VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi-590018

A MINI PROJECT REPORT

ON

"ROBOT WALKING SIMULATION"

A dissertation submitted in the partial fulfilment of the requirement for the Mini-Project

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

SINCHANA S	**1KI20CS100**
SPOORTHY N	**1KI20CS104**

Under the Guidance of

Prof. Harish B M

B.E, M Tech,

Assistant Professor

Dept. of CSE, KIT, Tiptur

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KALPATARU INSTITUTE OF TECHNOLOGY

NH -206, TIPTUR-572201

2022-2023

KALPATARU INSTITUTE OF TECHNOLOGY

(Affiliated to Visvesvaraya Technological University, Belagavi & Recognized by AICTE, New Delhi)

Department of Computer Science & Engineering
NBA ACCREDITED 2022-25
NH - 206, TIPTUR -572201

CERTIFICATE

Certified that the mini project work entitled **"ROBOT WALKING SIMULATION"** is a bonafide work carried out by

SINCHANA S	**1KI20CS100**
SPOORTHY N	**1KI20CS104**

in partial fulfilment for the Mini-Project of **Bachelor of Engineering** in **Computer Science & Engineering** of the **Visvesvaraya Technological University, Belagavi** during the year 2022-2023. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements.

Name of the guide
Prof. Harish B M
B.E, M Tech,
Assistant Professor
Dept. of CSE, KIT, Tiptur

Head of Department
Prof. Shashidara M S
Associate Professor & HOD
Dept. of CSE,
KIT, Tiptur

Name of Examiners	**Signature with date**
1.	
2.	

KALPATARU INSTITUTE OF TECHNOLOGY

(Affiliated to Visvesvaraya Technological University, Belagavi & Recognized by AICTE, New Delhi)

Department of Computer Science & Engineering

NBA ACCREDITED 2022-25

NH - 206 , TIPTUR-572201

DECLARATION

We, the students of six semester of Computer Science & Engineering, Kalpataru Institute of Technology NH – 206 , Tiptur -572201, declare that the work entitled **"ROBOT WALKING SIMULATION"** has been successfully completed under the guidance of **Prof. Harish B M,** Department of Computer Science & Engineering. This dissertation work is submitted to Visvesvaraya Technological University in partial fulfilment for the mini-project of Engineering in ***Computer Science & Engineering*** during the academic year **2022 - 2023.**

Place: Tiptur

Date: 26/06/2023

Project Associates:

Sl. No.	*Student Name*	*USN*
1	SINCHANA S	1KI20CS100 -
2	SPOORTHY N	1KI20CS104 -

ACKNOWLEDGEMENT

The satisfaction and great happiness that accompany the successful completion of any task would be incomplete without mentioning about the people who made it possible. Here we make an honest attempt to express our deepest gratitude to all those who have been helpful and responsible for the successful completion of our project.

We would like to thank **Dr. G. D. GURUMURTHY, Principal,** Kalpataru Institute of Technology, Tiptur for his continuous support and encouragement throughout the course of this project.

We would like to thank **Prof. SHASHIDHARA M S, Head of Department**, Department of Computer Science and Engineering, Kalpataru Institute of Technology, Tiptur for his continuous support and encouragement throughout the course of this project.

We are immensely indebted to our internal guide, **Prof. HARISH B M**, Department of Computer Science and Engineering, Kalpataru Institute of Technology, Tiptur for his support, technical assistance and constructive suggestions and guidance for successful completion of our project. We are very much thankful to him for the encouragement that has infused at most real into us to complete the project work.

We would like to thank all faculty members of Department of Computer Science and Engineering, KIT, Tiptur, our family members, and to our friends who are directly or indirectly responsible for our success.

Thanking you,

SINCHANA S

SPOORTHY N

ABSTRACT

The idea behind this project is to display the concept of Robot Walking Simulation with computer graphics. This graphics package is based on the OpenGL library functions. The programming language used here is C using OpenGL libraries. The interface between interactive computer graphics and robotics is examined in terms of design, analysis, and programming. Dynamic and kinematic simulations are used to design and evaluate adaptive control systems for three-link articulated robots and to design robot operating strategies. A prototype interactive robotic design amd programming system is used to design and simulate the robotic manipulator. The concept of interactive graphics and programming for industrial robots is described. The dynamic and kinematic simulations play key roles in this concept.
Main theme of the project is, Legged robots are being the target of several studies and research. The idea is to develop machines that present characteristics approximate to the ones observed in biological living creatures. However, this objective is still relatively faraway and the development of prototypes for these studies is expensive and time consuming, which leads to the creation of models that allow the realization of the intended studies in software. The presented work describes the development of a quadruped robot model using opengl function. This model is intended to be used in the development of gaits for legged robots based on Central Pattern Generators. With this purpose in mind, the model was developed in a way to accept different gaits by direct introduction of the angular positions of the knee and hip joints.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

A robot is a programmable machine that can complete a task, while the term robotics describes the field of study focused on developing robots and automation. Each robot has a different level of autonomy. These levels range from human-controlled bots that carry out tasks to fully-autonomous bots that perform tasks without any external influences. With the rapid development of modern science and technology, robot technology has been widely used in various fields of manufacturing industrial. The main components of robot are : Control System, Sensors, Actuators, Power Supply, End Effectors. The following are things robots do better than humans:

- Automate manual or repetitive activities in corporate or industrial settings.
- Work in unpredictable or hazardous environments to spot hazards like gas leaks.
- Deliver online orders, room service and even food packets during emergencies.
- Assist during surgeries.
- Robots can also make music, monitor shorelines for dangerous predators, help with search and rescue and even assist with food preparation.

1.1 Problem Statement

The aim of this project is to design and simulate the “Robot Walking Simulation”. The key challenges here are:

1) how to model the Robot Walking Simulation?
2) how to make movements in Robot?

In this project, this model is simulated by using OpenGL functions. This implementation is achieved by using a standard user input interface.

1.2 Objective of the Project

1) The main objective of the project is to display with the concept of toll collection booth.
2) To implement the concepts of Computer Graphics we have learnt.
3) To implement the barrier that is lifted by user.
4) To incorporate colouring effects.

CHAPTER 2
HISTORICAL REVIEW

2.1 History of Robot

At their core, robots can best be described as artificial servants, an idea that stretches all the way back to Ancient Egypt. And while the technology to build robots as we might recognize them today was centuries from being invented, the idea of a robot was certainly alive and well.

Ancient myths are some of the earliest sources for how our ancestors saw themselves and the world around them. It’s somewhat telling that you can find many examples of artificial machines created in one way or another to supplant organic human activity.

While some of these myths clearly fused the idea of robots with artificial intelligence, many other examples exist of machines that carried out particular physical tasks without the need for direct human operation. These were often powered by hydraulics or steam and could perform tasks as varied as telling the time to providing entertainment for rich citizens of the ancient world.

These depictions of what would eventually become robots had little in the way of development until the 20th century. Aside from great thinkers such as Leonardo daVinci, few examples of “classic” robots exist. Despite this, many inventors around the world used pneumatic and hydraulic power to create rudimentary automatons that could perform a wide range of tasks. It wasn’t until the advent of the Industrial Revolution and the new leaps in energy technology that came along with it did we start to see fully autonomous machines capable of performing increasingly complex tasks.

The current stage of robotic development has brought us full circle to the myths of antiquity: intelligent automatons that can both act *and* think independent of human intervention. Creations that could one day come to share the very elements that make us human.

While the current advances in robotic technology have been no surprise to a society inundated with science-fiction narratives, the astounding leaps made even as far back as ancient. Egypt are another matter entirely, and given much less attention than they deserve. Read on for some of the greatest achievements in robotics from the dawn of civilization to today.

2.2 Evolution of Robot

People have been pondering robots since ancient civilizations incorporated myths and beliefs of “thinking machines” into their societies and invented the water clock. Robotics has drastically changed since the time of the Greeks, Romans and Egyptians, but its history is vast. Here’s a look at some of the most important events that shaped the history of robotics.

1700s

- (1737) Jacques de Vaucanson builds the first biomechanical automaton on record. Called the Flute Player, the mechanical device plays 12 songs.

1920s

- (1920) The word “robot” makes its first appearance in Karel Capek’s play R.U.R. Robot is derived from the Czech word “robota,” which means “forced labour.”
- (1926) The first movie robot appears in *Metropolis.*

1930s

- (1936) Alan Turing publishes “On Computable Numbers,” a paper that introduces the concept of a theoretical computer called the Turing Machine.

1940s

- (1948) *Cybernetics or Control and Communication in the Animal* is published by MIT professor Norbert Wiener. The book speaks on the concept of communications and control in electronic, mechanical and biological systems.
- (1949) William Grey Walter, a neurophysiologist and inventor, introduces Elmer and Elsie, a pair of battery-operated robots that look like tortoises. The robots move objects, find a source of light and find their way back to a charging station.

1950s

- (1950) Isaac Asimov publishes the *Three Laws of Robotics*.
- (1950) Alan Turing publishes the paper “Computing Machinery and Intelligence,” proposing what is now known as the Turing Test, a method for determining if a machine is intelligent.

1960s

- (1961) The first robotic arm works in a General Motors facility. The arm lifts and stacks metal parts and follows a program for approximately 200 movements. The arm was created by George Devol and his partner Joseph Engelberger.

- (1969) Victor Scheinman invents the Stanford Arm, a robotic arm with six joints that can mimic the movements of a human arm. It is one of the first robots designed to be controlled by a computer.

1970s

- (1972) A group of engineers at the Stanford Research Institute create Shakey, the first robot to use artificial intelligence. Shakey completes tasks by observing its environment and forming a plan. The robot uses sensors, a range-finder and touch-sensitive apparatus to plan its moves.
- (1978) Hiroshi Makino, an automation researcher, designs a four-axis SCARA robotic arm. Known as the first “pick and place” robot, the arm is programmed to pick an object up, turn and place it in another location.

1980s

- (1985) The first documented use of a robot-assisted surgical procedure uses the PUMA 560 robotic surgical arm.
- (1985) William Whittaker builds two remotely-operated robots that are sent to the Three Mile Island nuclear power plant. The robots work in the damaged reactor building’s basement to survey the site, send back information and drill core samples to measure radiation levels.
- (1989) MIT researchers Rodney Brooks and A. M. Flynn publish *Fast, Cheap and Out of Control: A Robot Invasion of the Solar System*. The paper argues for building many small, cheap robots rather than few big, expensive ones.

1990s

- (1990) A group of researchers from MIT found iRobot, the company behind the Roomba vacuum cleaner.
- (1992) Marc Raibert, another MIT researcher, founds robotics company Boston Dynamics.
- (1997) Sojourner lands on Mars. The free-ranging rover sends 2.3 billion bits of data back to Earth, including more than 17,000 images, 15 chemical analyses of rocks and soil and extensive data on Mars’ weather.
- (1998) Furby, a robotic toy pet developed by Tiger Electronics, is released and eventually sells tens of millions of units. Furbys are preprogrammed to speak gibberish and learn other languages over time.
- (1999) Aibo, a robotic puppy powered by AI hits the commercial market. Developed by Sony, the robotic dog reacts to sounds and has some pre-programmed behavior.

2000s

- (2000) Cynthia Breazeal creates a robotic head programmed to provoke emotions as well as react to them. Called Kismet, the robot consists of 21 motors, audio sensors and algorithms to understand vocal tone.
- (2001) iRobot’s PackBot searches the World Trade Center site after September 11th.
- (2002) iRobot creates Roomba. The vacuum robot is the first robot to become popular in the commercial sector amongst the public.
- (2003) Mick Mountz and the cofounders of Amazon Robotics (formerly Kiva Systems) invent the Kiva robot. The robot maneuvers around warehouses and moves goods.
- (2004) Boston Dynamics unveils BigDog, a quadruped robot controlled by humans. The robot is known for being more nimble than previous iterations of robots, as it is capable of only having two feet on the ground at a time. It has 50 sensors and an onboard computer that manages the gait and keeps it stable.
- (2005) A Volkswagen Touareg named Stanley wins the second DARPA Grand Challenge. The car uses AI trained on the driving habits of real-world humans and five lidar laser sensors to complete a 131.2-mile course in the Mojave Desert.

2010s

- (2011) NASA and General Motors collaborate to send Robonaut 2, a humanesque robotic assistant, into space on space shuttle Discovery. The robot becomes a permanent resident of the International Space Station.
- (2013) Boston Dynamics releases Atlas, a humanoid biped robot that uses 28 hydraulic joints to mimic human movements — including performing a backflip.
- (2012) The first license for a self-driven car is issued in Nevada. The car is a Toyota Prius modified with technology developed by Google.
- (2014) Canadian researchers develop hitch BOT, a bot that hitchhikes across Canada and Europe as part of a social experiment.
- (2016) Sophia, a humanoid robot dubbed the first robot citizen, is created by Hanson Robotics. The robot is capable of facial recognition, verbal communication and facial expression.

