2023.12.29.

간단한 스네이크 게임 구현

자기주도PJT con18 - rep01

공유 Day

광주 2반
이현호

1. 프로젝트 개요

1. 프로젝트 개요

본 프로젝트는 어릴 적 누구나 한번쯤은 플레이 해보았던 스네이크 게임을 직접 구현해보는 과제입니다.
이 과제를 통해서 게임 오브젝트의 이동 원리와 프레임의 이해, 충돌 영역의 체크 등 기본적인 동작을 익히고 화면에 물체를 표현하는 방법을 익힙니다.

2. 과제 목표

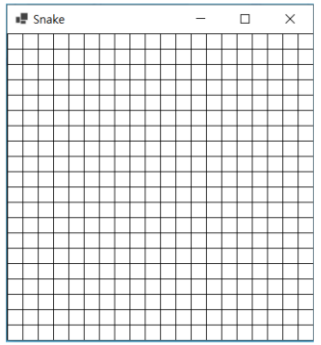
과제 목표는 스네이크 게임을 구현하는 것으로 언어에 관계없이 해당 명세서 내용을 따라 실습해보는 것이다.

다만, 아래 개발 목표 및 게임 룰은 반드시 포함되어야 함이 필수 조건이다.

- 전체 맵 해상도 및 구성 방법 : (X : 400 pixel, Y : 400 pixel)으로 격자 무늬(그리드)를 가로 20 개, 세로 20 개의 선으로 그려서 표시한다.

2. 과제 목표 (1)

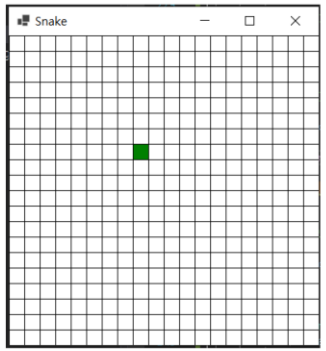
對外秘



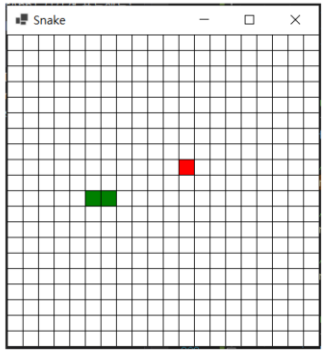
- 뱀의 머리와 몸통 이미지 : (X : 20 pixel, Y : 20 pixel)의 초록색 사각형으로 표현하며 먹이를 먹을 때 마다 머리를 포함한 몸통의 길이가 1 칸씩 커지도록 구현한다.

  단, 뱀의 머리는 맵 상에서 랜덤한 위치에서 시작하도록 하며, 상, 하, 좌, 우 4 방향 중 한 방향이 랜덤 하게 설정되도록 한다.

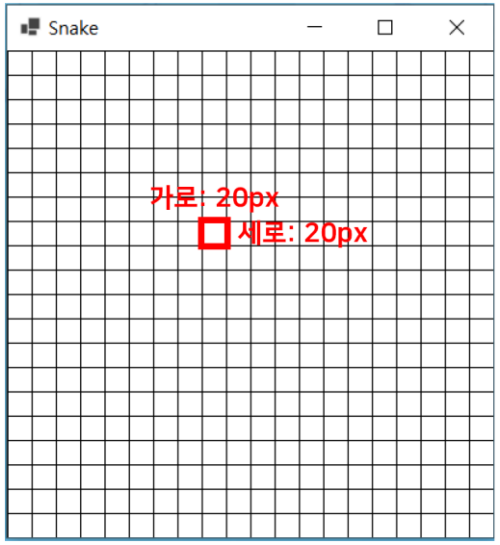
  뱀의 몸통은 길이 제한이 없으며, 1 프레임이 지날 시 N 번째 몸통은 N-1 번째 몸통의 위치로 이동할 수 있어야 한다.

