

1. '시작 화면' 재구성, 로그인 기능 변경

```

3  sosil_ataxx()
4  {
5      echo -e "
6      echo -e "
7      echo -e "
8      echo -e "
9      echo -e "
10     echo -e "\n"
11
12     echo -e "
13     echo -e "
14     echo -e "
15     echo -e "
16     echo -e "
17     echo -e "\n"
18     echo -e "
19 }

```



- 시작화면을 구성하는 함수명인 sosil_ataxx 함수입니다.

```

21 start_page_0()
22 {
23     clear
24     sosil_ataxx
25     echo -e "          \033[44m  1P LOGIN  \033[0m"      "  \033[44m  SIGN IN \033[0m"
26     echo -e "\n"
27     echo -e "          \033[44m  2P LOGIN  \033[0m"      "  \033[44m  SIGN OUT \033[0m"
28     echo -e "\n"
29     echo -e "          \033[44m  JOIN    \033[0m"      "  \033[44m  EXIT    \033[0m"
30     echo -e "\n"
31 }

```

- 하이라이팅 되는 부분만을 따로 함수로 만들어 작성했습니다.
- 'start_page'로 시작하는 함수명에 숫자로 구분을 하였습니다.
- start_page_0은 하이라이팅되는 부분이 없는 함수입니다.
- start_page_1 ~ start_page_6은 숫자에 따라 하이라이팅 되는 부분이 다릅니다.



```

2023 sosil_location()
2024 {
2025     count=0
2026     now_location=2
2027     while [ "$count" -ne 1 ]
2028     do
2029         move $now_location
2030         now_location="$now_location"
2031         p1_loginSuccess
2032         p1_loginSuccess=$p1_loginSuccess
2033
2034         if [ "$now_location" -eq 0 ]; then
2035             if [ "$p1_loginSuccess" -eq 0 ]; then
2036                 sosil_loginPage_1 $id
2037             else
2038                 start_page_1
2039             fi
2040             if [ "$oper" -eq 0 ]; then
2041                 clear
2042                 p1_login_page
2043                 p1_location
2044             fi
2045         elif [ "$now_location" -eq 1 ]; then
2046             if [ "$p1_loginSuccess" -eq 0 ]; then
2047                 sosil_loginPage_2 $id
2048             else
2049                 start_page_2
2050             fi
2051             if [ "$oper" -eq 0 ]; then
2052                 clear
2053                 signIn_login
2054                 signIn_location
2055             fi

```

- count 변수는 EXIT 되기 전까지, 반복하기 위해서 만든 변수이다.
- now_location 변수는 현재 커서의 위치를 알기 위해서 만든 변수이다.
- move는 함수이고, 밑에서 설명할 예정입니다.
- p1_loginSuccess는 함수이고, 1p가 로그인 되었는지 안되었는지 확인하는 함수입니다.
- 2034 - 2055까지의 코드는 1P_LOGIN과 SIGN IN이 하이라이팅 되는지, 또한 1P가 로그인 되었다면 1P의 id를 보여주게 만들어주는 코드이다.

```

2056         elif [ "$now_location" -eq 2 ]; then
2057             if [ "$p1_loginSuccess" -eq 0 ]; then
2058                 sosil_loginPage_3 $id
2059             else
2060                 start_page_3
2061             fi
2062             if [ "$soper" -eq 0 ]; then
2063                 clear
2064                 p2_login_page
2065                 p2_location
2066             fi
2067         elif [ "$now_location" -eq 3 ]; then
2068             if [ "$p1_loginSuccess" -eq 0 ]; then
2069                 sosil_loginPage_4 $id
2070             else
2071                 start_page_4
2072             fi
2073         elif [ "$now_location" -eq 4 ]; then
2074             start_page_5
2075             if [ "$p1_loginSuccess" -eq 0 ]; then
2076                 sosil_loginPage_5 $id
2077             else
2078                 start_page_5
2079             fi
2080         elif [ "$now_location" -eq 5 ]; then
2081             if [ "$p1_loginSuccess" -eq 0 ]; then
2082                 sosil_loginPage_6 $id
2083             else
2084                 start_page_6
2085             fi
2086             if [ "$soper" -eq 0 ]; then
2087                 count=1
2088             fi
2089         fi
2090     done
2091 }

```

- 2056 - 2089까지의 코드는 2P_LOGIN, SIGN OUT, JOIN, EXIT이 하이라이팅 되는지 또한, 1p가 로그인이 되었다면 1p의 아이디도 보여주는 코드를 작성해 놓았다.
- 2080 - 2088은 EXIT이 하이라이팅 되는 코드를 작성해 놓았고, EXIT이 하이라이팅 되었을 때, enter를 누르면 while문이 종료되어 프로그램이 종료되도록 만들었다.

```

2004 function move()
2005 {
2006     input_location
2007     oper="$operator"
2008     if [ "$oper" -eq -2 ]; then
2009         temp=`expr $now_location - 2`
2010         now_location=`expr $temp % 6`
2011     elif [ "$oper" -eq 2 ]; then
2012         temp=`expr $now_location + 2`
2013         now_location=`expr $temp % 6`
2014     elif [ "$oper" -eq 1 ]; then
2015         temp=`expr $now_location + 1`
2016         now_location=`expr $temp % 6`
2017     elif [ "$oper" -eq -1 ]; then
2018         temp=`expr $now_location - 1`
2019         now_location=`expr $temp % 6`
2020     fi
2021 }

```

- move 함수이다.
- input_location은 함수이고 방향키를 받는 함수이다.
- 2008 - 2020까지의 코드는 input_location에서 받은 방향키에 적절한 연산을 통해 now_location의 위치를 바꾸도록 만든 코드이다.
- '위'방향키는 -2, '아래'방향키는 +2, '오른쪽'방향키는 +1, '왼쪽'방향키는 -1이고, 나머지 연산자를 통해 now_location값을 구체적으로 정하게 하였다.


```

1982 input_location()
1983 {
1984     read -n 1 input
1985     if [ $input == '^[' ]; then
1986         read -n 1 input
1987         if [ $input == '[' ]; then
1988             read -n 1 input
1989             if [ $input == 'A' ]; then
1990                 operator=-2
1991             elif [ $input == 'B' ]; then
1992                 operator=2
1993             elif [ $input == 'C' ]; then
1994                 operator=1
1995             elif [ $input == 'D' ]; then
1996                 operator=-1
1997             fi
1998         fi
1999     elif [ $input == `` ]; then
2000         operator=0
2001     fi
2002 }

```

- input_location 함수이다.
- A: 위, B: 아래, C: 오른쪽, D: 왼쪽을 입력받았을 때, operator 변수에 -2, 2, 1, -1을 저장하고, 만약 enter라면 operator 변수에 0을 저장한다.

```

177 p1_login()
178 {
179     echo -e "
180     echo -e "
181     echo -e "
182     echo -e "
183     echo -e "
184     echo -e "\n"
185 }

```

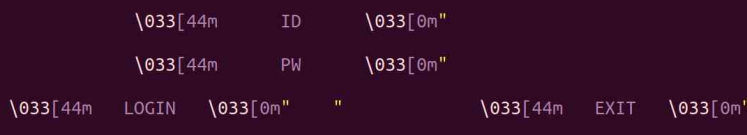


- p1_login 함수이다.
- 1P_LOGIN 페이지에서 보여지는 부분을 함수로 만들었다

```

187 p1_login_page()
188 {
189     clear
190     p1_login
191     echo -e "
192     echo -e "\n"
193     echo -e "
194     echo -e "\n"
195     echo -e "
196     echo -e "\n"
197 }

```



- p1_login_page 함수는 하이라이팅을 하기 위해 만든 함수이다.
- p1_login_page는 하이라이팅 되는 부분이 없는 함수이다.
- p1_login_page_1 - p1_login_page_4는 ID, PW, LOGIN, EXIT 중 한 부분이 하이라이팅 되는 것을 함수로 만들었다.



