|  |  |  |
| --- | --- | --- |
| **TABLE OF CONTENTS** | | |
| **Chapter No** | **Chapter Name** | **Page No** |
|  | Acknowledgement |  |
|  | Table of Contents | 1 |
| 1 | Introduction | 2 |
|  | 1.1 About the Project | 2 |
|  | 1.2 Problem Statement | 3 |
|  | 1.3Objectives of the Project | 3 |
| 2 | System Requirements and Specifications  2.1 Software Requirements  2.2Hardware Requirements | 4  4  4 |
|  |  |  |
| 3 | Implementation  3.1 Function/Method description  3.2Results (Screen shots of the output) | 5  6-7  8-9 |
| 4 | Conclusion | 10 |

**CHAPTER 1: INTRODUCTION**

Computer Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D Or 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly. Computers have become a powerful medium for the rapid and economical production of pictures. Graphics provide a so natural means of communicating with the computer that they have become widespread. Interactive graphics is the most important means of producing pictures since the invention of photography and television. We can make pictures of not only the real-world objects but also of abstract objects such as mathematical surfaces on 4D and of data that have no inherent geometry. A computer graphics system is a computer system with all the components of the general-purpose computer system. There are five major elements in system: input devices, processor, memory, frame buffer, output devices.

**1.1 PROBLEM STATEMENT**

To design and develop a software system to simulate an airplane crash using OpenGL.

**1.2 OBJECTIVES**

* To create a simulation of an airplane crash.
* To depict the sequence of events leading up to and during the crash.
* To start stimulation, press the key ‘s’ and to exit from screen press ‘q’.
* To move the plane, press the key ‘m’.

**CHAPTER -2: SYSTEM REQURIMENTS AND SPECIFICATIONS**

**2.1 SOFTWARE REQUIREMENTS**

• Windows os/ubuntu os.

• Eclipse compiler.

**2.2 HARDWARE REQUIREMENTS**

•Processor- Intel or AMD(Advanced Micro Devices)

• RAM- 512MB(minimum)

• Hard Disk-1MB(minimum)

• Mouse

• Keyboard

• Monitor

**CHAPTER -3: IMPLEMENTATION**

**3.1 FUNCTION/METHOD DESCRIPTION**

GL\_LINES –

Treats each pair of vertices as an independent line segment.

Vertices 2n - 1 and 2n define line n. N/2 lines are drawn.

GL\_LINE\_LOOP –

Draws a connected group of line segments from the first vertex to the last, then back to the first. Vertices n and n + 1 define line n. The last line, however, is defined by vertices N and N lines are drawn.

Basic Functions:

glPushMatrix, glPopMatrix Function The glPushMatrix and glPopMatrix functions push and pop the current matrix stack.

SYNTAX: void glPushMatrix();

void glPopMatrix(void);

glBegin, glEnd Function :

The glBegin and glEnd functions delimit the vertices of a primitive or a group of like primitives. SYNTAX: void glBegin, glEnd(GLenum mode);

PARAMETERS:

mode –

The primitive or primitives that will be created from vertices presented between glBegin and the subsequent glEnd. The following are accepted symbolic constants and their meanings:

**Transformation Functions**

**glTranslate Functions:**

The glTranslated and glTranslatef functions multiply the current matrix by a translation matrix. SYNTAX: void glTranslate( x, y, z);

PARAMETERS: x, y, z - The x, y, and z coordinates of a translation vector.

**Funtions used to display**

**glMatrixMode Function**

The glMatrixMode function specifies which matrix is the current matrix.

SYNTAX: void glMatrixMode(GLenum mode);

PARAMETERS:

mode - The matrix stack that is the target for subsequent matrix operations. The modeϖ parameter can assume one of three values:

Value Meaning

GL\_MODELVIEW Applies subsequent matrix operations to the model view matrix stack.

**glLoadIdentity Function**

The glLoadIdentity function replaces the current matrix with the identity matrix.

SYNTAX: void glLoadIdentity(void);

**FUNCTIONS USED TO SET THE VIEWING VOLUME**

**glOrtho**

This function defines orthographic viewing volume with all parameters measured from the centre of projection. multiply the current matrix by a perspective matrix.

SYNTAX:

void glOrtho( GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far)

PARAMETERES:

left, right –

Specify the coordinates for the left and right vertical clipping planes.

bottom, top

Specify the coordinates for the bottom and top horizontal clipping planes.

nearVal, farVal –

Specify the distances to the nearer and farther depth clipping planes. These values are negative if the plane is to be behind the viewer.

