ARM-CORTEX M3 를 활용한

게임 개발 프로젝트

밀림서바이벌: 잊혀진 땅에서의 생존기

AI 반도체 설계 2기
서울기술교육센터
김민규

목차

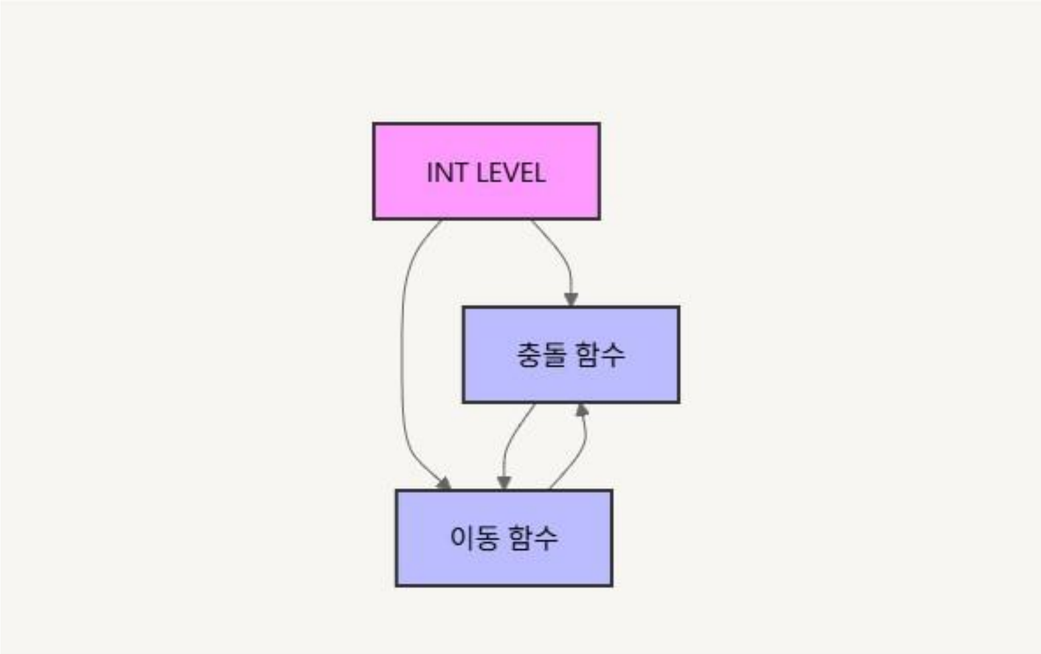
과제 개요. Arm cortex-M3 프로세서 사용.

- 유행했던 서바이벌류 게임을 참고.
- 게임의 핵심 요소 3가지
- 1. 사방에서 등장하는 오브젝트
- 2.유저 중심으로 자동 발사되는 무기.
- 3. 무기 커스텀 진화.

