Steam Engine

CS352: Computer Graphics & Visualization Lab

Project Report

Course Instructor:
Dr. Somnath Dey

Submitted By:

Koneti Anuhya – 200001037
Kothuru Sharvani – 200001038
Nelavalli Sri Nikhitha – 200001052

Introduction

The goal of this project is to create a steam engine which comprises of Piston,Engine pole,Cylinder head,Flywheel,crankbell,and a crank.The engine's crank can be rotated either clockwise or anticlockwise by the viewer and also crank speed can be decreased or increased by the viewer. Using the myCylinder() function, a Piston, Engine Pole, Cylinder Head, Flywheel, Crank Bell, and Crank are first created. gluCylinder() and gluDisk() are the primary primitives utilised in the myCylinder() function to generate a Cylinder. So every time, myCylinder( ) function is called inside the functions used to create Piston, Engine Pole, Cylinder Head, Flywheel, Crank Bell and Crank. The steam engine uses the force produced by steam pressure to push a piston back and forth inside a cylinder. This pushing force can be transformed, by a connecting rod and crank, into rotational force for work.

We can make steam engine transparent and display. In display function, at first it clears the drawing buffer and if transparency is set, displays the model twice, first time accepting those fragments with a ALPHA value of 1 only, then with DEPTH_BUFFER writing disabled for those with other values. Initially when the animation is not called, the crank angle will not change and the window is idle. When called increments the crank angle by ANGLE_STEP, updates the head angle and notifies the system that the screen needs to be updated. When a menu option has been selected, it translates the menu item identifier into a keystroke, then calls the keyboard function. A menu will be associated with the mouse too. The viewer can also see the shaded steam engine.

The technique of creating a photorealistic or non-photorealistic image from a 2D or 3D model using a computer programme is known as rendering or image . Another type of steam engine graphics is the technical illustration, which shows the engine in a more realistic and detailed way. These illustrations can be used to help engineers and designers understand the various components and how they fit together, as well as to show how the engine operates.In addition to these types of graphics, there are also many historical illustrations and photographs of steam engines that provide a glimpse into the past and the evolution of steam power.

Specifications:

A significant portion of our code is done by using three libraries' functionalities to directly access OpenGL. The primary GL library contains functions with names that start with the letters gl that are kept in a library that is often called GL . Apart from this the OpenGL Utility Library (GLU) contains code for making common objects and optimising viewing but only requires GL functions.This library uses only GL functions but contains code for creating common objects and simplifying viewing. All functions in GLU can be created from the core GL library but application programmers prefer not to write the code repeatedly. The GLU library is available in all OpenGL implementations; functions in the GLU library begin with letters glu.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.To interface with the window system and to get input from external devices into our programs, we need at least one more system-specific library that provides the "glue" between the window systemand OpenGL. For the X window system, this library is functionality that should be expected in anymodern windowing system.OpenGL commands use the prefix gl and initial capital letters for each word making up the command name. Similarly, OpenGL defined constants begin with GL_, use all capital letters and useunderscores to separate words (like GL_COLOR_BUFFER_BIT).

void glScalef(TYPE sx, TYPE sy, TYPE sz):

This alters the current matrix by a scaling of (sx, sy, sz). TYPE here is GLfloat.

void glRotatef(TYPE angle, TYPE dx, TYPE dy, TYPE dz):

This rotates the current matrix by a certain amount around the axes (dx, dy, dz). Here, TYPE is GLfloat.

void glTranslatef(TYPE x, TYPE y, TYPE z):

This changes the current matrix by (x, y, z) displacement. Here, TYPE 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 glutSwapBuffers()

Swaps the front and back buffers. User defined functions are used to color the curves in a standard cycle rainbow manner which becomes very easy for the user to identify the levels of recursion for the curves.

void glutInit(int *argc, charargv)**

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 glViewport(int x,int y,GLsizei width,GLsizei height)

Specifies a width * height viewport in pixels whose lower-left corner is at (x,y) measured from he origin of the window.

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

The controls are:-

1) 's' -> Shading
2) ' ' -> Animation
2) 'o' -> Transparency.
3) 'n' -> Light(1) on/off
4) 'x' -> Light(2) on/off
4) '+' -> To speed up
5) '-' -> To Speed down
6) '8' -> To scale down
7) '9' -> To scale up
6) 'a' -> Rotation in anti clockwise direction of both crank & piston
7) 'z' -> Rotation in clockwise direction of both crank & piston
8) '1' -> Rotating the viewport along y axis in anticlockwise direction
9) '2' -> Rotating the viewport along z axis in anticlockwise direction
10) '3' -> Rotating the viewport along x axis in anticlockwise direction
11) '4' -> Rotating the viewport along y axis in clockwise direction
12) '5' -> Rotating the viewport along z axis in clockwise direction
13) '6' -> Rotating the viewport along x axis in clockwise direction

Functionalities Implemented:

Void myBox(GLdouble x, GLdouble y, GLdouble z)

Draws a box by scaling a glut cube of size 1. Also checks the shaded toggle to see which rendering style to use. NB Texture doesn't works correctly due to the cube being scaled.