**CALL BACK FUNCTIONS**

**glutDisplayFunc Function**

glutDisplayFunc sets the display callback for the current window.

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

**glutReshapeFunc Function**

glutReshapeFunc sets the reshape callback for the current window.

SYNTAX: void glutReshapeFunc(void (\*func)(int width, int height));

**MAIN FUNCTION**

**glutInit Function**

glutInit is used to initialize the GLUT library.

SYNTAX: glutInit(int \*argcp, char \*\*argv);

PARAMETERS:

argcp - A pointer to the program's unmodified argc variable from main. Upon return, theϖ value pointed to by argcp will be updated, because glutInit extracts any command line options intended for the GLUT library.

Argv -

The program's unmodified argv variable from main. Like argcp, the data for argv will AEROPLANE CRASH Dept.. of CSE,CEC 2010-2011 11 be updated because glutInit extracts any command line options understood by the GLUT library.

glutInit(&argc,argv);

**glutInitDisplayMode Function**

glutInitDisplayMode sets the initial display mode.

SYNTAX: void glutInitDisplayMode(unsigned int mode);

PARAMETERS:

mode –

Display mode, normally the bitwise OR-ing of GLUT display mode bit masks. See values below: GLUT\_RGB: An alias for GLUT\_RGBA.

GLUT\_DOUBLE:Bit mask to select a double buffered window.

This overrides GLUT\_SINGLE. GLUT\_DEPTH: Bit mask to select a window with a depth buffer.

**glutMainLoop Function**

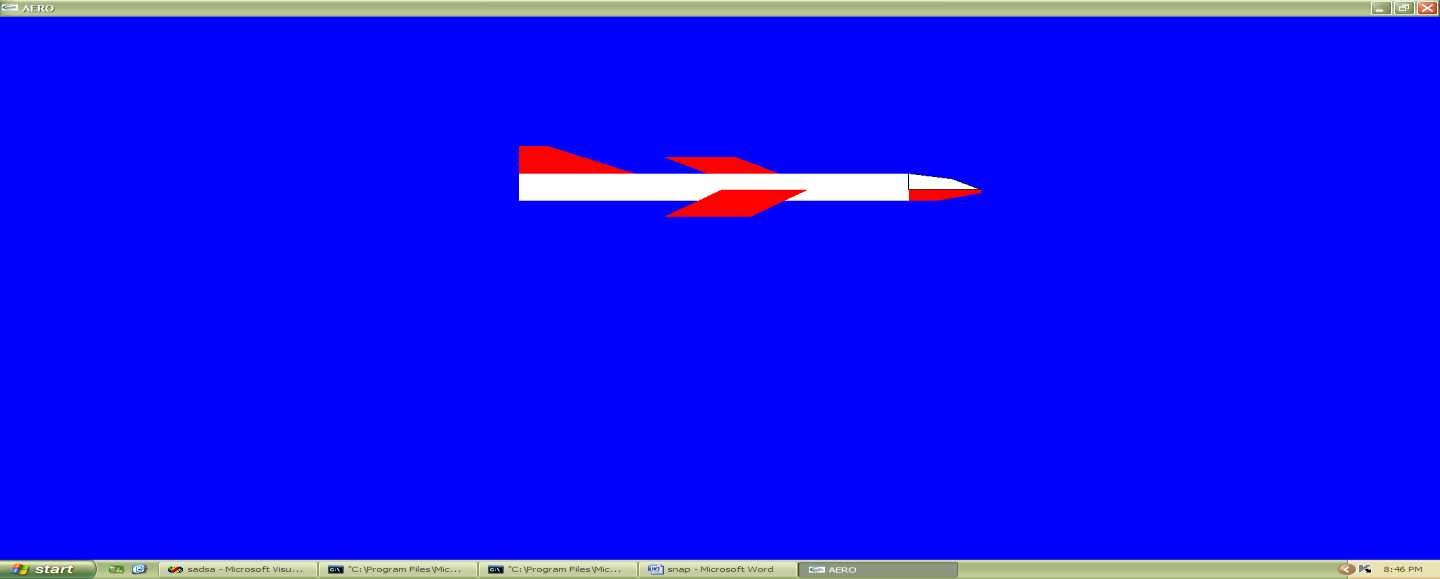
glutMainLoop enters the GLUT event processing loop.

