**시스템 프로그래밍 팀 프로젝트**

**<Avoiding Stars Game>**

**Report**

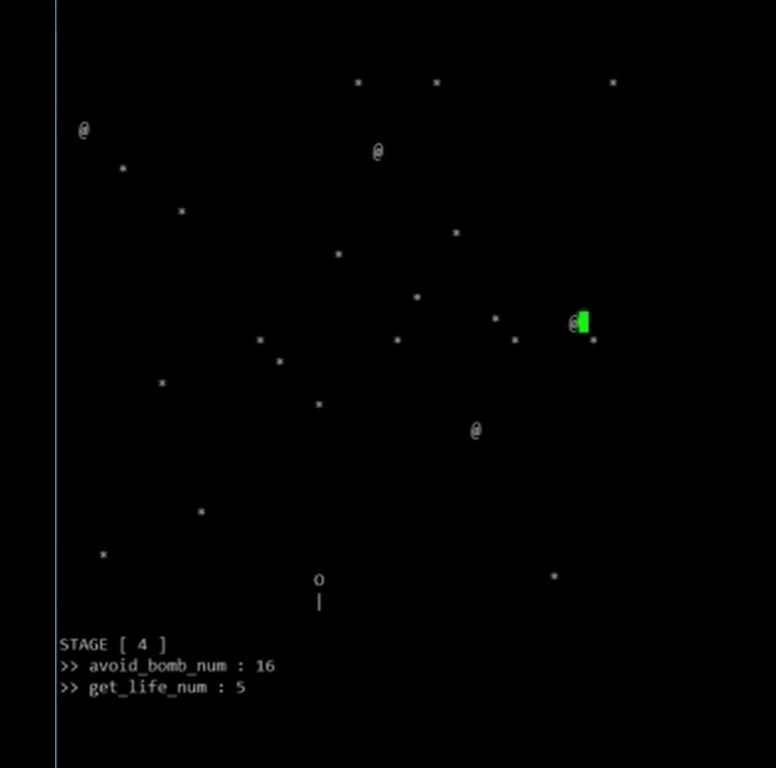
**2013105079 이중혁**

**2017111620 정다윤**

**2017116261 나지혜**

**<목차>**

1. **Avoiding Stars Game 소개**
2. **게임방식 설명**
3. **코드 설명**
4. **Avoiding Stars Game 소개**



linux환경에서 curses.h를 이용하여 제작한 아케이드 게임입니다.

이 게임은 제작된 field 안에서 하나의 문자를 이용한 별과 item들이 무수히 랜덤하게 아래로 내려오는데 field의 아래에 있는 유저를 의미하는 character가 좌우로 이동하여 별들을 피하고 item을 습득하면서 stages를 클리어하는 게임입니다.

1. **게임방식 설명**

**Start**



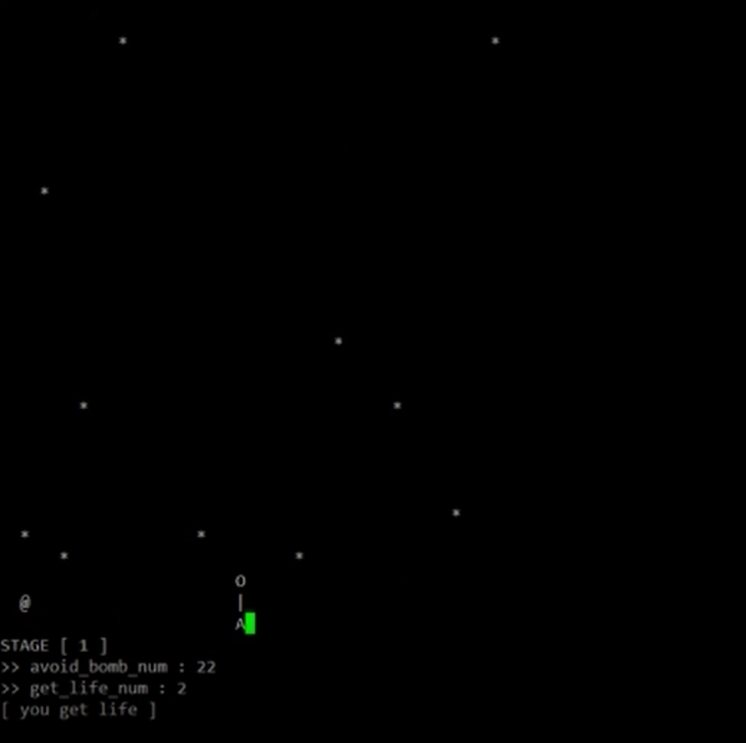
게임을 시작하면 start 화면이 표시가 됩니다. 게임에 대해 간략한 설명이 담겨있고 3초가 지나면 화면이 바꿔면서 게임을 플레이 할 수 있는 화면으로 넘어갑니다.

**PLAY(초기화)**



게임이 시작되면 처음 보이는 화면은 아래쪽에 위치한 Character와 좌측 맨 아래쪽에는 stage 단계, 피한 별의 개수, 가지고 있는 생명의 개수가 표시됩니다. 초기에 stage는 1이고 피한 별의 개수는 0, 가지고있는 생명의 개수는 1입니다.

**PLAY(실행)**



Charater는 좌우로 움직이면서 별을 피하고 아이템을 습득합니다.

별들은 field의 맨아래에 닿으면 피한 별의 개수 항목에 숫자가 1씩 증가합니다. 만약 별과 Character가 충돌한다면 가지고 있는 생명이 1씩 감소하고 item과 충돌한다면 생명이 1씩 증가합니다.

