**Chapter 1**

**Preamble**

* 1. **Introduction**

Pictorial synthesis of real/imaginary objects from computer-based models is Computer Graphics. Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and efficiently. Interactive computer graphics is the most important means of producing pictures since the invention of photography and television. It has the added advantage that, with the computer, we can make pictures not only of concrete real world objects but also of abstract, synthetic objects, such as mathematical surfaces and of data that have no inherent geometry, such as survey results.[1]

* + 1. **History of Computer Graphics**

Computer Graphics is the creation, manipulation, and storage of models and images of picture objects by the aid of computers. This was started with the display of data on plotters and CRT. Computer Graphics is also defined as the study of techniques to improve the communication between user and machine, thus Computer Graphics is one of the most effective medium of communication between machine and user.

William fetter was credited with coning the term Computer Graphics in 1960, to describe his work at Boeng. One of the first displays of computer animation was future world (1976), which included an animation of a human face and hand-produced by Carmull and Fred Parkle at the University of Utah.

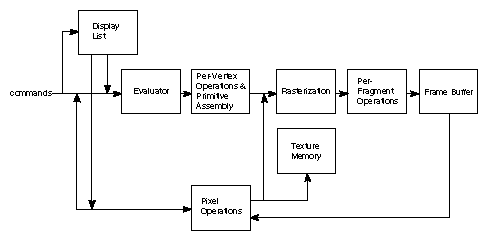
There are several international conferences and journals where the most significant results in computer-graphics are published. Among them are the SIGGRAPH and Euro graphics conferences and the association for computing machinery (ACM) transaction on Graphics journals.[5]

* + 1. **Introduction to Open GL**

As a software interface for graphics hardware, OpenGL's main purpose is to render two and three-dimensional objects into a frame buffer. These objects are described as sequences of vertices (which define geometric objects) or pixels (which define images). OpenGL performs several processing steps on this data to convert it to pixels to form the final desired image in the frame buffer.*[3]*

OpenGL draws *primitives*—points, line segments, or polygons—subject to several selectable modes. Primitives are defined by a group of one or more *vertices*. A vertex defines a point, an endpoint of a line, or a corner of a polygon where two edges meet. Data (consisting of vertex coordinates, colors, normal, texture coordinates, and edge flags) is associated with a vertex, and each vertex and its associated data are processed independently, in order, and in the same way.*[3]*

The figure shown below gives an abstract, high-level block diagram of how OpenGL processes data. In the diagram, commands enter from the left and proceed through what can be thought of as a processing pipeline. Some commands specify geometric objects to be drawn, and others control how the objects are handled during the various processing stages.



**Figure 1.1**: OpenGL Block Diagram

We can choose to accumulate some of commands in a *display list* for processing at a later time. The *evaluator* stage of processing provides an efficient means for approximating curve and surface geometry by evaluating polynomial commands of input values.*Rasterization* produces a series of frame buffer addresses and associated values using a two-dimensional description of a point, line segment, or polygon. Each *fragment* so produced is fed into the last stage, *per-fragment operations*, which perform the final operations on the data before it's stored as pixels in the *frame buffer.[3]*

* 1. **Objectives**

Objectives of Dijkstra in Network Simulation are

* To demonstrate Dijkstra algorithm in network simulation.
* To find the shortest distance from the source to destination which are given by user
  1. **Organization of the Report**

Chapter 1 provides the information about the basics of OpenGL. In Chapter 2, all the OpenGL functions used in our program is described. Chapter 3 gives the idea of the project and its actual implementation. Chapter 4 discusses about the testing and limitations of the program. Chapter 5 concludes by giving the direction for future enhancement.

**1.4 Summary**

The chapter discussed before is an overview about the computer graphics, its history and OpenGL interface. It even includes the OpenGL block diagram. The scope of study and objectives of the project are mentioned clearly. The organization of the report is been pictured to increase the readability. Further, coming up chapter depicts the OpenGLbuilt-in functions used in project source code.

