Kathmandu University
Department of Computer Science and Engineering
Dhulikhel, Kavre

A Project Report
on
"Tower of Hanoi Visualizer"

[Code No.: COMP 202]

(For partial fulfillment of the requirements for internal assessment in Data Structure and Algorithm)

Submitted by
Binay Bista (037962-24)

Submitted to
Mr. Sagar Acharya
Department of Computer Science and Engineering

February 25, 2026

Acknowledgements

I would like to express my heartfelt gratitude to everyone who supported and contributed for the successful completion of my mini-project titled “Tower of Hanoi Visualizer”.

I am deeply thankful to our lecturer, Mr. Sagar Acharya, for providing us with this golden opportunity through which I learned to apply what we learned in our Data Structure and Algorithm (DSA) lectures.

I would like to extend my thank to my classmates, friends, and families for their moral support and motivation during this mini-project.

Abstract

The Tower of Hanoi is a classical problem in Mathematics and Computer Science.This problem demonstrates the fundamental concepts of recursion, algorithmic efficiency, and data structures such as stacks. The Tower of Hanoi Visualizer is a C++ based graphical program that demonstrates the classic Tower of Hanoi puzzle through an interactive and animated visualization using the Raylib graphics library. The program visually demonstrate the movement of disks between three pegs -source, destination and auxilary- while strictly following the constraints of the problem: only one disk can be moved at a time and no larger disk may be placed on a smaller one.

The project implements a custom stack data structure to represent each of the three towers, where each tower follows the Last In, First Out (LIFO) principle to manage disk placement and movement.The recursive Hanoi algorithm is used to precompute all the required moves, which are then stored in a vector and executed sequentially during the animation loop, allowing the user to visually observe each disk movement step by step.

This project serves as an educational tool aimed at improving conceptual clarity and learning engagement by transforming a theoretical algorithm into an intuitive visual experience. It is particularly useful for students seeking to understand recursion, stack operations, and algorithmic problem solving through visualization.

Contents

List of Figures

Abbreviations

CPU Central Processing Unit.

CSS Cascading Style Sheets.

DSA Data Structure and Algorithm.

HTML Hyper Text Markup Language.

IDE Integrated Development Environment.

LIFO Last In, First Out.

OS Operating System.

Chapter 1

Introduction

1.1 Background

The Tower of Hanoi is a classical problem in Mathematics and Computer Science which is widely used to illustrate fundamental concepts of recursion, algorithmic thinking, and stack-based data structures. The problem involves moving a set of disks from one peg (source) to another (destination) using an auxiliary peg, while following strict rules:

- Only one disk can be moved at a time
- A larger disk cannot be placed on top of a smaller disk

Although the problem is quite simple to state, its solution demonstrates the power of recursive problem-solving and highlights the exponential growth of operations with increasing input size.

The Tower of Hanoi Visualizer is a C++ based graphical application developed by using Raylib graphics library. It visually simulates the movement of disks between three pegs using animated transitions. Each peg is implemented using a custom stack data structure that follows the LIFO principle. The recursive Hanoi algorithm pre-computes all valid moves, which are then executed step by step during the animation loop.

This approach bridges the gap between algorithmic theory and actual visual understanding by transforming a well-known recursive problem into an interactive learning experience which can also be used to teach this concept.

1.2 Objectives

- To visually demonstrate the Tower of Hanoi puzzle with the help of animation
- To help students understand recursion by observing how complex problems can be broken down into smaller subproblems and solved
- To implement and apply a custom stack data structure for managing disk movements

- To enhance understanding of algorithmic flow and constraints in problem-solving
- To create an educational visualization tool for learning DSA

1.3 Motivation and Significance

Many students find concepts of recursion and stack operations difficult to understand when taught only through code or theoretical explanation. The Tower of Hanoi is often introduced as a textbook example, but learners struggle to mentally visualize how disks moves across pegs during recursive calls.

This project is motivated by the idea that visual learning improves conceptual clarity. By animating each move of the disks in real time, students can:

- See the actual sequence of recursive operations.
- Better grasp how stacks manage disk placement.
- Understand why the number of moves grows rapidly with more disks.

The project aims to make learning more engaging and interactive by turning a classic algorithm into a visual simulation rather than a purely theoretical exercise taught on classes.