2020s

- (2020) Robots are used to distribute COVID-19 tests and vaccinations.
- (2020) 384,000 industrial robots are shipped across the globe to perform various manufacturing and warehouse jobs.
- (2021) Cruise, an autonomous car company, conducts its first two rob taxi test rides in San Francisco.

2.3 History of Computer Graphics

Computer graphics deals with generating images with the aid of computers. Today, computer graphics is a core technology in digital photography, film, video games, cell phone and computer display, and many specialized applications.

The first cathode ray tube, the Braun tube, was invented in 1897 – it in turn would permit the oscilloscope and the military control panel – the more direct precursors of the field, as they provided the first two-dimensional electronic displays that responded to programmatic or user input. New kinds of displays were needed to process the wealth of information resulting from such projects, leading to the development of computer graphics as a discipline.

In 1996, Krishnamurty and Levoy invented normal mapping – an improvement on Jim Blinn's bump mapping. By the end of the decade, computers adopted common frameworks for graphics processing such as DirectX and OpenGL. Since then, computer graphics have only become more detailed and realistic, due to more powerful graphics hardware and 3D modeling software.

2.4 History of OpenGL

By the early 1990s, Silicon Graphics (SGI) was a leader in 3D graphics for workstations. Their IRIS GL API became the industry standard, used more widely than the open standards-based PHIGS. This was because IRIS GL was considered easier to use, and because it supported immediate mode rendering. By contrast, PHIGS was considered difficult to use and outdated in functionality.

SGI's competitors (including Sun Microsystems, Hewlett-Packard and IBM) were also able to bring to market 3D hardware supported by extensions made to the PHIGS standard, which pressured SGI to open source a version of IrisGL as a public standard called OpenGL.

However, SGI had many customers for whom the change from IrisGL to OpenGL would demand significant investment. Moreover, IrisGL had API functions that were irrelevant to 3D graphics. For example, it included a windowing, keyboard and mouse API, in part because it was developed before the X Window System and Sun's NeWS. And, IrisGL libraries were unsuitable for opening due to licensing and patent issues. These factors required SGI to continue to support the advanced and proprietary Iris Inventor and Iris Performer programming APIs while market support for OpenGL matured.

CHAPTER 3
REQUIREMENT SPECIFICATION

3.1 System Requirement

The basic requirements for the development of this mini project are as follows:

3.1.1 Hardware Constraints

- Processor : Pentium PC
- RAM : 512MB
- Hard Disk : 20GB(approx)
- Display : VGA Color Monitor

3.1.2 Software Constraints

- Software: OpenGL 4.3 or above
- Operating System : Windows 98SE/2000/XP/Vista/UBUNTU
- Compiler : Eclipse/Microsoft Visual studio 2005
- Programming languages: C/C++

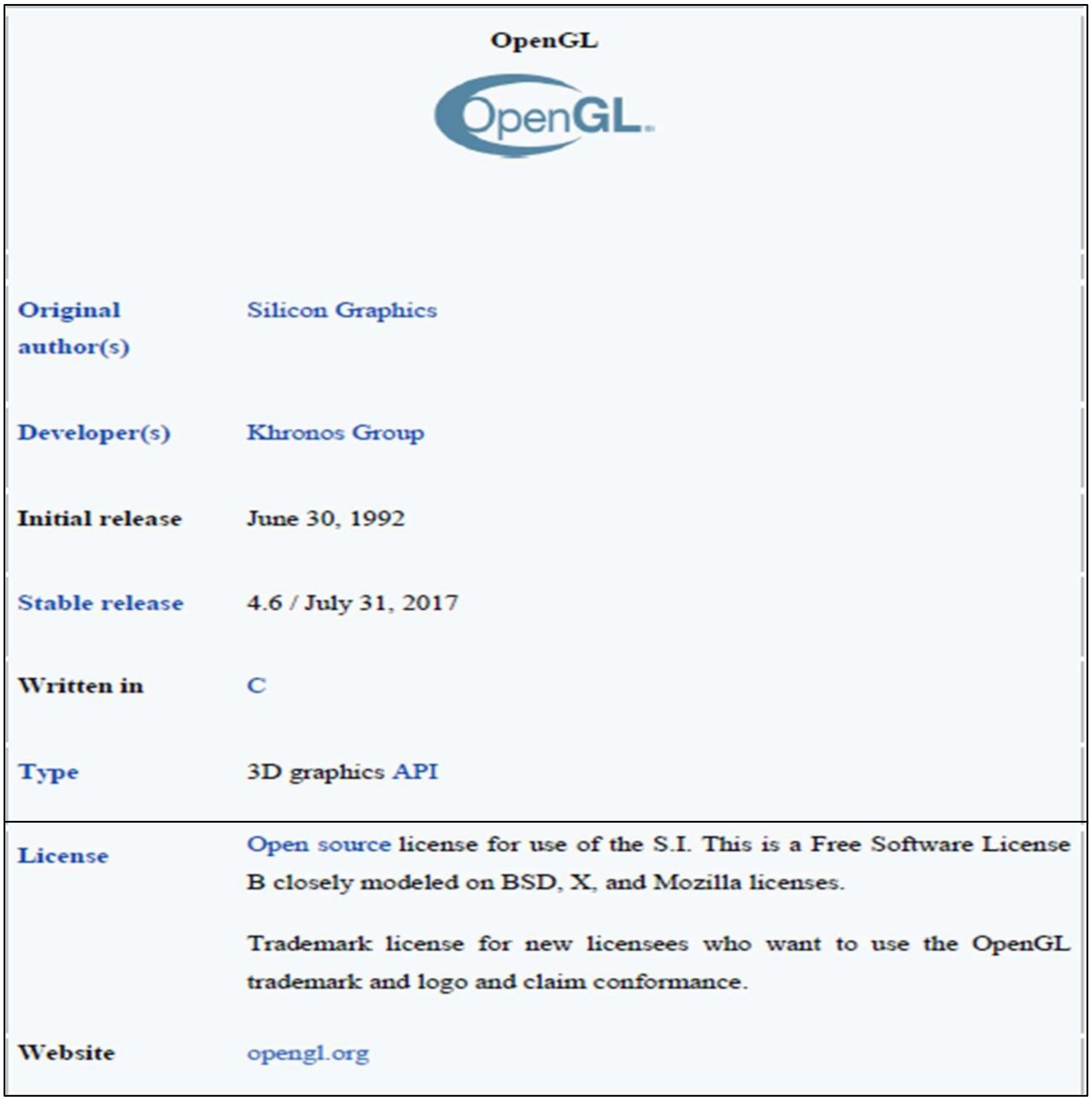
3.2 Development Environment

1. Software – OpenGL:

OpenGL (Open Graphics Library) is a cross-language, cross-platform 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.

Silicon Graphics, Inc. (SGI) began developing OpenGL in 1991 and released it on June 30, 1992; applications use it extensively in the fields of computer-aided design (CAD), virtual reality, scientific visualization, information visualization, flight simulation, and video games. Since 2006, OpenGL has been managed by the non-profit technology consortium Khronos Group.

The OpenGL specification describes an abstract API for drawing 2D and 3D graphics. Although it is possible for the API to be implemented entirely in software, it is designed to be implemented mostly or entirely in hardware. The API is defined as a set of functions which may be called by the client program, alongside a set of named integer constants (for example, the constant GL_TEXTURE_2D, which corresponds to the decimal number 3553).



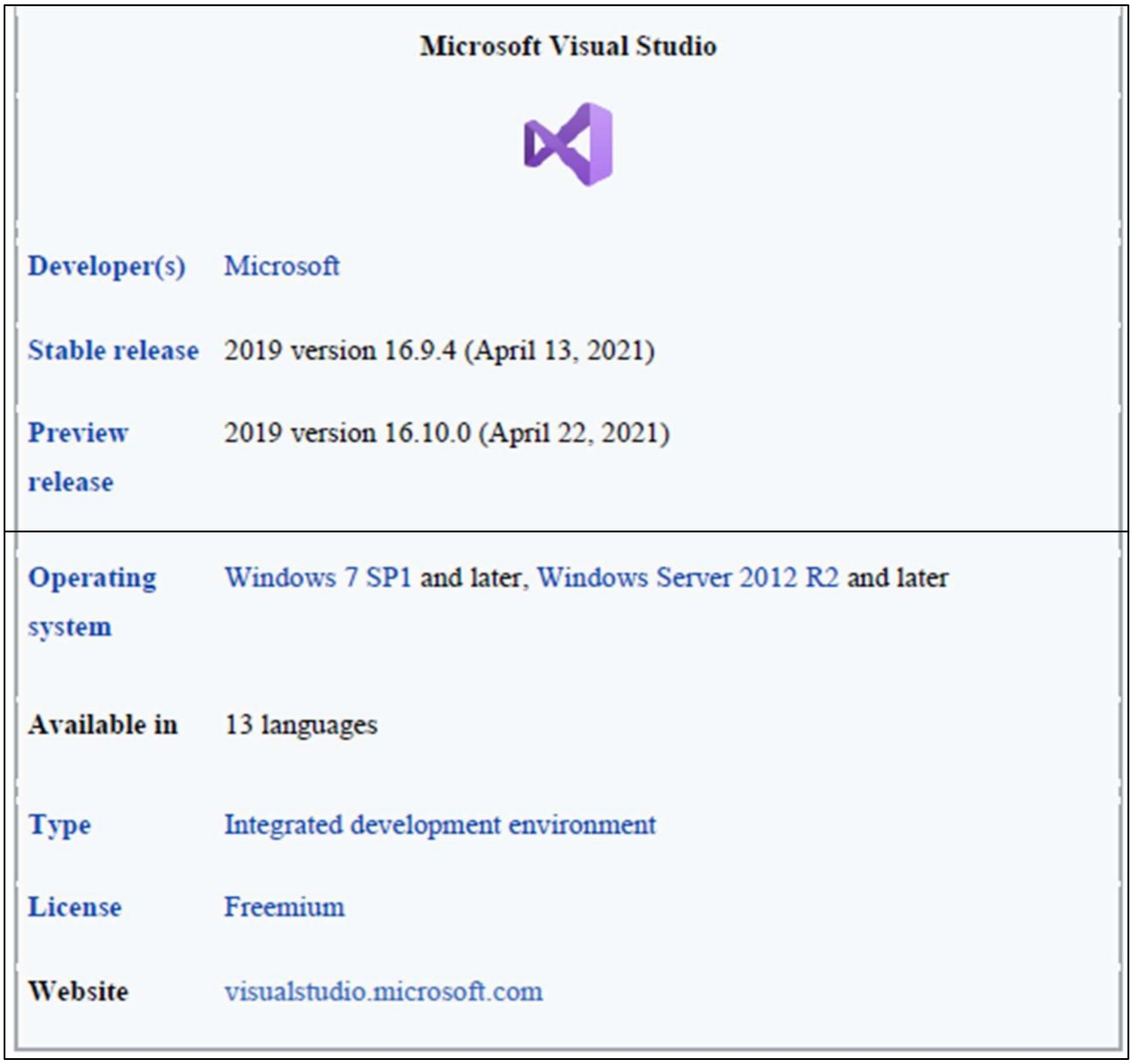
OpenGL

Original author(s)	Silicon Graphics
Developer(s)	Khronos Group
Initial release	June 30, 1992
Stable release	4.6 / July 31, 2017
Written in	C
Type	3D graphics API
License	Open source license for use of the S.I. This is a Free Software License B closely modeled on BSD, X, and Mozilla licenses. Trademark license for new licensees who want to use the OpenGL trademark and logo and claim conformance.
Website	opengl.org

Fig 3.a OpenGL

2. Development tool – Microsoft Visual Studio:

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services, mobile apps and GUI applications. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code.



