# CSED101 프로그래밍과 문제해결

## Assignment #4

2024년 6월 18일

**학과** 무은재학부

**학번** 2024****

**이름** 박재원

**POVIS ID** ****

명예서약 (Honor code)

"나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다."

# 차 례



# 1 개요

본 과제는 Connect Four 게임을 구현하는 것이다. Python과 Tkinter를 활용하여 GUI로 구현해야 하며, 두 플레이어가 하나의 컴퓨터로 플레이 할 수 있도록 구현해야 한다.

Connect Four는 두 플레이어가 번갈아 가며 본인 색의 공을 직사각형 판에 떨어뜨려, 본인의 공 4개를 직선으로 연속해서 연결한 플레이어가 승리하는 게임이다.

# 2 설계

## 2.1 구조도

프로그램의 전체 구조도는 그림 1와 같다.

Info 클래스는 게임 화면 상단의 현재 플레이어 정보를 담당하며, Board는 하단의 게임판 및 사용자 입력 정보를 담당한다. Ball 클래스는 게임판의 공 하나를 담당하여 공 표시 및 이동 애니메이션을 담당한다.

**입력부** 마우스 이동(현재 선택된 열 판단), 마우스 클릭(착수)

**처리부** 착수 처리, 승부 결정/무승부 판단 등

**출력부** 플레이어 정보, 게임판 및 착수된 공, 사용자 입력 정보 GUI로 출력, 승부 결과 window 표시 등

## 2.2 알고리즘

프로그램의 알고리즘을 의사코드로 나타내면 알고리즘 1과 같다.

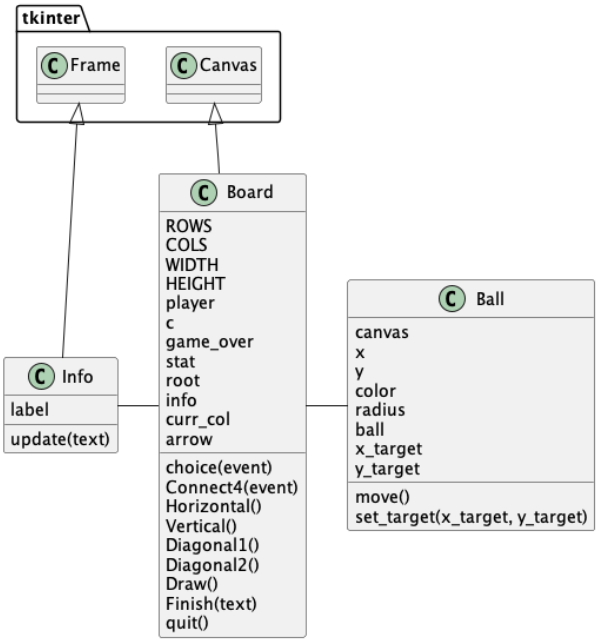


그림 1: 본 프로그램의 전체 구조도

**Algorithm 1** 본 프로그램 주요 부분의 의사 코드

위젯 인스턴스 생성 및 화면 구성 요소 출력
**while do**
  **if** 마우스 이동 **then**
    선택된_열 ← (마우스 x 좌표) ÷ (한 cell 가로 길이) 의 몫
    사용자 입력 정보 갱신
  **end if**
  **if** 마우스 클릭 **then**
    **if** 선택된 열 맨 꼭대기(index 0)에 공이 있다 **then**
      이벤트 무시하기
    **end if**
    선택된_행 ← 선택된 열의 항목 중 값이 white면서 가장 아래에 있는(index가 가장 큰) 행
    공 인스턴스 생성 및 선택된 열 및 행으로 이동
    stat 업데이트
    **if** 연속된 공 4개가 존재한다 **then**
      마지막으로 착수한 유저 승리 메시지 출력 및 게임 오버 창 생성 및 프로그램 종료
    **end if**
    **if** 모든 게임판이 공으로 찼다 **then**
      무승부 메시지 출력 및 게임 오버 창 생성 및 프로그램 종료
    **end if**
    사용자 턴 전환 및 현재 턴 출력
  **end if**
**end while**

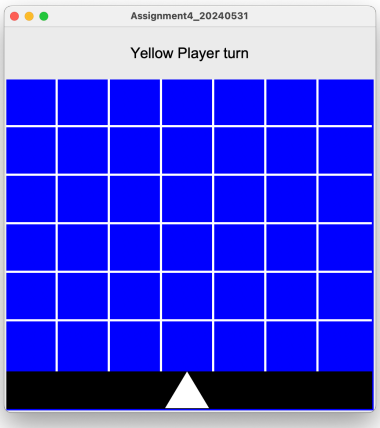


그림 2: 실행 모습. 초기 화면

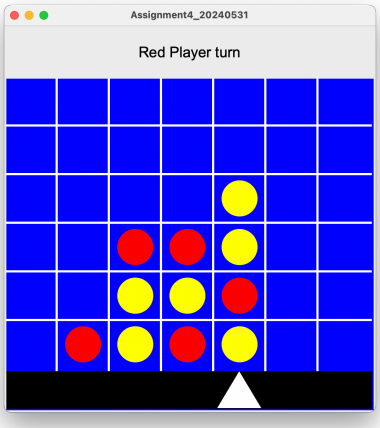


그림 3: 실행 모습. 게임 중간 모습

## 3 실행 방법 및 예제

본 과제는 MacOS[1], CPython 3.12.2에서 작성 및 테스트되었다.

