VISUALIZATION OF NUCLEAR FUEL HANDLING SYSTEM

by

MADHUMITHA P 2009103563
RAMKISHORE S 2009103045
MAHESWARI T 2009103564

A project report submitted to the

FACULTY OF INFORMATION AND

COMMUNICATION ENGINEERING

in partial fulfillment of the requirements

for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ANNA UNIVERSITY, CHENNAI – 25

MAY 2013

CERTIFICATE

Certified that this project report titled **"VISUALIZATION OF NUCLEAR FUEL HANDLING SYSTEM "** is the *bonafide* work of **MADHUMITHA P (2009103563), RAMKISHORE S (2009103045), MAHESWARI T (2009103564)** who carried out the project work under my supervision, for the fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science and Engineering. Certified further that to the best of my knowledge, the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or an award was conferred on an earlier occasion on these or any other candidates.

Place: Chennai
Date: 13.05.2013

Dr. P.Geetha
Associate Professor,
Department of Computer Science and Engineering,
Anna University, Chennai – 25

COUNTERSIGNED

Head of the Department,
Department of Computer Science and Engineering,
Anna University, Chennai – 25

ACKNOWLEDGEMENTS

We express our deep gratitude to our guide, **Dr.P.Geetha** for guiding us through every phase of the project. We appreciate her thoroughness, tolerance and ability to share her knowledge with us. We thank her for being easily approachable and quite thoughtful. Apart from adding her own input, she has encouraged us to think on our own and give form to our thoughts. We owe her for harnessing our potential and bringing out the best in us. Without her immense support through every step of the way, we could never have it to this extent.

We are extremely grateful to **Dr.C.Chellappan**, Head of the Department of Computer Science and Engineering, Anna University, Chennai – 25, for extending the facilities of the Department towards our project and for his unstinting support.

We express our thanks to the panel of reviewers **Dr.A.P.Shanthi, Dr.V.Mary Anita Rajam and Dr.S.Chitrakala** for their valuable suggestions and critical reviews throughout the course of our project.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

Madhumitha P **Ramkishore S** **Maheswari T**

ABSTRACT

In this project, we have developed an application for Visualization of Nuclear Fuel Handling System using Stereoscopy for creating the illusion of depth in the background image. All the required algorithms have been parallelized using OpenCL to inculcate parallel programming from the basic level.

We make use of the algorithm for variable depth mapping of real time stereoscopic image for 3D displays. It differs from the multi-region algorithm initially proposed by N.Holliman with respect to the distance that is considered between the viewer and the object. This is computed using Laplacian Equation.

The primary purpose of introducing parallelism is to process the huge amount of real time data coming in from the 120 sensors related to the Nuclear Fuel Handling Transfer Arm. We have parallelized all the algorithms that have been used in this application with the help of OpenCL. Compared with the conventional CPU-based implementation, the proposed method reaches extensive speedup. Experimental results show a great performance boost while utilizing both the GPU as well as the CPU.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

OpenCL Open Computing Language

OpenGL Open Graphics Library

AMD Advanced Micro Devices

AGP Accelerated Graphics Port

ATI Automatic Technology Inc.

FA Fuel Assembly

GF Gripper Finger

GH Gripper Hoist

GT Guide Tube

RTC Real Time Computer

SA Sub Assembly

SDC Synchro to Digital Converter

TA Transfer Arm

TS Top Structure

VFD Variable Frequency Drive

DDCS Distributed Digital Control System

CHAPTER 1

INTRODUCTION

This chapter deals with the domain of our problem, scope of the project and the organization of thesis. We have applied parallel programming to our application so that it could improve the speedup. For parallel programming we have made use of OpenCL which is platform-independent. Stereoscopic technique has been applied in our application to get better quality of visualization of the system.

1.1 PROBLEM DOMAIN

1.1.1 Stereoscopy 3D Imaging

Stereoscopy creates the illusion of three-dimensional depth from images on a two-dimensional plane. It is used to enhance the illusion of depth in a photograph or other two-dimensional images by presenting a slightly different image to each eye, and thereby combining both of the 2D offset images in the brain to give the perception of 3D depth.

The algorithm to generate stereoscopic image is based on the fact that human eye cannot detect the depth difference between two objects if they are located beyond 350m from the viewer. So it is useless to consider the scene depth of object if its distance is very far from the viewer. This is the reason why we find the relationship and optimal mapping between scene depth and perceived depth for the best implementation of stereoscopic image.

This mapping between scene depth and perceived depth is non linear and is based on the Laplace equation which says Light intensity is inversely proportional to R^2, where R denotes distance of object from the viewer.

This mapping is achieved through nonlinear scaling along the three axes of the object by considering Laplace equation expressed as follows

$$x_v = x_c / R^2$$
$$y_v = y_c / R^2$$
$$z_v = z_c / R^2$$

where

x_c, y_c, z_c denotes the coordinates of the objects in camera space
R denote the distance of the object from the camera system
x_v, y_v, z_v is the virtual coordinate system

on which all the regular image processing is done for the corresponding 2D image. The main key point of 3-axis scaling is expanding or compression along each axis to maintain the same relation between human eyes and the object in both the coordinate system. [1]

1.1.2 Parallel Programming

Open Computing Language (OpenCL) is a fundamental technology for cross-platform parallel programming. The emerging of OpenCL provides portable and efficient access to the power of modern processors. In contrast to other parallel programming models, OpenCL is platform-independent, vendor- and hardware-independent, consisting of an API for coordinating parallel computation across heterogeneous processors and a cross-platform programming language with a well specified computation environment. Basic OpenCL structure consists of two parts: host program and kernels.

- Host program queries available compute devices and creates corresponding running contexts.
- Through issuing commands to command-queues by the host, kernels are invoked and computations will be performed on OpenCL devices.[2]

1.2 SCOPE OF THE PROJECT

The scope of the project is to provide better visualization of the Nuclear Fuel Handling System so that the operator who constantly stares at the animation could easily understand what is happening inside the working room of Nuclear Fuel Handling System. For better visualization we are making use of Stereoscopy and the speedup is improved by using parallel programming.

1.3 ORGANIZATION OF THE REPORT

The thesis is organized as follows: Chapter 1 gives a general introduction of the project. Chapter 4 explains the functional and non-functional requirements of the project. Chapter 5 describes the system architecture and module description. Chapter 6 is about the development of the project. Chapter 7 deals with the performance studies and analysis done in project and the results established by means of the project. Chapter 8 gives the overall conclusion of the project.

CHAPTER 2

BACKGROUND

2.1 GENERAL PURPOSE GPU PROGRAMMING

Graphical Processing Units (GPU) are specialized processors designed to reduce the workload on the Central Processing Units (CPU) during compute intensive tasks, for processing parallel tasks and calculating floating point operations. Multicore CPUs are able to achieve some degree of parallelism by multi-threading concept compared to single core CPUs. GPUs can achieve high degree of parallelism as they are designed with multiple streaming multiprocessors each having multiple cores. Architecture diagram of our GPU (ATI Mobility Radeon) is shown in figure 2.1.

2.2 OPENCL MEMORY MODEL

Open Computing Language (OpenCL) is an open royalty-free standard for general purpose parallel programming across CPUs, GPUs and other processors. It was publicly released by Khronos Group in December 2008 and has been developing very fast with the cooperation of industry leaders, such as Apple, NVIDIA and AMD.
In contrast to other parallel programming models, OpenCL is platform-, vendor- and hardware-independent, consisting of an API for coordinating parallel computation across heterogeneous processors and a cross-platform programming language with a well specified computation environment. The following hierarchy of models is used:

- Platform Model : OpenCL Platform model consists of a host connected to one or more OpenCL devices.

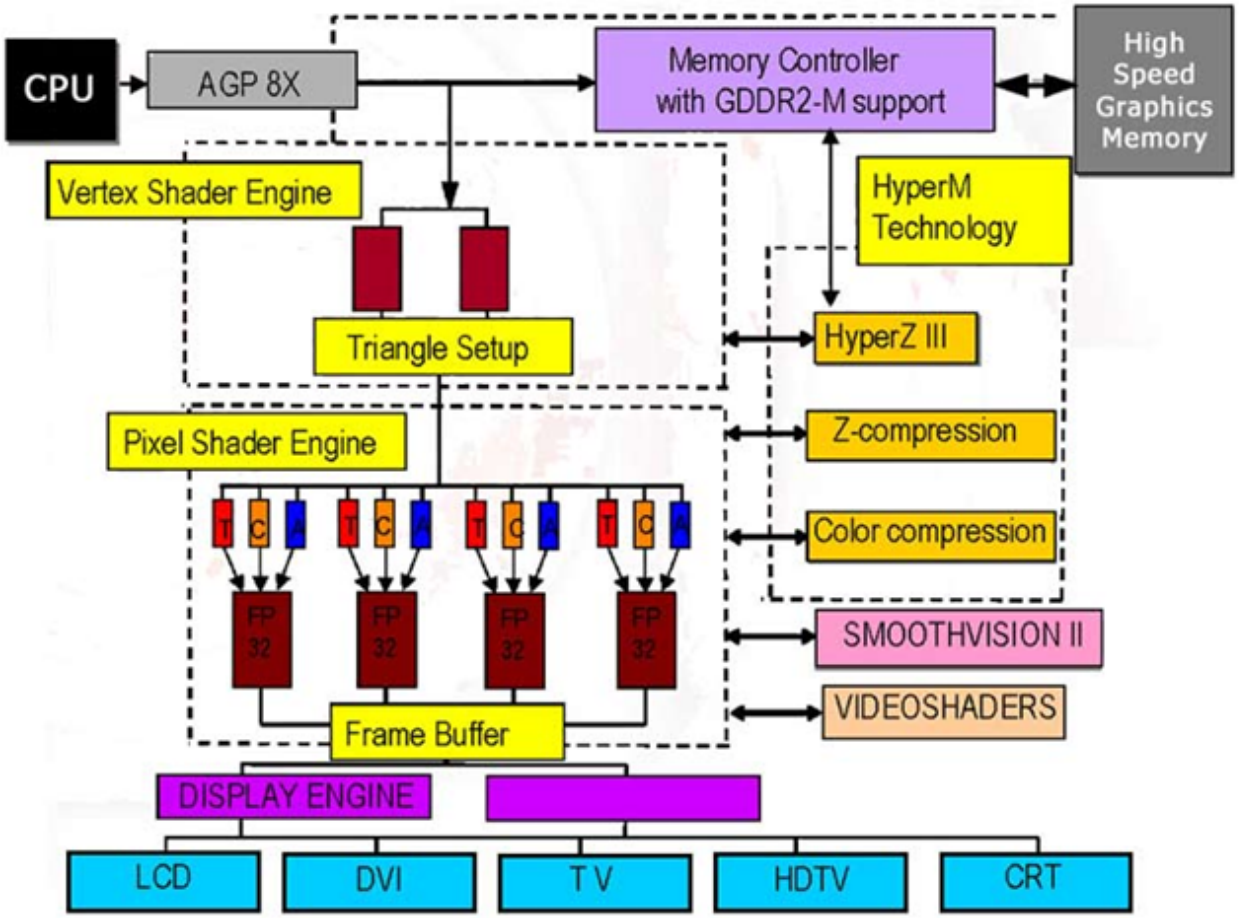

Figure 2.1 Architecture Diagram of ATI Mobility Radeon

- Execution Model: Execution of OpenCL program include two parts: kernels execute on OpenCL devices and a host program executes on the host.
- Memory Model: Four distinct memory regions are defined: global memory, constant memory, local memory and private memory.
- Programming Model: Both data parallel and task parallel programming models are supported.

