컴퓨터 공학 프로그래밍 팀프로젝트
‘I Love Soju’

[팀명] : 그라가스

[팀장] 김제헌 201311269
[팀원] 김상민 200910044
[팀원] 김태준 201311271
[팀원] 김병식 201311264

최종 보고서 : 기본적으로 1,2차 보고서의 내용도 조금씩 언급하였고,
프로젝트를 내용 및 진행하면서 느꼈던 내용을 중점으로 설명했습니다.

목차

컴퓨터 공학 프로그래밍 팀프로젝트
‘ I Love Soju ’. -그라가스-

1. 프로젝트 시작단계

1-1. 프로그램 개발목표

수업시간에 배운 C언어를 이용하여 다양한 종류의 술게임과 추가적인 이벤트를 구현시킨다.

1-2. I Love Soju의 소주제로 술게임을 설정한 이유

1) 술! 그것도 소주 하면 기본적으로 OT , MT 때의 술게임들이 떠올라서도 있다.
2) 또한 처음 1차 2차보고서에서 썼듯이 판매자 쪽보다는 구매자 쪽에서 덜 단조롭고 더욱 다양한 이벤트들이 있어서 소재가 풍부할 것 같았다.
3) 마지막으로 다양한 이벤트들이 있다 보니까, 기본적인 큰 틀(main.c) 위에서 게임을 종류별로 각각 팀원들이 맡아서 만들면 역할배정도 쉽게 될 것 같았다.

1-3. 문제, 요구사항 분석

1) 어떤 술게임을 만들어야 할 것인가?
이는 자신이 하고 싶은 데로, 만들고 싶은 데로 게임을 만든다고 되는 일이 아니었다. ‘게임을 어떻게 이끌어 갈 것인가?’, ‘컴퓨터가 게임에서 걸리게 하려면 어떻게 해야 하는가?’ 라는 문제와 부딪히게 된다.
‘술게임’에는 여러 가지 종류가 있고, 기본적으로 주변의 상황에 대한 즉각적인 판단, 순발력, 반응을 요구하는 것이 특징이다. 따라서 이러한 게임들을 구현하기 위해서는 주변의 상황을 제시하는 것과 더불어 한마디로 ‘타이밍’을 측정 할 수 있어야 했다.
하지만 이러한 조건은 프로젝트를 준비하는 초기단계에는 너무 높은 벽이었다. 나 자신도 C를 잘 알고 있는 편은 아니었고, Time함수 부분은 아직도 잘 모를뿐더러 설령 잘 알고 있다고 해도 팀원들이 잘 몰랐기에 보편적으로 그런 게임을 만들 수 가 없었다.
그래서 알고 있는 술게임이 그렇게 적지는 않았지만, 구현 할 수 있는 술게임에는 한계가 있었다.
그렇다보니 내가 지금까지 알고 있었던 ‘술게임’의 종류들이 무엇이 있는지 고민해 보기 시작했고, 그 중에서 게임으로 ‘구현’ 한다면 어떤 게임이 가능할 수 있을지를 생각해봤다.
그중에서 가능성이 보였던 것이 베스킨라빈스 31, 업앤다운 게임, 가위바위보를 이용한 게임 등등이었다. 이들의 공통적인 요소는 게임의 승패를 결정짓는 요인이 앞에서 말했던 ‘타이밍’과는 관련이 없는 ‘운’이라는 것이다.
이는 '운'적인 요소는 rand함수를 이용해서 충분히 구현할 수 있다고 생각해서 구현할 게임들을 모두 운과 관련된 게임들로 작성했다.

2) CUI 화면 전개를 어떻게 해 나갈 것인가?
보통 GUI로만 구성된 프로그램을 사용해 왔지만 GUI의 경우, C의 API까지 갈 것도 없이, 아직 모르기 때문에 애당초 포기했었고 처음부터 CUI로 가기로 방향을 잡았었다.
여기서는 ‘운’으로 가든 어떤 방식으로 나가든 간에 게임을 이끌어 나갈 때, ‘화면을 어떻게 보여주면서 이끌어 나갈 것인가?’ 가 중요한 관점이었다.

