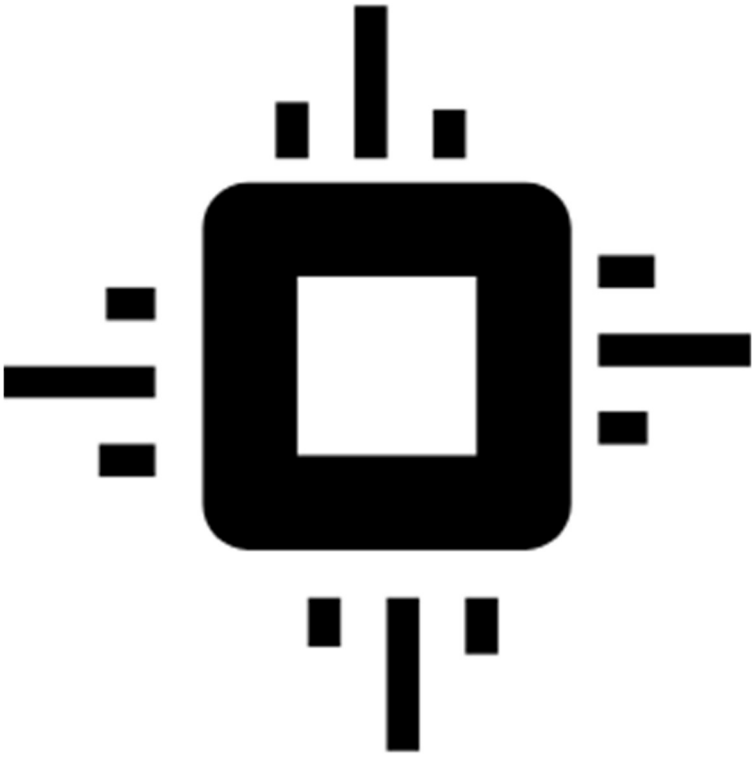
게임 설명서



제작자: 장찬원

교육명: ARM 아키텍처

작성일: 2025.05.06

목차

1. 게임 개요

1.1 개발 목적

본 프로젝트는 ARM 아키텍처 및 STM32F10x/Cortex-M3 기반 임베디드 시스템 학습을 목적으로 진행되었으므로, 일부 기능이 미완성되었거나 최적화가 덜 되어 있을 수 있습니다. STM32F10x 시리즈 MICOM과 Cortex - M3 프로세서를 직접 활용함으로써, 하드웨어 제어와 시스템 동작 원리에 대한 실질적인 경험을 쌓기 위해 개발되었습니다.

1.2 게임 소개

- **게임 제목:** 시골쥐 도시에 오다!
- **게임 목표:** 플레이어는 시골쥐가 되어 충돌요소(고양이, 올빼미)를 피하고, 날아오는 치즈를 최대한 많이 먹어 최고 점수를 달성해야 합니다.
- **주요 컨셉:** 동화 "시골쥐와 도시쥐" 이야기를 모티브한 게임입니다. 도시로 놀러 온 시골쥐가 자신 고향으로 다시 돌아가기 위해 충돌요소들을 피해 도시를 탈출하는 컨셉입니다.
- **장르:** 2D / 달리기 / 어드벤쳐 / 점프 회피

1.3 주요 특징

- STM32 기반 임베디드 점프 액션 게임
- 자동 달리기 + 타이밍 점프 시스템
- 아이템 획득 → 무적 모드 시스템
- 충돌 판정 및 무적 처리
- BGM 및 효과음 시스템

2. 게임 조작 및 규칙

2.1 조작키 안내

SW_1 (시작 버튼)

↪ 메뉴에서 → 게임 시작
↪ 게임오버 화면에서 → 게임 재시작

JOG_KEY_ ⬆ (점프 키)

↪ 게임 플레이 중 → 캐릭터 점프

2.2 게임 상태별 화면 설명

게임에서는 총 (MENU, PLAY, GAMEOVER) 3가지 상태로 구성됩니다.

(메뉴 상태)

↪ 음악 재생
↪ SW_0 누르면 → 게임 시작

(게임 상태)

↪ 자동 달리기 시작, 버튼 0으로 점프
↪ 치즈를 먹으면 점수 +1
↪ 고양이, 올빼미에 부딪히면 게임오버

(게임오버 상태)

↪ 게임오버 음악 출력, 점수 표시
↪ 버튼 5 누르면 → 다시 메뉴로

2.3 점수 및 아이템

- 게임 중 날아오는 치즈를 먹게 되면 +10 점수를 얻게 됩니다.
- 치즈를 5개를 먹게 되면 3초 동안 무적이 되며 속도도 빨라집니다.

