AOP
Project Submission Report – AspectC++ with Qt
Vishwanath Raman

This project attempted to test AspectC++ on a Qt program. This was to be achieved by simulating a chess game in a Qt environment, and then using AspectC++ to add/modify additional functionalities to the original program, without actually modifying the source.

About the code:

The original code consists of the following parts
`main.cpp`
`Box.cpp`
`Box.h`

The QApplication is defined in main.cpp. The Box class holds the details regarding each tile to be displayed on the chessboard such as whether the tile is light or dark, size, background color, events to perform on a mouse click or mouse hover, etc.

The program was still pretty modular and easy to understand.

Figure 1. Program output without aspects

As seen in Figure 1, the purple tile is where the mouse is hovering, it is not permanent. When a particular tile is clicked on, it turns green. However it stays green, which is undesired.
Also it can be noted that the images denoting the chess pieces are not fit into their respective tiles.

Adding Aspects :

We add an aspect file to the existing code and a configuration file

```
chessAspects.ah
acxx.prf
```

Now with Aspects, we can modify the game state dynamically. We add a mouseReleaseEvent() function to all instances of class Box, and we set the scaling of contents to true after every call to a constructor of class Box.

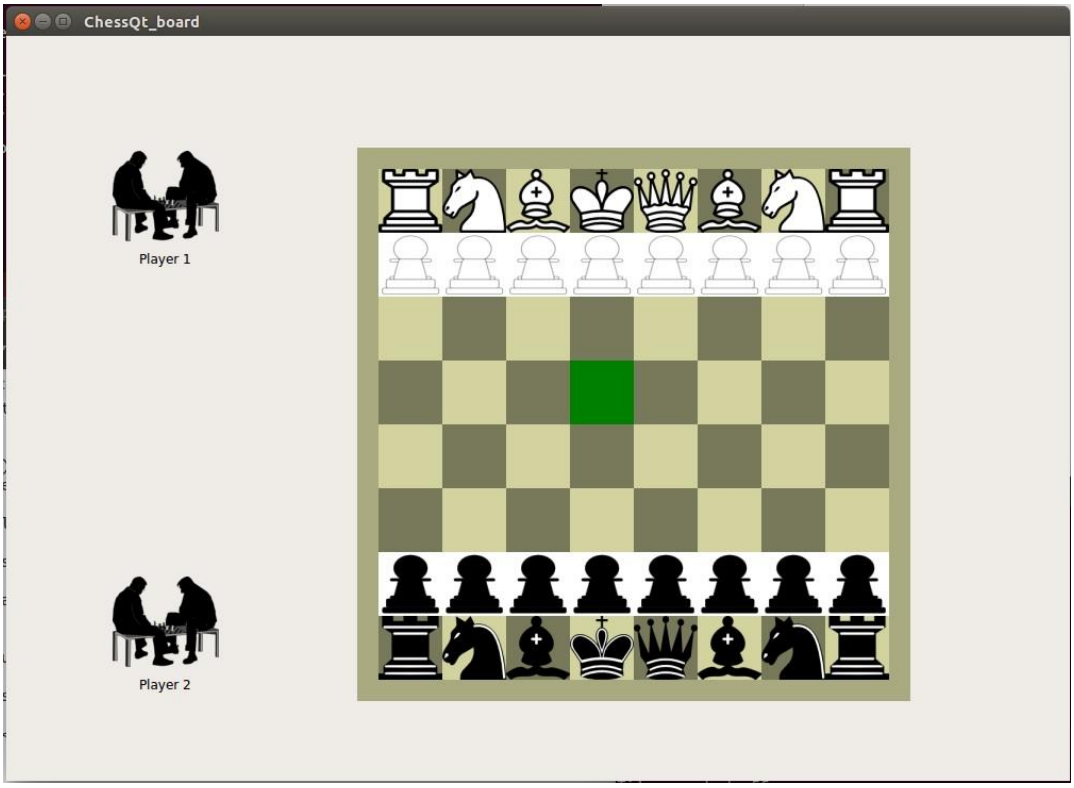

Figure 2. Program output with Aspects enabled

As seen from Figure 2. and Figure 3. on clicking the mouse the tile turns green, and as soon as it is released it goes back to its normal state. The tile continues to turn purple if the cursor is hovered over it

In addition the Aspect file also displays a confirmation `QWidget` which simply reads “Aspect started”

The code and the installation procedures are discussed further.