- **Educational Value:**This helps students develop a deeper understanding of recursion, stacks, and algorithms through visualization.
- **Conceptual Clarity:** Transforms an complex recursive algorithm into a step-by-step visual process, making learning easier and fun.
- **Practical Skill Development:**This project provides hands-on experience with programs such as:
  - C++ programming
  - Custom data structure implementation
  - Graphics rendering using the Raylib library
- **Learning Engagement:** Interactive visualization increases curiosity and engagement compared to static code examples.
- **Academic Use:** Can be used as a teaching aid in DSA courses or as a demonstration tool during presentations and lab sessions.

Chapter 2

Related Works

The Tower of Hanoi problem has been widely studied and is implemented in various forms across educational platforms such as books and lectures, programming tutorials, and visualization tools.

Several online interactive simulators and educational websites provide visual demonstrations of the Tower of Hanoi puzzle, allowing users to manually move disks or observe automated solutions. For example, the interactive visualization available on GeeksforGeeks explains the recursive algorithm along with sample code implementations in multiple programming languages.

There also exist web-based visualizers developed using Hyper Text Markup Language (HTML),Cascading Style Sheets (CSS), and JavaScript that demonstrate disk movements with simple animations. While these tools are useful for conceptual understanding, they often abstract away the internal implementation details such as stack operations and low-level memory management.

In comparison, the proposed Tower of Hanoi Visualizer is implemented in C++ using the Raylib graphics library and incorporates a custom stack data structure to represent each tower. Unlike many existing solutions that rely on built-in data structures or pre-defined animation frameworks, this project emphasizes data structure implementation, recursion, and graphics programming together. This makes the project more suitable as a learning exercise for students studying Data Structures and Algorithms, as well as basic computer graphics programming.

Additionally, some academic demonstrations and classroom tools illustrate recursion through pseudocode animations or step-by-step traces. While these are effective for understanding call stacks conceptually, they do not always provide a visually engaging representation of disk movements. The proposed visualizer enhances learning by combining algorithm execution with real-time animation, enabling learners to directly observe how recursive calls translate into actual operations on stacks.

Chapter 3

Literature Review

The Tower of Hanoi problem has long been used in computer science education as a standard example to demonstrate recursion and algorithmic problem-solving. Classical algorithm textbooks describe the problem as a recursive procedure that breaks a complex task into simpler subproblems. Authors such as Thomas H. Cormen present recursive problem-solving techniques and discuss the computational complexity of such divide-and-conquer approaches in foundational algorithm literature. The Tower of Hanoi is often cited as an illustrative example for understanding recursive call structures and exponential time complexity.

Educational resources and programming tutorials available on platforms such as GeeksforGeeks provide step-by-step explanations of the Tower of Hanoi algorithm along with code implementations in languages such as C, C++, Java, and Python. These resources emphasize the recursive nature of the problem and explain how the number of moves grows as $2^n - 1$, where n is the number of disks. While such tutorials are effective for learning the logic of the algorithm, they primarily rely on text and static diagrams, which may limit intuitive understanding for visual learners.

Several studies and educational tools highlight the importance of algorithm visualization in enhancing student comprehension. Interactive visualizations and animated simulations have been shown to improve engagement and retention when learning abstract concepts such as recursion and data structure operations. Various web-based implementations of the Tower of Hanoi usingHTML and JavaScript provide simple animations of disk movements. However, these tools often abstract away the underlying data structure implementation, focusing mainly on the final visual effect.

Research and learning materials on stack data structures explain how stacks operate using the LIFO principle and how they can be used to model constrained storage systems. Standard data structure textbooks discuss how recursive algorithms implicitly use the system call stack, and how explicit stack implementations can be used to simulate recursive behavior. This conceptual relationship between recursion and stacks makes the Tower of Hanoi problem particularly suitable for demonstrating both concepts together.

In terms of graphics programming for educational visualization, lightweight graphics libraries such as **Raylib** have been increasingly adopted for teaching purposes due to their simplicity and ease of integration with C and C++. These libraries allow developers to create interactive visual applications without the overhead of complex graphics engines. The proposed project leverages such a library to combine algorithm execution with real-time animation, providing an integrated learning

tool that connects theoretical concepts with practical visualization.