Basic OpenCL structure is shown in figure 2.2. There are two parts: host program and kernels. Host program queries available compute devices and creates corresponding running contexts. Through issuing commands to command-queues by the host, kernels are invoked and computations will be performed on OpenCL

devices.

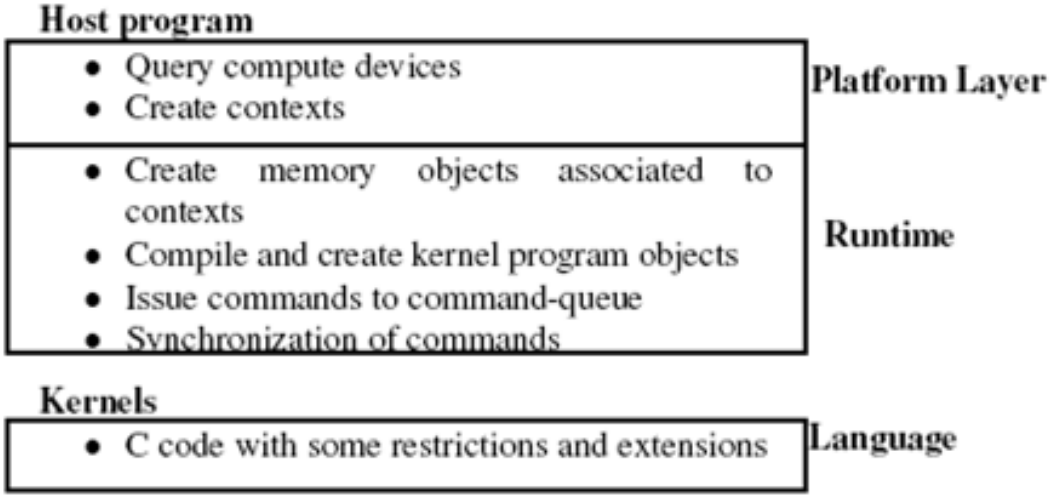

Figure 2.2 Basic OpenCL Structure

As shown in figure 2.3, kernels are executed under data parallel model in an N-dimensional index space (N is one, two or three). Each point in the index space is a single kernel instance, called a work-item. And work-items are further organized into work-groups. Kernel executions are managed by host program and coordinated through command-queue. Kernels are the basic functions executed on OpenCL devices.

2.3 STEREOSCOPY ALGORITHM

There has been many researches for the mapping from the scene depth to image depth. N. Holliman , one of the famous researchers in this research area proposed an algorithm called multi-region algorithm. In this algorithm we utilize the famous physical law called Laplacian equation expressed by

$$\nabla^2 V(x,y,z)=0$$

The physical meaning of above Laplacian equation is that the brightness function V (x, y, z) at a point P in a space must satisfy the above equation under the assumption that the light source does not exists at a point P. Usually the light

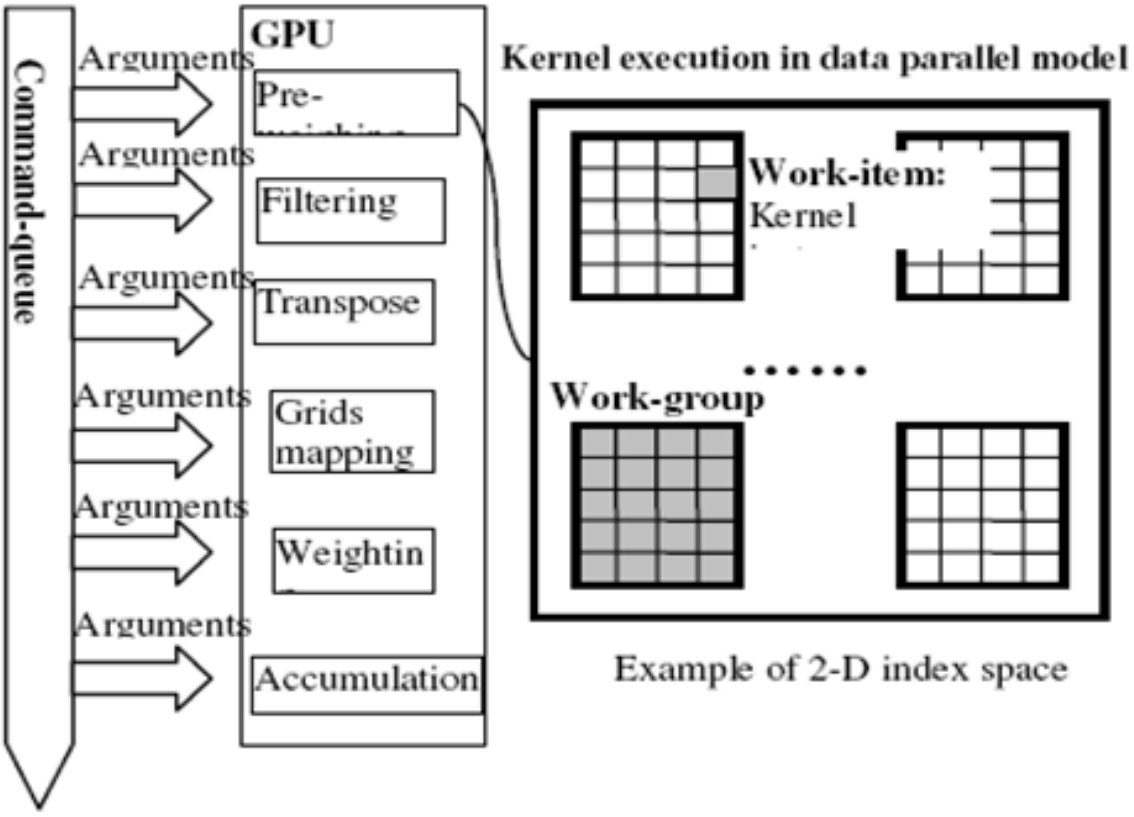

Figure 2.3 Illustration of Kernel Execution

sources are located far from the object on which our eyes are focusing. From the Laplacian equation, we can obtain averaging method expressed as follow for lightness function V (x, y, z).

$$V(x,y,z)= 1/8 \sum_{i=-1,1} \sum_{j=-1,1} \sum_{k=-1,1} V(x-i,y-j,z-k)$$

We can obtain following fact from the physical meaning of Laplacian equation, when the light sources are located near the viewer and far from the objects we are focusing on.

Fact 1: Light intensity is inversely proportional to R^2, where R denotes distance of object from the viewer.

Fact 2: When we process the 3D vision image, we must consider a nonlinear mapping between scene depth and perceived depth. Non linear mapping between scene depth and perceived depth can be obtained as follows:

$$d = d_1 / \tan\theta$$
$$d_1 = 0.03[m]$$
$$\theta_o = \tan^{-1}(0.200/0.03)$$
$$\triangle\theta = 0.0049[deg]$$

From the above fact, it may be better to preprocess nonlinear mapping between scene depth and perceive depth and then we process all the regular imaging processing introduced by N. Holliman, such as camera frustrum at al. We recommend a nonlinear scaling of the zcomponent of the object as a nonlinear mapping by considering Laplacian equation expressed as follow. We convert the camera co-ordinates into virtual co-ordinates using the formula:

$$x_v = x_c / R^2$$
$$y_v = y_c / R^2$$
$$z_v = z_c / R^2$$

where

x_c, y_c, z_c denotes the coordinates of the objects in camera space

R denote the distance of the object from the camera system

x_v, y_v, z_v is the virtual coordinate system

Figure 2.4 shows the relation between perceived depth for the same resolution of retina of human eye and distance under the assumption that the unit of x-axis denotes angle of retina on which an object is mapped

2.4 REAL-TIME DATA

We have obtained the real-time data required for our application from IGCAR (Indira Gandhi Centre for Atomic Research).

IGCAR is a premier atomic research centre of India. The centre is engaged in broad based multidisciplinary programme of scientific research and advanced

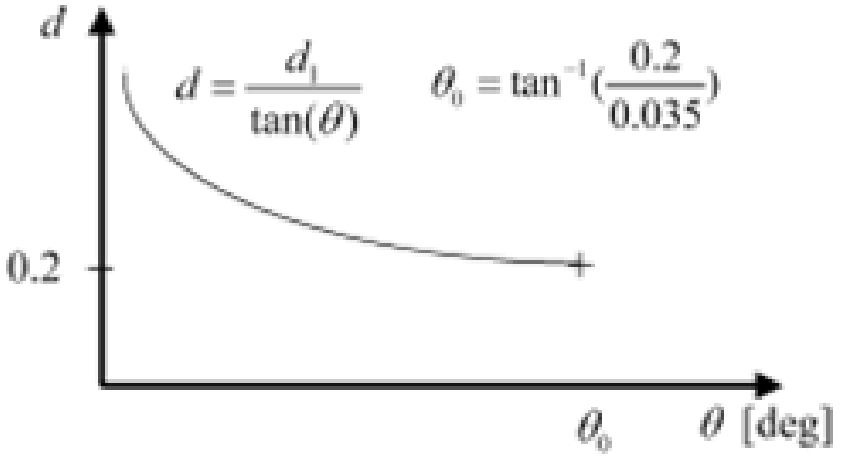

Figure 2.4 Perceived Equi-Depth Surface

engineering directed towards the development of Fast Breeder Reactor technology. The Reactor Research Centre set up at Kalpakkam, India, 80 km south of Chennai in 1971 under the Department of Atomic Energy (DAE) was renamed Indira Gandhi Center for Atomic Research (IGCAR) in 1985. The Reactor Research Centre was established in 1971. The Central Workshop, Safety Research Laboratory and Materials Sciences Laboratory were constructed in 19751976. Soon, the Radio-Chemistry Lab, and Electronics and Instrumentation Lab were constructed.

There are two reactors at IGCAR:

FBTR is a liquid metal cooled fast breeder reactor aerated and maintained by Reactor Operation and Maintenance Group (ROMG). In addition the Research Facility also built the 100MWe reactor for India's first nuclear submarine the Arihant class submarine project and operated it on land for testing purposes since it attained criticality in December 2004. The submarine to be launched on 26 July 2009 has been fitted with this reactor.