Microsoft Visual Studio

Developer(s)	Microsoft
Stable release	2019 version 16.9.4 (April 13, 2021)
Preview release	2019 version 16.10.0 (April 22, 2021)
Operating system	Windows 7 SP1 and later, Windows Server 2012 R2 and later
Available in	13 languages
Type	Integrated development environment
License	Freemium
Website	visualstudio.microsoft.com

Fig 3.b Microsoft Visual Studio

CHAPTER 4
SYSTEM DESIGN

4.1 Flow Chart

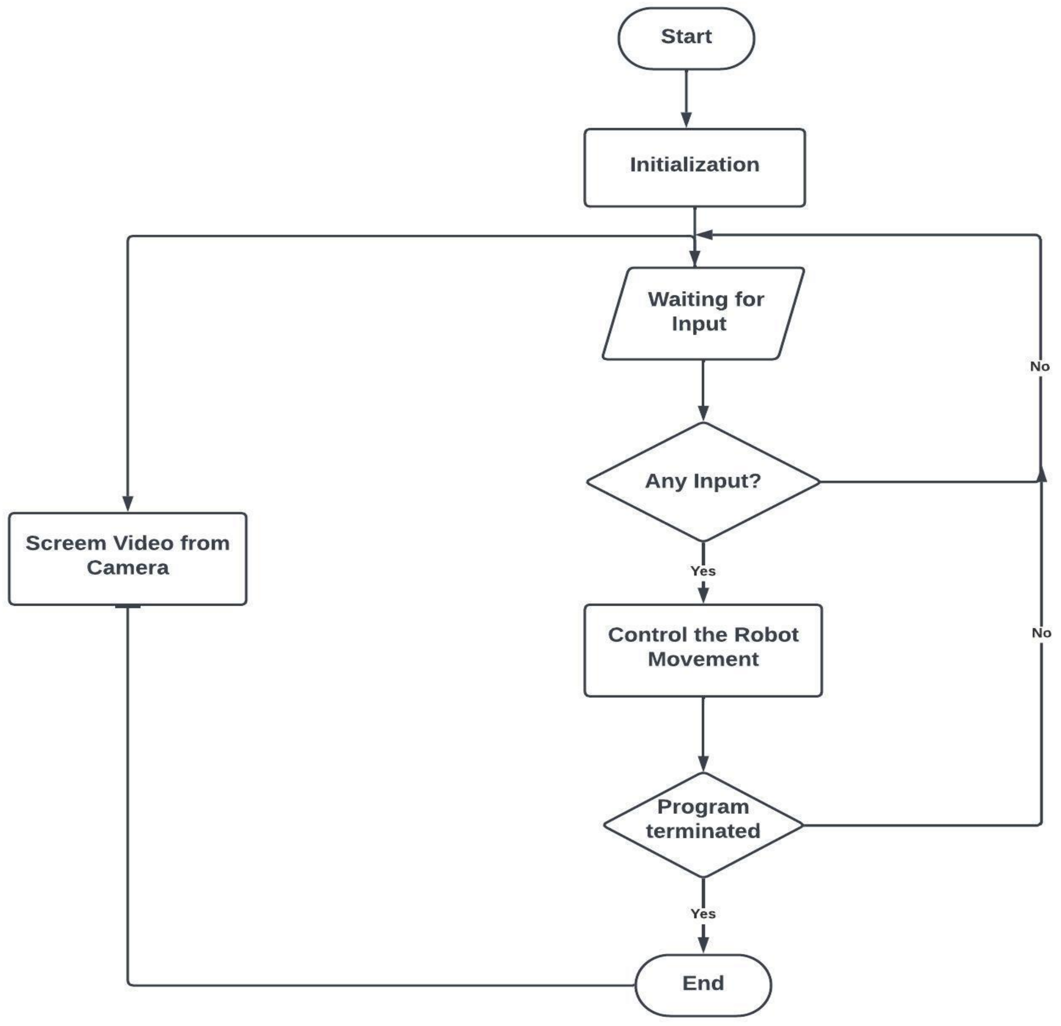


Fig 4.a Flow Chart

Initially it creates the main and the sub menus, whereas main menu is to “Quit, increase size and decrease size” of the objects. And sub menusare used for color. The menu is assigned to mouse click, when mouse (LEFT or RIGHT) button is pressed the menu will be appeared and action is performed.

4.2 Keyboard Functions

- The keys q, w, a, s are used to move shoulders backward and forward.
- The keys z, x, Z, X are used to move shoulders outward and inward.
- The keys Q, W, A, S are used to move elbows upward and downward.
- The keys 1, 2, 3, 4 are used to move elbows outward and inward.
- The keys page up is used to fire the Vulcan guns.
- The keys y, u, h, j, Y, U, H, J are used to move legs Bottom part forward, backward, outward and inward.
- The keys n, m, N, M are used to move legs Knee part forward and backward.
- The keys K, L, k, l are used to move legs Ankle part upward and downward.
- The keys d, g are used to move Torso part to turn left and right.
- The keys f, r are used to move the Upper portion backward and forward.
- To rotate the Scene we use right arrow, left arrow, down arrow and up arrow.
- To keys p, i, 9, o are used to rotate the light source.

4.3 Mouse Functions

- Using the mouse user can make the robot to start walk, stop walk and to Toggle Wireframe.
- The options that display when we right-click the mouse are listed as follows:
  - Start Walk
  - Stop Walk
  - Toggle Wireframe
  - How do I?
  - Quit

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 OpenGL Libraries

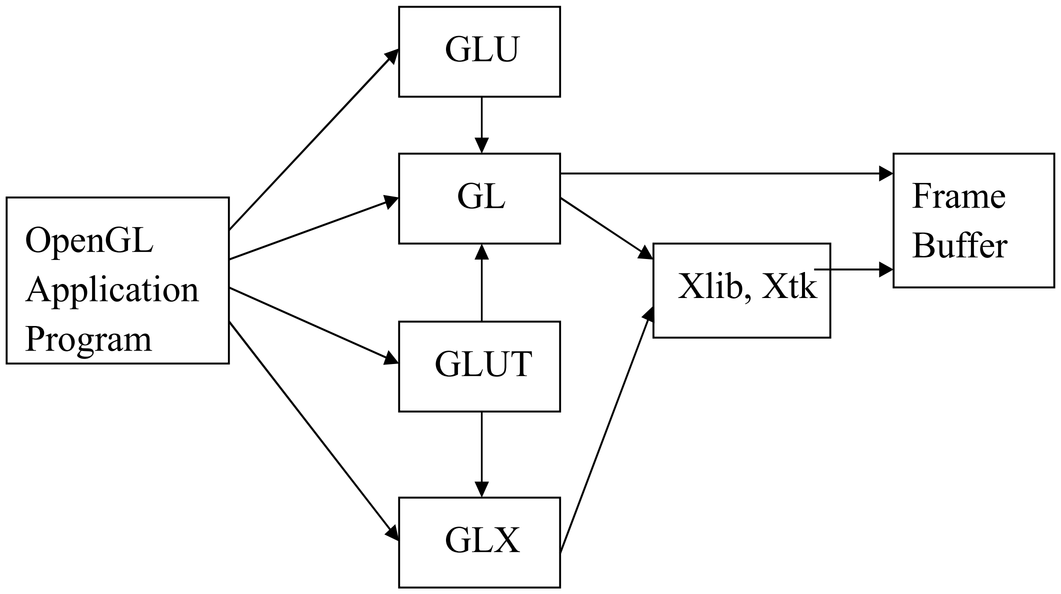


Fig 5.a OpenGL Libraries

Fig 5.1 shows the organization of the libraries for an X Window System environment. For this window system, GLUT will use GLX and the X libraries. The application program, however, can use only GLUT functions and thus can be recompiled with the GLUT library for other window systems.

1. GLU Library:

The OpenGL Utility Library (GLU) is a computer graphics library for OpenGL. It consists of a number of functions that use the base OpenGL library to provide higher-level drawing routines from the more primitive routines that OpenGL provides. It is usually distributed with the base OpenGL package.

Among these features are mapping between screen- and world-coordinates, generation of texture mipmaps, drawing of quadric surfaces, NURBS, tessellation of polygonal primitives, interpretation of OpenGL error codes, an extended range of transformation routines for setting up viewing volumes and simple positioning of the camera, generally in more human-friendly terms than the routines presented by OpenGL. All GLU functions start with the 'glu' prefix.

2. GL Library:

A graphics library is a program library designed to aid in rendering computer graphics to a monitor. This typically involves providing optimized versions of functions that handle common rendering tasks. This can be done purely in software and running on the CPU, common in embedded systems, or being hardware accelerated by a GPU, more common in PCs. By employing these functions, a program can assemble an image to be output to a monitor. Graphics libraries are mainly used in video games and simulations.

3. GLUT Library:

The OpenGL Utility Toolkit (GLUT) is a library of utilities for OpenGL programs, which primarily perform system-level I/O with the host operating system. Functions performed include window definition, window control, and monitoring of keyboard and mouse input. Routines for drawing a number of geometric primitives (both in solid and wireframe mode) are also provided, including cubes, spheres and the Utah teapot. All GLUT functions start with the 'glut' prefix. (for example, "glutPostRedisplay" marks the current window as needing to be redrawn).

FreeGLUT is an open-source alternative to the OpenGL Utility Toolkit (GLUT) library. GLUT (and hence FreeGLUT) allows the user to create and manage windows containing FreeGLUT is intended to be a full replacement for GLUT, and has only a few differences.

4. GLX Library:

GLX (initialism for "OpenGL Extension to the X Window System") is an extension to the X Window System core protocol providing an interface between OpenGL and the X Window System as well as extensions to OpenGL itself. It enables programs wishing to use OpenGL to do so within a window provided by the X Window System. GLX distinguishes two "states": indirect state and direct state.

5. Frame Buffer:

A Framebuffer is a collection of buffers that can be used as the destination for rendering. OpenGL has two kinds of framebuffers: the Default Framebuffer, which is provided by the OpenGL Context; and user-created framebuffers called Framebuffer Objects (FBOs). The buffers for default framebuffers are part of the context and usually represent a window or display device. The buffers for FBOs reference images from either Textures or Render buffers; they are never directly visible.

Default framebuffers cannot change their buffer attachments, but a particular default framebuffer may not have images associated with certain buffers. For example, the GL_BACK_RIGHT buffer will only have an image if the default framebuffer is double-buffered and uses stereoscopic 3D.

5.2 OpenGL Primitives

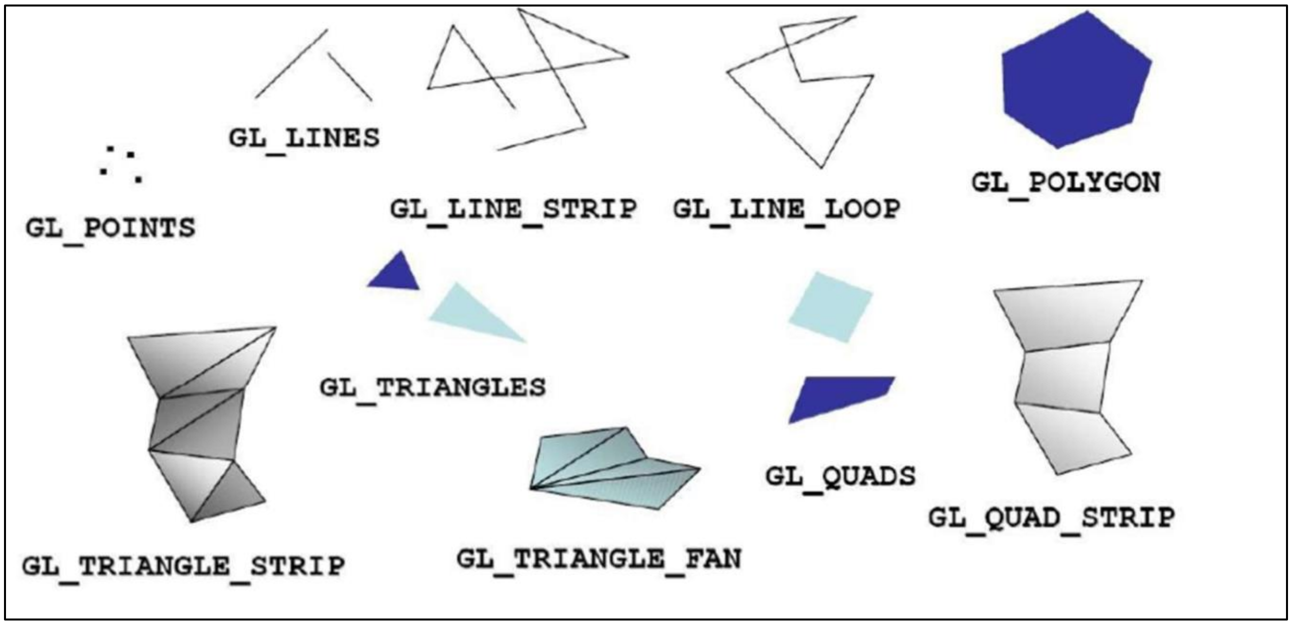


Fig 5.b OpenGL Primitives

The term Primitive in OpenGL is used to refer to two similar but separate concepts. The first meaning of "Primitive" refers to the interpretation scheme used by OpenGL to determine what a stream of vertices represents when being rendered ex: "GL_POINTS". Such sequences of vertices can be arbitrarily long.