Overall, the existing literature and tools emphasize the importance of recursion, stacks, and visualization in computer science education. However, there is limited availability of beginner-friendly C++-based graphical implementations that integrate custom data structures, recursive algorithm execution, and real-time visualization within a single learning application. This project aims to bridge that gap by offering a unified educational tool for understanding the Tower of Hanoi problem through interactive animation.

Chapter 4

Design and Implementation

4.1 System Architecture

The Tower of Hanoi Visualizer is designed as a modular C++ application that combines and uses algorithm logic, data structures, and graphical rendering into a single system. The overall architecture is divided into the following main components:

4.1.1 Recursive Algorithm Module

- Implements recursion to solve Tower of Hanoi Problem.
- Pre-computes all valid disk movements for solveing problem and stores them in a vector of move operations.
- Ensures that the constraints of the problem are always satisfied.

4.1.2 Stack Data Structure Module

- Each peg (source, auxiliary, destination) is represented using a stack.
- The stack follows LIFO principle.
- Push and pop operations are used to simulate placing and removing disks from pegs.

4.1.3 Visualization & Rendering Module

- Uses the Raylib graphics library for rendering and visualization.
- Draws pegs, disks, and animations of disk movements.
- Updates the screen in a loop to display each step of the solution visually which flow can be reversed.

4.1.4 Control & Animation Module

- Controls the animation flow by allowing the user to manually navigate between moves using keyboard input (Enter for next move and Backspace for previous move).
- Reads pre-computed moves from the vector and applies them one by one to the stack structures.

4.2 Algorithm Design

The recursive algorithm for the Tower of Hanoi follows the standard divide-and-conquer approach:

1. Move $n - 1$ disks from the source peg to the auxiliary peg.
2. Move the largest disk from the source peg to the destination peg.
3. Move $n - 1$ disks from the auxiliary peg to the destination peg.

This recursive process generates a sequence of moves stored as pairs $(fromPeg, toPeg)$.

Instead of directly moving disks during recursion, the algorithm stores the moves first, which allows:

- Smooth animation
- Controlled execution speed
- Step-by-step visualization

4.3 Data Structure Design

Each peg is implemented using a stack class:

- **Push Operation:** Places a disk on top of a peg.
- **Pop Operation:** Removes the top disk from a peg.
- **Top Operation:** Returns the disk currently on top of the peg.

This design ensures that the rules of the Tower of Hanoi are always enforced:

4.4 Visualization and Animation Design

The graphical interface displays:

- Three vertical pegs
- Multiple horizontal disks of decreasing width
- Animated disk movement from one peg to another

Using the Raylib graphics library, the application:

- Animates disk transitions between pegs.
- Updates disk positions based on the current stack state.
- let us control the animation in steps from keyboard.

4.5 Workflow Algorithm

1. Start
2. Initialize three stacks (source, auxiliary, destination).
3. Push all disks onto the source stack in decreasing order.
4. Run the recursive Hanoi algorithm to generate all moves.
5. Store each move in a vector.
6. Start the graphics loop.
7. For each frame:
   (a) Read the next move.
   (b) Pop the disk from the source stack.
   (c) Push the disk onto the destination stack.
   (d) Update visual positions.
8. Continue until all moves are completed.
9. Stop

Chapter 5

System Overview

5.1 System Requirement Specifications

5.1.1 Software Requirements

This section presents the required software components for developing and running the ”Tower of Hanoi Visualizer”

Software Requirements

- **Operating System (OS):** Windows 10 / macOS
- **Programming Language:** c++
- **Libraries / Packages:** Raylib
- **Integrated Development Environment (IDE) / Code Editor:** VS Code

5.1.2 Hardware Specifications

This section describes the recommended hardware required for the system.

Hardware Requirements

- **Processor:** Dual-core Central Processing Unit (CPU)
- **RAM:** 2 GB
- **Storage:** 500 mb free space
- **Graphics Card:** Not required

5.2 Functional Requirements

The functional requirements describe what the system should be able to do. For the ”Tower of Hanoi Visualizer”, the key functional requirements are:

1. **Disk Initialization:** The system shall initialize a given number of disks on the source peg in correct size order.
2. **Tower Representation:** The system shall represent each peg using a custom stack data structure and support push and pop operations.
3. **Move Generation Using Recursion:** The system shall generate all valid disk movements using the recursive Tower of Hanoi algorithm and store them for execution.
4. **Step-by-Step Execution:** The system shall execute disk movements one step at a time while enforcing all Tower of Hanoi rules.
5. **Graphical Visualization:** The system shall visually display and animate the movement of disks between pegs using the Raylib library.

