Domain analysis for a user interface toolkit

Guus Bonnema
member of ABI team 33

21/10/2014

Abstract

This document reflects a domain analysis for user interface tools. It looks for available tools that are free to use, will remain available and can do the job required for the design tool of an xMAS network. It describes the user interface tool kits found, the search approach, the criteria derived from requirements and the team preferences, all used in selecting the UI toolkits. The result is a motivated recommendation for the project of team 33 to choose a user interface toolkit.

The document presents the results from abstract to detailed to facilitate the reader in understanding the document. This presents conclusions first, progressively providing more details. Reading up to and including appendix A leads to full, high level understanding.

Contents

1 Problem domain

1.1 Description

The customer's request is to build a graphical chip editor for Network on chip designs that uses existing C++ programs to interface with and that runs on all defined platforms: Linux, ms Windows and Mac. This implies part of the solution is a user interface tool that works toward graphical representation of chip networks. The customer has specified some qualitative requirements like maintainability and install ability. This document is the result of gathering information and analysing existing user interface tools that we could use for this purpose. The goal is to formulate a recommendation for the team with respect to the best user interface tool kit to use.

1.2 Requirements for the approach

The number of available user interface tool kits is large. Moreover the number of features each tool kit has, is large. This warrants careful consideration of tool kits where we start from the project's requirements and the team's preference. The primary goal is that the UI tool suffices: it does not have to be perfect.

2 Search and Selection

2.1 Search approach motivation

Normally one searches literature for information on science subjects. For tool kits this may not be the most efficient or effective way of finding current tool kits. The problem with literature is that it is constantly behind on the current state of affairs. Even though information on the internet is generally less reliable than literature is, in this case it forms a better basis to find a more complete list of tool kits. Literature does supplement what we find on the internet.

This analysis is able to determine a list of user interface tool kits starting from en.widipedia and fanning out through the links wikipedia provides and a search engine. The dates on the sites provide information on how actual the data is. Cross referencing sites gives confidence in the correctness of the information. The next section lays out the criteria we used to select tool kits.

2.2 Selection criteria

Table 3 on page 13 contains a list of the main requirements for the design tool. These requirements generate criteria that we can use to find the most appropriate tool kits available. Table 4 on page 14 shows a list of the team's preferences. Table 6 on page 16 shows the implications of the requirements resulting in specific user interface requirements. The next section shows what steps I followed and what the results were.

2.3 Phase 1: Primary deselection

The first phase deselects tool kits that do not suffice one of the important requirements. Table 7 on page 17 shows the tool kits that were deselected and why.

2.4 Phase 2: Secondary deselection

The remaining tool kits were examined in more detail. Each one of these tool kits finds a summarized description in appendix G. This results in rating the tool kits according to the main characteristics required in Table 2 on page 10.

Analysis of the results leads to accept the first four for definite comparison in the last phase. These tool kits have all necessary characteristics required. The next three tool kits represent 2D tool kits that could be used in conjunction with the first four.

2.5 Phase 3: Final selection

This phase concerns the tool kits : **GTK+**, **QT**, **WXWIDGETS** and **FLTK**. See appendix G for a summary of each tool kit and appendix A for a more detailed description. Each of these tool kits satisfies all requirements either autonomously or in combination with other tool kits. In terms of requirement based selection, the search is finished.

The next section documents the considertions and the resulting recommendation.

3 Result and recommendation

3.1 Considerations and Motivation

Considering the goals of the project, and the competence wishes, the team has different options as a result of different viewpoints. One option would be to go with a toolkit that is "`mean and lean`" and add libraries only where necessary. The other option would be to go with a tool kit that contains "all++" and thus is comfortable allbeit a little bloated. Whatever the choice, it will affect the end product.

In my recommendation I go with the tool kit **FLTK** (pronounce *fulltick*) . The tool kit is targetted at workstations and embedded hardware. The focus is on being `mean and lean` and fast and small, as advertised on the **FLTK** web site and user forum.

The recommendation is based on this information (also see appendix A) and the personal assessment of the project in next paragraph. Of course the final decision is up to the team.

The project emphasizes a Network on Chip design, with most user interface effort going into the 2D drawing interface and the general user interface. This makes the focus of **FLTK** on user interface with respect to gui a good choice. Should we need more emphasis on looks or more complex drawing than **FLTK** can provide, then we could substitute **OpenGL** or **Cairo**[1].

The argument is solid, but with a shift of viewpoints a different choice could be valid. The team should think about what is important in this project and decide on that basis. See appendix A for a summary of main characteristics of the selected tool kits.

