



Programming - Project 01

- **Anipang**



▶ Anipang

- 그래픽 인터페이스를 이용하여 애니팡 게임을 구현한다
 - 7x7의 보드로 이루어져 있으며 각각의 블록에는 5가지 문양 중 하나가 있다.
 - 블록 하나를 오른쪽, 왼쪽, 위, 아래의 4가지 방향 중 하나를 선택하여 이동할 수 있다. 블록을 이동하면 그 방향의 블록과 자리가 바뀐다.
 - 가로 또는 세로 방향으로 3개 이상의 문양이 일치하면 일치하는 문양들이 없어진다.
 - 빈칸을 채우기 위해 위쪽에 있는 문양들이 아래로 이동하며 맨 위칸을 채우기 위해서는 랜덤 문양이 내려온다.
 - 3개 이상의 일치하는 문양이 없을 때까지 위 작업을 반복한다.



▶ Anipang 기본 코드

- 주어진 그래픽 인터페이스 기본 코드를 이용하여 구현한다 (e-class에서 다운로드)
 - OpenGL 라이브러리를 이용한 인터페이스
 - 파일
 - data 폴더 – 캐릭터 이미지 파일이 저장된 폴더 (0.bmp~5.bmp)
 - bmp.cpp, bmp.h – 이미지를 읽어오는 코드
 - draw.cpp, draw.h – 화면을 그리는 코드
 - anipang.cpp, anipang.h – 게임 코드
 - *.lib, *.dll – 사용되는 라이브러리 파일
 - anipang.cpp에서 코드를 작성하면 된다



▶ Anipang 자료 구조

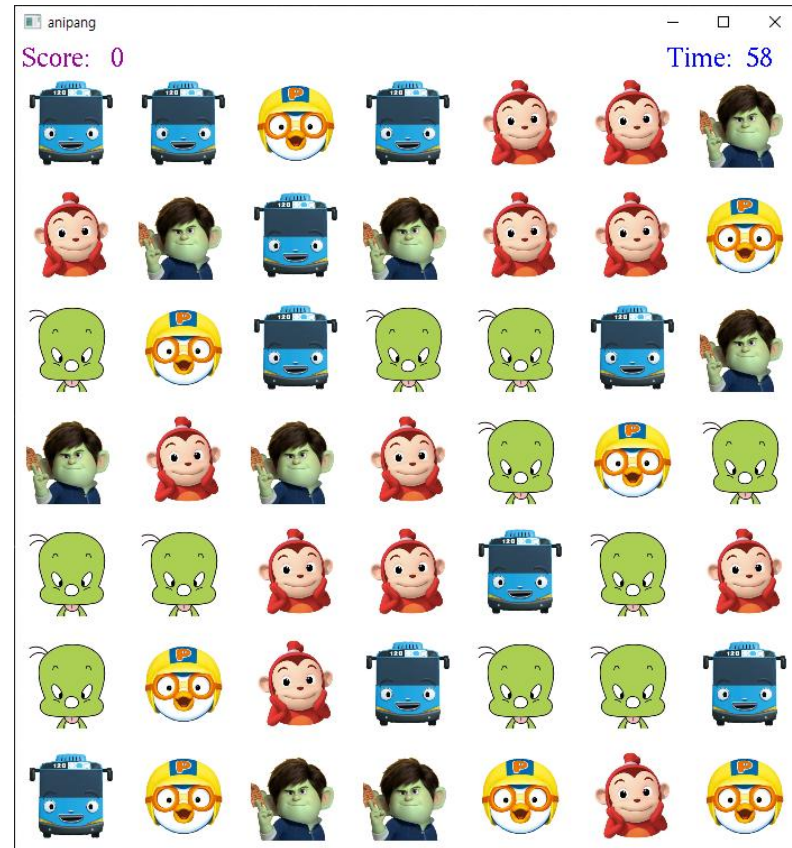
- 보드 정보를 저장하는 struct

```
typedef struct _board {  
    int tiles[YSIZE][XSIZE]; // 보드 내용을 저장  
    int xsize; // 보드의 가로 크기. 상수 XSIZE 값 사용  
    int ysize; // 보드의 세로 크기. 상수 YSIZE 값 사용  
    int nitems; // 캐릭터의 종류. 상수 NUM_ITEM 값 사용  
    int timeout; // 타임 아웃 시간 (초)  
    int time; // 게임 시간 카운트 다운. 0이 되면 게임 종료  
    int score; // 점수. 매치된 블록 개수  
} Board;
```