개발 일정

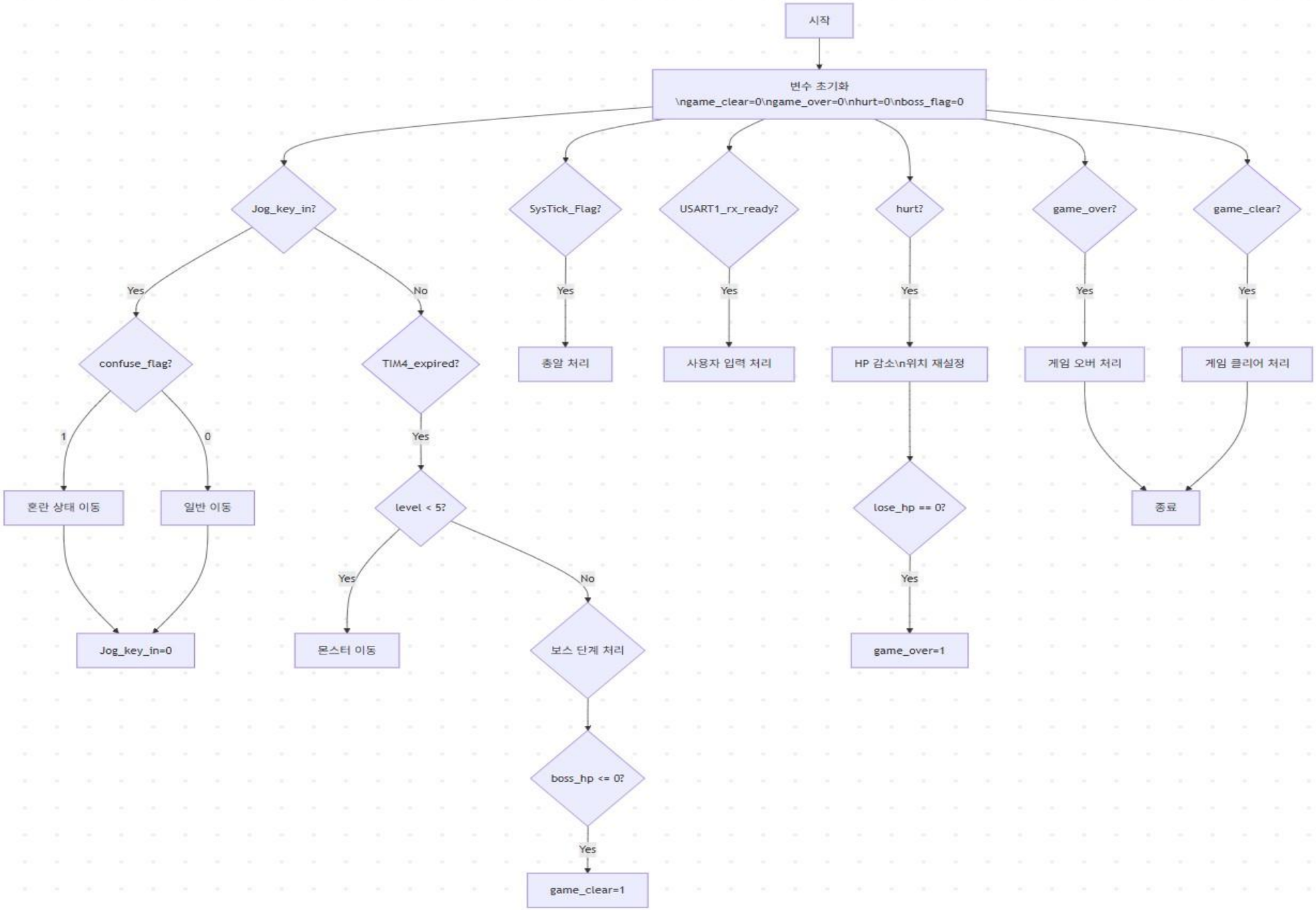
날짜	개발 내용	진행률
4월 26일	개발 목표 설정	10%
4월 27일	적 오브젝트 구현 총알 구현	30%
4월 28일	적 오브젝트 패턴 구상 총알의 위치 이동 구현	50%
4월 29일	적 오브젝트와의 상호 작용 구상	60%
4월 30일	적 오브젝트의 레벨별 패턴 구상 총알 오브젝트 추가 구현	80%
5월 1일	UI 디자인 적 오브젝트 파괴시 상호 작용 구현	90%
5월 3일	적 오브젝트 패턴 추가	100%

