대한상공회의소
서울기술교육센터

ARM Architecture

Mini Project

[두근두근 광물 캐기 게임ㅅ]

과정명	AI 시스템 반도체 설계 2기
제작	최현우

Game Description

게임 설명

[게임 목표]

- 경비와 종유석을 피해 광물들을 채광하세요!
- 광물 등급 별로 채광 시간과 획득 점수가 다릅니다
- 각 Stage마다 모든 광물을 채광해 Game을 Clear하세요!

Gold
- 채광 시간: 1초
- 획득 점수: 1점

Emerald
- 채광 시간: 3초
- 획득 점수: 2점

Diamond
- 채광 시간: 5초
- 획득 점수: 3점

사용 장비 & 툴

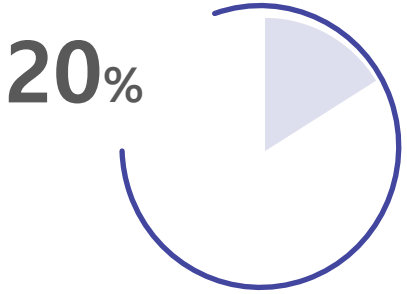
[사용 장비]

- M3 MINI Board
- Cortex-M3 기반(STM32F10 series)

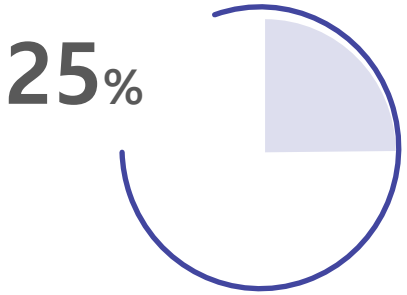
[개발 환경]

- Editor: VSCODE
- Emulator: Tera Term
- Logic Analyzer

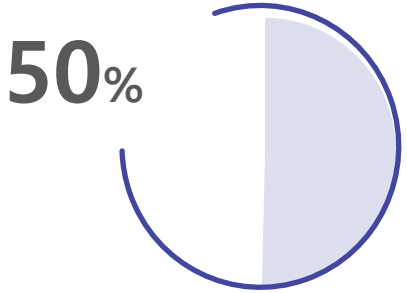
Development Schedule



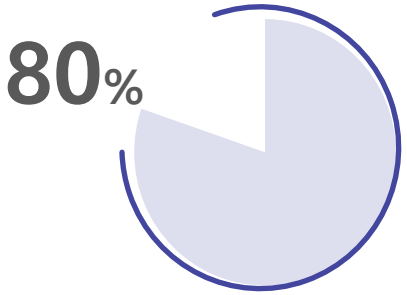
4/25
게임 아이디어 기획



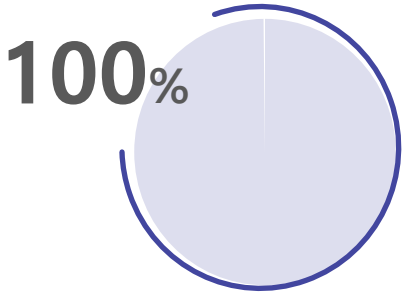
4/28
LCD 드라이버 분석



4/29
Stage1 개발



5/1
최종 단계 개발



5/2
버그 해결 및 검증

Result Stage 1

스테이지 1

[설명]

- 1개의 경비 픽셀 등장
- 경비 픽셀과 충돌 시, Crash! 메시지와 함께, 플레이어 체력 차감

#채광 메시지

```
MINING GOLD
Mine Gold! +1point
Score: 1
```

#Clear 메시지

```
Congratulations! total Score = 6
Stage 1 Clear! Press any key to continue
```