3) 술게임의 흐름을 어떻게 구현할 것인가?
보통 술게임에서 이전게임이 끝난 후에 걸렸던 술래가 원하는 게임이 시작되며, 순서대로 돌아가는 게임이라면 술래로부터 시계방향[반시계방향이든]으로 돌아가면서 게임이 진행되기 때문에 이러한 사실도 고려해 줘야 했다.
이는 ‘각각의 술게임을 할 때, 술래가 나인경우와 컴퓨터인 경우의 고려를 어떻게 할 것인가?’ 라는 질문으로 좀 더 구체화 시킬 수 있었다.

4) 술게임에 걸렸을 때의, 인트로 종류는 무엇이 있는가?
딱히 설명할 말은 없다. 말 그대로 ‘어떤 인트로를 보여줄까?’ 하고 생각했었다.

5) 술게임 외의 다양한 이벤트에는 무엇이 있을까?
술게임에는 여러 가지 술게임과 더불어 게임설명[마시면서 배우는 술게임!] 및 게임진행, 인트로, 벌칙수행과 관련된 내용들이 있지만, 이 밖에 부가적으로 ‘어떤 내용들을 더 추가시켜볼까?’ 라는 생각을 했었다.

6) 전체적인 흐름 및 설계를 어떻게 구현 할 것인가?
처음에 가지고 있었던 생각은 위에 적었듯이 큰 틀(main.c)을 만들고, 그 위에 게임 종류들을 개인별 작성, 메뉴와 switch-case구조를 이용해서 흐름을 분배할 생각을 가지고 있었다. 하지만 대략적이 아닌 ‘세부적으로 어떻게 구현 할 것인지?’가 문제였다.

7) 수업시간에 배운 내용을 어디에 어떻게 적용할 것인가?
프로젝트를 진행하는 도중에도 수업시간에서는 포인터, 구조체, 힙 할당, 파일 입출력 등등의 많은 내용들을 배우고 있었고 가능한 많이 프로젝트에 적용시켜 보고 싶었기에 ‘어디에 어떻게 적용할 것인가?’의 질문이 계속 구석에 자리 잡고 있었다.

1-4. 프로그램 설정 [1,2차 제안서의 흐름을 거의 갖고 옴]

1) 건국대학교 컴공과에서 종강파티로 ㅁㅁ식당에 감. : 시점이 종강시즌이다 보니.

2) 총 6개의 테이블이 존재 : 처음에는 그중 1번 테이블에 입석. 추가 4명의 사람들.

3) 술 종류는 Only 소주 : 1병에 8잔으로 계산.

4) 모든 사람에 대해서 한계 주량을 3병으로 제한 : 자기 자신의 주량을 못 정함.
각각의 주량이 자기 목숨. 즉, 목숨은 총 24잔. 주량이 다 차면 집으로 귀가한다.

5) 술게임에서 걸리면, 인트로와 함께 벌칙으로 술을 마신다. : 당연한 것 같지만 각 게임마다 인트로가 나오고 해당 술래의 주량을 증가시키지 않고 한 번에 처리하려는 설정.

6) 시작할 때 또는 자신이 전 술게임에서 걸렸던 경우만 게임을 선택 가능
: 컴퓨터가 걸리면 자동으로 램덤게임 실행.

7) 주량이 쌓이다가 자신이 아웃되면 집으로 귀가, 즉 게임 오버.
컴퓨터 중 한명이 아웃되면 뉴페이스 한명이 더 추가되면서 계속 진행.

8) 술게임 도중에 선택사항을 선택 가능 : 메인메뉴로 ‘1. 술게임 진행’, ‘2. 주량상황 확인하기’, ‘3. 테이블 옮기기’, ‘4. 화장실 가기’, ‘5. 집으로 가기’의 선택사항 들이 있다.

2. 프로젝트 결과물

- 저희 그라가스 팀은 주제를 술게임으로, ‘다양한 종류의 술게임과 추가적인 이벤트를 구현’을 개발 목표로 설정하고 프로젝트를 시작했습니다.
- 결과적으로 위에서 쭉 적었듯이 말 그대로 ‘술게임’을 구현하게 됐습니다.