3.2 Recommendation

I recommend using **FLTK**.

[1] Integration of **OpenGL** or **Cairo** into **FLTK** should be trivial

A Final summary of applicable tool kits

In order to support the final decision, in the following sections I summarize the main characteristics.

A.1 QT

QT is a complete development environment based on an extension of C++. It supports many features among which are 2D, concurrency and memory management. It also has many dependencies as shown in [12].

Main characteristic of QT is that it uses a pre-compiler called `moc` extending standard C++ with facilities needed for signal and slots. The owner is a company DIGIA that licenses the QT toolkit both as free software and with a commercial license. The tool kit QT supports the observer pattern with signals and slots.

Why to choose QT? Because it is cross platform. It is well known and used in many applications. Appealing is also its simple signal and slot feature and moreover, support for it's user interface building is widely appreciated. Finally, QT has an active community and very clear documentation both online and in books.

Why not to choose QT? Because QT extends C++, uses macro's heavily and thus needs an extra pre-processor to build the application. Also, it provides a lot more than we need for our application.

A.2 GTKMM

GTKMM combines a graphical user interface with many non gui features like signals and memory management. For 2D it builds on cairo or opengl. The library for gtkmm has some dependencies on other libraries that have their own dependencies as shown in [9].

Main characteristic of GTKMM is that it is primarily linux oriented and is the main development environment for the GNOME window manager. It uses standard C++ and additionally collections and other features from GLIB.

The tool kit GTKMM supports the observer pattern using signals.

Why choose GTKMM? Because it is cross platform. It is well known and used in many applications. Because GTKMM is C++ even if it is a wrapper: it does not use macro's heavily. Because it supports cairo natively. Finally GTKMM has an active community and clear documentation both online and in books[2].

Why not to choose GTKMM? Because GTKMM is linux oriented (developments starts from the GNOME window manager). The primary aim is unix, not windows or mac. Also, it provides a lot more than we need for our application.

[2] QT has more literature than GTK+

A.3 WXWIDGET

WXWIDGET is a complete cross platform development environment. It is built as multiple libraries by default. The features cover support for openGL and concurrent execution. The **WXWIDGET** depends on different libraries for each platform.

Main characteristic of **WXWIDGET** is that it has a separate underlying interface to each platform. The result is an application with the look and feel of the platform. The disadvantage is that applications run differently on different platforms and may show platform dependent errors. Also, the binaries of the application are usually bigger than for the other tool kits.

The tool kit **WXWIDGET** does not seem to support the observer pattern directly, but some examples of how to define one are available like [4][3].

Why choose WXWIDGET? Because it is cross platform. It is well known and used in many applications.

Why not to choose WXWIDGET? It produces overweight applications that could make the application feel bloated. Also, it provides more than we need for our application.

A.4 FLTK

FLTK is a graphical user interface with 2D drawing and nothing more. For concurrency we look to another cross platform tool kit, like **BOOST**. The tool kit **FLTK** depends on the window manager X11 or Wayland or Win32.

Main characteristic of **FLTK** is that it has no additional facilities. The motto "Do one thing and do it right" is one that unix programmers often adhere to. The binaries for programs using this tool kit are small in comparison to other tool kits.

The tool kit **FLTK** does not support an observer pattern, but one can define one as stated on the forum at urlhttp://groups.google.com/forum/#!forum/fltkgeneral.

Why choose FLTK? Because it is focussed on graphical user interface. It does one thing and it does it well. It is easy to learn. It is easy to combine with other libraries like **OpenGL**, **Cairo** or **BOOST**. The community is active and willing to support users of their library[4].

Why not to choose FLTK? Because it does not offer more than graphical user interface. Also, the code avoids many C++ constructs like namespace and templates.

[3]The surrounding text is in Russion, but the example is clear.

[4]I found their support to be both fast and objective (in an **FLTK** biased way of course).

B Phase 3

B.1 Phase 3 selection results

Table 6 in appendix E shows the tool kit requirements. The final comparison in Table 1 takes the remaining tool kit requirements into account.

id	req	fitness	gtkmm	qt	wx	fltk
7	Ease of use	reputation	+	+	+	+
8	Ease of learning	rep & tutorials	+	+	+	+
9	Documentation	website & rep	+	+	+	+
10	Observer pattern	signal processing	+	+	+	+[a]
14	Concurrency	support for concurrency	+	+	+	+[a]
15	Concurrent observer	is it possible?	+	+	+	+[a]

Table 1: Final comparison of selected tool kits

[a] using standard C++ or another library like boost

Ease of use and ease of learning We can only measure the requirements for ease of use and learning subjectively. This is an experience result and changes in time: any tool kit becomes easy to use and learn after enough experience using it.