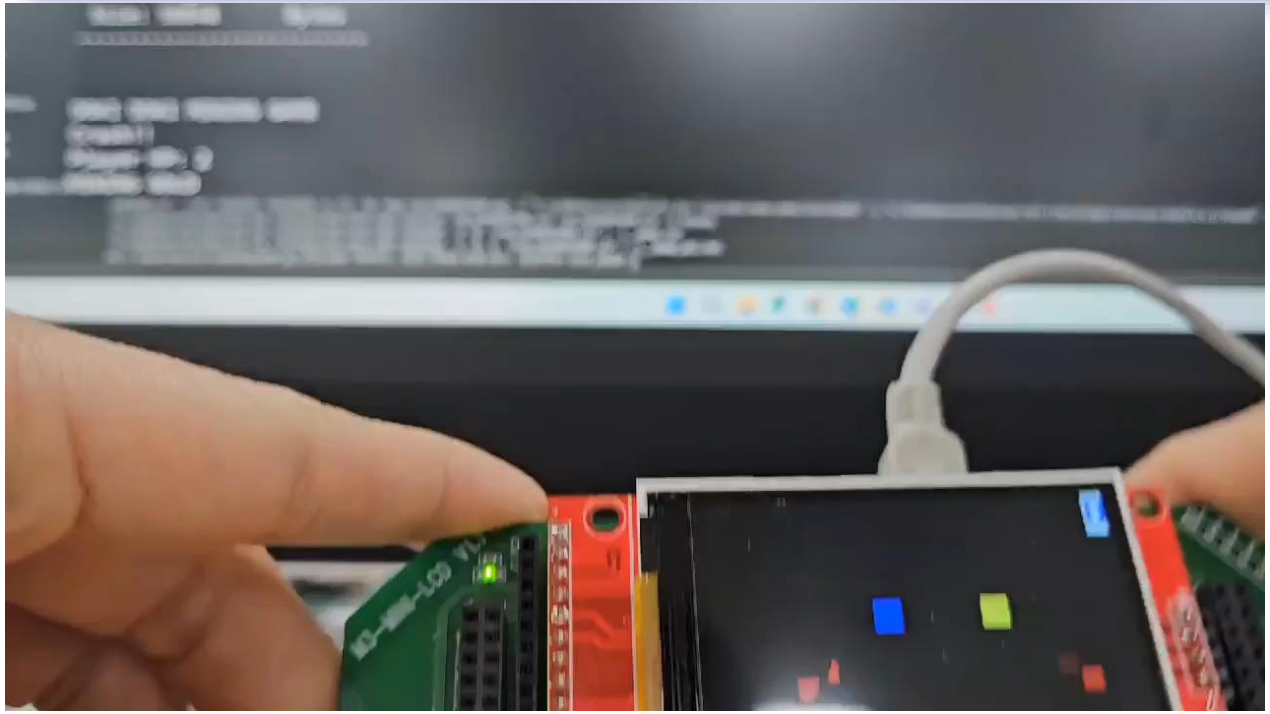
#충돌 메시지

```
Crash!!
Player HP: 2
```

참고)

- 시연을 위해 채광 시간을 1초로 통일하였습니다
- Game over 옵션을 꺼놓았습니다(충돌 시 무적)

Result Stage 2



스테이지 2

[설명]

- 2개의 경비 픽셀 등장
- 1단계보다 크기는 작으나 이동속도 증가

#채광 메시지

```
MINING GOLD
Mine Gold! +1point
Score: 1
```

#Clear 메시지

```
Congratulations! total Score = 6
Stage 2 Clear! Press any key to continue
```

#충돌 메시지

```
Crash!!
Player HP: 2
```

Result Stage 3

스테이지 3

[설명]

- 3개의 경비 픽셀 등장
- 2단계에 비해 크기 및 이동속도 증가
- 종유석 등장
  - 2~5초 사이에 랜덤한 시간에 등장
  - 랜덤한 위치에서 발생

