ISSN: 2321-8169
2612 – 2616

HMI for Interactive 3D Images with Integration of Industrial Process Control

Gadhia Deep H.
Embedded System Design
GTU PG School
Ahmedabad, India
deepgadhia30@gmail.com

Mr. Chaitannya Mahatme
Zeroes & Ones Technologies
Pune, India
chaitannya@znotech.com

Vaghela Megha N.
Embedded System Design
GTU PG School
Ahemedabad, India
megha.vaghela2012@gmail.com

Abstract—This paper presents interactive 3D image use for HMI in industrial process control application. Visualization information has a very important role in industrial process. HMI (Human Machine Interface) is a device used in industries for GUI based display and control of the industrial processes. The goal is user should be able to view simple process through 3D images, interactive way and utilize touch based GUI to control process and change its behavior at run time. HMI currently implemented using 2D images to display information about the industrial process. This paper describes approaches for 3D images that are interactive and uses for controlling industrial process. OpenGL is used to render 3D graphics and QML (Qt Modeling Language) provide functionality for user interface using the Qt cross platform framework. Qt3D library provides a set of APIs to make 3D graphics programming easy and declarative. The developed system will be extended to integrate industrial process control application. Industrial Process communicates with target hardware using Modbus protocol.

Keywords— *HMI, QML, OpenGL, Qt3D, Modbus protocol*.

*****

I. INTRODUCTION

As there is an increased complexity of industrial plants, it leads to a growing amount of process information that has to be monitored and controlled by the operators. HMI is a device used in industries for GUI based display and control of the industrial processes. HMI is the part of device which serves the information exchange between user and machine. HMI consists of three parts which are operating elements, displays and inner structure.

Nowadays 2D representations are common in process data visualization in Human Machine Interface (HMI) like bar graphs, line diagrams and tables [1]. With the growing amount of information, these conventional types of visualization reach their limits and process visualization gets more confusing. User interaction is kept minimum in this kind of visualization. 3D diagrams are rarely used in process control. Classical HMI applications allow designing process images in 2D and connecting it with the process values in a simple way. 3D image integration can benefit the reduced complexity of the environment as compared with real tasks [10].

This paper is organized as follows. In section II explains 3D development process and technology and tool selection. Section III shows the system architecture in brief. Section IV includes implementation of communication protocols. Section V shows experimental results. Finally, Section VI concludes the paper with future work.

II. 3D DEVELOPMENT PROCESS AND SELECTION OF 3D DEVELOPING TOOL

The 3D development process consists of three major steps. They are building graphics using modeling tools, programming the application to use the graphics, and finally compiling and deploying the application [2]. This process is more generalized and applicable for most 3D development processes.

3D image content mainly classified into following three categories such as geometry, appearance and scene information.

A. Geometry:

The geometry of a model is defined as a set of vertices. Each primitive (basic building block) is known as a vertex or vertices. Thus, it is necessary to define vertices to manipulate them efficiently. Figure 1 shows a basic sphere with basic building block of the triangle. Set of triangles creates sphere.

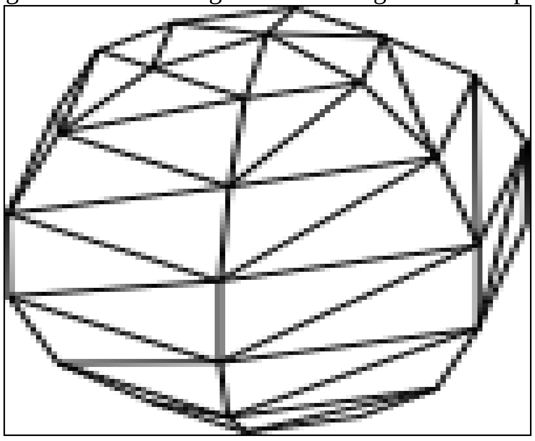


Figure 1 Geometry of Sphere

B. Appearance:

Any material entails applying an image or texture to surface of the model is known as appearance. Appearance is related to mapping three dimensional vertex to the related point to that 2D image. Figure 2 shows example of appearance.

