컴퓨터 그래픽스
워밍업 Part 2

2020-2 2주차

3. 도형 간의 충돌 체크

- 비교할 2개 사각형의 좌표값을 [0, 500] 사이의 랜덤한 값으로 정한다.
  - 한 개의 사각형은 두 개의 좌표값으로 표시하고, 각 좌표값은 x와 y의 두 개의 정수값으로 표시한다.
  - 즉, 첫 번째 사각형의 (x1, y1) (x2, y2) 값과 두 번째 사각형의 (x3, y3) (x4, y4) 값을 랜덤하게 설정한다.
- 키보드 명령을 사용하여 두 번째 도형의 좌표값을 이동시킨 후, 첫 번째 도형과의 충돌 체크 여부를 출력한다.
  - 키보드 명령어 wasd: 위/좌/하/우 측으로 일정 길이만큼 이동
  - 도형의 좌표는 범위를 벗어나지 않는다. 범위를 벗어나면 에러메시지를 출력하고 이동하지 않는다.

출력 예)

```
(이동 Command: a(left) / d(right) /s(down) / w(up))
Input Shape Coordinates value:
        Shape 1: (100, 100) (300, 400)
        Shape 2: (400, 400) (500, 500)
Command: a                          // 두 번째 도형을 왼쪽으로 이동시킨다: x 좌표값을 –40 이동
        Shape 1:(100, 100) (300, 400)
        Shape 2: (340, 400) (440, 500)
Command: w                          // 두 번째 도형을 아래쪽으로 이동시킨다: y 좌표값을 -40 이동
        Shape 1:(100, 100) (300, 400)
        Shape 2: (340, 360) (440, 460)
Command: a                          // 두 번째 도형을 왼쪽으로 이동시킨다: x 좌표값을 –40 이동
        Shape 1:(100, 100) (300, 400)
        Shape 2: (280, 360) (380, 460)
        Rectangle 1 & Rectangle 2 collide!!
```

4. 저장 리스트 만들기 (구조체 데이터 사용)

- 점 (x, y, z) 데이터 값을 저장하는 리스트를 만든다. (점 데이터는 구조체를 사용하도록 한다.)
- 최대 10개의 점 데이터를 저장하도록 한다.
- 리스트에 데이터를 입력하거나 삭제하고 출력하는 명령어를 실행한다.
  - 각 명령어를 입력 받으면 결과 리스트를 아래의 표와 같이 항상 10개의 항목을 가진 리스트로 (인덱스와 데이터 값) 출력한다.

- 구현 함수 프로토타입과 명령어:
  - + x y z: 리스트의 맨 위에 입력 (x, y, z: 숫자)
  - -: 리스트의 맨 위에서 삭제한다.
  - e x y z: 리스트의 맨 아래에 입력 (명령어 +와 반대의 위치, 리스트에 저장된 데이터값이 위로 올라간다.)
  - d: 리스트의 맨 아래에서 삭제한다. (리스트에서 삭제된 칸이 비어있다.)
  - l: 리스트의 길이를 출력한다.
  - c: 리스트를 비운다.
  - m: 원점에서 가장 먼 거리에 있는 점의 좌표값을 출력한다.
  - n: 원점에서 가장 가까운 거리에 있는 점의 좌표값을 출력한다.
  - s: 원점과의 거리를 정렬하여 오름차순 (또는 내림차순)으로 정렬하여 출력한다. 인덱스 0번부터 빈 칸없이 저장하여 출력한다.
  - q: 프로그램을 종료한다.

** 리스트에서 맨 위(인덱스 9번)까지 차고 아래칸(인덱스 0번)이 비어있으면 다음 데이터 입력할 때는 0번에 입력된다. 즉, 10개의 항목을 다 채울 수 있어야 함.

4. 저장 리스트 만들기 (구조체 데이터 사용)

실행 예)

+ 0 1 0	→ 리스트 1번 출력, (리스트 각 항목 옆에 length 출력)
+ 0 1 1	→ 리스트 2번 출력, (리스트 각 항목 옆에 length 출력)
-	→ 리스트 3번 출력, (리스트 각 항목 옆에 length 출력)
e 1 1 1	→ 리스트 4번 출력, (리스트 각 항목 옆에 length 출력)
e 1 0 1	→ 리스트 5번 출력, (리스트 각 항목 옆에 length 출력)
+ 1 1 0	→ 리스트 6번 출력, (리스트 각 항목 옆에 length 출력)
d	→ 리스트 7번 출력, (리스트 각 항목 옆에 length 출력)
s	→ 리스트 8번 출력, (리스트 각 항목 옆에 length 출력)
m	→ (1, 1, 1) 출력
n	→ (0, 1, 0) 출력
l	→ 리스트 길이: 3