2.4 적 충돌 규칙

- 충돌 요소는 2가지(고양이, 올빼미)로 구성됩니다.
- 무적 상태가 되면 충돌 규칙을 무시하여 게임오버가 되지 않습니다.

3. 코드 구조 요약

이 프로젝트는 ARM 아키텍처 강의에서 활용된 GAME_PROJECT의 소스 코드를 기반으로 하여 제작되었으며, 해당 코드를 레퍼런스 삼아 게임 로직, 하드웨어 제어, 그래픽 처리 등을 재구성하였습니다.

파일별 역할 설명

- **Main.c** 게임의 핵심 흐름 제어 (메인 루프, 게임 상태 전환, 충돌체크, BGM 등)
- **Lcd.c** lcd 화면 초기화 및 픽셀 출력 제어
- **Timer.c** 충돌요소 및 BGM 재생 제어
- **Device_driver.h / define.h** GPIO, 버튼, 전역 상수, 전처리 매크로 등 정의

3.1 define.h 파일

Main.c 소스 코드에 사용되는 전역 상수를 정의한 파일입니다. 주로 오브젝트 SIZE, 위치, 속도, 이동 주기, TIMER_PERIOD 등을 정의했습니다.

3.2 Main.c 파일

- **주요 구조체**

  PLAYER_OBJ, 플레이어 위치, 크기, 속도, 점프 상태를 저장합니다.

  CAT_OBJ, OWL_OBJ, CHEESE_OBJ, 충돌요소 위치, 크기, 상태, 이전 사이즈를 저장합니다.

  GAME_STATE, 게임의 MENU, PLAYING, GAMEOVER 상태를 저장합니다

  RUN_STATE, 게임 진행 중인 플레이어의 상태를 저장합니다.

- **MainLoop(void)**

  MainLoop() 함수는 TIMER_PERIOD(현재 60ms) 주기로 발생하는 TIM4 인터럽트에서 호출되어, 게임의 전체 상태에 따른 LCD 호출, 상황에 맞는 BGM, 적-아이템 스폰 및 이동, 무적 상태 관리 등을 처리하는 핵심 루프입니다.

- **BGM 및 효과음**

  MUSIC_ON( )은 시작 메뉴에서 반복 재생되는 BGM입니다. **GAMEOVER_MUSIC_ON( )**은 게임 오버 상태가 되었을 때 한번 재생 후 멈추도록 설계되어 있습니다. 점프 효과음은 음에서 랜덤으로 하

나를 재생하도록 설정했습니다. **Enum**을 통해 음계, 음표를 정의하고 2차원 배열로 악보로 만들어 재생하는 기반입니다.

- **Game_Updata_ALL(void)**

  플레이어의 위치 업데이트, (무적 상태 아닐 경우) 충돌 검사, 충돌요소 및 아이템 업데이트, 게임 점수 및 경과 시간 lcd 출력 순으로 기능을 동작합니다. 게임 도중 매 프레임 (TIMER_PER 60ms) 마다 화면과 내부 상태를 갱신하는 역할입니다.

- **ResetGame(void)**

  게임 내부 상태(점수, 경과시간, 타이머 등) 및 모든 충돌요소 등을 초기화한 후 Game_Init()함수를 호출하여 게임화면을 초기 상태로 되돌리는 기능입니다. 그리고 여기서 return의 기능은 메뉴나 게임오버 화면일 때 이후 로직을 건너뛰기 위한 기능입니다.

- **GameOver_Cleanup(void)**

  GameOver_Cleanup() 함수는 Timer 사용 중일 때 Lcd_Clr_Screen 했을 때 계속 잔상이 남는 문제가 있어 이 함수를 통해 게임오버 상태에서 화면과 모든 충돌요소의 잔상을 완전히 제거하여 lcd를 깨끗하게 표시되도록 준비합니다.

- **Player 동작 함수**

  Player_Jump_Start() 함수는 Player.jumping == 0일 때만(땅에 있을 때만) 점프를 시작할 수 있습니다. Player.vy에 초기 점프 속도(JUMP_INIT_VY)를 대입한 후 Player.jumping == 1 플래그를 설정하여 점프 중임을 표시합니다.

  Player_Draw(color index) 함수를 통해 현재 플레이어의 위치에 그림을 그립니다. 그리고 이전에 있던 위치 값을 저장합니다.