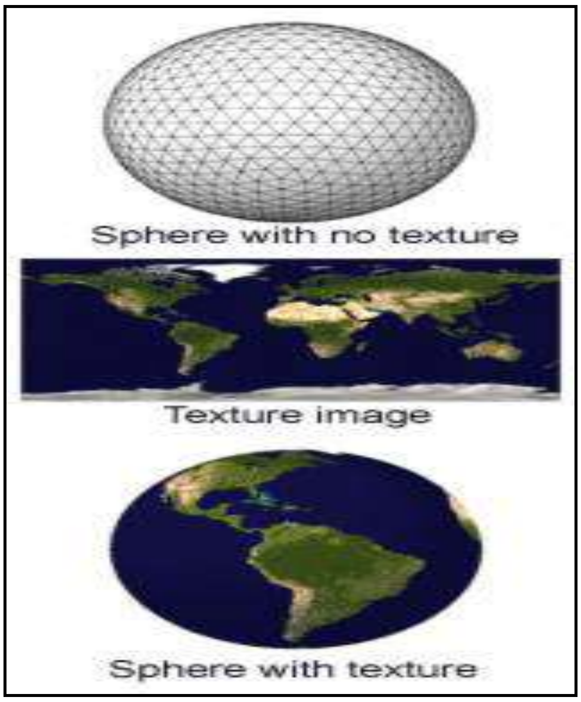


Figure 2 Sphere with texture

C. *Scene:*

Scene refers to layout of model with regards to the camera, shadow, light source and other many 3D models. Radiosity is a technique that models inter-reflection of light. Figure 3 shows radiosity example compared to ray tracer algorithm.

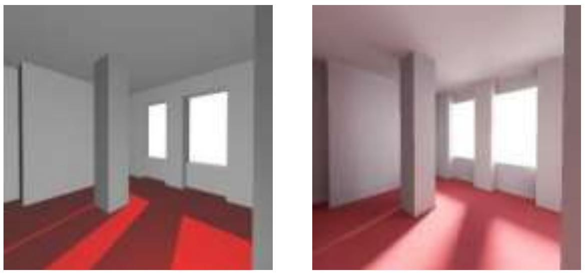


Figure 3 Scene effect with ray tracer and with technique radiosity

OpenGL:

OpenGL stands for Open Graphics Library. It is a software interface to graphics hardware. For 3D graphics API, OpenGL is most widely used. As it is highly portable, scalable, cross-platform specification that defines the interface for graphics accelerator [3]. OpenGL requires the programmer to tell exact scene. It requires geometry primitives in 3D space, apply color and lighting effects and render objects onto the screen. It is only concerned with rendering into the framebuffer. There is no support for other peripherals that are associated with hardware like user input. For that purpose Qt is useful.

Figure 4 shows OpenGL rendering pipeline that describes the basic steps of OpenGL takes to render any 3D picture. The graphics card uses its own memory and a GPU is highly specialized in processing 3D data like a small powerful computer. As shown in the figure 4 vertex specification, is done by the ordered list of vertices that gets streamed. In vertex shader processes data receives this stream of vertices along with additional attributes like associating texture coordinates or color values. The vertex shader can also pass data to the fragment shader directly. After that the primitive assembly stage is done by vertices are composed into primitives. These can be triangles, lines, point sprites, or more complex entities. During the clipping and culling stage, primitives that lie beyond the viewing volume are removed because it is not visible. The rasterization process also called fragments. These fragments related to pixels on the screen. Each fragment is then processed by the fragment shader. Stage of texture mapping and lighting are done here. Per-sample operations that is deciding which fragments should actually be written to the framebuffer and the final image is stored in the framebuffer [3].

