1 A tower defense game based on Qt library

----a kingdom rush-like tower defense game

1.1 General game description

In the game, the enemy army will be generated at the starting point of the enemy's marching path at a specific time and march towards the end point. The marching path is a specific enemy walking route for different maps. There can be many paths, and the enemies will be selected in turn according to the order in which they are generated. After the enemy reaches the end, the player's life value will -1, which means that the base camp is invaded by the enemy. When the player's life value returns to 0, the level challenge will fail. If the player kills all the enemies, the level will succeed. Players can build defensive towers to prevent enemies from reaching the end. There are two types of defense towers: melee towers built on the enemy's marching path and ranged towers built beside the enemy's marching path. Melee towers generally have the function of blocking the enemy and attacking the enemy to a certain extent. The ranged tower has a larger attack range, but its own defense is weak and is easily destroyed by the enemy's cunning siege equipment. There are also many types of enemies, including ordinary robbers, scouts who ignore the blocking effect of melee towers, pilots who can cause damage to melee towers but can't be hit by them, and archers who can attack melee towers from a distance, the catapult that can attack all defense towers from a long distance, the shaman who can heal the friendly forces, and the generals who buff the friendly forces...Facing different enemies, player should adopt different strategies and build different towers, and finally smash their terrible attacks.

1.2 Brief description of operation

After entering the game, there are many tower bases on the map. Click on the tower base to see the towers that can be built, and the number below it is the cost. After building the tower, in order to upgrade the it, click it again (after upgrading, only the newly increased HP of the upgraded building will be obtained). For ranged towers, its attack range will also be displayed after clicking. If it is already the highest level, the upgrade interface will not appear. Not having enough money or clicking elsewhere cancels the upgrade.

The defense tower will automatically attack the enemy, so the player's operation is only to build and upgrade. Bounties can be obtained after killing enemies to further strengthen the defense.

1.3 For game modders

The game has good scalability. Modders can design any mission they want in the basic frame of the game and I gives a manual for creating a new mission. Modders can use the .txt file to customize the level according to a certain given syntax described in the example.

