ISSN:0975-9646

Mrunali Tandel et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 7 (3) , 2016, 1242-1245

Literature Review : Scripting in Qt for Scripting Engine development

Mrunali Tandel[#1], Prof. Uday Joshi[#2], Amit Golhani[*3]

[#]*Department of Computer Engineering, University of Mumbai*
K J Somaiya College of Engineering, Mumbai, India

[*]*Larsen & Toubro Automation,*
Navi Mumbai, India

***Abstract*— For an application development, the organization selects programming language according to the various requirements of the application. C/C++ is considered to be one of the strongest programming languages with advantages such as excellent performance and control over memory-layout to list a few. Though developing an entire application using C/C++ provides high performance, it becomes tedious for extending it. Only a programmer proficient in those languages can extend them and this dependency can be time consuming. Scripting language can be considered as one of the option. If the entire application is developed in scripting language, it can be extended easily; enabling rapid application development, but the performance of the application will get degraded. In the competitive market of software, performance of the application, frequent updating and customisation plays an important role. The concept of scripting engine is introduced to ease the above tasks. From the number of frameworks available for C/C++ programming, Qt framework is considered for this research, as it is widely used and third-party scripting extensions are available in the market for Qt. With the Qt framework, the developer can integrate the performance critical application developed in C/C++ with the script to manage the user-interaction and non-performance critical events.**

***Keywords*— Scripting Engine, cross-platform, Qt Framework, library.**

I. INTRODUCTION

With the advent of many programming languages, the programmer got the wide variety of choices for developing their applications. In the past, the entire software was written in high performance programming language such as C/C++. No matter, it provides an excellent performance as the code is precompiled; it becomes very difficult to program the application. High performance programming languages are very difficult to master in. On the other hand, scripting languages do not provide a high performance as, the conversion to machine code is on-fly, it is very easy to write an application in scripting language compared to compiler-based language. Also, programming in scripting language enables rapid development of an application. Thus, either the application developed will have low performance, or it will be very difficult to program or extend. There are many frameworks available in market, with Qt being used widely in industrial software development. Qt is a cross-platform application development framework, developed by Digia and Qt Projects (currently maintained by the Qt company), used for developing applications that can be run on various software and hardware platforms. It is available with both commercial and open source licence. The main development language for Qt is C++ and it supports GCC C++ compiler and MSVC. [5]There are many third-party scripting extensions for Qt which enables extending the Qt application using scripts. The language includes but not limited to JavaScript, Lua, Python and Ruby.

In order to balance the high performance provided languages such as C++/C and ease of programming provided by scripting language, the concept of scripting engine is proposed. The scripting engine is a tool/library, which is capable of loading, compiling and running script code. It can be used in an application to handle all the non-performance critical and user-interface events. The scripting engine will enable to develop or extend an existing application easily using scripts, without hampering the performance of overall application. The further chapters explains the proposed concept of scripting engine as well as the comparative study of various scripting extensions available in Qt, which enable integrated development using high performance programming language(C++) and scripts.

II. LITERATURE REVIEW

The Qt application can be extended by scripting language using scripting modules provided by The Qt Company or the scripting extensions provided by third-party. The scripting languages for which, the scripting extension is available for Qt are JavaScript, Lua, Python and Ruby. The extensions for JavaScript are QtScript [7] and QJSEngine [8], both provided by Qt framework. The extensions for Lua in Qt are QtLua by Savannah projects [9] and Lqt [10]. The extensions for Python in Qt are PyQt by Riverbank Computing [11], Pyside by Nokia [12], Pyotherside [13] and PythonQt [14]. The extensions for Ruby in Qt are QtRuby by Korundum [15] and Ruby-QML by seanchas116 [16]. Thus, with any of the extensions available, scripting engine can be developed, where the core part of the engine is written in C++ and non-performance critical part using scripts or the extension of the application in scripting language which can interact with the user-interface.