- 방향키를 누르면 원하는 부분이 하이라이팅 되는 모습을 볼 수 있다.
- 처음 하이라이팅을 할 때, '위' 방향키를 눌러야 정상적으로 하이라이팅이 가능하다.

```

199 p1_login_page_1()
200 {
201     clear
202     p1_login
203     echo -ne "
204     echo -e "\n"
205     echo -ne "
206     echo -e "\n"
207     echo -e "
208     echo -e "\n"
209 }

```



- p1_login_page_1 - p1_login_page_4까지는 id가 보여지면서, ID, PW, LOGIN, EXIT이 하이라이팅 되게 만든 함수이다.

```

4031 p1_location()
4032 {
4033     count=0
4034     now_location=2
4035     change=0
4036     while [ "$count" -ne 1 ]
4037     do
4038         move $now_location
4039         now_location="$now_location"
4040
4041         if [ "$now_location" -eq 0 -o "$now_location" -eq 1 ]; then
4042             if [ "$oper" -eq 0 ]; then
4043                 p1_login_page_1
4044                 read id
4045                 p1_enterId $id
4046                 change=1
4047             else
4048                 if [ "$change" -eq 1 ]; then
4049                     p1_enterId $id
4050                 else
4051                     p1_login_page_1
4052                 fi
4053             fi
4054         elif [ "$now_location" -eq 2 -o "$now_location" -eq 3 ]; then
4055             if [ "$oper" -eq 0 ]; then
4056                 p1_enterId_pw $id
4057                 read pw
4058                 p1_enterIdPw $id $pw
4059             else
4060                 if [ "$change" -eq 1 ]; then
4061                     p1_enterId_pw $id
4062                 else
4063                     p1_login_page_2
4064                 fi
4065             fi

```

- p1_location 함수이다. sosil_ataxx화면에서 1P_LOGIN에 enter를 눌렀을 때, 나오는 1P_LOGIN 화면에 대해 하이라이팅 되는 것을 구현하였다.
- count를 사용하여 반복문이 계속 참인 조건을 만들었고, move를 통해 now_location값을 받아왔다.
- now_location이 0 혹은 1일 때, ID 부분이 하이라이팅 되게 만들었고, now_location이 2 혹은 3일 때, PW 부분이 하이라이팅 되게 만들었다.
- change변수가 0이면 기존 그대로의 모습이 보여지지만, change변수가 1이면 1p의 아이디가 화면에 보여지게 만들었다.
- p1_login_page_1함수에 id 값을 전달하여 화면에 보이게 했다.
- p1_enterId_pw는 PW입력을 받을 때, id가 화면에 보이게 했다.
- id와 pw를 다 입력 받으면, p1_enterIdPw 함수에 id와 pw를 전달하여 화면에 보이게 했다.

```

4066         elif [ "$now_location" -eq 4 ]; then
4067             if [ "$soper" -eq 0 ]; then
4068                 join $id $pw
4069                 result1=$result
4070                 if [ "$result1" -eq 0 ]; then
4071                     p1_loginSuccess=0
4072                     p2_loginSuccess=5
4073                     p1p2_join $id $p2_loginSuccess
4074                     sosil_location
4075                 else
4076                     p1_loginSuccess=1
4077                     count=1
4078                 fi
4079             else
4080                 if [ "$change" -eq 1 ]; then
4081                     p1_enter_login $id $pw
4082                 else
4083                     p1_login_page_3
4084                 fi
4085             fi
4086         elif [ "$now_location" -eq 5 ]; then
4087             if [ "$soper" -eq 0 ]; then
4088                 if [ "$change" -eq 1 ]; then
4089                     p1_enter_exit $id $pw
4090                     count=1
4091                 else
4092                     p1_login_page_4
4093                     count=1
4094                 fi
4095             else
4096                 if [ "$change" -eq 1 ]; then
4097                     p1_login_page_4
4098                 else
4099                     p1_login_page_4
4100                 fi
4101             fi
4102         fi
4103     done
4104 }

```

- now_location이 4이면, JOIN이 하이라이팅 되고, enter를 누르면 ID와 PW의 계정이 있는지 확인해야한다.
- 이를 join 함수에 id, pw값을 넘겨서 result값을 받아와 result1변수에 넣는다.
- p1_loginSuccess가 0이면 로그인이 정상적으로 된것이고, p2_loginSuccess가 0이 아니면 p2는 아직 로그인이 안된 것이다.
- 만약, enter를 누르지 않았다면, id와 pw값을 p1_enter_login 함수에 넘겨 화면에 하이라이팅과, id, pw값이 보여지게 만들었다.
- now_location이 5이면, EXIT이 하이라이팅 되고, enter를 누르면 count가 1이 되어, 프로그램이 종료된다.
- p1p2_join 함수에 id, p2_loginSuccess변수를 전달해 id값이 조건에 맞게 보이도록 했다.


```

4250 function join()
4251 {
4252     file=`cat ./auth.txt`
4253     line=`echo $file`
4254     database=`echo $line | tr "," "\n"`
4255
4256     count=0
4257     id_success=1
4258     pw_success=1
4259
4260     for data in $database
4261     do
4262         standard=`expr $count % 2`
4263         if [ "$standard" -eq 0 ]; then
4264             if [ "$count" -eq 0 -o "$count" -eq 4 ]; then
4265                 if [ "$data" == $1 ]; then
4266                     id_success=0
4267                 else
4268                     id_success=1
4269                 fi
4270             else
4271                 id_success=1
4272             fi
4273         elif [ "$standard" -eq 1 ]; then
4274             if [ "$count" -eq 1 -o "$count" -eq 5 ]; then
4275                 if [ "$data" == $2 ]; then
4276                     pw_success=0
4277                 else
4278                     pw_success=1
4279                 fi
4280             else
4281                 pw_success=1
4282             fi
4283         fi
4284         if [ "$id_success" -eq 0 -a "$pw_success" -eq 0 ]; then
4285             result=0
4286             break
4287         else
4288             result=1
4289         fi
4290         count=`expr $count + 1`
4291     done
4292 }

```

- join 함수이다.
- file 변수에 auth.txt를 불러오게 만들었고, database는 auth.txt에 있는 한 줄씩, ';'를 기준으로 구분하게 만들었다.
- for 반복문을 사용하여 database에 있는 id, pw, 승수, 패수 부분을 확인했다.
- standard 변수가 0이면 id, 승수 부분만 비교되게 만들었고, standard 변수가 1이면 pw, 패수 부분만 비교되게 만들었다.
- 그 중에서, count가 0이거나 4이면 id만 비교하게 되므로, p1_location이나 p2_location에서 전달받은 id값이 data 변수와 같다면, id_success값을 0으로 변경해준다.
- count가 1이거나 5이면 pw만 비교하게 되므로, p1_location이나 p2_location에서 전달받은 pw값이 data 변수와 같다면, pw_success값을 0으로 변경해준다.
- id_success가 0이고, pw_success가 0임을 동시에 만족하면 result 변수에 0을 저장한다.
- 이렇게, for문이 한번씩 끝날때마다 count변수를 1씩 증가시킨다.

```

4106 p2_location()
4107 {
4108     count=0
4109     now_location=2
4110     change=0
4111     while [ "$count" -ne 1 ]
4112     do
4113         move $now_location
4114         now_location="$now_location"
4115
4116         if [ "$now_location" -eq 0 -o "$now_location" -eq 1 ]; then
4117             if [ "$soper" -eq 0 ]; then
4118                 p2_login_page_1
4119                 read p2Id
4120                 p2_enterId $p2Id
4121                 change=1
4122             else
4123                 if [ "$change" -eq 1 ]; then
4124                     p2_enterId $p2Id
4125                 else
4126                     p2_login_page_1
4127                 fi
4128             fi
4129         elif [ "$now_location" -eq 2 -o "$now_location" -eq 3 ]; then
4130             if [ "$soper" -eq 0 ]; then
4131                 p2_enterId_pw $p2Id
4132                 read p2Pw
4133                 p2_enterIdPw $p2Id $p2Pw
4134             else
4135                 if [ "$change" -eq 1 ]; then
4136                     p2_enterId_pw $p2Id
4137                 else
4138                     p2_login_page_2
4139                 fi
4140             fi

```

- p2_location 함수이다.
- p1_location 함수와 동일한 방법으로 구현하였다.

```

4141         elif [ "$now_location" -eq 4 ]; then
4142             if [ "$soper" -eq 0 ]; then
4143                 join $p2Id $p2Pw
4144                 result2=$result
4145                 if [ "$result2" -eq 0 ]; then
4146                     p2_loginSuccess=0
4147                     p1p2_join $p2Id $p2_loginSuccess
4148                     sosil_login_location $p2Id
4149                 else
4150                     p2_loginSuccess=1
4151                     count=1
4152                 fi
4153             else
4154                 if [ "$change" -eq 1 ]; then
4155                     p2_enter_login $p2Id $p2Pw
4156                 else
4157                     p2_login_page_3
4158                 fi
4159             fi
4160         elif [ "$now_location" -eq 5 ]; then
4161             if [ "$soper" -eq 0 ]; then
4162                 if [ "$change" -eq 1 ]; then
4163                     p2_enter_exit $p2Id $p2Pw
4164                     count=1
4165                 else
4166                     p2_login_page_4
4167                 fi
4168             else
4169                 if [ "$change" -eq 1 ]; then
4170                     p2_enter_exit $p2Id $p2Pw
4171                 else
4172                     p2_login_page_4
4173                 fi
4174             fi
4175         fi
4176     done
4177 }