**Chapter 2**

**A Preview of OpenGL Functions**

When we start drawing any graphics in OpenGL using C language we need a header file called <GL/glut.h>. The header file contains definitions and explanations of all functions and constants we’ll need, whereas the graphics functions are kept in the library file. Both these files are provided as a part of TURBO C compliers.

* **stdio.h:**

This is a standard input header file which is used in any program. This file contains all the built-in functions like printf(),scanf(),fopen(),fclose() etc.It also contains data types and global variables. Some of the examples are BUFSIZ, EOF, and NULL etc.

* **stdlib.h:**

This is also one the standard library header file which contains the entire standard library functions like exit, free, alloc, mallocetc.Some of the constants and data types are NULL,size\_t etc.

* **GL/glut.h :**

This is very familiar library function of visual basic graphics library. This header file contains so many numbers of built in functions of a graphics library.

* **math.h :**

This is library file used for mathematical operations.

The different OpenGL functions[2] used in our project are described as follows:

* **Name:**glBegin()

**Cspecification**: glBegin(glEnum mode)

**Description**: Initiates a new primitive of type mode and starts collection of vertices. Values of mode include GL\_POINTS, GL\_LINES and GL\_POLYGON

* **Name:**glEnd()

C **specification**: void glEnd()

**Description**: Terminates a list of vertices.

* **Name:**glPushMatrix()

**Cspecification**: void glPushMatrix( )

**Description**: Pushes and pops from the attributes stack.

* **Name:**glPopMatrix( )

**Cspecification**: void glPopMatrix( )

**Description**: All attributes are poped from the stack.

* **Name:** glutInit( )

**Cspecification**: void glutInit ( )

**Description**: All Initializes the GLUT. The arguments from main are passed in and can be used by the application.

* **Name:**glutCreateWindow( )

**Cspecification**: void glutCreateWindow( )

**Description**: creates a window on the display. The string title can be used to label the window.

* **Name:**glutDisplayMode()

**CSpecification**: void glutDisplayMode()

**Description**: Requests a display with the properties in mode. The mode is determined by the logical OR of options including the color model.

* **Name:**glutWindowSize( )

**CSpecification**: void glutWindowSize( )

**Description**: Specifies the initial height and width of the window in pixels.

* **Name:** glutInitwindowPosition ()

**CSpecification**: void glutInitwindowPosition(int x, int y)

**Description**: Specifies the initial position of the top-left corner of the window in pixels.

* **Name:**glutMainLoop()

**CSpecification**: Void glutMainLoop()

**Description**: Cause the program to enter an event-processing loop.It should be the last statement main.

* **Name**: glutDisplayFunc()

**C Specification**: Void glutDisplayFunc(void(\*func)(void))

**Description**: registers the display functions func that is executed when the window must to be redrawn.

* **Name:**glutPostRedisplay()

**CSpecification:** void glutPostRedisplay()

**Description**: Requests that the display callback be executed after the current callback returns.

* **Name:**glMatrixMode( )

**C Specification**: void glMatrixMode(GLenum mode)

**Description**: Specify matrix will be affected by subsequent transformation mode can be GL\_MODELVIEW, GL\_PROJECTION.

* **Name:**glTranslatef( )

C **specification:** void glTranslatef(GLfloat*x*, GLfloat*y*, GLfloat*z* )

**Description**: glTranslatef produces a translation by (x, y, z).

* **Name**: glut keyboard Func( )

**Cspecification**: void key (unsigned char keys, int x, int y)

**Description**: sets the keyboard interaction with the current window.

* **Name:**glClear()

**Cspecification:**glClear(GL\_COLOR\_ BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT**);**

**Description:** to refresh the color buffer and depth buffer so that ifalgorithm stores information in the depth buffer, we must clear this buffer whenever we wish to redraw the display.

* **Name:**glutSwapBuffers()

**C specification:** glutSwapBuffers()

**Description:** Swaps the buffers of the current window if double buffered Performs a buffer swap on the layer in use for the current window. Specifically, glutSwapBuffers promotes the contents of the back buffer of the layer in use of the current window to become the contents of the front buffer. The contents of the back buffer then become undefined. The update typically takes place during the vertical retrace of the monitor, rather than immediately after glutSwapBuffers is called.

