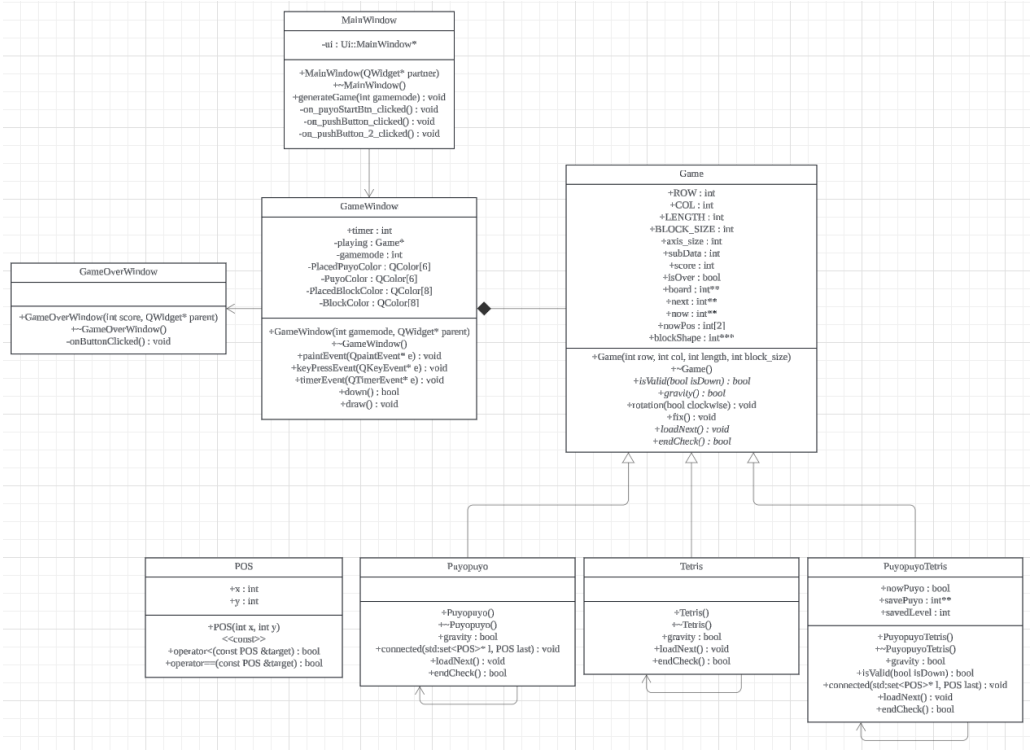
PuyoPuyoTetris Qt Project

2023202027 정정윤

A. 프로그램의 UML diagram



플레이어의 눈에 보이는 Window의 경우, MainWindow -> GameWindow -> GameOverWindow로 이전 Window에서 다음 Window를 생성하며 다루는 구조를 가지고 있다.

GameWindow는 사용자 입력을 받고 Game 클래스의 이런저런 멤버 함수를 사용하거나, 변수를 조작한다.

Game을 부모 클래스로 Puyopuyo, Tetris, PuyopuyoTetris클래스가 이를 상속받고 있다.

