VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JnanaSangama, Belgaum-590014

A COMPUTER GRAPHICS Mini Project Report

On

"PAC-MAN"

Submitted in Partial fulfilment of the Requirements for the VI semester of the Degree of

Bachelor of Engineering

In

Computer Science & Engineering

By

SURABHI G R(1CE18CS084)

CHETANA NATH(1CE18CS061)

Under the Guidance of

Mr. RAMESH B
Professor, Dept. Of CSE

CITY ENGINEERING COLLEGE

Doddakallasandra, Kanakapura Road,

Bengaluru-560061

CITY ENGINEERING COLLEGE

Doddakallasandra, Kanakapura Road, Bengaluru-560061

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certificate that the COMPUTER GRAPHICS Mini Project work entitled **"PAC-MAN"** has been carried out by **SURABHI G R(1CE18CS090)** and **CHETANA NATH(1CE18CS030),** bonafide students of City Engineering College in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year **2020-2021**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The COMPUTER GRAPHICS Mini Project Report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said Degree.

Mr. Ramesh b	**Mr. B Vivekavardhana Reddy**	**Dr. Thippeswamy H N**
Prof, Dept. Of CSE	Head, Dept. Of CSE	Principal

External Viva

Name of the examiners	Signature with date
1.	
2.	

ABSTRACT

A 2d graphics-based based game “PAC-MAN” is a great start for a student who starts learning computer graphics &visualization. The development of the game has large scope to learn computer graphics from scratch. We used OpenGL utility toolkit to implement the algorithm, written it in c++ language.

As we all know PAC MAN is the most popular game. We have implemented the same with the help of OpenGL. The controls of the game completely dependent on the arrow keys. When the user presses the UP-arrow key the character in the game moves upwards same applies to the rest of the keys. This allows the user to hover around the maze. We have implemented 3 lives to user in order to reach the finishing game.

Finally, we could say by developing the game we have learnt the basics of computer graphics and in future by developing it further we shall learn more. It will be our pleasure if we could develop in 3d graphics package.

ACKNOWLEDGEMENT

While presenting this COMPUTER GRAPHICS project on “**PAC-MAN”,** we feel that it is our dutyto acknowledge the help rendered to us by various persons.

Firstly, we thank God for showering his blessings on us. We are grateful to our institution City Engineering College for providing us a congenial atmosphere to carry out the project successfully.

We would like to express our heartfelt gratitude to**, Dr. Thippeswamy H N,** Principal, CEC, Bangalore, for extending his support.

We would also like to express our heartfelt gratitude to **Prof. Vivekavardhana Reddy,** HOD, Computer Science and Engineering whose guidance and support was truly invaluable.

We are very grateful to our guide, **Mr. Ramesh b** and our co guides, **Mrs. Nandini.S.B.** and **Mrs. Sangeetha rao S** Prof., Department of Computer Science, for their able guidance and valuable advice at every stage of our project which helped me in the successful completion of our project.

We would also have indebted to our parent and friends for their continued moral and materialsupport throughout the course of project and helping me in finalize the presentation.

Our hearty thanks to all those have contributed bits, bytes and words to accomplish this project.

SURABHI G R(1CE18CS084)

CHETANA NATH(1CE18CS061)

TABLE OF CONTENTS

SL NO.	CHAPTERS	PAGE NO.
1.	**INTRODUCTION**	
	1.1 Overview	1
	1.2 History	1
	1.3 Applications of Computer Graphics	**2**
2.	**Literature Survey**	
	2.1 OpenGL (Open Graphics Library)	**3**
	2.2 Problem Section Statement	**5**
	2.3 Existing Systems	**5**
	2.4 Proposed Systems	**6**
	2.5 Objectives of the Project	**6**
3.	**System Requirement**	
	3.1 User Requirement	**7**
	3.2 Hardware Requirement	**7**
	3.3 Software requirement	**7**
4.	**Design and Implementation**	
	4.1 Design	**8**
	4.2 Header Files	**9**
	4.3 Simple Geometry	**9**
	4.4 Interaction	**10**
	4.5 Transformation	**11**
	4.6 Viewing	**11**
5.	**Source Code**	**12**
6.	**Snapshots**	**29**
	Conclusion	**31**
	Future Enhancement	**31**
	Bibliography	**32**

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The term “Computer Graphics” includes almost everything on computers that is not text or sound. Today nearly all computers use some graphics and users expect to control their computer through icons and pictures rather than just by typing. Computer graphics is the field of visual computing, where one utilizes computers both to generate visual images synthetically and to integrate or alter visual and spatial information sampled from the real world. The term computer graphics has several meanings:

- The representation and manipulation of pictorial data by a computer.
- The various technologies used to create and manipulate such pictorial data.
- The sub-field of computer science which studies for digitally synthesizing and manipulating visual content.

This field can be divided into several areas: real-time 3D rendering (often used in video games), video capture and video creation rendering, special effects editing (often used for movies and television), image editing and modelling (often used for engineering and medical purposes).

1.2 HISTORY

The phrase “Computer Graphics” was coined in 1960 by William fetter, a graphic designer for Boeing. The field of computer graphics developed with the emergence of computer graphics hardware. Early projects like the whirlwind and SAGE Projects introduced the CRT as a viable display and interaction interface and introduced the light pen as an input device.

In 1959, the TX-2 computer was developed at MIT‟s Lincoln Laboratory. The TX-2 integrated a number of new man-machine interfaces. A light pen could be used to draw sketches on the computer using Ivan Sutherland‟s revolutionary Sketchpad software. Using a light pen, Sketchpad allowed one to draw simple shapes on computer screen, save them and even recall them later.

Many of the most important early breakthroughs in computer graphics research occurred at the University of Utah in the 1970s. A student by the name of Edwin Catmull saw computers as the natural progression of animation and they wanted to be part of the evolution. He created an animation of his hand opening and closing. The first major advance in 3D computer graphics was created at UU by these early pioneers, the hidden surface algorithm.

In the 1980s, artist and graphics designer began to see the personal computer, particularly the Commodore Amiga and Macintosh, as a serious design tool, one that could save time and draw more accurately than other methods. In the late 1980s, SGI computers were used to create some of the first fully computer-generated short films at Pixar.

3D graphics became more popular in the 1990s in gaming, multimedia and animation. In 1995, Toy Story, the first full-length computer-generated animation film, was released in cinemas worldwide.

1.3 APPLICATIONS OF COMPUTER GRAPHICS

- Computational biology
- Computational physics
- Computer-aided design
- Computer simulation
- Digital art
- Graphic design
- Info graphics
- Information visualization

CHAPTER 2
LITERATURE SURVEY

2.1 OpenGL (Open Graphics Library):

OpenGL has become a widely accepted standard for developing graphics application. OpenGL is easy to learn, and it possesses most of the characteristics of another popular graphics system. It is top-down approach. OpenGL is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. The interface consists of over 250 different function calls which can be used to draw complex three-dimensional scenes from simple primitives.

OpenGL was developed by Silicon Graphics Inc. (SGI) in 1992and is widely used in CAD, virtual reality, scientific visualization, information visualization, and flight simulation. It is also used in video games, where it competes with Direct3D on Microsoft Windows platforms.