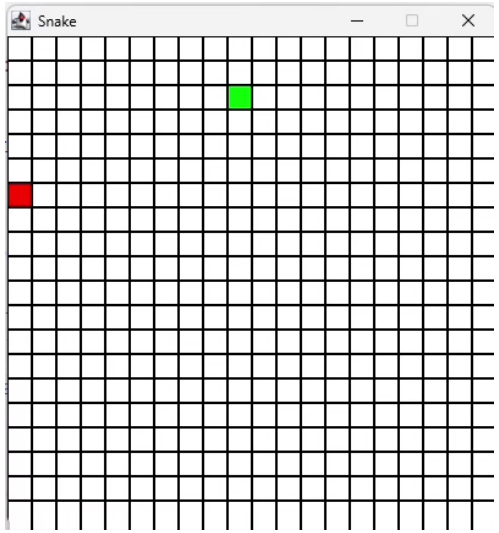


시작 화면



먹이를 1 개 먹은 화면





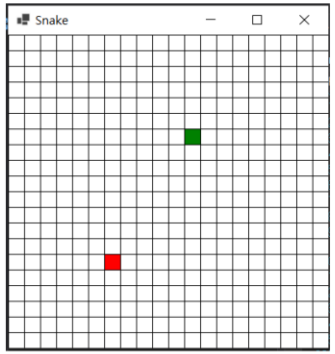
게임을 시작하면

1. 뱀의 머리는 맵 상에서 랜덤한 위치에 생성
2. 상, 하, 좌, 우 4방향 중 한 방향을 랜덤하게 설정

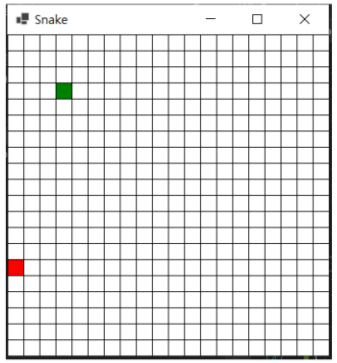
2. 과제 목표 (2)

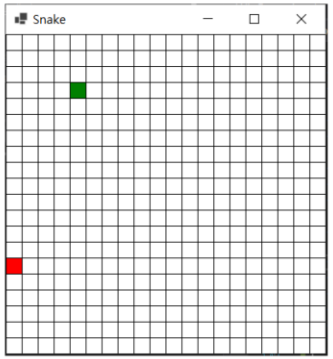
對外秘

- 먹이 : (20, 20)의 붉은색 사각형으로 표현하며 맵 상에 랜덤하게 등장해야 하며
  등장 위치는 뱀의 머리와 중복되지 않도록 한다.



- 이동 조건 : 뱀의 머리는 처음에 세팅된 방향으로 1 프레임 당 20 픽셀 씩
  자동으로 이동하며, 키보드 의 방향키 상, 하, 좌, 우 입력 시
  뱀의 머리가 입력된 방향으로 바뀔 수 있도록 구현 한다.
  아래 그림은 뱀의 방향이 오른쪽으로 설정이 되어 있는 경우 1 칸 이동하는 예시이다.





[조건]

1. 먹이는 맵 상에 랜덤하게 등장하며 등장 위치는 뱀의 머리와 겹치지 않아야 함
2. 키보드의 방향키 상, 하, 좌, 우 입력으로 방향 전환

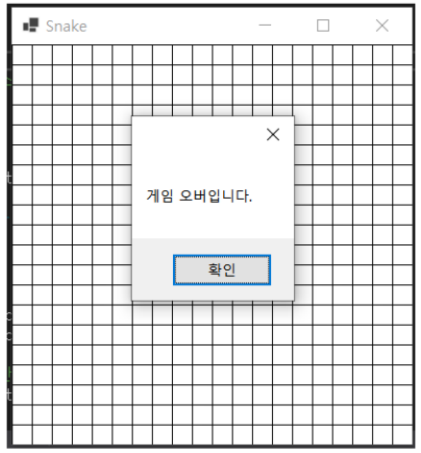
5

2. 과제 목표 (3)

對外秘

뱀의 위치(4 * 20, 4 * 20)　　뱀의 위치 (5 * 20, 4 * 20)로 업데이트

- 게임 오버 조건 : 뱀이 자신의 몸에 닿거나, 화면 밖 (X 축, Y 축 값이 0 보다 작거나 400 pixel 보다 큰 경우)으로 나갔을 경우 게임 오버 라는 메시지 창을 띄운다.
  확인 버튼을 누르면 뱀의 위치와 먹이의 위치가 초기화되어 게임을 재 시작한다.