▶ Anipang 화면 구성

- Board의 tiles의 내용이 화면에 출력된다.
 - 코드 0은 캐릭터가 매치되어 지워진 것을 뜻하고 코드 1~5는 캐릭터를 뜻한다.

| | | | | | | |
|---|---|---|---|---|---|---|
| 3 | 3 | 2 | 3 | 4 | 4 | 5 |
| 4 | 5 | 3 | 5 | 4 | 4 | 2 |
| 1 | 2 | 3 | 1 | 1 | 3 | 5 |
| 5 | 4 | 5 | 4 | 1 | 2 | 1 |
| 1 | 1 | 4 | 4 | 3 | 1 | 4 |
| 1 | 2 | 4 | 3 | 1 | 1 | 3 |
| 3 | 2 | 5 | 5 | 2 | 4 | 2 |



▶ Anipang 화면 구성

- 화면 좌표와 2차원 배열(Board.tiles)과의 관계

| | | | | |
|-------|-------|-------|-----|-------|
| (0,0) | (1,0) | (2,0) | ... | (N,0) |
| (0,1) | (1,1) | ... | | ... |
| (0,2) | ... | | | |
| ... | | | ... | ... |
| (0,M) | ... | | ... | (N,M) |

(x, y) 화면 좌표

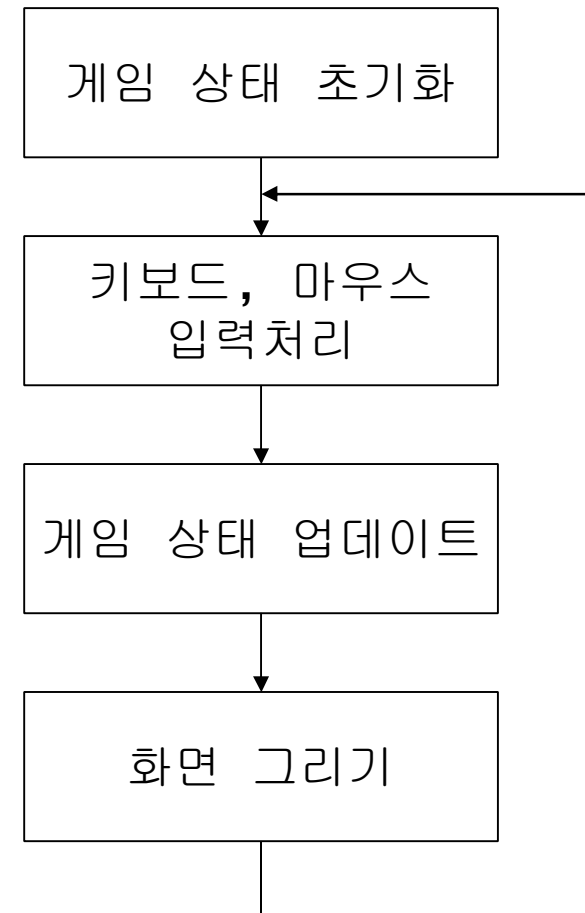
| | | | | |
|--------|--------|--------|-----|--------|
| [0][0] | [0][1] | [0][2] | ... | [0][N] |
| [1][0] | [1][1] | ... | | ... |
| [2][0] | ... | | | |
| ... | | | ... | ... |
| [M][0] | ... | | ... | [M][N] |

`tiles[i][j]` 2차원 배열 인덱스

▶ Anipang 게임 로직

■ 기본 구조

- 맨 처음 상태를 초기화 해야 한다.
 - **gameInit()** 함수에서 상태가 초기화되며 **mkBoard()** 함수를 이용하여 보드를 초기화 한다.
 - 보드를 초기화할 때 3개 이상 연속된 문양이 없도록 코드 작성 필요
- 무한 루프가 반복되고 있으며 입력 처리, 상태 업데이트, 화면 그리기가 반복된다.
- 마우스를 드래그하면 클릭된 블록이 드래그된 방향으로 이동되어야 한다.
 - **mouseMotion()** 함수가 호출되며 이를 작성해야 한다.