Chapter 6

Results and Discussion

6.1 Results

The Tower of Hanoi Visualizer was successfully implemented in C++ using the Raylib graphics library. The program correctly visualizes the Tower of Hanoi puzzle by animating disk movements between the three pegs (source, auxiliary, and destination) while strictly following the rules of the problem.

The recursive algorithm accurately generates the complete sequence of moves required to solve the puzzle for a given number of disks. These moves are stored and executed sequentially during visualization. The custom stack implementation correctly manages disk placement and removal, ensuring that at no point is a larger disk placed on a smaller one.

The user-controlled navigation feature allows the visualization to proceed step by step:

- Pressing **Enter** moves to the next disk movement.
- Pressing **Backspace** returns to the previous move.

This interaction model enables users to carefully observe each step of the recursive solution, improving understanding of how disks move between pegs.

The system was tested with different numbers of disks. In all test cases, the visualization produced the correct final configuration, where all disks were successfully transferred to the destination peg in the minimum number of moves $(2^n - 1)$.

6.2 Discussion

The results demonstrate that combining recursion, stack data structures, and real-time visualization provides a strong educational tool for understanding algorithmic problem-solving. The visual representation makes it easier to follow the flow of the recursive algorithm compared to traditional text-based outputs.

The decision to precompute all moves before starting the animation proved effective, as it allowed smooth navigation between steps using keyboard controls. The

ability to move backward using the **Backspace** key is particularly useful for learning, as users can revisit previous states and analyze how a certain configuration was reached.

Using the Raylib library enabled efficient rendering and responsive keyboard interaction, making the application lightweight and suitable for educational environments. The modular design, which separates algorithm logic, stack operations, and visualization, improved code readability and maintainability.

However, the current implementation is best suited for a small to moderate number of disks-up to 9 disks. As the number of disks increases, the number of moves grows exponentially, which can impact memory usage and usability. This limitation highlights the inherent time and space complexity of the Tower of Hanoi problem and serves as an additional learning point for understanding algorithmic efficiency.

Overall, the project successfully achieved its objectives of visualizing recursion and stack operations in an intuitive manner. The results confirm that algorithm visualization can significantly enhance conceptual understanding and learning engagement for students studying data structures and algorithms.

Chapter 7

Conclusion and Future Works

7.1 Conclusion

This mini-project did a job of creating a Tower of Hanoi Visualizer using C++ and the Raylib graphics library. The Tower of Hanoi Visualizer project shows the Tower of Hanoi problem in a fun way with animations and it follows the rules of the puzzle.

The Tower of Hanoi Visualizer project uses a stack data structure and a recursive move-generation algorithm. This helps connect the ideas of the Tower of Hanoi problem to how it works. Users can go through each step of the solution using the keyboard, which helps them understand the Tower of Hanoi problem

The system is designed in a way with separate parts for the algorithm, data and visuals. This makes the code easy to read and work with. Overall the Tower of Hanoi Visualizer project does what it is supposed to do: it helps students learn about recursion, stack operations and solving problems in a way.

7.2 Future Works

The current Tower of Hanoi Visualizer project is a start.. There are some things that can be added to make it better and more fun to use. Here are some ideas:

- **Playback Controls:** Add buttons to play, pause and reset the animation in addition to going through it step by step.
- **Adjustable Animation Speed:** Let users control how fast the animation plays so they can see it in motion or faster.
- **Performance Optimization:** Make the program use memory so it can handle more disks.
- **Enhanced User Interface:** Make the visuals better with labels, counters and highlights, for the pegs.

References

[for Geeks] for Geeks, G. Tower of hanoi algorithm. https://www.geeksforgeeks.org/dsa/c-program-for-tower-of-hanoi/. Accessed: 17 feb, 2026.

[Raylib] Raylib. Raylib-cheatsheet. https://www.raylib.com/cheatsheet/cheatsheet.html. Accessed: 18 feb, 2025.

[3] Reducible (2020). Towers of hanoi: A complete recursive visualization. https://www.youtube.com/watch?v=rf6uf3jNjbo&t=1003s. Accessed: 17 feb, 2026.