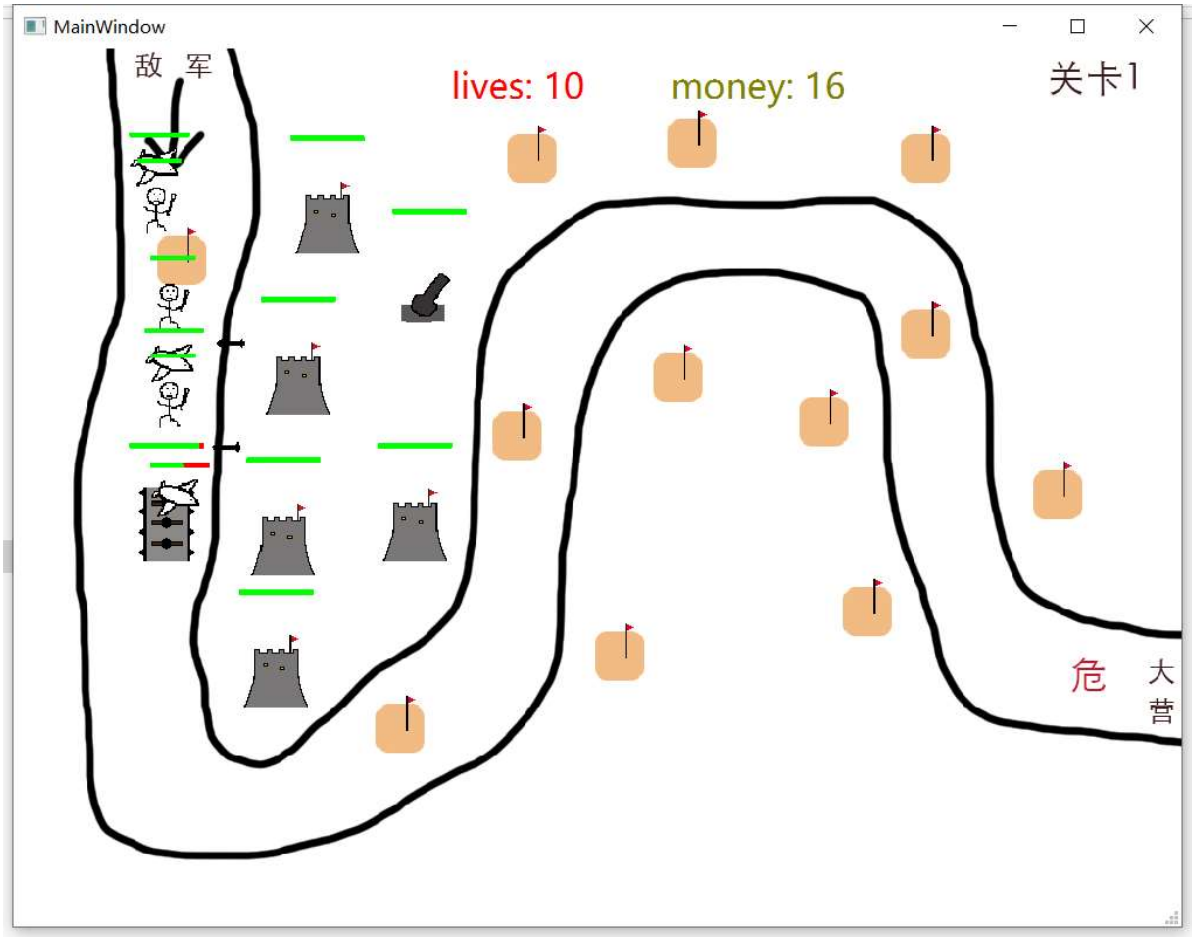
1.4 Personal feelings

In the beginning, two ready-made games were given for reference in the experimental documentation, but I have never played Tomorrow's Ark, so basically the design idea of the entire game is derived from the game kingdom rush that I especially liked to play when I was young. Seeing that I can make a game more and more playable, I have a sense of achievement.

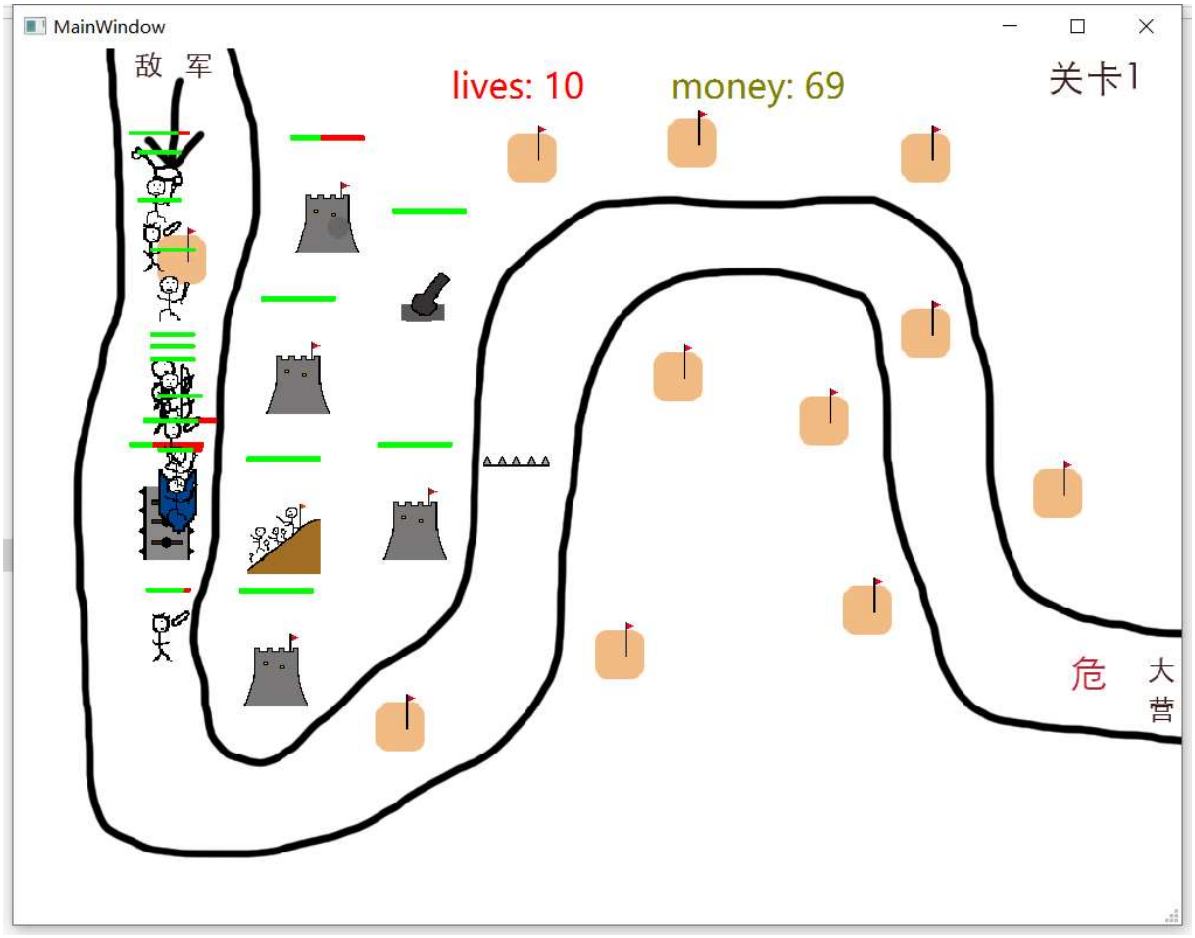
This is the first time to independently complete a large project from scratch ,and the whole project is all done by myself from designing to coding, but the progress is not as slow as expected. Most of the time, I have passion to complete this project.