SYNTAX: void glutMainLoop(void);

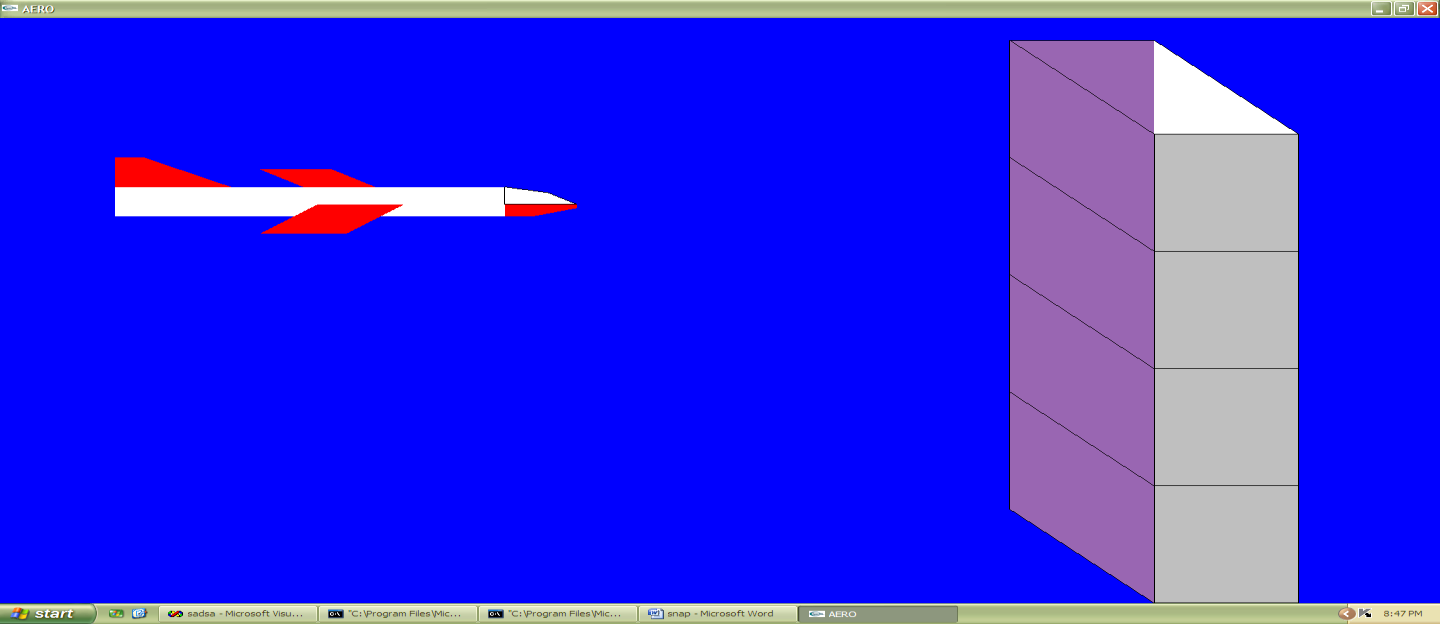
**3.2 RESULTS (SCREEN SHOTS OF THE OUTPUT)**