Prototype Fast Breeder Reactor, a 500MWe fast breeder reactor using Mixed Oxide (Uranium oxide + Plutonium oxide) fuel based on the Sodium-cooled fast reactor design.

The following chapter deals with the Literature survey of our system.

CHAPTER 3

RELATED WORK

This chapter deals with the literature survey for our system.

Stereoscopy

Based on the papers researched on Stereoscopy, the following are the latest algorithms:

The approach to solve the problem in stereoscopic image which we adopt in this paper is similar to N.Holliman multi-region algorithm.The most efficient algorithm is based on Laplacian equation which discusses the mapping of the depth from scene to displayed image by considering the distance between viewer and object. [1]

The second algorithm combines two different images into single one. The first of these images has cyan tint, while the second has red. The user wears a simple anaglyph glass, which consists of red and cyan lenses.User wears a simple anaglyph glass, which consists of red and cyan lenses. [4]

The third algorithm delivers different images to the users eyes by rendering two different images. These images are then sent to some of the specialized devices, such as three-dimensional projectors. The two input images are combined and displayed on the screen. Usually the user needs polarized glasses to feel the depth of the rendered scene. [4]

OpenCL

Open Computing Language (OpenCL) is a fundamental technology for cross-platform parallel programming that provides portable and efficient access to

the power of modern processors. Compared to the conventional CPU-based implementation, the proposed method reaches over 57 times speedup with respect to some applications. [2]

Heterogeneous computing system increases the performance of parallel computing in many domain of general purpose computing with CPU, GPU and other accelerators. Open Computing Language (OpenCL) is the first open, royalty free standard for heterogeneous computing on multi hardware platforms. In this paper, we propose a parallel Motion Estimation (ME) algorithm implemented using OpenCL and present several optimization strategies applied in our OpenCL implementation of the motion estimation.According to experiments, motion estimation algorithm achieves 100 to 150 speed-up on a platform of a CPU and a GPU while it is implemented using OpenCL. [3]

Personal computer and graph workstation with multiple CPUs are also impossible to process extremely large datasets. Therefore the challenge is to visualize the extremely large datasets efficiently. Visible Chinese Human datasets are so massive in size they require the use of parallel computing resources for effective visualization. At present a parallel visualization program has been developed based on parallelism visualization toolkit (pVTK) on high performance cluster to solve the problem. The visualization results and performance demonstrate that the parallel program developed provides a promising solution of handling extremely large datasets. [7]

Medical image dataset is so massive in size that it is often overload for common PC and even for graph workstation with multiple CPUs. It is an efficient way to visualize and simulate large medical datasets in parallel on high performance clusters of super-computers or multiple-node clusters of PCs. Thus Maximum Intensity Projection algorithm is realized in parallel. Experimental results demonstrate that this parallel technology can provide promising and real-time results. [8]

OpenMP parallelization of the matrix calculation yields an almost linear performance gain on the desktop, while for the cluster machine no significant

effect was observed. Thus a speedup factor of roughly 4.5 was achieved by parallelization of the Navier-Stokes equations for the blood flow. [9]

The Simplex algorithm is commonly used for solving Linear Optimization problems. Solving large optimization problems can be time consuming, which has to be taken into consideration for time-critical applications. With the invention of the GPU assisted computing the situation in this field has progressed.The maximum speedup is 28.9 times while running 10 iterations of the dual Simplex method and 10 iterations of the standard simplex method. [10].

The next chapter deals with the requirement analysis of the system.

CHAPTER 4

REQUIREMENTS ANALYSIS

This chapter gives an overall view about the requirements of the system. The requirements are specified in terms of functional and non-functional necessities.

4.1 FUNCTIONAL REQUIREMENTS

4.1.1 Real-Time Processing Requirements

The software will visualize the operations of the movement of Guide Tube, Gripper Hoist, Gripper Finger and Top Structure of Transfer Arm drives. In addition it has to control (turn ON/OFF) the oval shield plug heater. The various functions that are to be performed by the TA-RTC are described below.

4.1.1.1 Global Conditions for Transfer Arm Operation

Prior to do any operation on Transfer Arm the TA-RTC system has to check the global conditions to operate the Transfer Arm. The control system of Transfer Arm (TA-RTC) will take the inputs from TA and the various conditions will be checked.

4.1.1.2 Guide Tube Position Information

The total measurement range for Guide Tube is 5000mm (22000 - 27000 mm).The position of Guide Tube is measured by both potentiometer and Synchro. The 0-10V output from the GT potentiometer signal conditioner is taken as an analog input to TA-RTC and converted to 22000-27000 mm position. Similarly the GT synchro value is read through SDC card. The synchro rotates multiple turns during

the movement of GT and one rotation of synchro corresponds to 10mm of GT movement. From the SDC information, the multiple rotation count of synchro shaft is calculated and the GT synchro value is converted to 22000-27000 mm. The converted position is validated by checking whether it is $>=$ 22000 mm and $<=$ 27000 mm. The discrepancy between the GT potentiometer and GT synchro is checked and if the discrepancy is more than 10 mm, then movement of GT is stopped and an alarm message is generated. The position of GT is measured with an accuracy of ± 1 mm with synchro transmitter and ± 3 mm with potentiometer. The position information of GT and multiple rotation count will be sent through safety class-2 data highway to process computer for logging and display purposes.

4.1.1.3 Gripper Hoist Position Information

The total measurement range for Gripper Hoist is 5000mm (22000-27000 mm) .The position of Gripper Hoist is measured by both potentiometer and Synchro. The 0-10V output from the GH potentiometer-signal conditioner is taken as an analog input to TA-RTC and converted to 22000-27000 mm position. Similarly the GH synchro value is read through SDC card. The synchro rotates multiple turns during the movement of GH and one rotation of synchro corresponds to 2393.9 mm of GH movement. From the SDC information, the multiple rotation count of synchro shaft is calculated and the GH synchro value is converted to $22000-27000$ mm and the converted position is validated by checking whether it is $>=$ 22000 mm and $<=$ 27000 mm. Movement of GH is stopped if the discrepancy between the GH potentiometer and GH synchro is more than 10 mm and an alarm message is generated. The position of GH is measured with an accuracy of ± 1.63mm with synchro transmitter and ± 3.53mm with potentiometer. The position information of GH and multiple rotation count will be sent through safety class-2 data highway to process computer for logging and display purposes.

4.1.1.4 Top Structure Position Information

The position of Top Structure is measured by both Synchro and Optical encoder. Both Synchro and Optical encoder signals are read through the SDC and encoder interface card respectively. The synchro rotates multiple turns during the movement of TS and one rotation of synchro corresponds to 59.36^o of TS rotation. From the SDC information, the multiple rotation count of synchro shaft is calculated. Both syncrho and encoder values are converted to 0 to $\pm 90^o$ and the converted positions are validated by checking whether it is $>= -90^o$ and $<= 90^o$. Also discrepancy between the encoder and synchro is checked and if the discrepancy is more than 30 arc minutes, then movement of Top Structure is stopped and an alarm message is generated. The position information of TS and multiple rotation count will be sent through safety class-2 data highway to process computer for logging and display purposes.

4.1.1.5 Gripper Finger Position Information

The analog input (0V-10V) received from the signal conditioning unit of the Gripper Finger LVDT is converted to engineering unit of position 0 to 100 mm by TA-RTC. The converted value is validated by checking whether it is $>= 0$ mm and $<= 100$ mm. The continuous movement of GF is measured with using LVDT with an accuracy of ± 0.1mm. The position information will be sent through safety class-2 data highway, for logging and display purposes, to process computer.

4.1.2 Parallel Programming Requirements

To this visualization application parallel programming is inculcated to bring speedup using OpenCL language. Following are the requirements for this application:

- Using platform and device layers to build robust OpenCL

- Program compilation and kernel objects
- Managing buffers
- Kernel execution
- Kernel programming Transformation, Rotation, Laplace Equation
- Kernel programming synchronization

4.1.2.1 AMD Catalyst

AMD Catalyst (formerly named ATI Catalyst) is a device driver and utility software package for ATI line of video cards. It runs on Microsoft Windows and Linux, on 32-bit and 64-bit x86 processors. Catalyst is a software suite that includes unified driver and software applications to enable [ATI's] Radeon family of graphics products. The Catalyst consists of the following elements:

- A new, unified driver for ATI Radeon graphics cards
- Hydravision, ATI's proprietary desktop management software
- An ATI "Multimedia Center"
- ATI's Remote Wonder software
- A new AGP diagnostic and stability tool
- A newly redesigned control panel

4.1.2.2 Visual C♯

Microsoft Visual C♯ is Microsoft's implementation of the C♯ specification, included in the Microsoft Visual Studio suite of products. This product is packaged with a graphical IDE and supports rapid application development of Windows-based applications.

Installation Steps:

The C♯ Express Edition is a free download from Microsoft. Download the Microsoft Visual C♯ 2008 Express Edition and follow the below steps to get it installed.

- Once the download is complete double click on the vcssetup.exe to launch the C♯ Express setup application.
- Click the checkbox to accept the terms of License Agreement and click the Next button.
- Connect to the Internet and choose the location to install C♯ Express Edition and press the Install button to proceed with the installation.
- Once the download is complete C♯ Express Edition will now begin installing which will take some time.
- C♯ Express Edition is now installed successfully in our computer.

4.1.2.3 OpenCL

Open Computing Language (OpenCL) is a framework for writing programs that execute across heterogeneous platforms consisting of central processing units (CPUs), graphics processing units (GPUs), DSPs and other processors. OpenCL includes a language (based on C99) for writing kernels (functions that execute on OpenCL devices), plus application programming interfaces (APIs) that are used to define and then control the platforms. OpenCL provides parallel computing using task-based and data-based parallelism. OpenCL is an open standard maintained by the non-profit technology consortium Khronos Group. It has been adopted by Intel, Advanced Micro Devices, Nvidia, and ARM Holdings. For example, OpenCL can be used to give an application access to a graphics processing unit for non-graphical computing (see general-purpose computing on graphics processing units).

4.1.2.4 OpenGL

Open Graphics Library(OpenGL) is a cross-language, multiplatform API for rendering 2D and 3D computer graphics. The API is typically used to interact with a GPU, to achieve hardware-accelerated rendering. OpenGL was developed by Silicon Graphics Inc. in 1992 and is widely used in CAD, virtual reality,

scientific visualization, information visualization, flight simulation, and video games. OpenGL is managed by the non-profit technology consortium Khronos Group. As well as being language-independent, OpenGL is also platform independent.

4.2 NON-FUNCTIONAL REQUIREMENTS

4.2.1 Stereoscopy Requirements

4.2.1.1 3D Color Anaglyph Glasses

Anaglyph 3D is the name given to the stereoscopic 3D effect achieved by means of encoding each eye's image using filters of different (usually chromatically opposite) colors, typically red and cyan. Anaglyph 3D images contain two differently filtered colored images, one for each eye. When viewed through the ”color-coded” ”anaglyph glasses”, each of the two images reaches one eye, revealing an integrated stereoscopic image. The visual cortex of the brain fuses this into perception of a three dimensional scene or composition.