- 명확한 목표를 설정한 후, 프로그램의 내용을 팀원들과 함께 생각하여 ‘운’적인 요소로 게임의 승부가 결정되는 7종류의 게임을 엄선, 각자 분담하여 코딩을 시작했습니다.

- 기본적으로 ‘술게임을 한다’ > ‘진 사람이 술을 마신다’ 의 반복 구조입니다.
여기에 저희 팀은 각종 추가 요소를 집어 넣고, 각 항목들에서 예외처리 기능을 수행할 수 있도록 했습니다.

- 프로그램 설정대로, 메인메뉴로는 ‘1. 술게임 진행’부터 ‘5. 집으로 가기’까지 다섯가지의 선택사항들이 있습니다.
- ‘1. 술게임하기’ 가 핵심적으로 처음에 밀고 나가고자 했던 목표고, 이 외의 나머지선택지들은 ‘술게임 외의 다양한 이벤트에는 무엇이 있을까?’라는 질문과정에서 부가적으로 추가된 선택지들입니다.
- ‘메인 메뉴선택하기’는 gotoxy를 이용해서 작성하였습니다.

2-1. 술게임 하기

1) 총 7개의 개임과 램덤게임을 합쳐서 8가지의 선택을 할 수가 있다.
- ‘게임 메뉴선택하기’ 역시 gotoxy를 이용해서 작성하였다.

2) 마지막으로 술을 마셨던 술래가 게임을 선택 할 수 있다.
- 시작 또는 자신이 전 술게임에서 걸렸던 경우만 플레이어가 게임을 선택 할 수 있다.
- 컴퓨터가 걸리면 자동으로 램덤게임이 실행된다.

3) 게임을 선택 시, 처음 하는 게임에 한해서 게임설명이 나온다.
- 이후 다시 게임을 들을 것인지 묻는다.
1번을 선택하면, 이후 게임을 하는 도중에 다시 똑같은 게임이 나올 때 같은 설명이 나오면서 다시 게임을 들을 것인지 묻는다.
2번을 선택하면, 같은 게임을 하게 될 때 설명 없이 바로 게임으로 진행된다.

4) 설명 이후 게임이 진행된다.
5) 게임에서 걸리면, 인트로와 함께 걸린 사람이 벌칙으로 술을 마시고 다시 메인화면으로 돌아온다. 인트로는 램덤으로 실행된다.

6) 앞에서 말했듯이 해당게임에서 걸린 사람이 다음 게임에서 게임을 선택 할 수 있다.

2-2. 주량상황 확인하기

현재 테이블에 앉아있는 사람들의 주량을 알 수 있다.
- ”사람이름 : a병 b잔 마심. 총 c잔” 으로 나타냈다. a x 8 + b = c 의 관계이다.

2-3. 테이블 옮기기

- 총 6개의 테이블이 표시된다. 각 테이블에는 미리 정해둔 플레이어들이 배치되어 있으며 자신이 원래 있던 테이블은 1번 테이블이다.
- 사용자는 키보드의 상하좌우키를 이용하여 각 테이블로 가면 창이 뜨고, 확인 후 해당 테이블에 참가하게 된다. 다시 올수도 있고, 이외의 5곳으로도 갈 수가 있다.
- gotoxy를 나름대로 중점적으로 활용했다.

2-4. 화장실 가기

- 크게 별 건 없다. 그저 사용자가 화장실에 갔다가 온다.
- 부가 효과로 화장실에서 약간 취기가 가시며 사용자의 현재 주량이 소량 감소한다.
- 나름대로의 밸런스 조정을 위해서 화장실을 5번째 갈 때 마다 주량이 1씩 감소하며,
- 주량이 0일 때 화장실에 갈 경우, 더 감소해서 -1이 되지 않도록 당연히 처리했다.