1. <u>Point Primitive:</u>

There is only one kind of point primitive:

- GL_POINTS: This will cause OpenGL to interpret each individual vertex in the stream as a point. Points that have a Texture mapped onto them are often called "point sprites".

2. <u>Line Primitive:</u>

There are 3 kinds of line primitives, based on different interpretations of a vertex stream.

- GL_LINES: Vertices 0 and 1 are considered a line. Vertices 2 and 3 are considered a line. And so on. If the user specifies a non-even number of vertices, then the extra vertex is ignored.

- GL_LINE_STRIP: The adjacent vertices are considered lines. Thus, if you pass n vertices, you will get n-1 lines. If the user only specifies 1 vertex, the drawing command is ignored.

- GL_LINE_LOOP: As line strips, except that the first and last vertices are also used as a line. Thus, you get n lines for n input vertices. If the user only specifies 1 vertex, the drawing command is ignored. The line between the first and last vertices happens after all of the previous lines in the sequence.

3. <u>Triangle Primitives:</u>

A triangle is a primitive formed by 3 vertices. It is the 2D shape with the smallest number of vertices, so renderers are typically designed to render them. Since it is created from only 3 vertices, it is also guaranteed to be planar. There are 3 kinds of triangle primitives, based again on different interpretations of the vertex stream:

- GL_TRIANGLES: Vertices 0, 1, and 2 form a triangle, Vertices 3, 4, and 5 form a triangle, and so on.
- GL_TRIANGLE_STRIP: Every group of 3 adjacent vertices forms a triangle. The face direction of the strip is determined by the winding of the first triangle. Each successive triangle will have its effective face order reversed, so the system compensates for that by testing it in the opposite way. A vertex stream of n length will generate n-2 triangles.

4. Quads:

A quad is a 4-vertex quadrilateral primitive. The four vertices are expected to be coplanar; failure to do so can lead to undefined results. A quad is typically rasterized as a pair of triangles. This is not defined by the GL specification, but it is allowed by it. This can lead to some artifacts due to how vertex/geometry shader outputs are interpolated over the 2 generated triangles.

- GL_QUADS: Vertices 0-3 form a quad, vertices 4-7 form another, and so on. The vertex stream must be a number of vertices divisible by 4 to work

- GL_QUAD_STRIP: Similar to triangle strips, a quad strip uses adjacent edges to form the next quad. In the case of quads, the third and fourth vertices of one quad are used as the edge of the next quad.

5.3 Header Files

The headers that are used are as follows:

- #include<GL/glut.h>: To include glut library files.
- #include<stdio.h>: To include standard input and output files.
- #include<math.h>: To include various mathematical functions.

5.4: Functions

- void glScalef (TYPE sx, TYPE sy, TYPE sz) alters the current matrix by a scaling of (sx, sy, sz). TYPE here is GLfloat. Here in the above considered example we use scaling to minimize the length of the curve at each iteration. For this curve we use the scale factor to be 3 units because we substitute a line by 4 lines in each iteration.

- void glRotatef(TYPE angle, TYPE dx, TYPE dy, TYPE dz) alters the current matrix by a rotation of angle degrees about the axis(dx, dy, dz). TYPE heris GLfloat. For a Koch curve we rotate by 60° about the z-axis.

- void glTranslatef(TYPE x, TYPE y, TYPE z) alters the current matrix by a displacement of (x, y, z). TYPE here is GLfloat. We need to translate to display the new position of the line from the old position and also to go out to the beginning of the next side while drawing.

- void glLoadIdentity() sets the current transformation matrix to an identity matrix.

- void glPushMatrix() pushes to the matrix stack corresponding to the current matrix mode.

- void glPopMatrix() pops from the matrix stack corresponding to the current matrix mode.

- void gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top) defines a two-dimensional viewing rectangle in the plane z=0.

- void glutMouseFunc(myMouse) refers to the mouse callback function. Here mouse interface is given to increase a level of recursion by clicking mouse button and also to decrease a level of recursion by doing the same holding the shift on the keyboard.

- void glutKeyboardFunc(myKey) refers to the keyboard callback function. Here keyboard interface is given to quit, the user can quit by pressing ‘q’ and to see next example of the implementation, the user should press ‘n’.

- void glutInit(int *argc, char**argv) Initializes GLUT< the arguments from main are passed in and can by the application.

- void glutCreateWindow(char *title) Creates a window on the display. The string title can be used to label the window. The return value provides a reference to the window that can be used when there are multiple windows.

- void glutInitDisplaymode(unsigned int mode) Requests a display with the properties in mode. The value of mode is determined by the logical OR of options including the color model(GLUT_RGB<GLUT_INDEX) and buffering (GLUT_SINGLE<GLUT_DOUBLE).

- void glutInitWindowSize(int width,int heights) Specifies the initial height and width of the window in pixels.
- void glutInitWindowPosition(int x,int y) Specifies the initial position of the top-left corner of the window in pixels.

- void glutMainLoop() Cause the program to enter an event –processing loop.it should be the statement in main.

- void glutPostRedisplay() Requests that the display callback be executed after the current callback returns.

- void gluLookAt(GLdouble eyex,GLdouble eyey, GLdouble eyez, GLdouble atx, GLdouble aty, GLdouble atz, GLdouble upx, GLdouble upy, GLdouble upz) o Postmultiplies the current matrix determined by the viewer at the eye point looking at the point with specified up direction.

- void gluPerscpective(GLdouble fov, GLdouble aspect, GLdouble near, GLdouble far) o Defines a perspective viewing volume using the y direction field of view fov measured in degree,the aspect ratio of the front clipping plane, and the near and far distance.