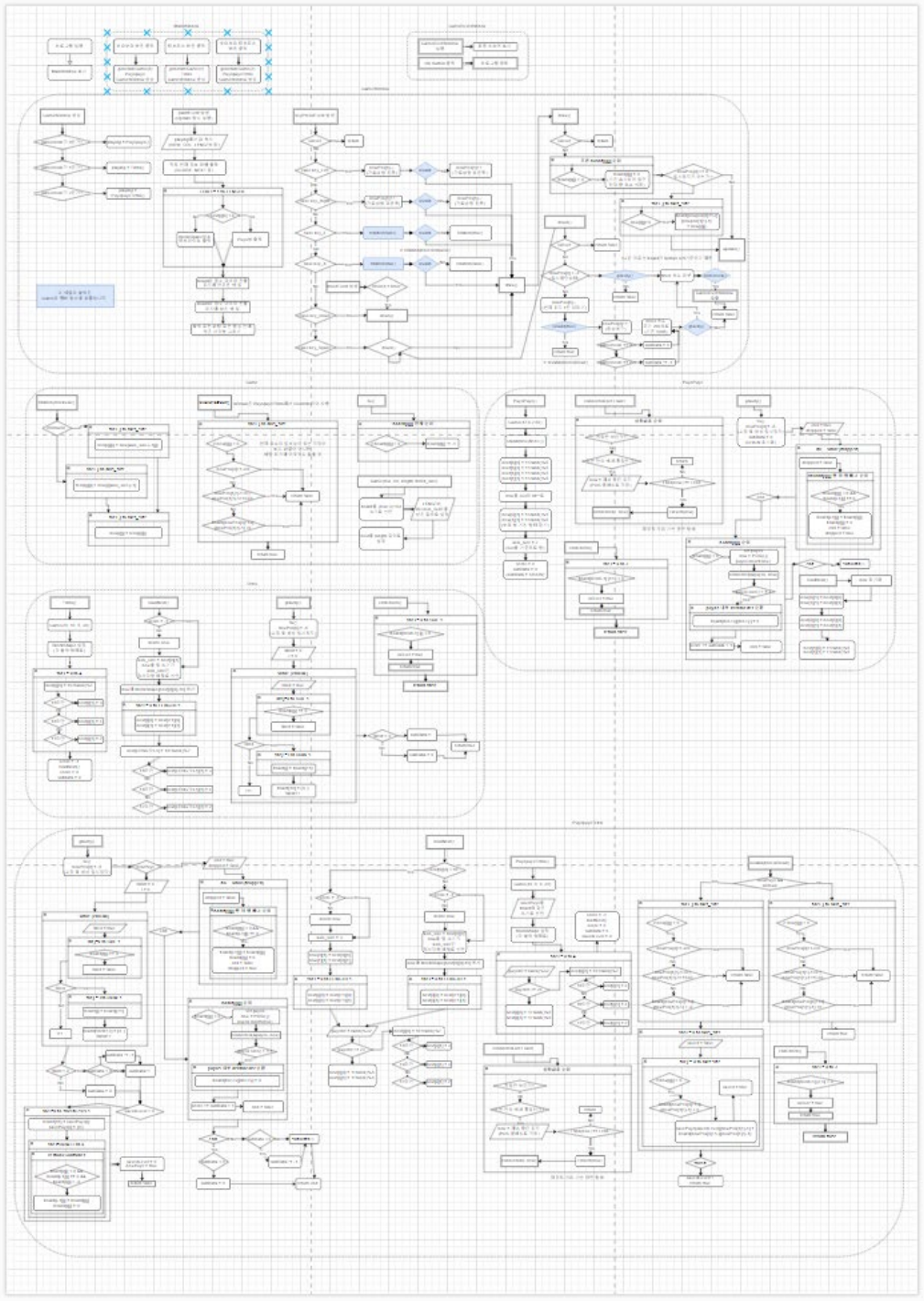
Game에는 뿌요뿌요, 테트리스, 뿌요뿌요테트리스에서 공통적으로 존재해야 하는 속성인 ROW, COL, LENGTH, BLOCK_SIZE의 기본적인 게임판 정보부터, 현재 요소의 크기인 axis_size, 점수 계산을 위한 보조 수단 subData, 점수인 score, 현재 게임오버 상태를 나타내는 isOver의 변수와 현재 게임의 상태를 본격적으로 표현하는 board, next, now

의 세 요소와 현재 요소의 위치를 나타내는 nowPos, 복잡한 블록 형태를 나타내기 위한 blockShape를 두었다. 또, 선언자 또한 기본적인 정보 입력까지 다시 작성할 필요가 없도록 미리 Game에서 작성하여 상속받도록 하고, isValid의 경우 여러 조건이 겹쳐 특수한 상황이 되는 경우(뿌요뿌요테트리스)가 아니라면 Game에서 선언해둔 그대로의 형태를 사용하게 되기 때문에, 각 Game 클래스에서 따로 선언하지 않고 상속받도록 하였다. 현재 조작 중인 요소를 회전시키는 rotation, 현재 조작 요소의 위치를 그자리에 고정하는 fix함수 또한 상속받아 사용할 수 있는 함수이다. 이런 상속을 통해 OOP의 Inheritance(상속성)을 만족시켰다.