#채광 메시지

```
MINING GOLD
Mine Gold! +1point
Score: 1
```

#Clear 메시지

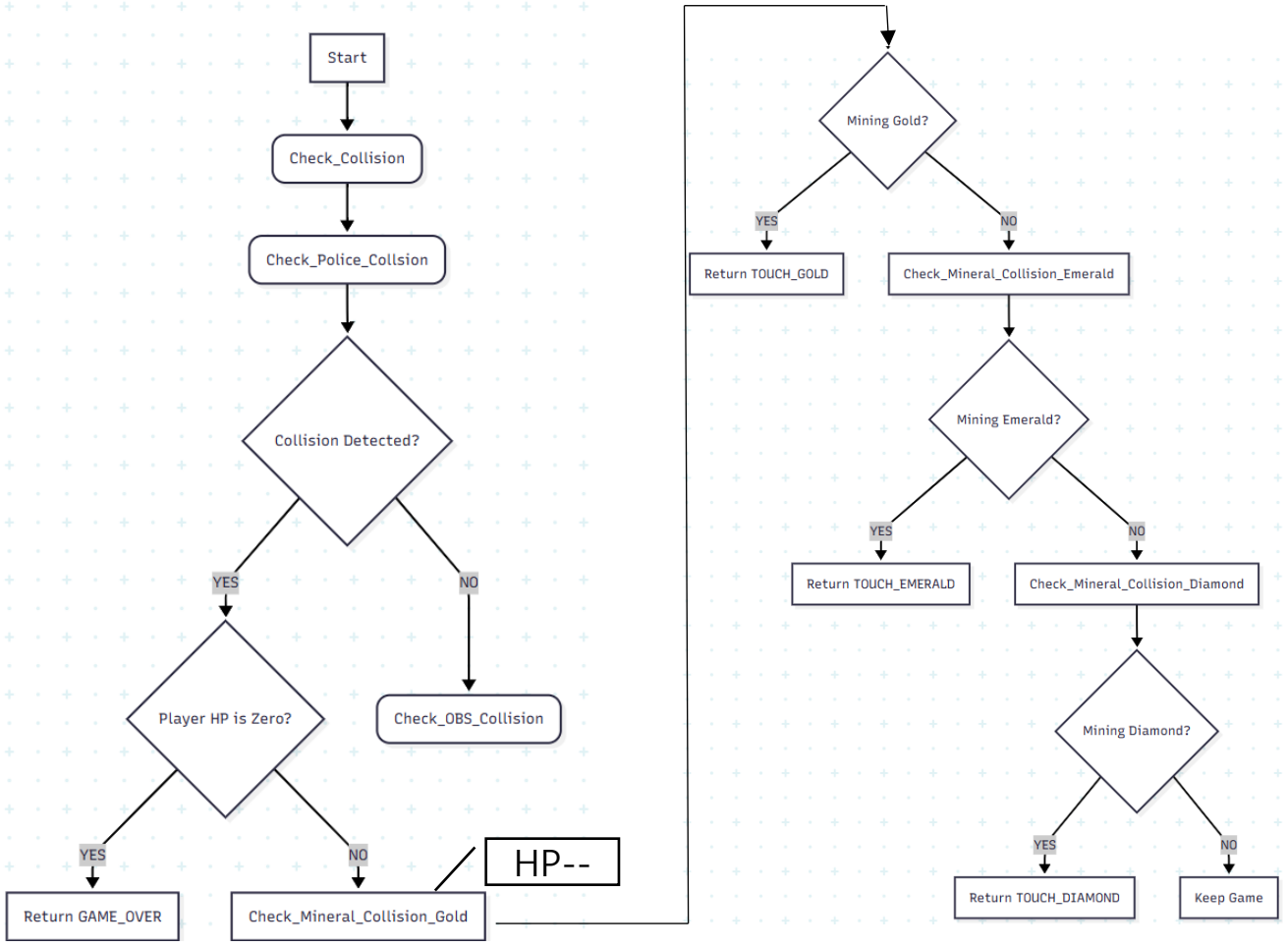
```
Congratulations!
You Clear The Game!
Total Score = 6
Press any key to restart
```

#충돌 메시지

```
Crash!!
Player HP: 2
```

Main Idea

#충돌 판정 알고리즘



흐름도

1. Police, Player, Obstacle(종유석)이 이동할 때마다 충돌 판정 함수를 동작

2. Police Pixel → 종유석 Pixel → 광물 Pixel 순으로 충돌 여부 판단

3. Police Pixel 혹은 종유석 픽셀과 충돌 시, col 변수 3 대입

- x축 충돌: col |= (1<<0)
- y축 충돌: col |= (1<<1)