The comparative study based on the requirement of product of each of the extensions available is as follows:

A. Ease of using

The scripting language to be used in scripting engine should be easy to use, as the purpose of its introduction in application is to extend the core part of library using script for rapid development. If an easy to learn and readable scripting language is selected, then the application can be extended easily by anyone, including a non-programmer or the person with least expertise in programming. Considering JavaScript, Lua, Python and Ruby, all are easy to use compared to high performance language like C/C++. Considering the readability of language, Lua and Python are readable. Ruby is bit confusing, as in Ruby there are more than one way of doing things in Ruby. eg. Conditional statements, aliasing methods and opposite methods [29][30].

B. Support provided for extensions

As some of the extensions provided for scripting are third party extensions, support provided by this third party organization plans a vital role in development of this project. Support includes the forums and mailing list which addresses the issues related to these extensions. QtScript and QJSEngine are the part of Qt framework, thus all forums related to Qt addresses the issues. The forums/mailing list for QtScript and QJSEngine are [31], [32], [33], [34]. For Python, the extension PyQt has maximum support such as [32], [35], [36], [37]. PySide has a mailing list which addresses PySide related questions [38]. QtLua, lqt and QtRuby have mailing list each [39], [40] and [41] respectively.

C. Licensing

Licensing plays an important role, when the aim is to develop a proprietary application. QtScript and QJSEngine, both are licensed under GNU Library General Public License (LGPL) [7] [8]. PyQt is available with both commercial license and GNU General Public License (GPL) with no functional difference between them. Informal support is provided for commercial license holder [17]. PySide and PythonQt licensed under LGPL version 2.1 license [18][19]. Pyotherside is licensed under ISC/BSD license [20]. QtLua is licensed under LGPL version 3 license and Lqt is licensed under MIT(X11) license [21][10]. QtRuby is licensed under LGPL version 2.1 license [22]. Ruby-QML is licensed under MIT license [23].

D. Roadmap to support future versions

The extension should provide support to the recent versions of Qt and should support the maximum features of Qt. Considering QtScript and QJSEngine, both are the part of Qt framework. QtScript, which is based on JavaScriptCore engine, has been deprecated since Qt 4.7, since; it failed to give high performance as JavaScript V8 engine. Thus, QJSEngine was introduced, which is based on JavaScript V8 engine. It is much faster and present in Qt QML module, has better integration with C++ (and QML) and is more compliant with ECMA standard [42][43]. But, it is still in developing phase and lacks features which are present in QtScript such as instantiating QObject from JavaScript, exposing individual native function to JavaScript and debugging API. [24][44].

PyQt supports Qt4 and Qt5.4.x. The support for Qt 5.5 is still not available [11]. PySide supports only Qt4, there is no official support for Qt5 [25]. Pyotherside currently supports Qt5 and is actively maintained one [13]. PythonQt currently supports Qt4 and Qt5. It is actively maintained are developers are working to provide support for Qt 5.5 [14]. The QtLua supports Qt 5.1, 5.2, 5.3, 5.4.x. The support for Qt 5.5 is not official [9]. Lqt project is no longer maintained by the author and the existing Lqt contains many bugs and limited features. Thus, Lqt's support for Qt is stranded to Qt 4.7.

QtRuby is no longer maintained by its developers and the existence QtRuby library supports only Qt 4.x [26]. Ruby-QML provides binding between Ruby and QML. It currently provides support up to Qt 5.4 and is actively maintained [23].

E. Platform Supported

The product should support multiple platforms, as it can be used by anyone on the desired platforms. The table below shows the platform supported by each of the extensions [27][11][18]

TABLE I
PLATFORM SUPPORTED BY THE BINDINGS

	QtScript	QJSEngine	PyQt	PySide	Pyotherside	PythonQt	QtLua	Lqt	QtRuby	Ruby-QML
Windows(Xp, 7, 8)	✔	✔	✔	✔	✔	✔	✔	✔	✔	
Linux x/x11	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Mac OS	✔	✔	✔	✔		✔			✔	✔
Android	✔		✔	✔	✔					
iOS	✔									
Maemo				✔						

F. Speed of execution

Compared to the high performance language like C/C++, the scripting language plays low. Table I below shows the time taken in milliseconds by each language for execution [6]:

TABLE III
COMPARISON WITH RESPECT TO SPEED

Line Size (Kilo bytes)	Lua	Python3	Python	Ruby	JavaScript (V8)
256	49000	17000	7000	7000	3000
512	203000	81000	32000	29000	21000
768	480000	201000	78000	75000	51000
1024	886000	373000	144000	141000	91000
1280	1423000	598000	232000	225000	144000
1536	2090000	877000	342000	328000	208000
1792	2886000	1211000	476000	452000	283000
2048	3856000	1598000	634000	597000	370000
2304	4963000	2039000	815000	758000	469000
2560	6198000	2533000	1019000	941000	578000
2816	7568000	3070000	1248000	1143000	700000
3072	9084000	3659000	1497000	1366000	834000
3328	10759000	4300000	1771000	1607000	979000
3584	12594000	4992000	2064000	1869000	1136000
3840	14564000	5729000	2381000	2150000	1304000
4096	16674000	6534000	2720000	2455000	1484000

III. CONCEPT OF SCRIPTING ENGINE

The concept of scripting engine revolves around scripting languages and high performance programming language.[4] High performance programming languages like C/C++ are performance efficient whereas scripting languages are easy to use, can achieve maximum functionality with minimum lines of code and provide the programmer the advantage of automatic memory management and typecasting. But scripting languages are weak in terms of performance, compared to programming languages such as C/C++. In C/C++, the code is compiled to native code and there is more control over memory layout, which is not possible using scripting language. And thus, scripting language consumes more CPU cycles which may compromise the overall performance of an application. An application developed using C++ programming language and scripting language is as shown below in Figure 1: [4]

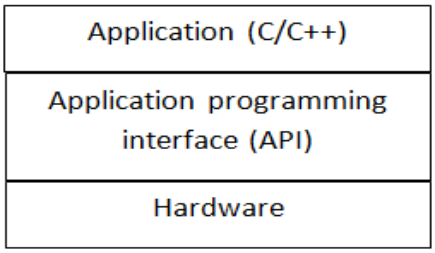

Figure 1 Application in C++

Application written in C/C++ is first compiled which can be run any number of time, on any platform, thus providing high performance. Applications written in scripting language are loaded and interpreted on-fly at runtime by scripting engine and thus are not performance efficient. The concept of scripting engine is to develop the part of an application which is performance critical, in high performance language and to manage the non-performance critical code, User-interface components and interaction in scripting language. The performance critical library can be exposed to script-based application, thus, it provides an advantage of accessing the code with minimum scripts and the core library can be modified independently, without affecting the interaction and GUI module. Thus, the resulting application will give almost equal performance compared to the application developed in C/C++ and enables rapid development. The resulting application will be we as shown below in figure 3:

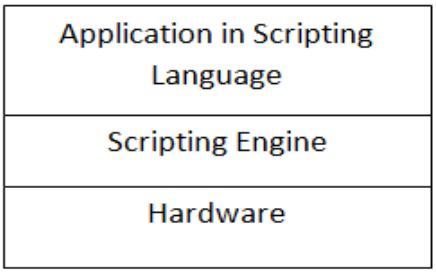

Figure 2 Application in scripting language

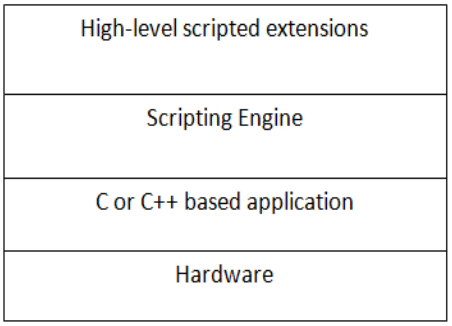

Figure 3 Application using Scripting extension of C++ library (Scripting Engine)

Figure 4 demonstrates an example of the working of Scripting Engine concept. In figure 4, the provision is given by scripting engine to a non-programmer to extend the core library. The extension can be to develop the interactive user-interface. Thus, the developer can use the easy to use script to extend the application without investing efforts, at the same time; the performance requirement of the application is taken care of at the core library level.