Appendix A

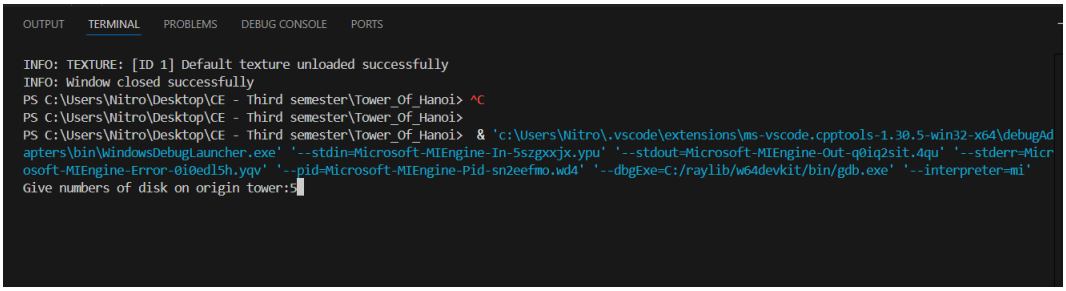

Figure 7.1: Taking Number of disks in primary peg from user

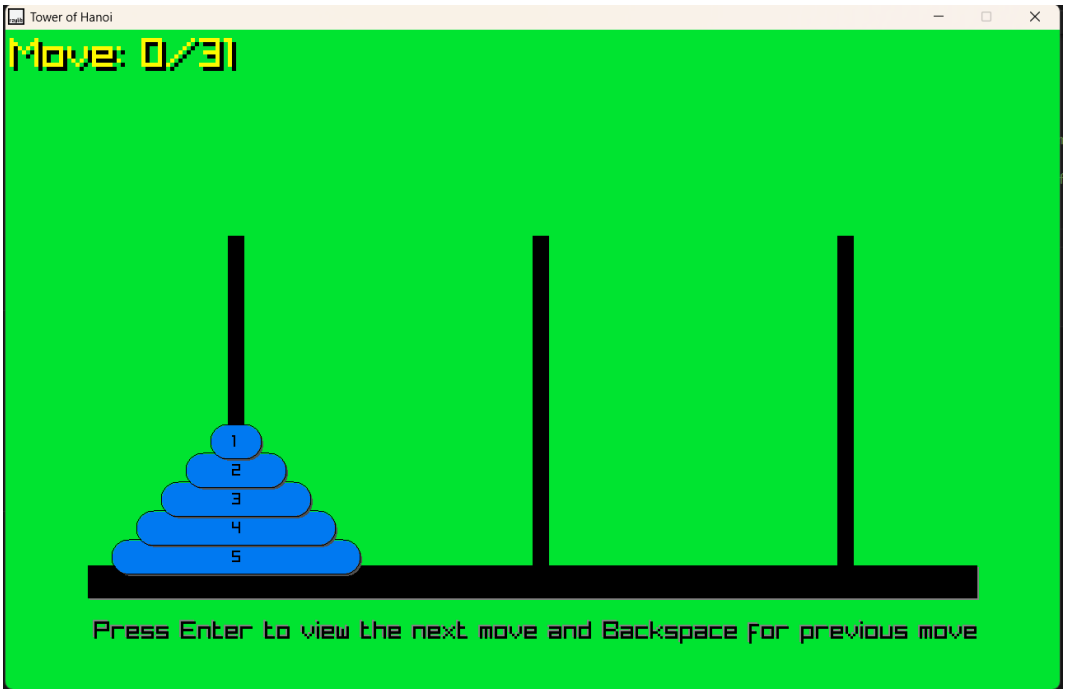

