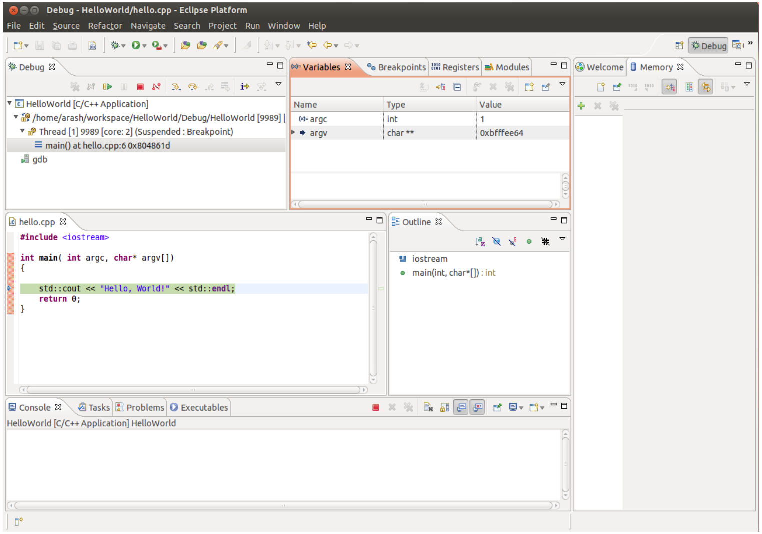
Conditional Breakpoints

- Normal Breakpoints: program execution should stop when a certain point in the code is reached
- Watchpoints: program execution should stop when a data value is changed
  - useful when a variable receives a wrong value and it's hard to track down where this happens just by looking at the code
- Conditional Breakpoints: program execution should stop at the breakpoint only if a certain condition is met (for example $X > 100$)
  - useful for example in a loop: allow a loop to run 100 times before breaking.

Stepping

- Stepping is the process of running one statement at a time
- Step Into: Executes the current statement if the current statement is a function, then the debugger steps into that function otherwise it stops at the next statement
- Step Over: Executes the current statement If the current statement is a function, then the debugger executes the whole function, and it stops at the next statement after the function call.
- Step Out: Steps out of the current function and up one level if the function is nested
- Continue: Continues execution to the end, or to the next breakpoint

Using Eclipse For Debugging


Debug - HelloWorld/hello.cpp - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
Debug
Variables
Breakpoints
Registers
Modules
Welcome
Memory
hello.cpp
Outline
Console
Tasks
Problems
Executables

```
HelloWorld [C/C++ Application]
  /home/arash/workspace/HelloWorld/Debug/HelloWorld [9989]
    Thread [1] 9989 [core: 2] (Suspended : Breakpoint)
      main() at hello.cpp:6 0x804861d
  gdb
```

Name	Type	Value
argc	int	1
argv	char **	0xbfffee64

```cpp
#include <iostream>

int main( int argc, char* argv[])
{

    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```

```
iostream
main(int, char*[]) : int
```

```
HelloWorld [C/C++ Application] HelloWorld
```