5.5 SOURCE CODE

```
#define SPHERE
#define COLOR
#define LIGHT
#define TORSO
#define HIP
#define SHOULDER
#define UPPER_ARM
#define LOWER_ARM
#define ROCKET_POD
#define UPPER_LEG
#define LOWER_LEG
#define NO_NORM
#define ANIMATION
#define DRAW_MECH
#define DRAW_ENVIRO
#define MOVE_LIGHT
#include <stdlib.h>
#include <math.h>
#define GLUT
#define GLUT_KEY
#define GLUT_SPEC
#include <GL/glut.h>
#define TEXTID      19
void DrawTextXY(,double,double,double,char *);
#define SOLID_MECH_TORSO              1
#define SOLID_MECH_HIP         2
#define SOLID_MECH_SHOULDER     3
#define SOLID_MECH_UPPER_ARM    4
#define SOLID_MECH_FOREARM            5
#define SOLID_MECH_UPPER_LEG          6
#define SOLID_MECH_FOOT               7
#define SOLID_MECH_ROCKET             8
#define SOLID_MECH_VULCAN 9
#define SOLID_ENVIRO              10
#ifndef M_PI
#define M_PI 3.14
#endif
GLUquadricObj *qobj;
char leg = 0;
int shoulder1 = 0, shoulder2 = 0, shoulder3 = 0,
shoulder4 = 0, lat1 = 20, lat2 = 20,
 elbow1 = 0, elbow2 = 0, pivot = 0, tilt = 10,
ankle1 = 0, ankle2 = 0, heel1 = 0,
  heel2 = 0, hip11 = 0, hip12 = 10, hip21 = 0,
 hip22 = 10, fire = 0, solid_part = 0,
  anim = 0, turn = 0, turn1 = 0, lightturn = 0,
lightturn1 = 0;
float elevation = 0.0, distance = 0.0, frame = 3.0
#ifdef LIGHT
GLfloat mat_specular[] = {0.0, 0.0, 1.0, 1.0};
GLfloat mat_ambient[] ={0.0, 0.0, 1.0, 1.0};
GLfloat mat_diffuse[] ={0.0, 0.0, 1.0, 1.0};
GLfloat mat_shininess[] ={128.0 * 0.4};
GLfloat mat_specular2[] ={0.508273, 0.508273,
0.508373};
GLfloat mat_ambient2[] ={0.19225, 0.19225,
0.19225};
GLfloat mat_diffuse2[] ={0.50754, 0.50754,
0.50754};
GLfloat mat_shininess2[] ={128.0 * 0.6};
GLfloat mat_specular3[] = {1.0, 1.0, 0.0};
GLfloat mat_ambient3[] ={1.0, 1.0, 0.0};
GLfloat mat_diffuse3[] ={1.0, 1.0, 0.0};
GLfloat mat_shininess3[] ={0.0 * 0.0};
GLfloat mat_specular4[] = {0.633, 0.727811,
0.633};
GLfloat mat_ambient4[] = {0.0215, 0.1745,
0.0215};
GLfloat mat_diffuse4[] = {0.07568, 0.61424,
0.07568};
GLfloat mat_shininess4[] = {128 * 0.6};

GLfloat mat_specular5[] =
{0.60, 0.60, 0.50};
GLfloat mat_ambient5[] =
{0.0, 0.0, 0.0};
GLfloat mat_diffuse5[] =
{0.5, 0.5, 0.0};
GLfloat mat_shininess5[] =
{128.0 * 0.25};
#endif
void Heel1Add(void)
{
  heel1 = (heel1 + 3) % 360;
}
void Heel1Subtract(void)
{
  heel1 = (heel1 - 3) % 360;
}
void Heel2Add(void)
{
  heel2 = (heel2 + 3) % 360;
}
void Heel2Subtract(void)
{
  heel2 = (heel2 - 3) % 360;
}
void Ankle1Add(void)
{
  ankle1 = (ankle1 + 3) % 360;
}
void Ankle1Subtract(void)
{
  ankle1 = (ankle1 - 3) % 360;
}
```

```
void Ankle2Add(void)
{
  ankle2 = (ankle2 + 3) % 360;
}
void Ankle2Subtract(void)
{
  ankle2 = (ankle2 - 3) % 360;
}
void RotateAdd(void)
{
  pivot = (pivot + 3) % 360;
}
void RotateSubtract(void)
{
  pivot = (pivot - 10) % 360;
}
void MechTiltSubtract(void)
{
  tilt = (tilt - 10) % 360;
}
void MechTiltAdd(void)
{
  tilt = (tilt + 10) % 360;
}
void elbow1Add(void)
{
  elbow1 = (elbow1 + 2) % 360;
}
void elbow1Subtract(void)
{
  elbow1 = (elbow1 - 2) % 360;
}
void elbow2Add(void)
{
  elbow2 = (elbow2 + 2) % 360;
}
void elbow2Subtract(void)
{
  elbow2 = (elbow2 - 2) % 360;
}
void shoulder1Add(void)
{
  shoulder1 = (shoulder1 + 5) % 360;
}
void shoulder1Subtract(void)
{
  shoulder1 = (shoulder1 - 5) % 360;
}
void shoulder2Add(void)
{
  shoulder2 = (shoulder2 + 5) % 360;
}
void shoulder2Subtract(void)
{
  shoulder2 = (shoulder2 - 5) % 360;
}
void shoulder3Add(void)
{
  shoulder3 = (shoulder3 + 5) % 360;
}
void shoulder3Subtract(void)
{
  shoulder3 = (shoulder3 - 5) % 360;
}
void shoulder4Add(void)
{
  shoulder4 = (shoulder4 + 5) % 360;
}
void shoulder4Subtract(void)
{
  shoulder4 = (shoulder4 - 5) % 360;
}
void lat1Raise(void)
{
  lat1 = (lat1 + 5) % 360;
}
void lat1Lower(void)
{
  lat1 = (lat1 - 5) % 360;
}
void lat2Raise(void)
{
  lat2 = (lat2 + 5) % 360;
}
void lat2Lower(void)
{
  lat2 = (lat2 - 5) % 360;
}
void FireCannon(void)
{
  fire = (fire + 20) % 360;
}
void RaiseLeg1Forward(void)
{
  hip11 = (hip11 + 3) % 360;
}
void LowerLeg1Backwards(void)
{
  hip11 = (hip11 - 3) % 360;
}
void RaiseLeg1Outwards(void)
{
  hip12 = (hip12 + 10) % 360;
}
void LowerLeg1Inwards(void)
{
  hip12 = (hip12 - 10) % 360;
}
void RaiseLeg2Forward(void)
{
 hip21 = (hip21 + 3) % 360;
}
void LowerLeg2Backwards(void)
{
 hip21 = (hip21 - 3) % 360;
}
void RaiseLeg2Outwards(void)
{
 hip22 = (hip22 + 10) % 360;
}
void LowerLeg2Inwards(void)
{
 hip22 = (hip22 - 10) % 360;
}
void TurnRight(void)
{
 turn = (turn - 10) % 360;
}
void TurnLeft(void)
{
 turn = (turn + 10) % 360;
}
void TurnForwards(void)
{
 turn1 = (turn1 - 10) % 360;
}
void TurnBackwards(void)
{
 turn1 = (turn1 + 10) % 360;
}
void LightTurnRight(void)
{
 lightturn = (lightturn + 10) % 360;
}
void LightTurnLeft(void)
{
 lightturn = (lightturn - 10) % 360;
}

void LightForwards(void)
{
 lightturn1 = (lightturn1 + 10) % 360;
}

void LightBackwards(void)
{
 lightturn1 = (lightturn1 - 10) % 360;
}

void DrawTextXY(double x,double y,double
z,double scale,char *s)
{
  int i;
  glPushMatrix();
  glTranslatef(x,y,z);
glScalef(scale,scale,scale);
for (i=0;i<strlen(s);i++)
glutStrokeCharacter
(GLUT_STROKE_ROMAN,s[i]);
glPopMatrix();
}
void Box(float width, float height, float depth,
char solid)
{
char i, j = 0;
float x = width / 2.0, y = height / 2.0, z = depth / 2.0;
for (i = 0; i < 4; i++) {
glRotatef(90.0, 0.0, 0.0, 1.0);
if (j) {
if (!solid)
glBegin(GL_LINE_LOOP);
else
glBegin(GL_QUADS);
glNormal3f(-1.0, 0.0, 0.0);
glVertex3f(-x, y, z);
glVertex3f(-x, -y, z);
glVertex3f(-x, -y, -z);
glVertex3f(-x, y, -z);
glEnd();
if (solid) {
glBegin(GL_TRIANGLES);
glNormal3f(0.0, 0.0, 1.0);
glVertex3f(0.0, 0.0, z);
glVertex3f(-x, y, z);
glVertex3f(-x, -y, z);
glNormal3f(0.0, 0.0, -1.0);
glVertex3f(0.0, 0.0, -z);
glVertex3f(-x, -y, -z);
glVertex3f(-x, y, -z);
glEnd();
}
 j = 0;
} else {
 if (!solid)
glBegin(GL_LINE_LOOP);
else
glBegin(GL_QUADS);
glNormal3f(-1.0, 0.0, 0.0);
glVertex3f(-y, x, z);
glVertex3f(-y, -x, z);
glVertex3f(-y, -x, -z);
glVertex3f(-y, x, -z);
glEnd();
if (solid) {
glBegin(GL_TRIANGLES);
glNormal3f(0.0, 0.0, 1.0);
glVertex3f(0.0, 0.0, z);
glVertex3f(-y, x, z);
glVertex3f(-y, -x, z);
glNormal3f(0.0, 0.0, -1.0);
glVertex3f(0.0, 0.0, -z);
glVertex3f(-y, -x, -z);
glVertex3f(-y, x, -z);
glEnd();
}
j = 1;
}
}
}
void Octagon(float side, float height, char solid)
{
char j;
float x = sin(0.7) * side, y = side / 2.0, z = height /
2.0, c;
  c = x + y;
for (j = 0; j < 8; j++) {
glTranslatef(-c, 0.0, 0.0);
if (!solid)
glBegin(GL_LINE_LOOP);
else
glBegin(GL_QUADS);
glNormal3f(-1.0, 0.0, 0.0);
glVertex3f(0.0, -y, z);
glVertex3f(0.0, y, z);
glVertex3f(0.0, y, -z);
glVertex3f(0.0, -y, -z);
glEnd();
glTranslatef(c, 0.0, 0.0);
if (solid) {
glBegin(GL_TRIANGLES);
glNormal3f(0.0, 0.0, 1.0);
glVertex3f(0.0, 0.0, z);
glVertex3f(-c, -y, z);
glVertex3f(-c, y, z);
glNormal3f(0.0, 0.0, -1.0);
glVertex3f(0.0, 0.0, -z);
glVertex3f(-c, y, -z);
glVertex3f(-c, -y, -z);
glEnd();
}
glRotatef(45.0, 0.0, 0.0, 1.0);
}
}
#ifdef NORM
void Normalize(float v[3])
{
GLfloat d = sqrt(v[1] * v[1] + v[2] * v[2] + v[3] *
v[3]);
if (d == 0.0) {
printf("zero length vector");
return;
}
v[1] /= d;
v[2] /= d;
v[3] /= d;
}
void NormXprod(float v1[3], float v2[3], float v[3],
float out[3])
{
GLint i, j;GLfloat length;
out[0] = v1[1] * v2[2] - v1[2] * v2[1];
out[1] = v1[2] * v2[0] - v1[0] * v2[2];
out[2] = v1[0] * v2[1] - v1[1] * v2[0];
Normalize(out);
}
#endif
void SetMaterial(GLfloat spec[], GLfloat amb[],
GLfloat diff[], GLfloat shin[])
{
glMaterialfv(GL_FRONT, GL_SPECULAR,
spec);
glMaterialfv(GL_FRONT, GL_SHININESS,
shin);
glMaterialfv(GL_FRONT, GL_AMBIENT, amb);
glMaterialfv(GL_FRONT, GL_DIFFUSE, diff);
}
void MechTorso(char solid)
{
glNewList(SOLID_MECH_TORSO,
GL_COMPILE);
#ifdef LIGHT
SetMaterial(mat_specular,
mat_ambient, mat_diffuse, mat_shininess);
#endif
glColor3f(1.0, 0.0, 0.0);//torso red color
Box(1.0, 1.0, 3.0, solid);
glTranslatef(0.75, 0.0, 0.0);
#ifdef LIGHT
SetMaterial(mat_specular2,
mat_ambient2, mat_diffuse2, mat_shininess2);
#endif
glColor3f(0.0, 0.0, 1.0);//torso blue color
Box(0.5, 0.6, 2.0, solid);
glTranslatef(-1.5, 0.0, 0.0);
Box(0.5, 0.6, 2.0, solid);
glTranslatef(0.75, 0.0, 0.0);
glEndList();
}
void MechHip(char solid)
{
int i;
glNewList(SOLID_MECH_HIP, GL_COMPILE);
#ifdef LIGHT
SetMaterial(mat_specular,
mat_ambient, mat_diffuse, mat_shininess);
#endif
glColor3f(0.0, 1.0, 0.0);//hip lines form green
Octagon(0.7, 0.5, solid);
#ifdef SPHERE
for (i = 0; i < 2; i++) {
```

```
if (i)
glScalef(-1.0, 1.0, 1.0);
glTranslatef(1.0, 0.0, 0.0);
#ifdef LIGHT
SetMaterial(mat_specular2, mat_ambient2,
mat_diffuse2, mat_shininess2);
#endif
glColor3f(0.0, 1.0, 0.0);//hip line form green
if (!solid)
gluQuadricDrawStyle(qobj, GLU_LINE);
gluSphere(qobj, 0.2, 16, 16);
glTranslatef(-1.0, 0.0, 0.0);
}
glScalef(-1.0, 1.0, 1.0);
#endif
glEndList();
}
void Shoulder(char solid)
{
glNewList(SOLID_MECH_SHOULDER,
GL_COMPILE);
#ifdef LIGHT
SetMaterial(mat_specular,
mat_ambient, mat_diffuse, mat_shininess);
#endif
glColor3f(0.0, 1.0, 0.0);//sholder color green
Box(1.0, 0.5, 0.5, solid);
glTranslatef(0.9, 0.0, 0.0);
#ifdef LIGHT
SetMaterial(mat_specular2, mat_ambient2,
mat_diffuse2, mat_shininess2);
#endif
glColor3f(0.0, 1.0, 0.0);// sholder color green
#ifdef SPHERE
if (!solid)
gluQuadricDrawStyle(qobj, GLU_LINE);
gluSphere(qobj, 0.6, 16, 16);
#endif
glTranslatef(-0.9, 0.0, 0.0);
glEndList();
}
void UpperArm(char solid)
{
int i;
glNewList(SOLID_MECH_UPPER_ARM,
GL_COMPILE);
#ifdef LIGHT
SetMaterial(mat_specular,
mat_ambient, mat_diffuse, mat_shininess);
#endif
glColor3f(1.0, 0.0, 0.0);//arm red
Box(1.0, 2.0, 1.0, solid);
glTranslatef(0.0, -0.95, 0.0);
glRotatef(90.0, 1.0, 0.0, 0.0);
#ifdef LIGHT
SetMaterial(mat_specular2,mat_ambient2,
mat_diffuse2, mat_shininess2);
#endif
glColor3f(1.0, 0.0, 0.0);//arm red
if (!solid)