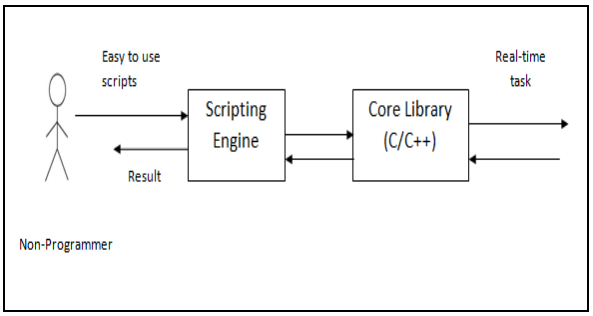

Figure 4 Working of the concept

A. *Advantages of the resulting approach*

- Enables easy extension of the application using script. Thus, the core application can be extended using the scripting language easily without any significant performance difference compared to an application developed purely in C/C++, handling performance critical task at C/C++ library level.
- No type-casting, memory management and complex syntax for extending an application.
- Automation of repetitive task based on the event generation at user-interface level or at library level.
- The application can be easily customised using the scripting engine, if the user interface and the interaction are managed by scripts.
- Rapid prototyping of an application.

IV. APPLICATION OF THE RESULTING APPROACH

The resulting scripting engine can be used in various approaches, such as:

- The approach can be used for rapid development of embedded systems, where the engineer can use the resulting scripting engine for generating panel user-interface with ease. The development of user-interface will be independent of the changes or modification in the core libraries.

- Development of games, where the common actions based on events are defined in the core libraries and the UI can be developed rapidly using those libraries through script, thus, automating the action of tiles in the game.[1]
- Development of User Interface of Direct TV, where the scripts can be defined to load the functions in library based on the event from user. [2]
- Development of applications, where the development time is critical and the user interface developer has to experiment with the designs. [3]

V. CONCLUSIONS

The concept of scripting engine using Qt framework aims to make an application development or extension easy, balancing the performance requirement desired for the resulting application. Using Qt extensions for scripting, it enables to develop rich user-interface and event-handling of the interaction with user-interface using scripts. Considering the requirement of the application, different extension can be. As one of the objective of Scripting Engine is the ease of extending an existing application by a non-programmer, scripting engine like Lua and Python can be selected which are easily readable and unambiguous to use. If the aim is to develop a commercial application, PyQt provides commercial licence and informal support for the commercial licence holder. If the speed of the application is priority, QJSEngine can be used for scripting with the fastest JavaScript version 8 engine, provided few of the functions can be compromised. Pyotherside or Ruby-QML can be considered, if the developer does not wish to purchase the commercial license to develop proprietary software, as they are licensed under ISC/BSD and MIT respectively. The online support provided by exensions such as QtScript, QJSEngine and PyQt will provide the developer an added bene_t. The Scripting Engine developed using extensions such as QtScript, QJSEngine, PyQt and PySide can be extended on many platforms including the non-conventional ones, whereas, the other extensions provide support for only conventional ones. Thus, using a strong framework like Qt, an appropriate extension can be selected to develop a Scripting Engine for rapid application development, meeting the high performance requirement. The resulting approach will enable any non-programmer to extend an application with ease and the product can be marketed within the stipulated time.

ACKNOWLEDGMENT

The authors are thankful to Dr. Shubha Pandit, the Principal of K J Somaiya College of Engineering, Prof. Bharthi Narayan, Head of Computer Department, Mr. Rahul Rane and Mr. Shishir Gupta from Larsen & Toubro Automation, Navi Mumbai, for providing the opportunity and necessary facilities for the preparation of the paper.