▶ Anipang 게임 로직

- `void mouseMotion(Board *board, int tile[2], int move[2]);`
 - `board`: 보드 정보가 저장된 `struct` 변수 포인터
 - `tile`: 마우스가 클릭된 위치의 `x, y` 화면 좌표
 - `tile[0]`이 `x`, `tile[1]`이 `y`이며 인덱스로 사용할 때는 `tiles[y][x]`가 되어야 한다
 - `move`: 마우스의 드래그 방향
 - `move[0]`이 `x`, `move[1]`이 `y`이다
 - `(1, 0)`, `(-1, 0)`, `(0, 1)`, `(0, -1)` 중 하나의 값이다.



▶ Anipang 게임 로직

■ mouseMotion() 로직

1. tile위치의 블록을 move방향으로 이동하여 그 방향의 블록과 바꾼다 (swap)
2. 3개 이상의 일치하는 블록이 있으면 블록을 지운다 (0 할당). 지운 개수에 따라 점수가 증가한다.
3. 블록을 아래로 이동하여 빈 칸을 채운다. 맨 위의 빈 칸은 랜덤 블록으로 채운다.
4. 재배치된 보드에서 일치하는 블록이 있으면 위 2~3을 반복한다.
5. 일치하는 블록이 처음부터 하나도 없으면 1의 swap을 취소한다.

■ 시각적 피드백을 위해 위 과정을 화면에 보여준다


- 보드 내용이나 점수가 바뀌면 display()함수를 호출하여 그려준다
- 블록 변화가 너무 빠르면 잘 안보이므로 Sleep() 함수로 적당히 기다린다

▶ Anipang 게임 로직

■ 특수 패턴

□ 가로 또는 세로로 4개 이상 일치하면 해당 줄을 모두 지운다

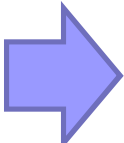
| | | | | | | |
|-------|---|---|---|---|---|---|
| 3 | 3 | 2 | 3 | 4 | 4 | 5 |
| 4 | 5 | 5 | 5 | 5 | 4 | 2 |
| 1 | 2 | 3 | 1 | 1 | 3 | 5 |
| | | | | | | |



| | | | | | | |
|-------|---|---|---|---|---|---|
| 3 | 3 | 2 | 3 | 4 | 4 | 5 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 2 | 3 | 1 | 1 | 3 | 5 |
| | | | | | | |

□ 가로와 세로가 만나면 해당 직사각형을 모두 지운다

| | | | | | | |
|-------|---|---|---|---|---|---|
| 3 | 3 | 5 | 3 | 4 | 4 | 5 |
| 4 | 5 | 5 | 5 | 1 | 4 | 2 |
| 1 | 2 | 5 | 1 | 1 | 3 | 5 |
| | | | | | | |



| | | | | | | |
|-------|---|---|---|---|---|---|
| 3 | 0 | 0 | 0 | 4 | 4 | 5 |
| 4 | 0 | 0 | 0 | 1 | 4 | 2 |
| 1 | 0 | 0 | 0 | 1 | 3 | 5 |
| | | | | | | |

▶ Anipang 화면 출력 예제

- t를 누르면 예제 함수 testRemove()가 실행된다

```
void testRemove(Board *board)
{
    int i, k, n = 3;

    for (k = 0; k < n; k++)
        board->tiles[5 - k][5] = 0; // 값이 0이면 폭발그림이 출력
    display(); // 화면에 변화된 보드를 그린다
    sleep(500); // 너무 빠르면 과정이 안보이므로 적당히 기다린다
    for (i = 0; i < n; i++) {
        for (k = 5; k > 0; k--) {
            board->tiles[k][5] = board->tiles[k - 1][5];
        }
        board->tiles[0][5] = rand() % board->nitems + 1;
        display(); // 화면에 변화된 보드를 그린다
        sleep(100); // 너무 빠르면 과정이 안보이므로 적당히 기다린다
    }
}
```