I use a self drawn matchstick men style art to present the enemy and the tower because it is abstract but easy to understand while it really saves time and makes the game quite funny.

The experience is still insufficient for the first time, and various pointers are used indiscriminately. In addition, the code is not very standardized, some functions that don't change the members of the class do not write 'const' to them. These problems appeared many times during the initial programming process, and it has been gradually corrected in the latter.



playing the tower defense game



some other towers

win or lose game

2 A C-- Compiler

C-- is a C-style programming language created by the lecture tutors. My job is to use C to realize a C-- compiler and run some programs based on it. It is a quite tough project and the teacher gives few hints.

This project is divided into four parts, the first two parts realize context-free grammar and syntax directed translation, and the latter two parts realize intermediate code generation and assembly code generation.

The most difficult part is the generation of intermediate code. First, the amount of code is large and error-prone. Second, the generated intermediate code is complex and huge, which needs to be optimized to improve the running speed, and the optimization method such as common subexpression elimination, copy propagation and dead code elimination are difficult to achieve. Finally, there are many small problems that will cause There are some bugs in the program, which are not even resolved in the end. The second most difficult part is code generation. This part requires complex operations such as rewriting the intermediate code into assembly language, manipulating registers, and writing assembly code for function calls.

```
5
6   .text
7   read:
8   li $v0, 4
9   la $a0, _prompt
10  syscall
11  li $v0, 5
12  syscall
13  jr $ra
14
15  write:
16  li $v0, 1
17  syscall
18  li $v0, 4
19  la $a0, _ret
20  syscall
21  move $v0, $0
22  jr $ra
23
```

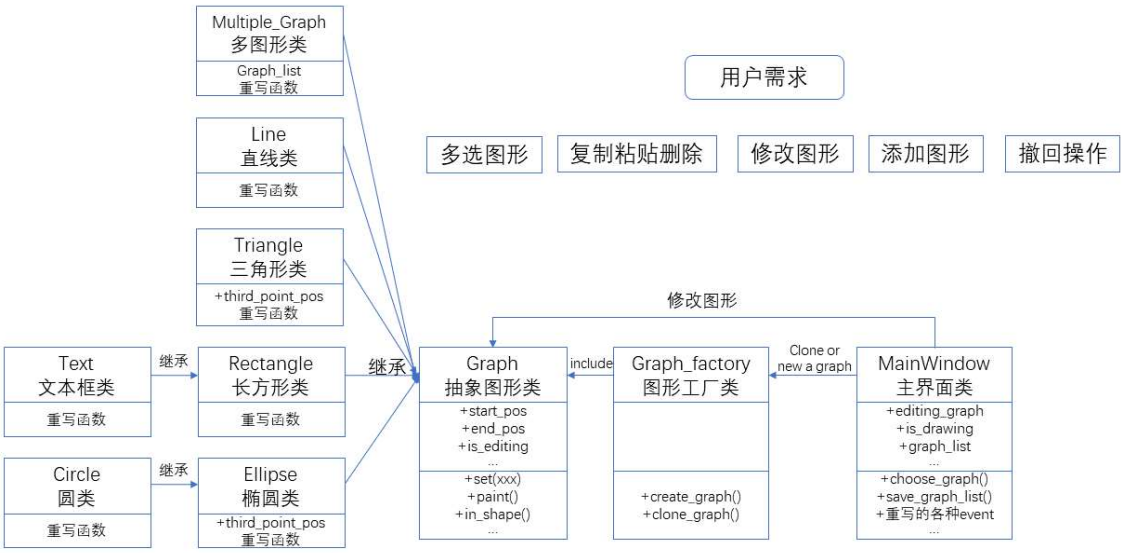
part of the code parsed by the compiler

result of a simple program parsed by the compiler

3 A 2D shape drawing tool

3.1 general description

As the final project of Object Oriented Design Method lecture, this project uses a lot of object-oriented programming ideas such as factory pattern, combination pattern, memo pattern, singleton pattern...



For the needs of multiple kind of shapes, a virtual base class (abstract class) 'Graphic' class is used as the parent class, and various graphic classes rewrite each function in the parent class. For the implementation of MainWindow, in order to create objects more conveniently and concisely, the simple factory pattern is used to generate graphic objects in a factory class.

