IJARIIT
INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact Factor: 6.078

(Volume 7, Issue 2 - V7I2-1491)
Available online at: https://www.ijariit.com

XML based UI designer using C++

Medhavi Pimple
medhavipimple@ternaengg.ac.in
Terna Engineering College, Navi Mumbai, Maharashtra

Harshada Bauskar
harshadabauskar@ternaengg.ac.in
Terna Engineering College, Navi Mumbai, Maharashtra

Vartika Chand
vartikachand@ternaengg.ac.in
Terna Engineering College, Navi Mumbai, Maharashtra

Sneha Sabale
snehasabale@ternaengg.ac.in
Terna Engineering College, Navi Mumbai, Maharashtra

Nilesh Kulal
nileshkulal@ternaengg.ac.in
Terna Engineering College, Navi Mumbai, Maharashtra

ABSTRACT

The application aims for generating a Graphical User Interface (GUI) from XML-based documents. Within the application, display components for building a GUI are defined, also a layout hierarchy will be describing the layout relationships between the displayed objects, specifying how relative these display components are on a layout window in the GUI. The appearance of display objects within the GUI may also be altered which are humanly readable that makes debugging relatively easy. So, this is an application wherein the XML files can be read, modified, and also be extended to contain new information without breaking existing implementations. Most significantly, describing interfaces as structured documents allows efficient reuse of composite UI objects. Thanks to this manipulation and storage, users can create and customize interactive layouts.

Keywords— *XML, TinyXML, UI layout, parsing, modified, GUI*

1. INTRODUCTION

Graphical User Interface (GUI) is what everybody gets attracted to and is interested in creating one of its own. People like to try new layouts, color combinations, or components of an interface, etc. and so keeps modifying it frequently. As and when one starts to build, it gets complicated with time. With the day-to-day evolution and the improvement in the programming languages, we can simplify working with multiple interfaces. XML is a markup language that is developed to be self-descriptive and self-understanding. It stores information in plain text format that provides the software to store, transport, and share information in an independent way. TinyXML, a small, simple XML parser for the C++ language which can be easily integrated with different programs. It reads the document and creates objects representing the XML file. The objects are reusable as they can be manipulated, updated, and saved again as XML. As the name suggests, the principal advantage for TinyXML is its size. It parses the XML into a DOM-like tree that can both read and write XML files. In this project, we will build an application that will accept an XML file as an input and it will create a GUI layout based on the XML file that was provided to the system.

2. RELATED WORK

[1] C ++ programming language is familiar to those in the programming world. This language is a powerful programming language used for several applications. Borland C ++ Builder is a visual programming language that uses the C ++ language. This software can be used, for creating graphical application programs easily and quickly. [2] XML to User Interface, a way that permits developers to simply create dialog boxes for its behaviours and effects. By using XML for the user interface, you will be able to offer users a way to modify parameters for your behaviour and to perform scripts without having to build common interface containers or any other "widgets". [3] The main task of C++ implementation of the XML parser and the corresponding document object model interface is to provide a simple, high-speed, and low memory XML parsing interface for applications, and provide as much DOM support as possible under the premise of ensuring performance. [4] Choosing widgets once coming up with user interfaces may be a rather intuitive task. No clearly established methodology with précised descriptions of widgets or explicit rules for selecting them has been enclosed within the main development processes. Widget catalogues do not openly view the representation of domain knowledge rather they are only partially specified as related to current tasks. Moreover, these representations haven't been the

topic of any systematic approach with respect to its structure. The primary objective is to formally describe the widgets that can form from its structures. We then look after the series of functionalities permitting to classify widgets with their data content structure and to define a framework for the outline of widgets. [5] A light-weight extractor utilizes XML tools to extract static data from C++ programs. Partial parsing makes the strategy light-weight. The trade off to this approach is that queries on some low-level details can't be directly addressed. This technique is applied to extractor benchmark as comparison with different, heavier weight extractors.