```

- join 함수에 id와 pw 값을 확인하여 result2에 result 값을 받아와, p2_loginSuccess 값이 0이면 정상적으로 로그인인 된 것이고, 0이 아니면 로그인이 안되는 상태이다.
- 로그인이 안되는 상태이면 프로그램을 종료한다.
- p2_enter_login 함수에 p2Id, p2Pw 값을 전달해 화면에 보여주도록 한다.
- now_location이 5이고, enter를 눌렀을 때, id와 pw 값이 보여지면서 종료를 할 것인지, 아닌지, 혹은 enter를 누르지 않고 하이라이팅만 되었을 때, id와 pw 값이 보여지면서 하이라이팅이 되는지 아닌지를 구분하기 위해 p2_enter로 시작하는 함수와, p2_login_page로 시작하는 함수를 만들었다.
- 2p까지 정상적으로 로그인이 되었다면, sosil_login_location 함수로 넘어간다.

```

2093 sasil_login_location()
2094 {
2095     count=0
2096     now_location=2
2097     while [ "$count" -ne 1 ]
2098     do
2099         move $now_location
2100         now_location="$now_location"
2101         p1_loginSuccess
2102         p1_loginSuccess=$p1_loginSuccess
2103
2104         if [ "$now_location" -eq 0 ]; then
2105             join_p1
2106         elif [ "$now_location" -eq 1 ]; then
2107             join_signIn
2108         elif [ "$now_location" -eq 2 ]; then
2109             join_p2
2110         elif [ "$now_location" -eq 3 ]; then
2111             join_signOut
2112         elif [ "$now_location" -eq 4 ]; then
2113             join_join
2114             if [ "$soper" -eq 0 ]; then
2115                 lobby_page
2116                 lobby_location
2117             fi
2118         elif [ "$now_location" -eq 5 ]; then
2119             join_exit
2120             if [ "$soper" -eq 0 ]; then
2121                 count=1
2122             fi
2123         fi
2124     done
2125 }

```

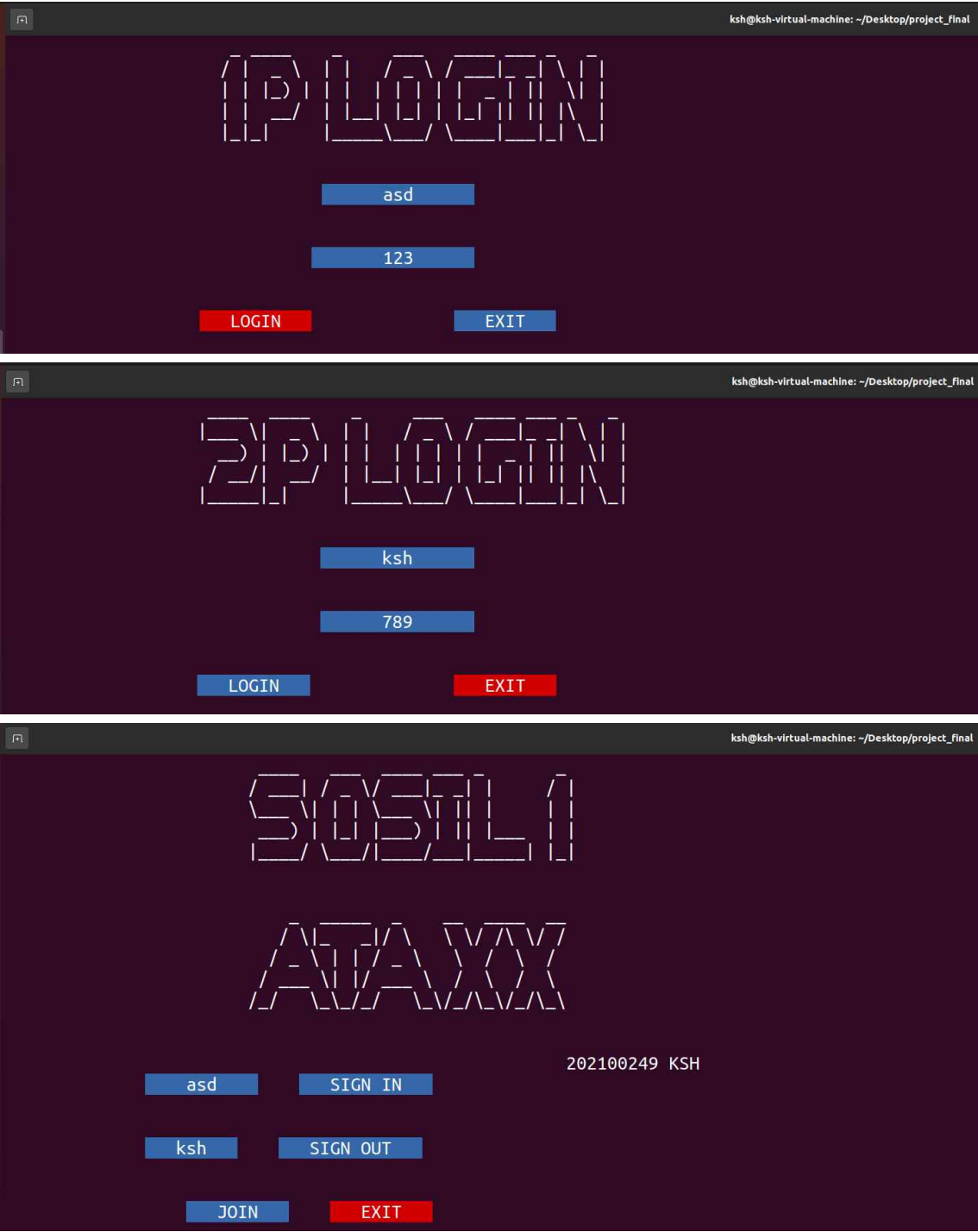
```

390 join_exit()
391 {
392     clear
393     sasil_ataxx
394     echo -e "          \033[44m    "$id"    \033[0m"          " \033[44m    SIGN IN    \033[0m"
395     echo -e "\n"
396     echo -e "          \033[44m    "$p2Id"  \033[0m"          " \033[44m    SIGN OUT   \033[0m"
397     echo -e "\n"
398     echo -e "          \033[44m    JOIN    \033[0m"          " \033[41m    EXIT    \033[0m"
399     echo -e "\n"
400 }

```

- join 으로 시작하는 함수는, 1p와 2p가 정상적으로 로그인이 되고, 1p의 아이디, 2p의 아이디를 각각 보여주게 하는 함수이다.
- 방향키 입력에 맞게, 하이라이팅 되는 부분을 "join_"뒤에 작성하여 방향키에 알맞게 하이라이팅 되도록 만들었다.

1번 UI 예시사진



2. '게임 대기 화면' - UI 구현

```
711 ataxx_lobby()
712 {
713     clear
714     echo -e "
715     echo -e "
716     echo -e "
717     echo -e "
718     echo -e "
719     echo -e "\n"
720     echo -e "
721     echo -e "
722     echo -e "
723     echo -e "
724     echo -e "\n"
725     echo -e "
726     echo -e "
727     echo -e "
728     echo -e "
729     echo -e "
730     echo -e "\n"
731     echo -e "
732     echo -e "
733     echo -e "
734     echo -e "\n"
735     echo -e "\n"
736 }
```

- ataxx_lobby 함수이다.
- id는 1p의 아이디, p2Id는 2p의 아이디를 나타내는 변수이다.
- 승수, 패수는 0,0이므로 0으로 작성하였다.

```
738 lobby_page()
739 {
740     clear
741     ataxx_lobby
742     echo -e " \033[44m START \033[0m" " \033[44m EXIT \033[0m"
743     echo -e "\n"
744 }
```

- lobby_page는 하이라이팅 되는 부분이 없는 함수이다.
- lobby_start 함수는 start가 하이라이팅 되는 함수이고, lobby_exit 함수는 exit가 하이라이팅 되는 함수이다.