4. 충돌 발생 시, Player 체력 차감

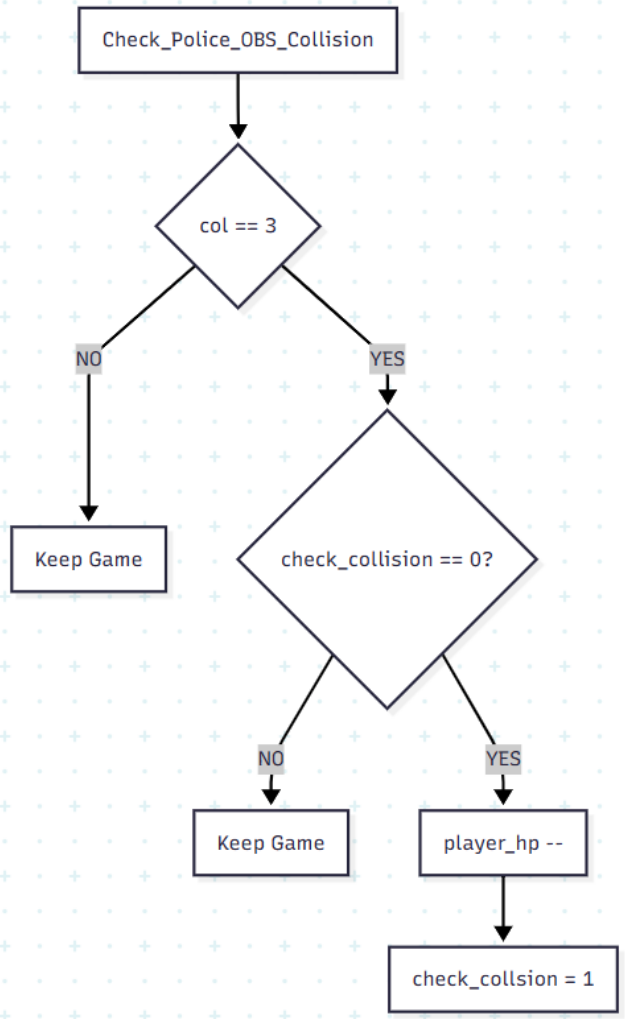
- 변수 player_hp가 0일 때 col 플래그 켜지면 GAME_OVER Flag를 세움

5. 장애물 미 충돌 시, 광물 충돌 함수 동작

- 각 광물 별 충돌 시, 해당 광물 Mining flag 발생

Main Idea

#연속 충돌 판정 알고리즘



#Problem

[문제점]

- 처음 충돌 때만 플레이어의 체력을 깎아야 함
- 플레이어 픽셀과 장애물 픽셀이 충돌하는 매 틱마다 체력이 깎임

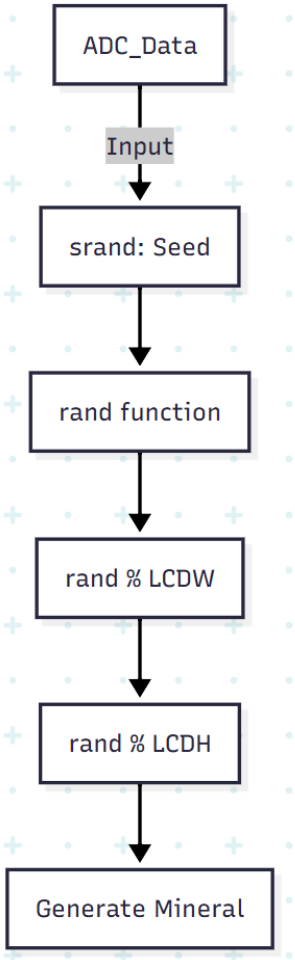
#Solution

[해결방법]

- check_collsion Flag 사용
- 처음 충돌 시, Flag를 띄움
- 이후, 충돌 판정 알고리즘에서 col 플래그가 0일 시
- Flag를 내림

Main Idea

#광물 랜덤 발생 알고리즘



#Problem

[문제]

- 랜덤한 위치에 광물 픽셀이 생성돼야함
- 생성할 때마다 시드 값이 바뀌어, 생성 패턴이 반복되어선 안됨
- 생성 픽셀이 LCD 화면 안에 존재해야 함