2-5. 집으로 가기

- 역시 별거 없다. 그냥 사용자가 집으로 간다. (사실상, 프로그램 종료)
- 부연설명으로는 실수로 눌렀을 경우를 방지하기 위해, 정말로 집으로 갈 것인지 물어보며 확인한다.
- 남으면, ‘남아서 논다’의 내용이 출력되며 계속 돌아가고,
  가면 ‘집으로 갔다’ 의 내용이 출력되며 프로그램이 종료된다.

3. 프로그램 설명

3-1. 프로그램 파일들

ILoveSoju.h와 ILoveSoju.c 그리고 Game_Spoon.c, Game_YiSunShin.c, Game_Updown.c, Game_TheGameOfDeath.c, Game_BR31.c, Game_RCP.c, Game_ChamChamCham.c 이 있다.

- ILoveSoju.h와 ILoveSoju.c가 핵심적이라고 볼 수가 있다.
ILoveSoju.h 는 define변수선언, 학생 구조체 선언 및 각 코드(ㅁㅁ.c)에서 사용하는 모든 함수의 선언을 포함하고 있다.

ILoveSoju.c 는 Main함수가 존재한다. 프로그램의 모든 흐름의 제어를 맡았다.
main함수에서의 메인메뉴와 Alcoholic_Games함수에서의 게임메뉴를 switch-case구조를 이용해서 흐름을 분배했다.
‘술게임’과 관련된 게임 코드를 제외하고, 이 외의 다양한 이벤트와 관련된 함수코드들이 들어있다.

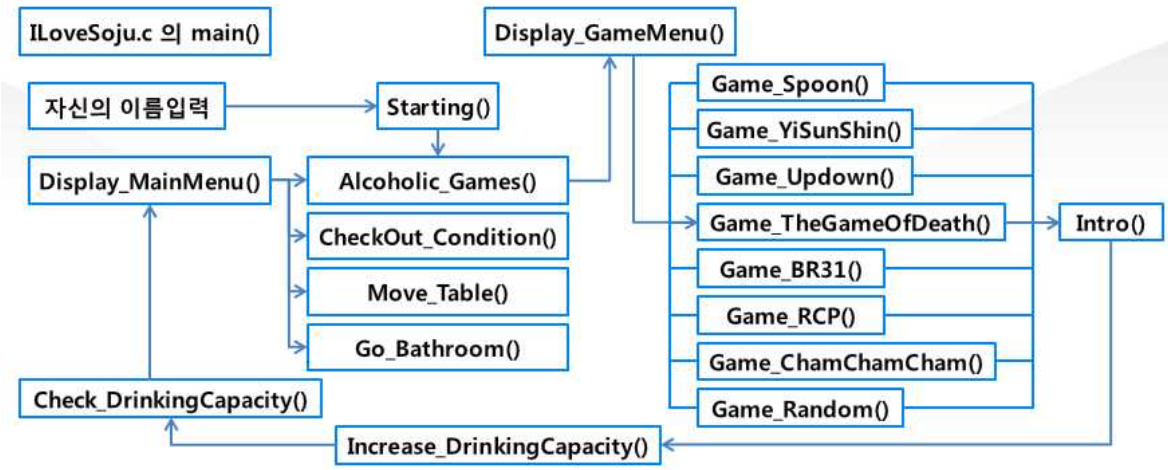
3-2. 구조체

이번 프로그램에 사용한 구조체로는 학생구조체가 있다.

```
char name[15];        //학생이름
int drinkingCapacity;//현재주량
```

이 구조체를 배열로 사용해서 각 학생들의 주량을 증가시키고, 누가 무엇을 했는지 보여주면서 CUI의 화면 전개를 해 나갔다. 그냥 플레이어, 컴퓨터1, 컴퓨터2, 컴퓨터3... 으로 표현하는 것 보다는 훨씬 더 낳아졌다.

3-3. 프로그램의 전체적인 흐름도 및 주요 함수들.



1) 시작하면서 사용자[자신]의 이름을 입력받습니다.

2) 이름을 입력받으면 도입부분에 정해진 대사가 출력되면서 뭔가의 말들이 오고간 다음에

3) 술게임을 시작하게 됩니다 (**2-1. 술게임하기** 항목 참조)

4) 술게임 메뉴를 선택하고 술게임을 진행한다.

