

# Report for Project 1: Dash

---

## Demand Analysis

为所有用户提供尽量友好的操作界面

- 对功能列表菜单化，并设置多级菜单方便不同用户使用
- 充分考虑程序运行中可能出现的异常输入和特殊情况，并在其发生时提供有效的提示

为普通用户提供账户和游戏相关的操作

- 账户：注册、登录，申请重置密码
- 游戏：选择难度、持续时长等参数，查看练习记录
- 注意，在实现以上操作时，务必保证多用户间的隐私隔离以及用户的合理权限，比如用户有权利申请重置密码，但是否通过须由管理员决定

为管理员提供一系列管理操作

- 对练习文本的增删
- 对重置密码的响应

游戏的执行逻辑

- Dash的设计思路来源于经典的飞机大战、节奏大师、俄罗斯方块等游戏
- 游戏中，文本以单个字符的形式从控制台上方不断落下，玩家需要在字符掉落到分隔线之前完成对应按键的敲击，以获取得分
- 对用户的键入须作出即时相应，计算正确率
- 游戏分为若干个难度，难度的区别主要表现在字符下降的速度
- 若强制退出，须重新计算持续时长

整体信息的存取

- 在启动时，需要从文本读入所有用户的记录信息、练习文本的列表等
- 在退出时，需要向文本写入更新过的用户记录信息、练习文本列表等
- 需要一个合理的文件层次来存储这些信息

## Module Design

**FYI:** 具体设计细节在注释中给出

## User

User类负责普通用户的相关操作，包括：

- 玩家信息的初始化、修改
- 游戏记录的存储、查阅

```
1 //一次游戏的记录
2 struct record
3 {
4     int index;        //记录编号
5     difficulty diff;    //难度等级
6     int score;        //得分
7     float duration;    //持续时间
8     float accuracy;    //正确率
9 };
```

```
1 class User
2 {
3 private:
4     string id;        //唯一标识符
5     string nickname;    //昵称
6     string pwd;        //密码
7     vector<record> records; //练习记录
8 public:
9     //每个用户都用一个单独文件存储，文件名即为id
10    //所有id用"ids.txt"另行存储
11    User(string id);
12    User(string id, string nm, string pwd); //构造函数重载
13    void displayRecord(); //展示练习记录
14    void storeRecord(); //将记录保存到文件中
15    friend ostream& operator<<(ostream& os, record& r); //重载
    record输出
16    friend class Admin; //为了方便Admin调用User成员，将其声明为友类
17 };
```

## Admin

Admin类负责管理员的相关操作，包括：

- 程序整体的运行逻辑控制
- 用户信息的维护、修改、存取（权限很高）
- 练习文本的增删
- 在用户输入不合法时提供帮助信息

```
1 class Admin
2 {
3 private:
4     //保证Admin类仅有一个实例
5     static Admin* a;
6     map<string, User> users; //用户组成的字典
```

```

7     vector<string> src;           //打字文本集合
8     User* activeUser;           //当前活跃用户
9     vector<rank_pair> rankList; //排行榜
10
11     void storeIDs();             //保存所有id至"ids.txt"
12     void login();               //登录
13     void regist();              //注册
14     void resetPwd(string);       //重置密码
15     bool changePwd(string);      //更改密码
16     void play();                //加载游戏
17     void storeTxts();            //保存所有文本名至"src.txt"
18     void addTxt(string);         //增加文本
19     void rmTxt(string);          //移除文本
20     void userMenu();             //用户菜单
21     void adminMenu();           //管理员菜单
22     void rank(const User& self);  //展示排行榜
23 public:
24     static Admin* getInstance();
25     void mainMenu();            //主菜单
26 };

```

## Game

Game类负责游戏的控制逻辑，包括：

- 游戏的运行、停止
- 对玩家输入的即时响应
- 游戏记录的计算

```

1 //定义难度等级
2 enum difficulty
3 {
4     EASY, MEDIUM, HARD, HELL
5 };

```

```

1 class Game
2 {
3 private:
4     static Game* g; //保证Game类仅有一个实例
5     float duration; //游戏持续时间
6     float accuracy; //正确率
7     int score = 0; //游戏得分
8     difficulty diff; //难度等级
9     void checkInput(); //检查输入是否正确 多线程
10 public:
11     static Game* getInstance(string str, float du, difficulty
    di);
12     void mainGame(); //负责游戏的主体逻辑及画面的构建、刷新
13     friend class Admin; //为了方便Admin调用Game成员，将其声明为友类
14 };

```

## Novelty

### 排行榜

在用户菜单中可进入排行榜界面，查看分数排名前五位的练习记录（包括其他玩家），且为便于区分，玩家的个人记录被标红。

### 密码重置&修改

若用户忘记密码，可在登录界面向管理员申请重置密码（"12345"），也可以在用户菜单选择修改密码（须原密码进行验证）。

## Problems & Solutions

**P:** 最初Game模块采用单线程设计，一个大循环同时控制字符的不断下落和用户键入的判断。然而，区分这两者操作的依据是当前循环是否收到了键入信号，这就导致了在键入比较密集时（尤其在困难模式下），键入并不能得到即时反馈，有时还会因此造成程序的混乱，影响玩家游戏体验。