Void myCylinder(GLUquadricObj * object, GLdouble outerRadius, GLdouble innerRadius, GLdouble lenght)

Draws a cylinder using glu function, drawing flat disc's at each end, to give the appearence of it being solid.

Void draw_piston(void)

Draws a piston.

Void draw_engine_pole(void)

Draws the engine pole and the pivot pole for the cylinder head.

Void draw_cylinder_head(void)

Draws the cylinder head at the appropreate angle, doing the necesary translations for the rotation.

Void draw_flywheel(void)

Draws the flywheel.

Void draw_crankbell(void)

Draws the crank bell, and the pivot pin for the piston. Also calls the appropriate display list of a piston doing the nesacary rotations before hand.

Void draw_crank(void)

Draws the complete crank. Piston also gets drawn through the crank bell function.

Void animation(void)

Called when the window is idle. When called increments the crank angle by ANGLE_STEP, updates the head angle and notifies the system that the screen needs to be updated.

Void keyboard(unsigned char key, int x, int y)

Called when a key is pressed. Checks if it recognises the key and if so acts on it, updating the screen.

Void make_image(void)

Makes a simple check pattern image. (Copied from the redbook example "checker.c".).

REFERENCES:

- https://github.com/itmesneha/Steam-Engine-using-OpenGL/blob/master/main.cpp
- https://stackoverflow.com/questions/14998400/displaying-image-using-soil-with-opengl
- https://stackoverflow.com/questions/18886598/how-do-i-install-soil-simple-opengl-image-loader

