**1.INTRODUCTION**

* Introduction to OpenGL:

OpenGL is a low-level graphics library specification. It makes available to the programmer a small set of geometric primitives - points, lines, polygons, images, and bitmaps. OpenGL provides a set of commands that allow the specification of geometric objects in two or three dimensions, using the provided primitives, together with commands that control how these objects are rendered (drawn).

Since OpenGL drawing commands are limited to those that generate simple geometric primitives (points, lines, and polygons), the OpenGL Utility Toolkit (GLUT) has been created to aid in the development of more complicated three-dimensional objects such as a sphere, a torus, and even a teapot. GLUT may not be satisfactory for full-featured OpenGL applications, but it is a useful starting point for learning OpenGL.

GLUT is designed to fill the need for a window system independent programming interface for OpenGL programs. GLUT simplifies the implementation of programs using OpenGL rendering. The GLUT application programming interface (API) requires very few routines to display a graphics scene rendered using OpenGL. The GLUT routines also take relatively few parameters.

* Introduction to Encapsulation And Decapsulation:

This project is to demonstrate how data moves across different layers in the network with the concept of encapsulation and decapsulation

* Encapsulation:

When data moves from upper layer to lower level of [TCP/IP protocol stack](http://www.omnisecu.com/tcpip/tcpip-model.php) (outgoing transmission) each layer includes a bundle of relevant information called a header along with the actual data. The data package containing the header and the data from the upper layer then becomes the data that is repackaged at the next lower level with lower layer's header. Header is the supplemental data placed at the beginning of a block of data when it is transmitted. This supplemental data is used at the receiving side to extract the data from the encapsulated data packet. This packing of data at each layer is known as data encapsulation.

* Decapsulation:

The reverse process of encapsulation (or decapsulation) occurs when data is received on the destination computer. As the data moves up from the lower layer to the upper layer of [TCP/IP protocol stack](http://www.omnisecu.com/tcpip/tcpip-model.php) (incoming transmission), each layer unpacks the corresponding header and uses the information contained in the header to deliver the packet to the exact network application waiting for the data.

**2.System Requirements**

* Software Requirements:

1. Microsoft Visual Studio(2010)
2. glut32.lib
3. glut.h
4. glut32.dll

* Hardware Requirements:

1. Operating System-Microsoft Windows 10
2. RAM: 4GB
3. Hard Disk: 3GB

**3.Description**

* Project Title:

Computer Graphics And Visualization Project for Encapsulation And Decapsulation using OpenGL.

* Project Description:
* The main idea of this project is to encapsulate and decapsulate packets through different layers of osi model.
* This project also helps in understanding the header formats of each protocol layers.
* Encapsulation is a process used to hide or protect a process from the possibility of outside interference or misuse of the system while simplifying the use of the system itself ,also making one type of network data packets to other data types.
* Encapsulation occurs when the protocal that is on the lower layer receives data from the protocol that is at the higher layer and put the data into a data format that is understood by the protocol. Access to the internal system so arraged through a set of interfaces.With the encapsulation of data into an identity.
* A simple example encapsulation process in the process of mail delivery ,if a letter would be sent but without the envelop , addresss and postage.
* The letter should have an identity in order to get to the destination ,if it does not have an identity the letter will not be able to reach the destination .Envelops with address and stamp act the same as the data encapsulation.
* Data decapsulation is simply the reverse of encapsulation. Decapsulation is used when an incoming transmission (to be received by the destination computer) is unpacked as it moves up the protocol stack.
* The data at the bottom of the layer are packaged several times to ensure security. As they are sent along the transport layer, these data are unpacked until they reach the network application awaiting the data.

**4.OpenGL API USED**

* OpenGL API And Functions:

1.GL\_QUADS -- draws a series of separate four-sided polygons

The following draws a rectangle:

glRectf(0f, 0f, 1f, 1f); // x0, y0, x1, y1: two opposite corners  
 // of the rectangle.

it is the same as:

glBegin(GL\_QUADS);  
 glVertex2f(0.0f, 0.0f); //note 2D form  
 glVertex2f(1.0f, 0.0f);  
 glVertex2f(1.0f, 1.0f);  
 glVertex2f(0.0f, 1.0f);  
 glEnd();

2. GL\_POLYGON -- draws a polygon from vertex **v0** to **vn-1**

The following code constructs a filled in parallelogram on the x-y plane:

glBegin(GL\_POLYGON);  
 glVertex2f(0.0f, 2.0f); //note 2D form  
 glVertex2f(1.0f, 2.0f);  
 glVertex2f(0.0f, -2.0f);  
 glVertex2f(-1.0f, 0.0f);  
 glEnd();

#### 

#### 3.Specifying Color

glColor\*( )  
 For example, glColor3f() function takes three floating-point  
 values for the red, green and blue color to select. A value of 0  
 means zero intensity; a value of 1.0 is full intensity, and any  
 value in between is a partial intensity.

