작품 제목: Quick Quick Find!

비디오 업로드 주소: https://youtu.be/fmAS6Fsrhk0

학번: 2014707040

이름: 유진혁

1. 작품 소개

<Quick Quick Find!>는 마우스 클릭을 이용한 간단한 인터렉티브 게임이다. 우측의 노란색 PLAY 버튼을 클릭하면 게임이 시작되고 타이머가 구동된다. 표시된 중앙 상단의 카드를 보고 같은 모양의 카드를 아래 나열된 카드 안에서 찾아내면 된다. 같은 모양의 카드를 클릭하면 새로 찾아야 할 카드가 가운데 상단에 표시된다. 모든 카드를 다 찾아내면 Game Over! 문구와 함께 타이머가 멈추고 사용자의 기록이 표시된다. 해당 기록이 최고 기록(최단 시간)일 경우 저장되어 다음 플레이 시에도 우측 상단에 출력된다.

2. 필수 요구 사항 구현 내용 설명

A. 마우스 혹은 키보드 (혹은 둘 다)의 입력에 따라 반응하는 프로그램이어야 함.

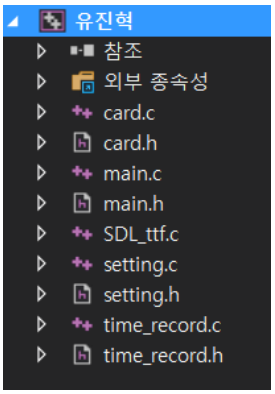
```
else if(e.type == SDL_MOUSEBUTTONDOWN)
{
    int x, y;
    SDL_GetMouseState( &x, &y );

    // Check if mouse is in button
    for(i = 0; i < SIZE; i++)
    {
        if (checkMouseInRect(playRect, x, y))
        {
            if (game == READY)
            {
                deck[match].visible = VISIBLE;
                for (i = 0; i < SIZE; i++)
                    cards[i].visible = VISIBLE;
            }
            game = PLAYING;
            SDL_DestroyTexture(play);
        }

        if (checkMouseInRect(cards[i].rect, x, y))
        {
            if ((deck[match].cardNum == cards[i].cardNum) && (cards[i].visible == VISIBLE))
            {
                cards[i].visible = INVISIBLE;
                if (i < SIZE / 2)
                {
                    cards[i + SIZE/2].rect.x = cards[i].rect.x;
                    cards[i + SIZE/2].rect.y = cards[i].rect.y;
                }
                deck[match++].visible = INVISIBLE;
                if (match != SIZE)
                    deck[match].visible = VISIBLE;
                count--;
```

마우스 클릭 이벤트로 클릭 좌표에 따라 알맞은 작업을 수행한다. PLAY 버튼 위에서 클릭할 경우 게임을 PLAYING 상태로 바꾸고 버튼을 지운다. 찾아야 할 카드 위에서 클릭할 경우 해당 카드를 숨기고 다음 카드를 표시한다.

B. 여러 개의 .h 및 .c 파일에 나누어 코드를 작성해야 함.



카드 관련 함수/변수는 card.c와 card.h에, 타이머 및 기록 관련 함수/변수는 time_record.c와 time_record.h에, 게임 초기화 및 설정에 관련된 함수/변수는 setting.c와 setting.h에 나눠 작성했다. main.h에는 모든 파일에서 필요로 하는 헤더파일과 전역변수, 열거형 등을 담고 있고, main.c에는 실제 게임 구동 코드가 작성돼 있다. SDL_ttf.c는 텍스트 출력을 위한 오픈 소스이다.

C. main() 함수를 제외하고 1개 이상의 함수를 정의하여 사용

```c
/* 마우스 좌표가 Rect 내부 인지 확인 */
int checkMouseInRect(SDL_Rect rect, int x, int y)
{
    if(x < rect.x)
        return 0;
    else if(x > rect.x + rect.w)
        return 0;
    else if(y < rect.y)
        return 0;
    else if(y > rect.y + rect.h)
        return 0;

    return 1;
}
```

```c
/* 카드 섞음 */
void shuffle(Card cards[], int start, int end)
{
    int temp, i;
    for (i = 0; i < (end - start); i++)
    {
        int n = rand() % (end - start) + start;

        temp = cards[n].cardNum;
        cards[n].cardNum = cards[end - 1 - i].cardNum;
        cards[end - 1 - i].cardNum = temp;
    }
}
```

마우스 좌표가 Rect 내부인지 확인하는 checkMouseInRect 함수, 인자로 받은 카드 배열을 임의로 섞는 shuffle 함수를 정의했다. 이 외에도 시간을 증가시키는 timeElapse 함수, 모든 이미지를 불러오는 loadAllImage 함수 등 다수의 함수를 정의하여 사용하였다.

