OllQ 对战平台第三节

一、服务器棋盘的数据的初始化结构定义

```
var DSQ_qi = []int{ // 1-8 A ;9-16 B ; 17 未翻牌; 18 已翻牌
        Proto2.Elephant, Proto2.Lion, Proto2.Tiger, Proto2.Leopard,
Proto2.Wolf, Proto2.Dog, Proto2.Cat, Proto2.Mouse,
        Proto2.Mouse + Proto2.Elephant, Proto2.Mouse + Proto2.Lion,
Proto2.Mouse + Proto2.Tiger, Proto2.Mouse + Proto2.Leopard,
        Proto2.Mouse + Proto2.Wolf, Proto2.Mouse + Proto2.Dog,
Proto2.Mouse + Proto2.Cat, 2 * Proto2.Mouse}
```

二、棋盘的初始化

```
    // 初始化牌型
func InitDSQ(data1 []int) [4][4]int {

        data, erdata, j, k := data1, [4][4]int{}, 0, 0

        for i := 0; i < Proto2.Mouse*2; i++ {
                icount := util.RandInterval_LollipopGo(0, int32(len(data))-1)
                //fmt.Println("随机数: ", icount)
                if len(data) == 1 {
                        erdata[3][3] = data[0]
                } else {
                        //-------------------------------------------------------------
                        if int(icount) < len(data) {
                                erdata[j][k] = data[icount]
                                k++
                                if k%4 == 0 {
                                        j++
                                        k = 0
                                }
                                data = append(data[:icount], data[icount+1:]...)
                        } else {
                                erdata[j][k] = data[icount]
                                k++
                                if k%4 == 0 {
                                        j++
                                        k = 0
                                }
                                data = data[:icount-1]
```

```
                }
                //------------------------------------------------------------
            }
            //fmt.Println("生成的数据", erdata)
        }

        return erdata
}
```

三、下节介绍

第四节 前段棋盘的数据如何处理 1