gluQuadricDrawStyle(qobj, GLU_LINE);
gluCylinder(qobj, 0.4, 0.4, 1.5, 16, 10);
#ifdef LIGHT
SetMaterial(mat_specular,
mat_ambient, mat_diffuse, mat_shininess);
#endif
glColor3f(0.0, 1.0, 0.0);// arm joint green
glRotatef(-90.0, 1.0, 0.0, 0.0);
glTranslatef(-0.4, -1.85, 0.0);
glRotatef(90.0, 0.0, 1.0, 0.0);
for (i = 0; i < 2; i++) {
if (!solid)
gluQuadricDrawStyle(qobj, GLU_LINE);
if (i)
gluCylinder(qobj, 0.5, 0.5, 0.8, 16, 10);
else
gluCylinder(qobj, 0.2, 0.2, 0.8, 16, 10);
}
for (i = 0; i < 2; i++) {
if (i)
glScalef(-1.0, 1.0, 1.0);
if (!solid)
gluQuadricDrawStyle(qobj, GLU_LINE);
if (i)
glTranslatef(0.0, 0.0, 0.8);
gluDisk(qobj, 0.2, 0.5, 16, 10);
if (i)
glTranslatef(0.0, 0.0, -0.8);
}
glScalef(-1.0, 1.0, 1.0);
glRotatef(-90.0, 0.0, 1.0, 0.0);
glTranslatef(0.4, 2.9, 0.0);
glEndList();
}
void VulcanGun(char solid)
{
int i;
glNewList(SOLID_MECH_VULCAN,
GL_COMPILE);
#ifdef LIGHT
SetMaterial(mat_specular2, mat_ambient2,
mat_diffuse2, mat_shininess2);
#endif
glColor3f(0.0, 0.0, 1.0);//gun color
if (!solid) {
gluQuadricDrawStyle(qobj, GLU_LINE);
}
gluCylinder(qobj, 0.5, 0.5, 0.5, 16, 10);
glTranslatef(0.0, 0.0, 0.5);
gluDisk(qobj, 0.0, 0.5, 16, 10);
```

```
for (i = 0; i < 5; i++) {
glRotatef(72.0, 0.0, 0.0, 1.0);
glTranslatef(0.0, 0.3, 0.0);
if (!solid) {
gluQuadricDrawStyle(qobj, GLU_LINE);
}
gluCylinder(qobj, 0.15, 0.15, 2.0, 16, 10);
gluCylinder(qobj, 0.06, 0.06, 2.0, 16, 10);
glTranslatef(0.0, 0.0, 2.0);
gluDisk(qobj, 0.1, 0.15, 16, 10);
gluCylinder(qobj, 0.1, 0.1, 0.1, 16, 5);
glTranslatef(0.0, 0.0, 0.1);
gluDisk(qobj, 0.06, 0.1, 16, 5);
glTranslatef(0.0, -0.3, -2.1);
}
glEndList();
}
void ForeArm(char solid)
{
char i;
glNewList(SOLID_MECH_FOREARM,
GL_COMPILE);
#ifdef LIGHT
SetMaterial(mat_specular, mat_ambient,
mat_diffuse, mat_shininess);
#endif
glColor3f(0.0, 1.0, 1.0);//fore arm light green
for (i = 0; i < 5; i++) {
glTranslatef(0.0, -0.1, -0.15);
Box(0.6, 0.8, 0.2, solid);
glTranslatef(0.0, 0.1, -0.15);
Box(0.4, 0.6, 0.1, solid);
}
glTranslatef(0.0, 0.0, 2.45);
Box(1.0, 1.0, 2.0, solid);
glTranslatef(0.0, 0.0, -1.0);
glEndList();
}
void UpperLeg(char solid)
{
int i;

glNewList(SOLID_MECH_UPPER_LEG,
GL_COMPILE);
#ifdef LIGHT
SetMaterial(mat_specular, mat_ambient,
mat_diffuse, mat_shininess);
#endif
glColor3f(1.0, 1.0, 0.0);//color yellow
if (!solid) {
gluQuadricDrawStyle(qobj, GLU_LINE);
}
glTranslatef(0.0, -1.0, 0.0);
Box(0.4, 1.0, 0.7, solid);
glTranslatef(0.0, -0.65, 0.0);
for (i = 0; i < 5; i++) {
Box(1.2, 0.3, 1.2, solid);
glTranslatef(0.0, -0.2, 0.0);
Box(1.0, 0.1, 1.0, solid);
glTranslatef(0.0, -0.2, 0.0);
}
glTranslatef(0.0, -0.15, -0.4);
Box(2.0, 0.5, 2.0, solid);
glTranslatef(0.0, -0.3, -0.2);
glRotatef(90.0, 1.0, 0.0, 0.0);
#ifdef LIGHT
SetMaterial(mat_specular2,
mat_ambient2, mat_diffuse2, mat_shininess2);
#endif
glColor3f(0.5, 0.5, 0.5);//leg joints grey
gluCylinder(qobj, 0.6, 0.6, 3.0, 16, 10);
#ifdef LIGHT
SetMaterial(mat_specular,
mat_ambient, mat_diffuse, mat_shininess);
#endif
glColor3f(0.0, 1.0, 0.0);//above the leg joint n
below the fore leg
glRotatef(-90.0, 1.0, 0.0, 0.0);
glTranslatef(0.0, -1.5, 1.0);
Box(1.5, 3.0, 0.5, solid);
glTranslatef(0.0, -1.75, -0.8);
Box(2.0, 0.5, 2.0, solid);
glTranslatef(0.0, -0.9, -0.85);
#ifdef LIGHT
SetMaterial(mat_specular2,
mat_ambient2, mat_diffuse2, mat_shininess2);
#endif
glColor3f(1.0, 0.0, 0.0);//leg joints between fore
leg and leg
gluCylinder(qobj, 0.8, 0.8, 1.8, 16, 10);
for (i = 0; i < 2; i++) {
if (i)
glScalef(-1.0, 1.0, 1.0);
if (!solid)
gluQuadricDrawStyle(qobj, GLU_LINE);
if (i)
glTranslatef(0.0, 0.0, 1.8);
gluDisk(qobj, 0.0, 0.8, 16, 10);
if (i)
glTranslatef(0.0, 0.0, -1.8);
}
glScalef(-1.0, 1.0, 1.0);
glEndList();
}
void Foot(char solid)
{
glNewList(SOLID_MECH_FOOT,
GL_COMPILE);
#ifdef LIGHT
SetMaterial(mat_specular2,
mat_ambient2, mat_diffuse2, mat_shininess2);
#endif
glColor3f(1.0, 0.0, 0.0);// color foot
glRotatef(90.0, 1.0, 0.0, 0.0);
Octagon(1.5, 0.6, solid);
glRotatef(-90.0, 1.0, 0.0, 0.0);
glEndList();
}
void LowerLeg(char solid)
{
float k, l;
#ifdef LIGHT
SetMaterial(mat_specular
mat_ambient, mat_diffuse, mat_shininess);
#endif
glColor3f(1.0, 0.0, 0.0);//leg joint
for (k = 0.0; k < 2.0; k++) {
for (l = 0.0; l < 2.0; l++) {
glPushMatrix();
glTranslatef(k, 0.0, l);
#ifdef LIGHT
SetMaterial(mat_specular,
mat_ambient, mat_diffuse, mat_shininess);
#endif
glColor3f(1.0, 0.0, 0.0);//red
Box(1.0, 0.5, 1.0, solid);
glTranslatef(0.0, -0.45, 0.0);
#ifdef LIGHT
SetMaterial(mat_specular2,
mat_ambient2, mat_diffuse2, mat_shininess2);
#endif
glColor3f(1.0, 0.0, 0.0);
#ifdef SPHERE
if (!solid)
glutWireSphere(0.2, 16, 10);
else
glutSolidSphere(0.2, 16, 10);
#endif
if (leg)
glRotatef((GLfloat) heel1, 1.0, 0.0, 0.0);
else
glRotatef((GLfloat) heel2, 1.0, 0.0, 0.0);
glTranslatef(0.0, -1.7, 0.0);
#ifdef LIGHT
SetMaterial(mat_specular, mat_ambient,
mat_diffuse, mat_shininess);
#endif
glColor3f(1.0, 0.0, 0.0);
Box(0.25, 3.0, 0.25, solid);
glTranslatef(0.0, -1.7, 0.0);
#ifdef LIGHT
SetMaterial(mat_specular2,
mat_ambient2, mat_diffuse2, mat_shininess2);
#endif
glColor3f(0.0, 0.0, 1.0);//joints
#ifdef SPHERE
if (!solid)
glutWireSphere(0.2, 16, 10);
else
glutSolidSphere(0.2, 16, 10);
#endif
glColor3f(0.0, 1.0, 0.0);// leg n foot joints color
Box(1.0, 0.5, 1.0, solid);
if (!k && !l) {
int j;
glTranslatef(-0.4, -0.8, 0.5);
if (leg)
glRotatef((GLfloat) ankle1, 1.0, 0.0, 0.0);
else
glRotatef((GLfloat) ankle2, 1.0, 0.0, 0.0);
glRotatef(90.0, 0.0, 1.0, 0.0);
if (!solid)
gluQuadricDrawStyle(qobj, GLU_LINE);
gluCylinder(qobj, 0.8, 0.8, 1.8, 16, 10);
for (j = 0; j < 2; j++) {
if (!solid)
gluQuadricDrawStyle(qobj, GLU_LINE);
if (j) {
glScalef(-1.0, 1.0, 1.0);
glTranslatef(0.0, 0.0, 1.8);
}
gluDisk(qobj, 0.0, 0.8, 16, 10);
if (j)
glTranslatef(0.0, 0.0, -1.8);
}
glScalef(-1.0, 1.0, 1.0);
glRotatef(-90.0, 0.0, 1.0, 0.0);
glTranslatef(0.95, -0.8, 0.0);
glCallList(SOLID_MECH_FOOT);
}
glPopMatrix();
}
}
}
void RocketPod(char solid)
{
int i, j, k = 0;
glNewList(SOLID_MECH_ROCKET,
GL_COMPILE);
#ifdef LIGHT
SetMaterial(mat_specular2, mat_ambient2,
mat_diffuse2, mat_shininess2);
#endif
glColor3f(0.0, 1.0, 0.0);//rocket port color
glScalef(0.4, 0.4, 0.4);
glRotatef(45.0, 0.0, 0.0, 1.0);
glTranslatef(1.0, 0.0, 0.0);
Box(2.0, 0.5, 3.0, solid);
glTranslatef(1.0, 0.0, 0.0);
glRotatef(45.0, 0.0, 0.0, 1.0);
```

```
glTranslatef(0.5, 0.0, 0.0);
Box(1.2, 0.5, 3.0, solid);
glTranslatef(2.1, 0.0, 0.0);
glRotatef(-90.0, 0.0, 0.0, 1.0);
#ifdef LIGHT
SetMaterial(mat_specular,
mat_ambient, mat_diffuse, mat_shininess);
#endif
glColor3f(1.0, 1.0, 0.0);
Box(2.0, 3.0, 4.0, solid);
glTranslatef(-0.5, -1.0, 1.3);
for (i = 0; i < 2; i++) {
for (j = 0; j < 3; j++) {
if (!solid) {
gluQuadricDrawStyle(qobj, GLU_LINE);
}
glTranslatef(i, j, 0.6);
#ifdef LIGHT
SetMaterial(mat_specular3,
mat_ambient3, mat_diffuse3, mat_shininess3);
#endif
glColor3f(1.0, 1.0, 1.0);
gluCylinder(qobj, 0.4, 0.4, 0.3, 16, 10);
glTranslatef(0.0, 0.0, 0.3);
#ifdef LIGHT
SetMaterial(mat_specular4,
mat_ambient4, mat_diffuse4, mat_shininess4);
#endif
glColor3f(0.0, 1.0, 0.0);
gluCylinder(qobj, 0.4, 0.0, 0.5, 16, 10);
k++;
glTranslatef(-i, -j, -0.9);
}
}
glEndList();
}
void Enviro(char solid)
{
int i, j;
glNewList(SOLID_ENVIRO, GL_COMPILE);
SetMaterial(mat_specular4, mat_ambient4,
mat_diffuse4, mat_shininess4);
glColor3f(1.0, 1.0, 0.0);//out line of the walking
path
Box(20.0, 0.5, 30.0, solid);
SetMaterial(mat_specular4, mat_ambient3,
mat_diffuse2, mat_shininess);
glColor3f(1.0, 0.0, 0.0);//the surrounding area
color
glTranslatef(0.0, 0.0, -10.0);
for (j = 0; j < 6; j++) {
for (i = 0; i < 2; i++) {
if (i)
glScalef(-1.0, 1.0, 1.0);
glTranslatef(10.0, 4.0, 0.0);
Box(4.0, 8.0, 2.0, solid);
glTranslatef(0.0, -1.0, -3.0);
Box(4.0, 6.0, 2.0, solid);
glTranslatef(-10.0, -3.0, 3.0);
}
glScalef(-1.0, 1.0, 1.0);
glTranslatef(0.0, 0.0, 5.0);
}  glEndList();
}
void Toggle(void)
{
if (solid_part)
solid_part = 0;
else
solid_part = 1;
}
void disable(void)
{
glDisable(GL_LIGHTING);
glDisable(GL_DEPTH_TEST);
glDisable(GL_NORMALIZE);
glPolygonMode(GL_FRONT_AND_BACK,
GL_LINE);
}
void lighting(void)
{
GLfloat position[] =
{0.0, 0.0, 2.0, 1.0};
#ifdef MOVE_LIGHT
glRotatef((GLfloat) lightturn1, 1.0, 0.0, 0.0);
glRotatef((GLfloat) lightturn, 0.0, 1.0, 0.0);
glRotatef(0.0, 1.0, 0.0, 0.0);
#endif
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glEnable(GL_NORMALIZE);
glDepthFunc(GL_LESS);
glPolygonMode(GL_FRONT_AND_BACK,
GL_FILL);
glLightfv(GL_LIGHT0,
GL_POSITION, position);
glLightf(GL_LIGHT0,
GL_SPOT_CUTOFF, 80.0);
glTranslatef(0.0, 0.0, 2.0);
glDisable(GL_LIGHTING);
Box(0.1, 0.1, 0.1, 0);
glEnable(GL_LIGHTING);
}
void DrawMech(void)
{
int i, j;
glScalef(0.5, 0.5, 0.5);
glPushMatrix();
glTranslatef(0.0, -0.75, 0.0);
glRotatef((GLfloat) tilt, 1.0, 0.0, 0.0);
glRotatef(90.0, 1.0, 0.0, 0.0);
#ifdef HIP
glCallList(SOLID_MECH_HIP);
#endif
glRotatef(-90.0, 1.0, 0.0, 0.0);
glTranslatef(0.0, 0.75, 0.0);
glPushMatrix();
glRotatef((GLfloat) pivot, 0.0, 1.0, 0.0);
glPushMatrix();
#ifdef TORSO
glCallList(SOLID_MECH_TORSO);
#endif
glPopMatrix();
glPushMatrix();
glTranslatef(0.5, 0.5, 0.0);
#ifdef ROCKET_POD
glCallList(SOLID_MECH_ROCKET);
#endif
glPopMatrix();
for (i = 0; i < 2; i++) {
glPushMatrix();
if (i)
glScalef(-1.0, 1.0, 1.0);
glTranslatef(1.5, 0.0, 0.0);
#ifdef SHOULDER
glCallList(SOLID_MECH_SHOULDER);
#endif
glTranslatef(0.9, 0.0, 0.0);
if (i) {
glRotatef((GLfloat) lat1, 0.0, 0.0, 1.0);
glRotatef((GLfloat) shoulder1, 1.0, 0.0, 0.0);
glRotatef((GLfloat) shoulder3, 0.0, 1.0, 0.0);
} else {
glRotatef((GLfloat) lat2, 0.0, 0.0, 1.0);
glRotatef((GLfloat) shoulder2, 1.0, 0.0, 0.0);
glRotatef((GLfloat) shoulder4, 0.0, 1.0, 0.0);
}
glTranslatef(0.0, -1.4, 0.0);
#ifdef UPPER_ARM
glCallList(SOLID_MECH_UPPER_ARM);
#endif
glTranslatef(0.0, -2.9, 0.0);
if (i)
glRotatef((GLfloat) elbow1, 1.0, 0.0, 0.0);
else
glRotatef((GLfloat) elbow2, 1.0, 0.0, 0.0);