The next chapter deals with the detailed design description of modules developed for our application.

CHAPTER 5

SYSTEM DESIGN

This chapter deals with the high level design and detailed design of our system.It also gives a brief description of each modules.

5.1 HIGH LEVEL DESIGN

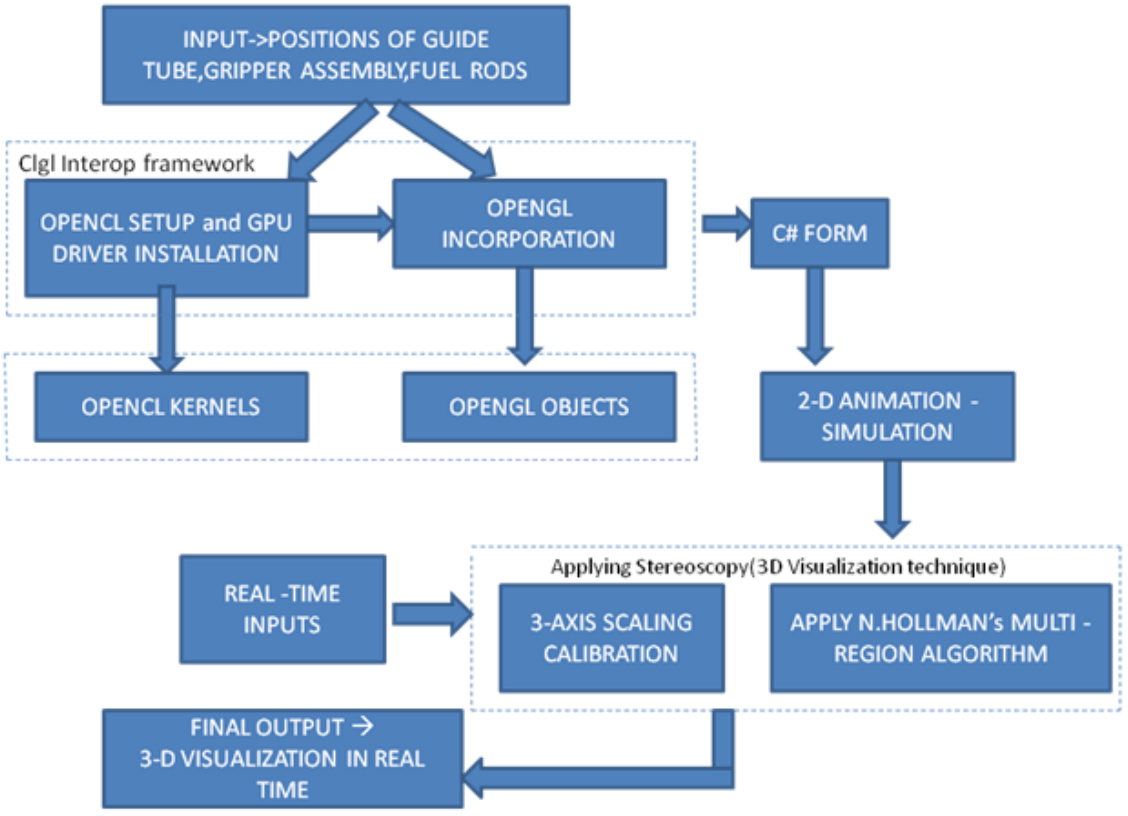

Figure 5.1 High level Design

The application takes real-time inputs from various sensors like potentiometer, motion sensors, position sensors etc based on which the application runs accordingly. The various real-time inputs are

FLOAT32 AI_Data[11]	44 Bytes	Analog Inputs of Float
FLOAT32 SDC_Data[3]	12 Bytes	Synchro Inputs of Float
FLOAT32 OEIC_Data[1]	4 Bytes	Encoder Inputs of Float
FLOAT32 AO_Data[3]	12 Bytes	Analog Inputs-Float
SINT16 MT_Count[3]	6 bytes	Synchro Multicount-Integer
UINT8 DI_Data[5][30]	150 bytes	Digital Inputs-Bytes
UINT8 Soft_Outputs[24]	24 Bytes	Soft Outputs- Bytes
UINT8 ROP_Readback[4][15]	60 bytes	Relay Outputs -60 Bytes
UINT8 Card_Healthy_Status[13]	13 bytes	Card Health Status-13

Table 5.1 Details of Real-Time Data

The scanned values, processed data status, alarm and coded messages of the system are sent to DDCS as an Ethernet packet through Ethernet port of CPU card of RTC system. TCP/IP protocol is used for this communication.
The CPU of RTCs has two ports with four sockets in each port for Ethernet communication. The data are sent through both the ports. If anyone port fails, the communication failure is indicated in the four-character display of the respective CPU card and RTC stops sending healthy status to SOLS. The system continues to send the data through the healthy port till confirming the communication error for 3 cycles. When both the ports fail the system comes to halt. The application is built by the creation of OpenGL Objects in C♯ and the speedup is improved by parallelizing the transformation algorithm as OpenCL Kernels.
Reference model is constructed with the various imported methods and models to show that the run time performance of our visualization application using parallel programming is found to be more efficient.

5.2 DETAILED DESIGN

5.2.1 Use Case Diagram

The use case of our application is shown in figure 5.2. Figure 5.2 describes the actors involved in our system and the various functions that the actors in the system

Figure 5.2 Use case diagram for the System

perform are mentioned as use case.The actors for our system include the Operator and Transfer Arm Application.

5.2.2 Class Diagram

The class diagram for the overall system is shown in figure 5.3. Figure 5.3 gives the actual program design that is needed to perform the functions. It consists of the following classes:

- OpenCL Algorithm

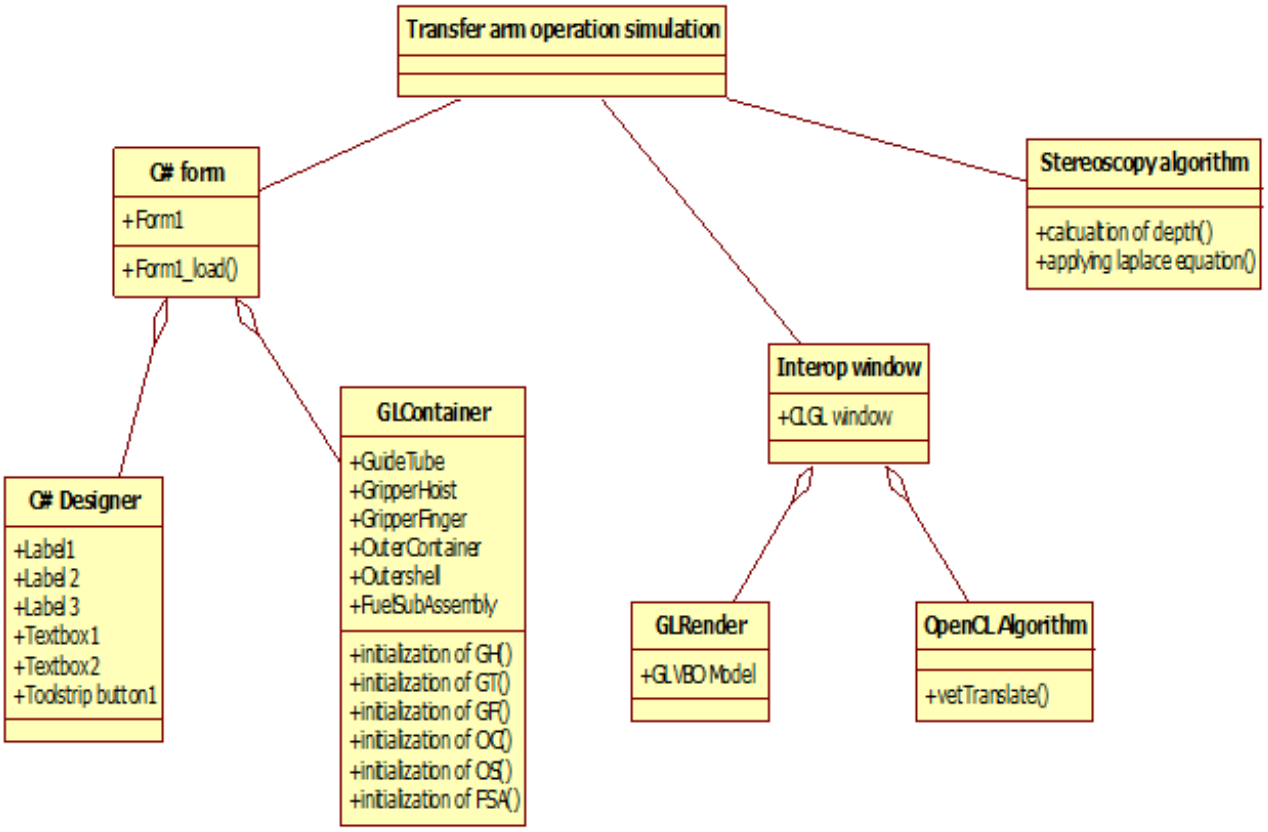

Figure 5.3 Class diagram for the System

- GLRender
- GL Container
- Stereoscopy algorithm
- Interop window

Interop window provides a framework so that we could interconnect both OpenCL and OpenGL.OpenGL objects are created using C♯.Various transformation algorithms have been coded as OpenCL kernels to improve the speedup.

5.2.3 Sequence Diagram

The sequence of operations that happens in our application is shown in the sequence diagram i.e. figure 5.4.