Outputs

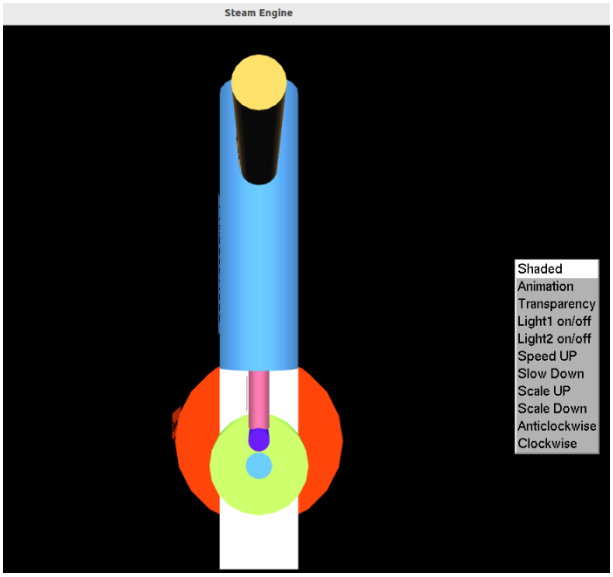

Fig 1: Shaded View of Steam Engine associated with menu

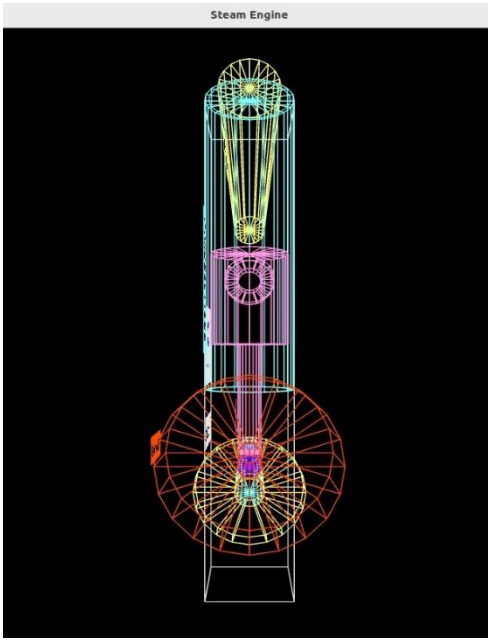

Fig2: Wire gauge form of steam engine

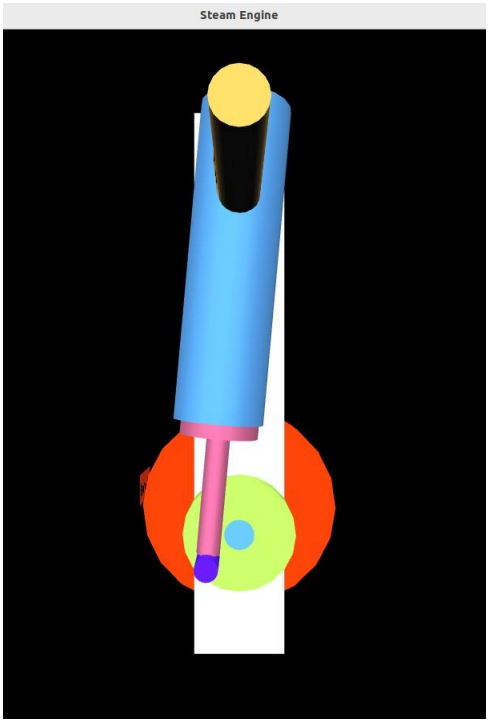

Fig3: Rotation in Anticlockwise direction

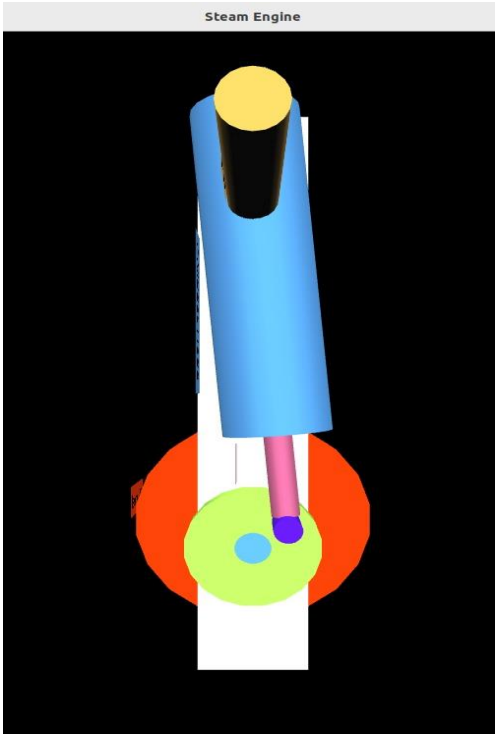

Fig4: Rotation in Clockwise direction