5) 각 술게임이 끝나면서 패자가 정해진 후, 인트로와 함께 벌칙이 시작된다.

6) 전체적인 테이블의 주량을 증가시키고

7) 주량을 넘는 사람이 넘는 사람이 있는지 확인한 후에

8) 메인메뉴를 선택 할 수가 있다. 이때 사용자는 다시 술게임을 하거나 자리를 옮기기, 화장실에 가기 혹은 집에 가기를 선택할 수 있다.

9) 이 후 ‘집으로 가기’를 선택하지 않는 한 ‘메뉴선택하기’~’주량을 넘는 사람이 넘는 사람이 있는지 확인’의 흐름이 계속 반복적으로 돌아간다.

cf. ILoveSoju.c의 중요한 함수들은 대부분 넣어 놨다.

3-4. 기타등등 구현방법

1) Move_Table이라는 함수를 만들기 위해서, students[6][5]로 선언했다.

2) main()에서 int myTable이라는 변수를 이용해서 현재 자신이 위치한 테이블이 어디였는지 표현했다.
- 이후 함수(&students[myTable])로 필요한 주소를 넘겼다.

3) Increase_DrinkingCapacity함수에서 myTable을 제외한 다른 테이블 마다, 사람들 중의 한 명씩 주량을 증가시킨다.
- 주량의 전체적인 균형을 맞추기 위해서, 테이블로 가는 경우도 고려해 줘야 해서이다.
- 사용자가 첫 테이블에서 게임을 하다 다른 테이블로 옮겼을 때, 우리 테이블만 주량이 증가했고 나머지 플레이어들의 주량이 0인 사태를 미연에 방지하기 위해 사용자의 테이블에서 게임이 실행될 때마다 나머지 테이블에서 랜덤으로 1명의 주량을 1씩 증가시키게 했다.

4) Check_DrinkingCapacity함수에서 현재 자신의 테이블에 앉은 사람 중 주량이 한도 [3병=24잔]를 초과한 사람이 있는지 검사한다.
- 한도를 초과한 사람이 자신이면 바로 집으로 가도록, 다른 사람이면 새로운 사람으로 교체되도록 설정했다.
- static char name[13]과 static int index를 이용해서 매번 다른 사람이 교체되도록 했다.

5) Alcoholic_Games함수에서 static int drinker 라는 변수를 사용하여 각 함수별로 술래를 기억하면서 진행 할 수 있도록 했다.
- drinker가 0이면 [플레이어] Display_GameMenu함수로 게임선택 할수 있도록 했고, 0이 아니면[컴퓨터] rand함수로 바로 값을 넘기게 했다.

6) 각 게임 별로 static int gameExplanation 라는 변수를 사용하여 게임설명을 더 들을지 말지를 선택 할 수 있도록 했다.

7) students[i].name 외 에도 system("cls")와 rand함수, Sleep함수, 을 적절히 이용했다.
- 이외에도 ‘ ₩n’, ‘_’, ‘ ₩t’ 등을 이용해서 CUI를 꾸몄다.
- 추가적으로 ILoveSoju.c 는 gotoxy도 사용해서 CUI를 꾸몄다.

4. 마지막으로...

4-1. 구현하는데 힘들었던 점

결과물만 놓고 보면 얼추 처음에 구상하고 방향 잡은 데로 잘 만든 것 같지만 그 과정은 결코 순탄치 않았고, 그 원인은 결국 '준비 부족 및 설계 부실'이라고 볼 수가 있다.
기본적으로 구현할 내용은 기본 제안서에 대부분 적어놔서 '뭘 만들지?' 하며 고민할 필요가 없이 거의 다 구현했지만 각각을 구현시키는 경우에 대해서 설계가 안 되어 있었기에 시행착오가 좀 많이 생겼고 시간이 꽤나 걸렸다.