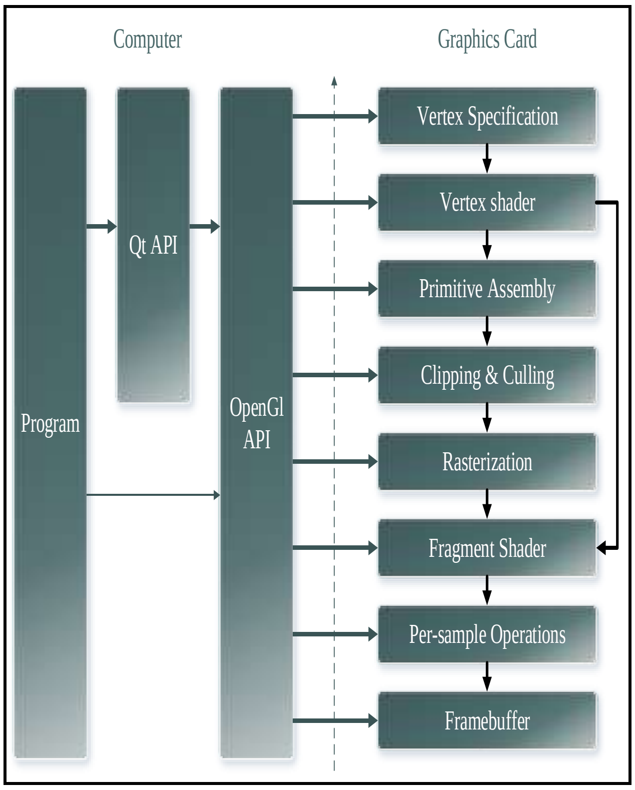


Figure 4 OpenGL rendering pipeline

Qt:

Qt is C++ framework which mostly supports cross-platform GUI application with “write once, compile anywhere” approach. The reason to use Qt is because it has excellent cross-platform support 3D graphics. Qt also support for OpenGL and OpenGL ES to integrate 3D graphics into applications. With the help of Qt, it is easy to create graphical user interface applications fast and simple [4].

QML:

QML stands for Qt declarative Markup Language or Qt Meta Language or Qt Modeling Language. QML mainly uses for user interface which is JavaScript base declarative language. The reason to choose QML is because of its performance. The interface rendering speed is higher than HTML 5 [6]. QML also benefits at less memory used by the runtime. QML has some effective properties like property bindings, states, animation, Qt signal handlers [9].

Qt3D:

The Qt3D module is a set of APIs. The reason to use Qt3D is for OpenGL and aim to 3D development easier and more platform independent. Qt3D module includes many features like asset loading, shapes, texture management and shaders. Qt3D uses C++ and QtQuick API. With help of Qt3D, a programmer need not worry about 3D architecture because it provides abstraction layer. So it is possible to create complex 3D application. Qt3D API included in Qt Quick. So first one needs to import Qt Quick module to use Qt3D. Mostly include elements in Qt3D are Viewport, Camera, Item3D, Mesh.

ARM processors now dominate in most of applications due to their lower power requirements and they have a very competitive price. By developing cross-platform applications, it is possible to use ARM based systems as industrial process and benefit from the advantages of this platform over traditional x86 systems. While developing applications, it is necessary that it can easily be ported to systems. Here in this paper, processer used for an experiment is an ARM Cortex A8 processor. Another requirement like touch screen interface and GPU, target hardware is Texas instruments am335x starter kit.