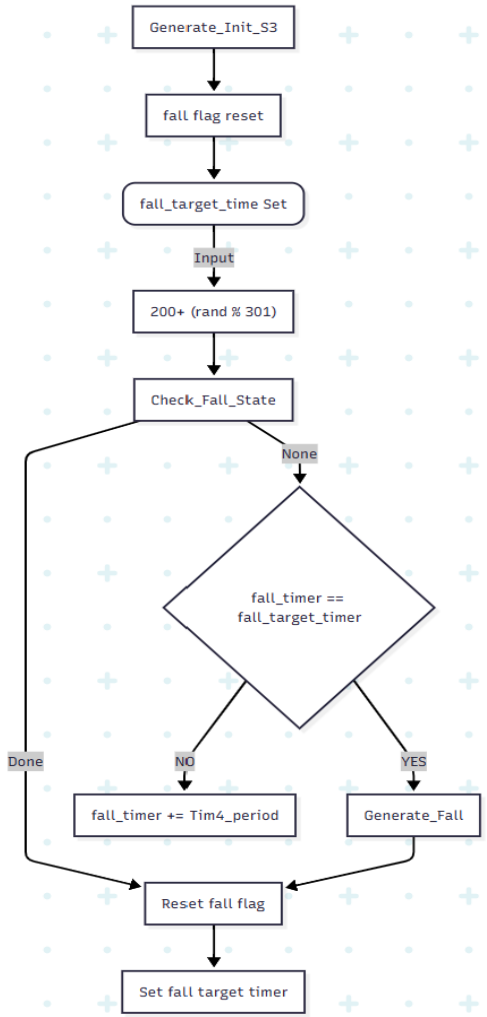
#Solution

[해결방법]

- Srand의 Seed값으로 ADC Data 이용
- Generate_Mineral 함수가 동작할 때마다 새로운 Seed 값 생성
- Rand() 함수를 통해 생성된 좌표를 LCD의 Size로 나누어, 생성 좌표를 제한함

Main Idea

#종유석 랜덤 발생 알고리즘



#Problem

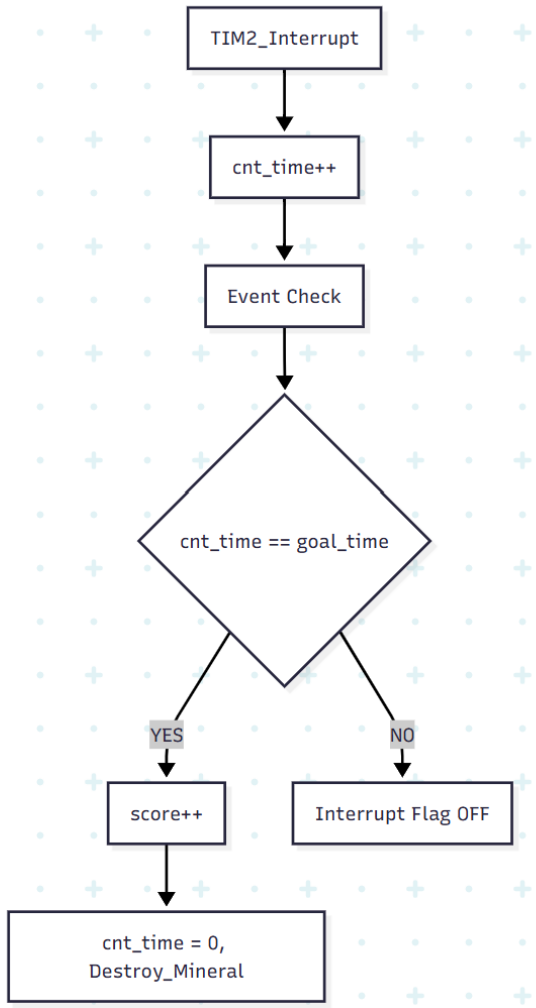
[문제]
- 랜덤한 위치에 종유석이 생성돼야함
- 랜덤한 시간(2~5초 사이)에 생성돼야함

#Solution

[해결방법]
- 종유석 랜덤 생성을 위해 Fall flag 이용
  - Fall_active: 현재 종유석이 생성됐는지
  - Fall_timer: 현재 생성 타이머의 값
  - Fall_target_timer: 종유석이 생성될 시간값
- Fall_target_timer값에 종유석이 생성되는 시간값 설정
  - 해당 시간이 되면, 종유석 발생
- Fall_active가 OFF이면, fall_timer에 Timer4 주기를 누적
- Fall_timer == Fall_target_timer일 때 종유석 생성