So the only measure that we could use is reputation, plus tutorials for ease of learning. Although subjective, the cumulative criticism and opinions indicate real quality. Neither of the systems have bad rep on any of the subjectively measurable qualities (7, 8 and 9). Also, the main website for the tool kit easily revealed the available documentation. The assumption that any one of these tool kits lacks in ease of use, learning or in documentation can be rejected on the basis of experience and reputation.

Observer pattern. The observer pattern is important in two ways. First, all signals from the user interface should relay flawlessly to modules, even if they execute concurrently. Secondly, any change of status or content in the modules, even concurrent modules should relay flawlessly to the user interface thread. Measuring this is a question of checking the docs and asking around.

B.2 Phase 3 selection analysis

Table 1 shows that tool kit requirements 7, 8 and 9 are satisfactory for all. The observer pattern and concurrent process communication is not as simple. In summary, **QT**, **GTKMM** and **WXWDIGET** all suffice without modification. For **FLTK** we need to use one of the available cross platform libraries for concurrency. It turns out to be easy to define ones own observer pattern, or use a library for this like **BOOST**.

As all packages can fulfil the requirements the question is, what distinguishes these tool kits and on what basis should we choose? The short answer is: they all suffice. So the choice is no longer one of user interface requirements, but one of preference.

C Phase 2 Main Selection

C.1 Phase 2 selection results

Table 2 shows the remaining tool kits. The column COMMUNITY indicates the expected support for the tool kit. The tool kits qualified as **PARTIAL** leave out either the GUI widgets or the 2D drawing capability. They could still be viable in combination with one of the other tool kits. Among the tool kits qualified as **MAYBE** are some unconventional, but powerful frameworks. Still, this analysis only considers the categories **YAY** or **PARTIAL**.

nr	Toolkit Name	C++	2D	GUI	Community	Yay or Nay Yay	Part Maybe	Nay
1	GTK+	1	1	1	Active	Yay		
2	Qt	1	1	1	Active	Yay		
3	WxWidgets	1	1	1	Active	Yay		
4	FLTK	1	1	1	Active	Yay		
5	Cairo	1	1	0	Large		Partial	
6	OpenGL	1	1	0	Large		Partial	
7	SDL	1	1	0			Partial	
8	JUCE	1	1	1	Small		Maybe	
9	CEGUI	1	1	1	Active		Maybe	
10	Gled	1	1	1	Small		Maybe	
11	Mozilla A.F.	1	1	1	Large		Web-oriented	
12	Tk	0.5	1	1	Large			Nay
13	fpGUI	0						Nay
14	GDK	1	1	0	Large			Nay

Table 2: Selected tools against must have product/project requirements

C.2 Phase 2 selection analysis

Tool kits included The tool kits 1 through 4 satisfy the requirements.

The tool kits 8 and 9 satisfy all but one requirement (the GUI) but could still function if they add sufficient function to a GUI oriented toolkit.

Tool kits excluded Tool kit 10 (SDL) is not clear on the supporting community, but does seem to have a following. The comparison from **WXWIDGETS** ([42]) mentions SDL as a viable addition to **WXWIDGETS**, but oriented towards gaming. The tool kits 12 through 13 do not satisfy the requirements fully, and tool kit 14 (GDK) is low level and incorporated in GTK+. The Mozilla Application Framework (M.A.F.) is oriented towards the web. The deciding feature is that MAF contains a sub selection of the regular GUI widgets geared towards html and css.

For **FPGUI**, **SDL** and **GLED** the community activity is not clear from the home website, so the community is left out in table. The system **JUCE** is a one man project and aimed at multimedia and gaming. **CEGUI** is a small community of developers specifically aimed at gaming, not graphical user interface. **GLED** is a framework that stems from scientific work and features the easy distribution of nodes across threads, processes and machines. See [10] for a discussion of the scientific goals of the **GREED** project.

Next phase will contain toolkits $1, 2, 3$ and 4: GTK+, Qt, WxWidgets, FLTK with possible additions of OpenGL or Cairo.

D Requirements for design tool

Table 3 on page 13 summarizes the high level requirements for the chip design tool. These are the main base for selecting the tool kits. Each requirement may fully impact the UI tool, partially or not at all. Table 4 on page 14 shows the team members' requirements in terms of competence goals. Table 6 on page 16 summarizes the impact of the project's requirements and the team's competence wishes on the user interface toolkit.