In the graphics sequence stored in MainWindow, there are single graphics and composite graphics composed of multiple graphics. Using the combination pattern, programmers don't need to consider other graphics when writing the same graphics in MainWindow (such as copying, pasting, and deleting). Whether it is a combination of graphics or a single graphics, can be done with a unified interface.

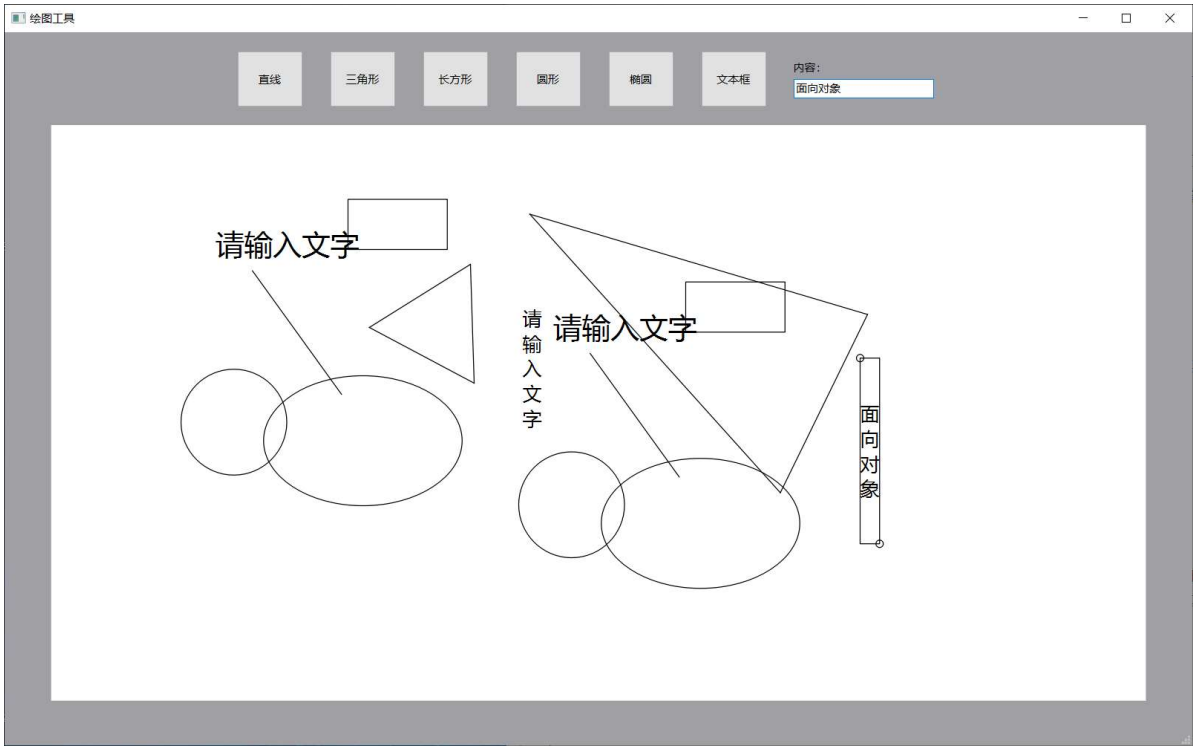
In order to realize the undo function, the original state is saved before each operation on the graph. Using the memo pattern, it can be easily restored to the previous state. The disadvantage is that it takes up a lot of memory space.

A graphic composed of multiple graphics will only appear when the graphic is selected and edited, and there can only be one graphic being edited at the same time, so using the singleton mode can prevent the object composed of multiple graphics from being created multiple times, which violates the author's intent.

3.2 functions realized

The project realizes five shapes of straight lines, triangles, squares, circles, and ellipses. Users can click the corresponding option on the menu bar, and then pull the graphics on the canvas.