1) 처음 시작 할 때[2차 제안서까지], 기본적으로 각 게임에 대해서는 설계를 다 한 상태로 시작해서 문제가 없었으나, 이는 Alcoholic_Games( )만 해당하는 내용이었다.
- 나 같은 경우 각 함수에 대해서 알고리즘이 잘 돌아가서 그대로 짜놓고 잘 돌아가서 나중에 CUI같은 부분이나 Sleep함수로 적당이 꾸미며 추가시키기만 했지만, 설상가상으로 팀원들의 경우 내가 제안한 방식대로 설계도를 짜는 게 처음이다 보니 NS-Chart 자체도 많이 부실했고, - 그 당시에는 맘에 안 들었지만 별수 없이 그대로 모아서 제출했었다. - 아니나 다를까 프로젝트를 시작하면서 에러가 발생하기 시작했다. 병식이나 태준이의 경우 추가적으로 내가 보수점검 해주고 검토 해주고 하다 보니 시간이 오래 걸렸다.

2) 또한 맨 처음에 설계한 내용이 딱 '술게임'만 이기에 구현한 내용에 추가의 추가의 개념으로 이어져서 계속 점점 일이 커졌다.
- 예를 들어, 기본 개임이 만들어진 이후에 각 C파일과 게임함수 이름을 void Game_ㅁㅁ(void), Game_ㅁㅁ.c 등으로 통일하는 작업을 했으며,
- '구조체'를 이용한 Student students의 개념을 적용할 때, 팀원들에게 설명해 주고 각 게임에 students의 주소값을 넘겨서 void Game_ㅁㅁ(Student (*students))로 바꾼 후에 이름을 사용해라고 수정하라고 지시했으며,
- 마지막으로 static int drinker라는 개념을 사용해서 전 게임의 술래가 다음 게임을 선택하도록 하려고 할 때, int Game_ㅁㅁ(Student (*students), int drinker)로 바꿔서 drinker부터 게임이 시작되도록, 반환값으로 술래의 위치[=drinker]를 넘기도록 수정하라고 했더니 상민이 형 같은 경우, 초기에 게임을 짤 때, 너무 고정적으로 함수를 작성해 놔서 도저히 수정을 못한다고 하는 사태도 벌어지기도 했다.
- 부가적으로 static int gameExplanation 부분 적용하며 각자 다 양식을 통일 하라고 한 점, CUI로 각자 꾸미라고 했던 점 등등이 있다.

3) 전체 코드 중에서 Games부분만 설정한 후에 시작했고, ILoveSoju.c의 모든 흐름들은 switch-case구조라는 대충의 느낌만 알고 있었지 세부적인 것은 처음부터 어떻게 구현할지 생각해야 했다.
- 또한 gotoxy를 이용하여 Move_Table, Print_Box라든지 Display_MainMenu, Display_GameMenu를 만들 때도 처음부터 부딪혀서 gotoxy의 사용법을 익혔고 그렇게 사용해서 추가한 것이다.

- Move_Table 때문에 int myTable이라는 변수가 생겨나게 됐으며, 배열도 원래는 1차원 배열에서 2차원 배열로 변경하게 됐다. 그리고 Increase_DrinkingCapacity란 함수도 만들어야 했다.

4) 부가적으로는 gotoxy사용법을 익혀서 적용하는 점이 어려웠다.
막상 알고 나니 그렇게 어렵지는 않았지만 처음 하는 내용이다 보니, 인터넷상에 떠도는 예제들 수집하고 하나하나 테스트 해보고 이것저것 적용시켜보면서 익혔다.
그러고 나서 코드에 반영을 한 것이다.

계속 그렇게 기존 제안서에서 고려했던 내용들의 추가의 추가와 수정에 수정을 거듭하면서 지금의 ILoveSoju가 탄생했다.

4-2. 아쉬웠던 점

팀원들의 공통된 아쉬운 점은 UI가 꽤 난잡하다는 것이었다.

1) 게임들의 실행화면이 아주 단조롭다.
- 물론 복잡하게 보여서 보기 싫은 것 보다는 낫겠지만, 정말 검은 건 배경이요 하얀 건 글씨다 보니 더 꾸며보고 싶다는 생각이 솟구치는 것을 느낄 수 있었다.
- ILoveSoju.c는 gotoxy를 이용해서 나름대로 꾸며보긴 했는데, 이외의 게임과 관련된 부분은 팀원들이 gotoxy를 모르는데다가 지금까지 계속 일을 확장해왔기에 더 이상 일을 벌려서 부담을 주고 싶지 않아서 멈췄던 면도 있다.