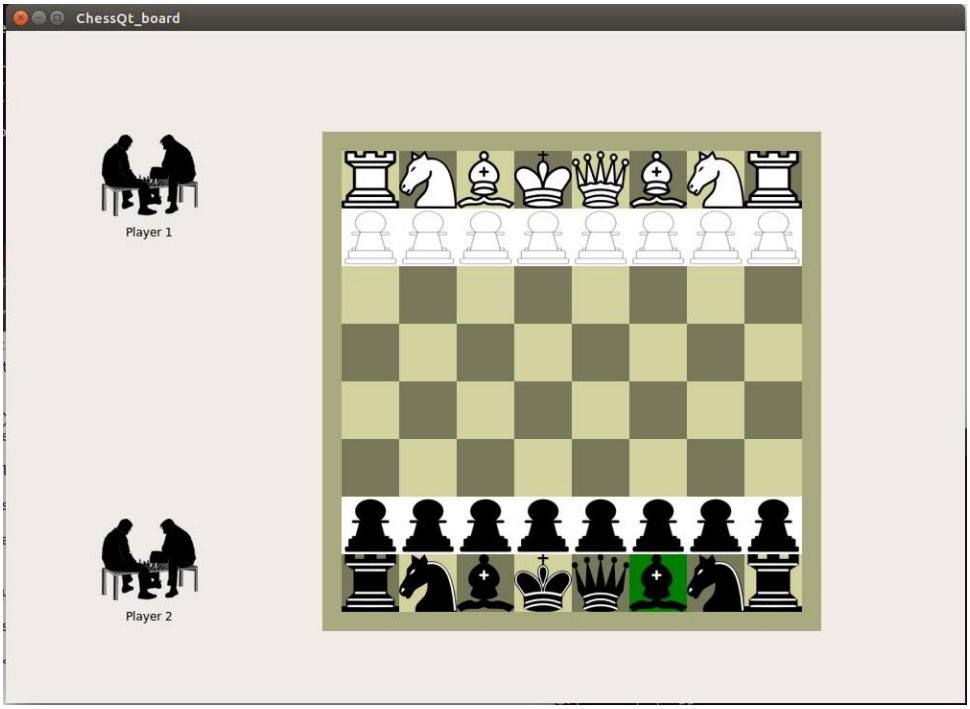

Figure 3. Program Output with Aspects

Figure 4. Confirmation of Aspect catching the execution of the Qt Application

```
pointcut boxClass() = "Box";
pointcut scaleTiles () = call("...::Box::Box(...)");

advice boxClass() : void mouseReleaseEvent(QMouseEvent *event)
{
    this->setBackground();
}
```

The above code is used to add a function to all classes returned by pointcut boxClass()

```
advice construction (boxClass ()) : after()
{
    tjp->that()->setScaledContents(true);
}
```

The above code sets scaling to true for current Box class object being created. We can also write it as follows without using the inbuilt construction() function

```
advice construction (scaleTiles ()) : after()
{
    tjp->that()->setScaledContents(true);
}
```

Installation Procedure:

Platform:
This project was done on Ubuntu 14 using VMware Player 6.0.4.
Similar results can be expected on a native Ubuntu installation as well.

Installing Qt:

```
username@ubuntu:~/Folder_name$ apt-get build-dep libqt4-core \
                                                  libqt4-dev
```

Get all the dependencies by executing the above line.

This project requires Qt 4.8 to be installed on your system.
Unpack the source files for Qt 4.8 and change inside that directory.
To prevent Qt from taking too much time install it from sources using the following options.

```
username@ubuntu:~/Folder_name$ ./configure  \
                                    -debug-and-release \
                                    -opensource \
                                    -developer-build \
                                    -no-webkit \
                                    -nomake examples \
                                    -nomake demos \
                                    -nomake docs \
                                    -fast
```

Then make so it builds the required files. This step took approximately 30 minutes.

```
username@ubuntu:~/Folder_name$ make
```

Then install those files, on receiving no major errors while building

```
username@ubuntu:~/Folder_name$ install
```

Setting up AspectC++:
Make sure g++ is installed on your system.

```
username@ubuntu:~/Folder_name$ sudo apt-get install g++
```

Download the source files for AspectC++ and unpack and install them

```
username@ubuntu:~/Folder_name$ sudo apt-get install aspectc++
```

Calls to *ac++*, *ag++*, *g++* and *c++* should call the respective compilers.