각 게임별로 블록이 완전히 떨어졌을 때의 처리 방식이 다른데 이 부분 또한 같은 함수명을 사용하되 게임에 따라 다르게 작동하도록 virtual function을 통해 완전히 떨어졌을 때의 중력을 처리하는 gravity 함수, 각 게임별 종료 조건을 다룬 endCheck, next를 읽고 다음 요소를 불러오는 loadNext 함수, 뿌요뿌요테트리스에 한해 블록의 이동조건이 특수하기에 이런 부분을 처리할 수 있도록 isValid 또한 virtual function으로 되어 있다. 이런 virtual function들은 각 게임에서 오버라이딩하여 구현하도록 함으로써 OOP의 Polymorphism(다형성)을 만족시켰다.

function Pointer의 개념을 학습하기 이전에 작성한 코드라서 GameWindow에서 불러오게 되는 함수가 포함되는 경우, GameWindow의 멤버 함수로 구현되어 있는 문제가 있어 다소 Encapsulation이 잘 이루어지지 않았다고는 보고 있으나, 게임에 관련된 정보는 모두 Game이 가지고 있고, 단순한 조작(좌우 이동: nowPos[]를 증감)이나 Update와 같이 GameWindow에서 불러와야 하는 함수가 포함된 경우가 아니라면 게임에 관련된 정보의 처리(gravity, rotation, fix)를 하는 함수는 Game의 멤버 함수로 처리하도록 하여, Game에 관련된 정보를 건들이게 되는 요소를 가능한 최소화하여 Encapsulation(캡슐화)를 하고자 하였다.

B. **Flow Chart**



https://drive.google.com/file/d/1WpJ58s4NJ6j4Zd1WE4Los19Sb3jvbeor/view?usp=sharing

위 링크에 작성한 순서도가 모두 그려져있다.

각 순서도를 크게 둘러싸고 있는 부분의 위쪽에는 해당 부분이 어떤 부분의 순서도인지를 나타내고 있다. 각 기능의 작동을 합쳐놓은 작동 과정은 실행 검증에서 기술한다.

C. Pseudo Code

i. 전체 게임

```
<<키보드 입력>>
If Key_Left
        현재 가로 위치 – 1
If Key_right
        현재 가로 위치 + 1
If Key_Z
        현재 요소 회전(반시계)
If Key_X
        현재 요소 회전(시계)
If Key_Down
        <아래 이동> 실행
If Key_space
        While 불가능할 때까지 <아래 이동> 실행
화면 그리기

<<아래 이동>>
If 이미 게임이 끝났다면
        즉시 반환
If 현재 위치가 불가능하다면
        If 중력 적용이 끝났다면
                If 종료 조건에 맞다면
                        게임 오버 화면 실행
Else
        현재 요소 한 칸 아래로
        If 유효하다면
                성공 반환
        Else
                움직임 되돌리기
                If 뿌요뿌요라면
                        연쇄 초기화
                If 뿌요뿌요테트리스라면
                        연쇄 수치 음수화
```

If 중력 적용이 끝났다면
If 종료 조건에 맞다면
게임 오버 화면 실행
이동 실패 반환
이동 실패 반환

<<유효한 움직임 확인>>
For 현재 위치 (-1, -1) 부터 (+요소 크기-1, +요소 크기-1)
If 이 위치에 요소의 일부분이 속한다면
If 이 위치가 보드판 밖이라면
무효 반환
If 이 위치에 이미 무언가 있다면
무효 반환
유효 반환

<<고정>>
For 모든 보드판 구간 순회
if 현재 항이 양수라면
음수로 변경

ii. 뿌요뿌요

<<뿌요뿌요 실행>>
15 * 6 배열 board 선언
모든 항 0으로 초기화
2 * 2 배열 next 선언
for i, j in [2*2]
next[i][j] = 1~5의 무작위 수
<다음 요소 불러오기>
현재 요소 위치 (12, 2)로 설정
요소 크기는 3 고정
점수 0으로 초기화
연쇄 0으로 초기화

뿌요뿌요::<<다음 요소 불러오기>>
현재 요소에 next[0]을 적용
next인덱스 하나씩 줄이기
next의 마지막에 1~5의 무작위 수

뿌요뿌요::<<연결된 뿌요 확인>>
For 상하좌우 순회
    If 해당 칸의 색이 현재 칸과 같다면
        If 해당 칸의 위치가 집합에 없다면
            집합에 해당 칸의 위치를 삽입
            해당 칸에서 <연결된 뿌요 확인>
집합 반환