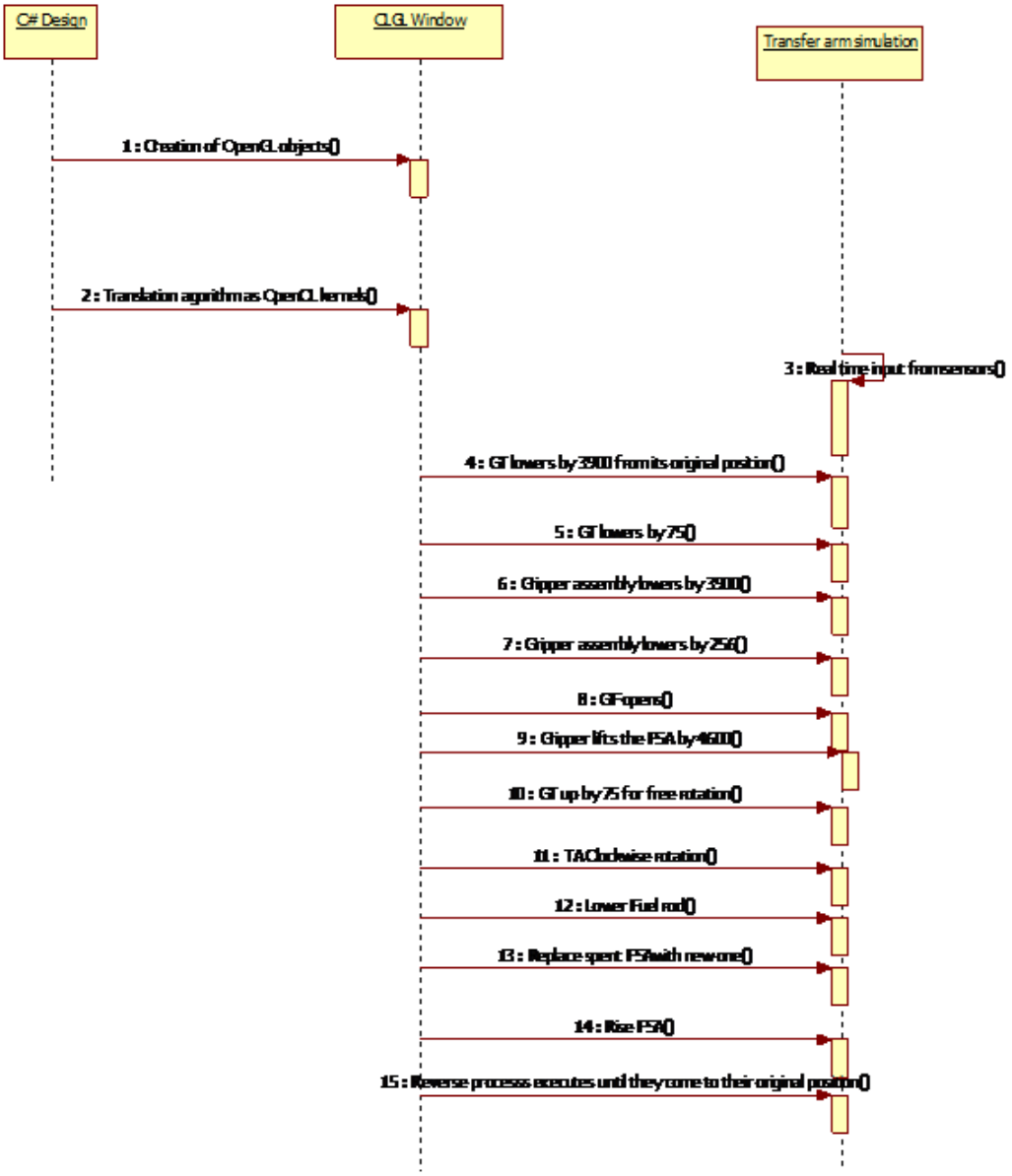

Figure 5.4 Sequence diagram for Overall System

Figure 5.4 is the overall sequence diagram of the system. It depicts the flow of events from the initialization to the visualization which is seen by the operator.

5.3 MODULES DESCRIPTION

We have split our application into three modules.It includes the following:

- Visualization of Fuel Handling System
- Stereoscopic Technique
- Real-Time Visualization of Fuel Handling System

5.3.1 Visualization of Fuel Handling System

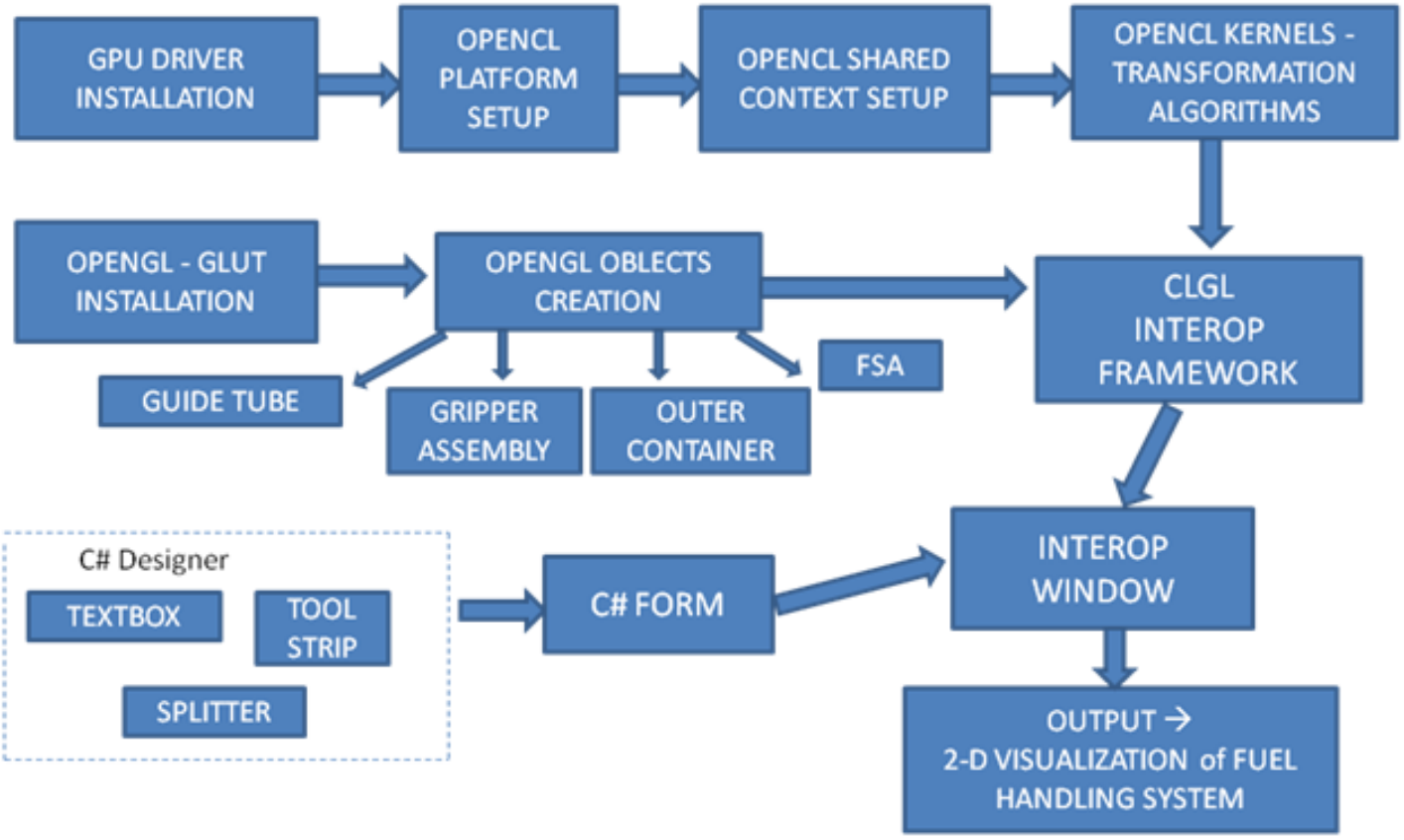

Figure 5.5 Module Diagram for 2D Visualization of Fuel Handling System

The above architecture takes the positions of Guide Tube, Gripper Hoist and the Fuel Assemblies as input. Initially, OpenCL platforms, shared contexts and kernels are set up. Then OpenGL glut installation is done and the objects for Guide Tube, Gripper Hoist and the Fuel Assemblies are created. The C♯ designer form with the splitter, tool strip etc are also created. The output of this module is the 2D visualization of the Nuclear Fuel Handling System.

5.3.2 Stereoscopic Technique

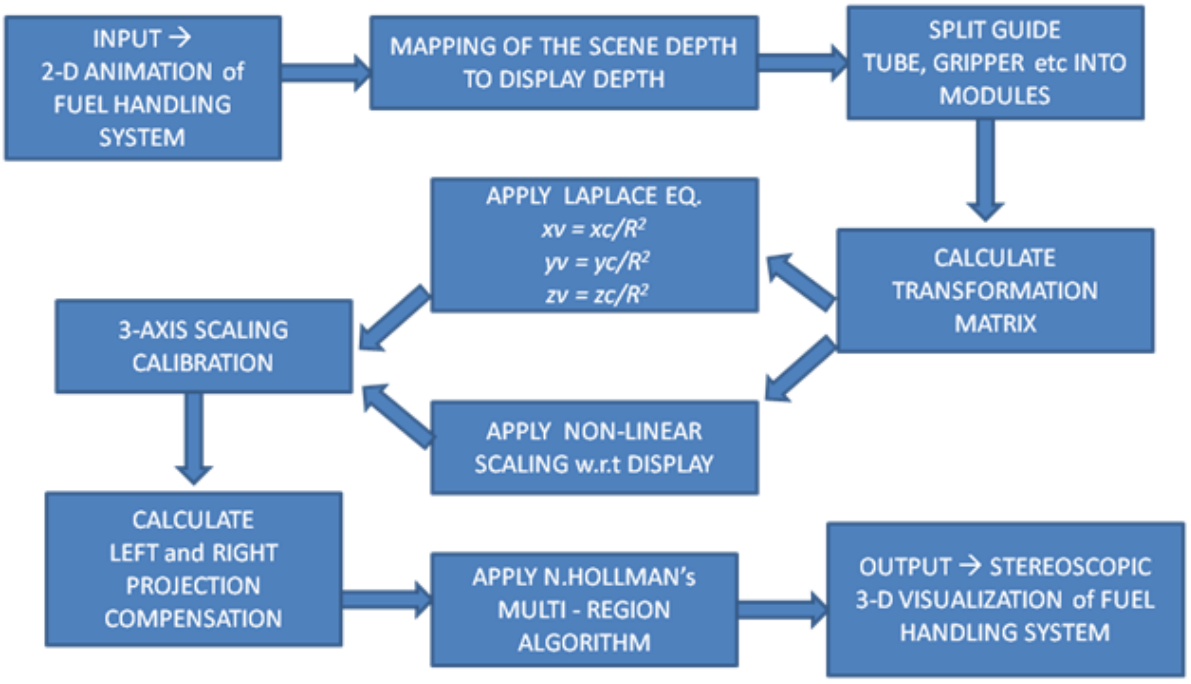

Figure 5.6 Module Diagram for Stereoscopy

The above architecture takes the 2D animation of Nuclear Fuel Handling System as input. Then we have applied the Laplace equation in order to map the camera co-ordinates into virtual co-ordinates. The Hollimans algorithm is then applied to calculate the brightness function. The final output of this module is the Stereoscopic 3D visualization of Nuclear Fuel Handling System.

5.3.3 Real-Time Visualization of Fuel Handling System

The architecture shown below takes input from the real-time sensors like potentiometers, motion sensors and position sensors which is converted to required format and then applied to the Stereoscopic visualization of Nuclear Fuel Handling System. The OpenCL optimizations are applied and the final output of visualization of Fuel Handling System developed.