* **Name:** glFlush()

**C specification:** glFlush()

**Description:**the call to glFlush ensures that points are rendered to the screen.

* **Name:** glutIdleFunc ()

**C specification**: voidglutIdleFunc (void (\*f) (void))

**Description:**It is used to register the display callback function ‘f’ that is executed whenever there are no other events to be handled.

* **Name:** glOrtho()

**Cspecification:**voidglOrtho(GLdoubleleft,GLdoubleright, GLdoublebottom,GLdoubletop,GLdoublenear,GLdouble far)

**Description:**This function defines an orthographic viewing volume with all parameters measured from the center of the projection plane.

* **Name:** void glutBitmapCharacter()

**C specification**: void glutBitmapCharacter(void \*font, int character)

**Description**:glutBitmapCharacter renders the character in the named bitmap font

* **Name:**glRasterPos3f ()

**C specification:** void **glRasterPos3f**(GLfloat x, GLfloat y, GLfloat z);

**Descrciption:**It specify the raster position for pixel operations

**Chapter 3**

**System Design and Implementation**

**3.1 Introduction**

Systems design is the process or art of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development.[4]

This Project is implemented using OpenGl, which is proven to be a very efficient tool in the field of computer graphics, programming is done under linux platform. Glut.h library is used to create the objects and to translate them. C programming language is used to implement the entire code. Interface to the program is provided with the help of input device keyboard.

The objects in Dijkstra in Network Simualtion are created using **primitive** functions.The nodes is created using GL\_POLYGON, path between the nodes is created using GL\_LINES , the shortest path between source and destination is created using GL\_LINE\_STRIP .

**3.2 Overall Design Process**

There are few graphical functions used in our project. The keyboard acts as an interface. Keys such as e or E is used to run the program to input the source and destination, keys 1, 2 , 3 ,4 ,5 is used to select the source and destination from the user. Any invalid key pressed will result in terminaton of program with key invalid pressed message on the screen.

**3.2.1 User defined functions:**

* void destpath():it calculates the destination path node by node.
* void dijkstra(): calcutaing the shortest path from the given source and destination given by the user using dijkstra algorithm.
* void callpath():it call both the destpath() and dijkstra() ,to connect all nodes in final path.
* void distanceMatrix():assigning the cost to all the paths between the nodes.
* void rev():reversing the given string s.
* void my\_itoa():function to convert integer to the null terminated string.
* void drawString():it specifies the raster position for the pixel operation and givbes the font to the string.
* void delay():it specifes the time period for which the screen remains.
* void getCube1():it does the cube formation for all five nodes with the specified nodes name in it.
* void drawLine():it specifies the final shortest path line from source to destination.
* void getLines():it checks whether the path exist between the nodes and further draws the line.
* void display2():it displays all the 5 nodes and their path cost.
* void display1():it displays the initial window for few second when program is executed.
* void display4():it displays the final shortest highlighted path between source and destination.
* void dispaly3():it displays the window for specific time period which is before the which final output window.
* void myKey():it specifies the user operations to be performed with the key interactions.
* void display():it call the other display functions in the sequential order
* void init():It is used to set the OpenGLstate variables dealing with viewing and attributes.
* void reshape():It defines what to do when the window is resized.
* void finalCube():it create the highlighted cubes for the final shortest path from source to destination.
* main():This function is used for creating window for display of the model of atom.

**3.2.2 Algorithm for Dijkstra In Network Simulation**

**Step 1: BEGIN**

**Step 2:** Initially, window will open with the 5 node with the mentioned paths with values to travel from each path.

**Step 3:** If the button ‘e’ or ‘E’ is pressed from the keyboard, then window with values to input for source node appears press key ‘1’, ’2’ ,’3’ , ‘4’, ‘5’.

**Step 4:** If the value for node pressed from is valid then it goes to step 5 else it terminates the program with message on window that key pressed is invalid.