The interface between the application program and the graphics system can be specified through that set of function that resides in graphics library. The specification is called the APPLICATION PROGRAM INTERFACE (API). The application program sees only the API and is thus shielded from the details both the hardware and software implementation of graphics library. The software driver is responsible for interpreting the output of an API and converting these data to a form that is understood by the particular hardware.

Most of our applications will be designed to access openGL directly through functions in three libraries. Function in the main GL library have name that begin with the letter gl and stored in the library. The second is the openGL utility Library (GLU). This library uses only GL function but contains codes for creating common object and viewing. Rather than using a different library for each system we used available library called openGL utility toolkit (GLUT). It used as #include<glut.h>

A graphics editor is a computer program that allows users to compose and edit pictures interactively on the computer screen and save them in one of many popular “bitmap” or “raster” a format such as TIFF, JPEG, PNG and GIF.

Graphics Editors can normally be classified as:

- 2D Graphics Editors
- 3D Graphics Editors

A 3D Graphics Editor is used to draw 3D primitives Rectangles, Circle, polygons, etc. and alter those with operations like cut, copy, paste. These may also contain features like layers and object precision etc.

3D Graphics Editor should include the following features:

- Facilities: Cursor Movement, Editing Picture Objects
- Good User Interface: GUI / Toolbars / Icons Based UI

.

Computer Graphics is concerned with all aspects of producing pictures or images using a computer. A particular graphics software system called OpenGL, which has become a widely accepted standard for developing graphics applications.

The applications of computer graphics in some of the major areas are as follows

1. Display of information.
2. Design.
3. Simulation and Animation.
4. User interfaces.

My project titled “HELICOPTER GAME” uses OpenGL software interface and develops 2D images. This project uses the techniques like Translation, motion, display list, transformation techniques, etc.

2.2 PROBLEM SECTION STATEMENT

Computer graphics is no longer a rarity. It is an integral part of all computer user interfaces, and is indispensable for visualizing 2D, 3D and higher dimensional objects. Creating 3D objects, rotations and any other manipulations are laborious process with graphics implementation using text editor. OpenGL provides more features for developing 3D objects with few lines by built in functions.

The geometric objects are the building blocks of any individual. Thereby developing, manipulating, applying any transformation, rotation, scaling on them is the major task of any image development.

Thereby we have put our tiny effort to develop 2D objects and perform different operations on them by using OpenGL utilities.

2.3 EXISTING SYSTEM

The existing system involves computer graphics. Computer graphics started with the display of data on hardcopy plotters and cathode ray tube screens soon after the introduction of computer themselves. It includes the creation, storage and manipulation of models and images of objects.

These models include physical, mathematical, engineering, architectural and so on Computer graphics today is largely interactive –the user controls the contents, structure and appearance of objects and their displayed images by using input devices, such as keyboard, mouse or touch-sensitive panel on the screen. Interactive computer graphics is the most important means of producing pictures since the invention of photography and television.

2.4 PROPOSED SYSTEM

In proposed system, the OpenGL is a graphic software system designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms. To achieve these qualities, no commands for performing windowing tasks or obtaining user input are included in OpenGL; instead, you must work through whatever windowing system controls the particular hardware you're using.

OpenGL doesn't provide high-level commands for describing models of three-dimensional objects. Such commands might allow you to specify relatively complicated shapes such as automobiles, parts of the body, airplanes, or molecules. With OpenGL, you must build up your desired model from a small set of geometric primitives - points, lines, and polygons.

2.5 OBJECTIVES OF THE PROJECT

- Developing a package using computer graphics with OpenGL.
- Migration from text editor to OpenGL.
- To show that implementation of Translation is easier with OpenGL.
- Implementing certain technical concept like Translation, motion, and use of Idle Function.
- How to use Lightning effects used to produce computer animation.

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 USER REQUIREMENTS

- Easy to understand and should be simple.
- The built-in functions should be utilized to the maximum extent.
- OpenGL library facilities should be used.

3.2 HARDWARE REQUIREMENTS

- Intel i5 5th gen 2.1 Ghz
- 16 GB RAM
- QWERTY Keyboard
- Monitor resolution 800x600

3.3 SOFTWARE REQUIREMENTS

- OpenGL Tools
- Windows 64-Bit Operating System
- Turbo C++ on Dos/Windows platform
- Eclipse IDE: Microsoft Visual Studio 2020

CHAPTER 4

DESIGN AND IMPLEMENTATION

To design the "PULWAMA ATTACK" using the glut library, we need to understand various concepts, components and utility functions that are essential to implement/integrate the required visual (and audio) effects. Hence by using the following functions we design our projects.

4.1 DESIGN

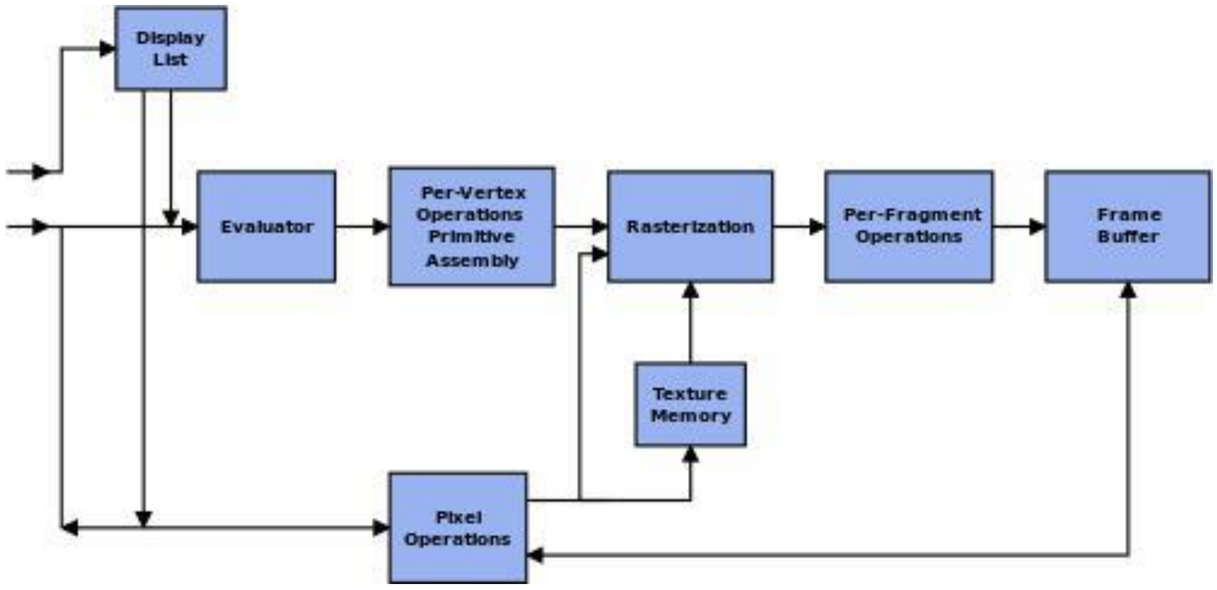

FIGURE 4.1 OPENGL PIPELINE

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 number of functions which may be called by the client program, alongside a number of named integer constants (for example, the constant GL_TEXTURE_2D, which corresponds to the decimal number 3553). Although the function definitions are superficially similar to those of the C programming language, they are language-independent. As such, OpenGL has many language bindings, some of the most noteworthy being the JavaScript binding WebGL (API, based on OpenGL ES 2.0, for 3D rendering from within a web browser); the C bindings WGL, GLX and CGL; the C binding provided by iOS; and the Java and C bindings provided by Android.

