Providing an Interactive Approach to Solve Tower of Hanoi Problem Using Open Graphics Library

Deepankar Singh and Anitha Premkumar
Dept.of.CSE, Sir MVIT, Bangalore, Karnataka, India

Abstract. This paper illustrates graphical simulation of the tower of Hanoi puzzle in Open Graphics library (OpenGL) environment. OpenGL is the premier environment for developing portable, interactive 2D and 3D graphics applications. The simulation allows the users to get the optimized solutions for ‘n’ number of disks. This setup is carefully designed to meet the performance on the computers that are generally built with low-end graphic modules. This paper discusses two way approach of problem solving: A manual approach, facilitating the user to manually solve the puzzle in a given time span. An automatic simulation, featuring the automated solution of the puzzle. This paper summarizes performance and the efficiency of the tower of Hanoi simulation using OpenGL.

Keywords: Open Graphics Library (OpenGL), Graphical simulation

1. Introduction

The term computer graphics has been used in a broad sense to describe almost everything on computers that is not text or sound. Typically, the term computer graphics refers to several different things [1]:

- the representation and manipulation of image data by a computer
- the various technologies used to create and manipulate images
- the images so produced
- The sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content see study of computer graphics.

Today, computers and computer-generated images touch many aspects of daily life. Computer imagery is found on television, in newspapers, for example in weather reports, or in all kinds of medical investigation and surgical procedures. A well-constructed graph can present complex statistics in a form that is easier to understand and interpret. In the media such graphs are used to illustrate papers, reports, thesis, and other presentation material.3D computer graphics uses a three-dimensional representation of geometric data that is stored in the computer for the purposes of performing calculations and rendering 2D images [2]. Such images may be for later display or for real time viewing. Despite these differences, 3D computer graphics rely on many of the same algorithms as 2D computer vector graphics in the wire frame model and 2D computer raster graphics in the final rendered display. In computer graphics software, the distinction between 2D and 3D is occasionally blurred; 2D applications may use 3D techniques to achieve effects such as lighting, and primarily 3D may use 2D rendering techniques [3].

2. Foundations

* Corresponding author. E-mail address: Ankur4u007@gmail.com, anithapkhd@gmail.com
OpenGL is the premier environment for developing portable, interactive 2D and 3D graphics applications. Since its introduction in 1992, OpenGL has become the industry’s most widely used and supported 2D and 3D graphics Application Programming Interface (API), bringing thousands of applications to a wide variety of computer platforms. OpenGL fosters innovation and speeds application development by incorporating a broad set of rendering, texture mapping, special effects, and other powerful visualization functions. Developers can leverage the power of OpenGL across all popular desktop and workstation platforms, ensuring wide application deployment [4].

Dev-C++ is a free integrated development environment (IDE) distributed under the GNU General Public License for programming in C and C++. It is bundled with MinGW, a free compiler. The IDE is written in Delphi. Bloodshed Dev-C++ is a full-featured Integrated Development Environment (IDE) for the C and C++ programming languages. It uses the MinGW port of the GCC as its compiler. Dev-C++ can also be used in combination with Cygwin or any other GCC-based compiler. One additional aspect of Dev-C++ is its use of DevPaks, packaged extensions on the programming environment with additional libraries, templates, and utilities. DevPaks often contain, but are not limited to, GUI utilities, including popular toolkits such as GTK+, wxWidgets, and FLTK. Other DevPaks include libraries for more advanced function use.

3. Implementation

The Tower Of Hanoi puzzle game is designed with an interface that allows the user to solve the puzzle within a given time period. The game is divided into series of levels, Level one starts with minimum number of disks required to carry out the operation i.e. 3 which is increased up to 12 in the Level 10. The time allocated to each level is calculated on the basis of the moves made by the disks. The game also provides a move counter that runs continuously to catch the total number of moves made while transferring the disks. The implementation is achieved in the following two ways: Manual mode and Auto mode.

