REPORT

제출 일자 :	2023/12/07
담당 교수 :	최차봉 교수님
과목 :	임베디드 소프트웨어 입문
학부(과) :	소프트웨어학과
이름 :	남서아
학번 :	2019125021
과제번호 :	프로젝트

1. 배경과 아이디어

이 프로젝트는 "**Dancing Dino**"라는 이름의 파이썬 기반 리듬 게임으로, RaspberryPi4와 Adafruit LCD 디스플레이 콘솔을 사용하여 플레이할 수 있습니다. 아이들 모두가 즐길 수 있는, 간단하고 밝은 분위기의 게임을 만들고자 이 게임을 구상하게 되었습니다.

이 게임은 오프닝 애니메이션으로 시작해, **아기 공룡이 멋진 댄서로 성장하는 여정**을 스토리로 반영하고 있습니다. 도입부의 애니메이션에서는 아기 공룡이 졸다가 꿈을 꾸는데, 그 꿈 속에서 자신은 멋진 댄서가 되어있습니다. 꿈에서 깬 직후, 눈이 초롱초롱해진 공룡은 춤을 연습하기로 결심합니다. 이후 집 마당, 연습실, 공연장의 순서로 총 3개의 스테이지가 진행됩니다. 각 스테이지에 따라 배경 이미지가 전환되며, 그에 맞게 난이도 또한 높아집니다. (화살표가 내려오는 속도가 빨라짐). 아기 공룡은 사용자의 조작에 맞게 움직이며 춤을 춥니다

또한, 기존 리듬 게임들의 메커니즘을 채용하여, 구간에 따른 점수 시스템(MISS/GOOD/PERFECT)과 콤보에 따른 보너스 점수 시스템, 그리고 홀드 시스템 등이 존재합니다. 플레이어는 아기 공룡이 멋진 댄서가 될 수 있도록 스테이지를 클리어해야 합니다.

2. 게임 방법

기본 조작

- **조이스틱 사용**
    - 화면 상단에서 하단으로 이동하는 화살표의 방향에 따라 조이스틱을 움직입니다.
    - 화살표가 점수 존에 도달할 때 정확한 방향으로 조이스틱을 조작합니다.

화살표 유형

- **기본 화살표** : 간단한 조작으로, 화살표가 가리키는 방향으로 조이스틱을 움직입니다.
- **'홀드' 화살표** : 지정된 방향으로 조이스틱을 계속 유지하고 'A' 버튼을 눌러야 합니다.

    '홀드' 화살표는 화면에 나타난 뒤, 그 꼬리(노란색 사각형)가 구간을 다 지나갈 때까지 조이스틱을 지정된 방향으로 유지해야 점수를 얻을 수 있습니다.

점수 체계

- **점수 획득**
    - 알맞은 점수 존에서, 화살표의 방향과 조이스틱 조작이 정확히 일치할 경우 점수를 얻습니다.

      빨간색 구간 - miss (-10) / 초록색 구간 - good (+30)

      보라색 구간 - perfect (+50)

    - 'Perfect', 'Good' 등의 피드백으로 정확도를 알려줍니다.
    - 연속적인 성공적 매칭은 콤보 점수를 증가시킵니다.

- **점수 감소**
    - 조작이 부정확하면 점수가 감소합니다. ('Miss'의 피드백과 함께 점수 10 감소)

      hit 타이밍이 맞지 않은 경우, 잘못된 방향으로 화살표를 조작한 경우 등이 해당됩니다.

단계별 진행

- 게임은 다양한 배경과 증가하는 난이도를 가진 여러 단계로 구성됩니다.
- **50초** 동안 **2000점**을 얻어야 다음 스테이지로 진행할 수 있으며, 그렇지 못한 경우 B 버튼을 눌러 해당 스테이지를 다시 플레이 해야 합니다.

추가 기능 및 버튼

- **'A' 버튼**
    - 주로 '홀드' 화살표에서 사용되며, 특정 조건에서 조이스틱 방향을 유지하는 데 사용됩니다.
    - 도입부 애니메이션이 끝난 후, 스테이지를 최초로 시작할 때 사용됩니다.
- **'B' 버튼**
    - 게임 시작 및 스테이지 재시작에 사용됩니다.

3. 사용된 기술

기본적인 기술

- Raspberry Pi 4 & Adafruit LCD Display : 디스플레이 및 입력 장치로 활용
- Python 프로그래밍 : 게임 로직, 사용자 인터페이스, 그래픽 처리 등을 구현
- Tkinter 라이브러리: 사용자 인터페이스 및 그래픽 렌더링
- PIL (Python Imaging Library): 이미지 처리 및 렌더링

기술적 강조점

1. '홀드' 화살표의 구현

'홀드' 화살표는 플레이어가 지정된 방향으로 조이스틱을 유지하면서 'A' 버튼을 누르고 있어야 합니다. 이를 위해서는 게임에 지속적인 입력과 정밀한 타이밍을 감지하는 로직을 구현해야 했습니다.

홀드의 성공 여부를 판단하는 것, 그리고 화살표의 꼬리를 또한 화살표의 속도 및 홀드 시간을 고려하여 알맞은 크기로 렌더링 하는 것 또한 지속적인 계산을 요구합니다.

2. 게임 스테이지 및 화살표 관리

각 스테이지는 다른 배경 이미지와 화살표 속도를 가지며, 매 스테이지 시작 전 ready, 3, 2, 1, start 등을 보여주는 것 또한 상태 변수를 두어 각각을 단계로서 관리해야 했습니다. 또한 해당 변수들을 모두 렌더링 시 고려 해야했습니다. 그리고 50초의 플레이 타임으로 각 스테이지가 동작하다보니, 현재 남은 시간 또한 스테이지 종료 조건으로서 계속해서 계산 및 확인해야 합니다.