Main Idea

채굴 시간 측정 알고리즘



#Problem

[문제]
- 광물 등급 별, 채굴 시간 측정 필요
- 목표 채굴 시간 도달 시, 광물 제거 및 점수 획득

#Solution

[해결방법]
- Timer2 인터럽트 이용
- Timer2를 Repeat모드로 1초마다 인터럽트 발생 설정
- 인터럽트 발생시, cnt_time을 증가시키고, 인터럽트 플래그 발생
- Event driven 방식으로, 메인함수 내에서, cnt_time이 해당 광물의 채광 시간과 일치하는지 판별
- 채광 시간과 일치 시, 광물 픽셀 제거 후, 점수 획득

Main Code

#1. 종유석 랜덤발생 소스코드

```
static void Game_Init_S3(void)
{
    player_hp = 3;
    score = 0;
    cnt_time = 0;
    fall_active = 0;
    fall_timer = 0;
    fall_target_time = 200 + (rand() % 301); // 2~5초 랜덤 초기값
```

```
if (stage == 3)
{
    // 종유석이 이미 떨어지고 있는 경우
    if (fall_active)
    {
        fall.ci = BACK_COLOR;
        fall2.ci = BACK_COLOR;
        Draw_Object_O(&fall); // 이전 위치 지우기
        Draw_Object_O(&fall2); // 이전 위치 지우기
        check_state = OBS_Move(stage); // Y 위치 증가 + 충돌 확인
        if(check_state == GAME_OVER) game_over = 1;
        fall.ci = OBS_COLOR;
        fall2.ci = OBS_COLOR;
        Draw_Object_O(&fall); // 새 위치 그림
        Draw_Object_O(&fall2); // 새 위치 그림
```

[1. 초기화]

- 종유석 생성을 위한 Flag 설정
- Fall_active: 종유석 활성 여부
- Fall_timer: 종유석 생성까지 측정 시간 값
- Fall_target_time: 종유석 생성 시간
    - 기본 2초(200)에 (rand()%3)을 더해 2~5초 사이 랜덤한 시간에 종유석이 생성되도록 함

[2. fall_active]

- Fall_active 활성 시, 구조체 fall object를 LCD상에 그려줌
- 이때, 이전에 위치했던 종유석은 지워야하므로 색상을 black으로 바꾼 뒤 그림을 clear 해준 후, 현재 위치에 색상을 가진 종유석을 그려줌

Main Code

#1. 종유석 랜덤발생 소스코드

```
// 종유석이 사라졌다면 타이머 리셋
if (fall.y + fall.h == 0)
{
    fall_active = 0;
    fall_timer = 0;
    fall_target_time = 200 + (rand() % 301);  // 2000~3000ms
}
```

[4. 제거]

- 종유석이 마지막 y좌표에 도달하여, 좌표가 초기화 되었을 때 종유석 Flag를 초기화해줌
- Fall_target_timer는 2~5초 사이의 랜덤 시간이어야하므로 200+(rand() % 3)을 대입해줌

```
else
{
    // 종유석 비활성 상태일 때 대기 시간 누적
    fall_timer += TIMER_PERIOD;  // TIM4 인터럽트 주기만큼 증가
```

[5. 타이머 누적]

- 종유석이 비휠성 상태 시, fall_timer(현재 시간값)에 Timer4의 주기를 누적하여 더함

```
if (fall_timer >= fall_target_time)
{
    Generate_Fall(); // 새 종유석 생성
    Generate_Fall2(); // 새 종유석 생성
}
```

[6. 종유석 생성]

- Fall_timer(현재 시간)값이 fall_target_timer(목표시간)보다 크거나 같으면 종유석 픽셀을 생성함
- Generate_Fall 함수 내에서, fall_active flag가 ON 상태가 됨