4.2 HEADER FILES

#include<stdio.h>

'stdio.h' which stands for "standard input/output header" is the header in the C standard library that contains macro definitions, constants and declarations of functions and types used for various standard input and output operations.

#include<math.h>

'math.h' is a header file in the standard library of the C programming language designed for basic mathematical operation.

#include<GL/glut.h>

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.

#include<stdbool.h>

The header '**stdbool.h**' in the C Standard Library for the C programming language contains four macros for a Boolean data type. This header was introduced in C99. The macros as defined in the ISO C standard are: bool which expands to _Bool.

4.3 SIMPLE GEOMETRY

Void glBegin(glEnum mode): This function initiates a new primitive of type mode and starts the collection of vertices. Values of this mode include GL_POINTS, GL_LINE_STRIP and GL_QUADS. **Void glEnd ():** Terminates a list of vertices.

glVertex2f(coordinates): This function defines the vertices of 2D figure with float as data type. **GlutInit(int argc,char *argv):** Initialize GLUT. The arguments from main arepassed in and can be used by the application.

glutCreateWindow(char*title): Create 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.

glutDisplayMode(unsigned int mode): Request 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)

glutInitWindowSize(intwidth,int height): Specifies the initial height and width of the window in pixels.

glutInitWindowPosition(int x, int y): Specifies the initial position of top-left corner of the window in pixels.

glutMainLoop(): Causes the program to enter an event processing loop. It should be the last statement in main.

glutDisplayFunc(void (*func)(void)): Registers the display function that is executed when the window needs to be redrawn.

glutPostRedisplay(): Requests that the display callback be executed after the current callback returns

4.5 INTERACTION

glutKeyboardfunc (void(*func) (unsigned char key, int x, int y)): Sets the keyboard callback for the current window. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback.

glutIdleFunc (void(*f)(void)P): Returns an identifier for a top-level menu and registers the callback function f that returns an integer value corresponding to the menu entry selected

4.6 TRANSFORMATION

glMatrixMode(GL_PROJECTION): Here the mode will be projection mode, specifies subsequent transformation matrix to an identity matrix.

glLoadIdentity(): Set the current transformation matrix to an identity stack corresponding to the current matrix mode.