뿌요뿌요::<<중력 적용>>
do
    For 보드 전체 순회
        If 현재 칸이 0이 아니라면
            If 아래 칸이 0이라면
                아래칸으로 이동
While 이동한 칸이 있다면
위에서 이동한 적이 없다면
// 시간 딜레이를 위해서 바로 연결된 뿌요 처리를 하지 않아야 함
for 모든 보드판 칸 순회
    if 현재 칸이 0이 아니라면
        공집합 선언
        공집합에 현재 위치 삽입
        <연결된 뿌요 확인>
        If 집합에 4개 이상의 요소가 있다면
            해당 요소들 순회 : 0으로 만들기
If 한 번이라도 뿌요가 제거되었다면
    연쇄를 1 증가
If 한 번이라도 뿌요가 제거되었거나, 중력이 적용되었다면
    "끝 아님" 반환
"끝" 반환

뿌요뿌요::<<종료 조건 확인>>
If (11, 2)칸이 0이 아니라면
    게임 종료된 상태로 설정
    참 반환
거짓 반환

iii. 테트리스

<<테트리스 실행>>
테트리스의 모든 블록 형태를 blockShape에 저장
23*10 배열 board 선언
모든 칸 0으로 초기화
5*2 배열 next 선언
For next의 모든 항
    값 = 11~16의 무작위 수
다음 요소 불러오기
현재 요소 위치 (20, 4)로 설정
If 현재 요소가 L,J,S,Z,T라면
    요소 크기 = 3
If 현재 요소가 I라면
    요소 크기 = 4
If 현재 요소가 O라면
    요소 크기 = 2
현재 점수 0으로 초기화
현재 콤보 0으로 초기화

테트리스::<<다음 요소 불러오기>>
now초기화
if next[0]이 L,J,S,Z,T라면
    now는 3*3 배열
if next[0]이 I라면
    now는 4*4 배열
if next[0]이 O라면
    now는 2*2 배열
next[0]의 모양 blockshape 참조해 now에 집어넣기
next 인덱스 하나씩 줄이기

마지막 요소[0]에 11~16의 무작위 수 넣기
현재 모양 L,J,S,Z,T,I,O에 맞는 요소 크기 마지막 요소[1]에 넣기

테트리스::<<중력 적용>>
While 행 끝까지
    채워짐 = 참
    For 행의 모든 칸 순회
        If 빈 칸이라면
            채워짐 = 거짓
    If 채워짐
        For 행의 모든 칸 순회
            칸 비우기
        For 위로 모든 행 순회
            해당 행 칸들 아래 행으로 이동
    Else 다음 행으로
if 제거된 행이 하나 이상
    콤보를 1 증가
"끝" 반환

테트리스::<<종료 조건 확인>>
For 맨 위 행 순회
    If 칸이 0이 아니라면
        게임 종료된 상태로 설정
        참 반환
거짓 반환

iv. 뿌요뿌요 테트리스

<<뿌요뿌요테트리스 실행>>
테트리스의 모든 블록 형태를 blockShape에 저장
19 * 8 배열 board 선언
모든 칸 0으로 초기화
5 * 2 배열 next 선언
For (next의 모든 항)
    임시값 = 0~31의 무작위 수
    If 임시값이 24보다 크다면

```
            Next의 현재 항[0] = 10+rand%7
            If next의 현재 항[0] < 15
                Next의 현재 항[1] = 3;
            Else If next의 현재 항[0] < 16
                Next의 현재 항[1] = 4;
            Else if next의 현재 항[0] < 17
                Next의 현재 항[1] = 2;
        Else
            Next의 현재 항[0] = 1+rand()%5
            Next의 현재 항[1] = 1+rand()%5
<다음 요소 불러오기>
점수 0으로 초기화
연쇄 / 콤보 0으로 초 기화
저장된 뿌요 행 수 0으로 초기화

뿌요뿌요테트리스::<<다음 요소 불러오기>>
Now 초기화
If next[0]이 뿌요라면
        현재 상태를 뿌요로 설정
        요소 크기 = 3
        요소 크기에 맞게 now 배열 할당(정사각배열)
        Now[0][1] = next[0][0]
        Now[1][1] = next[0][1]
        Next 인덱스 하나씩 줄이기
Else
        현재 상태를 테트리스로 설정
        요소 크기 = next[0][1]
        요소 크기에 맞게 now 배열 할당(정사각배열)
        blockShape에서 현재 블록에 맞는 형태 복사해오기
        next 인덱스 하나씩 줄이기
임시값 = 0~31의 무작위 수
If 임시값이 24보다 크다면
        Next의 마지막 항[0] = 10~16의 무작위 값
        If next의 마지막 항[0] < 15
```