Main Code

#2. 광물 채광 소스코드

```
int touching_gold = 0;
int touching_emerald = 0;
int touching_diamond = 0;
```

```
//현재 접촉 상태
int now_touching_gold = 0;
int now_touching_emerald = 0;
int now_touching_diamond = 0;
```

```
check_state = Player_Move(Jog_key, stage);

switch (check_state)
{
    case GAME_OVER:
        game_over = 1;
        break;
    case TOUCH_GOLD:
        now_touching_gold = 1;
        if (!touching_gold) {

            cnt_time = 0;
            TIM2_Repeat_Interrupt_Enable_time(1, 1000);

            Uart1_Printf("MINING GOLD\n");
        }
```

[1. 광물 접촉 Flag]

- 어떤 광물을 채광 중인지 알리는 Flag를 설정함
- now_touching_광물 Flag를 이용하여 현재 광물과 접촉 중인지 판별함
    - 해당 Flag를 이용해, 광물과 떨어졌을 때, 채광 시간 측정 타이머 값을 초기화하는데 이용

[2. 광물 접촉 판단]

- Jog key를 눌러, Player_Move함수 동작
- Player_move에서 각 광물과 충돌여부 판단
- 해당 광물과 충돌 판정 시, 그 광물에 할당된 값을 check_state에 할당
- 이전에 접촉하지 않던 상태라면, 채광시간을 측정하기위한 타이머를 작동시킴

Conclusion

성과

1. Flag 사용을 통해 얻는 이점을 배움
    - 코드 가독성 향상
    - 불필요한 중복 연산 방지
    - 디버깅 시, 여러 상태에 대한 추적용이
    - 인터럽트나 비동기 이벤트에 대한 연산 처리 시, 코드 간소화
2. 개발과정에서 파일 유지보수 법 체득
    - 백업파일(bak.c)을 통한 버전관리
    - #if DEBUG 전처리기를 통한 코드 유지보수

차후 Update 내용

1. 단계별 등장 광물 수 증가
    - 기존: 각 광물 1개씩 등장
    - Update
        - 각 광물의 개수 랜덤 등장
        - 스테이지 증가할수록 광물 수 증가
        - 목표 점수도 같이 증가
2. 종유석 개수 랜덤 결정
    - 기존: 종유석 2개 고정
    - Update
        - 종유석의 개수를 2~5개 정도로 랜덤 발생
3. 아이템 등장
    - 채굴 시간 단축, 경비 정지 등 아이템 요소 추가

Review

어려웠던 경험들

1. 채광 시간을 측정하는 타이머가 정상동작 하지않아, 광물이 캐지지 않거나, 의도한 시간과 다른 채광시간이 설정됨

<문제점>

- 채광 시간을 측정하기 위한 TIM2 인터럽트 방식을 Interrupt Based 방식으로 처리
- ISR에서 관련 Flag들을 모두 초기화해줘야한다는 고정관념의 오류를 범함
- 광물을 채광하거나, 게임을 초기화했을 때만, 측정 타이머 변수(cnt_time)을 초기화해야되지만, 인터럽트 발생 때마다 초기화 해줌
- 그 결과 광물 픽셀 접촉과 무관하게 채광시간 측정 타이머가 초기화됨

<해결>

- 기존 interrupt based방식에서 Event driven방식으로 바꿈
- 채광 Flag 및 스테이지 클리어 플래그를 이용해 해당 플래그가 ON일 때만, 측정 타이머를 초기화해줌

어려웠던 경험들

2. ADC 사용시, LCD 멈춤증상

<문제점>

- ADC 사용시, ADC를 초기화를 함
    - 사용되는 초기화함수가 Cds_Init과 IN5_init이 있음
- 처음 ADC 사용시 기존 예제코드의 IN5_init함수 사용
- IN5_init으로 ADC를 초기화 시, LCD가 먹통이 되는 문제 경험

<해결>

- IN5_init은 보드의 PA5을 Analog input 포트로 사용
- PA5는 LCD의 SPI 통신을 위한 SCLK 포트였음
- 때문에 ADC 사용시 LCD가 동작하지 않는 것이었음
- ADC 초기화 함수를 Cds_init으로 사용하여 PB1 포트를 ADC의 input포트로 사용함
- 충돌하는 포트를 다른 포트로 옮겨줌으로서 드라이버 충돌문제를 해결함