3. 참고 자료

- python 으로 구현

pygame 설치 후 사용 예제 코드 https://wonhwa.tistory.com/44

- java 로 구현

Java swing 설치 후 사용 예제 코드 https://binghedev.tistory.com/50

- c#으로 구현

[조건]

3. 뱀이 자신의 몸에 닿거나 화면 밖으로 나가면 게임 오버
4. 확인 버튼을 누르면 뱀의 위치와 먹이의 위치가 초기화되어 게임을 재시작

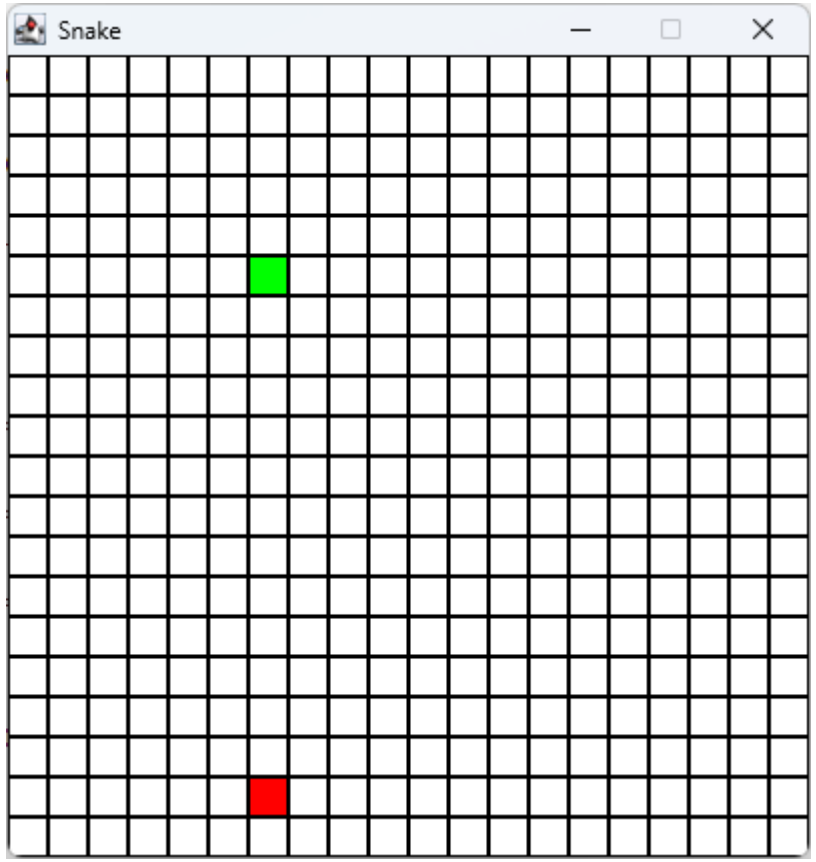
3. 구현

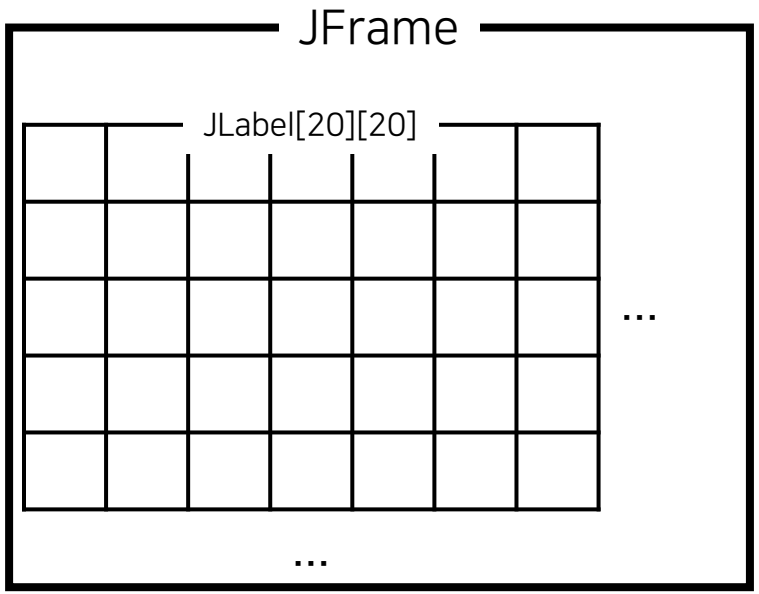
0. 흐름

```
1 static void play() {
2     makeMap();                //맵 생성
3     randomPositioning();      //랜덤 위치 설정
4     move();                   //뱀 이동 시뮬레이션 시작
5 }
6
7 public static void main(String[] args) {
8     play(); //게임 시작
9 }
```

3. 구현

1. 맵 생성





3. 구현

2. 랜덤 위치 설정

```
1  //뱀의 위치와 이동 방향, 먹이의 시작 위치를 랜덤하게 결정
2  static void randomPositioning() {
3      //뱀의 시작 위치 랜덤 결정
4      headX = (int) (Math.random() * N);
5      headY = (int) (Math.random() * N);
6
7      jLabels[headX][headY].setBackground(Color.GREEN);
8
9      //먹이의 위치를 뱀의 시작 위치로 덮어씀 (겹치지 않기 위함)
10     foodX = headX;
11     foodY = headY;
12
13     while(foodX == headX && foodY == headY) {
14         //먹이의 시작 위치 랜덤 결정
15         foodX = (int) (Math.random() * N);
16         foodY = (int) (Math.random() * N);
17     }
18
19     jLabels[foodX][foodY].setBackground(Color.RED);
20
21     //뱀의 이동 방향 랜덤 결정
22     direction = (int) (Math.random() * 4);
23 }
```