  Player_Update() 함수에서 이전 위치 값의 그림을 배경색으로 지웁니다. 점프 중인 상태라면 현재 속도(Player.vy)를 위치(Player.y)에 더하고, 중력(GRAVITY)을 속도(Player.vy)에 더합니다. 화면 하단까지 도달하면 위치를 고정하고, 속도를 0으로 리셋한 뒤 Player.jumping = 0으로 플래그 설정해 점프 상태를 종료합니다.

- **Cat 동작 함수**

  Cat_Init() 함수는 Cats[ ]의 active 상태를 비활성화 하여 Cat 오브젝트를 초기화한다.

  Cat_Update() 함수에서는 Cat 오브젝트끼리 서로 겹쳐지지 않게 조건문을 사용했습니다. 또한 오

브젝트가 우측 Lcd에서 시작하여 좌측으로 이동합니다. 만약 좌측 lcd 끝에 도달했다면 그림을 지운 후 새로운 Cat 오프젝트를 생성하도록 설계했습니다. 게임이 시작되면 초기화 -> 위치 갱신 및 잔상 제거 -> 조건 성립 시 스폰 과정을 처리합니다.

- **Owl 동작 함수**

  Owl_Init(), Owl_Update() 함수는 앞서 말한 **Cat_Init()** 함수와 같은 기능을 합니다.

  Owl_Spawn() 함수는 **define.h**에서 설정한 Max_Owl 값을 넘지 않는 오브젝트를 스폰합니다. 스폰된 Owl 오브젝트는 랜덤 한 높이를 가질 수 있게 **rand()** 난수 함수를 활용했습니다.

- **Cheese 동작 함수**

  Cheese_Update() 함수에는 Cheese 오브젝트가 움직일 때 발생할 때 생기는 애니메이션을 작동시켜 줍니다. **CHEESE_ANIM_INTERVAL**의 일정 간격마다 Cheese 오브젝트를 커졌다 작아졌다 반복시킵니다. Player와 Cheese 오브젝트끼리 충돌 검사를 통해 점수 추가, 무적 상태 돌입하는 기능을 수행합니다.

  Cheese_Spawn() 함수는 Cheese 오브젝트가 Lcd 화면 내 랜덤으로 스폰할 수 있게 설계했습니다.

- **Main(void)**

  시스템 - 타이머 초기화 및 인터럽트 설정 -> 게임 상태 초기 설정 -> 무한 루프 내 게임 상태에 따른 BGM, MENU, PLAYING, GAMEOVER 처리를 통해 전체 흐름을 제어합니다.

  초기 게임 시작 시 MENU 상태로 시작할 수 있게 게임상태 변수를 초기 설정한 후 게임 상태에 따라 처리할 수 있게 설계했습니다.

  MENU 상태일 때 **Draw_Start_Menu()** 함수를 통해 한 번만 그린 뒤, 키 입력(SW_1) 시 BGM 중지 및 화면 초기화 후 **ResetGame()** 함수 호출하여 플레이 상태로 전환하게 됩니다.

  PLAYING 상태일 때 **MainLoop()** 함수에서 점프, 오브젝트 스폰, 충돌, 렌더링을 TIME_PERIOD 주기로 처리하도록 설계했습니다.

  GAMEOVER 상태일 때 **Cleanup()** 함수를 통해 Lcd화면을 정리 후 **Draw_GameOver_Screen()** 함수를 호출하도록 합니다. 점프(KE5) 입력 시 BGM 중지 및 화면 초기화 후 **ResetGame()** 함수를 호출하여 재시작 준비를 하도록 설계했습니다.

3.3 Lcd.c 파일

- **Lcd_Draw_Line(int x1, int y, int x2, int y_fixed, unsigned short color)**

  입력 받은 값을 통해 수평선을 Lcd에 그려주는 함수입니다.

- **Lcd_Draw_Line(nt x, int y, int size, unsigned short color)**

  입력 받은 값을 기준으로 **define.h** 파일의 SIZE 값의 정도에 삼각형을 그려주는 Lcd에 그려주는 함수입니다.

4. 향후 개선 및 확장 방향성

- **화면 렌더링 최적화**

  프레임 버퍼 방식을 추가하여 게임 내 깜빡임 최소화합니다. (STM32의 용량이 20KB이기 때문에 더블 버퍼 방식은 사용할 수 없습니다.)

- **게임 콘텐츠 확장**

  여러 스테이지 및 보스 전투 추가, 새로운 파워업 아이템 도입, 목숨 시스템 도입

- **사용자 인터페이스 강화**

  각 오프제트의 애니메이션 추가, 이미지 및 모션 추가