랜덤한 방향으로 홀드 또는 일반 화살표를 생산해 내는 것 뿐만 아니라, 화살표들의 위치와 사용자의 입력을 계속해서 추적하여 점수를 처리하는 것 또한 구현에 많은 시간이 소요되었습니다. 특히, **화살표에 대한 점수 처리 부분이 가장 복잡**했습니다. 단순히 hit 만으로 점수를 얻을 수 있는 게 아니기에, 처리해야하는 경우의 수가 많았기 때문입니다.

화살표의 단순한 방향과 복합적인 방향(대각선)의 일치 여부와 함께, 화살표의 홀드가 지속되지 않는 경우 및 점수 존과의 위치 비교까지 복합적인 부분을 모두 고려해서 점수를 처리해야 했습니다.

3. 그래픽 및 애니메이션 렌더링

단순히 화살표를 움직이는 것 뿐만 아니라, 조이스틱의 입력에 따른 플레이어(아기공룡) 캐릭터의 동작이 애니메이션처럼 보여지도록 구현했습니다. 이는 타이머를 기반으로 하여, 매우 짧은 시간 간격마다 포즈가 이어지며 렌더링 되도록 구성한 것입니다.

따라서 플레이어의 조작이 없을 때에도 공룡은 위아래로 몸을 살짝 흔드는 모습을 보여주며, 사용자가 조이스틱을 조작하는 경우에도, 공룡은 해당 방향에 맞는 기본 포즈와 해당 방향에 맞는 포즈를 연결해서 보여주어 보다 매끄러운 시각적 경험을 제공합니다.

도입부 애니메이션 또한 시간 간격을 기반으로 25장의 프레임 이미지를 준비하여 구현하였습니다.

화살표 또한 hold 화살표에 별도의 꼬리를 렌더링 하며, 이 경우 화살표의 속도 및 홀드 시간, 위치 등을 모두 고려해야 했으므로 이 점 또한 구현에 많은 시간이 소요되었습니다.

4. 문제점과 해결책

문제점

원래는 Adafruit LCD 디스플레이 출력을 통해 게임의 화면을 구성하였으나, **LCD 패널의 고장**으로 인해 기존의 방식대로 게임을 플레이할 수 없게 되었습니다. (장시간 사용 및 과열로 인한 고장으로 추정)

해결책

수업시간에 교수님이 말씀하신 대로, **tk 라이브러리**를 추가로 사용하여 문제를 해결했습니다. 이를 위해 PIL 출력 이미지를 tk 이미지로 변환 후 tk 그래픽 창에 출력하는 방식으로 변경하였습니다. 또한, 게임의 지속적인 update 또한 무한루프로 game.update()를 실행하는 것이 아닌, Game 클래스 내부에서 self.master.update(250, self.update)와 같이 딜레이를 둔 재귀를 수행하는 방식으로 코드를 수정했습니다.

5. 코드 설명

클래스 구조 및 각 클래스의 기능

JoystickController 클래스

- 목적: 조이스틱 입력 관리, 디스플레이 초기화, 버튼 상태 업데이트.
- 핵심 기능
    - SPI 통신을 사용하여 조이스틱과 디스플레이를 제어.
    - 사용자의 입력(조이스틱 방향, 버튼 클릭)을 실시간으로 감지하고 처리.

Player 클래스

- 목적: 플레이어의 스프라이트 이미지 관리, 포즈 업데이트, 애니메이션 렌더링.
- 핵심 기능
    - 플레이어의 다양한 상태(예: 기본, 왼쪽, 오른쪽)에 대한 스프라이트 이미지 관리.
    - 애니메이션의 프레임 전환 및 렌더링.

Arrow 클래스

- 목적: 화살표의 움직임 및 렌더링 처리.
- 핵심 기능
    - 화살표의 방향, 속도, '홀드' 상태를 관리.
    - 화면상의 화살표 위치 업데이트 및 렌더링.

Stage 클래스

- 목적: 게임 스테이지 관리, 배경 렌더링, 점수 존 관리, 화살표 생성 및 스테이지 전환.
- 핵심 기능:
    - 각 스테이지에 맞는 배경 이미지 및 설정 로드.
    - 화살표의 생성 및 스테이지별 난이도 조절.
    - 점수 존 설정 및 플레이어의 점수 계산.

Game 클래스

- 목적: 게임의 메인 루프와 상태 관리.
- 핵심 기능:
    - 게임 시작 화면, 게임 오버 상태 관리.
    - 게임의 주요 루프 실행 및 전체 게임 흐름 제어.

클래스 간 연결 및 상호작용

- **JoystickController**는 Player와 Stage 클래스에 사용자 입력 정보를 제공합니다.
- **Player** 클래스는 Stage 클래스 내에서 플레이어의 애니메이션과 상태 변화를 처리합니다.
- **Arrow** 클래스는 Stage 클래스 내에서 화살표의 생성, 이동 및 렌더링을 담당합니다.
- **Stage** 클래스는 게임의 각 스테이지를 관리하며, 화살표, 점수, 배경과 같은 요소들을 통합적으로 조정합니다.
- **Game** 클래스는 게임의 주요 루프를 구성하며, JoystickController, Player, Stage 클래스를 이용하여 게임의 전반적인 흐름을 제어합니다.

시연 동영상 (유튜브)

https://www.youtube.com/watch?v=G-tjcUxmIwA

깃허브 링크

https://github.com/Wendy-Nam/DancingDino