**Step 5:** Now the window to input value for the destination node opens with values of nodes ‘1’,’2’,’3’,’4’,’5’.

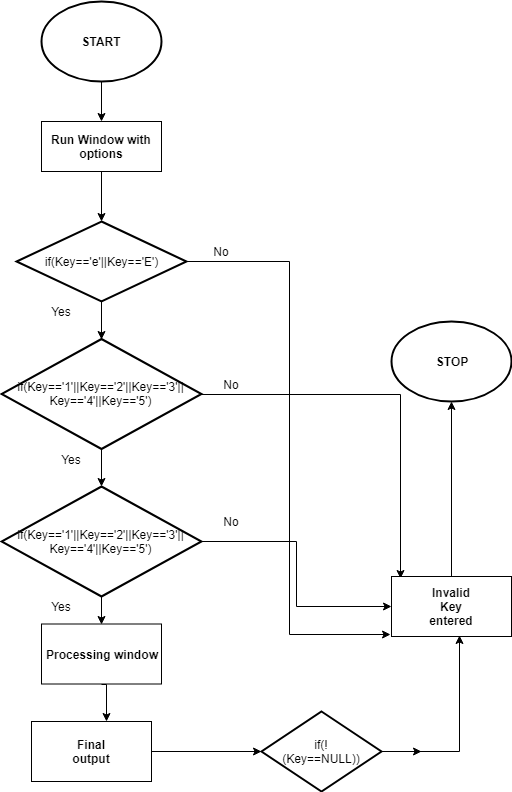
**Step 6**: If the value for node pressed from is valid then it goes to step 7 else it terminates the program with message on window that key pressed is invalid.

**Step 7:** This shows the final window with the calculated shortest path from source to destination.

**Step 8:** Press any key to quit.

**Step 9: END**

**3.3 Flowchart of Dijkstra in Network Simulation**

****

**Figure 3.1**: Flowchart of Dijkstra in Network Simulation

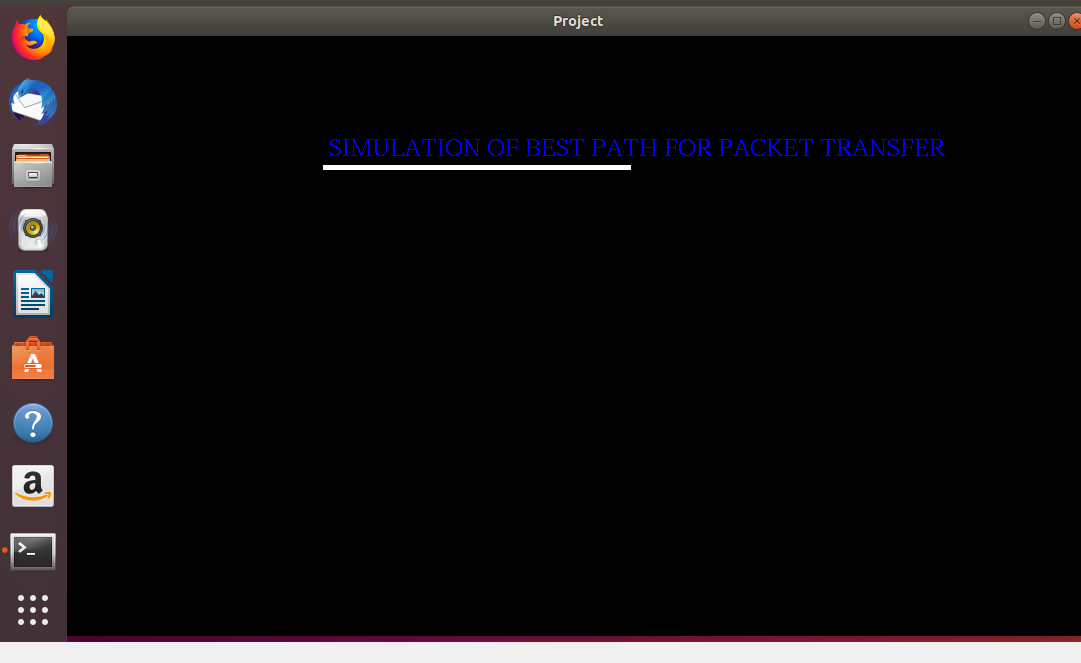
**Chapter 4**

**Results and Discussions**

The project is compiled and executed in Opengl . We have put in few screenshots in here to show the working of Dijkstra Algorithm in network simulation.