glTranslatef(0.0, -0.9, -0.2);
#ifdef LOWER_ARM
glCallList(SOLID_MECH_FOREARM);
glPushMatrix();
glTranslatef(0.0, 0.0, 2.0);
glRotatef((GLfloat) fire, 0.0, 0.0, 1.0);
glCallList(SOLID_MECH_VULCAN);
glPopMatrix();
#endif
glPopMatrix();
}
glPopMatrix();

glPopMatrix();

for (j = 0; j < 2; j++) {
glPushMatrix();
if (j) {
glScalef(-0.5, 0.5, 0.5);
leg = 1;
} else {
glScalef(0.5, 0.5, 0.5);
leg = 0;
}
glTranslatef(2.0, -1.5, 0.0);
if (j) {
glRotatef((GLfloat) hip11, 1.0, 0.0, 0.0);
glRotatef((GLfloat) hip12, 0.0, 0.0, 1.0);
} else {
glRotatef((GLfloat) hip21, 1.0, 0.0, 0.0);
glRotatef((GLfloat) hip22, 0.0, 0.0, 1.0);
}
glTranslatef(0.0, 0.3, 0.0);
#ifdef UPPER_LEG
glPushMatrix();
glCallList(SOLID_MECH_UPPER_LEG);
glPopMatrix();
#endif
glTranslatef(0.0, -8.3, -0.4);
if (j)
glRotatef((GLfloat) - hip12, 0.0, 0.0, 1.0);
else
glRotatef((GLfloat) - hip22, 0.0, 0.0, 1.0);
glTranslatef(-0.5, -0.85, -0.5);
#ifdef LOWER_LEG
LowerLeg(1);
#endif
glPopMatrix();
}
}
void display(void)
{
glClearColor(0.0, 0.0, 0.0, 0.0);
glClear(GL_COLOR_BUFFER_BIT |
GL_DEPTH_BUFFER_BIT);
glEnable(GL_DEPTH_TEST);
glPushMatrix();
glRotatef((GLfloat) turn, 0.0, 1.0, 0.0);
glRotatef((GLfloat) turn1, 1.0, 0.0, 0.0);
#ifdef LIGHT
if (solid_part) {
glPushMatrix();
lighting();
glPopMatrix();
```

```
} else
disable();
#endif
glPopMatrix();
glLoadName(TEXTID);
glColor3f(1,1,0);
DrawTextXY(-2.5,0.2,2.0,0.0015,"1KI20CS100");
DrawTextXY(-2.5,0.5,2.0,0.0015,
"SINCHANA S");
DrawTextXY(2.5,2.2,-2.0,0.0025,"1KI20CS104");
DrawTextXY(2.5,2.7,-2.0,0.0030,
"SPOORTHY N");
glFlush();
glutSwapBuffers();
}
void myinit(void)
{
char i = 1;
qobj = gluNewQuadric();
#ifdef LIGHT
SetMaterial(mat_specular2,
mat_ambient2, mat_diffuse2, mat_shininess2);
#endif
glEnable(GL_DEPTH_TEST);
MechTorso(i);
MechHip(i);
Shoulder(i);
RocketPod(i);
UpperArm(i);
ForeArm(i);
UpperLeg(i);
Foot(i);
VulcanGun(i);
Enviro(i);
}
void myReshape(int w, int h)
{
glViewport(0, 0, w, h);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(65.0, (GLfloat) w / (GLfloat) h,
1.0, 20.0);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(0.0, 1.2, -5.5);
}
#ifdef ANIMATION
void animation_walk(void)
{
float angle;
static int step;
if (step == 0 || step == 2) {
if (frame >= 0.0 && frame <= 21.0) {
if (frame == 0.0)
frame = 3.0;
angle = (180 / M_PI) * (acos(((cos((M_PI / 180) *
frame) * 2.043) + 1.1625) / 3.2059));
if (frame > 0) {
elevation = -(3.2055 - (cos((M_PI / 180) * angle) *
3.2055));
} else
elevation = 0.0;
if (step == 0) {
hip11 = -(frame * 1.7);
if (1.7 * frame > 15)
heel1 = frame * 1.7;
heel2 = 0;
ankle1 = frame * 1.7;
if (frame > 0)
hip21 = angle;
else
hip21 = 0;
ankle2 = -hip21;
shoulder1 = frame * 1.5;
shoulder2 = -frame * 1.5;
elbow1 = frame;
elbow2 = -frame;
} else {
hip21 = -(frame * 1.7);
if (1.7 * frame > 15)
heel2 = frame * 1.7;
heel1 = 0;
ankle2 = frame * 1.7;
if (frame > 0)
hip11 = angle;
else
hip11 = 0;
ankle1 = -hip11;
shoulder1 = -frame * 1.5;
shoulder2 = frame * 1.5;
elbow1 = -frame;
elbow2 = frame;
}
if (frame == 21)
step++;
if (frame < 21)
frame = frame + 3.0;
}
}
if (step == 1 || step == 3) {
if (frame <= 21.0 && frame >= 0.0) {
angle = (180 / M_PI) * (acos(((cos((M_PI / 180) *
frame) * 2.043) + 1.1625) / 3.2029));
if (frame > 0)
elevation = -(3.2055 - (cos((M_PI / 180) * angle) *
3.2055));
else
elevation = 0.0;
if (step == 1) {
elbow2 = hip11 = -frame;
elbow1 = heel1 = frame;
heel2 = 15;
ankle1 = frame;
if (frame > 0)
hip21 = angle;
else
hip21 = 0;
ankle2 = -hip21;
shoulder1 = 1.5 * frame;
shoulder2 = -frame * 1.5;
} else {
elbow1 = hip21 = -frame;
elbow2 = heel2 = frame;
heel1 = 15;
ankle2 = frame;
if (frame > 0)
hip11 = angle;
else
hip11 = 0;
ankle1 = -hip11;
shoulder1 = -frame * 1.5;
shoulder2 = frame * 1.5;
}
if (frame == 0.0)
step++;
if (frame > 0)
frame = frame - 3.0;
}
}
if (step == 4) step = 0;
distance += 0.1678;
glutPostRedisplay();
}
void animation(void)
{
animation_walk();
}
#endif
#ifdef GLUT
#ifdef GLUT_KEY
void keyboard(unsigned char key, int x, int y)
{
int i = 0;
if (key == 27) exit (0);
switch (key) {
case 'q':{
shoulder2Subtract();
i++;  i++;
} break;
case 'a':{
shoulder2Add();  i++;
} break;
case 'w':{
shoulder1Subtract(); i++;
}  break;
case 's':{
shoulder1Add(); i++;
}break;
case '2':{
shoulder3Add();  i++;
}  break;
case '1':{
shoulder4Add(); i++;
} break;
case '4':{
shoulder3Subtract(); i++;
}break;
case '3':{
shoulder4Subtract();i++;
} break;
case 'z':{
lat2Raise();i++;
} break;
case 'Z':{
lat2Lower(); i++;
}  break;
case 'x':{
lat1Raise(); i++;
} break;
case 'X':{
lat1Lower();i++;
} break;

case 'A':{
elbow2Add();i++;
} break;
case 'Q':{
elbow2Subtract(); i++;
} break;
case 'S':{
elbow1Add(); i++;
} break;
case 'W':{
elbow1Subtract();i++;
}break;
case 'd':{
RotateAdd();i++;
} break;
case 'g':{
RotateSubtract(); i++;
} break;
case 'r':{
MechTiltAdd();  i++;
}  break;
case 'f':{
MechTiltSubtract();  i++;
}break;
case 'h':{
RaiseLeg2Forward();  i++;
}break;
```

```
case 'y':{
LowerLeg2Backwards();
i++;
} break;
case 'Y':{
RaiseLeg2Outwards();i++;
}break;
case 'H':{
LowerLeg2Inwards(); i++;
} break;
case 'j':{
RaiseLeg1Forward();  i++;
}break;
case 'u':{
LowerLeg1Backwards(); i++;
}  break;
case 'U':{
RaiseLeg1Outwards();  i++;
} break;
case 'J':{
LowerLeg1Inwards(); i++;
}  break;
case 'N':{
Heel2Add();  i++;
}   break;
case 'n':{
Heel2Subtract();  i++;
}  break;
case 'M':{
Heel1Add();  i++;
}  break;
case 'm':{
Heel1Subtract();  i++;
}   break;
case 'k':{
Ankle2Add();   i++;
}   break;
case 'K':{
Ankle2Subtract(); i++;
} break;
case 'l':{
Ankle1Add();   i++;
}   break;
case 'L':{
Ankle1Subtract();   i++;
}  break;
case 'p':{
LightTurnRight();  i++;
} break;
case 'i':{
LightTurnLeft(); i++;
}  break;
case 'o':{
LightForwards();   i++;
}  break;
case '9':{
LightBackwards();   i++;
}  break;
} if (i)
glutPostRedisplay();
}
#endif
#ifdef GLUT_SPEC
void special(int key, int x, int y)
{
int i = 0;
switch (key) {
case GLUT_KEY_RIGHT:{
TurnRight();i++;
}
break;
case GLUT_KEY_LEFT:{
TurnLeft();  i++;
}   break;
case GLUT_KEY_DOWN:{
TurnForwards(); i++;
break;
case GLUT_KEY_UP:{
TurnBackwards();   i++;
} break;
case GLUT_KEY_PAGE_UP:{
FireCannon();   i++;
}  break;
}
if (i)
glutPostRedisplay();
}
#endif
#endif
void menu_select(int mode)
{
switch (mode) {
#ifdef ANIMATION
case 1:
glutIdleFunc(animation);   break;
#endif
case 2:
glutIdleFunc(NULL);   break;
case 3:
Toggle();
glutPostRedisplay();   break;
case 4:
exit(EXIT_SUCCESS);
}
}
void null_select(int mode)
{
}
void glutMenu(void)
{
```

```
int glut_menu[13];
glut_menu[5] = glutCreateMenu(null_select);
glutAddMenuEntry("forward       : q,w", 0);
glutAddMenuEntry("backwards     : a,s", 0);
glutAddMenuEntry("outwards      : z,x", 0);
glutAddMenuEntry("inwards       : Z,X", 0);
glut_menu[6] = glutCreateMenu(null_select);
glutAddMenuEntry("upwards       : Q,W", 0);
glutAddMenuEntry("downwards     : A,S", 0);
glutAddMenuEntry("outwards      : 1,2", 0);
glutAddMenuEntry("inwards       : 3,4", 0);
glut_menu[1] = glutCreateMenu(null_select);
glutAddMenuEntry(" : Page_up", 0);
glut_menu[8] = glutCreateMenu(null_select);
glutAddMenuEntry("forward       : y,u", 0);
glutAddMenuEntry("backwards     : h.j", 0);
glutAddMenuEntry("outwards      : Y,U", 0);
glutAddMenuEntry("inwards       : H,J", 0);
glut_menu[9] = glutCreateMenu(null_select);
glutAddMenuEntry("forward       : n,m", 0);
glutAddMenuEntry("backwards     : N,M", 0);
glut_menu[10] = glutCreateMenu(null_select);
glutAddMenuEntry("toes up       : K,L", 0);
glutAddMenuEntry("toes down     : k,l", 0);
glut_menu[11] = glutCreateMenu(null_select);
glutAddMenuEntry("right         : right arrow", 0);
glutAddMenuEntry("left          : left arrow", 0);
glutAddMenuEntry("down          : up arrow", 0);
glutAddMenuEntry("up            : down arrow", 0);
glut_menu[12] = glutCreateMenu(null_select);
glutAddMenuEntry("right         : p", 0);
glutAddMenuEntry("left          : i", 0);
glutAddMenuEntry("up            : 9", 0);
glutAddMenuEntry("down          : o", 0);
glut_menu[4] = glutCreateMenu(NULL);
glutAddSubMenu("at the shoulders? ",
glut_menu[5]);
glutAddSubMenu("at the elbows?",
glut_menu[6]);
glut_menu[7] = glutCreateMenu(NULL);
glutAddSubMenu("at the bottompart? ",
glut_menu[8]);
glutAddSubMenu("at the knees?", glut_menu[9]);
glutAddSubMenu("at the ankles? ",
glut_menu[10]);
glut_menu[2] = glutCreateMenu(null_select);
glutAddMenuEntry("turn left    : d", 0);
glutAddMenuEntry("turn right    : g", 0);
glutAddMenuEntry("Rocketpod    : v", 0);
glut_menu[3] = glutCreateMenu(null_select);
glutAddMenuEntry("tilt backwards : f", 0);
glutAddMenuEntry("tilt forwards  : r", 0);
glut_menu[0] = glutCreateMenu(NULL);
glutAddSubMenu("move        the        arms..        ",
glut_menu[4]);
glutAddSubMenu("fire the vulcan guns?",
glut_menu[1]);
glutAddSubMenu("move the legs.. ",
glut_menu[7]);
glutAddSubMenu("move the torso?",
glut_menu[2]);
glutAddSubMenu("move the upper portion?",
glut_menu[3]);
glutAddSubMenu("rotate the scene..",
glut_menu[11]);
#ifdef MOVE_LIGHT
glutAddSubMenu("rotate the light source..",
glut_menu[12]);
#endif
glutCreateMenu(menu_select);
#ifdef ANIMATION
glutAddMenuEntry("Start Walk", 1);
glutAddMenuEntry("Stop Walk", 2);
#endif
glutAddMenuEntry("Toggle Wireframe", 3);
glutAddSubMenu("How do I ..", glut_menu[0]);
glutAddMenuEntry("Quit", 4);
glutAttachMenu(GLUT_LEFT_BUTTON);
glutAttachMenu(GLUT_RIGHT_BUTTON);
}
int main(int argc, char **argv)
{
#ifdef GLUT
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_DOUBLE |
GLUT_RGBA | GLUT_DEPTH);
glutInitWindowSize(1000, 1000);
glutCreateWindow("glutmech: Vulcan Gunner");
myinit();
glutDisplayFunc(display);
glutReshapeFunc(myReshape);
#ifdef GLUT_KEY
glutKeyboardFunc(keyboard);
#endif
#ifdef GLUT_SPEC
glutSpecialFunc(special);
#endif
glutMenu();
glPointSize(2.0);
glutMainLoop();
#endif
return 0;
return int.
}
```