#### 4. Clearing the rendering window

glClearColor(0.0f, 0.0f, 0.0f);  
 glClearDepth(1.0f);  
  
 // once the clear color and clear depth values have been  
 // set, both buffers can be cleared, the following command  
 // is usually issued just before you begin to render a  
 // scene, usually as the first rendering step to a  
 // WM\_PAINT message  
 // clear both buffers in the mean time  
 glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

5.The **glBegin** and [**glend**](https://docs.microsoft.com/en-us/windows/desktop/opengl/glend) functions delimit the vertices of a primitive or a group of like

primitives.

6.The **glBitmap** function draws a bitmap.

void WINAPI glBitmap(

GLSizei width,

GLSizei height,

GLfloat xorig,

GLfloat yorig,

GLfloat xmove,

GLfloat ymove,

const GLubyte \*bitmap

);

7.The **glFlush** function forces execution of OpenGL functions in finite time.

void WINAPI glFlush(void);

8.The [**glTranslatef**](https://docs.microsoft.com/en-us/windows/desktop/opengl/gltranslate) function multiplies the current matrix by a translation matrix.

void WINAPI glTranslatef(

GLfloat x,

GLfloat y,

GLfloat z

);

9.The **glViewport** function sets the viewport.

void WINAPI glViewport(

GLsizei width,

GLsizei height);

**5.Implementation Details**

* Implementation of project begins from the start page ,which mentions the tiltle of the project and gives brief description of the project .
* Keyboard Navigations have been used to implement user interface. On pressing Enter key we are directed to the main page of the project.
* Components in the main page,
* Two computers -indicating the sender and the receiver.
* TCP/IPModel – consisting of 5 layers