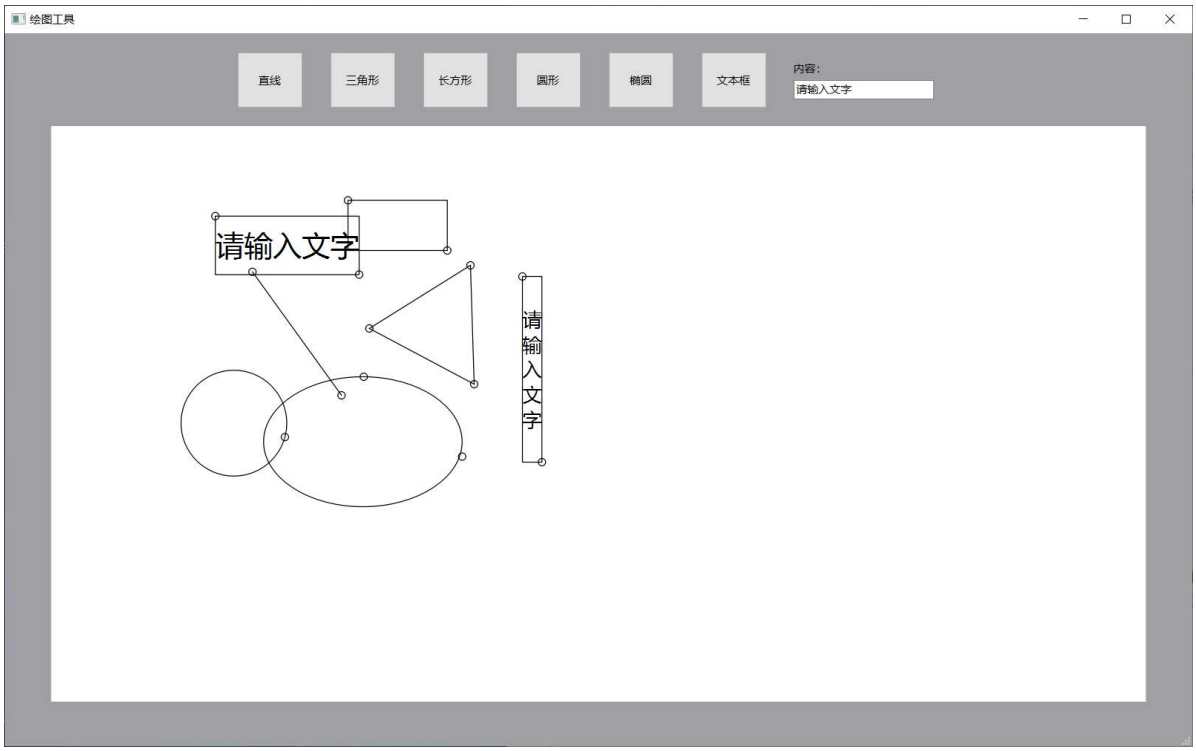
The project also realizes the text box. After the text box drawing is completed, click the text box to modify its content on the upper toolbar. The font size of the text automatically adapts to the size of the text box.