4~5: 뱀의 시작 위치 랜덤 결정

7: 뱀의 시작 위치 초록색으로 색칠

10~11: 먹이의 위치를 뱀의 시작 위치로 덮어쓰고
13~17: 뱀의 머리 위치와 먹이의 위치가 같지 않을 때 까지 랜덤 실행

19: 먹이의 위치 빨간색으로 색칠

22: 뱀의 이동 방향 랜덤 결정

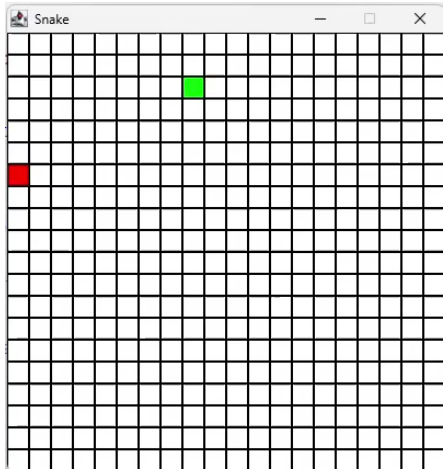
3. 구현

3. 뱀 이동 시뮬레이션 시작

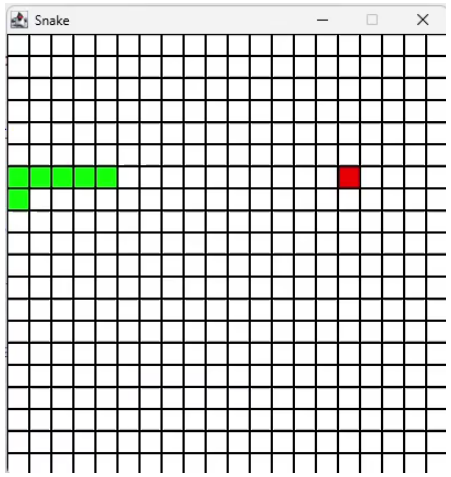
```java
static void move() {
        Timer timer = new Timer();
        TimerTask task = new TimerTask() {
            @Override
            public void run() {
                try {
                    snake.offerFirst(new int[] {headX, headY});                //현재 머리의 위치를 ArrayDeque에 추가 (몸통의 위치가 됨)

                    //머리의 위치 갱신
                    headX += dx[direction];
                    headY += dy[direction];

                    //1. 벽 또는 자기 자신의 몸과 부딪힌 경우 -> 게임 종료
                    if(isCollide()) {
                        /** 타이머 취소 */
                        this.cancel();
                        timer.cancel();

                        int result = JOptionPane.showConfirmDialog(null, "게임 오버입니다.", "확인", JOptionPane.PLAIN_MESSAGE);

                        if(result == JOptionPane.OK_OPTION) {
                            play();     //게임 재시작

                            return;     //아래 코드 실행 안함 (NullPointerException 방지)
                        }
                    }

                    jLabels[headX][headY].setBackground(Color.GREEN);          //새로운 머리의 위치를 초록색으로 칠함