**PLAY(stage clear)**



Stage는 10단계까지 있고 Stage마다 피해야 되는 별의 개수가 다른데 1단계는 30개이고 단계가 늘어날수록 30개씩 늘어납니다. 피해야 되는 별의 개수가 최대에 달하면 stage를 클리어 합니다. 그러면 stage항목에서 stage의 숫자가 늘어납니다.

**PLAY(Game Over)**



가지고 있는 생명이 1이 남은 상황에서 한번 더 별과 Character가 충돌한다면 화면이 전환되고 game over 창이 뜹니다.

**3. 코드설명**

**- main()**

맨 처음에 게임이 시작되는 화면을 띄운다.

3초 있다가 게임실행 화면으로 넘어간다

**- init\_field()**

bad item과 good item이 떨어지는 천장을 출력한다

**- init\_game()**

게임의 기본적인 초기화 작업을 수행한다

max\_good\_item과 max\_bad\_item 은 각 스테이지에서 최대로 얻을 수 있는 아이템의 개수이다

아이템의 좌표는 랜덤하게 얻는다.

게임캐릭터의 초기 위치 값을 정해준다.

**- print\_bad\_item()**

아이템을 하늘에서 떨어지도록 출력하는 함수이다

아이템이 땅에 떨어졌을 경우에는 캐릭터가 그 아이템을 피했다는 뜻이므로

기존 위치의 폭탄은 삭제해주고 피한 폭탄의 숫자를 하나 늘려준다.

현재 상황을 알려주는 부분을 출력한다. (스테이지 수, 피한 폭탄 개수, 얻은 생명 개수)

만약 max\_good\_item이 count\_bad\_item과 같으면 스테이지를 클리어 했다는 의미이므로

다음 스테이지로 넘어간다. - endGame(CLEAR)

만약 max\_good\_item이 count\_bad\_item보다 크면 떨어진 폭탄 수가 아직 최대 개수에 다다르지 않았으므로 폭탄을 다시 위에서 내리게 한다.

아이템이 아직 땅에 떨어지지 않은 경우에는 기존위치의 폭탄을 삭제하고 새로운 위치에 폭탄을 출력한다.

**- print\_good\_item()**

print\_bad\_item()과 같은 방법으로 생명을 하늘에서 떨어지도록 출력한다.

**- check\_bad\_hit()**

이 함수는 캐릭터가 폭탄에 맞았는지 확인하는 함수이다

사용자의 열과 행이 모두 폭탄의 열과 행에 일치하면, 폭탄에 맞았다는 뜻이므로 생명을 하나 줄인다.

만약 생명이 0이 되면 게임이 끝나므로 endGame(DEAD)함수로 이동한다.

**- check\_good\_hit()**

이 함수는 캐릭터가 생명에 맞았는지 확인하는 함수이다.

사용자의 열과 행이 모두 생명의 열과 행에 일치하면, 생명을 얻었다는 의미이므로 생명을 하나 늘린다.

**- game\_clear()**

스테이지 10까지 끝내면 게임을 클리어한 것이다.

화면을 clear하고 게임이 끝났다는 문구를 출력한다

**- endGame()**

flag == DEAD이면 사용자는 죽은 것이고 DEAD가 아닐 시에는 사용자가 스테이지를 클리어한 것이다.

죽었을 때에는 화면을 clear하고 게임오버 화면을 출력한다. 이때 close\_keyboard()함수를 통해서 좌우로 움직이기 위해서 꺼둔 키보드 모드를 다시 원래대로 돌려준다.

스테이지를 클리어했을 때에는 화면을 clear하고 스테이지 클리어 했다는 화면을 출력한다.

다음 스테이지로 넘어가야 하니까 다시 맵을 출력하고, 스테이지 변수는 증가하고 게임을 초기화한다. 스테이지가 올라가면 갈수록 난이도를 높이기 위해서 내리는 폭탄의 개수를 늘리고 게임의 스피드는 usleep()을 이용하여 더 빠르게 하였다.

**- start\_game()**

init\_keyboard()함수를 통해서 키보드 모드를 꺼주고 키보드 j, l 입력에 따라서 좌우로 움직이게 한다.

만약 input이 j이면 캐릭터는 왼쪽으로 움직이도록 하고, input이 l이면 오른쪽으로 움직이도록 한다.

input에 아무것도 들어오지 않으면 그때 print\_bad\_item()과 print\_good\_item() 을 통해 폭탄과 생명이 무작위로 내려오도록 한다.

그리고 check\_bad\_hit()와 check\_good\_hit()를 통해 폭탄이나 생명에 맞았는지 확인한다

**- init\_keyboard()**

tcgetattr()를 통해서, 현재 터미널 드라이버의 모드를 initial\_settings에 저장해두고, new\_settings에 바꿀 터미널 드라이버 모드를 저장한다. 이때는 CANONICAL모드와 ECHO모드를 꺼줘야 한다. 키보드를 입력 받으면 그 입력 받은 문자를 출력하는 것이 아니라 캐릭터를 좌우로 움직이게 해야 하기 때문이다.

설정을 new\_settings에 다 저장하고 나면, 현재 터미널 드라이버 모드를 new\_settings로 바꿔준다.

**- close\_keyboard()**

터미널 드라이버 모드를 tcsetattr()을 통해 원래 모드로 다시 바꿔준다.

init\_keyboard()에서 저장해둔 initial\_settings에 원래 모드의 특성들이 저장 되어있다.