4.7 VIEWING

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

CHAPTER 5

SOURCE CODE

```
#include<ctype.h>
#include<GL/glut.h>
#include<math.h>
#include<stdio.h>
#define M_PI 3.14159265358979323846264338327950288419716939937510
#define false 0
#define true 1
const int BOARD_X = 31;
const int BOARD_Y = 28;
int board_array[BOARD_X][BOARD_Y] =
{{8,5,5,5,5,5,5,5,5,5,5,5,5,1,1,5,5,5,5,5,5,5,5,5,5,5,5,7},
{6,0,0,0,0,0,0,0,0,0,0,0,0,2,4,0,0,0,0,0,0,0,0,0,0,0,0,6},
{6,0,8,1,1,7,0,8,1,1,1,7,0,2,4,0,8,1,1,1,7,0,8,1,1,7,0,6},
{6,0,2,11,11,4,0,2,11,11,11,4,0,2,4,0,2,11,11,11,4,0,2,11,11,4,0,6},
{6,0,9,3,3,10, 0,9,3,3,3,10,0,9,10,0,9,3,3,3,10,0,9,3,3,10,0,6},
{6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6},
{6,0,8,1,1,7,0,8,7,0,8,1,1,1,1,1,1,7,0,8,7,0,8,1,1,7,0,6},
{6,0,9,3,3,10,0,2,4,0,9,3,3,11,11,3,3,10,0,2,4,0,9,3,3,10,0,6},
{6,0,0,0,0,0,0,2,4,0,0,0,0,2,4,0,0,0,0,2,4,0,0,0,0,0,0,6},
{9,5,5,5,5,7,0,2,11,1,1,7,0,2,4,0,8,1,1,11,4,0,8,5,5,5,5,10},
{0,0,0,0,0,6,0,2,11,3,3,10,0,9,10,0,9,3,3,11,4,0,6,0,0,0,0,0},
{0,0,0,0,0,6,0,2,4,0,0,0,0,0,0,0,0,0,0,2,4,0,6,0,0,0,0,0},
{0,0,0,0,0,6,0,2,4,0,8,5,5,1,1,5,5,7,0,2,4,0,6,0,0,0,0,0},
{5,5,5,5,5,10,0,9,10,0,6,0,0,0,0,0,0,6,0,9,10,0,9,5,5,5,5,5},
{0,0,0,0,0,0,0,0,0,0,6,0,0,0,0,0,0,6,0,0,0,0,0,0,0,0,0,0},
{5,5,5,5,5,7,0,8,7,0,6,0,0,0,0,0,0,6,0,8,7,0,8,5,5,5,5,5},
{0,0,0,0,0,6,0,2,4,0,9,5,5,5,5,5,5,10,0,2,4,0,6,0,0,0,0,0},
{0,0,0,0,0,6,0,2,4,0,0,0,0,0,0,0,0,0,0,2,4,0,6,0,0,0,0,0},
{0,0,0,0,0,6,0,2,4,0,8,1,1,1,1,1,1,7,0,2,4,0,6,0,0,0,0,0},
{8,5,5,5,5,10,0,9,10,0,9,3,3,11,11,3,3,10,0,9,10,0,9,5,5,5,5,7},
{6,0,0,0,0,0,0,0,0,0,0,0,0,2,4,0,0,0,0,0,0,0,0,0,0,0,0,6},
{6,0,8,1,1,7,0,8,1,1,1,7,0,2,4,0,8,1,1,1,7,0,8,1,1,7,0,6},
{6,0,9,3,11,4,0,9,3,3,3,10,0,9,10,0,9,3,3,3,10,0,2,11,3,10,0,6},
{6,0,0,0,2,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,4,0,0,0,6},
{2,1,7,0,2,4,0,8,7,0,8,1,1,1,1,1,1,7,0,8,7,0,2,4,0,8,1,4},
{2,3,10,0,9,10,0,2,4,0,9,3,3,11,11,3,3,10,0,2,4,0,9,10,0,9,3,4},
{6,0,0,0,0,0,0,2,4,0,0,0,0,2,4,0,0,0,0,2,4,0,0,0,0,0,0,6},
{6,0,8,1,1,1,1,11,11,1,1,7,0,2,4,0,8,1,1,11,11,1,1,1,1,7,0,6},
{6,0,9,3,3,3,3,3,3,3,3,10,0,9,10,0,9,3,3,3,3,3,3,3,3,10,0,6},
{6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6},
{9,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,10}};
int pebble_array[BOARD_X][BOARD_Y] =
{{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,0},
{0,1,0,0,0,0,1,0,0,0,0,0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,1,0},
{0,3,0,0,0,0,1,0,0,0,0,0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,3,0},
{0,1,0,0,0,0,1,0,0,0,0,0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,1,0},
{0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0},
{0,1,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,1,0},
{0,1,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,1,0},
{0,1,1,1,1,1,1,0,0,1,1,1,1,0,0,1,1,1,1,0,0,1,1,1,1,1,1,0},
{0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0},
{0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0},
{0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0},
{0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0},
{0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0},
{0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0},
{0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0},
{0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0},
{0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0},
{0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0},
{0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0},
{0,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,0},
{0,1,0,0,0,0,1,0,0,0,0,0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,1,0},
{0,1,0,0,0,0,1,0,0,0,0,0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,1,0},
{0,3,1,1,0,0,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,0,0,1,1,3,0},
{0,0,0,1,0,0,1,0,0,1,0,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,0,0},
{0,0,0,1,0,0,1,0,0,1,0,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,0,0},
{0,1,1,1,1,1,1,0,0,1,1,1,1,0,0,1,1,1,1,0,0,1,1,1,1,1,1,0},
{0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0},
{0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0},
{0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}};
GLubyte list[5];
int tp_array[31][28];
int pebbles_left;
double speed1 = 0.1;
double angle1 = 90;
double a=13.5, b=23;
bool animate = false;
int lives=3;
int points=0;
void keys();
unsigned char ckey='w';
void mykey(unsigned char key,int x,int y);
bool Open(int a,int b);
void Move()
{
a += speed1*cos(M_PI/180*angle1);
b += speed1*sin(M_PI/180*angle1);
if(animate&&ckey==GLUT_KEY_UP&& (int) a - a > -0.1 && angle1 != 270) //w
{
if (Open(a,b-1))
{
animate = true;
angle1 = 270;
}
}
else if(animate&&ckey==GLUT_KEY_DOWN&& (int) a - a > -0.1 && angle1 != 90)// s
{
if (Open(a,b+1))
{
animate = true;
angle1= 90;
}
}
else if(animate&&ckey==GLUT_KEY_LEFT&& (int) b - b > -0.1 && angle1 != 180)//a