Application layer

Transport layer

Network layer

Data link layer

Physical layer

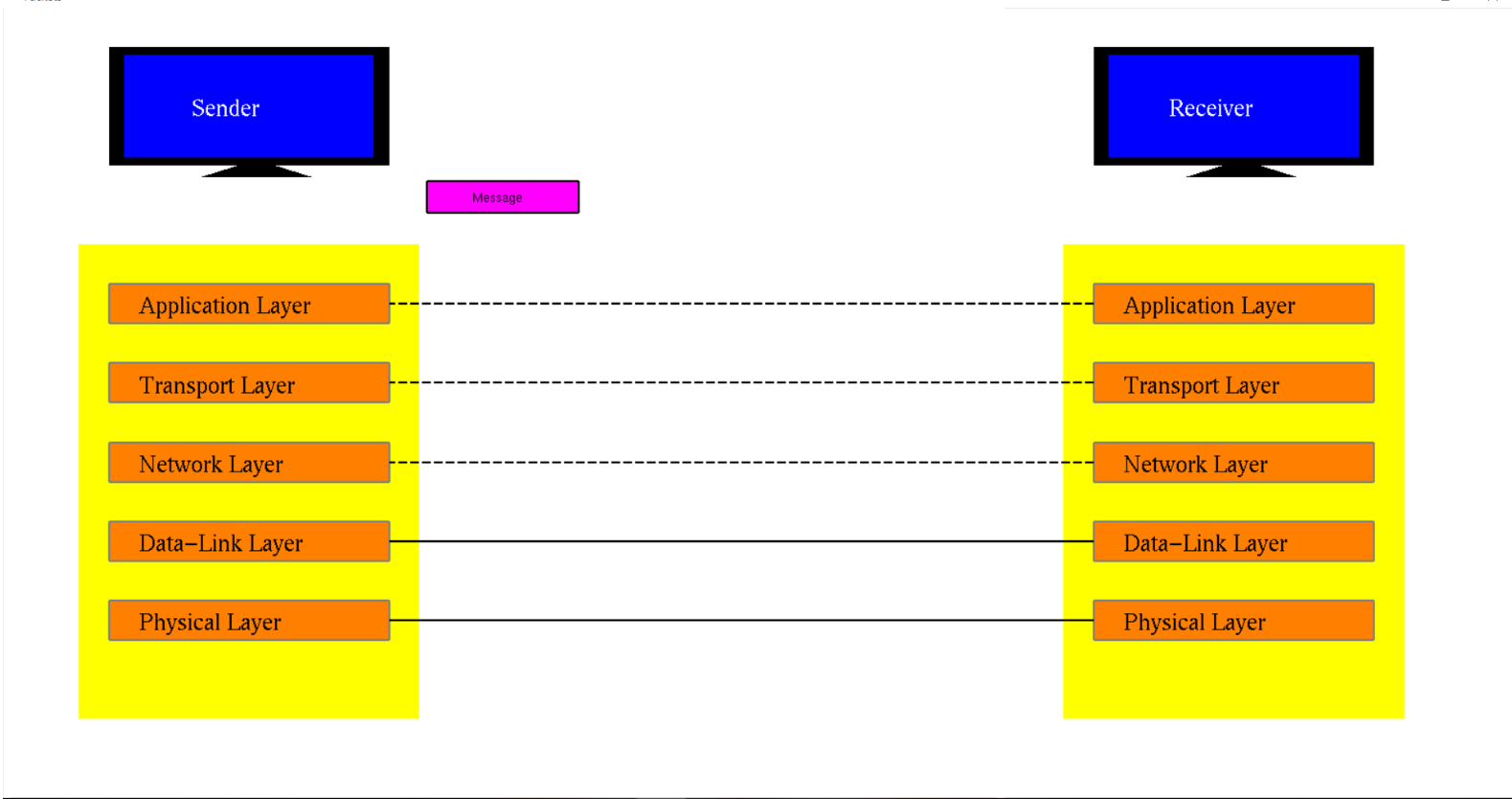
* Message Packet
* Again in the main page keyboard interface is implemented . A short message is displayed to know the keys to pressed to start the functioning of messge through the different layers.
* On pressing key ‘c’ again ,the message packet is translated or moved to application layer and it displays the data format of application .The application layer provides network services to all the network based applications by providing them with an interface that they can utilize for all their network based communication.
* The Application layer message is again encapsulated at the [Transport Layer](http://www.omnisecu.com/tcpip/transport-layer.php). If the protocol used at the [Transport Layer](http://www.omnisecu.com/tcpip/transport-layer.php) is [TCP (Transmission Control Protocol)](http://www.omnisecu.com/tcpip/transmission-control-protocol-tcp.php), the data packet is known as a "[TCP SEGMENT](http://www.omnisecu.com/tcpip/tcp-header.php)". If the protocol used at the Transport layer is [UDP (User Datagram Protocol)](http://www.omnisecu.com/tcpip/udp-user-datagram-protocol.php), the data packet is known as a "[UDP DATAGRAM](http://www.omnisecu.com/tcpip/udp-user-datagram-protocol.php)".
* On pressing key ‘c’ again ,the message packet is translated or moved to transport layer with the header part encapsulated with the data ,then it displays the header format of transport layer.
* The [Transport Layer](http://www.omnisecu.com/tcpip/transport-layer.php) contains two important protocols: [TCP (Transmission Control Protocol)](http://www.omnisecu.com/tcpip/transmission-control-protocol-tcp.php) and [UDP (User Datagram Protocol)](http://www.omnisecu.com/tcpip/udp-user-datagram-protocol.php). [TCP (Transmission Control Protocol)](http://www.omnisecu.com/tcpip/transmission-control-protocol-tcp.php) is more reliable but consumes more resource. [UDP (User Datagram Protocol)](http://www.omnisecu.com/tcpip/udp-user-datagram-protocol.php) is less reliable but consume fewer resources than [TCP (Transmission Control Protocol)](http://www.omnisecu.com/tcpip/transmission-control-protocol-tcp.php) and is faster than [TCP (Transmission Control Protocol)](http://www.omnisecu.com/tcpip/transmission-control-protocol-tcp.php).