3.1. Manual Mode

The game starts with an interface allowing the user to select the levels to play. The user is then presented with the set of towers containing disks that varies according to the level chosen by the user. The disks along with the towers are mounted on a Platform that continuously rotates giving a 3-Dimensional view to the user as shown in Fig (1).

![Figure 1: Manual Mode transfer of the disks](image)

The disks can be moved between the towers using the help of buttons provided at the bottom of the interface. Once a button is pressed the move counter is incremented. The user is required to solve the puzzle before the time elapses else is redirected back to the main page of the game. However, if the user is able to
solve the puzzle within the given time, the level is incremented automatically and the time counter resets with predefined time for that level. During the game running time, the user can hold down the left mouse button and move the mouse to see a 3-Dimensional view of the game. The continuous rotations of the platform can be stopped by pressing the ‘r’ button on the keyboard. The user can also zoom in or zoom out the view by right-clicking the mouse and selecting the appropriate options. The user can also choose to view solution of the particular level by pressing the ‘S’ button on the keyboard at any point of time during the game.

3.2. Auto Mode

Once the user has opted for the automatic solution of the puzzle, the disks are transferred automatically from the tower ‘A’ to tower ‘C’ such that no smaller disks can be placed on top of a larger disk in size. The transfer that takes place is based on the fact that, to solve the puzzle containing ‘n’ number of disks, the total moves required is \((2^n - 1)\) moves, hence it is optimum. The Auto Mode is carried out by maintaining a data structure of nodes that points to several other nodes in succession in form of a link list. Hence the automatic solution of the puzzle is attained by the recursion such lists. Along with the other menu options the Auto Mode also consists of ‘Speed up’ and ‘Speed Down’ feature that lets the user to decide the speed with which the automatic transfer of the disks takes place. On the completion of the transfer the user is presented with a dialog box that asks the user whether to replay the solution or to exit to the main menu of the game as shown in Fig (2).

3.3. Texture Mapping

The game is inbuilt with various .bmp image files that are processed in form of textures to produce a realistic 3-Dimensional effect. The textures are employed at the following places: the main menu, the select level page, and the buttons. The main menu texture is the interface the user encounters whenever the game is started or user chooses an option to go back to main menu. A simple rectangular box is made around the selected options that change according to the arrow keys (Up/Down) pressed by the user as shown in Fig (3).
4. Result Analysis

We have implemented and shown our result in terms of graphs. Graph 1 depicts result of manual approach and graph 2 shows the result of automated approach. In the graph x axis indicates the number of moves required for each level and y axis indicates time allocated and time taken for manual and automated approach respectively.

The above graph explains manual operation mode of solving this problem. In level 1, the number of disc is three and the number of moves is 7 (i.e. $2^3-1$). As we increase the levels, the number of disc per level increases by one. The time shown in the graph is the maximum allocated time for each level. The graph shows that time taken to complete a particular level increases as we proceed to the higher level.
Having an automated operation mode in this project, we can visually see the solution obtained by this approach. This gives idea and knowledge to the user for solving this problem with user interaction. The graph shows, automated approach takes lesser time at each level to solve it as compared to the manual approach. In manual mode, the number of moves depends on the user at each level. But in automated approach the number of moves are optimal i.e. for ‘n’ number of discs the moves required are $2^n-1$. Thus, the automated approach helps the player to interactively learn and understand the concept of how to solve the Tower of Hanoi problem. The time taken to solve this problem by automated approach is very less.

5. Conclusion

Here we have provided two approaches to resolve the problem i.e. the Manual approach and the Automated approach. An Automated approach guides the user in a right direction to solve it. In manual approach, User interaction was provided in all the levels that user wants to play. In automated approach, the solution for all levels was provided thereby increasing interactivity and performance of this project. This project includes graphical user interface for better interactive learning and understanding of the problem. Thus it increases the usability of the project.

6. References