Next의 현재 항[1] = 3
Else If next의 현재 항[0] < 16
Next의 현재 항[1] = 4
Else if next의 현재 항[0] < 17
Next의 현재 항[1] = 2
Else
Next의 마지막 항[0] = 1~5의 무작위 값
Next의 마지막 항[1] = 1~5의 무작위 값
현재 위치를 (16, 3)으로 설정

뿌요뿌요테트리스::<유효한 이동 확인>>
아래 이동인지 여부를 인자로 받음
If 현재 상태가 테트리스이며, 아래 이동일 경우
For 현재 위치부터 요소 크기만큼의 주변 칸 순회
If(요소의 한 부분이 게임판 바깥으로 감)
무효
if 요소의 한 부분이 테트리미노와 겹침
무효
For 현재 위치부터 요소 크기만큼의 주변 칸 순회
If 요소의 한 부분이 뿌요뿌요와 겹침
` 저장된 뿌요 행에 해당 뿌요들의 열을 기록
유효한 이동 처리
저장된 뿌요 행 개수 1 증가
Else
For 현재 위치부터 요소 크기만큼의 주변 칸 순회
If 요소의 한 부분이 게임판 바깥으로 감
무효
If 요소의 한 부분이 이미 설치된 영역과 겹침
무효
유효한 이동 처리

뿌요뿌요테트리스::<<중력 적용>>
<위치 고정>
현재 위치(행)을 -5로 설정 (생성 일시 정지)

```
If 현재 상태가 뿌요라면
    Do
        For (board의 모든 칸 순회)
            If 현재 칸이 0이 아니고, 바로 아래 칸이 0이라면
                현재 칸의 정보를 바로 아래칸으로 이동
    While 한번이라도 칸의 정보를 아래칸으로 내린 적이 있다면
    If 한번이라도 칸의 정보를 아래칸으로 내린 적이 있다면
        For (board의 모든 칸 순회)
            if 현재 칸이 0보다 작고 -9보다 크다면
                공집합 puyos 선언
                Puyos에 현재 칸 위치 삽입
                뿌요뿌요::<<연결된 뿌요 확인>>
                If 집합 puyos의 크기가 4 이상이라면
                    For 집합 내 요소 순회
                        요소 좌표의 칸 = 0
        If 뿌요를 제거했다면
            If 연쇄 / 뿌요가 음수라면
                양수로 변환
            연쇄 / 뿌요 값 1 증가
        Else if 연쇄 / 뿌요가 음수라면
            0으로 변환
        If 한 번이라도 중력을 적용했거나, 뿌요를 제거했다면
            "끝 아님" 반환
    "끝" 반환
Else
    While 행 끝까지
        채워짐 = 참
        For 행의 모든 칸 순회
            If 빈 칸이라면
                채워짐 = 거짓
        If 채워짐
            For 행의 모든 칸 순회
                칸 비우기
            For 위로 모든 행 순회
```