D.1 Justification of free software requirement

The requirement for free software was not in the originally formulated high level requirements, so some justification follows. This requirement ($M0$) was added to ensure the free availability of the UI tool kit. The source of this requirement is a remark from the Open University instructions "Hulpmiddelen en bronnen" ("tools and sources") with the following text:

> "Wij gaan ervan uit dat de producten van het team uiteindelijk veelal als open source beschikbaar gesteld worden, dus iedereen mag in principe de producten inzien. In de overeenkomst met de opdrachtgever dient op het punt van de rechten duidelijkheid gegeven te worden."
>
> "We assume open source availability of the team's products so that the products are publicly available. The contract with the customer should be clear on this point."

Demanding free software follows from this specification and the fact that non-free software could potentially lead to non-availability of the software due to license fees or the author retracting the software. We specify free software as a must-have requirement to be "clear on this point".

D.2 The project and product requirements

	requirement class	
	req name	**req description**
M	*Must have requirements*	
M0	Free software	The application is free software as defined by the FSF.
M1	Cross platform	The application must run equally well on the defined platforms with respect to all relevant features. Linux, Mac and MS Windows are the defined platforms.
M2	C++ integration	The application must integrate seamlessly with the existing C++ code for analysis and verification tools.
M3	2D drawing	The application must be able to draw items on a canvas and treat graphical objects with composite behaviour or properties as a primitive object.
M4	GUI widgets	The application must be able to create GUI widgets on all defined platforms.
M5	xmas primitives	The application must be able to draw the 8 xmas primitives.
M6	xmas macros	The application must be able to define and draw a macro of xmas primitives.
S	*Should have requirements*	
S1	expandability	the application should be expandable in terms of analysis and verification modules, xmas-primitives and network reporting.
S2	installability	The application should be easy to install.
S3	maintainability	The application should be as easily maintainable as possible.
S4	plugability	The application should use plugins for analysis and verification modules.
S5	performance	The application should be performant enough for the GUI interface to be "snappy"
C	*Could have requirements*	
N	*Nice to have requirements*	
N1	install util	It would be nice to have the application provide installable packages to the users.

Table 3: High level requirements for the design tool

D.3 Team preferences

The team has explicit wishes concerning improving or learning competences. Table 4 specifies these wishes per team member.

	Prio	Competence	Goal
		Guus	
GM0	Must have	Free software	Prefer to build free software.
GM1	Must have	UI tool	Learning to work with a cross platform UI tool like QT or GTK+.
GS1	Should have	Agile	Learning to work in an agile environment.
GS2	Should have	C++	Learning to program C++ (standard 2011).
GC1	Could have	versioncontrol	Experiencing and learning teamwork with versioncontrol.
GC2	Could have	Requirements Eng.	Learning to apply Requirements engineering.
		Stefan	
SS1	Should have	Agile	Learning to work in agile environment, like DAD.
SS2	Should have	Online Agile tools	Learning to work agile tools.
SS3	Should have	distributed team	Learning to work in a distributed team.
SC1	Could have	Non Microsoft tools	Learning to work with platform independent tools.
		Jeroen	
JS1	Should have	Project teamwork	Learning to work in a project team
JS2	Should have	C++	Learning to program C++ (standard 2011)
JS3	Should have	TDD / testing	Learning unit testing, if possible TDD

Table 4: Competence wishes of the team

E Phase 0 Requirement Impact

E.1 Phase 0 selection results

Using the requirements for the application plus the competence requirements of the team, the impact of all these demands is summarized in table 6. Only requirements directly relevant to the ui toolkit have consequential impact formulated.

In Table 6 the term "support for plugins" is meant to be a weak requirement in the sense that it does not hinder the design of plugins. The ability to support plugins directly is a nice to have requirement. The derived requirements "ease of use" and "ease of learning" are important requirements for the people maintaining the system.

Remark. The term "UI toolkit" may indicate one set of tools (like qt) or a multiple sets (like one for 2D and one for GUI elements).

id: requirement	**impact**	**id**	**UI requirement**
M0: Free software	full	0	UI toolkit must be free software as defined by FSF.
M1: Cross platform	full	1	UI toolkit runs equally well on the defined platforms for all features.
M2: C++ integration	full	2	UI toolkit does not hinder C++ integration in any way.
M3: 2D drawing	full	3	UI toolkit has equal 2D features on all defined platforms.
M4: GUI Widgets	full	4	UI toolkit has relevant GUI widgets on all defined platforms.
M5: xmas primitives	none		
M6: xmas macros	none		
S1: expandability	limited	5	UI toolkit does not hinder creating plugins.
S2: installability	limited	6	UI toolkit does not hinder install procedure.
S3: maintainability	moderate	7	UI toolkit is easy to use.
		8	UI toolkit is easy to learn.
		9	UI toolkit is well documented.
		10	UI toolkit supports observer pattern.
		11	UI toolkit dev. community should have enough mass and be active in order to inspire confidence in the tool kits future.
		12	UI toolkit user community should have enough mass. and be active in order to inspire confidence in the tool kits future.