CHAPTER 6

RESULTS

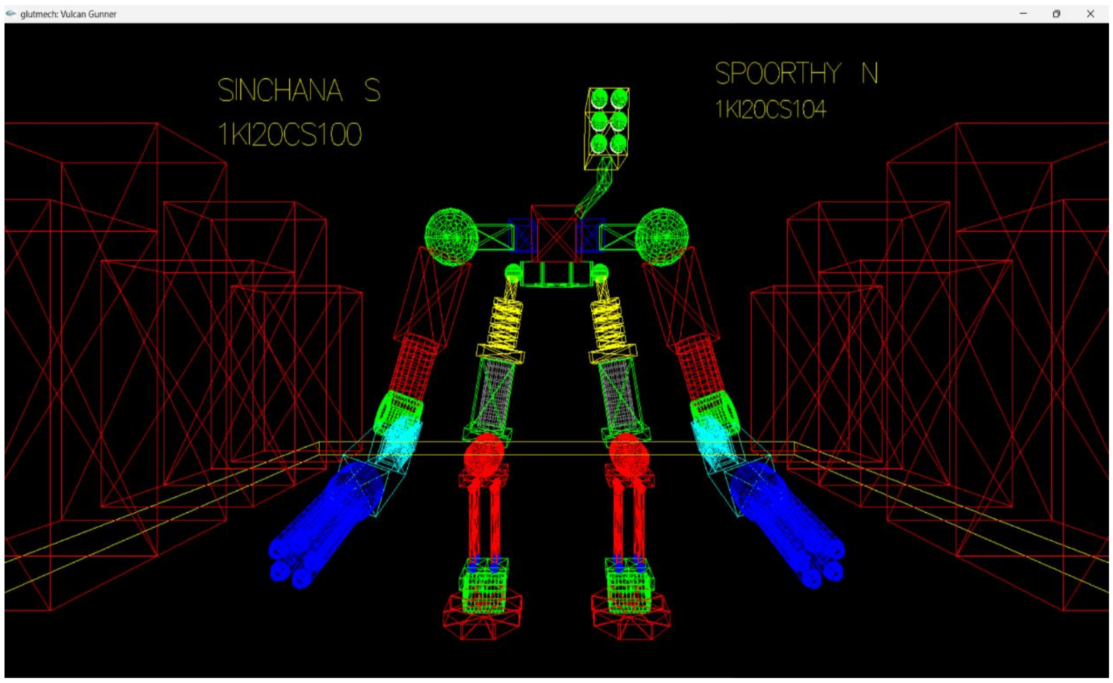


Fig 6.a Robot Walking Simulation

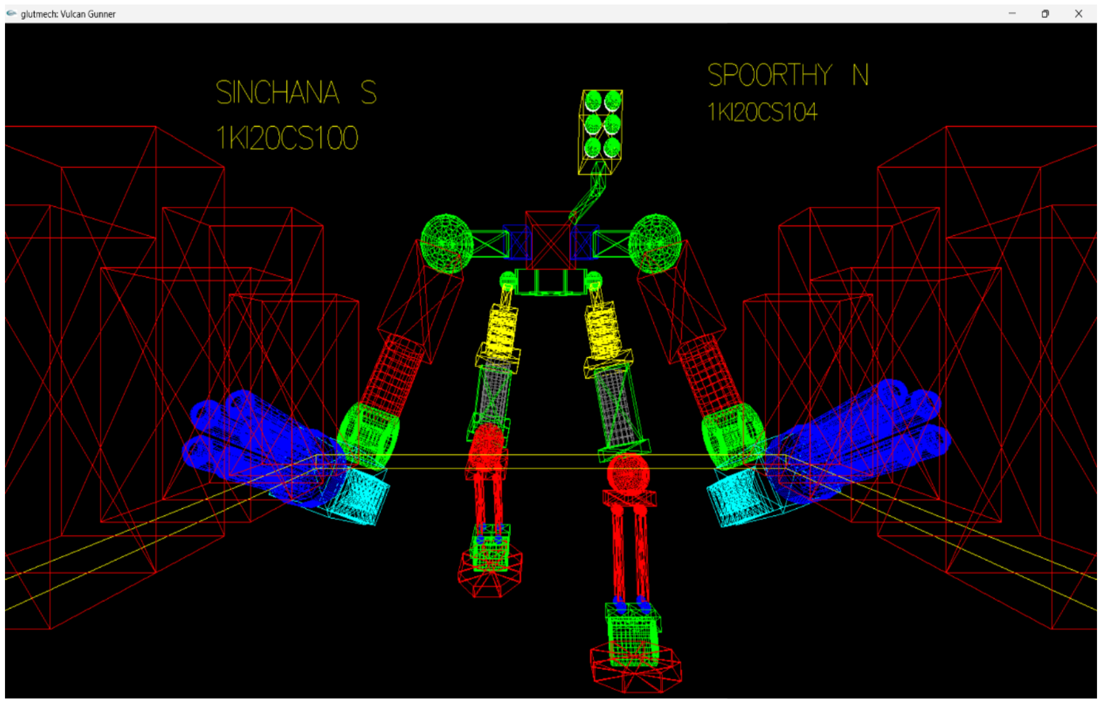


Fig 6.b Robot with Torso movement

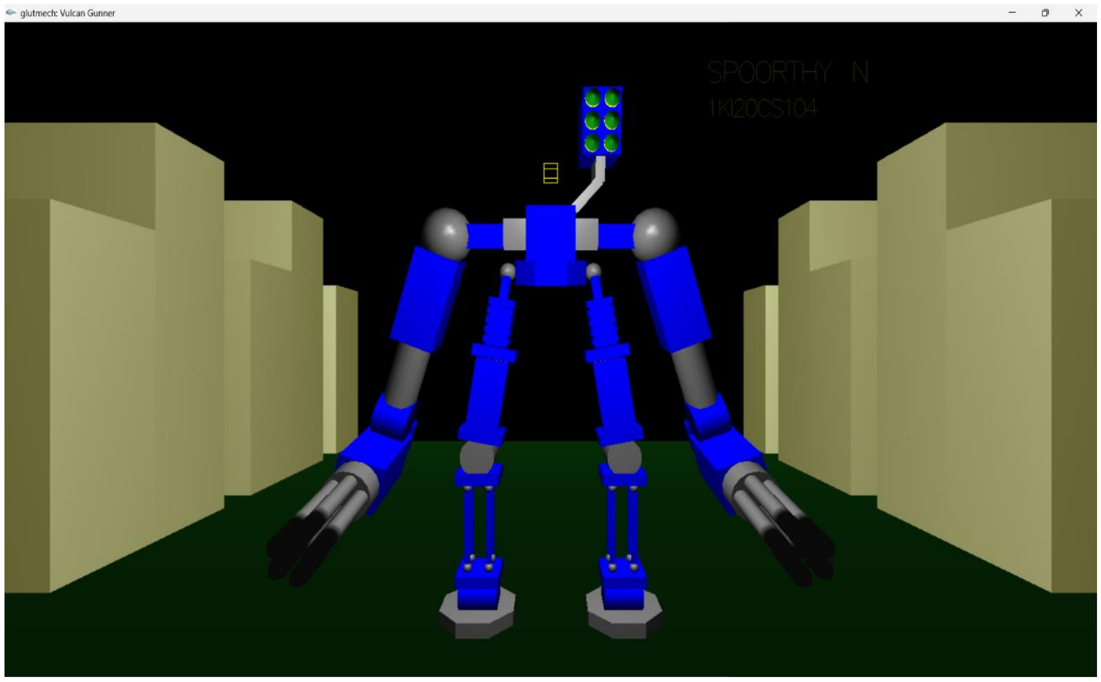


Fig 6.c Robot in Toggle Wireframe

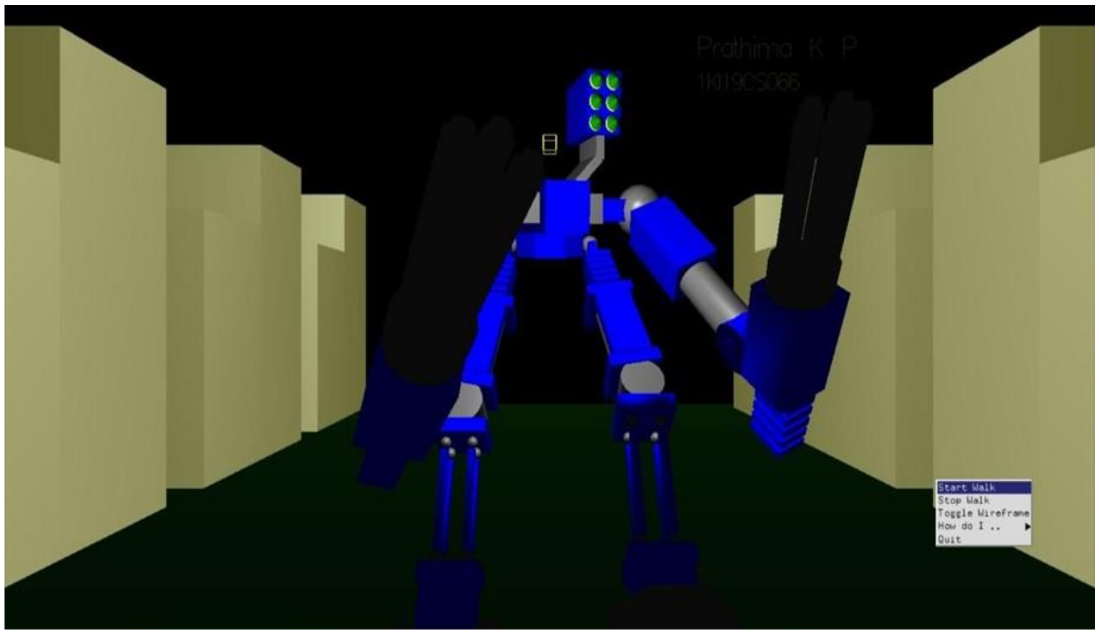


Fig 6.d Robot Control

CHAPTER 7

CONCLUSION

This is very reliable graphics package supporting various primitive objects like polygon, line loops, etc. Also, color selection, mouse, keyboard based interface are included. Transformations like translation, rotation is provided. This project has been implemented using optimized algorithms and thus is fast. Special attention has been provided to the interfaces that make its use comfortable. We hope this project proves to be flexible in all respects to one and all. It utilizes various inbuilt functions provided by the C with OpenGL language. An effort has been to keep the software easy to use and understand.

The project has been tested efficiently. By this project we explored vast vistas of knowledge in the fields of ROBOTICS and its operations. It also provided valuable experience on BatronixProg. Studio etc., which are being currently used in the various fields. We became aware of challenges, work criterion, teamwork and other activities performed during the project analysis and implementation. The exercise has helped us to gain a lot of technical and practical knowledge. We are sure that it will serve as an important experience in our professional career

BIBLIOGRAPHY

- **Books:**

1. Computer graphics-Edward angel, 5th Edition.
2. An introduction to graphics programming with openGL-Toby Howard,2nd Edition.
3. Computer graphics-openGL Version-Donald Hearn,2nd Edition.

- **Links:**

1. https://en.wikipedia.org/wiki/Computer_graphics - History of Computer Graphics
2. https://en.wikipedia.org/wiki/OpenGL - History of OpenGL
3. https://en.wikipedia.org/wiki/OpenGL - About OpenGL
4. https://en.wikipedia.org/wiki/OpenGL_Utility_Library - About GLU
5. https://en.wikipedia.org/wiki/OpenGL_Utility_Toolkit - About GLUT
6. https://www.khronos.org/opengl/wiki/Primitive - About OpenGL Primitives
7. https://www.techtarget.com/searchenterpriseai/definition/robot - System Architecture