3. '게임 대기 화면' - 기능 구현

```
3976 lobby_location()
3977 {
3978     count=0
3979     now_location=2
3980     while [ "$count" -ne 1 ]
3981     do
3982         move $now_location
3983         now_location="$now_location"
3984         p1_loginSuccess
3985         p1_loginSuccess=$p1_loginSuccess
3986
3987         if [ "$now_location" -eq 0 -o "$now_location" -eq 2 -o "$now_location" -eq 4 ]; then
3988             lobby_start
3989             if [ "$oper" -eq 0 ]; then
3990                 select_page
3991                 select_location
3992             fi
3993         elif [ "$now_location" -eq 1 -o "$now_location" -eq 3 -o "$now_location" -eq 5 ]; then
3994             lobby_exit
3995             if [ "$oper" -eq 0 ]; then
3996                 count=1
3997             fi
3998         fi
3999     done
4000 }
```

- now_location이 0 혹은 2 혹은 4이면, START가 하이라이팅 되도록 만든다.
- START를 enter 눌렀을 때, select_page로 넘어가고, select_location 함수를 사용하게 만든다.
- now_location이 1 혹은 3 혹은 5이면, EXIT가 하이라이팅 되도록 만든다.
- EXIT를 enter 눌렀을 때, count가 1이 되면서 프로그램이 종료되도록 만든다.

4. '맵 선택 화면' - UI 구현 및 하이라이팅 구현

```

762 map_select()
763 {
764     l=`echo -e "\033[43m  \033[0m "`
765     echo -e "
766     echo -e "
767     echo -e "
768     echo -e "
769     echo -e "
770     echo -e "
771     echo -e "
772     echo -e "
773     echo -e "
774     echo -e "
775     echo -e "\n"
776     echo -e "
777     echo -e "
778     echo -e "
779     echo -e "
780     echo -e "
781     echo -e "
782     echo -e "
783     echo -e "
784     echo -e "
785     echo -e "
786     echo -e "
787     echo -e "
788     echo -e "
789     echo -e "
790     echo -e "
791     echo -e "
792     echo -e "
793     echo -e "\n"
794 }

```

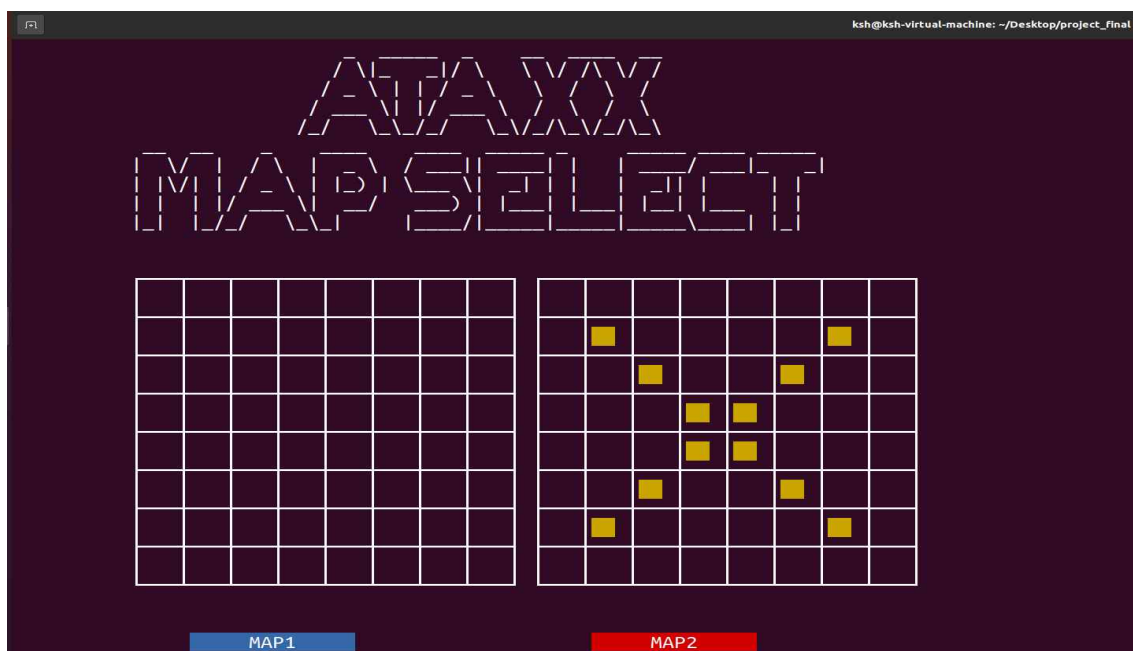
- 맵 선택 화면 UI를 구현하기 위해 map_select 함수를 만들었다.
- l 변수는 해당 위치에 노란색 하이라이팅을 사용하여 노란색 네모가 칠해지게 만들었다.

```

796 select_page()
797 {
798     clear
799     map_select
800     echo -e "
801     echo -e "\n"
802 }

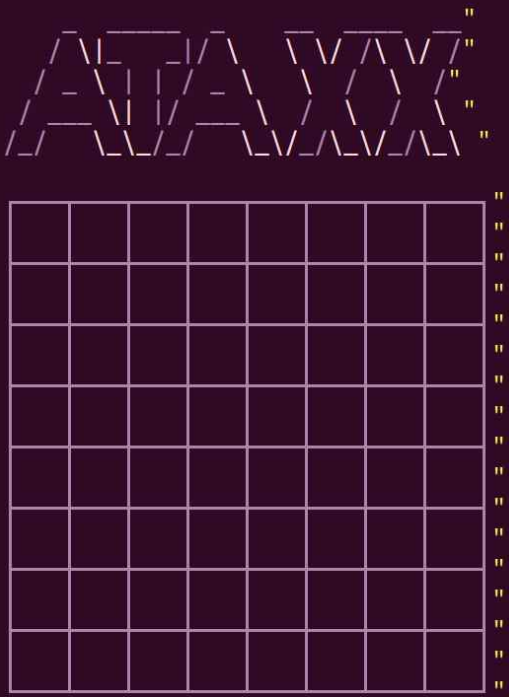
```

- select_page는 하이라이팅이 되지 않는 함수이다.
- select_map1, select_map2는 각각 MAP1, MAP2가 하이라이팅 되게 만드는 함수이다.



5. '맵 1' - UI 구현

```
820 map1_ataxx()
821 {
822     clear
823     echo -e "
824     echo -e "
825     echo -e "
826     echo -e "
827     echo -e "
828     echo -e "\n"
829     echo -e "
830     echo -e "
831     echo -e "
832     echo -e "
833     echo -e "
834     echo -e "
835     echo -e "
836     echo -e "
837     echo -e "
838     echo -e "
839     echo -e "
840     echo -e "
841     echo -e "
842     echo -e "
843     echo -e "
844     echo -e "
845     echo -e "
846     echo -e "\n"
847 }
```



The terminal output of the `map1_ataxx()` function shows the ATAXX logo at the top, followed by a 10x10 grid of empty cells. The grid is represented by a series of vertical and horizontal lines forming a 10x10 array.

- `map1_ataxx`는 MAP1을 선택했을 때, 보여지는 UI를 구현한 것이다.

```
873 map1_score()
874 {
875     echo -e "
876 }
```



The terminal output of the `map1_score()` function shows the score display. It displays "1P : \$1" and "2P : 0" on the same line.

- `map1_score`는 MAP1이 영역을 선택한 개수를 표현하는 함수이고, 이를 \$1로 받아 화면에 보이게 만들었다.