UI toolkit requirements (Continue on next page)

id: requirement	**impact**	**id**	**UI requirement**
S4: plugability	limited	13	UI toolkit should not hinder creating plugins.
S5: performance	moderate	14	UI toolkit should support running concurrent processes for modules.
		15	UI toolkit should support observer pattern from independent processes.
		16	UI toolkit should support interruption during concurrent processing.
N1: install util	none		
GM0: Free Software			Enclosed in requirement M0.
GM1: UI tool			Enclosed in requirement M1.
GS1: Agile			
GS2: C++	compatible	20	Prefer a UI toolkit that works with C++.
GC1: versioncontrol			
GC2: req. engineering			
SS1: agile			
SS2: online agile tools			
SS3: distributed team			
SC1: platform independent	compatible	21	Prefer a platform independent toolkit.
JS1: project			
JS2: C++	compatible	20	Prefer a UI toolkit that works with C++.
JS3: TDD			

Table 6: UI toolkit requirements derived.

F Phase 1 Preselection

The amount of tool kits available is staggering, so the first step is weeding out the tool kits that do not support one of our defined platforms, are not free software of just do not interact with C++ sufficiently. For that reason this analysis does not consider the following tool kits:

name	**reason**
AppearIQ	non-free software. Ref: [1].
Eclipse	a compiler, not a graphical framework. Ref: [17].
GeneXus	non-free software. Also not for linux or mac. Ref: [21].
Haxe	not a graphics platform. Ref: [23]
Max	not for linux, not free software. Ref: [26]
Mono	not a graphics environment, emulation of C#. Ref: [27].
MonoCross	aimed at C#. Ref: [28].
MoSync	aimed at mobile platforms, no longer maintained. Ref: [29].
Xojo	non-free software. Ref: [40].
Smartface	non-free software[5]. Ref: [34] and [13].
WebDev	aimed at creating websites. Ref: [37].
WinDev	aimed at data centric apps with forms, works with web-dev. Ref: [38].
XPower++	Insufficient information, looks like an advertisment. Ref: [41].
Lazarus	A Pascal development environment. Ref: [25].
Ultimate++	Does not support MacOS. Ref: [36] and [42].

Table 7: Platforms not considered with reason

G Phase 2 List of cross platform tool kits

General source: see [16]. Left out non-free software or software not for defined platforms. Due to the amount of toolkits, any obvious disconnect with C++ is also reason to drop a choice[6].

Cairo is a library used to provide a vector graphics-based, device-independent API for software developers. It is designed to provide primitives for 2-dimensional drawing across a number of different backends. Cairo is designed to use hardware acceleration when available. The software is free software with a user license based on GPL and MPL.
Ref: [14].

Cairo has a wrapper `cairomm` for C++. Cairo is popular in the free software communicty for providing cross-platform support for advanced 2D drawing. Other GUI library (notably GTK+) already include Cairo. This library does not contain GUI widgets and so should be combined with `SomeGUILibrary`. The verdict for Cairo is Partial.

FLTK The Fast, Light Toolkit (FLTK, pronounced fulltick) is a cross-platform graphical control element (GUI) library developed by Bill Spitzak and others. Made to accommodate 3D graphics programming, it has an interface to OpenGL, but it is also suitable for general GUI programming.
Ref: [18].

FLTK is a GUI only library. For 2D a different library should be used. The toolkit is not included in the final comparison.

fpGUI the Free Pascal GUI toolkit, is a cross-platform graphical user interface toolkit developed by Graeme Geldenhuys. fpGUI is open source and free software, licensed under a Modified LGPL license. The toolkit has been implemented using the Free Pascal compiler, meaning it is written in the Object Pascal language. fpGUI consists only of graphical widgets or components, and a cross-platform 2D drawing library. fpGUI is statically linked into programs and is licensed using a modified version of LGPL specially designed to allow static linking to proprietary programs. The only code you need to make available are any changes you made to the fpGUI toolkit - nothing more.
Ref: [19].

fpGUI is a pascal library based on free pascal (lazarus). The author is also the sole maintainer of the system. According to the WxWidget comparison Lazarus has no C++ integration to speak of. For this reason this toolkit is not included in the final comparison.
Ref: [6] and [42].