III. SYSTEM ARCHITECTURE

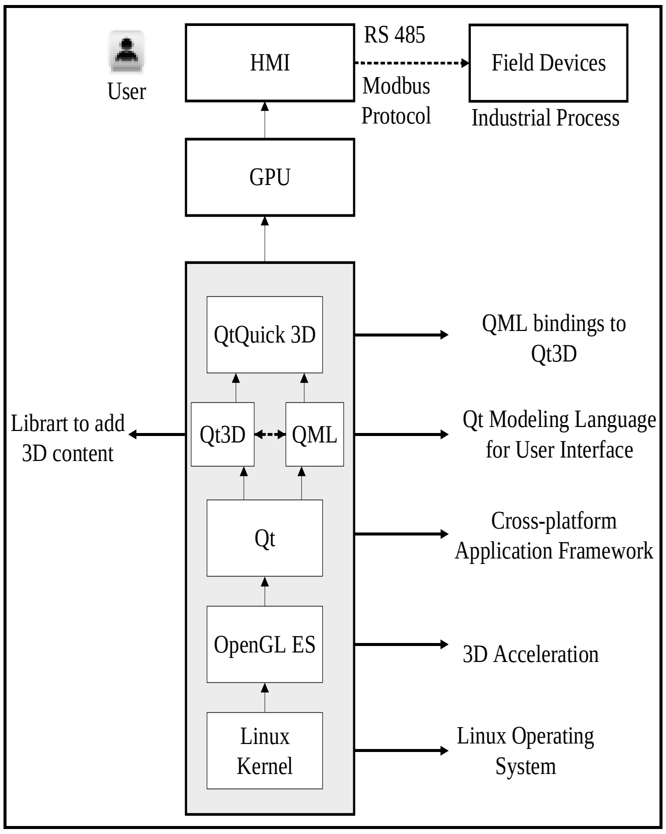


Figure 5 System architecture

Figure 5 shows the architecture of the system. The operating system that used to experiment is Linux based. For making the application platform independent, Qt framework is used. Qt includes QML programming for user interface and one can change system behavior at run time with QML. For rendering a 3D image Qt3D module used. Qt3D module includes OpenGL ES APIs. So, it gives basic abstraction layer to create 3D programming. One does not need to study deep knowledge of 3D graphics and OpenGL system. QtQuick3D combines Qt3D and QML bindings and other supporting tools. So product resulting from Qt3D project is QtQuick3D. It is made up of new user-defined QML3D items using C++ [7]. The Whole program runs on the GPU because 3D rendering is a lot heavier on the CPU. Here program use GPU accelerated rendering means CPU + GPU rendering alongside to speed up process.

A complex task is to cross-compile the Qt3D library for target hardware. After successfully compiled the program, porting different layer needed. The First thing to port is Qt and QML library on am335x starter kit. Ti's forum has lots of documents to support the development Qt application. Qt3D library based on GUI for 3D graphics rendering needs to be ported. Generated 3D image can use to fit any application. It is up to programmer for which application to use. To communicate 3D image with a field device, Modbus protocol is used. Modbus is an application layer protocol and for physical link can be wired or wireless. Here for experiment physical layer used is RS 485 serial line. Gateway is used to communicate in between HMI and field device. For bulb on and off use case, PLC can be used as an intermediate.

IV. COMMUNICATION PROTOCOL APPLICATION

There are many kinds of communication protocol exist that can implement here. In this paper, Modbus RTU protocol particular used. Modbus is a serial communication protocol which is mostly used in industry for devices like PLC, HMI. Here reason to choose a Modbus protocol is because of its Master slave principle. Modbus protocol can run over RS485 to gain faster speed, longer distances. Master HMI continuously sends signal to slave devices to check its status through RTU (remote terminal unit). Master can communicate with multiple slave devices. When Modbus RTU master wants information from the device, the master sends a message that contains the device's address with data and checksum for error detection. Every other device on the network can sees message, but only the device that is addressed responds. Slave devices can not initiate communication and can only respond to master [8].

Modbus message frame structure shown in table 1.

Address Field	Function Code	Data	Error Check

Table 1 Modbus message frame

Function code for user input is defined for Modbus as table 2.

Function Code	Action	Description
01 (01 hex)	Read	Discrete output coils
05 (05 hex)	Write signal	Discrete output coil
15 (0F hex)	Write multiple	Discrete output coils
02 (02 hex)	Read	Discrete output contacts
04 (04 hex)	Read	Analog input contacts
03 (03 hex)	Read	Analog output holding registers

06 (06 hex)	Write single	Analog output holding register
16 (10 hex)	Write multiple	Analog output holding registers

Table 2 Modbus function code

QML has the functionality of signals and slots. In QML code, 3D image has been tagged to a specific ID for communicating with a field device. So, Modbus message frame sent using RS 485 serial line and gateway sends it to other field devices.