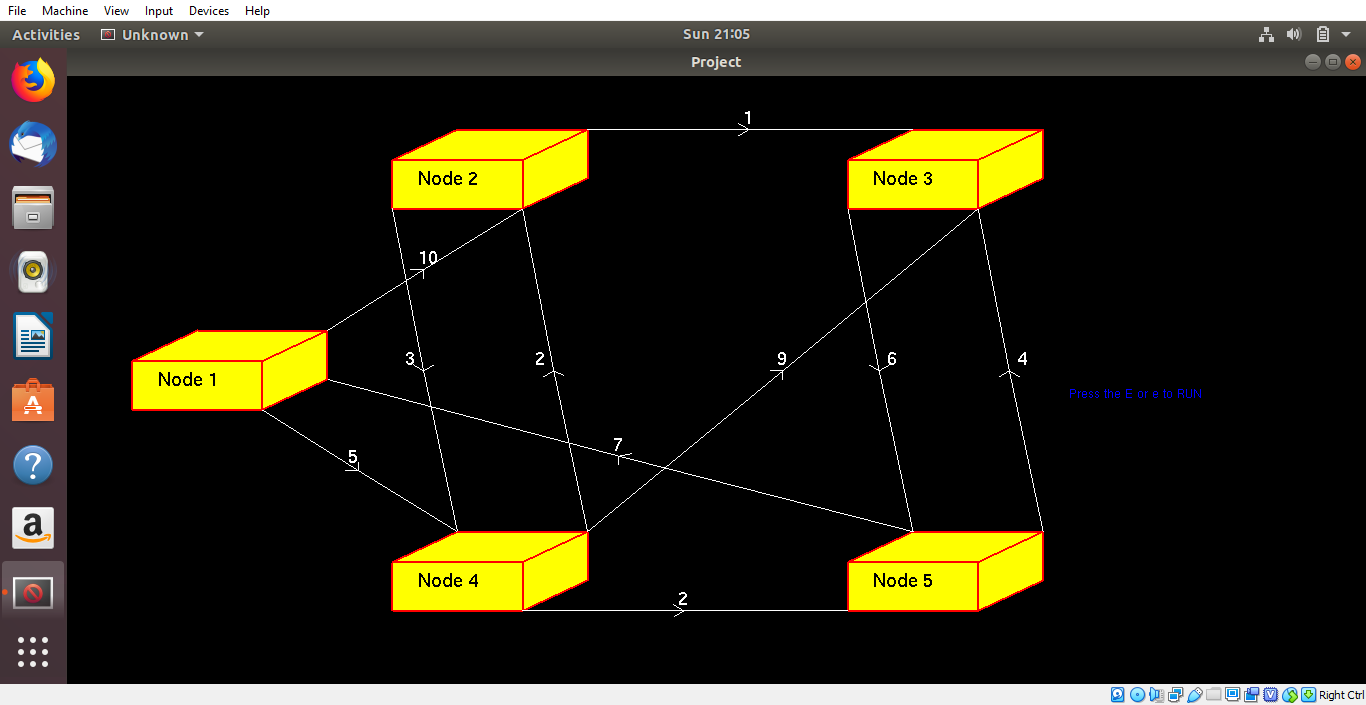
**Snapshots:**

Initial, displayed window for few seconds before showing all 5 nodes with the path cost.