6. - 9.까지 한번에 작성했습니다.

```
2127 map1_area()  
2128 {  
2129     count=0  
2130     x_location=0  
2131     y_location=-1  
2132     area=0  
2133     arr1White=()  
2134     arr1Blue=()  
2135     change0=0  
2136     change1=0  
2137     change2=0  
2138     change3=0  
2139     change4=0  
2140     change5=0  
2141     change6=0  
2142     change7=0  
2143     change8=0  
2144     change9=0  
2145     change10=0  
2146     change11=0  
2147     change12=0  
2148     change13=0  
2149     change14=0  
2150     change15=0  
2151     change16=0  
2152     change17=0  
2153     change18=0  
2154     change19=0  
2155     change20=0  
2156     change21=0  
2157     change22=0  
2158     change23=0  
2159     change24=0  
2160     change25=0  
2161     change26=0  
2162     change27=0  
2163     change28=0  
2164     change29=0  
2165     change30=0  
2166     change31=0  
2167     change32=0  
2168     change33=0  
2169     change34=0
```

```
2170     change35=0  
2171     change36=0  
2172     change37=0  
2173     change38=0  
2174     change39=0  
2175     change40=0  
2176     change41=0  
2177     change42=0  
2178     change43=0  
2179     change44=0  
2180     change45=0  
2181     change46=0  
2182     change47=0  
2183     change48=0  
2184     change49=0  
2185     change50=0  
2186     change51=0  
2187     change52=0  
2188     change53=0  
2189     change54=0  
2190     change55=0  
2191     change56=0  
2192     change57=0  
2193     change58=0  
2194     change59=0  
2195     change60=0  
2196     change61=0  
2197     change62=0  
2198     change63=0
```

- 2135 - 2198까지 총 64칸에 맞게 64개의 변수를 만든다.
- change변수는 해당자리가 이미 색이 바뀌었다고 알려주는 변수이다.

```

2200     while [ "$count" -ne 1 ]
2201     do
2202         input_location
2203         oper="$operator"
2204
2205         if [ "$oper" -eq -2 ]; then
2206             y_location=`expr $y_location + 1`
2207         elif [ "$oper" -eq 2 ]; then
2208             y_location=`expr $y_location - 1`
2209         elif [ "$oper" -eq 1 ]; then
2210             x_location=`expr $x_location + 1`
2211         elif [ "$oper" -eq -1 ]; then
2212             x_location=`expr $x_location - 1`
2213         fi
2214
2215         if [ "$x_location" -lt -7 ]; then
2216             x_location=-7
2217         elif [ "$x_location" -gt 0 ]; then
2218             x_location=0
2219         fi
2220
2221         if [ "$y_location" -gt 7 ]; then
2222             y_location=7
2223         elif [ "$y_location" -lt 0 ]; then
2224             y_location=0
2225         fi

```

- input_location에서 받은 방향키에 따라, x_location과 y_location의 값을 알맞게 바꿔준다.
- 만약 x_location이 -7보다 작으면, x_location값을 -7로 바꾸고, 0보다 크면 0으로 바꾼다.
- 만약 y_location이 7보다 크면, y_location 값을 7로 바꾸고, 0보다 작으면 0으로 바꾼다.


```

2227         if [ "$x_location" -eq 0 -a "$y_location" -eq 0 ]; then
2228             arr1White+=(0)
2229             map1Location $arr1White $arr1Blue
2230             map1_score $area
2231             if [ "$$oper" -eq 0 ]; then
2232                 if [ "$change0" -eq 0 ]; then
2233                     area=`expr $area + 1`
2234                     arr1Blue+=(0)
2235                     change0=1
2236                 fi
2237                 map1Location $arr1White $arr1Blue
2238                 map1_score $area
2239             fi
2240         elif [ "$x_location" -eq 0 -a "$y_location" -eq 1 ]; then
2241             arr1White+=(1)
2242             map1Location $arr1White $arr1Blue
2243             map1_score $area
2244             if [ "$$oper" -eq 0 ]; then
2245                 if [ "$change1" -eq 0 ]; then
2246                     area=`expr $area + 1`
2247                     arr1Blue+=(1)
2248                     change1=1
2249                 fi
2250                 map1Location $arr1White $arr1Blue
2251                 map1_score $area
2252             fi
2253         elif [ "$x_location" -eq 0 -a "$y_location" -eq 2 ]; then
2254             arr1White+=(2)
2255             map1Location $arr1White $arr1Blue
2256             map1_score $area
2257             if [ "$$oper" -eq 0 ]; then
2258                 if [ "$change2" -eq 0 ]; then
2259                     area=`expr $area + 1`
2260                     arr1Blue+=(2)
2261                     change2=1
2262                 fi
2263                 map1Location $arr1White $arr1Blue
2264                 map1_score $area
2265             fi

```

- 2227 - 3059까지의 코드는 x_location과 y_location이 각각 0일때부터 7일때까지의 경우를 if-elif 문을 사용하여 만든 코드이다.
- 2133줄에서 선언한 arr1White 배열과, 2134줄에서 선언한 arr1Blue 배열은 색을 변경하기 위해 만든 배열이다. -> **7. 칸 색깔 변경 구현**
- 해당 위치에 커서가 있으면, arr1White 배열에 해당 위치의 숫자를 원소로 추가한다.
- enter를 누르면, arr1Blue 배열에 해당 위치의 숫자를 원소로 추가한다.
- 해당 위치의 change 변수를 변경하여 다시 돌을 돌 수 없게 만든다.
- 이런식으로, x_location이 0-7까지, y_location이 0-7일 때, 총 64개의 경우의 수를 2227 - 3059까지 작성하였다.
- area 변수는 칸 수 증가를 위해 만든 변수이다. 이 변수를 사용하여 칸 수 증가를 구현하였다. -> **8. 1P 칸 수 증가 구현**
- while 문으로 반복하고 있기 때문에, 칸 색이 변경 된 후에도 그 위치에서부터 다시 이동이 된다. -> **9. 칸 색깔 변경 후 커서 이동 구현**


```

883 map1Location()
884 {
885     clear
886
887     for white in ${arr1White[-1]}
888     do
889         if [ "$white" -eq 0 ]; then
890             w00=`echo -e "\033[47m      \033[0m" `
891         else
892             w00=`echo -e "      "`
893         fi
894         if [ "$white" -eq 1 ]; then
895             w01=`echo -e "\033[47m      \033[0m" `
896         else
897             w01=`echo -e "      "`
898         fi
899         if [ "$white" -eq 2 ]; then
900             w02=`echo -e "\033[47m      \033[0m" `
901         else
902             w02=`echo -e "      "`
903         fi
904         if [ "$white" -eq 3 ]; then
905             w03=`echo -e "\033[47m      \033[0m" `
906         else
907             w03=`echo -e "      "`
908         fi
909         if [ "$white" -eq 4 ]; then
910             w04=`echo -e "\033[47m      \033[0m" `
911         else
912             w04=`echo -e "      "`
913         fi
914         if [ "$white" -eq 5 ]; then
915             w05=`echo -e "\033[47m      \033[0m" `
916         else
917             w05=`echo -e "      "`
918         fi

```

- map1_area 함수에서 전달받은 arr1White 배열과, arr1Blue 배열을 사용하여 해당 위치의 색을 지정했다.
- arr1White[-1]은 해당 배열의 마지막 원소값을 특정하므로, 마지막 원소만 흰색으로 바뀌게 만들고, 나머지는 빈칸을 저장한다.
- 887 - 1209까지, 64개에 대한 경우를 위와 같은 방법을 사용하여 구현하였다.
- arr1White 배열 안에 있는 숫자가 무엇인지에 따라 그 숫자가 해당하는 위치의 칸의 색을 변경하였다. -> 7. 칸 색깔 변경 구현
- arr1Blue 배열 안에 있는 숫자가 무엇인지에 따라 그 숫자가 해당하는 위치의 칸의 색을 변경하였다. -> 7. 칸 색깔 변경 구현

```

1211     for blue in ${arr1Blue[@]}
1212     do
1213         if [ "$blue" -eq 0 ]; then
1214             w00=`echo -e "\033[44m    \033[0m" `
1215         fi
1216         if [ "$blue" -eq 1 ]; then
1217             w01=`echo -e "\033[44m    \033[0m" `
1218         fi
1219         if [ "$blue" -eq 2 ]; then
1220             w02=`echo -e "\033[44m    \033[0m" `
1221         fi
1222         if [ "$blue" -eq 3 ]; then
1223             w03=`echo -e "\033[44m    \033[0m" `
1224         fi
1225         if [ "$blue" -eq 4 ]; then
1226             w04=`echo -e "\033[44m    \033[0m" `
1227         fi
1228         if [ "$blue" -eq 5 ]; then
1229             w05=`echo -e "\033[44m    \033[0m" `
1230         fi
1231         if [ "$blue" -eq 6 ]; then
1232             w06=`echo -e "\033[44m    \033[0m" `
1233         fi
1234         if [ "$blue" -eq 7 ]; then
1235             w07=`echo -e "\033[44m    \033[0m" `
1236         fi
1237         if [ "$blue" -eq 8 ]; then
1238             w08=`echo -e "\033[44m    \033[0m" `
1239         fi
1240         if [ "$blue" -eq 9 ]; then
1241             w09=`echo -e "\033[44m    \033[0m" `
1242         fi
1243         if [ "$blue" -eq 10 ]; then
1244             w10=`echo -e "\033[44m    \033[0m" `
1245         fi