Executing a Qt project with aspects :

Download the `acxx.prf` from the AspectC++ page. The feature file `acxx.prf` will tell `qmake` to replace the C++ compiler by `ag++` and to take into account that all object files depend
on all aspect headers.

Once all the software requirements are setup the appropriate calls need to be made to execute the AspectC++ on the Qt project.

Firstly while in the project folder to compile the example, we must make sure that `qmake` will find the feature file `acxx.prf` via the environment variable `QMAKEFEATURES`. Execute the following,

```
username@ubuntu:~/Folder_name$ export QMAKEFEATURES=.
```

Call `qmake` to make the `.pro` project file

```
username@ubuntu:~/Folder_name$ qmake -project
```

Open the `.pro` file and add the following line before the # *Input files* declaration

```
CONFIG += acxx.prf
```

Call `qmake` to make the platform specific Makefile, containing special rules, for example to use `ag++` instead of `g++`.

```
username@ubuntu:~/Folder_name$ qmake project_name.pro
```

This should display a message, "Project MESSAGE: AspectC++ Feature selected ".

Call make to create the modified executable.

```
username@ubuntu:~/Folder_name$ make
```

Run the project executable

```
username@ubuntu:~/Folder_name$ project_name
```

Miscellaneous:

It is extremely important to make sure that the aspect files (.ah) are defined as headers as follows,

```
#ifndef __chessAspects_ah__
#define __chessAspects_ah__

aspect insert_name { … };

#endif
```

All headers that can be included in the Qt .cpp files can be used and included in the .ah files as well.

Limitations:

The following code was to access the `*mainWidget` pointer from the `makeBoard()` function in `main.cpp`

```
pointcut numberBoard () = call("% makeBoard(QWidget *)");
advice numberBoard() : around()
{
      QLabel *tempLabel[4] = { NULL };
      tempLabel[0] = new QLabel("Columns", (QWidget*)tjp->arg(0));
      tempLabel[0]->setGeometry(330 , 105, 552, 20);
      tempLabel[0]->setStyleSheet("QLabel {background-color
                                  :rgb(170,170,127); color:black; }");
}
```

`tpj->arg(i)` returns a pointer to the memory location of the *ith* argument parameter passed to the function.

However using that parent leads to an error due to thread conflict detailed as follows :

```
QObject::setParent: Cannot set parent, new parent is in different
thread
```

As seen AspectC++ currently does not work well with threads

Similarly, although, not demonstrated here, AspectC++ does not work well with the Qt concepts of slots and signals. AspectC++ cannot yet access the Qt extensions available through the `moc` compiler. Hence AspectC++ comes in after the `moc` compiler has run.

Conclusion

As seen from the output, AspectC++ can be used to add features to an existing Qt project. Thus a clean modular update can be made to the existing project.
There are still many limitations to carry out all desired tasks, such as, issues with multithreading, slots and signals, macros and templates. But even the current AspectC++ version is capable of accessing a lot of valuable program join points, which gives the programmer much control over what is going on in the code.
If the two main drawbacks, viz. threading and slots and signals are overcome, AspectC++ can deliver high control to modify and/or analyze a Qt code, without having to change the actual source code. This would especially make debugging a large program easier.

References:

- AspectC++ Language Reference, pure-systems GmbH, Matthias Urban and Olaf Spinczyk, Version 1.10, October 14, 2012.
- Using AspectC++ for Qt Application Development, Ute Spinczyk and Olaf Spinczyk, Version 1.0, 28 April 2011.
- Aspect-Oriented Programming with C++ and AspectC++, Daniel Lohmann and Olaf Spinczyk, AOSD 2007 Tutorial.

Software with source:

- AspectC++ v 1.2
  This project used: Linux/x86

  http://www.aspectc.org/Download.php

  or can also use:

  sudo apt-get install aspectc++)

- Qt 4.8.6
  This project used: `qt-everywhere-opensource-src-4.8.6-rc1.tar.gz`

  http://www.mirrorservice.org/sites/download.qt-project.org/development_releases/qt/4.8/4.8.6-rc1/

- Ubuntu 14.04.1
  This project used: Ubuntu 14.04.1 LTS 32-bit

  http://www.ubuntu.com/download/desktop

- VMware Player 6.0.4
  This project used: VMware Player for Windows 64-bit operating systems

  https://my.vmware.com/web/vmware/free#desktop_end_user_computing/vmware_player/7_0

- Microsoft Windows 7 Professional with Service Pack 1 64-bit (English)

  http://windows.microsoft.com/en-us/windows/downloads