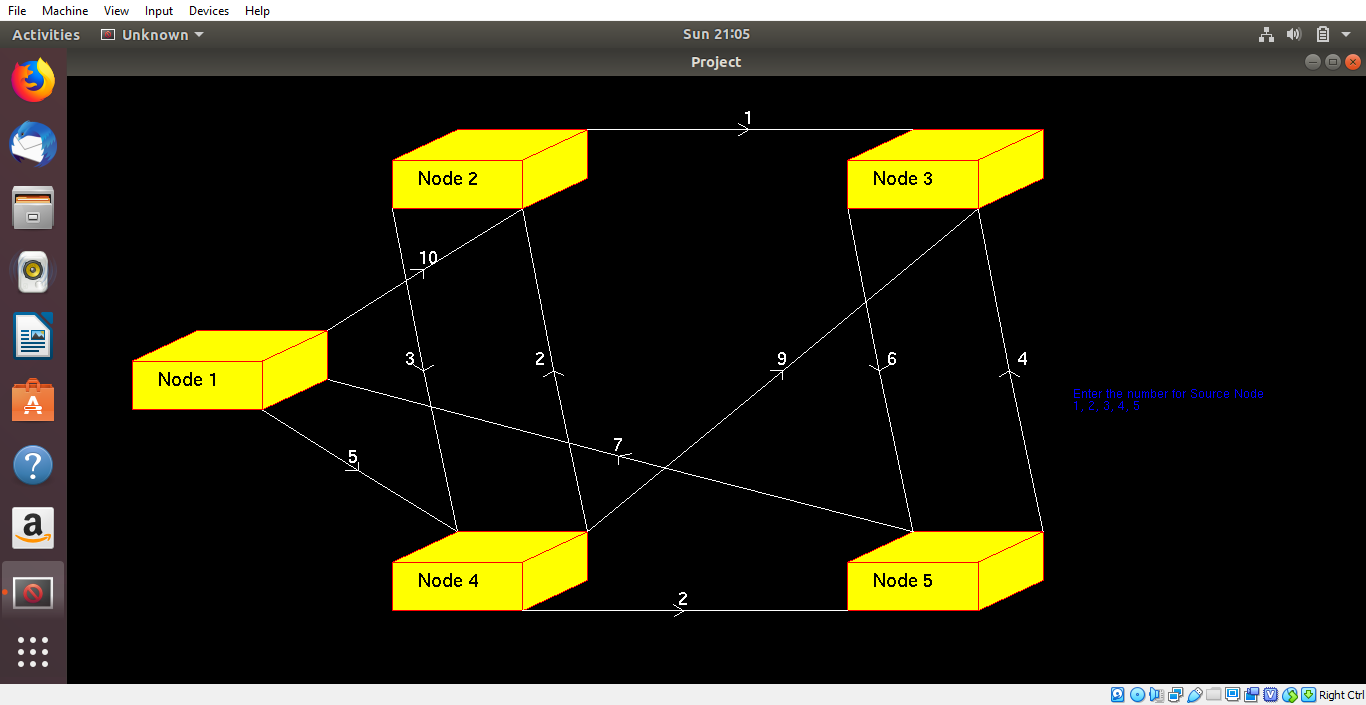
****

**Figure 4.1**: Initial Window.

All 5 nodes with the respective cost to each path.