개발 결과



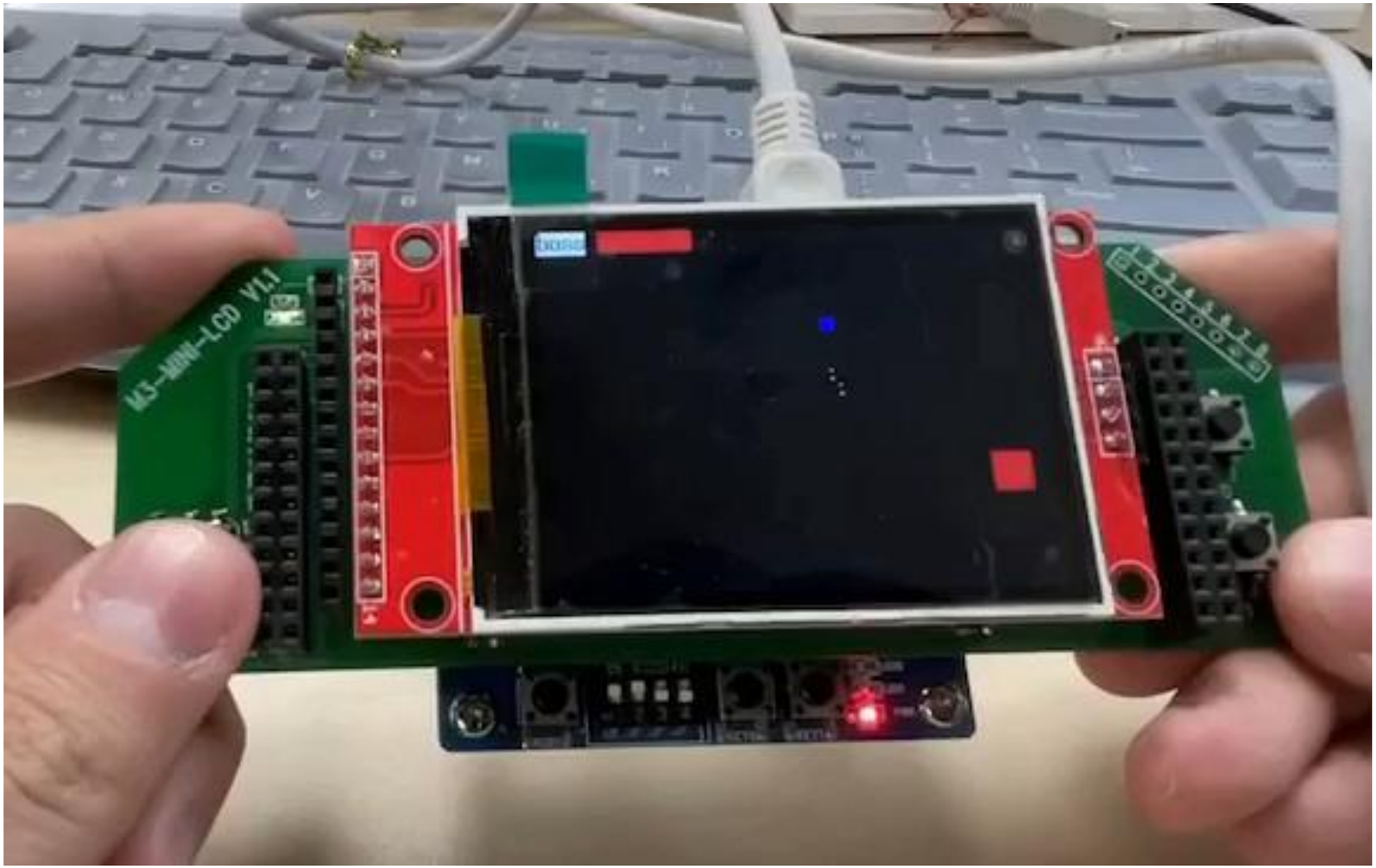
전체적인 메커니즘

- 대부분 각 오브젝트의 x,y,w,h,dx,dy값 조정 및 체크 함수로 되어있음.
- 각 오브젝트의 move와 Check_Collision이 동시에 이루어지고, 이것을 LEVEL 변수들이 제어하는 형태.