V. Results

3D image for HMI approach needs to be tested on real time application. For that purpose simple bulb on off demo is used. First QML and Qt libraries ported on am335x starter kit and run sample helloworld application. Figure 6 shows Qt and QML ported libraries demo.

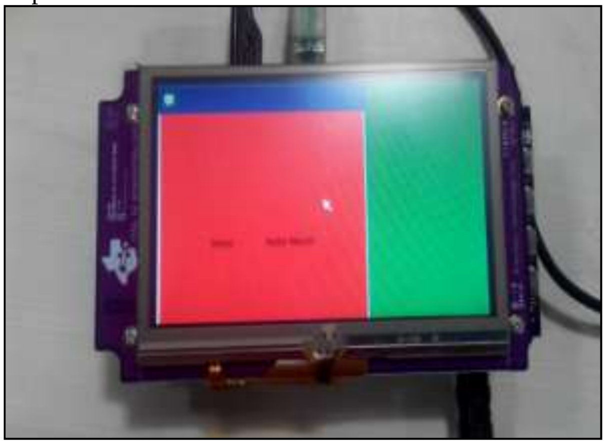
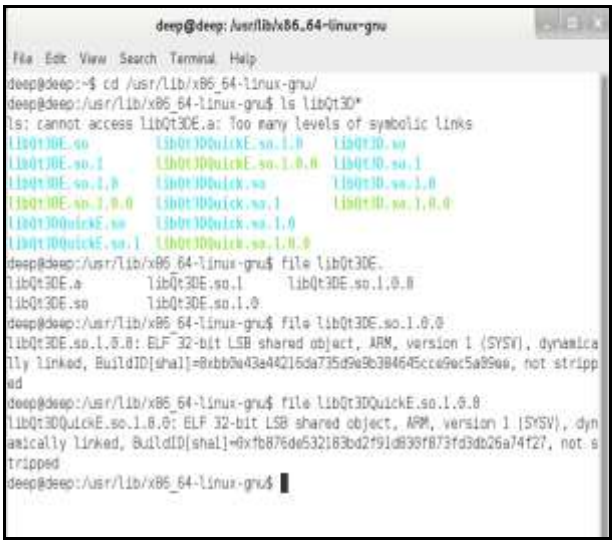


Figure 6 QML test application on am335x starter kit

Cross-compilation of Qt3D library generates shared library of Qt3D and Qt3DQuick for ARM. Figure 7 shows generated 3D library for ARM architecture.


deep@deep: /usr/lib/x86_64-linux-gnu
File Edit View Search Terminal Help

```
deep@deep:~$ cd /usr/lib/x86_64-linux-gnu/
deep@deep:/usr/lib/x86_64-linux-gnu$ ls libQt3D*
ls: cannot access libQt3DE.a: Too many levels of symbolic links
libQt3DE.so          libQt3DQuickE.so.1.0    libQt3D.so
libQt3DE.so.1        libQt3DQuickE.so.1.0.0  libQt3D.so.1
libQt3DE.so.1.0      libQt3DQuick.so         libQt3D.so.1.0
libQt3DE.so.1.0.0    libQt3DQuick.so.1       libQt3D.so.1.0.0
libQt3DQuickE.so     libQt3DQuick.so.1.0
libQt3DQuickE.so.1   libQt3DQuick.so.1.0.0
deep@deep:/usr/lib/x86_64-linux-gnu$ file libQt3DE.
libQt3DE.a           libQt3DE.so.1       libQt3DE.so.1.0.0
libQt3DE.so          libQt3DE.so.1.0
deep@deep:/usr/lib/x86_64-linux-gnu$ file libQt3DE.so.1.0.0
libQt3DE.so.1.0.0: ELF 32-bit LSB shared object, ARM, version 1 (SYSV), dynamica
lly linked, BuildID[sha1]=0xbb0a43a44215da735d9e9b384645cca9ec5a89ee, not stripp
ed
deep@deep:/usr/lib/x86_64-linux-gnu$ file libQt3DQuickE.so.1.0.0
libQt3DQuickE.so.1.0.0: ELF 32-bit LSB shared object, ARM, version 1 (SYSV), dyn
amically linked, BuildID[sha1]=0xfb876de532183bd2f91d838f873fd3db26a74f27, not s
tripped
deep@deep:/usr/lib/x86_64-linux-gnu$ ▌
```

Figure 7 3D library after cross-compilation

Qt3D module is not officially released, but one can download it from git. Simple bulb demo successfully gives output on the screen and the program runs on the GPU. Output of 3D bulb image shown in figure 8.