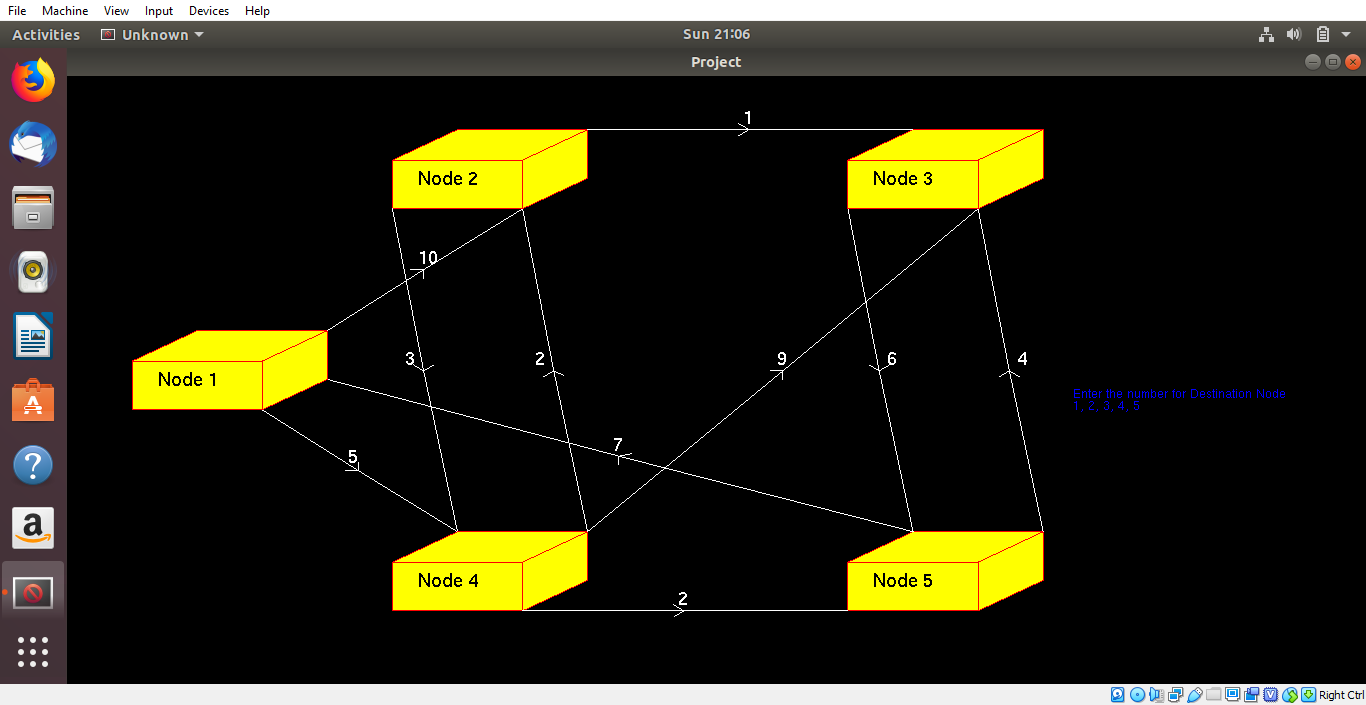
**Figure 4.2**: All the 5 nodes

When we press ‘E’ or ‘e’ the window to enter the source nodes opens

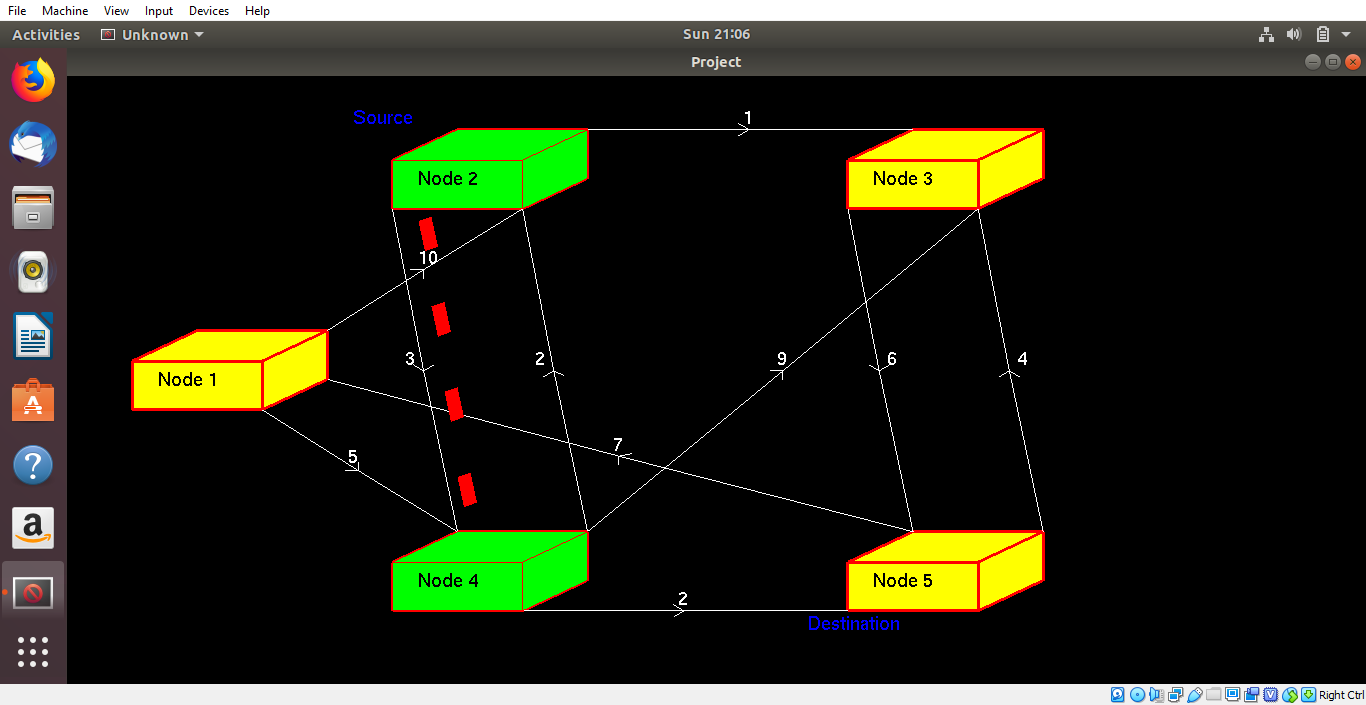


**Figure 4.3**: Window to input the source input node

When we press the input for the source that is keys ‘1’or ’2’ or ‘3’ or ‘4’ or ‘5’.The destination input window opens.

****

**Figure 4.4**:Window to input the destination input node



**Figure 4.5**:The final output for shortest path

**CHAPTER 5**

**Conclusion and Future Scope**

**5.1 Conclusion**

This project is all about the illustrations of OpenGL and GLUT. We have used suitable OpenGL functions to develop objects. Aim of this project is to show the shortest path from source to destination using Dijkstra algorithm. It renders a picture of 5 nodes which can be either source or destination as per the choice of the user. The keyboard is used for interaction to select the source and the destination of the user choice. All the functions of the project are the outcome of the openGL built-in function calls.

**5.2 Future Enhancement**

The future scope of our project is vast and can be used in extensive ways:

* The number of nodes taken could be dynamic as per number given by the user.
* There could be addition of mountains and terrains.

**References**

# [1] Computer Graphics: Principles and Practice by James D. Foley

# [2] Edward Angel, “Interactive Computer Graphics”, Pearson education,Fifth Edition.

[3] <http://www.glprogramming.com/blue/ch01.html>