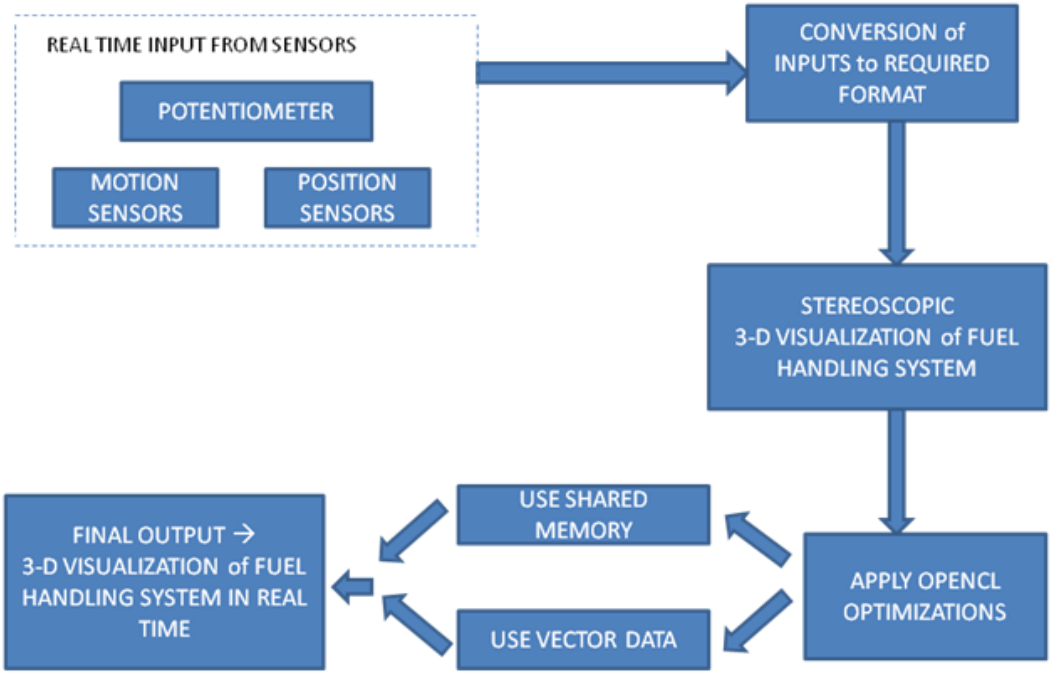

Figure 5.7 Module Diagram for Real-Time Visualization of Fuel Handling System

CHAPTER 6

SYSTEM DEVELOPMENT

6.1 TOOLS DESCRIPTION

6.1.1 AMD Catalyst Driver

AMD Catalyst (formerly named ATI Catalyst) is a device driver and utility software package for ATI line of video cards. It runs on Microsoft Windows and Linux, on 32-bit and 64-bit x86 processors. Catalyst was instituted with version 02.1 after the release of the Radeon 8500, as ”a software suite that includes unified driver and software applications to enable [ATI’s] Radeon family of graphics products” for Windows XP, Windows 2000 and Windows 7. The original Catalyst consisted of these elements:

- A new, unified driver for ATI Radeon graphics cards
- Hydravision, ATI’s proprietary desktop management software
- An ATI ”Multimedia Center”
- ATI’s Remote Wonder software
- A new AGP diagnostic and stability tool
- A newly redesigned control panel

6.1.2 ATI Mobility Radeon

Radeon is a brand of Graphics Processing Units and random-access memory produced by Advanced Micro Devices. The brand was launched in 2000 by ATI Technologies, which was acquired by AMD in 2006. Radeon is the successor to the Rage line. There are four different groups, which can be differentiated by the DirectX generation they support.

6.1.3 Microsoft Visual C♯ 2008 Express Edition

Microsoft Visual C♯ is Microsoft's implementation of the C♯ specification, included in the Microsoft Visual Studio suite of products. It is based on the ECMA/ISO specification of the C♯ language, which Microsoft also created. This product is packaged with a Graphical IDE and supports rapid application development of Windows based applications.

6.2 IMPLEMENTATION PLATFORM

Software Requirement
Operating system : Windows 7
Tools :Microsoft Visual C ♯ 2008 Express Edition, Graphics card GPU ATI Mobility Radeon, AMD Catalyst Driver.
Languages :OpenGL in C♯, OpenCL

Hardware Requirement
PC/Laptop with the above specified software configuration.

6.3 DEVELOPING THE APPLICATION

In order to develop our application,we mainly need a Graphics card and Microsoft Visual C♯ to be installed in our system and its installation procedure is given below.

6.3.1 ATI Graphics Card

The installation steps of ATI Graphics Card is as follows:

- Graphics card GPU ATI Mobility Radeon 4500-5000 which is the primary requirement on our system for this application needs to be installed.

- AMD Catalyst Driver (Graphics Card Driver) needs to be installed with OpenCL specification1.0.
- Install the OpenCLGL Interop framework by AMD
- Set up the device, context and OpenCL buffers.
- Compile and run a sample OpenCL kernel.

6.3.2 Microsoft Visual C♯ 2008 Express Edition

A simulation in Visual C♯ is done by creating a new project. The other steps include:

- Download the vcssetup.exe and install in our system.
- Download the required OpenGL glut files.
- Set up the environment in Visual C♯.
- Club the OpenCL and OpenGL and compile a sample OpenCL program in Visual C♯ environment.
- Design a C♯ form and create the various OpenGL objects required.

6.4 WORKING OF THE PROJECT

Since the working room of Nuclear Fuel Handling System is highly radio-active it is dangerous for the operator to go into the room and monitor its functions. Hence in order to make it easier for him to visualize we have developed this application. Since the system has to perform without any drag while receiving large amount of data from sensors we have parallelized the algorithm so that it could improve the speedup. Initially the Guide Tube (GT), Gripper Hoist (GH), Gripper Finger (GF), Outer Container (OC), Nuclear Fuel Rods and various static objects are created using OpenGL in C♯ and are kept in their original position say GT is at a distance of 25000 from fuel rods, GH is at a distance of 25000. On starting the simulation, the position of Guide Tube (GT), Gripper Hoist (GH), Gripper Finger (GF), Nuclear Fuel Rods varies according to the input from various sensors like potentiometer, motion sensors, and position sensors. When the GT reaches a

particular height say 3900, GF opens and lifts the spent Fuel Sub Assembly (FSA) and replaces it with the new FSA. And then the GF closes which is followed by the movement of GT, GH back to their original position. For a better perception of the depth in the application we have applied Stereoscopic algorithm which is based on the Laplacian equation.

We have developed a reference visualization application using various imported models and methods available in OpenGL in order to show that the run time performance of our visualization application using parallel programming is found to be more efficient.

The sequence of operations of Transfer Arm is listed below:

1.Guide Tube is lowered from Parking position (26400 mm) to transfer position (22400 m→ **Lowering Guide Tube in Parking mode**

2.SRP/LRP to be rotated to predetermined angle as programmed in process computer.

3.TA top structure is rotated to a predetermined angle → **Rotation of Top Structure in Empty Transfer Mode**

4.SA selector switch is positioned for CSA/FSA.

5.Guide Tube is lowered from transfer position to 30 mm above SA position (22430 mm)→ **Lowering Guide Tube in SA handling Mode**

6.Gripper Hoist is unlocked.

7.Gripper Hoist with fingers in closed condition is lowered into head of SA→ **Lowering Gripper Hoist in Empty Transfer Mode**

8.Gripper Fingers are opened on the head of SA→ **Opening Gripper Fingers in Loaded Transfer Mode**

9.Gripper Hoist is raised with the sub assembly to loaded gripper position (26850 mm)→ **Raising Gripper Hoist in Loaded Transfer Mode**

10.Gripper Hoist is locked

11.Guide Tube is raised from 30mm above SA (22430 mm) to transfer position (22500 mm)→ **Raising Guide Tube in SA Handling Mode**

12.SRP/LRP,TA top structure rotated by a pre determined angle where the SA has to be deposited→ **Rotation of Top Structure in Loaded Transfer Mode**

13.Guide Tube is lowered from transfer position (22500 mm) to 30 mm above SA

position (22430 mm)→ **Lowering Guide Tube in SA handling Mode**
14.Gripper Hoist is unlocked Gripper Hoist along with the SA is lowered stopped by low tension limit switch→ **Lowering Gripper Hoist in Loaded Transfer Mode**
15.Gripper Fingers are closed to release the SA → **Closing Gripper Fingers in Loaded Transfer Mode**
16.Gripper Hoist is raised to 22500 (100 mm above core top)→ **Raising Gripper Hoist in Empty Transfer Mode**
17.Gripper Fingers opened empty → **Opening Gripper Fingers in Seating Checkup Mode**
18.Gripper Hoist lowered over head of SA till low tension occurs.Confirm proper seating→ **Lowering Gripper Hoist in Seating Checkup Mode**
19.Gripper Hoist is raised to 22500 mm → **Raising Gripper Hoist in Seating Checkup Mode**
20.Gripper fingers are closed → **Closing Gripper Fingers in Seating Checkup Mode**
21.Gripper Hoist is raised to 26400 mm (parking/empty transfer position)→**Raising Gripper Hoist in Empty Transfer Mode**
22.Gripper hoist is locked
23.Guide Tube is raised from 22430 mm to transfer position (22500 mm)→ **Raising Guide Tube in SA Handling Mode**
24.Top structure is rotated to reference position → **Rotation of Top Structure in Empty Transfer Mode**
25.Guide tube is raised from transfer position to parking position → **Raising Guide Tube in Parking Mode**
One SA transfer operation is completed with step 24. If still sub assemblies need to be handled then steps from 2 to 24 shall be followed.

The next chapter deals with the results we have arrived at after implementing our application in parallel programming.

CHAPTER 7

RESULTS AND DISCUSSION

In this chapter, we give an account of the results we have got from the GPU implementation. We discuss the conclusions we arrived at after analyzing the performance of the same, for various specifications.

7.1 VISUALIZATION MODELS CREATED

This visualization of the Nuclear Fuel Handling System has been created which consists of the whole process involving elimination of the spent fuel rods and replacing them with new fuel rods has been showed in figure 7.1.This application has been developed using C♯ and Parallel programming has been incorporated using OpenCL to improve the speedup.