D. struct를 이용하여 1개 이상의 자료형을 정의하여 사용

```c
typedef struct
{
    SDL_Rect rect;
    Visible visible;
    int cardNum;
} Card;
```

Card 구조체를 정의하여 사용했다. Card 구조체 안에는 카드의 크기와 위치를 담는 SDL_Rect형 변수 rect, 해당 카드의 표시 정보는 담는 Visible 열거형 변수 visible, 해당 카드가 어떤 모양의 카드인지를 나타내는 int형 변수 cardNum가 있다.

E. enum을 이용하여 1개 이상의 자료형을 정의하여 사용

```c
typedef enum { VISIBLE, INVISIBLE } Visible;
typedef enum { READY, PLAYING, OVER } State;
```

화면에 표시할지 숨길지를 결정하는 Visible, 현재 게임의 상태를 표시하는 State 자료형을 enum으로 정의하여 사용했다.

F. #define을 이용하여 1개 이상의 매크로 상수를 정의하여 사용

```
#define ROW 6
#define COL 9
#define SIZE (ROW*COL)
```

```
#define MSEC 0.016
```

나열된 카드의 행과 열의 수를 각각 ROW와 COL 매크로 상수로 정의했고, 이 둘을 곱한 grid의 크기를 SIZE라는 매크로 상수로 정의하여 사용했다. 또, 시간 증가율 단위를 MSEC 매크로 상수로 정의하여 타이머에서 사용하였다.

G. 1개 이상의 1차원 배열을 선언하여 사용

```
Card cards[SIZE];
Card deck[SIZE];
```

아래 나열될 카드들과 찾아야 할 카드를 각각 1차원 Card 구조체 배열로 선언하여 사용했다. 나열될 카드의 배열을 cards로, 찾아야할 카드의 배열을 deck으로 하여 매크로 상수 SIZE 크기로 정의하였다.

3. 선택 요구 사항 구현 설명

A. 1개 이상의 변수 혹은 배열을 동적으로 할당하여 사용

(구현하지 않음.)

B. 1개 이상의 2차원 배열을 선언하여 사용

(구현하지 않음.)

C. 강의 시간에 다루지 않은 하나 이상의 SDL의 기능을 사용

```
/* 텍스트를 텍스쳐로 불러오기 */
SDL_Texture* loadText(const char* msg, const char* fontFile, SDL_Color color, int fontSize, SDL_Renderer *ren)
{
    TTF_Font *font = TTF_OpenFont(fontFile, fontSize);
    if (font == NULL) {
        printf("TTF_OpenFont₩n");
        printf("TTF_OpenFont: %s₩n", TTF_GetError());
        return NULL;
    }

    SDL_Surface* surf = TTF_RenderText_Blended(font, msg, color);
    if (surf == NULL) {
        TTF_CloseFont(font);
        printf("TTF_RenderText₩n");
        return NULL;
    }

    SDL_Texture* tex = SDL_CreateTextureFromSurface(ren, surf);
    if (tex == NULL) {
        printf("CreateTexture₩n");
    }

    SDL_FreeSurface(surf);
    TTF_CloseFont(font);
    return tex;
}
```

ttf 폰트를 가져와 이를 이용해 텍스트를 쓰고 출력하는 기능을 사용했다. 이는 오픈소스로 제공된 코드를 활용했다. 이 텍스트 출력 기능으로 화면에 타이머와 최고 기록 숫자가 텍스트 형태로 출력되도록 하였다.

D. 파일 입출력을 이용해 game 진행상황을 save / load하는 기능 구현

```
/* 최고 기록 불러오기 */
void loadRecord()
{
    FILE* fp = fopen("record.txt", "rt");
    if (fp == NULL)
    {
        recordString[0] = '0';
        recordString[1] = '.';
        recordString[2] = '0';
        recordString[3] = '0';
        recordString[4] = '₩0';
    }
    else
    {
        fscanf(fp, "%s", recordString);
        fclose(fp);
    }
}
```

```
/* 최고 기록 저장 */
void saveRecord()
{
    if (timeValue < atof(recordString) || atof(recordString) == 0)
    {
        FILE* fp = fopen("record.txt", "wt");
        fprintf(fp, "%.2f", timeValue);
        fclose(fp);
    }
}
```

파일 입출력을 이용해 사용자의 최고 기록 저장하고 불러오는 함수를 정의하여 사용했다. loadRecord는 파일을 열고 해당 파일이 존재하지 않으면 최고 기록을 0.00으로, 존재한다면 읽어들여 문자열에 저장한 뒤 파일을 닫는 함수이다. saveRecord는 현재 기록이 최고 기록일 때 파일을 열고 현재 기록을 쓴 뒤 닫는 함수이다.

4. 그 밖의 본인의 작품에서 추가로 강조하여 설명하고 싶은 부분 (optional)

깔끔하고 직관적인 UI 그래픽 디자인