3. EXISTING SYSTEM

Today, various programs and devices use it to handle the structure, store, transmit, and display the data. XML files are encoded in plaintext, so we can open them in text editors and can read or edit them. For a single change user had to navigate to the corresponding XML file and then traverse the whole code to find the block to be edited and then edit the required part. In real-world XML applications are enormous and changing the resources of the code is not an easy task. Also, various individuals simultaneously invest their work in a single application, so it becomes more and more difficult when a subsequent change is to be made. This will need us to bring change in the entire code and time management plays a crucial role for big business firms.

4. PROPOSED SYSTEM

We are building an XML-based UI designer application for designing UI layout using C++. Our project's main objective serves the purpose of giving a GUI layout based on the XML file that has been provided to the system. Keeping in mind it is necessary that before changing the actual code there is an opportunity to visualize the existing code or visualize the required change in the code before reflecting changes in it.

(a) Generating a UI layout with a more user-friendly and interactive XML-based documents.
(b) Various designs can be given to a single layout requirement keeping the functionalities untouched.
(c) Hassle-free modification, up-gradation of the GUI layout look is achievable.
(d) This will help the user to discover the most appropriate layout for the handlers or customers to interact with.
(e) As the project will not depend on the OS and the hardware, therefore, it will be platform independent system.

5. DETAILS OF HARDWARE AND SOFTWARE

5.1 Hardware requirements

Hardware CPU; Operating System Dependent Memory: 2GB, 32-Bit color, x86-64 (64 bit)

5.2 Software requirements

C++, MinGW

5.3 Operating System

Windows 7 (32-bit or 64-bit) and plus

6. SYSTEM IMPLEMENTATION AND DESIGN DETAILS

The following diagram, Figure-1 represents the flow of our system. It depicts how the user designs its:

XML file: This file carries a XML Layout according to the users' requirements which gives the outline of how the user wants its structure to look.

XML Parser: Sysem reads the documents received from the user and delivers it to the this phase, XML Parser. This is the essential part of the project where each and every widget in the XML file is parsed by the system using TinyXML.

GUI Layout: After a successfull parsing users gets to see the created UI layout on the command prompt.

Algorithm:

STEP 1: Create an XML file as per user requirements.
STEP 2: The XML document is imported in the application by the user created in 1st step.
STEP 3: In this step, the XML Layout gets parsed using TinyXML.
STEP 4: After the successful parsing the user achieves its UI layout displayed on the command prompt.

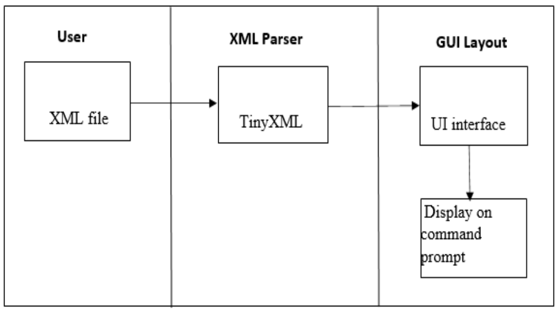
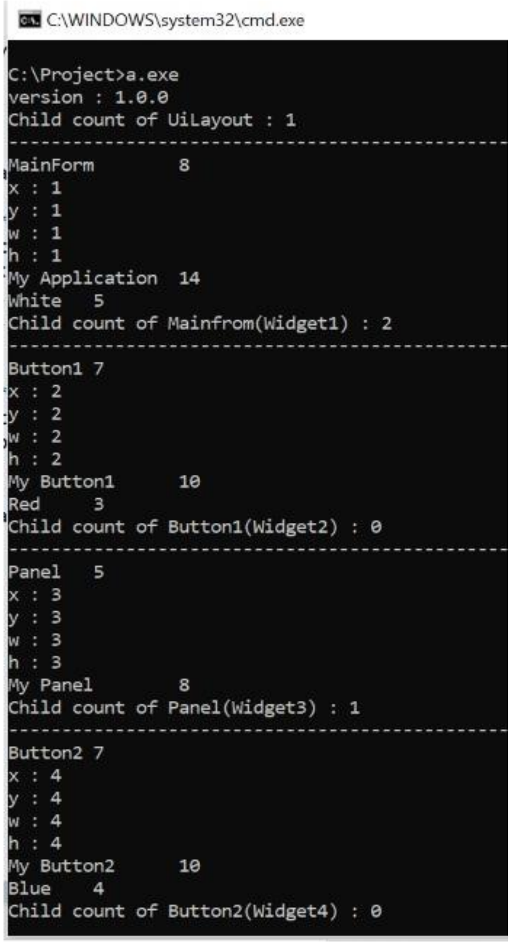