                    int[] tail = snake.pollLast();                             //꼬리가 위치한 칸을 비움

                    //2-1. 이동한 칸에 먹이가 없는 경우 -> 꼬리를 앞당김
                    if(!(headX == foodX && headY == foodY)) {
                        jLabels[tail[0]][tail[1]].setBackground(Color.WHITE);  //꼬리가 위치한 칸을 흰색으로 칠함
                    }
                    //2-2. 이동한 칸에 먹이가 있는 경우 -> 꼬리를 유지 (초록색 칠함), 새로운 위치에 먹이 생성
                    else {
                        //새로운 위치에 뱀의 먹이 생성
                        do {
                            foodX = (int) (Math.random() * N);
                            foodY = (int) (Math.random() * N);
                        } while(!isGoodPositionToMakeFood());

                        jLabels[foodX][foodY].setBackground(Color.RED);        //먹이의 위치 색칠
                        jLabels[tail[0]][tail[1]].setBackground(Color.GREEN);  //꼬리를 초록색으로 칠함

                        snake.offerLast(tail);                                 //꼬리 바로 앞 칸을 색칠하기 위해 빼냈던 꼬리를 다시 ArrayDeque에 집어넣음
                    }
                } catch (ArrayIndexOutOfBoundsException e) {
                    e.printStackTrace();
                }
            }
        };
        timer.schedule(task, 1000, 1000);
    }
```

4. 데모

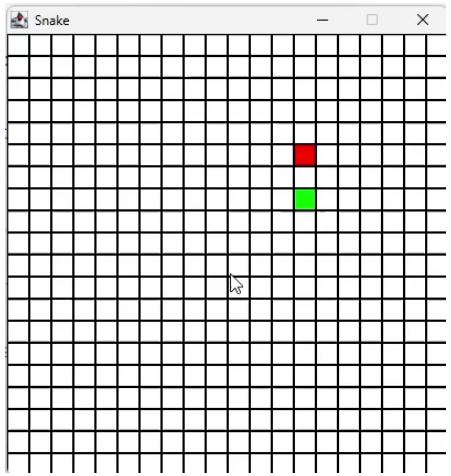
1. 먹이를 먹을 때



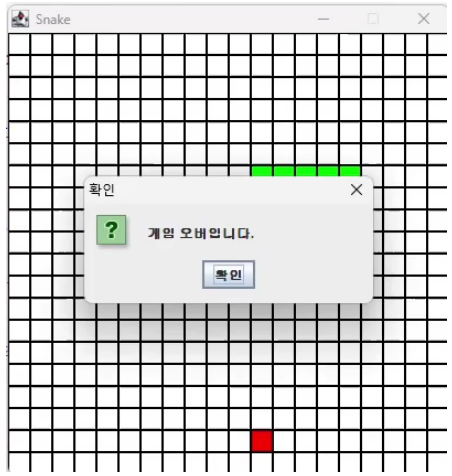
2. 자기 자신 충돌 (게임 종료)