Figure 7.2: step: zero

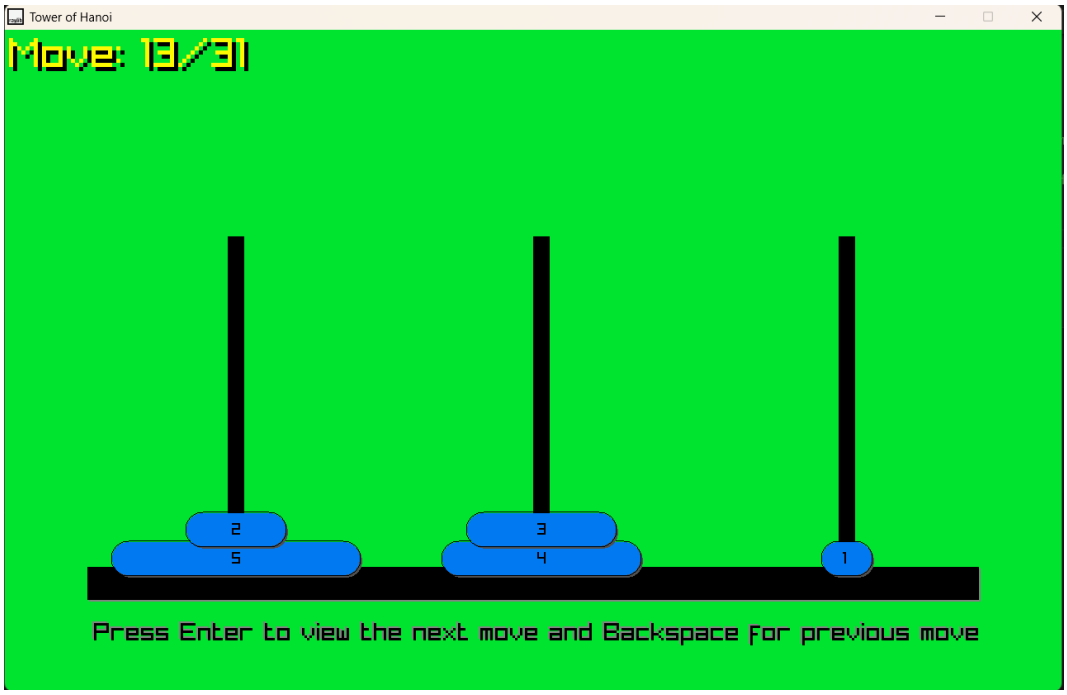

Figure 7.3: step: Thirteen

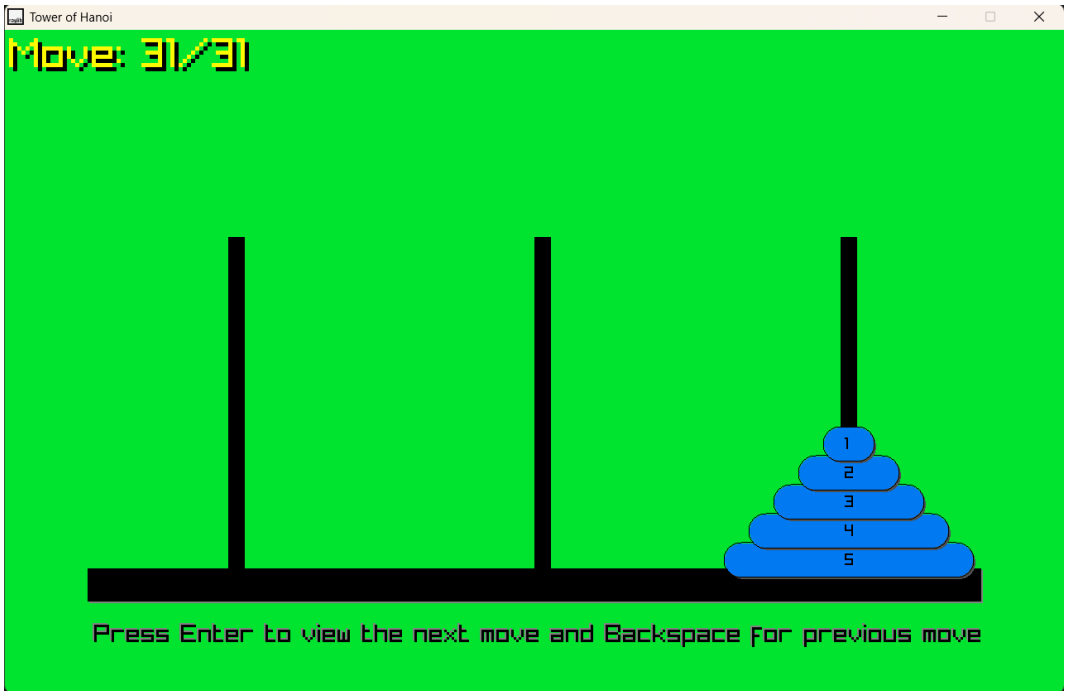

Figure 7.4: Puzzle Completed