Modbus RTU sends a message frame to the gateway using RS 485 and for serial communication am335x starter kit has 6 uart port. Message sends with providing tag to bulb address on field device and the device responds to that message in industry one can implement a whole system in 3D image like for example, water plant system and can control using touch based HMI.

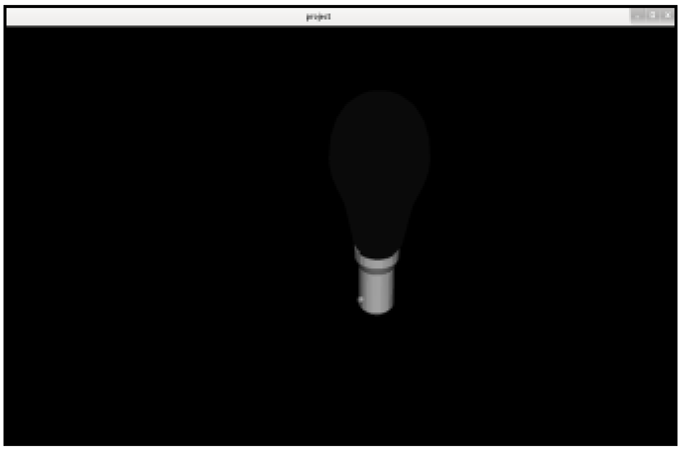


Figure 8 3D bulb example using Qt3D

VI. Conclusion and Future Work

The goal of this research work is user should able to view simple process through 3D image and also utilize touchscreen based GUI to control any industrial process. Experiment result shows 3D image that runs on the GPU and uses OpenGL ES APIs. For simple applications, here bulb image renders on screen and with providing tag to that bulb we can communicate real time bulb. The user can interact with any 3D industrial image as they need and with communication protocol. It can give us output of user defined application at runtime. 3D image technology for HMI is rarely used in industry. So in future work one can also test 3D image approach in big industrial plant like water plant or car automation industry and many other control application.

References

[1] Dorothea Pantförder, Birgit Vogel-Heuser,"Benefit and evaluation of interactive 3D process data visualization in operator training of plant manufacturing industry,"IEEE International Conference on Systems, Man and Cybernetics, 2009.

[2] Nils Johansson,Filip Williamsson,“ human-machine interface (HMI) for a Quality Control System produced by ABB” article on propose concepts on how 3D graphics could solve some of the issues with this interface.

[3] João Paulo Gois, Harlen C. Batagelo “Interactive Graphics Applications with OpenGL Shading Language and Qt,” 25th SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T), 2012.

[4] Claudiu Mosneang, Septimiu Mischie, Robert Pazsitka “Integrating the accelerometer of the AM335x Sitara Starter kit in a QT application,” 6th European Embedded Design in Education and Research Conference (EDERC), 2014 .

[5] “Qt3D,”[Online],available:http://qt.developpez.com/doc/5.0-snapshot/qt3d-reference/

[6] Casper van Donderen,"QCOMPARE(HTML5, QML) A comparison of UI development technologies," Hogeschool Utrecht,2010.

[7] "QtQuick3D,"[Online],available:http://doc.qt.digia.com/qt-quick3d-snapshot/

[8] "Modbus,"[online],available: //www.modbus.org/

[9] Lauri Paimen, Pietu Pohjalainen, "Case Study: QML for the Web," 13th IEEE International Symposium on Web Systems Evolution (WSE), 2011

[10] Knut Meissner, Prof. Dr. Ing. Hartmut Hensel, "Design of a Generic 3D Visualization System for Process Control," Sixth International Conference on Computational Intelligence and Multimedia Applications, 2005.