[6] like Lazarus, that in theory might be able to support the application, but has no direct integration with C++.

GTK+ (previously GIMP Toolkit, sometimes incorrectly referred to as the GNOME Toolkit) is a cross-platform widget toolkit for creating graphical user interfaces. It is licensed under the terms of the GNU LGPL, allowing both free and proprietary software to use it. It is one of the most popular toolkits for the Wayland and X11 windowing systems, along with Qt.
Ref: [22].

GTK+ builds on Cairo and GDK and has a large developer and user community. It has a wrapper for C++ (`GTKMM`). It fulfills the main requirements and is included in the final comparison.
Ref: [11].

JUCE is a free software, cross-platform C++ application framework, used for the development of GUI applications and plug-ins. The aim of JUCE is to allow software to be written such that the same source code will compile and run identically on Windows, Mac OS X and Linux platforms. It supports various development environments and compilers, such as GCC, Xcode and Visual Studio.
Ref: [24].

A one man project with emphasis on audio. It runs all defined platforms, written in C++. The user license is dual GPL and commercial. The verdict according to requirements is Maybe due to the size of the developer community. It is not included in the final comparison.

CEGUI Crazy Eddie's GUI System is a free library providing windowing and widgets for graphics APIs / engines where such functionality is not natively available, or severely lacking. The library is object-oriented, written in C++, and targeted at game and application developers who should be spending their time creating great games and not on building GUI sub-systems!
Ref: [2].

This toolkit satisfies the main requirements, but has a small developer community ([15] and [3]. The verdict for this toolkit is Maybe due to the size of the developer community. It is not included in the final comparison.

Mozilla Application Framework is a collection of cross-platform software components that make up the Mozilla applications. It was originally known as XPFE, an abbreviation of cross-platform front end. While similar to generic cross-platform application frameworks like GTK+, Qt and wxWidgets, the intent is to provide a subset of cross-platform functionality suitable for building network applications like web browsers, leveraging the cross-platform functionality already built into the Gecko layout engine.
Ref: [30].

The MAF is platform independent, web oriented with a large developer and user community[7]. The library is meant to support a subset of the standard GUI frame-

[7] All firefox users are part of the user community

works like GTK+, QT and WxWidgets, aimed at web-programs. Due to the web orientedness and it's partial support for widgets this toolkit is not selected for the final comparison.

OpenGL is a cross-language, multi-platform application programming interface (API) for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit (GPU), to achieve hardware-accelerated rendering[8].
Ref: [31].

OpenGL gives hardware acceleration and is a powerful but complex 3D user interface toolkit. It is also a low level library and does not directly support any GUI widgets. Many GUI libraries support the use of an OpenGL library. This toolkit is not selected for final comparison.

Qt (/kjut/ "cute", or unofficially as Q-T cue-tee) is a cross-platform application framework that is widely used for developing application software that can be run on various software and hardware platforms with little or no change in the codebase, while having the power and speed of native applications. Qt is currently being developed both by the Qt Company, a subsidiary of Digia, and the Qt Project under open-source governance, involving individual developers and firms working to advance Qt.

Digia owns the Qt trademark and copyright. Qt is available with both proprietary and open source GPL v3 and LGPL v2 licenses.
Ref: [32].

Qt has a large developer and user community. According to the WxWidget comparison with Qt, both are functionally comparable.
Ref: [42].

Simple DirectMedia Layer is a cross-platform software development library designed to provide a low level hardware abstraction layer to computer hardware components. Software developers can use it to write high-performance computer games and other multimedia applications that can run on many operating systems such as Android, iOS, Linux, Mac OS X, Windows and other platforms.

SDL manages video, audio, input devices, CD-ROM, threads, shared object loading, networking and timers.[5] For 3D graphics it can handle an OpenGL or Direct3D context.

The library is internally written in C and also provides the application programming interface in C, with bindings to other languages available. It is free and open-source software subject to the requirements of the zlib License since version 2.0 and with prior versions subject to the GNU Lesser General Public License. Because of zlib SDL 2.0 is freely available for static linking in commercial closed-source projects, unlike SDL 1.2. SDL is extensively used in the industry in both

[8]OpenGL is a standard that has libraries based on free software and commercial libraries. Programmers never need a license to use an OpenGL library.

large and small projects. Over 700 games, 180 applications, and 120 demos have also been posted on the library website.

