Subsystems.

In total, there are 4 subsystems in my program. They are:

1. A subsystem that interact with user (e.g. get input, show result on console)

2. A subsystem that maintains the information related to chess board (e.g. piece location).

3. A subsystem to maintain the sequence of a gameplay. (e.g. dice, next_player())

4. A subsystem that have certain algorithm so it can play with player (i.e. A.I.).

Key frame

There are two repeating keyframes in my program: 1. get instruction code. 2. draw chess board.

```
//For any get instruction phase, it has an uniformed format,
//so it is a repeating key frame
//example:
------------------------------------------------------------
|          |  Command  |  0: Exit  | 1: New Game           |
------------------------------------------------------------
                     Input code: 1

// For each game step, system will ask a user to choose a place
//and then print the chess board. So it is another repeating frame.

      Player 1 Which cell you want to place: 4
============================================================
|                                                          |
|                       =============                      |
|                       |   |   |   |                      |
|                       | X |   |   |                      |
|                       | 1 | 2 | 3 |                      |
|                       |---+---+---|                      |
|                       |   |   |   |                      |
|                       | X | O |   |                      |
|                       | 4 | 5 | 6 |                      |
|                       |---+---+---|                      |
|                       |   |   |   |                      |
|                       |   |   |   |                      |
|                       | 7 | 8 | 9 |                      |
|                       =============                      |
============================================================
------------------------------------------------------------
|          This is step 3, there are 6 more steps          |
------------------------------------------------------------
```

A.I information

I do think my computer player play as good as me (hard mode A.I.).

It has been proved that if each player always choose optimal place, this TTT game will always draw. So long as my A.I optimize its game strategy, it can play as good as a human player.

For my program, I used DFS(deep-first-search) algorithm. The A.I. will recursively simulate(i.e. search) all possible locations on board, it is equipped with the ability to see that some place on board will lead to lose, some will be draw (it is implemented by setting a weight to each location). My A.I will alway choose the optimal place, so it will at lease not lose a game (If human player also optimize his strategy, then it will draw).

anothor two aspects

UI design.

To enhance Users' experience, I spent a lot of time on UI design. The whole program has an uniform output format, which is string information inside a message box. Apart from it, many operation has a feedback, that is to say, repeat users selection, for example

```
------------------------------------------------------------
|      |  Select Difficulty |  0: Easy  | 1: Hard          |
------------------------------------------------------------
                     Input code: 1
------------------------------------------------------------
|                 You selected hard mode.                  |
------------------------------------------------------------
```

Instead of silently enter hard mode, the system will notify user that hard mode is selected (feedback), this will enhance users' experience.

Lastly, I used different color to represent different information, one example is the game board.

```
-------------
|   |   |   |
| X | O | X |
| 1 | 2 | 3 |
|---+---+---|
|   |   |   |
|   | O |   |
| 4 | 5 | 6 |
|---+---+---|
|   |   |   |
| O | X |   |
| 7 | 8 | 9 |
-------------
```

As you can see, not only the piece shape is different, their color is different too. This will help user better recognize their chess piece.

chess board implementation

In my program, The chess board is implemented using C-style array of 9 ints. Each location takes one place, if there is no piece, its value will be 0, if it is an X piece, its value will be 1, and if it is an O piece, its value will be 2.

This simple design – store pieces by an array of integers, greatly simplified the design of other subsystems and made the api easy to use.