```

- arr1Blue 배열에 해당 위치의 숫자가 있으면, 그 위치를 파란색으로 변경한다.
- 1213 - 1404까지 if 문을 사용하여 파란색으로 변경하는 코드를 작성하였다.

```

1407     echo -e "
1408     echo -e "
1409     echo -e "
1410     echo -e "
1411     echo -e "
1412     echo -e "\n"
1413     echo -e "
1414     echo -e "
1415     echo -e "
1416     echo -e "
1417     echo -e "
1418     echo -e "
1419     echo -e "
1420     echo -e "
1421     echo -e "
1422     echo -e "
1423     echo -e "
1424     echo -e "
1425     echo -e "
1426     echo -e "
1427     echo -e "
1428     echo -e "
1429     echo -e "
1430     echo -e "\n"
1431 }

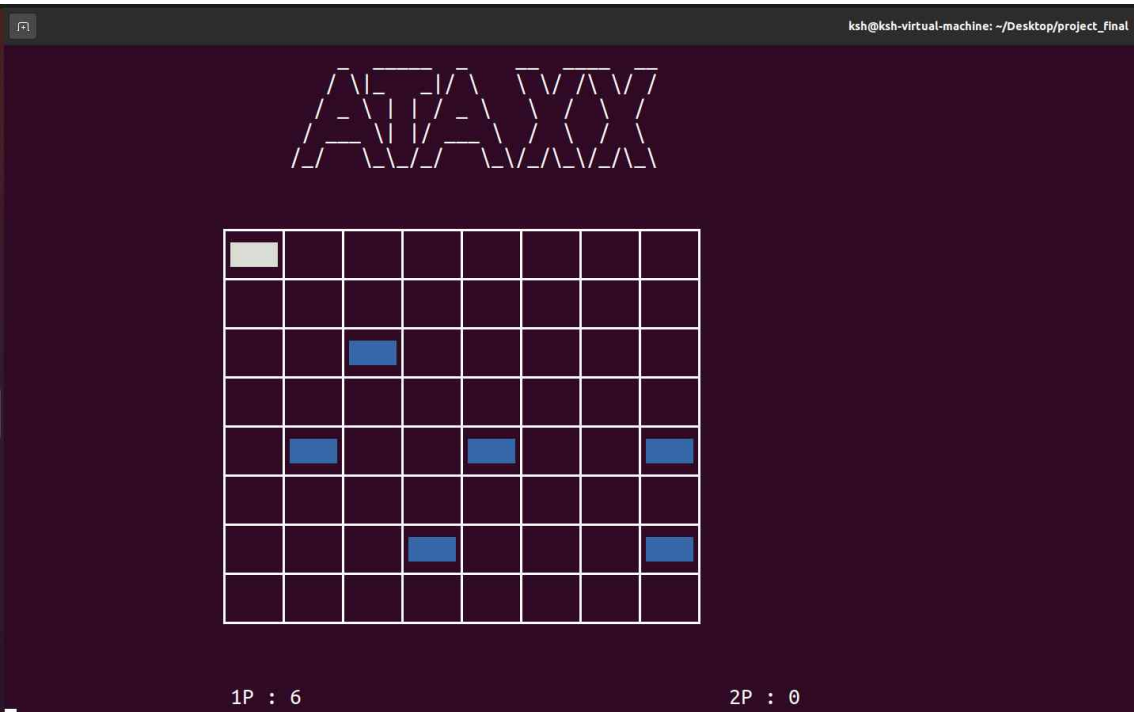
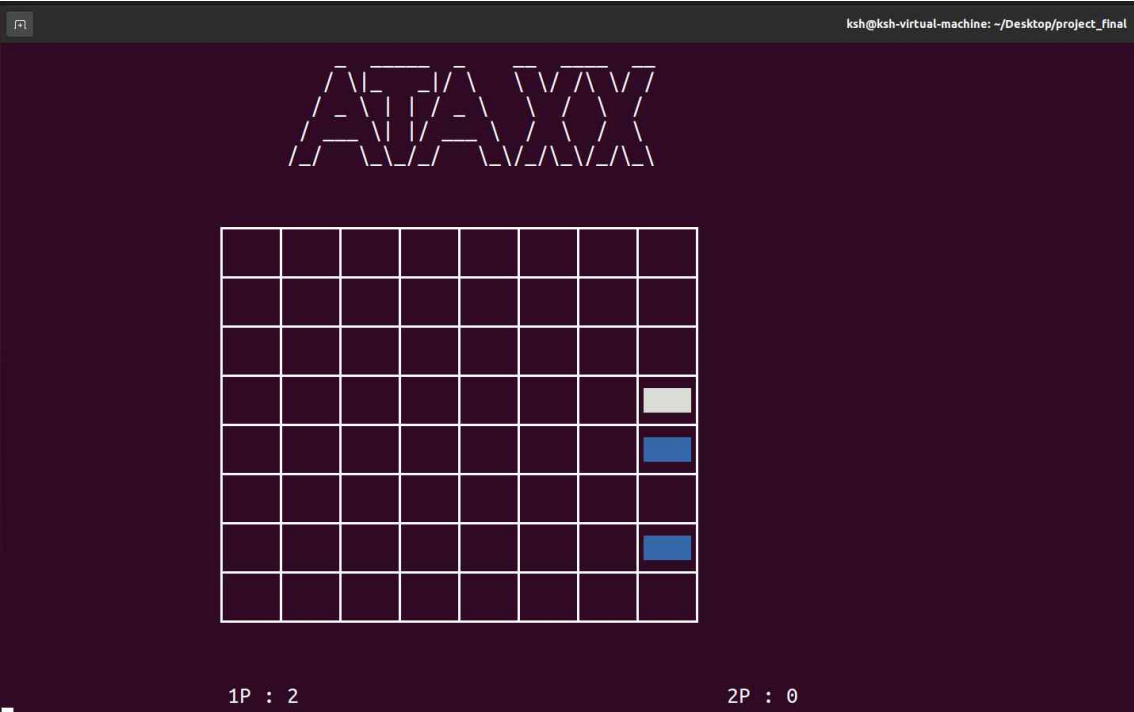
```



\$w63	\$w55	\$w47	\$w39	\$w31	\$w23	\$w15	\$w07	"
\$w62	\$w54	\$w46	\$w38	\$w30	\$w22	\$w14	\$w06	"
\$w61	\$w53	\$w45	\$w37	\$w29	\$w21	\$w13	\$w05	"
\$w60	\$w52	\$w44	\$w36	\$w28	\$w20	\$w12	\$w04	"
\$w59	\$w51	\$w43	\$w35	\$w27	\$w19	\$w11	\$w03	"
\$w58	\$w50	\$w42	\$w34	\$w26	\$w18	\$w10	\$w02	"
\$w57	\$w49	\$w41	\$w33	\$w25	\$w17	\$w09	\$w01	"
\$w56	\$w48	\$w40	\$w32	\$w24	\$w16	\$w08	\$w00	"

- w뒤에 붙는 숫자가 해당 위치의 숫자이다.

실행했을 때의 UI



10. '맵 2' - UI 구현 및 5-9번 기능 삽입 (방향키 이동 구현 시 장애물 이동 가능 및 칸 색깔 변경 불가)

```
3063 map2_area()  
3064 {  
3065     count=0  
3066     x_location=0  
3067     y_location=-1  
3068     area=0  
3069     arr2White=()  
3070     arr2Blue=()  
3071  
3072     change0=0  
3073     change1=0  
3074     change2=0  
3075     change3=0  
3076     change4=0  
3077     change5=0  
3078     change6=0  
3079     change7=0  
3080     change8=0  
3081     change9=0  
3082     change10=0  
3083     change11=0  
3084     change12=0  
3085     change13=0  
3086     change14=0  
3087     change15=0  
3088     change16=0  
3089     change17=0  
3090     change18=0  
3091     change19=0  
3092     change20=0  
3093     change21=0  
3094     change22=0  
3095     change23=0  
3096     change24=0  
3097     change25=0  
3098     change26=0  
3099     change27=0  
3100     change28=0  
3101     change29=0  
3102     change30=0  
3103     change31=0  
3104     change32=0  
3105     change33=0
```