REFERENCES

[1] M. McLaughlin and M. Katchabaw, A Reusable Scripting Engine for Automating Cinematics and Cut-Scenes in Video Games, Proceedings of CGSA 2006 Symposium.
[2] Milivoj Bozic, Dusan Zivkov, Istvan Pap and Goran Miljkovic, Scriptable Graphical user Interface Engine for embedded platform, 21st Telecommunication Forum TELFOR 2013.
[3] Hans Christian Woithe and Ulrich Kremer, A Light Weight Scripting for the Slocum Glider, IEEE Oceans 2010 Conference, Sydney, Australia, May 2010.
[4] Scripting|Mono, http://www.mono-project.com/docs/advanced/embedding/scripting/
[5] Qt(Software), https://en.wikipedia.org/wiki/Qt_(software)
[6] Perl, Python, Ruby, PHP, C, C++, Lua, tcl, JavaScript and Java comparison, http://raid6.com.au/~onlyjob/posts/arena/
[7] Qt Documentation, QtScript, http://doc.qt.io/qt-5/qtscript-index.html
[8] Qt Documentation, QJSEngine Class, http://doc.qt.io/qt-5/qjsengine.html
[9] Alexandre Becoulet.,2013, QtLua home project http://www.nongnu.org/libqtlua/
[10] Mauro Lazzi, Peter Kummel, Michal Kottman, lqt https://github.com/mkottman/lqt
[11] Riverbank Computing Limited.,2015, What is PyQt? https://riverbankcomputing.com/software/pyqt/intro
[12] Category:LanguageBindings::PySide https://wiki.qt.io/PySide
[13] Thomas Perl.,2014,Pyotherside:Asynchronous Python3 bingings for Qt5 https://thp.io/2011/pyotherside/
[14] PythonQt.,2015, http://pythonqt.sourceforge.net/
[15] KDE TechBase, https://techbase.kde.org/Development/Languages/Ruby
[16] Ruby-qml A QML/Qt Quick bindings for Ruby, http://seanchas116.github.io/ruby-qml/
[17] Riverbank Computing Limited, PyQt Commercial Version, https://riverbankcomputing.com/commercial/pyqt
[18] About PySide, https://wiki.qt.io/About_PySide
[19] PythonQt License, http://pythonqt.sourceforge.net/License.html
[20] Thomas Perl, Pyotherside/LICENSE, https://github.com/thp/pyotherside/blob/master/LICENSE
[21] QtLua Script Engine for Qt-Summary, http://savannah.nongnu.org/projects/libqtlua
[22] QtRuby, https://en.wikipedia.org/wiki/QtRuby
[23] List of language bindings for Qt5, https://en.wikipedia.org/wiki/List_of_language_bindings_for_Qt_5
[24] New features in Qt5.5, http://wiki.qt.io/New-Features-in-Qt-5.5
[25] PySide2- PySide Qt5 Support Underway, http://it.toolbox.com/blogs/enlightenment/pyside2-pyside-qt5-support-underway-68285
[26] QTbindings, https://github.com/ryanmelt/qtbindings/
[27] Qt Documentation, Community Supported Platforms, http://doc.qt.io/qt-5/supported-platforms.html
[28] Openbossa, PySide, https://en.wikipedia.org/wiki/PySide
[29] One way to do it? (Ruby vs Python), http://www.senktec.com/2013/09/one-way-to-do-it/
[30] Olugbenga Oluwagbemi, Adewole Adewumi, Folakemi Majekodunmi, Sanjay Misra, Luis Fernandez-Sanz, An Analysis of scripting languages for Research in Applied Computing.
[31] Qt Forum, https://forum.qt.io
[32] Qt Centre, www.qtcentre.org
[33] KDE Community, https://forum.kde.org
[34] QtForum.org, International Qt programming forum, www.qtforum.org
[35] Developpez.com, pyqt.developpez.com
[36] mail.python.org Mailing Lists, https://mail.python.org
[37] PyQt- Python bindings for Qt, https://riverbankcomputing.com/mailman/listinfo/pyqt
[38] PySide, http://lists.qt-project.org/mailman/listinfo/pyside
[39] Libqtlua-list, https://lists.nongnu.org/mailman/listinfo/libqtlua-list
[40] Mailing List, http://lists.qt-project.org/mailman/create
[41] Kde-bindings, KDE bindings for other programming languages, https://mail.kde.org/mailman/listinfo/kde-bindings
[42] Standard ECMA-262,ECMA International, 6th Edition.
[43] Knoll Lars, 2013, [Development] QML and JavaScripts Extensions, http://lists.qt-project.org/pipermail/development/2013-November/014185.html
[44] Ilya Diallo, 2015, QtScript to QJSEngine migration, http://lists.qt-project.org/pipermail/interest/2015-June/017446.html