It is often believed that SDL is a game engine, but this is not true. However, the library is well-suited for building an engine on top of it.
Ref: [33].

SDL is a low level library meant for 2D game development. It does not contain GUI widgets. It may be usable in combination with other libraries. It is not selected for final comparison.

Tk is a free and open-source, cross-platform widget toolkit that provides a library of basic elements of GUI widgets for building a graphical user interface (GUI) in many different programming languages.
Ref: [35].

Tk seems to contain everything necessary for both GUI and 2D drawing, but C++ integration is not clear ([35], [43]). According to the comparison on WxWidgets, for C++ it is better not to use Tk.
Ref: [42].

Ultimate++ is a C++ cross-platform development framework which aims to reduce the code complexity of typical desktop applications by extensively exploiting C++ features.
Ref: [36].

WxWidgets (formerly wxWindows) is a widget toolkit and tools library for creating graphical user interfaces (GUIs) for cross-platform applications. wxWidgets enables a program's GUI code to compile and run on several computer platforms with minimal or no code changes. It covers systems such as Microsoft Windows, OS X (Carbon and Cocoa), iOS (Cocoa Touch), Linux/Unix (X11, Motif, and GTK+), OpenVMS, OS/2 and AmigaOS. A version for embedded systems is under development.
Ref: [39]

The framework is complete and has no obvious disadvantages. According to their comparison to Qt, the two are functional comparable.
Ref: [42].

Gled is a C++ framework for rapid development of applications with support for GUI (using FLTK), 3D-graphics and distributed computing. It extends the ROOT framework (standard data-analysis tool in high-energy physics) with mechanisms for object collection management & serialization, multi-threaded execution, GUI auto-generation (object browser & editor) and dynamic visualization (OpenGL). Distributed computing model of Gled is a hierarchy of nodes connected via TCP/IP sockets. Gled provides authentication & access control, data exchange, proxying of object collections and remote method-call propagation & execution. Gled can be dynamically extended with library sets. Their creation is facilitated by a set of

scripts for creation of user-code stubs. Simple tasks and application configuration can be efficiently done via the interactive C++ interpreter (CINT). Gled is used for development of programs in high energy physics and as a research tool in distributed and grid computing.
Ref: [5]

This toolkit satisfies the main requirements, but has a small developer community ([7]). It is not included in the final comparison.

GDK (GIMP Drawing Kit) is a library that acts as a wrapper around the low-level functions provided by the underlying windowing and graphics systems. GDK lies between the display server and the GTK+ library, handling basic rendering such as drawing primitives, raster graphics (bitmaps), cursors, fonts, as well as window events and drag-and-drop functionality.

Like GTK+, GDK is licensed under the GNU Lesser General Public License (LGPL).
Ref: [20]

GDK is a low level library that is "An intermediate layer which isolates GTK+ from the details of the windowing system.". In the presence of GTK+ the toolkit is not included in the final comparison.
Ref: [8].

References

[1] appear. Welcome to appeariq - aiq8 developer portal. https://www.appeariq.com accessed oct/nov 2014.

[2] CEGUI. Getting started — cegui. http://cegui.org.uk/content/getting-started accessed oct/nov 2014.

[3] CEGUI. News —cegui. http://cegui.org.uk/ accessed oct/nov 2014.

[4] Chedman. a russian title - few programming: Model-view-controller (mvc) + wxwidgets. http://chedman.blogspot.nl/2011/07/model-view-controller-mvc-wxwidgets.html accessed oct/nov 2014.

[5] FLTK. Fast light toolkit (fltk). http://www.fltk.org/wiki.php?L+P22 accessed oct/nov 2014.

[6] Graeme Geldenhuys. Free pascal gui toolkit - fpgui, fpgfx, fpimg. http://opensoft.homeip.net/fpgui/ accessed oct/nov 2014.

[7] Gled. gled. http://www.gled.org accessed oct/nov 2014.

[8] Gnome. Gdk 3 reference manual. https://developer.gnome.org/gdk3/ accessed oct/nov 2014.

[9] Developer gnome. Dependencies. https://developer.gnome.org/gtkmm-tutorial/stable/sec-installation-dependencies.html.en accessed oct/nov 2014.

[10] Greed. Greed. http://greed.gled.org/ accessed oct/nov 2014.

[11] gtkmm. gtkmm - c++ interfaces for gtk+ and gnome. http://www.gtkmm.org/en/ accessed oct/nov 2014.