- map1_area 함수와 똑같은 원리로 change변수를 설정하고, arr2White, arr2Blue 배열을 선언하였다.
- 3072 - 3135까지 change 변수를 선언하였다.

```

1433 map2Location()
1434 {
1435     clear
1436
1437     for white in ${arr2White[-1]}
1438     do
1439         if [ "$white" -eq 0 ]; then
1440             w00=`echo -e "\033[47m      \033[0m" `
1441         else
1442             w00=`echo -e "      "`
1443         fi
1444         if [ "$white" -eq 1 ]; then
1445             w01=`echo -e "\033[47m      \033[0m" `
1446         else
1447             w01=`echo -e "      "`
1448         fi
1449         if [ "$white" -eq 2 ]; then
1450             w02=`echo -e "\033[47m      \033[0m" `
1451         else
1452             w02=`echo -e "      "`
1453         fi
1454         if [ "$white" -eq 3 ]; then
1455             w03=`echo -e "\033[47m      \033[0m" `
1456         else
1457             w03=`echo -e "      "`
1458         fi
1459         if [ "$white" -eq 4 ]; then
1460             w04=`echo -e "\033[47m      \033[0m" `
1461         else
1462             w04=`echo -e "      "`
1463         fi

```

- map2Location의 흰색 변경은 똑같다.
- 09, 14, 18, 21, 27, 28, 35, 36, 42, 45, 49, 54번 자리는 노란색으로 채워져 있는 자리이고, 이동은 가능하지만 색변경이 불가능한 자리이다.
- 따라서, 1484, 1509, 1529, 1544, 1574, 1579, 1614, 1619, 1649, 1664, 1684, 1709번째 줄은 w변수가 아닌, l변수에 담아 화면에 출력하게 만들었다.

```

1484         if [ "$white" -eq 9 ]; then
1485             l09=`echo -e "\033[47m    \033[0m" `
1486         else
1487             l09=`echo -e "\033[43m    \033[0m" `
1488         fi

```

```

1509         if [ "$white" -eq 14 ]; then
1510             l14=`echo -e "\033[47m    \033[0m" `
1511         else
1512             l14=`echo -e "\033[43m    \033[0m" `
1513         fi

```

```

1529         if [ "$white" -eq 18 ]; then
1530             l18=`echo -e "\033[47m    \033[0m" `
1531         else
1532             l18=`echo -e "\033[43m    \033[0m" `
1533         fi

```

```

1544         if [ "$white" -eq 21 ]; then
1545             l21=`echo -e "\033[47m    \033[0m" `
1546         else
1547             l21=`echo -e "\033[43m    \033[0m" `
1548         fi

```

```

1574         if [ "$white" -eq 27 ]; then
1575             l27=`echo -e "\033[47m    \033[0m" `
1576         else
1577             l27=`echo -e "\033[43m    \033[0m" `
1578         fi

```

```

1579         if [ "$white" -eq 28 ]; then
1580             l28=`echo -e "\033[47m    \033[0m" `
1581         else
1582             l28=`echo -e "\033[43m    \033[0m" `
1583         fi

```

- 이런식으로, l변수에 담아 이동하는 칸(흰색)이 해당 숫자의 위치에 있으면, 색이 잠시 바뀌고, 없으면 노란색으로 유지한다.

```

3164         if [ "$x_location" -eq 0 -a "$y_location" -eq 0 ]; then
3165             arr2White+=(0)
3166             map2Location $arr2White $arr2Blue
3167             map2_score $area
3168             if [ "$soper" -eq 0 ]; then
3169                 if [ "$change0" -eq 0 ]; then
3170                     area=`expr $area + 1`
3171                     arr2Blue+=(0)
3172                     change0=1
3173                 fi
3174                 map2Location $arr2White $arr2Blue
3175                 map2_score $area
3176             fi
3177         elif [ "$x_location" -eq 0 -a "$y_location" -eq 1 ]; then
3178             arr2White+=(1)
3179             map2Location $arr2White $arr2Blue
3180             map2_score $area
3181             if [ "$soper" -eq 0 ]; then
3182                 if [ "$change1" -eq 0 ]; then
3183                     area=`expr $area + 1`
3184                     arr2Blue+=(1)
3185                     change1=1
3186                 fi
3187                 map2Location $arr2White $arr2Blue
3188                 map2_score $area
3189             fi
3190         elif [ "$x_location" -eq 0 -a "$y_location" -eq 2 ]; then
3191             arr2White+=(2)
3192             map2Location $arr2White $arr2Blue
3193             map2_score $area
3194             if [ "$soper" -eq 0 ]; then
3195                 if [ "$change2" -eq 0 ]; then
3196                     area=`expr $area + 1`
3197                     arr2Blue+=(2)
3198                     change2=1
3199                 fi
3200                 map2Location $arr2White $arr2Blue
3201                 map2_score $area
3202             fi

```

- map1_area 에서와 마찬가지로, arr2White, arr2Blue 배열에 원소를 추가한다.


```

3281         elif [ "$x_location" -eq -1 -a "$y_location" -eq 1 ]; then
3282             arr2White+=(9)
3283             map1Location $arr2White $arr2Blue
3284             map1_score $area
3285             if [ "$soper" -eq 0 ]; then
3286                 if [ "$change9" -eq 0 ]; then
3287                     change9=1
3288                 fi
3289                 map2Location $arr2White $arr2Blue
3290                 map2_score $area
3291             fi

```

```

3344         elif [ "$x_location" -eq -1 -a "$y_location" -eq 6 ]; then
3345             arr2White+=(14)
3346             map2Location $arr2White $arr2Blue
3347             map2_score $area
3348             if [ "$soper" -eq 0 ]; then
3349                 if [ "$change14" -eq 0 ]; then
3350                     change14=1
3351                 fi
3352                 map2Location $arr2White $arr2Blue
3353                 map2_score $area
3354             fi

```

```

3394         elif [ "$x_location" -eq -2 -a "$y_location" -eq 2 ]; then
3395             arr2White+=(18)
3396             map2Location $arr2White $arr2Blue
3397             map2_score $area
3398             if [ "$soper" -eq 0 ]; then
3399                 if [ "$change18" -eq 0 ]; then
3400                     change18=1
3401                 fi
3402                 map2Location $arr2White $arr2Blue
3403                 map2_score $area
3404             fi

```

- 노란색칸이 위치한 자리에는 enter를 눌러도 파란색으로 바뀌지 않고, 칸 수도 증가하면 안된다. 따라서, area 변수를 증가시키지 않고 arr2Blue에 원소를 추가하지 않는다.
- 따라서, 위의 코드를 사용하면 노란색칸에는 이동은 가능하지만 색변경은 불가능하게 만들 수 있다.


```

1761     for blue in ${arr2Blue[@]}
1762     do
1763         if [ "$blue" -eq 0 ]; then
1764             w00=`echo -e "\033[44m    \033[0m" `
1765         fi
1766         if [ "$blue" -eq 1 ]; then
1767             w01=`echo -e "\033[44m    \033[0m" `
1768         fi
1769         if [ "$blue" -eq 2 ]; then
1770             w02=`echo -e "\033[44m    \033[0m" `
1771         fi
1772         if [ "$blue" -eq 3 ]; then
1773             w03=`echo -e "\033[44m    \033[0m" `
1774         fi
1775         if [ "$blue" -eq 4 ]; then
1776             w04=`echo -e "\033[44m    \033[0m" `
1777         fi
1778         if [ "$blue" -eq 5 ]; then
1779             w05=`echo -e "\033[44m    \033[0m" `
1780         fi
1781         if [ "$blue" -eq 6 ]; then
1782             w06=`echo -e "\033[44m    \033[0m" `
1783         fi
1784         if [ "$blue" -eq 7 ]; then
1785             w07=`echo -e "\033[44m    \033[0m" `
1786         fi
1787         if [ "$blue" -eq 8 ]; then
1788             w08=`echo -e "\033[44m    \033[0m" `
1789         fi
1790         if [ "$blue" -eq 9 ]; then
1791             l09=`echo -e "\033[43m    \033[0m" `
1792         fi
1793         if [ "$blue" -eq 10 ]; then
1794             w10=`echo -e "\033[44m    \033[0m" `
1795         fi