1

9	
8	
7	
6	
5	
4	
3	
2	
1	
0	0 1 0

2

9	
8	
7	
6	
5	
4	
3	
2	
1	0 1 1
0	0 1 0

3

9	
8	
7	
6	
5	
4	
3	
2	
1	
0	0 1 0

4

9	
8	
7	
6	
5	
4	
3	
2	
1	0 1 0
0	1 1 1

5

9	
8	
7	
6	
5	
4	
3	
2	0 1 0
1	1 1 1
0	1 0 1

6

9	
8	
7	
6	
5	
4	
3	1 1 0
2	0 1 0
1	1 1 1
0	1 0 1

7

9	
8	
7	
6	
5	
4	
3	1 1 0
2	0 1 0
1	1 1 1
0	

8

9	
8	
7	
6	
5	
4	
3	
2	1 1 1
1	1 1 0
0	0 1 0

5. 경로 만들기

- 50x50 크기의 2차원 배열을 만든다.
- 시작점에서 대각선에 놓여있는 끝점까지 랜덤한 경로를 설정하여 출력한다. (예, 좌측 상단 → 우측 하단)
  - 0은 길, 1은 벽 (혹은 반대로)으로 표시한다.
  - 이때, 길은 4방향인 좌/우/상/하로 연결되고, 다음의 조건에 맞게 길을 만든다.
    - ✓ 조건 1) 경로는 한쪽 방향으로 5칸 이상 계속 이동할 수 없다.
    - ✓ 조건 2) 경로는 좌우상하로 최소한 1번 이상 방향을 전환한 적이 있어야 한다.
  - 명령어: r - 경로를 다시 만든다. q – 프로그램을 종료한다.

출력 예) (샘플로 10X10으로 나타냈음)

0 0 1 1 1 1 1 1 1 1
1 **0 0 0** 1 1 1 1 1 1
1 1 1 **0** 1 0 0 0 1 1
1 1 1 **0** 1 0 1 0 1 1
1 1 1 **0 0 0 0 0** 1 1
1 1 1 1 1 1 1 **0** 1 1
1 1 **0 0 0** 1 **0 0** 1 1
1 1 **0** 1 **0 0 0** 1 1 1
1 1 **0** 1 1 1 1 1 1 1
1 1 **0 0 0 0 0 0 0 0**

6. 움직이는 도형 그리기

- 화면에 30x30크기의 보드를 그린다
  - 30x30의 각 칸에 .을 찍어 보드를 표현한다. (보드 크기는 변경 가능함).
  - 사용자에게 사각형의 두 개의 좌표값 (x1, y1) (x2, y2) 값을 입력받아 좌표값에 맞는 사각형을 특정 문자로 그린다.
- 명령어에 따라 도형을 이동, 확대/축소, 한 좌표 축으로의 확대/축소를 한다.
  - 입력 명령어: x/X - x축 우측/좌측으로 한 칸 이동
    y/Y - y축 아래쪽/위쪽으로 한 칸 이동
    s/S – x와 y 모두 한 칸 축소/확대 (범위를 넘어가는 축은 변경 안됨)
    i/j - x축 한 칸 확대/축소
    k/l - y축 한 칸 확대/축소
    r/q - 모든 것을 리셋하고 다시 좌표값을 입력받는다./ 프로그램을 종료한다.

출력 예) (샘플로 10x10 보드로 나타냈음)

input coord value: 2 2 5 6 (← (2, 2) (5, 6) 좌표값으로 이루어진 사각형)

```
. . . . . . . . . .
. 0 0 0 0 . . . . .
. 0 0 0 0 . . . . .
. 0 0 0 0 . . . . .
. 0 0 0 0 . . . . .
. 0 0 0 0 . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
```

6. 움직이는 도형 그리기

input order: x

```
. . . . . . . . . .
. . 0 0 0 0 . . . .
. . 0 0 0 0 . . . .
. . 0 0 0 0 . . . .
. . 0 0 0 0 . . . .
. . 0 0 0 0 . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
```

input order: s

```
. . . . . . . . . .
. . 0 0 0 . . . . .
. . 0 0 0 . . . . .
. . 0 0 0 . . . . .
. . 0 0 0 . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
```

** 구현된 파일은 각각 warming3.cpp ~ warming6.cpp로 저장하여 제출하기

** 각 문제는 3점씩 채점되고, 50% 이상 구현한 경우에 2점으로 채점됨

** 리드미 파일을 같이 첨부하여 각 문제에 대한 설명을 작성한다. 본인이 구현한 내용이나 명령어 등을 설명한다.