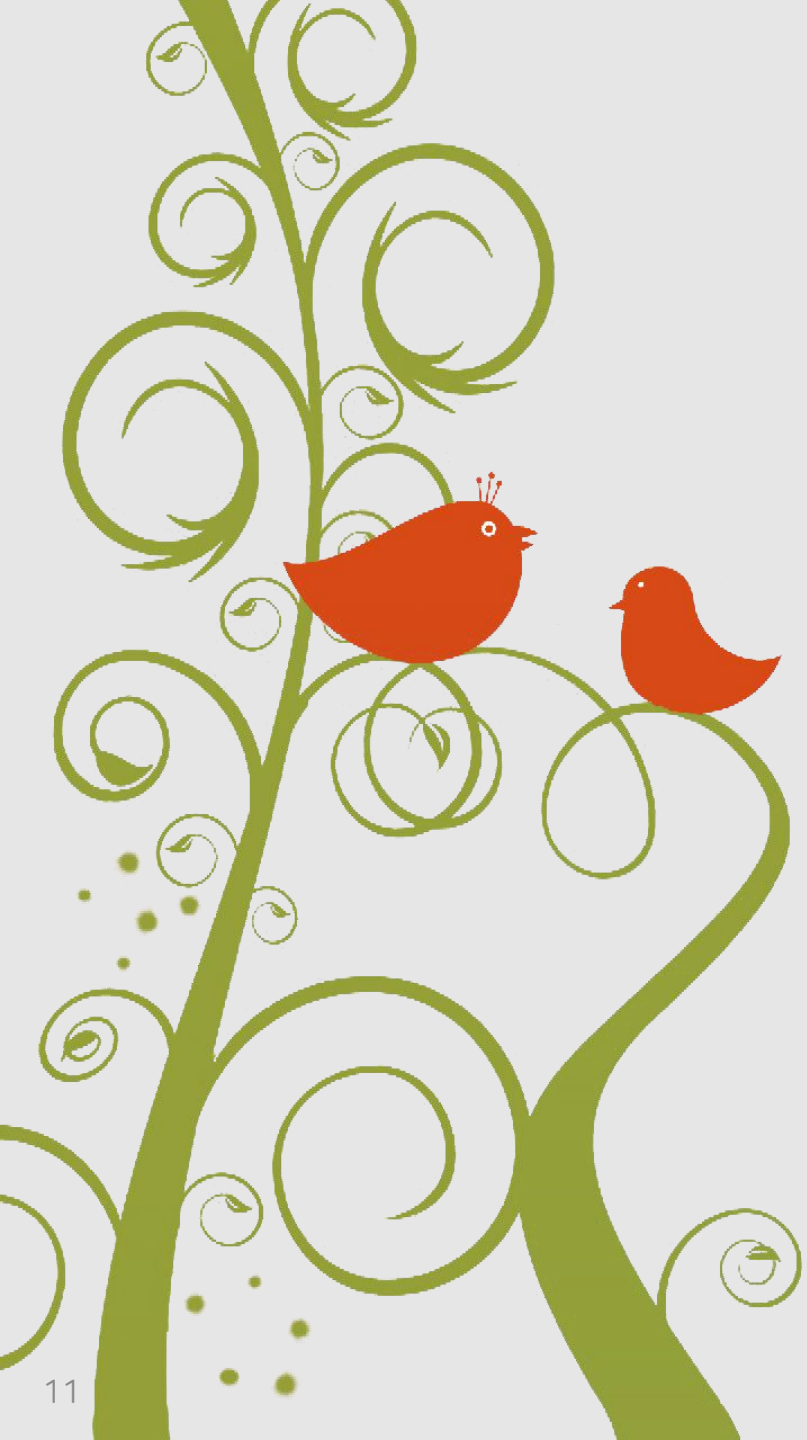


3. 벽 충돌 (게임 종료)



4. 메시지 창 확인 클릭 → 게임 다시 시작





감사합니다