```

- arr2Blue 배열에 해당 숫자가 있으면, 파란색으로 색을 변경하지만, 위에서 말한 숫자 09, 14, 18, 21, 27, 28, 35, 36, 42, 45, 49, 54번 자리는 노란색으로 화면에 출력하게 코드를 만들었다.

```

1956     echo -e "
1957     echo -e "
1958     echo -e "
1959     echo -e "
1960     echo -e "
1961     echo -e "\n"
1962     echo -e "
1963     echo -e "
1964     echo -e "
1965     echo -e "
1966     echo -e "
1967     echo -e "
1968     echo -e "
1969     echo -e "
1970     echo -e "
1971     echo -e "
1972     echo -e "
1973     echo -e "
1974     echo -e "
1975     echo -e "
1976     echo -e "
1977     echo -e "
1978     echo -e "
1979     echo -e "\n"
1980 }

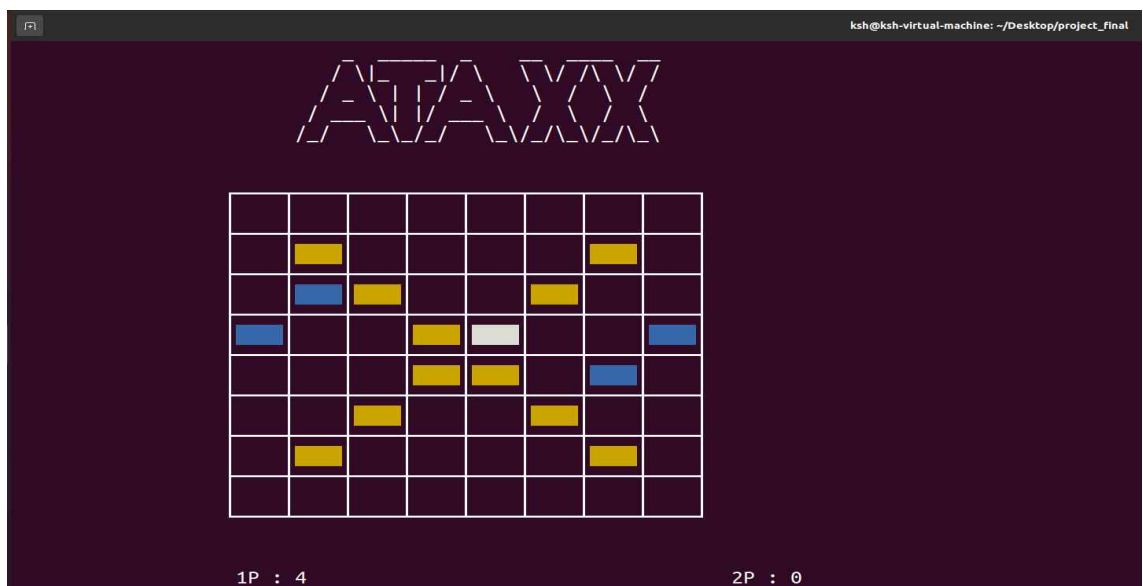
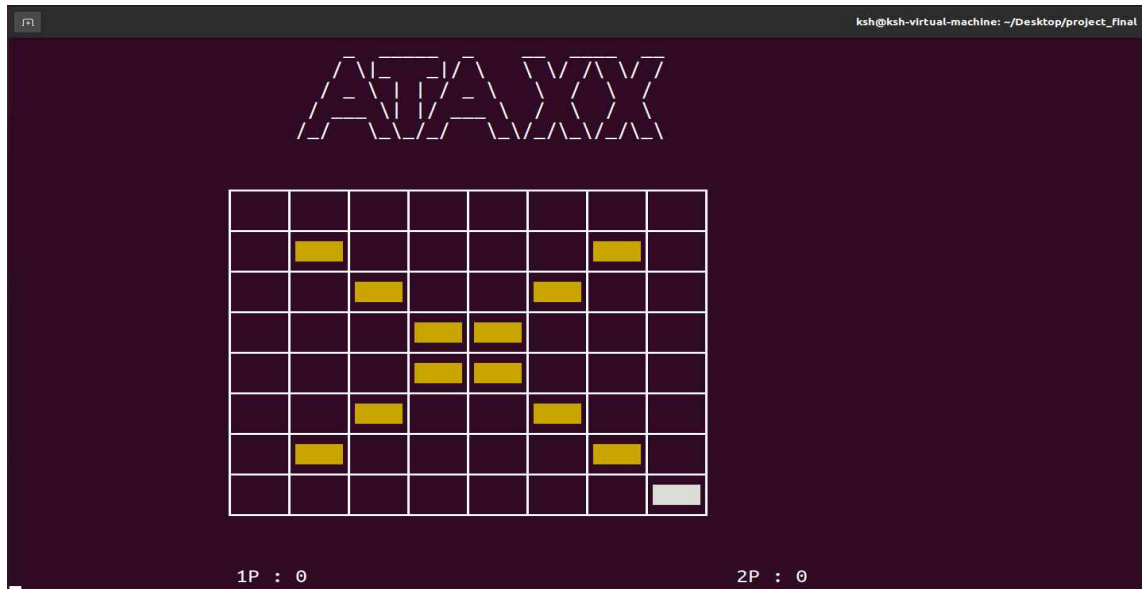
```



\$w63	\$w55	\$w47	\$w39	\$w31	\$w23	\$w15	\$w07	"
\$w62	\$l54	\$w46	\$w38	\$w30	\$w22	\$l14	\$w06	"
\$w61	\$w53	\$l45	\$w37	\$w29	\$l21	\$w13	\$w05	"
\$w60	\$w52	\$w44	\$l36	\$l28	\$w20	\$w12	\$w04	"
\$w59	\$w51	\$w43	\$l35	\$l27	\$w19	\$w11	\$w03	"
\$w58	\$w50	\$l42	\$w34	\$w26	\$l18	\$w10	\$w02	"
\$w57	\$l49	\$w41	\$w33	\$w25	\$w17	\$l09	\$w01	"
\$w56	\$w48	\$w40	\$w32	\$w24	\$w16	\$w08	\$w00	"

- 각 자리에 맞는 변수와 숫자가 위치한 코드를 볼 수 있다.

MAP2 UI 실행화면



※ 저번과제에서 했던, SIGN IN 함수를 다시 만들어서 코드와 실행화면만 따로 작성하였습니다.

※ 저번 과제에서 했던 Duplicate Check 부분등은 따로 만들지 않았습니다.

SIGN IN

```
4179 signIn_location()
4180 {
4181     count=0
4182     now_location=2
4183     change=0
4184     while [ "$count" -ne 1 ]
4185     do
4186         move $now_location
4187         now_location="$now_location"
4188
4189         if [ "$now_location" -eq 0 ]; then
4190             if [ "$soper" -eq 0 ]; then
4191                 signIn_id
4192                 read id
4193                 signIn_enterId $id
4194                 change=1
4195             else
4196                 if [ "$change" -eq 1 ]; then
4197                     signIn_enterId $id
4198                 else
4199                     signIn_id
4200                 fi
4201             fi
4202         elif [ "$now_location" -eq 1 ]; then
4203             signIn_duplicate_check
4204         elif [ "$now_location" -eq 2 -o "$now_location" -eq 3 ]; then
4205             now_location=2
4206             if [ "$soper" -eq 0 ]; then
4207                 signIn_enterPw $id
4208                 read pw
4209                 signIn_enterIdPw $id $pw
4210             else
4211                 if [ "$change" -eq 1 ]; then
4212                     signIn_enterPw $pw
4213                 else
4214                     signIn_pw
4215                 fi
4216             fi
4217         fi
4218     done
4219 }
```

- now_location이 0이면 id값을 입력받아 화면에 보여주게 만들고, id 변수에 저장한다.
- now_location이 1이면, Duplicate_check에 하이라이팅이 되게 만든다.
- now_location이 2 또는 3이면 now_location을 2로 변경한다. 왜냐면, 위, 아래 방향키를 입력받았을 때 정확한 위치로 가게 만들기 위해 변경했다.
- now_location이 2 또는 3이면 pw값을 입력받게 하였고, 입력받은 값을 화면에 보여주게 만들고, pw변수에 저장하였다.
- change변수는 id값을 입력받았는지 안받았는지 확인하는 변수이다. change가 0이면, id 값을 입력받지 않은것이고, change가 1이면 id값을 입력받은것이므로, 화면에 보여지는 결과물을 다르게 할 수 있다.

TIP.

- auth.txt파일을 만들고, login을 진행하였으므로 SIGN IN 기능을 활용하여 작동하시면 됩니다.
- 방향키로 하이라이팅을 할 때, 하이라이팅을 시작할때는 위 방향키를 눌러서 진행하시면 됩니다. (설정을 위 방향키를 눌렀을 때, 해야 원활하게 작동하게 만들었습니다.)