{
if (Open(a-1,b))
{
animate = true;
angle1 = 180;
}
}
else if(animate&&ckey==GLUT_KEY_RIGHT&& (int) b - b > -0.1 && angle1 != 0)//d
{
if (Open(a+1,b))
{
animate = true;
angle1 = 0;
}
}
}
void Pac(void)
{
//Draw Pacman
glColor3f(0,1,1);
glPushMatrix();
glTranslatef(a,-b,0);
glTranslatef(0.5,0.6,0);
glTranslatef((float)BOARD_X/-2.0f,(float)BOARD_Y/2.0f,0.5);
glutSolidSphere(0.5,15,10);
glPopMatrix();
}
//Monster Drawing And Moving Begins
bool open_move[4];
bool gameover = false;
int num_ghosts = 4;
int start_timer=3;
class Ghost
{
private:
public:
bool edible;
int edible_max_time;
int edible_timer;
```

```
bool eaten;
bool transporting;
float color[3];
double speed;
double max_speed;
bool in_jail;
int jail_timer;
double angle;
double x, y;
Ghost(double, double);
~Ghost(void);
void Move(); //Move the Monster
void Update(void); //Update Monster State
void Chase(double, double, bool*); //Chase Pacman
bool Catch(double, double); //collision detection
void Reinit(void);
void Vulnerable(void);
void Draw(void); //Draw the Monster
void game_over(void);
};
Ghost *ghost[4];
Ghost::~Ghost(void){}
Ghost::Ghost(double tx, double ty)
{
tx = x;
ty = y;
angle = 90;
speed = max_speed=1;
color[0] = 1;
color[1] = 0;
color[2] = 0;
eaten = false;
edible_max_time =300;
edible = false;
in_jail = true;
jail_timer = 30;
}
void Ghost::Reinit(void)
{
edible = false;
in_jail = true;
angle = 90;
}
//Move Monster
void Ghost::Move()
{
x += speed*cos(M_PI/180*angle);
y += speed*sin(M_PI/180*angle);
}
void Ghost::game_over()
```

```
{
}
void Ghost::Update(void)
{
if ((int)x == 0 && (int) y == 14 && (!(transporting)))
{
angle=180;
}
if (x < 0.1 && (int) y == 14)
{
x = 26.9;
transporting = true;
}
if ((int)x == 27 && (int) y == 14 && (!(transporting)))
{
angle=0;
}
if (x > 26.9 && (int) y == 14)
{
x = 0.1;
transporting = true;
}
if ((int)x == 2 || (int)x == 25)
transporting = false;
if (((int) x < 5 || (int) x > 21) && (int) y == 14 && !edible && !eaten)
speed = max_speed/2;
speed = max_speed;
//edibility
if (edible_timer == 0 && edible && !eaten)
{
edible = false;
speed = max_speed;
}
if (edible)
edible_timer--;
//JAIL
if (in_jail && (int) (y+0.9) == 11)
{
in_jail = false;
angle = 180;
}
if (in_jail && ((int)x == 13 || (int)x == 14))
{
angle = 270;
}
//if time in jail is up, position for exit
if (jail_timer == 0 && in_jail)
{
//move right to exit
if (x < 13)
angle = 0;
if (x > 14)
angle = 180;
}
//decrement time in jail counter
if (jail_timer > 0)
jail_timer--;
//EATEN GHOST SEND TO JAIL
if (eaten && ((int) x == 13 || (int) (x+0.9) == 14) && ((int)y > 10 && (int) y < 15))
{
in_jail = true;
angle = 90;
if((int) y == 14)
{
eaten = false;
speed = max_speed;
jail_timer = 66;
x = 11;
}
}
}
bool Ghost::Catch(double px, double py)
{
// Collision Detection
if (px - x < 0.2 && px - x > -0.2 && py - y < 0.2 && py - y > -0.2)
{
return true;
}
return false;
}
//called when pacman eats a super pebble
void Ghost::Vulnerable(void)
{
if (!(edible))
{
angle = ((int)angle + 180)%360;
speed = max_speed;
}
edible = true;
edible_timer = edible_max_time;
//speed1=0.15;
}
void Ghost::Chase(double px, double py, bool *open_move)
{
int c;
if (edible)
c = -1;
else
c = 1;
bool moved = false;
```

```
if ((int) angle == 0 || (int) angle == 180)
{
if ((int)c*py > (int)c*y && open_move[1])
angle = 90;
else if ((int)c*py < (int)c*y && open_move[3])
angle = 270;
}
else if ((int) angle == 90 || (int) angle == 270)
{
if ((int)c*px > (int)c*x && open_move[0])
angle = 0;
else if ((int)c*px < (int)c*x && open_move[2])
angle = 180;
}
if ((int) angle == 0 && !open_move[0])
angle = 90;
if ((int) angle == 90 && !open_move[1])
angle = 180;
if ((int) angle == 180 && !open_move[2])
angle = 270;
if ((int) angle == 270 && !open_move[3])
angle = 0;
if ((int) angle == 0 && !open_move[0])
angle = 90;
}
void Ghost::Draw(void)
{
if (!edible)
glColor3f(color[0],color[1],color[2]);
else
{
if (edible_timer < 150)
glColor3f((edible_timer/10)%2,(edible_timer/10)%2,1);
if (edible_timer >= 150)
glColor3f(0,0,1);
}
if (eaten)
glColor3f(1,1,0); //When Eaten By PacMan Change Color To Yellow
glPushMatrix();
glTranslatef(x,-y,0);
glTranslatef(0.5,0.6,0);
glTranslatef((float)BOARD_X/-2.0f, (float)BOARD_Y/2.0f,0.5);
glutSolidSphere(.5,10,10);
glPopMatrix();
}
void tp_restore(void)
{
for (int ISO = 0; ISO < BOARD_X; ISO++)
{
for (int j = 0; j < BOARD_Y; j++)
```

```
{
tp_array[ISO][j] = pebble_array[ISO][j];
}
}
pebbles_left = 244;
}
void Draw(void)
{
glColor3f(1,0,1);
for (int ISO = 0; ISO < BOARD_X; ISO++)
{
for (int j = 0; j < BOARD_Y/2; j++)
{
glColor3f(0,0,1);
int call_this = 0;
glPushMatrix();
glTranslatef(-(float) BOARD_X / 2.0f,-(float) BOARD_Y / 2.0f, 0);
glTranslatef(j, BOARD_Y - ISO,0);
glPushMatrix();
glTranslatef(0.5,0.5,0);
switch (board_array[ISO][j])
{
case 4:
glRotatef(90.0,0,0,1);
case 3:
glRotatef(90.0,0,0,1);
case 2:
glRotatef(90.0,0,0,1);
case 1:
call_this = 1;