* On pressing key ‘c’ again ,the message packet is translated or moved to Network layer with the new header part encapsulated with the data ,then it displays the header format of Network layer.The data packet created at the Internet layer by Internet Protocol, which again encapsulates the Transport layer segment/datagram, is known as a "[IP DATAGRAM](http://www.omnisecu.com/tcpip/internet-layer.php)".
* The data packet at the [Network Access layer](http://www.omnisecu.com/tcpip/network-access-layer.php), which encapsulates and may subdivide the [IP Datagram](http://www.omnisecu.com/tcpip/internet-layer.php), is known as a "[FRAME](http://www.omnisecu.com/tcpip/network-access-layer.php)" (generally Ethernet Frame).
* The main protocols included at [Internet layer](http://www.omnisecu.com/tcpip/internet-layer.php) are [IP (Internet Protocol)](http://www.omnisecu.com/tcpip/internet-layer.php), [ICMP (Internet Control Message Protocol)](http://www.omnisecu.com/tcpip/internet-control-message-protocol-icmp.php), [ARP (Address Resolution Protocol)](http://www.omnisecu.com/tcpip/address-resolution-protocol-arp.php), RARP (Reverse Address Resolution Protocol) and IGMP (Internet Group Management Protocol).
* On pressing key ‘c’ again ,the message packet is translated or moved to Datalink layer with the new header part encapsulated with the data ,then it displays the header format of data link layer.
* The [Frame](http://www.omnisecu.com/tcpip/network-access-layer.php) is converted into a bitstream at the lowest sublayer of the Network Access layer and then placed on medium.
* Now the same pattern of encapsulation takes place at the intermidiate routers.
* Decapsulation takes place in similar manner at each of the 5 layers ,unpacking if data from the header is carried out.
* Finally the message reaches the destination safely.
* Encapsulation is done at the source and [decapsulation](https://www.sciencedirect.com/topics/engineering/decapsulation) at the destination, whereas at the intermediate points, such as routers, both decapsulation and encapsulation are carried out.