2) 또한, 프로그램이 지루하다.
- 텍스트 출력 같은 부분에서 시간차를 두고 출력하기 위해 Sleep함수를 사용했지만, 시간 조절에 대한 타이밍을 일일히 실행시켜보면서 수작업으로 맞춰야 했는데 힘들었다.
- 각 게임부분이든 간에 적당한 Sleep함수를 사용했다고 했는데, 전체적으로 모아서 실행해보니 어디는 너무 빨리 넘어가고, 어디는 너무 길게 끌고 하는 점이 보였지만 제대로 수정을 하지 못했다. 또한 ‘어느정도가 적정한 시간인지?’라는 의문을 해결하지 못했다.
- 게임이 인트로 부분에서 3~10초 이상 씩 끊어먹으니, 프로그램의 전체적인 흐름을 느리게 만들었다. 테스트를 해보며 계속해보니 지루한 경향이 있었다.

3) 마지막으로, 프로그램의 이벤트, 선택사항이 부족했다.
- 하다못해 이벤트라도 좀 더 많이 늘려서 그럴듯하게 꾸며보고 싶어서 결과적으로는 초반에 구상했던 효과 및 이벤트들을 대부분 구현했지만, 계속 뭔가 많이 부족하다는 느낌을 받았다. ‘어떤 걸 더 추가할까?’하면서 ‘화장실 가기’ 같은 것도 추가해보면서 생각해 봤는데, 계속 일이 커지는 것을 감당 할 수 없어서 적당한 선에서 끝낸 것도 아쉽다.
- 그리고 있는 거라도 자세히 꾸밀까도 했지만, 이 역시 시간적으로 부족했고, 다른 일도 많았기에 기본적인 기능만 채우고 끝내야 한 점도 아쉽다.

4-3. 추가적으로 느낀점

매주 금요일마다 확률과 통계학 수업이 끝난 후로 새천년관에서 별도로 모여서 작업을 하면서 구상을 하고 하나, 둘씩 구현을 해 나가다 보니, 어느새 프로젝트 종료로 발표를 하고 있네요.

일단은 어떤걸 만들 지의 구상과 방향도 중요하기는 하지만 전체적인 흐름을 잡고 작업을 하는 것도 역시 중요하다는 것을 새삼 깨달았습니다.

아직도 팀 프로젝트가 익숙하지 않아서 소통이 원활하지는 않았던 것 같고, ‘이거 해라’, ‘저거 해라’ 라고 지시를 잘 내리지 못하다 보니 전체적으로 혼자서 작업을 많이 한 편인 것 같습니다.
어찌 보면 다른 사람들도 충분히 할 수 있는데도 불구하고 혼자서만 작업을 한 것 같기도 하고, 주로 혼자 만들다 보니 ppt 발표자료 및 최종결과보고서도 제가 전체적인 흐름을 달 알고 있다는 생각에 혼자서 작성하다보니 더 고생하고 괜히 더 분량이 많아 보이는 것 같기도 하네요.

매번 팀프로젝트를 할 때마다 느끼겠지만 앞으로는 좀 더 원활하게 소통하고 서로 각자 할 수 있는 범위 내에서 역할분담을 명확하게 시킬 필요성을 느끼고 있습니다. -by 김제헌

4-4. 역할분담

[팀장]김제헌 - 작성한 게임 : 최이선게임 이순신게임
- ILoveSoju.c 등 전체 흐름 잡고 코드 재조합.
- 전체적인 프로젝트 총괄.
- ppt발표자료 작성 및 발표

[팀원] 김상민 - 작성한 게임 : 더게임오브데스, 업다운게임

[팀원] 김태준 - 작성한 게임 : 묵찌빠, 참참참
- 데모 동영상 촬영 및 편집.