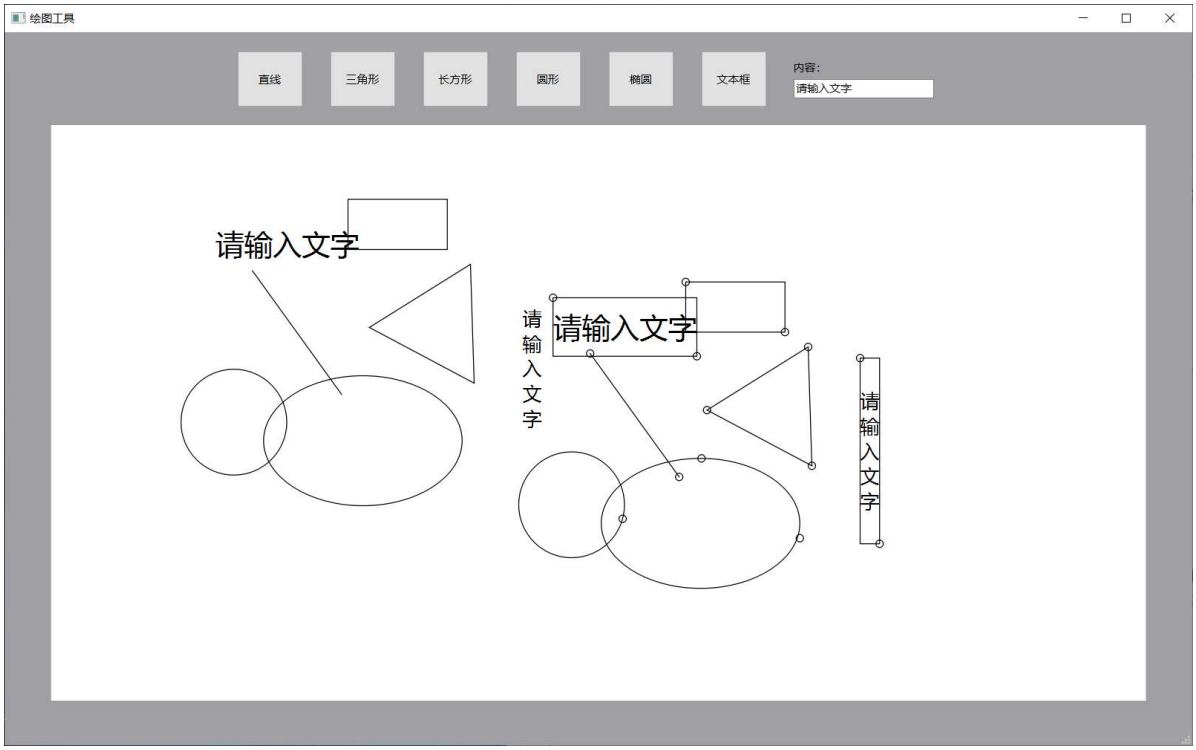
text modification

Completed the processing of clicking on a single graphic. If a blank area is clicked, the previous click effect will be removed. After clicking, some circles will appear on the selected graphics. At this time, the graphics can be copied, pasted, cut, deleted, and moved. The circles on the graphics are the base points of the graphics, which determine the shape of the graphics.

If you click on a graphic and then press ctrl to click another graphic (note that clicking on a graphic will only select the topmost graphic), multiple selections will be made, and multiple selections also support copy, paste, cut, delete, and move.



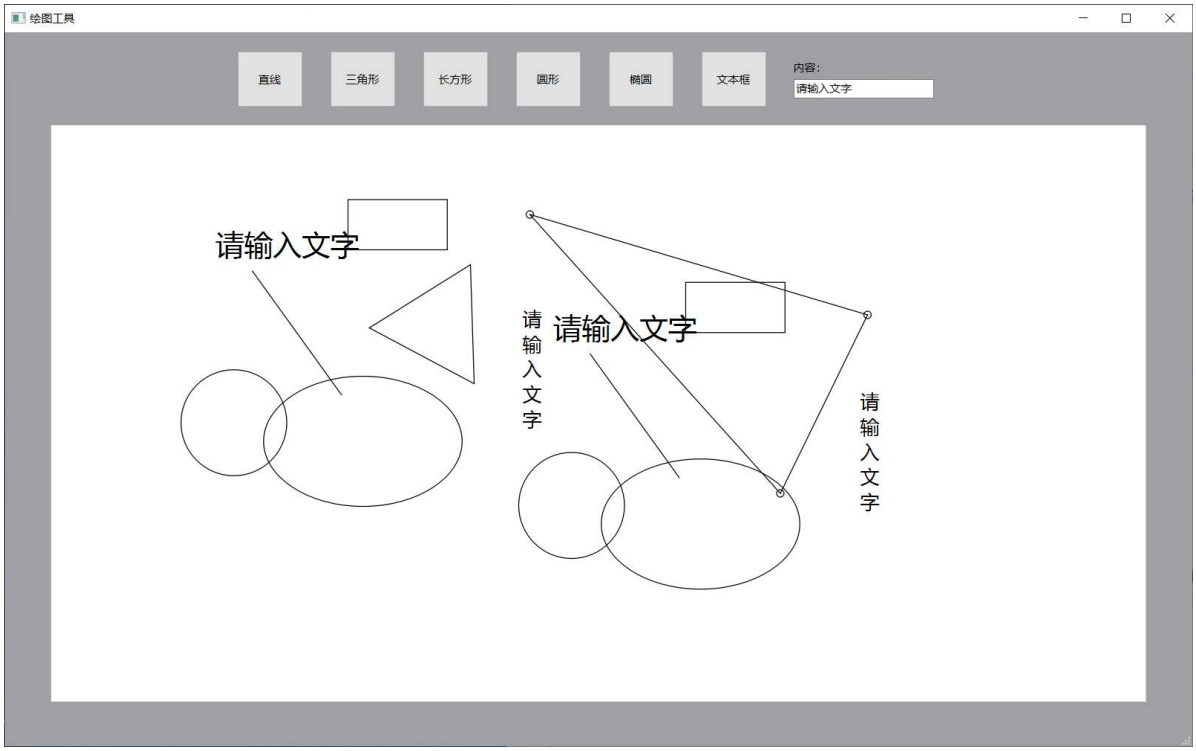
Graphical multiple selection



Copy and paste of multiple selection graphics

Press ctrl+z to achieve multi-step undo.

Support dragging the reference point of a single graph (that is, the hollow dot that appears after clicking the graph) to change the size and shape of the graph, and dragging other areas of the graph will move the graph



deformation of graphics

Support ctrl+u to set the graphics to the top; ctrl+l to set the graphics to the bottom.

4. Turing machine with a program to find greatest common divisor of two numbers

4.1 description

The experiment needs to complete a Turing machine that accepts input of a certain grammar to process a certain input tape. Therefore, the entire Turing machine is divided into two parts: the part that processes the input of the Turing machine program and the part that processes the input tape.

For the part that processes the input of the Turing machine program, a Turing machine class is defined to receive elements of the Turing program. Use the library in C++ to handle file input. Use string or char vector to store the contents of the 'collection' (duplication is allowed, if there is any duplication, ignore it), use map to store the transfer function 'delta', where the key is the current state, the value pointed to by the current tape head, and the value is written value, the moving direction of the tape head and the next state.