[4] <https://en.wikipedia.org/wiki/Systems_design>

[5] https://en.wikipedia.org/wiki/Computer\_graphics\_(computer\_science)

**Appendix A**

**OpenGL Routines:**

* **void glBegin (glEnum mode) :**

Initiates a new primitive of type mode and starts the collection of vertices. Values of mode include GL\_POINTS, GL\_LINES, and POLYGON

* **void glEnd () :**

Terminate a list of vertices.

* **void glutInitWindowPositin(intx,int y) :**

Specify the initial position of the top-left corner of the window in pixel.

* **void glutMainLoop() :**

Cause the program to enter an event-processing loop. It should be the last statement in Main

* **void glutDisplayFunc(void (\*func)(void)) :**

Registers the display functions func that is executed when the window needs to be redrawn.

* **void glutSwapBuffers() :**

Swaps the front and back buffers.

* **void glutAddmenuEntry (char \*name, int value) :**

Add the entry with the string name displayed to the current menu. Values are returned to the menu callback when the entry is selected.

* **void glMatrixMode (GLenum mode) :**

Specifies matrix will be affected by subsequent transformations. Mode can be GL\_MODELVIEW,GL\_PROJECTION

* **void LoadIdentity() :**

Set the current transformation matrix to an identity matrix.

**Appendix B**

**Software Requirements**

Operating system : WINDOWS XP or above

Editor : gedit text editor

Programming language : C with OpenGL

**Hardware Requirements**

Processor : INTEL PENTIUM 4 or AMD

Primary memory : 512 MB RAM or more

Frequency : 1 MHz or more