break;
case 6:
glRotatef(90.0,0,0,1);
case 5:
call_this = 2;
break;
case 10:
glRotatef(90.0,0,0,1);
case 9:
glRotatef(90.0,0,0,1);
case 8:
glRotatef(90.0,0,0,1);
case 7:
call_this = 3;
break;
}
glScalef(1,1,0.5);
glTranslatef(-0.5,-0.5,0);
glCallList(list[call_this]);
glPopMatrix();
```

```
//now put on the top of the cell
if (call_this != 0 || board_array[ISO][j] == 11)
{
glTranslatef(0,0,-0.5);
glCallList(list[4]);
}
glPopMatrix();
if (tp_array[ISO][j] > 0)
{
glColor3f(0,300,1/(float)tp_array[ISO][j]);
glPushMatrix();
glTranslatef(-(float) BOARD_X / 2.0f,-(float) BOARD_Y / 2.0f, 0);
glTranslatef(j, BOARD_Y - ISO,0);
glTranslatef(0.5,0.5,0.5);
glutSolidSphere(0.1f*((float)tp_array[ISO][j]),6,6);
glPopMatrix();
}
}
}
int ISO;
for (ISO= 0; ISO< BOARD_X; ISO++)
{
for (int j = BOARD_Y-1; j >= BOARD_Y/2; j--)
{
glColor3f(0,0,1);
int call_this = 0;
glPushMatrix();
glTranslatef(-(float) BOARD_X / 2.0f,-(float) BOARD_Y / 2.0f, 0);
glTranslatef(j, BOARD_Y - ISO,0);
glPushMatrix();
glTranslatef(0.5,0.5,0);
switch (board_array[ISO][j])
{
case 4:
glRotatef(90.0,0,0,1);
case 3:
glRotatef(90.0,0,0,1);
case 2:
glRotatef(90.0,0,0,1);
case 1:
call_this = 1;
break;
case 6:
glRotatef(90.0,0,0,1);
case 5:
call_this = 2;
break;
case 10:
glRotatef(90.0,0,0,1);
case 9:
```

```
glRotatef(90.0,0,0,1);
case 8:
glRotatef(90.0,0,0,1);
case 7:
call_this = 3;
break;
}
glScalef(1,1,0.5);
glTranslatef(-0.5,-0.5,0);
glCallList(list[call_this]);
glPopMatrix();
//now put on top
if (call_this != 0 || board_array[ISO][j] == 11)
{
glTranslatef(0,0,-0.5);
glCallList(list[4]);
}
glPopMatrix();
if (tp_array[ISO][j] > 0)
{
glColor3f(0,300,1/(float)tp_array[ISO][j]);
glPushMatrix();
glTranslatef(-(float) BOARD_X / 2.0f,-(float) BOARD_Y / 2.0f, 0);
glTranslatef(j, BOARD_Y - ISO,0);
glTranslatef(0.5,0.5,0.5);
glutSolidSphere(0.1f*((float)tp_array[ISO][j]),6,6);
glPopMatrix();
}
}
}
Pac();
}
bool Open(int a, int b)
{
if (board_array[b][a] > 0)
{
return false;
}
return true;
}
void RenderScene();
void mykey(unsigned char key,int x,int y)
{
if (start_timer > 0)
{
start_timer--;
}
}
void specialDown(int key,int x,int y)
{
```

```
if (start_timer > 0)
start_timer--;
ckey=key;
if(key==GLUT_KEY_UP&& (int) a - a > -0.1 && angle1 != 270) //w
{
if (Open(a, b - 1))
{
animate = true;
angle1 = 270;
}
}
else if(key==GLUT_KEY_DOWN&& (int) a - a > -0.1 && angle1 != 90)// s
{
if (Open(a,b + 1))
{
animate = true;
angle1= 90;
}
}
else if(key==GLUT_KEY_LEFT&& (int) b - b > -0.1 && angle1 != 180)//a
{
if (Open(a-1,b))
{
animate = true;
angle1 = 180;
}
}
else if(key==GLUT_KEY_RIGHT&& (int) b - b > -0.1 && angle1 != 0)//d
{
if (Open(a+1, b))
{
animate = true;
angle1 = 0;
}
}
}
void specialUp(int key,int x,int y)
{
}
void P_Reinit()
{
a = 13.5;
b = 23;
angle1 = 90;
animate = false;
Pac();
}
void G_Reinit(void)
{
start_timer = 3;
```

```
//ghost initial starting positions
int start_x[4] = {11,12,15,16};
float ghost_colors[4][3] = {{255,0,0},{120,240,120},{255,200,200},{255,125,0}};
for (int i = 0; i < num_ghosts; i++)
{
ghost[i]->Reinit();
ghost[i]->x = start_x[i];
ghost[i]->y = 14;
ghost[i]->eaten = false;
ghost[i]->jail_timer = i*33 + 66;
ghost[i]->max_speed = 0.1 - 0.01*(float)i;
ghost[i]->speed = ghost[i]->max_speed;
//colorize ghosts
for (int j = 0; j < 3; j++)
ghost[i]->color[j] = ghost_colors[i][j]/255.0f;
}
}
void renderBitmapString(float x, float y, void *font, char *string)
{
char *c;
glRasterPos2f(x,y);
for (c=string; *c != '\0'; c++)
{
glutBitmapCharacter(font, *c);
}
}
void Write(char *string)
{
while(*string)
glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, *string++);
}
void print(char *string)
{
while(*string)
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *string++);
}
//Display Function->This Function Is Registered in glutDisplayFunc
void RenderScene()
{
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
//Through Movement->From One End To The Other
if ((int)a == 27 && (int) b == 14 && angle1 == 0)
{
a = 0;
animate = true;
}
else
if ((int)(a + 0.9) == 0 && (int) b == 14 && angle1 == 180)
{
a = 27;
```

```
animate = true;
}
//Collision Detection For PacMan
if (animate)
Move();
if(!(Open((int)(a + cos(M_PI/180*angle1)),
(int)(b + sin(M_PI/180*angle1)))) &&
a - (int)a < 0.1 && b - (int)b < 0.1)
animate = false;
if (tp_array[(int)(b+0.5)][(int)(a+0.5)]== 1)
{
tp_array[(int)(b+0.5)][(int)(a+0.5)]= 0;
pebbles_left--;
points+=1;
}
//Super Pebble Eating
else if(tp_array[(int)(b+0.5)][(int)(a+0.5)] == 3)
{
tp_array[(int)(b+0.5)][(int)(a+0.5)]= 0;
pebbles_left--;
points+=5;
for (int i = 0; i < 4; i++)
{
if (!ghost[i]->eaten)
ghost[i]->Vulnerable(); //Calls A Function To Make Monster Weak
}
}
if (pebbles_left == 0)
{
G_Reinit();
P_Reinit();
tp_restore();
points=0;
lives=3;
}
if (!gameover)
Draw();
for (int d = 0; d < num_ghosts; d++)
{
if (!gameover && start_timer == 0)
ghost[d]->Update();
if (!ghost[d]->in_jail &&