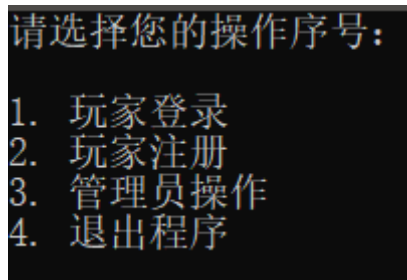
**S:** 基于此，我尝试将用户键入字符的正确性判断设计为一个新线程，从而取得了不错的效果。

**P:** 对于存储用户信息的文件结构，我最初没有清晰地构思好就开始写程序，试图将所有用户的信息放在同一个文件里进行存取。但事实证明，这种方案虽然不是完全不可行，但执行效率比较低，且信息夹杂在一起不利于后期的调试和维护。

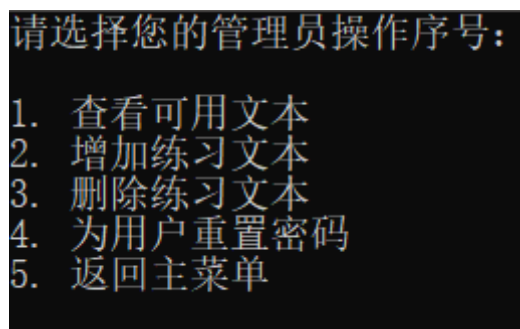
**S:** 基于此，我将所有用户的id放在一个名为“ids.txt”的文件中，再为每个用户建立一个以其id命名的文件存放相关的用户信息和游戏记录，使文件的脉络更清晰，便于维护和查找。

## Testing Snapshots

### 多级菜单



```
请选择您的操作序号:
1. 玩家登录
2. 玩家注册
3. 管理员操作
4. 退出程序
```

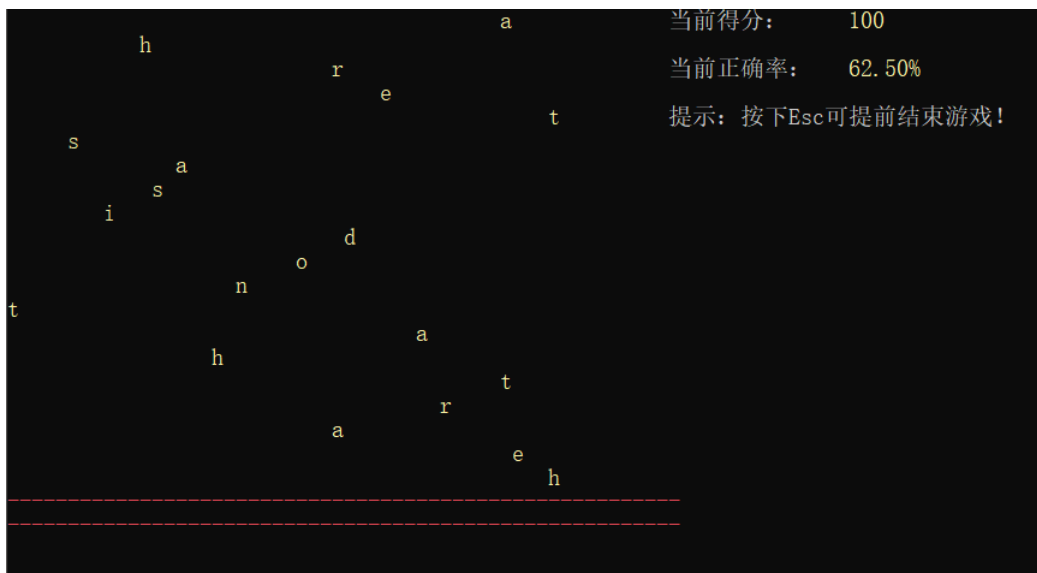


```
请选择您的管理员操作序号:
1. 查看可用文本
2. 增加练习文本
3. 删除练习文本
4. 为用户重置密码
5. 返回主菜单
```

请选择以下玩家操作：

1. 开始游戏！
2. 查看练习记录
3. 修改密码
4. 查看排行榜
5. 返回主菜单

## 游戏界面



## 游戏记录查看

以下是你的练习记录：

编号： 1  
难度： EASY  
得分： 20  
持续时间： 0.10min  
正确率： 80.00%

编号： 2  
难度： HELL  
得分： 500  
持续时间： 0.50min  
正确率： 45.45%

编号： 3  
难度： HARD  
得分： 915  
持续时间： 1.00min  
正确率： 70.11%

## 注册&登录

```
请输入您的用户名： （提示：不能包含空格、回车等空字符）
eric
请输入您的密码：
12345
请输入您的昵称：
Eric
```

```
请输入您的用户名和密码：
eric
12345
```

## 练习文本维护

```
以下是当前可用的文本：
0. Alice in Wonderland
1. Little Woman
2. Frankenstein
```

```
请输入希望增添的文本名称：
Pride and Prejudice
```

```
请输入希望删除的文本名称：
Pride and Prejudice
```

## 排行榜

用户名	得分	练习文本
111	915	Lttle woman
eric	876	Alice in Wonderland
simon	800	Frankenstein
111	500	Frankenstein
eric	233	Frankenstein