[팀원] 김병식 - 작성한 게임 : BR31 게임
- 게임시작 인트로부분 작성
- 최종보고서 초기 작성. -> 이후 제가 보고서 다시 쓸 때, 어느정도 참고함.

이외 각자 맡은 게임부분을 전체적인 양식, 틀에 맞춰서 수정하며 작성하였고, 추가적으로 디버깅작업[오류보고 등]도 같이 했습니다.

4-5. 사용 함수 리스트 - 코드는 생략

1) ILoveSoju 메인함수

```
int Display_MainMenu();//메인메뉴를 보여주고 선택을 입력받는 함수
void Alcoholic_Games(Student (*students));//술게임
int Display_GameMenu();//게음의 메뉴를 보여주고 선택을 입력받는 함수
int Game_Random(Student (*students),int drinker);//랜덤으로 게임이 선택되는 함수
int Intro();//술마실때의 인트로를 나타내는 함수
void CheckOut_Condition(Student (*students));//현재 주량을 확인하는 함수
int Move_Table(Student (*students)[5],int myTable);//테이블을 이동하는 창을 나타내는 함수
void Go_Bathroom(Student (*students));//화장실에 가는 함수
char Check_DrinkingCapacity(Student (*students));//한계주량을 확인하는 함수
void Increase_DrinkingCapacity(Student (*students)[5],int myTable);//주량을 증가시키는 함수
Print_Box(char ch[],int X_MIN,int Y_MIN,int X_MAX,int Y_MAX);//화면의 테두리
Print_Table(char ch[],int X_MIN,int Y_MIN,int X_MAX,int Y_MAX);//화면의 테이블
void gotoxy(unsigned int x, unsigned int y);//커서 좌표이동
void removeCursor(void);//커서의 점멸 제거
void SetColor(int font, int back);//텍스트의 색상을 변경
int Get_Only_Number();//숫자를 입력받는 함수
int Get_String(char (*string), int size);//문자를 입력받는 함수
void Strncpy(char (*string_coppy), char (*string), int size);//문자열을 복사하는 함수
void Starting(Student (*students));//시작할때의 스타팅부분
```

2) 각 게임의 함수

-Game_Spoon 숟가락게임

```
int Game_Spoon(Student (*students),int drinker);
void Input(int (*spoons)[5]);
int GameDecision(int (*spoons)[5]);
int GoingDecision(int (*spoons)[5]);
```

-Game_YiSunShin 이순신게임

```
int Game_YiSunShin(Student (*students),int drinker);
int Pitch_Coin();
```

-Game_Updown 업엔다운 게임

```
int Game_Updown(Student (*students), int drinker);
void UpInput(int *num, int min, int max);
void UpDown(int num, int answer, int *min, int *max, int (*com), int *drink);
void ComInput(int *num, int min, int max);
```

-Game_TheGameOfDeath 게임오브데스

```
int Game_TheGameOfDeath(Student (*students), int drinker);
void TheInput(int *num, int *play, Student (*students));
```

-Game_BR31 베스킨라빈스31

```
int Game_BR31(Student (*students),int drinker);
```

-Game_RCP 묵찌빠게임

```
int WhoWins1(int myHand1, int computerHand1);
int GetcomputerHand1();
int Game_RCP(Student (*students),int drinker);
```

-Game_ChamChamCham 참참참게임

```
void PrintWhoTrun(const int whoIsTurn,Student (*students),int drinker,int computer);
void PrintHands(int myHand, int computerHand,int drinker, int computer,Student (*students));
void PrintHands2(int myHand, int computerHand,int drinker, int computer,Student (*students));
int WhoWins(const int myHand, const int computerHand);
int Game_ChamChamCham(Student (*students),int drinker);
```

3). 주요 변수

```
typedef struct _student
{
    char name[15];          //이름
    int drinkingCapacity;//주량
}Student;

enum{
            BLACK, D_BLUE,    D_GREEN, D_SKYBLUE,   D_RED, D_VIOLET, D_YELLOW, GRAY,
D_GRAY, BLUE, GREEN, SKYBLUE, RED, VIOLET, YELLOW, WHITE,
};//사용할 색상을 열거
```