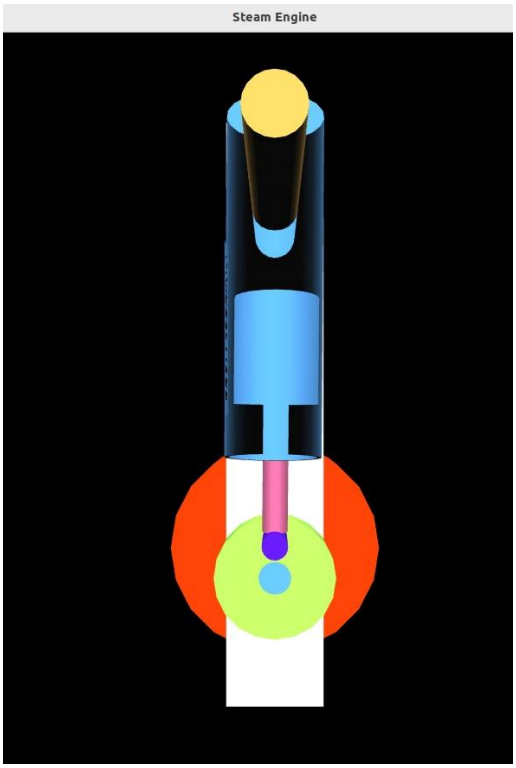

Fig5: Transparent View in shaded mode

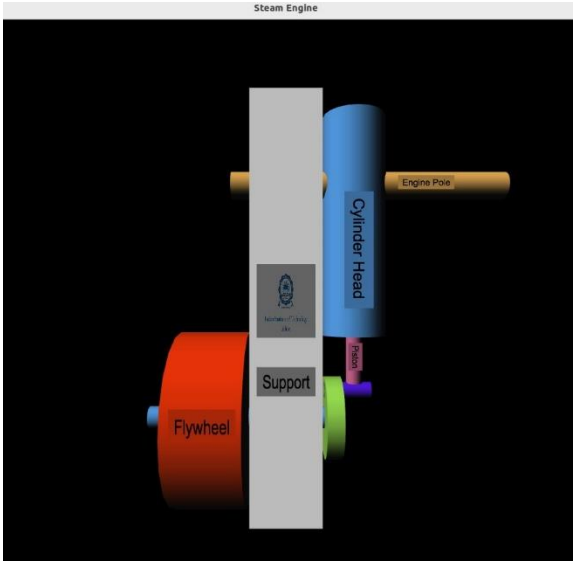

Fig6: Light1 off

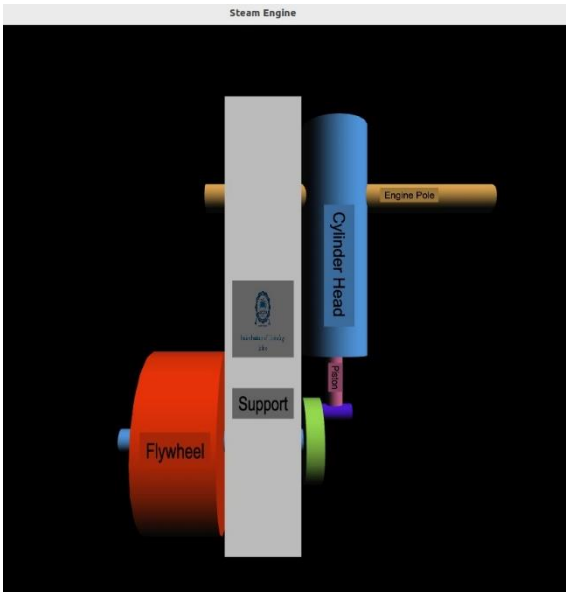

Fig7: Light2 off

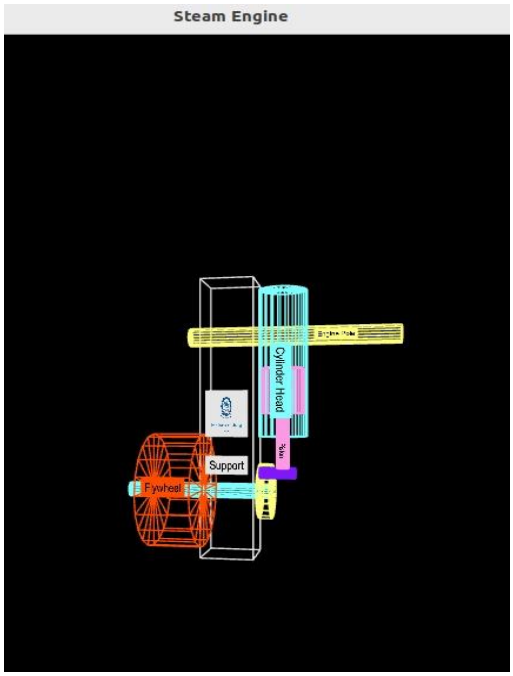

Fig8: Scale down

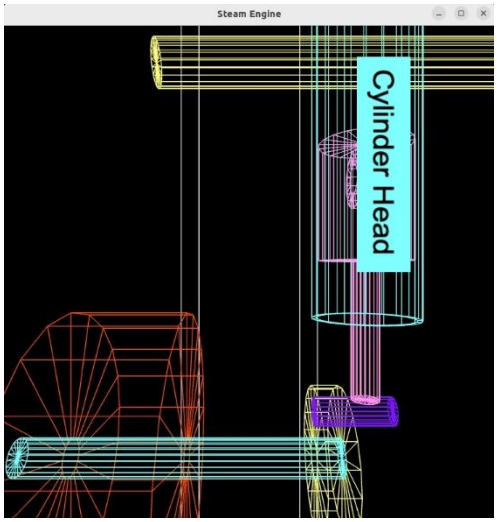

Fig9: Scale Up

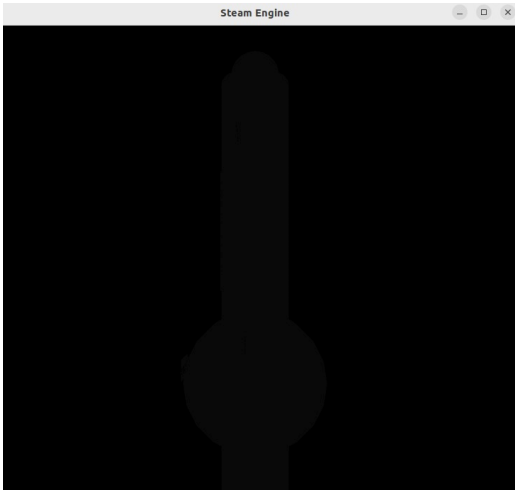

Fig10: Both the lights are off