ghost[d]->x - (int)ghost[d]->x < 0.1 && ghost[d]->y - (int)ghost[d]->y < 0.1)
{
bool open_move[4];
//Finding Moves
for (int ang = 0; ang < 4; ang++)
{
open_move[ang] = Open((int)(ghost[d]->x + cos(M_PI/180*ang*90)),
(int)(ghost[d]->y + sin(M_PI/180*ang*90)));
}
//Chase Pac Man
if (!ghost[d]->eaten)
{
if(ghost[d]->x - (int)ghost[d]->x < 0.1 && ghost[d]->y - (int)ghost[d]->y < 0.1)
ghost[d]->Chase(a, b, open_move);
}
else
{
if(ghost[d]->x - (int)ghost[d]->x < 0.1 && ghost[d]->y - (int)ghost[d]->y < 0.1)
ghost[d]->Chase(13, 11, open_move);
}
}
if (ghost[d]->in_jail && !(Open((int)(ghost[d]->x + cos(M_PI/180*ghost[d]->angle)),
(int)(ghost[d]->y + sin(M_PI/180*ghost[d]->angle)))) && ghost[d]->jail_timer > 0
&&ghost[d]->x - (int)ghost[d]->x < 0.1 && ghost[d]->y - (int)ghost[d]->y < 0.1)
{
ghost[d]->angle = (double)(((int)ghost[d]->angle + 180)%360);
}
if (!gameover && start_timer == 0)
ghost[d]->Move();
ghost[d]->Draw();
if(!(ghost[d]->eaten))
{
bool collide = ghost[d]->Catch(a,b);
//Monster Eats PacMan
if (collide && !(ghost[d]->edible))
{
lives--;
if (lives == 0)
{
gameover = true;
lives=0;
ghost[d]->game_over();
}
P_Reinit();
d = 4;
}
//PacMan Eats Monster And Sends It To Jail
else if (collide && ((ghost[d]->edible)))
{
ghost[d]->edible = false;
ghost[d]->eaten = true;
ghost[d]->speed = 1;
}
}
}
if(gameover==true)
{
glColor3f(1,0,0);
```

```
renderBitmapString(-5, 0.5,GLUT_BITMAP_HELVETICA_18 ,"GAME OVER");
}
char tmp_str[40];
glColor3f(1, 1, 0);
glRasterPos2f(10, 18);
sprintf(tmp_str, "Points: %d", points);
Write(tmp_str);
glColor3f(1, 0, 0);
glRasterPos2f(-5, 18);
sprintf(tmp_str, "PAC MAN");
print(tmp_str);
glColor3f(1, 1, 0);
glRasterPos2f(-12, 18);
sprintf(tmp_str, "Lives: %d", lives);
Write(tmp_str);
glutPostRedisplay();
glutSwapBuffers();
}
void create_list_lib()
{
list[1] = glGenLists(1);
glNewList(list[1], GL_COMPILE);
glBegin(GL_QUADS);
glColor3f(0,0,1);
glNormal3f(0.0, 1.0, 0.0);
glVertex3f(1.0, 1.0, 1.0);
glVertex3f(1.0, 1.0, 0.0);
glVertex3f(0.0, 1.0, 0.0);
glVertex3f(0.0, 1.0, 1.0);
glEnd();
glEndList();
list[2] = glGenLists(1);
glNewList(list[2], GL_COMPILE);
glBegin(GL_QUADS);
glColor3f(0,0,1);
glNormal3f(0.0, 1.0, 0.0);
glVertex3f(1.0, 1.0, 1.0);
glVertex3f(1.0, 1.0, 0.0);
glVertex3f(0.0, 1.0, 0.0);
glVertex3f(0.0, 1.0, 1.0);
glColor3f(0,0,1);
glNormal3f(0.0, -1.0, 0.0);
glVertex3f(1.0, 0.0, 0.0);
glVertex3f(1.0, 0.0, 1.0);
glVertex3f(0.0, 0.0, 1.0);
glVertex3f(0.0, 0.0, 0.0);
glEnd();
glEndList();
list[3] = glGenLists(1);
glNewList(list[3], GL_COMPILE);
```

```
glBegin(GL_QUADS);
glColor3f(0,0,1);
glNormal3f(0.0f, 1.0f, 0.0f);
glVertex3f(1.0, 1.0, 1.0);
glVertex3f(1.0, 1.0, 0.0);
glVertex3f(0.0, 1.0, 0.0);
glVertex3f(0.0, 1.0, 1.0);
glColor3f(0,0,1);
glNormal3f(1.0, 0.0, 0.0);
glVertex3f(1.0, 1.0, 0.0);
glVertex3f(1.0, 1.0, 1.0);
glVertex3f(1.0, 0.0, 1.0);
glVertex3f(1.0, 0.0, 0.0);
glEnd();
glEndList();
list[4] = glGenLists(1);
glNewList(list[4], GL_COMPILE);
glBegin(GL_QUADS);
glColor3f(-1,0.3,0);
glNormal3f(1.0, 0.0, 1.0);
glVertex3f(1, 1, 1.0);
glVertex3f(0, 1, 1.0);
glVertex3f(0, 0, 1.0);
glVertex3f(1, 0, 1.0);
glEnd();
glEndList();
}
void init()
{
/* float color[4];
Enable Lighting.
glEnable(GL_LIGHT0);
glEnable(GL_LIGHTING);
Ambient And Diffuse Lighting
glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
glEnable(GL_COLOR_MATERIAL);
color[0] = 1.0f; color[1] = 1.0f; color[2] = 0.0f; color[3] = 0.0f;
glLightfv(GL_LIGHT0, GL_DIFFUSE, color);
color[0] = 1.0f; color[1] = 0.0f; color[2] = 1.0f; color[3] = 1.0f;
glLightfv(GL_LIGHT0, GL_AMBIENT, color);*/
glEnable(GL_NORMALIZE);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(60,1.33,0.005,100);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(-1.5, 0, 40, -1.5, 0, 0, 0.0f,1.0f,0.0f);
}
void erase()
{
```

```
glColor3f(0.1,0.0,0.0);
glBegin(GL_POLYGON);
glVertex2f(0,0);
glVertex2f(0.5,0);
glVertex2f(0.25,0.5);
glEnd();
}
int main(int argc,char **argv)
{
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH );
glutInitWindowSize(1200, 780);
glutInitWindowPosition(0,0);
glutCreateWindow("Pac GL 3D");
init();
glutDisplayFunc(RenderScene);
create_list_lib();
glutKeyboardFunc(mykey);
glutSpecialFunc(specialDown);
glutSpecialUpFunc(specialUp);
glEnable(GL_DEPTH_TEST);
int start_x[4] = {11,12,15,16};
for (int ISO = 0; ISO < num_ghosts; ISO++)
{
ghost[ISO] = new Ghost(start_x[ISO],14);
}
float ghost_colors[4][3] = {{255,0,0},{120,240,120},{255,200,200},{255,125,0}};
int ISO;
for (ISO = 0; ISO < num_ghosts; ISO++)
{
ghost[ISO]->x = start_x[ISO];
ghost[ISO]->y = 14;
ghost[ISO]->eaten = false;
ghost[ISO]->max_speed = 0.1 - 0.01*(float)ISO;
ghost[ISO]->speed = ghost[ISO]->max_speed;
for (int j = 0; j < 3; j++)
ghost[ISO]->color[j] = ghost_colors[ISO][j]/255.0f;
}
for ( ISO = 0; ISO < BOARD_X; ISO++)
{
for (int j = 0; j < BOARD_Y; j++)
{
tp_array[ISO][j] = pebble_array[ISO][j];
}
}
pebbles_left = 244;
glShadeModel(GL_SMOOTH);
glutMainLoop();
return 0;
}
```

