Lab 4: Tower of Hanoi

In Lab 4, you are required to design and implement a game: Tower of Hanoi.

0 Story

The puzzle was invented by the French mathematician Édouard Lucas in 1883. There is a story about an Indian temple in Kashi Vishwanath which contains a large room with three time-worn posts in it, surrounded by 64 golden disks. Brahmin priests, acting out the command of an ancient prophecy, have been moving these disks in accordance with the immutable rules of Brahma since that time. The puzzle is therefore also known as the Tower of Brahma puzzle. According to the legend, when the last move of the puzzle is completed, the world will end.

Objective and Rules

The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

1. Only one disk can be moved at a time.
2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty rod.
3. No larger disk may be placed on top of a smaller disk.

1 Lab Requirements

You should implement the Hanoi game program with the following requirements:

1. Language: C++. Containers in STL(`std::vector, std::stack, std::list, std::queue`) are **NOT** allowed to use. Please implement your own `Stack` or `Queue` if needed.
2. Concepts:
   - Canvas: The UI of the game.
   - Rod: There are 3 rods in the canvas. Disks are pushed and popped among them.
   - Disk: Each disk is of different sizes. In the beginning, they are all pushed into Rod 1.
3. **Intial Phase:** When the program starts, it firstly asks player for the number of disks, the input should between [1, 5]. Invalid input should be ignored.

4. **Normal Mode:** Then, print the canvas in the console and the game starts. The program continuously asks for commands until the player wins. The canvas will be printed after each move. Command format: `from to`. Invalid input should be ignored.
5. **Auto Play Mode:** If the game receives command `0 0`. Auto-play mode will be activated. The game will automatically continues without player's commands. The canvas, as well as the auto-generated command, will be printed after each move.

2 Guidance

2.1 Design

You have known requirements of the lab and rules of the game, but maybe you do not know how to start. Hope these materials will help you.

- 怎样进行面向对象的程序设计？ - 翅膀的回答 - 知乎
- 什么是面向对象编程思想？ - 小耿的回答 - 知乎
- 什么是面向对象编程思想？ - 雨露天择的回答 - 知乎

If you are still at a loss, feel free to ask teaching assistants, we are glad to help.

2.2 Drawing the Canvas

We have offered you a file, `termio.h`. It might be helpful for you to deal with the canvas printing work.

- `CANVAS_WIDTH` and `CANVAS_HEIGHT`: the width and height of the canvas.
- `MAX_WIDTH` and `MAX_HEIGHT`: the width and height we need. `CANVAS_WIDTH` and `CANVAS_HEIGHT` defines how large a canvas is. `MAX_WIDTH` and `MAX_HEIGHT` defines how large you really use. Sometimes people don't eat 5kg rice in a meal just because there are 5kg rice left in the house.
- `Clear()` helps you clear the screen. Using it before `Draw()` will improve the user experience.
- `Draw()` helps you draw the buffer to console.
- `ResetBuffer()` fills the buffer with spaces.

2.3 Auto Play Mode

Basic Hanoi

Basic Hanoi problem: Assume all the disks are pushed in Rod A, generate a sequence of commands that could move all the disks to Rod C.

Solving Basic Hanoi problem leverages recursion. The algorithm could be described as following pseudo code:

```
function hanoi(n, A, B, C) {
     hanoi(n - 1, A, C, B);
     move(n, A, C);
     haoni(n - 1, B, A, C);
}
```

Note:

- Rod: A, B, C
- Number of disks: n
- Target: Move n disks from A to C
- function Hanoi(n, A, B, C): move n disks from A to C, use B as buffer
- function move(k, A, B): move No.k disk from A to B

Hanoi's Movement in Any Legal State

In our lab, players may enter auto play mode in any state. So how to generate the sequence of commands?

Simple idea: You can use a stack to log player's steps. When the player enters auto play mode, you could pop the commands from the stack, by this 'undo' operation, the state will roll back to the beginning. Then you just need to solve the basic Hanoi problem.

If you have any questions, feel free to ask teaching assistants, we are glad to help.

3 Upload

Remember to upload your code. You should also upload your code to the FTP as before.

4 Example

What does the game look like? Here is an example.

1 Initial Phase

```
./hanoi
How many disks do you want? (1 ~ 5)
2
```

After telling the program about the number of disks, we come to the normal mode.

2 Normal Mode

```
     |              |              |
     |              |              |
 *********          |              |
     |              |              |
***********         |              |
     |              |              |
-----|--------------|--------------|-----

Move a disk. Format: x y
1 3
```

We give a leagal command `1 3`. The canvas will be refreshed.

```
     |              |              |
     |              |              |
     |              |              |
     |              |              |
***********         |          *********
     |              |              |
-----|--------------|--------------|-----

Move a disk. Format: x y
1 3
```

As we can see, the top disk of Rod 1 is popped and pushed to Rod 3.

Then we give an ilegal command `1 3`. It will be ignored, since it obeys the rule: No larger disk may be placed on top of a smaller disk.

```
     |              |              |
     |              |              |
     |              |              |
     |              |              |
***********         |          *********
     |              |              |
-----|--------------|--------------|-----

Move a disk. Format: x y
0 0
Auto moving:3->1
     |              |              |
     |              |              |
 *********          |              |
     |              |              |
***********         |              |
     |              |              |
-----|--------------|--------------|-----

Auto moving:1->3
     |              |              |
     |              |              |
     |              |              |
     |              |              |
***********         |          *********
     |              |              |
-----|--------------|--------------|-----

Auto moving:1->2
     |              |              |
     |              |              |
     |              |              |
     |              |              |
     |         ***********     *********
     |              |              |
-----|--------------|--------------|-----

Auto moving:3->2
     |              |              |
     |              |              |
     |          *********          |
     |              |              |
     |         ***********         |
     |              |              |
-----|--------------|--------------|-----

Auto moving:1->3
     |              |              |
     |              |              |
     |          *********          |
     |              |              |
     |         ***********         |
     |              |              |
-----|--------------|--------------|-----
```

```
Auto moving:2->1
     |              |              |
     |              |              |
     |              |              |
     |              |              |
 *********     ***********         |
     |              |              |
-----|--------------|--------------|-----

Auto moving:2->3
     |              |              |
     |              |              |
     |              |              |
     |              |              |
 *********          |         ***********
     |              |              |
-----|--------------|--------------|-----

Auto moving:1->3
     |              |              |
     |              |              |
     |              |          *********
     |              |              |
     |              |         ***********
     |              |              |
-----|--------------|--------------|-----
```

Then we tried the auto play mode by giving the command `0 0`. As we can see, the program executes automatically until the player wins.