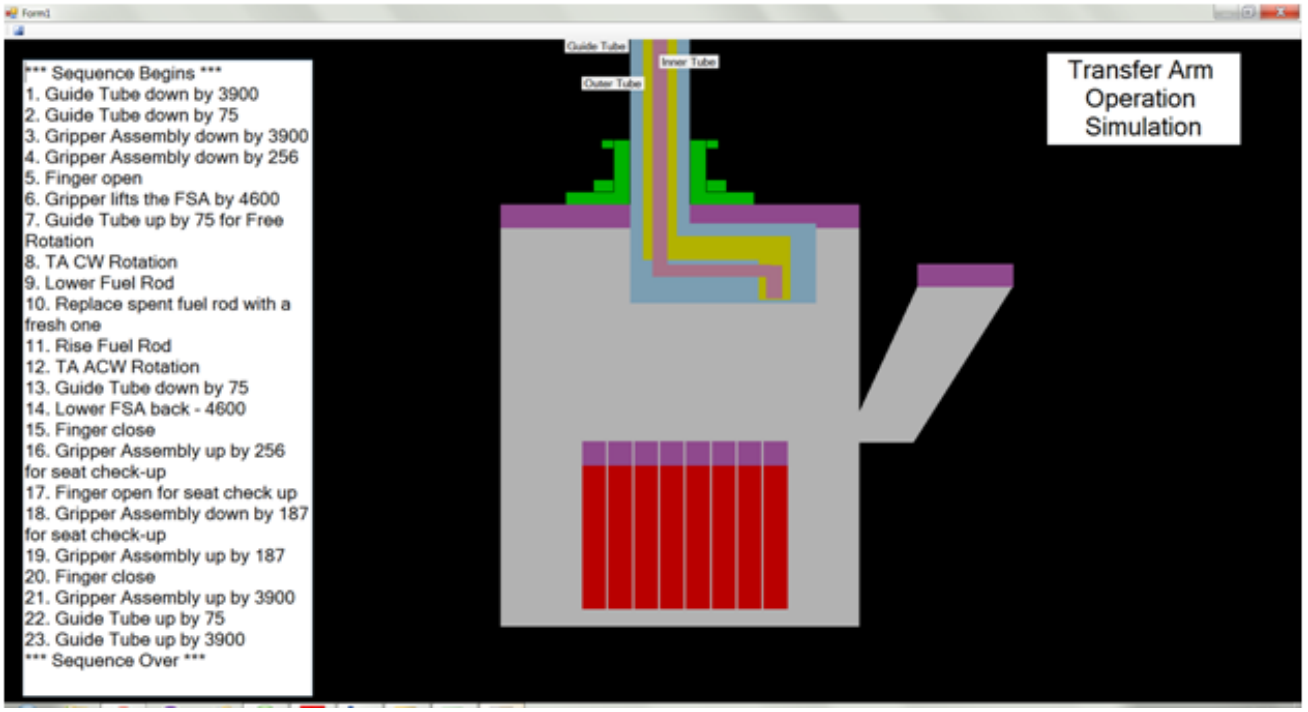

Figure 7.1 Visualization of the Fuel Handling System

A sample reference model of the Nuclear Fuel Handling System has been built using OpenGL glut functions as shown in figure 7.2.This model is created to show the improvement in speedup due to parallel programming.

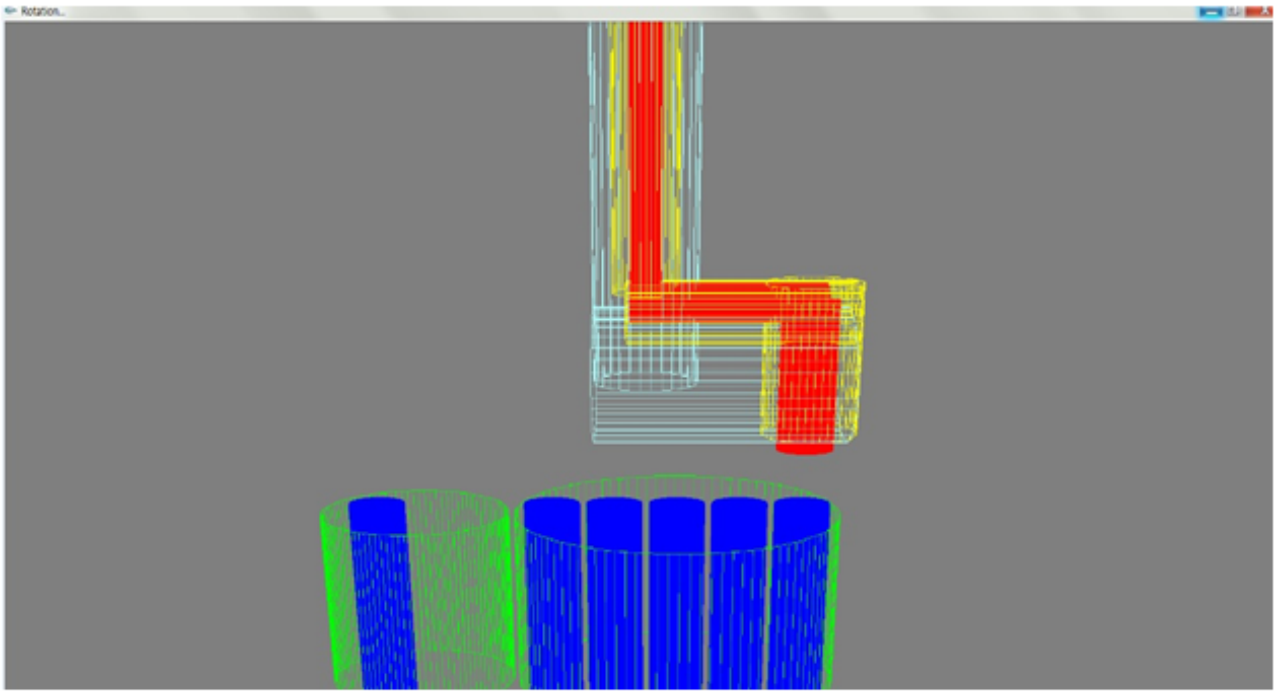

Figure 7.2 Reference model of the Fuel Handling System

7.2 STEREOSCOPY

An application has been built using C♯ to show the Stereoscopy effect of 3D visualization of the Nuclear Fuel Handling System. This technique is implemented based on the Laplacian Equation and N.Holliman multi region algorithm. Laplacian equation has been coded using OpenCL(parallel programming) to reduce the processing time.
The left and right eye perception of the images are provided to the application which combines it to a single image with the 3D depth perception that can be viewed with a 3D anaglyph glass as shown in the figure 7.3.

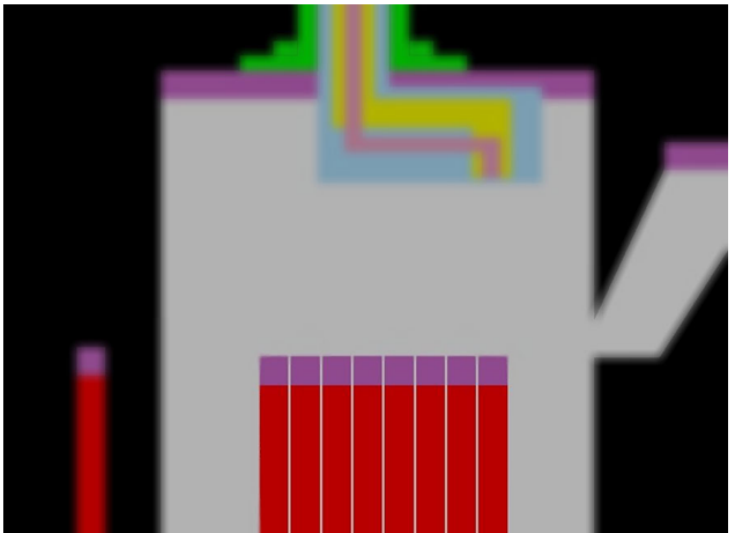

Figure 7.3 Stereoscopic image of the Fuel Handling System

```
FO2 = new GLRender.GLVBOModel(OpenTK.Graphics.OpenGL.BeginMoc
FO2.SetVertexData(fo1);
FO2.SetTexCoordData(texCoord);
FO2.SetNormalData(n);
FO2.SetElemData(elem);
```

Output

Show output from: Debug

```
'DemoCLGL.vshost.exe' (Managed): Loaded 'C:\Windows\assembly\GAC_MSIL\System.Configuration\2.0
WARNING! ComputeProgram(0) leaked.
RunTime 00:00:03.93
Created ComputeContext(560765472) in Thread(10).
Created ComputeCommandQueue(560798320) in Thread(10).
```

Figure 7.4 Processing time of application using Parallel Programming

7.3 SPEEDUP

The processing time of the visualization application built using parallel programming is 0.39 seconds while the processing time of the reference model which does not have parallel programming is 0.55 seconds as shown in figure 7.4 and 7.5. Thus a speedup of 0.16 seconds or 29% has been achieved in processing time without real-time data.

When real-time input data has been applied to both the applications the

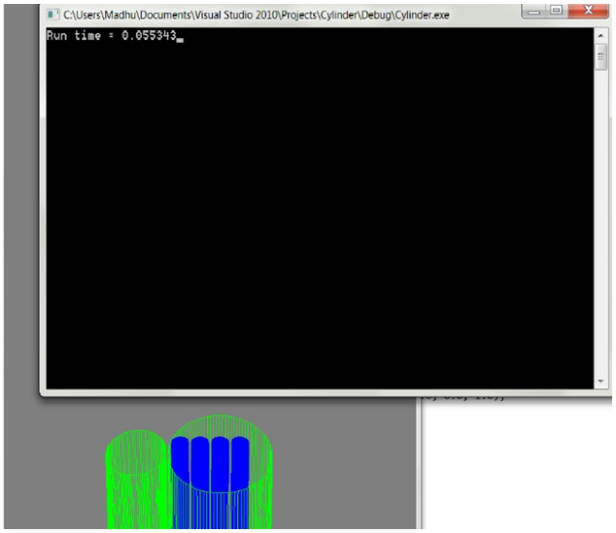

Figure 7.5 Processing time of the Reference Model without Parallel Programming

processing times are found to be 3.93 seconds for the visualization application built using parallel programming while the reference model which does not have parallel programming took 5.53 seconds as shown in figure 7.6 and figure 7.7.Thus a speedup of 1.6 seconds or 30% has been achieved in processing time after real-time data has been applied.

Figure 7.8 shows the speedup achieved due to optimizations. The x axis denotes the various stages of the application and the y axis denotes the process time (CPU + GPU) taken to complete each of those steps. The input considered for this performance speedup graph is real-time. Till GF opens the speedup has been improved by 10%.The next stage considered in the x axis is the lowering of the fuel rods. This step shows a speedup of 11%.The values taken after closing of the GF shows an increase of 11%.The next stage in the sequence of operations is the lowering of the Gripper Assembly. This step produces a speedup of 13%.The final stage considered is the GT getting back to the original position indicating the end of the process and the overall speedup achieved is 14%.