Fig. 1: Design Model

7. OBSERVATIONS AND OUTPUTS



```
C:\WINDOWS\system32\cmd.exe

C:\Project>a.exe
version : 1.0.0
Child count of UiLayout : 1
--------------------------------------------------
MainForm        8
x : 1
y : 1
w : 1
h : 1
My Application  14
White   5
Child count of Mainfrom(Widget1) : 2
--------------------------------------------------
Button1 7
x : 2
y : 2
w : 2
h : 2
My Button1      10
Red     3
Child count of Button1(Widget2) : 0
--------------------------------------------------
Panel   5
x : 3
y : 3
w : 3
h : 3
My Panel        8
Child count of Panel(Widget3) : 1
--------------------------------------------------
Button2 7
x : 4
y : 4
w : 4
h : 4
My Button2      10
Blue    4
Child count of Button2(Widget4) : 0
```

Fig. 2: UI Layout obtained after parsing of XML structure

8. FEATURES

(a) Safe to modify input XML files as rest of the data will remain undisturbed.
(b) Ease to obtain frequently modifying GUI layouts.
(c) Hassle-free execution as system is platform friendly.
(d) It does not compromise with user data.
(e) Good time management as subsequent changes won't harm rest of the code.
(f) High productivity due to reduced development time
(g) Lower development costs
(h) User satisfaction

9. CONCLUSION

In this growing technological world, we learn and understand that there is technical solution to uplift and upgrade every problem. Similarly, our system provides the user to open XML documents in an easy-going way which gives précised GUI layout as an output. This reduces a particular amount of time and focuses on user's fulfilment. As this system allows user to modify the document which brings visual attraction, it is user-friendly, less time-consuming and a system where a user can play with its XML inputs and try new UI Layouts.

10. FUTURE SCOPE

We will be using call-back that will come from the DLLs for writing extendable applications were the user provides a way to bind the implemented functions to the generated UI. So that the user could attach the functions with the widgets via XML. This could be an enhancement, where the user will provide two inputs: 1. XML for UI layout 2. A DLL for call-back.

11. REFERENCES

[1] Andik Setiawan, "Graphical User Interface with C++ Builder", Published by Deepublish, 2019.
[2] Keith Peters; Todd Yard, "XML to UI", Extending Macromedia Flash MX, Apress, Berkeley, CA, 2004, pp163-193.
[3] Ying Liu; Si-liang WANG; Yue-mei XIE, "Implementation and Research of C++-oriented XML Parser", IEEE Conference on Proceedings of International Conference on Software Maintenance, China, 2007.
[4] Tanguy Wettengel; "A Typology of Widgets for UI Design", in IEEE Proceedings of International Conference on Software Maintenance, Canada, 2009.
[5] Michael Collard, Huzefa Kagdi, Jonathan I. Maletic; "An XML-based lightweight C++ fact extractor", Conference on 11th International Workshop on Program Comprehension (IWPC 2003), Portland, Oregon, USA 2003
[6] https://www.technical-recipes.com/2014/getting-started-with-tinyxml/
[7] http://www.grinninglizard.com/tinyxml/
[8] https://ieeexplore.ieee.org/Xplore/home.jsp
[9] https://www.w3schools.com/
[10] https://stackoverflow.com/questions/18507351/how-to-create-custom-button-in-android-using-xml-styles
[11] http://www.interactivexml.com/2010/07/contains-one-or-more-tags-defined_14.html
[12] https://www.researchgate.net/publication/228405417_XML_Based_Graphical_User_Interface_Editor_and_Runtime_Parser_for_ISO_11783_Machine_Automation_Systems
[13] https://quizlet.com/494079240/xml-introduction-flash-cards