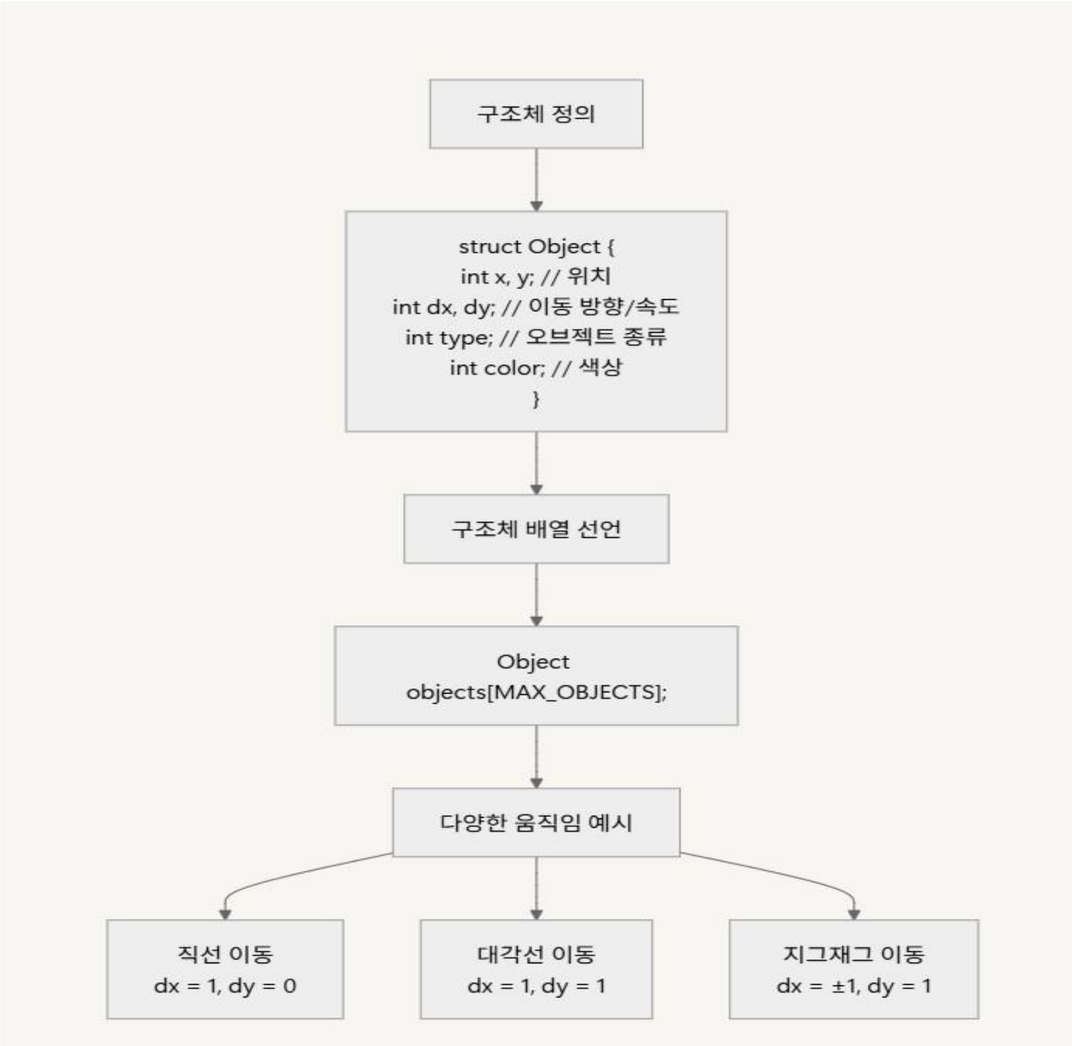

시작
변수 초기화
\ngame_clear=0\ngame_over=0\nhurt=0\nboss_flag=0
Jog_key_in?
SysTick_Flag?
USART1_rx_ready?
hurt?
game_over?
game_clear?
Yes
No
confuse_flag?
TIM4_expired?
충알 처리
사용자 입력 처리
HP 감소\n위치 재설정
게임 오버 처리
게임 클리어 처리
1
0
혼란 상태 이동
일반 이동
level < 5?
lose_hp == 0?
종료
Jog_key_in=0
몬스터 이동
보스 단계 처리
game_over=1
boss_hp <= 0?
game_clear=1

개발 결과

개발 결과



핵심 기술 및 핵심 코드



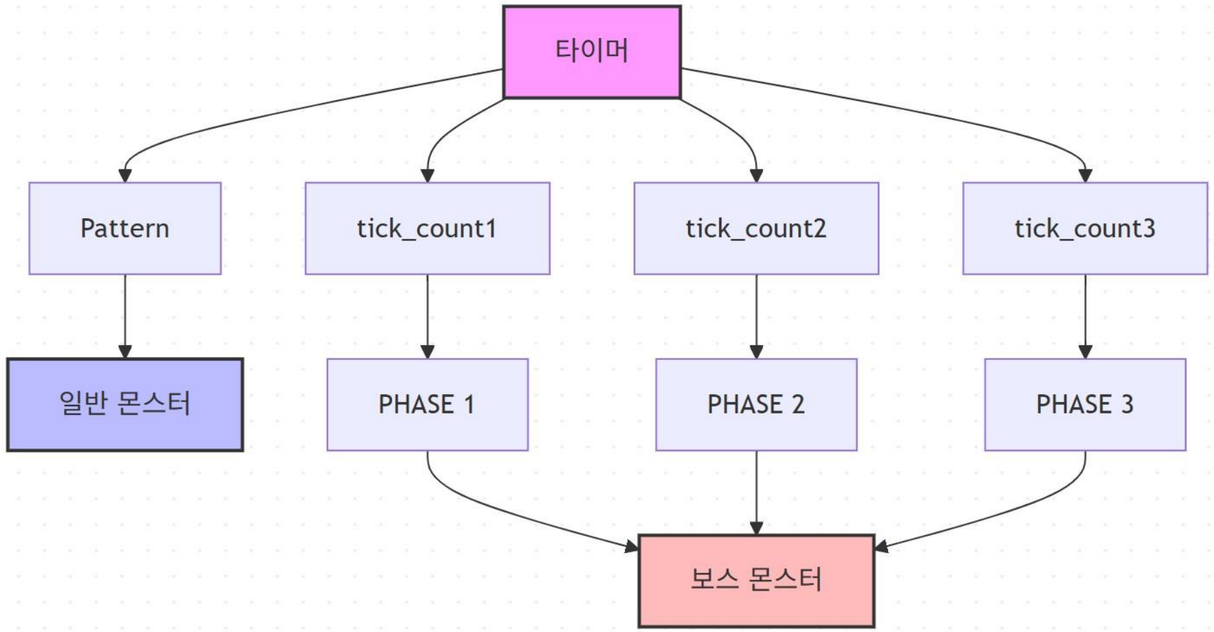
구조체 배열 활용.