Figure 7.9 shows the speedup achieved due to parallel programming. The x axis denotes the various stages of the application and the y axis denotes the process time (CPU + GPU) taken to complete each of those steps. The input considered

Figure 7.6 Processing time of the application with Parallel Programming after Real-time data is applied

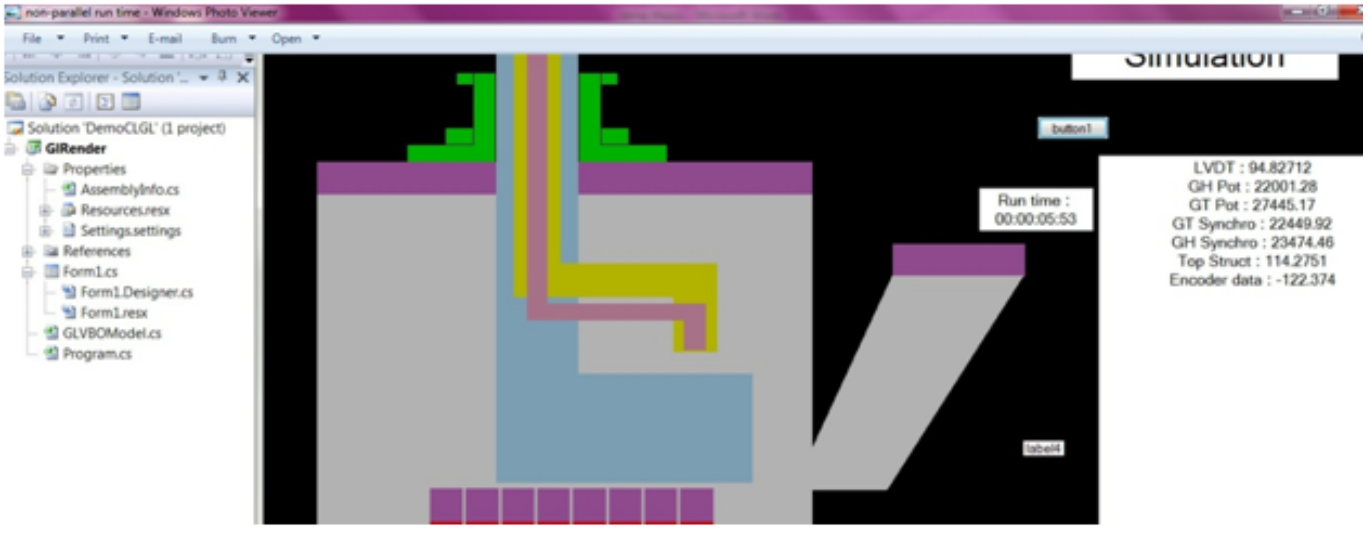

Figure 7.7 Processing time of the application without Parallel Programming after Real-time data is applied

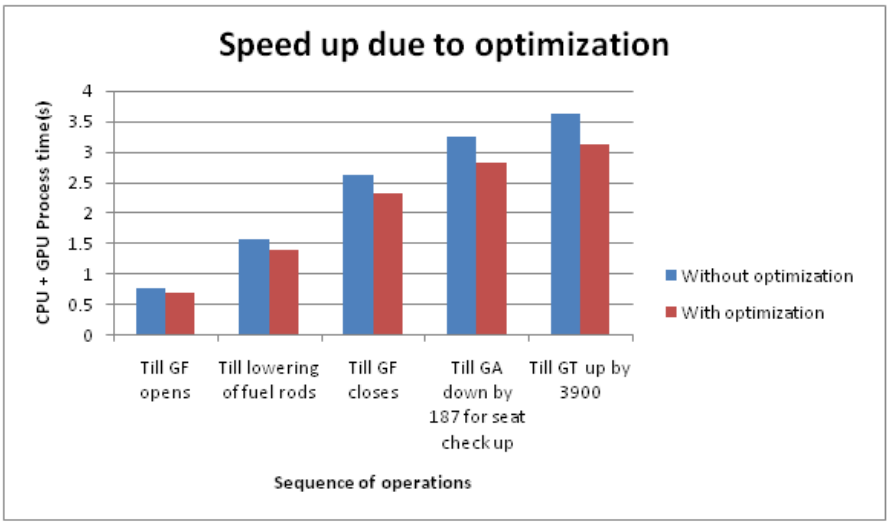

Figure 7.8 Speed up due to Optimizations in Real-time data

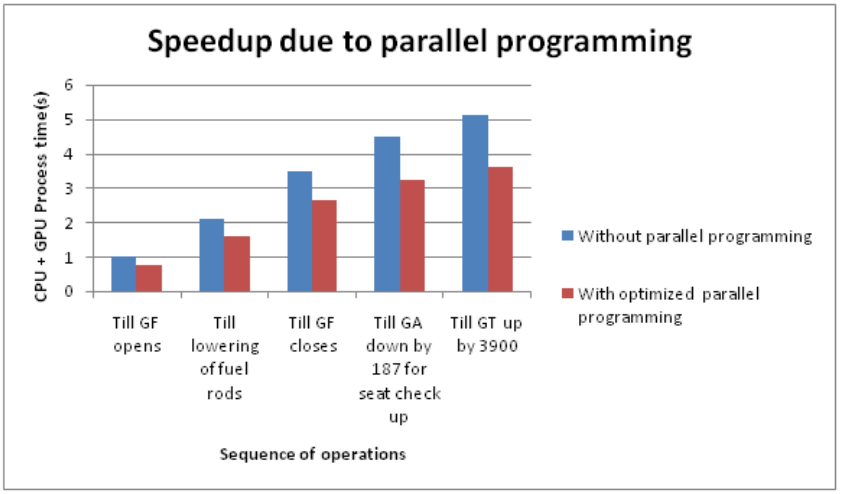

Figure 7.9 Speed up due to optimized Parallel Programming in Real-time

for this performance speedup graph is real-time. Till GF opens the speedup has been improved by 23%. The next stage considered in the x axis is the lowering of the fuel rods. This step shows a speedup of 25%. The values taken after closing of the GF shows an increase of 25%. The next stage in the sequence of operations is the lowering of the Gripper Assembly. This step produces a speedup of 28%. The final stage considered is the GT getting back to the original position indicating the end of the process and the overall speedup achieved is 29%.

CHAPTER 8

CONCLUSIONS

8.1 CONTRIBUTIONS

In this project we have animated the entire process that is taking place inside the Nuclear Fuel Handling System. Since the working room condition of the Nuclear Fuel Handling Transfer Arm is highly radio-active, a visualization mechanism is an absolute necessity. The earlier model was that of displaying values and data as such at the operator terminal. Our animation model will be extremely helpful for the operator at the Monitoring Station because he can comprehend the situation and take immediate action in case of emergencies. The real-time data relaying the information about the positions of GT, GH, GF, and FSA with the help of various sensors like motion sensors, position sensors, and potentiometer is the input to our visualization application.

A new smoothing algorithm for variable depth mapping for real-time stereoscopic Image has been implemented.[1] Stereoscopy, which is the relative perception of depth, gives a better perspective for understanding.

We have developed the application using parallel programming in order to improve the speedup as the amount of real-time data coming in from the sensors is huge and fast processing is critical. The application has been developed without using any imported methods or models. Therefore, when it is tested under real-time conditions, both the CPU and GPU are utilized and there is no drag in the visualization. The performance evaluation has been performed by comparing the run time of the parallel application versus the non-parallel application.

8.2 FUTURE WORK

This application will play a major role in controlling the operations of Nuclear Fuel Handling System by providing 3D visualization. Since it is a safety critical system, the application will play a positive role in shutting down the system during emergency situations. We hope that the Stereoscopy implemented in this project will be likewise implemented in the real Stereo camera system such as a broadcast system in the near future.

REFERENCES

[1] Woonchul Ham and Seunghwan Kim, “A New Algorithm for Depth Perception in 3D and Its Implementation”, *International Symposium on Information Technology Convergence* , 2007.

[2] Bo Wang, Lei Zhu, Kebin Jia, Jie Zheng, “Accelerated Cone Beam CT Reconstruction Based on OpenCL”, *The 12th Int. World Wide Web Conference*, 2010.

[3] Jinglin Zhang, Jean-Francois Nezan, Jean-Gabriel Cousin, “Implementation of Motion Estimation Based On Heterogeneous Parallel Computing System with OpenCL”, In *14th International Conference on High Performance Computing and Communications*, 2012.

[4] Cs. Szab, B. Sobota and Ł. Sin?k, “Depth Maps and Other Techniques of Stereoscopy in 3D Scene Visualization”, *IEEE 9th International Symposium on Intelligent Systems and Informatics,September 8-10, 2011, Subotica, Serbia* , 2011.

[5] Shi Kejian and Wang Fei, “The development of stereoscopic display technology”, *3rd International Conference on Advanced Computer Theory and Engineering*,2010 .

[6] Milan polvka, Jaroslav chlouba, Petr cern, Pavel hazdra, Zbynek kvor, “3D Vision of Electromagnetic Fields in Antenna and Microwave Technique”, *15th Conference on Microwave Techniques*, 2010.

[7] Liu Qian, Gong Hui, and Luo Qingming, “Parallel Visualization of Visible Chinese Human with Extremely Large Datasets”, *Proceedings of the*

2005 IEEE Engineering in Medicine and Biology 27th Annual Conference Shanghai, China, September 1-4, 2005.

[8] Min Tang and Zixue Qiu, "Parallel Visualization of Large Medical Datasets Based on Computer Cluster", *The Ninth International Conference on Electronic Measurement and Instruments*,2009 .

[9] Wolfgang Fenz, Johannes Dirnberger, Christoph Watzl and Michael Krieger, "Parallel Simulation and Visualization of Blood Flow in Intracranial Aneurysms", *11th IEEE/ACM International Conference on Grid Computing* ,2010 .

[10] Adis Hamzi?,Alvin Huseinovi?,Novica Nosovi?, "Implementation and performance analysis of the Simplex algorithm adapted to run on commodity OpenCL enabled graphics processors", *IEEE* , 2011.

[11] Sangmin Seo, Gangwon Jo and Jaejin Lee, "Performance Characterization of the NAS Parallel Benchmarks in OpenCL", *IEEE* , 2011.

[12] A. M. Tourapis, "Enhanced predictive zonal search for single and multiple frame motion estimation", *proceedings of Visual Communications and Image Processing, pp. 1069C79* ,2009 .

[13] F.Urban and JF.Nezan, "HDS, a real-time multi-DSP motion estimator for MPEG-4 H.264 AVC high definition video encoding", *Real-Time Image Processing, vol. 4, no. 1, pp. 2331* , 2009.

[14] N.-M. Cheung, "Video coding on multicore graphics processors", *Signal Processing Magazine, IEEE, vol. 27, no. 2, pp. 7989* , March 2010.

[15] N. Holliman, “Smoothing Region Boundaries in Variable Depth Mapping for Real Time Stereoscopic Images”, *Stereoscopic Displays and Virtual reality SystemXII, Proceedings of SPIE-IST Symposium on Electronic Imaging,vol.5664A* ,2005 .

[16] N. Holliman, “Mapping Perceived Depth to Regions of Interest in Stereoscopic Images”, *Stereoscopic Displays and Virtual reality System XI, Proceedings of SPIE 5291* , 2004.