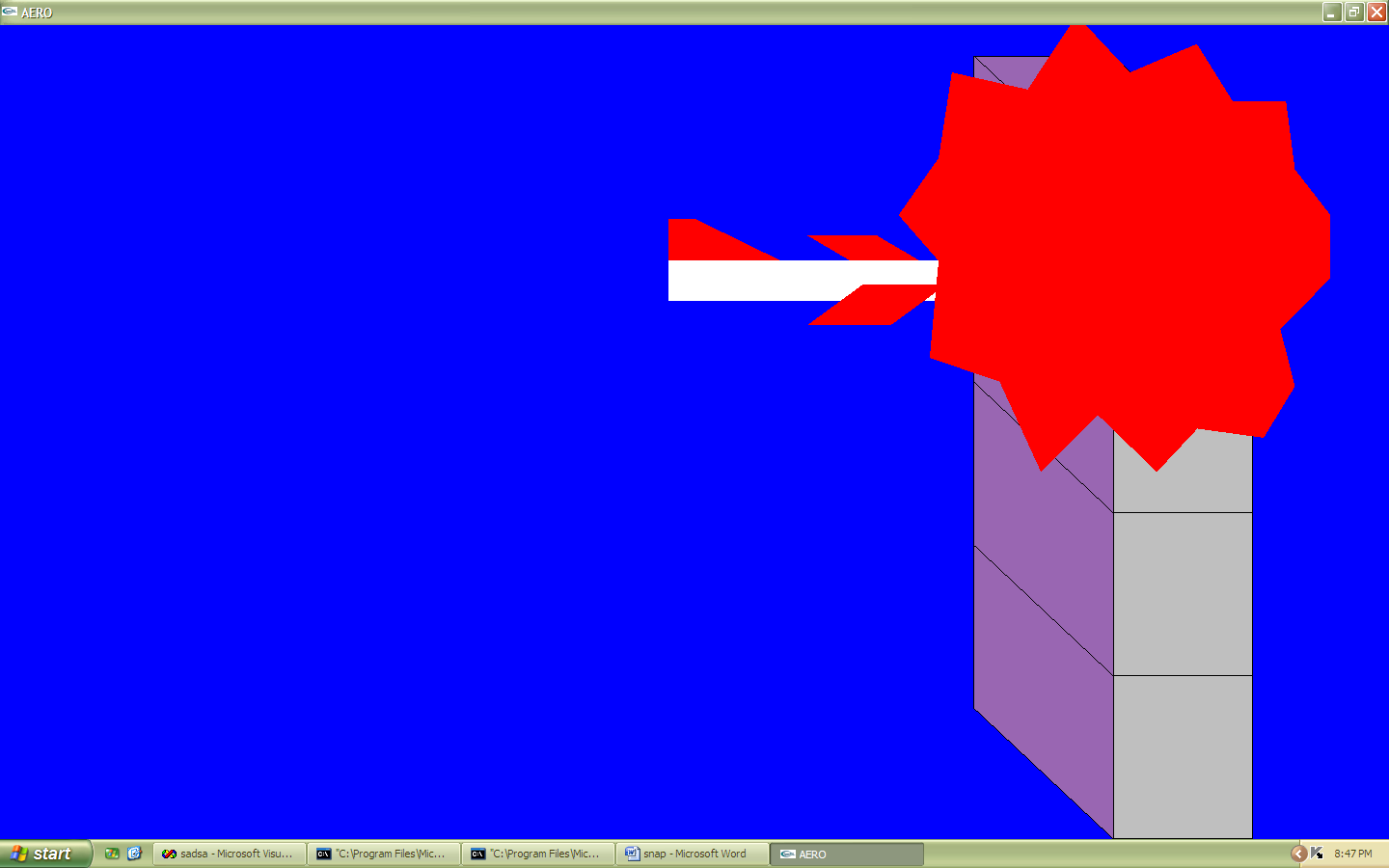
**Fig 3.2.1:aeroplane take-off**



**Fig 3.2.2: aeroplane flying**



**Fig 3.2.3:aeroplne flying**



**Fig 3.2.4:aeroplane is crashing**

**CHAPTER -4 CONCLUSION**

In conclusion, the computer graphics project successfully utilized OpenGL to simulate an airplane crash, resulting in a realistic and visually engaging depiction of the events. By leveraging OpenGL's advanced graphics rendering capabilities, the project accurately rendered the flight path, impact, and subsequent destruction in a visually compelling manner. The implementation of lighting and shading techniques in OpenGL enhanced the realism of the simulation, making the crash scene appear more authentic.

The utilization of OpenGL's particle systems allowed for the generation of realistic debris and smoke effects, further adding to the immersive experience. The interactive nature of OpenGL enabled users to navigate and explore the crash scene from different perspectives, enhancing their understanding of the event.

Through this project, the potential of OpenGL in simulating complex real-world scenarios was demonstrated, emphasizing its importance in computer graphics applications. The project also highlighted the significance of using realistic physics-based models and accurate object dynamics to create visually convincing crash simulations.

future iterations, additional improvements could be made by incorporating advanced OpenGL features, such as advanced shaders and real-time physics simulations, to further enhance the realism and interactivity of the airplane crash simulation. Additionally, the project could explore incorporating user-controlled parameters, allowing users to customize aspects of the crash scenario.

Overall, the project successfully showcased the capabilities of OpenGL in creating immersive and realistic visualizations of airplane crashes, contributing to the fields of aviation safety, education, and computer graphics research.