For the part of processing the input tape, first check the legality of the tape, then define the current state as the initial state, and then use the while loop to execute continuously until:

Execution reaches a final state, the program ends, and the result of the first tape is output

There is no corresponding transfer function in the current state, and an error is reported

Infinite loop (caused by Turing machine programs or input defects) program settings, if the program has executed 1,000,000 steps and still has no results, it will be forced to jump out

Realize a user-friendly verbose mode which indicates every step of the Truing machine when it operates and give error message when it meets error.

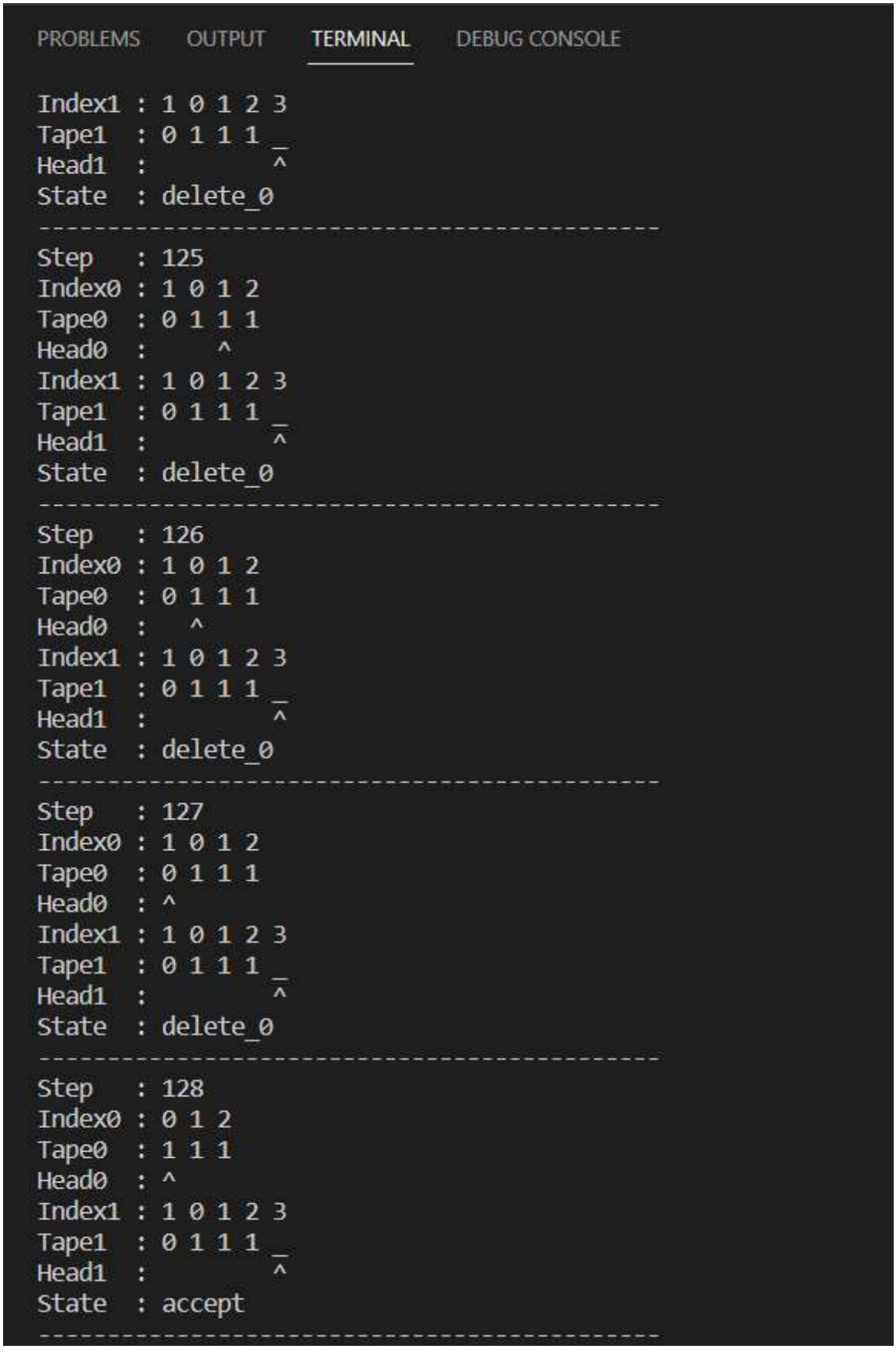
4.2 the thought of greatest common divisor

As requested, the input must be a positive integer and cannot be 0. You must enter strings like regular expression: '1+01+'. This represents two numbers divided by a '0', and the number of '1' means the size of a number.