본 과제를 실행하려면 다음과 같이 assn4.py 파이썬 파일을 실행하면 된다. 프로그램을 실행하면 GUI window가 나타난다. 해당 window와 마우스 이동 및 클릭으로 상호작용하면서 프로그램을 사용할 수 있다.

```
ls
# assn4.py
python assn4.py
```

실제 프로그램 실행 모습은 그림 2 ~ 5와 같다.

[1]특정 OS에 종속적인 기능을 사용하지 않으므로 다른 운영체제에서도 정상 작동하리라 예상된다.

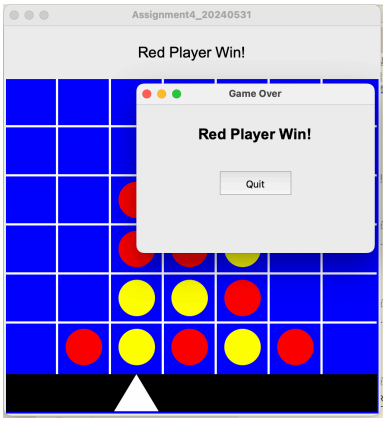


그림 4: 실행 모습. 게임 종료

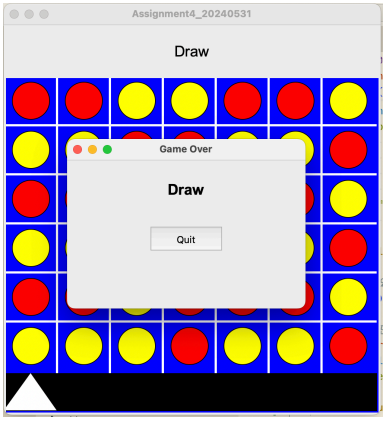


그림 5: 실행 모습. 게임 종료 (무승부)

- 그림 2는 실행 직후 초기 모습이다. 문제 파일의 요구사항대로 프로그램 제목 및 게임판, 기타 UI를 표현하고 있다. 상단 문구 ('Yellow Player turn')을 통해 Yellow Player의 차례임을 알 수 있다.
- 그림 3은 게임 진행 중간 모습으로, 번갈아 가며 공을 착수하고 있다. 현재 Red Player 턴으로 화면 상단에 'Red Player turn' 문구가 출력되고 있다.
- 그림 4는 게임 종료 모습으로, Red Player가 승리한 모습이다.
- 그림 5는 무승부로 게임이 종료된 모습이다.

# 4 토론

## 4.1 Info.update 메소드 (명세 추가)

게임 화면의 플레이어 정보를 업데이트하기 위해서는 Info 인스턴스의 변수 중 label 변수에 접근해 옵션을 바꾸어야 한다. 이를 단순화하기 위해서 Info.update 메소드를 정의해 사용했다.

이 메소드는 표시할 문구를 인자로 받으며, 이 문구대로 플레이어 정보 표시 문구를 수정하는 일을 한다.

## 4.2 Board.___init___ 메소드 (명세 변경)

Board class 내부에서 root 인스턴스와 info 인스턴드에 접근해야 할 일이 있었다. 플레이어 정보 문구를 수정하거나, 게임 종료 창을 생성할 때 각각 info 인스턴스와 root 인스턴스가 필요하다.

이를 위해 Board.___init___ 생성자의 인자를 수정하여 root와 info를 받도록 만들었고, 함수 내부에서 self.root와 self.info에 인스턴스를 저장해두었다가 나중에 필요할 때 사용하도록 구현했다.

# 5 결론

파이썬과 Tkinter를 이용하여 Connect Four 게임을 만들어봄으로써 GUI 프로그램 만드는 방법을 익혔다. 다양한 위젯 사용, 클릭 및 마우스 이동 이벤트 처리 등을 익히며 GUI 프로그래밍에 익숙해질 수 있었다. 또한 객체 지향 프로그래밍에도 더 익숙해질 수 있었다.

# 6 개선 방향

승부 결정을 판단할 때, 현재는 모든 영역에서 4개의 공이 연속되었는지 확인하도록 코드를 작성했다. 하지만 이전에 착수했던 공들의 위치는 변하지 않으므로 이 곳에서 연속된 공이 나오지 않을 것임이 보장되고, 이 영역을 다시 확인하는 것은 불필요하다. 그러므로 마지막으로 착수한 공 주변에서만 연속된 공 4개가 등장하는지 판단하도록 개선하면 훨씬 효율적인 프로그램이 될 것이다.