[12] Qt. show_library_dependencies — qt wiki — qt project. http://qt-project.org/wiki/Show_library_dependencies accessed oct/nov 2014.

[13] Smartface. Smartface. http://www.smartface.io/license/ accessed oct/nov 2014.

[14] wikipedia. Cairo (graphics) - wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Cairo_(graphics) accessed oct/nov 2014.

[15] Wikipedia. Cegui - wikipedia, the free encyclopedia.
http://en.wikipedia.org/wiki/CEGUI accessed oct/nov 2014.

[16] wikipedia. Cross Platform - Wikipedia, the free encyclopedia.
http://en.wikipedia.org/wiki/Cross-platform accessed oct/nov 2014.

[17] wikipedia. Eclipse (software) - wikipedia, the free encyclopedia.
http://en.wikipedia.org/wiki/Eclipse_(software) accessed oct/nov 2014.

[18] wikipedia. Fltk - wikipedia, the free encyclopedia.
http://en.wikipedia.org/wiki/FLTK accessed oct/nov 2014.

[19] wikipedia. fpgui - wikipedia, the free encyclopedia.
http://en.wikipedia.org/wiki/FpGUI accessed oct/nov 2014.

[20] wikipedia. Gdk - wikipedia, the free encyclopedia.
http://en.wikipedia.org/wiki/gdk accessed oct/nov 2014.

[21] wikipedia. Genexus - wikipedia, the free encyclopedia.
http://en.wikipedia.org/wiki/GeneXus accessed oct/nov 2014.

[22] wikipedia. Gtk+ - wikipedia, the free encyclopedia.
http://en.wikipedia.org/wiki/GTK+ accessed oct/nov 2014.

[23] wikipedia. Haxe - wikipedia, the free encyclopedia.
http://en.wikipedia.org/wiki/HaXe accessed oct/nov 2014.

[24] wikipedia. Juce - wikipedia, the free encyclopedia.
http://en.wikipedia.org/wiki/JUCE accessed oct/nov 2014.

[25] wikipedia. Lazarus (ide) - wikipedia, the free encyclopedia.
http://en.wikipedia.org/wiki/Lazarus_(IDE) accessed oct/nov 2014.

[26] wikipedia. Max (software) - wikipedia, the free encyclopedia.
http://en.wikipedia.org/wiki/Max_(software) accessed oct/nov 2014.

[27] wikipedia. Mono (software) - wikipedia, the free encyclopedia.
http://en.wikipedia.org/wiki/Mono_(software) accessed oct/nov 2014.

[28] wikipedia. Monocross - wikipedia, the free encyclopedia.
http://en.wikipedia.org/wiki/MonoCross accessed oct/nov 2014.

[29] wikipedia. Mosync - wikipedia, the free encyclopedia.
http://en.wikipedia.org/wiki/MoSync accessed oct/nov 2014.

[30] wikipedia. Mozilla application framework - wikipedia, the free encyclopedia.
http://en.wikipedia.org/wiki/Mozilla_application_framework accessed oct/nov 2014.

[31] wikipedia. Opengl - wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/OpenGL accessed oct/nov 2014.

[32] wikipedia. Qt (software) - wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Qt_(software) accessed oct/nov 2014.

[33] wikipedia. Simple directmedia layer - wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Simple_DirectMedia_Layer accessed oct/nov 2014.

[34] wikipedia. Smartface - wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Smartface accessed oct/nov 2014.

[35] wikipedia. Tk (software) - wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Tk_(software) accessed oct/nov 2014.

[36] wikipedia. Ultimate++ - wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/ultimate++ accessed oct/nov 2014.

[37] wikipedia. Webdev - wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/WebDev accessed oct/nov 2014.

[38] wikipedia. Windev - wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/WinDev accessed oct/nov 2014.

[39] wikipedia. Wxwidget - wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/wxwidget accessed oct/nov 2014.

[40] wikipedia. Xojo - wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Xojo accessed oct/nov 2014.

[41] wikipedia. Xpower++ - wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/XPower++ accessed oct/nov 2014.

[42] WxWidget. Wxwidgets compared to other toolkits. http://wiki.wxwidgets.org/WxWidgets_Compared_To_Other_Toolkits accessed oct/nov 2014.

[43] TCL Developer Xchange. Tcl/tk starkits. http://www.tcl.tk/starkits/ accessed oct/nov 2014.

H Definitions

GPL Gnu Public License

LGPL Lesser Gnu Public License

MPL Mozilla Public License

UI User Interface

GUI Graphical User Interface

the defined platforms for the design application Linux, Mac and MS Windows are the defined platforms.