**6.Snapshots(Result)**

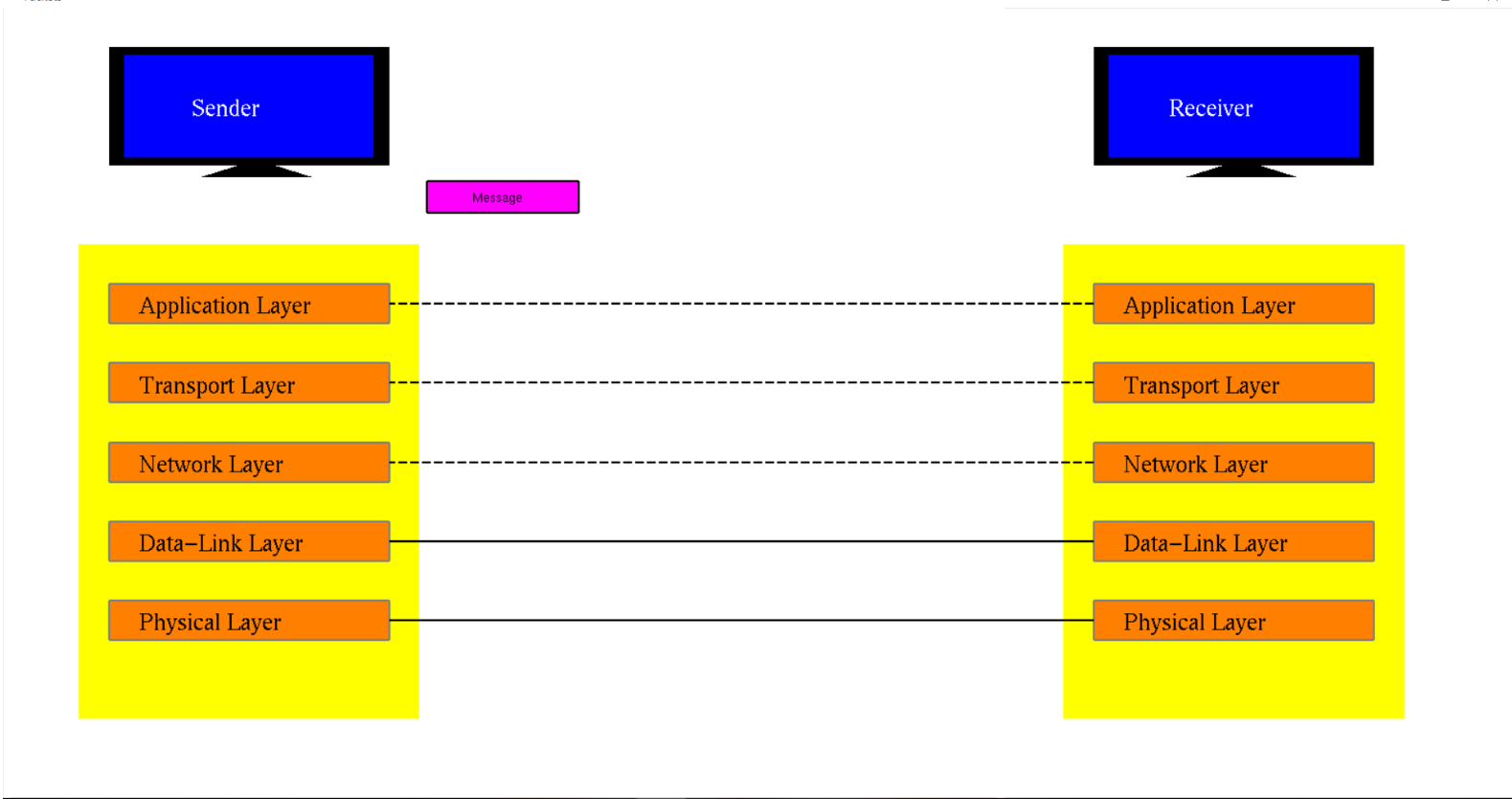
1.Start Page



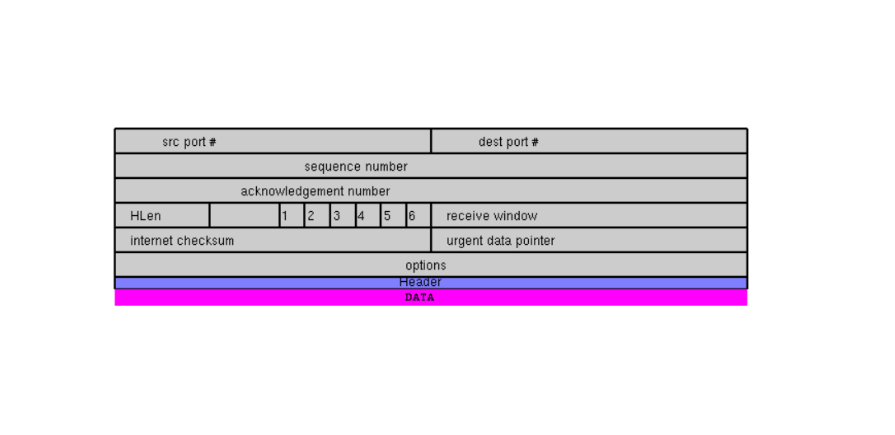
2.Main page



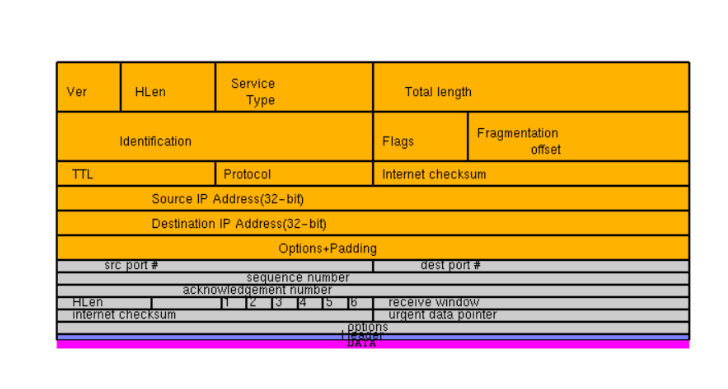
3.Data Encapsulation



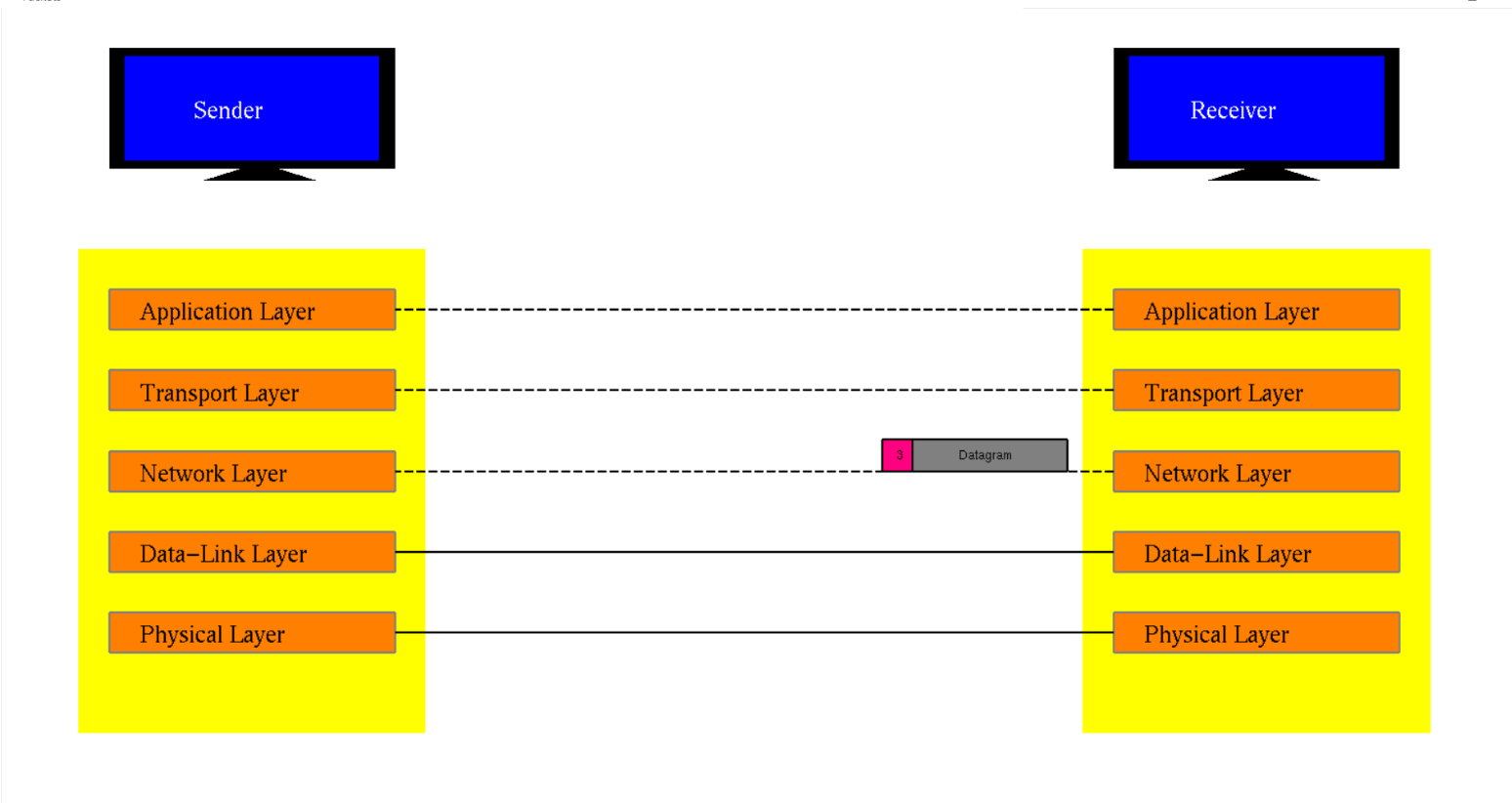
4.Header Format of Transport layer



5.Header Format at Network layer



6.Decapsulation of Data



**7.Conclusion**

Data Encapsulation And Decapsulation helps us to understand the basic concept of encapsulation and decapsulation in TCP/IP model.Through transformation functions using opengl we can understand how the meaasge packet moves across different layers.We can see how the header is added at each layer with the data.Through the images and description of header formats we get to understand the different protocols, implementation ,source address ,destination address port numbers etc.

Since the data is encapsulated at sender end and decapsulated at the receiver end , the message reaches from souce to destination safely without getting altered on the way of network.Thus,encapsulation and decapsulation plays important role network security.

**8.References**

1. Donald Hearn & Pauline Baker:Computer Graphics with OpenGL Version,3rd/4th Edition

Pearsom Education,2011

2. Edward Angel: Interactive Computer Graphics-A Top Down approach with OpenGL,

5th edition ,Pearson Education,2008

3. www.geekforgeeks.org

4. www.google.co.in

5. htttp://stackoverfllow.com/

6. www.scoopit.com

#### 