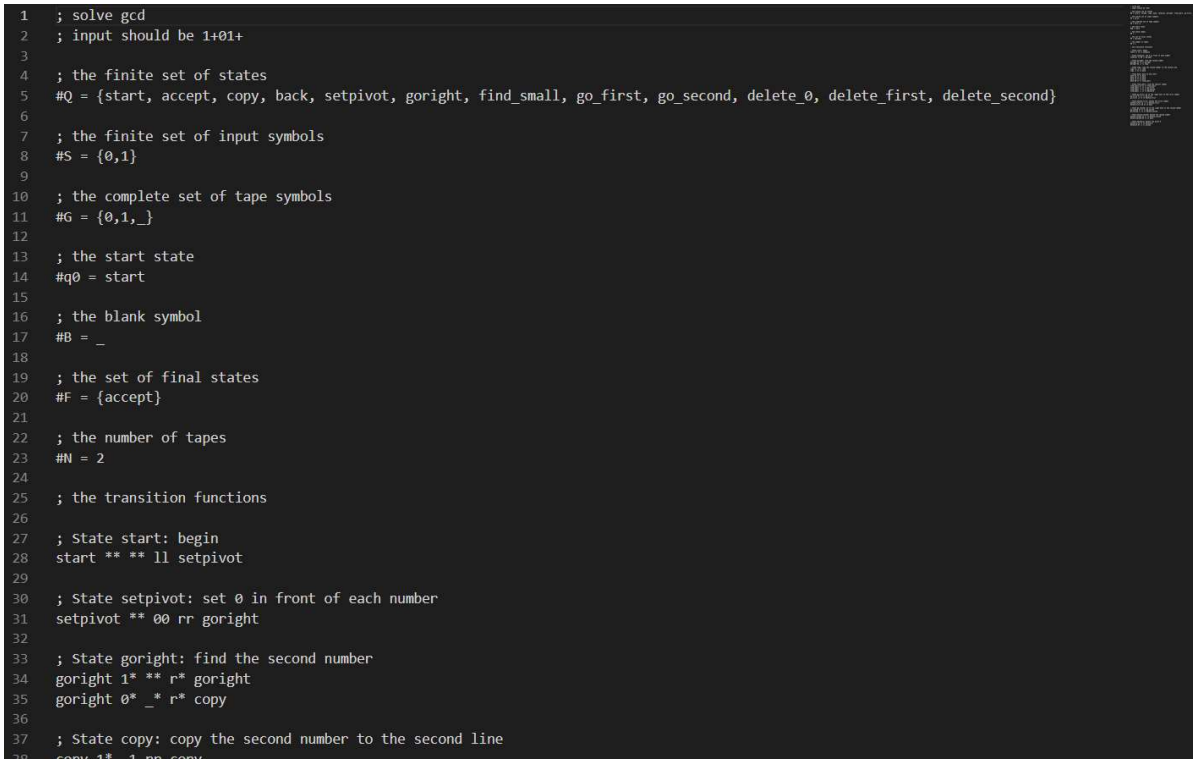
First set a 0 flag before the two numbers, indicating the beginning of the two numbers (convenient to return to the beginning of the two numbers), then look for the 0 of the input tape, and change the 0 to _, and move the 1 after 0 to the second Paper tape, then both tape heads return to the starting point of the two numbers, and then cycle: find the smaller (short) of the two numbers (if the same length, the description is over, delete the initial 0 of the first paper tape, and get the result), then the head of the large number tape goes to the tail of the large number, then the large number subtracts the small number, then both sides return to the starting point, and repeat.

```
PS E:\沈炫辰项目资料\Formal Languages and Automata\turing machine\turing_project> ./turing.exe gcd.tm 1111111111101111111
1
```

calculating the gcd of 13 and 7 using instructions



calculating the gcd of 15 and 9 by Turing program in verbose mode



```
; solve gcd
; input should be 1+01+

; the finite set of states
#Q = {start, accept, copy, back, setpivot, goright, find_small, go_first, go_second, delete_0, delete_first, delete_second}

; the finite set of input symbols
#S = {0,1}

; the complete set of tape symbols
#G = {0,1,_}

; the start state
#q0 = start

; the blank symbol
#B = _

; the set of final states
#F = {accept}

; the number of tapes
#N = 2

; the transition functions

; State start: begin
start ** ** ll setpivot

; State setpivot: set 0 in front of each number
setpivot ** 00 rr goright

; State goright: find the second number
goright 1* ** r* goright
goright 0* _* r* copy

; State copy: copy the second number to the second line
```

part of the Turing program