CHAPTER 6

SNAPSHOTS

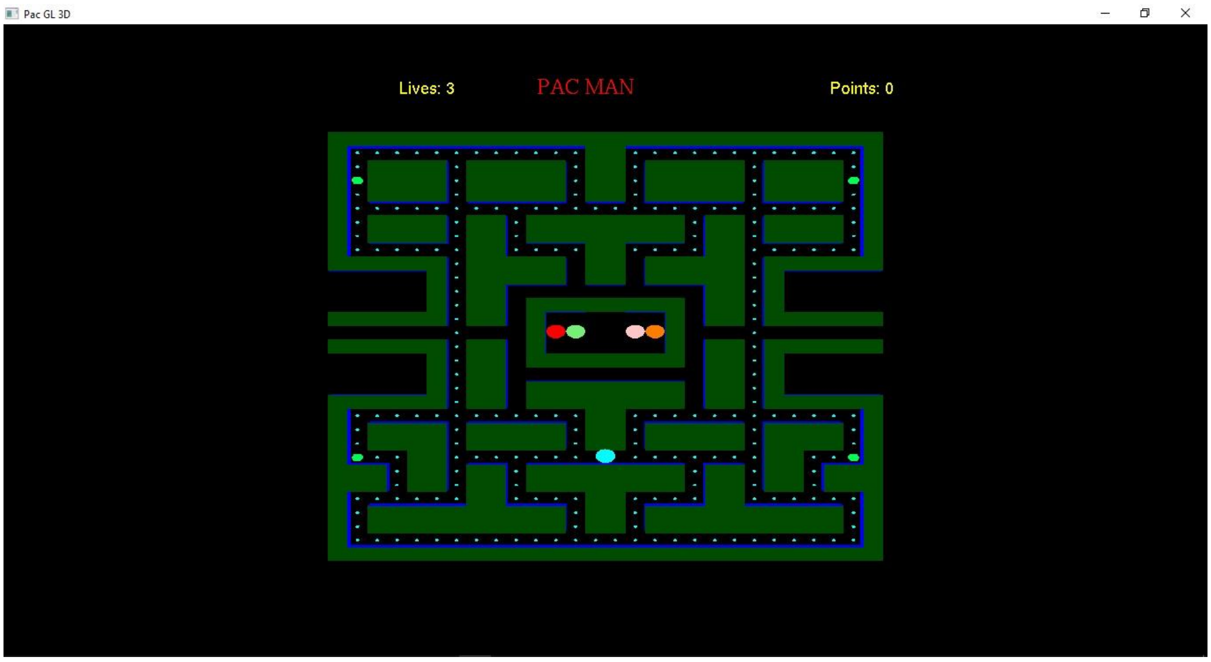

Fig 1: Beginning of the Game

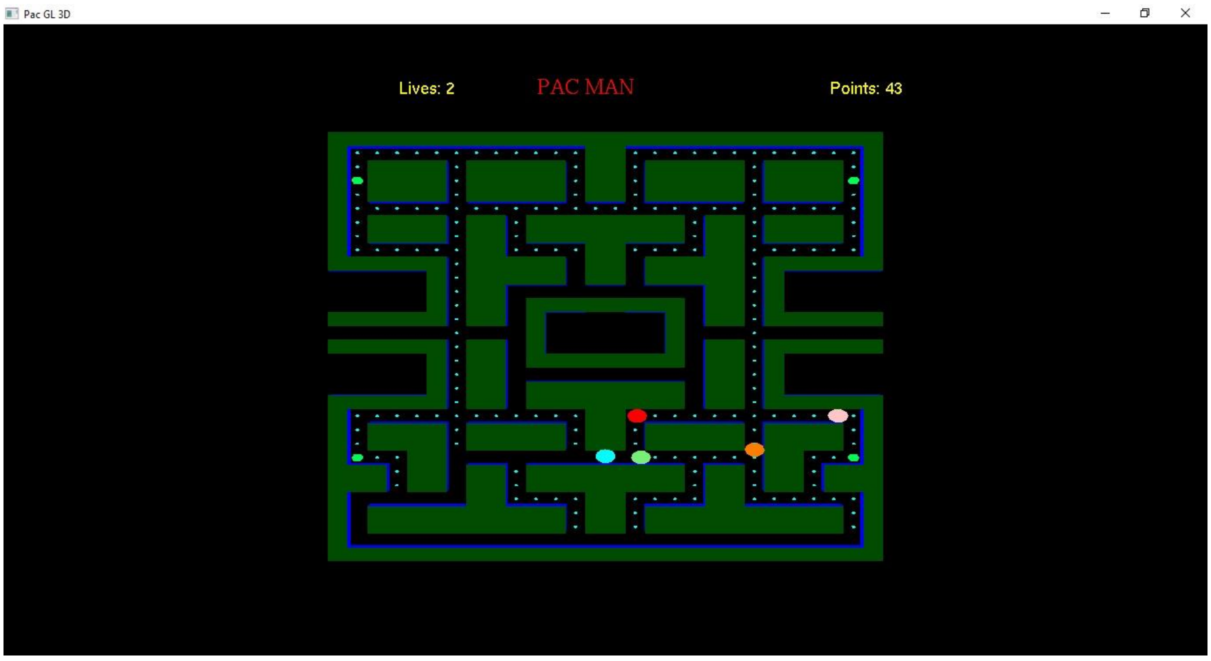

Fig 2: During the Game

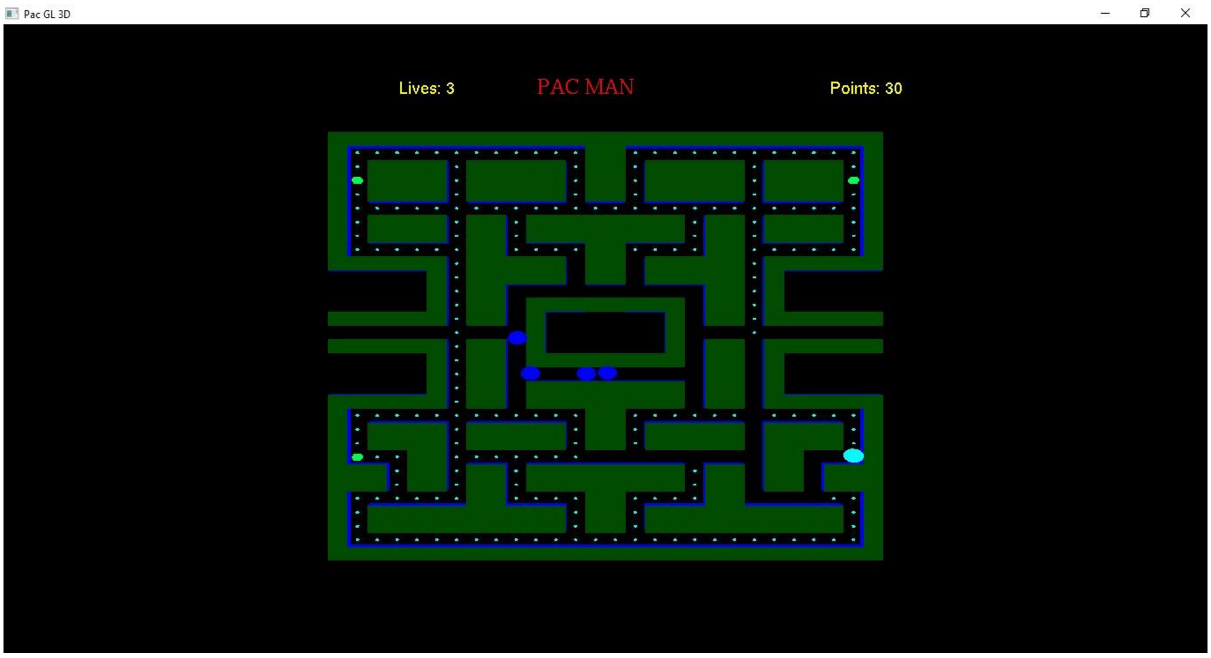

Fig 3: Safe Mode during Game

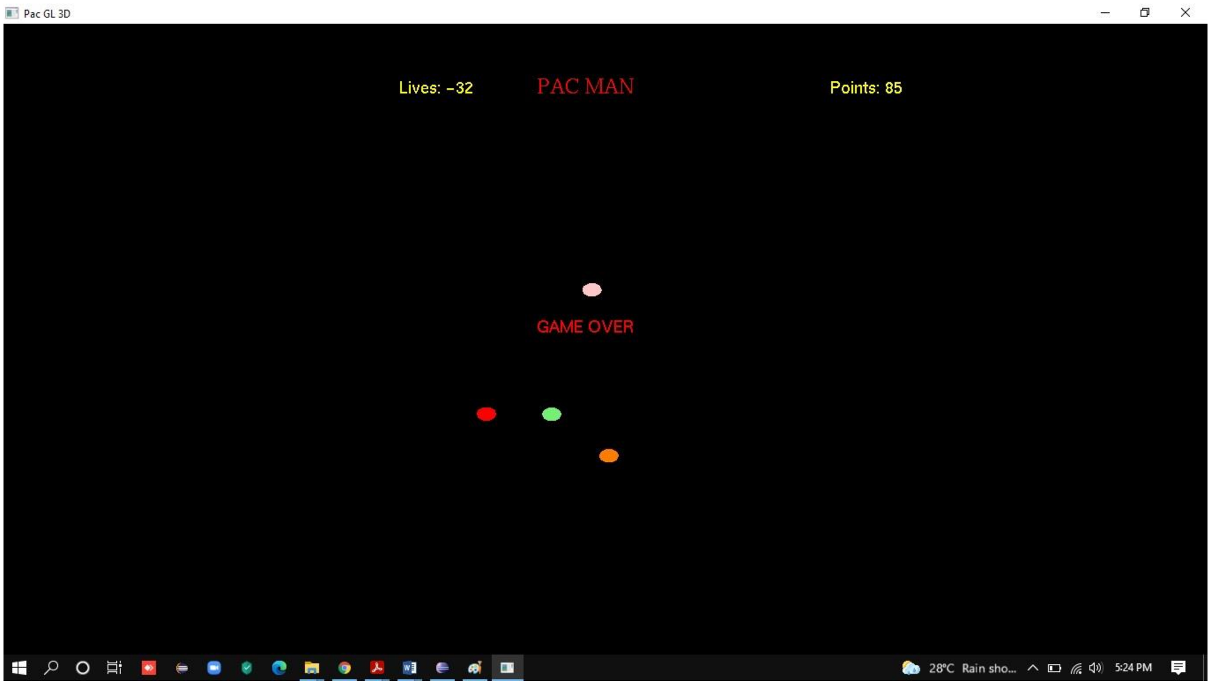

Fig 4: Game Over

CONCLUSION

We have attempted to design and implement "Pac-Man". OpenGl supports enormous flexibility in the design and the use of OpenGl graphics programs. The presence of many built in classes methods take care of much functionality and reduce the job of coding as well as makes the implementation simpler. The project was started with the designing phase in which we figured the requirements needed, the layout design, then comes the detail designing of each function after which, was the testing and debugging stage. We have tried to implement the project making it as user-friendly and error free as possible.

FURTHER SCOPE

Scope of further improvement:

- The game can include 3D graphics.
- Various lightening effects can be implemented to make it more attractive.
- The design of the blocks can be made more attractive by using various shading techniques.

BIBLIOGRAPHY

Books:

The books that helped us in implementing this project are as follows:

1.Computer Graphics (OpenGL Version)

2. Donald Hearn and Pauline Baker, 2nd edition, Pearson Education,2003

3. Interactive Computer Graphics

4. Edward Angel, 5th edition, Addison Wesley, 2009

Reference Websites:

1. www.opengl.com

2. www.learnopengl.com

3. www.tutorialspoint.com

4.http://Wikipedia.com/wiki/openGL

5.http://www.stackoverflow.com

DECLARATION

We student of 6th semester BE, Computer Science and Engineering College hereby declare that project work entitled “PAC-MAN” has been carried out by us at City Engineering College, Bangalore and submitted in partial fulfilment of the course requirement for the award of the degree of Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University, Belgaum, during the academic year 2020-2021.

We also declare that, to the best of our knowledge and belief, the work reported here does not form the part of dissertation on the basis of which a degree or award was conferred on a earlier occasion on this by any other student.

Date:

Place: Bangalore

SURABHI G R	CHETANA NATH
(1CE18CS084)	(1CE18CS061)