오브젝트 종류를 구조체 배열로 선언하여 수월하게 관리.

구조체에서 dx, dy값 추가

기존의 오브젝트들보다 다채로운 움직임 조정 .

핵심 기술 및 핵심 코드



핵심 오브젝트 함수를 전역변수 하나로 관리

- 최소한의 변수로 움직임 제어 가능 .

여러 count로 타이머 효율 증대

- 여러 패턴의 오브젝트를 하나의 타이머로 동시 운용 가능

핵심 기술 및 핵심 코드

```
for (i = 0; i <= level; i++) {
    MONSTER[i].ci = BACK_COLOR;
    Draw_Object(&MONSTER[i]);


    if (i >= 3) {
        if (MY.x > MONSTER[i].x) MONSTER[i].dx = 1;
        else if (MY.x < MONSTER[i].x) MONSTER[i].dx = -1;
        else MONSTER[i].dx = 0;

        if (MY.y > MONSTER[i].y) MONSTER[i].dy = 1;
        else if (MY.y < MONSTER[i].y) MONSTER[i].dy = -1;
        else MONSTER[i].dy = 0;
    }
```

코드 내용

- 움직이는 오브젝트를 배열로 선언하고 행동 지정.

- 코드가 적어지고 충돌 비교할 때 매우 편했음. (이중 for)

핵심 기술 및 핵심 코드

```
if(level>=5)
{
    tick_count2 += TIMER_PERIOD*5;
    if(boss_hp<=PHASE2)
    {
        tick_count3 += TIMER_PERIOD*5;
    }
    if(boss_hp<=PHASE3)
    {
        tick_count4 += TIMER_PERIOD*5;
    }
```

코드 내용

- 타이머 하나로 다양한 패턴을 구현
- 보스의 체력을 구분 지어서 패턴을 다양화 시키니까 게임이 흥미진진함.

핵심 기술 및 핵심 코드

```
static int banana_move(void)
{
    int i;

     if(banana_flag == 0)// 처음 한번만 실행.
    {
    banana_replace();
    for (i = 0; i <= 2; i++) {
        BANANA[i].w = BANANA_SIZE_X;
        BANANA[i].h = BANANA_SIZE_Y;
        BANANA[i].ci = BANANA_COLOR;
        BANANA[i].dir = RIGHT;
        BANANA[i].dx = (rand() % 2) ? 1 : -1;
        BANANA[i].dy = (rand() % 2) ? 1 : -1;

    }
    banana_flag = 1;
    }
```

코드 내용

- 값을 한번만 받아야 되는 경우에 플래그를 활용

핵심 기술 및 핵심 코드

.

```
if (USART1_rx_ready)
{  if(USART1_rx_data == '1' && skill_level!=0)
   {
      skill_level --;
      gun_range += 100;
      Uart1_Printf("총알 사정거리 증가!\n");
      Uart1_Printf("현재 총알 사거리, %d\n", gun_range);
      Uart1_Printf("남은 스킬 포인트 : %d \n", skill_level);
   }
   if(USART1_rx_data =='2' && skill_level!=0)
   {
      if(gun_level<=4)
      {
      skill_level --;
      gun_level += 1;
      Uart1_Printf("총알 개수 증가!\n");
      Uart1_Printf("현재 총알 개수: %d\n", gun_level);
      Uart1_Printf("남은 스킬 포인트 : %d \n", skill_level);
      }
   }
```

코드 내용

- USART 입력을 통해서 사용자 스킬 진화.
- 구조체 배열 구조라 관리가 수월 했음.

결론

아쉬운 점 및 업그레이드 아이디어.

항목	문제점	해결방안
편의성 UI	Tera Term을 통해 모든 상황을 확인하는 것은 불편하다.	간단한 보스 체력이나 스킬 레벨 UI 구현.
몬스터 패턴 미흡	다채로운 움직임이 X. (알고리즘 부족)	Rand함수를 잘 활용한 움직임 제어
효과음	음악보다 효과음의 필요성을 느낌	음악 off시 타격을 Buzzer로 출력

개발 후기

실제 느낀점	다시 한다면
기초적인 지식이 많이 부족	설계부터 확실하게 진행하기
알고리즘 지식 필요	명확한 목표를 두고 코딩하기
	함수, 변수명 잘 생각해서 적기