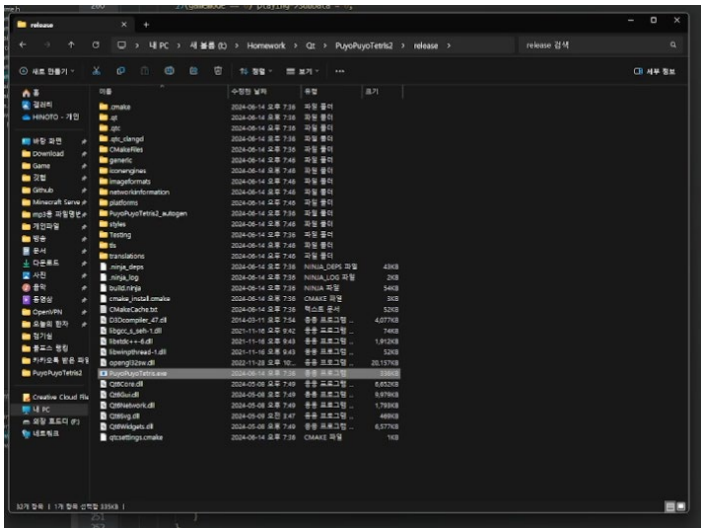
해당 행 칸들 아래 행으로 이동
Else 다음 행으로
if 제거된 행이 하나 이상
if 연쇄 / 콤보가 음수라면
양수로 변환
연쇄 / 콤보를 1 증가
Else 연쇄 / 콤보를 0으로 설정
If (저장된 뿌요 행 > 0)
For 저장된 뿌요 행 개수만큼
표시되는 보드 위 행에 저장된 뿌요 행 하나 불러오기
For 맨 위 행부터 모든 칸 순회
If 현재 칸이 0이 아니고 아래 칸이 0이라면
아래칸으로 정보 이동
현재 상태를 뿌요로 변환
"끝 아님" 반환
"끝 반환"

뿌요뿌요테트리스::<<종료 조건 확인>>
For 맨 위의 중앙 4칸 순회
If 현재 칸이 0이 아니라면
게임 종료된 상태로 설정
참 반환
거짓 반환

D. 실행 검증

[시연 영상]은 옆 글자를 클릭하면 확인할 수 있다.

<<기본 요구사항>>

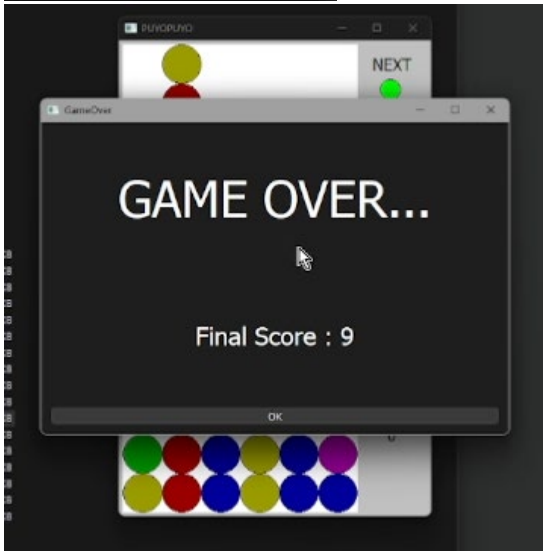


프로젝트 폴더에서 release 경로에 있는 PuyoPuyoTetris.exe 파일을 실행하면 프로그램이 실행될 것.

영상 **00:05**시점에서 release 폴더 내 PuyoPuyoTetris.exe 파일을 더블클릭하여 실행하는 것을 확인할 수 있다.

디렉토리의 "release" 디렉토리에 위치한 exe파일을 실행하여 프로그램을 실행할 수 있고, 어느 게임을 시작할 지 사용자가 선택할 수 있을 것

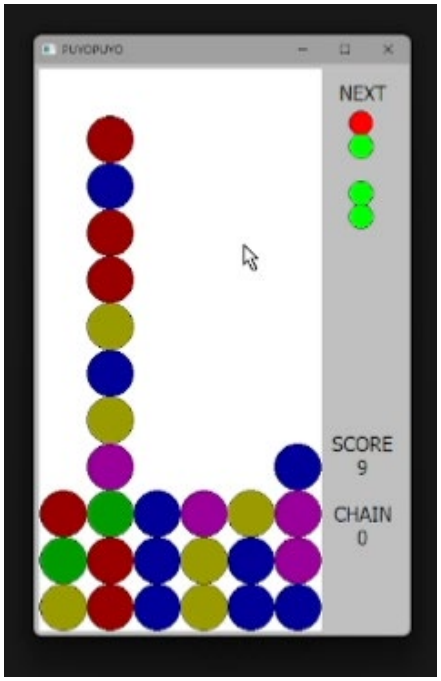
실행 시 켜지는 화면에서 어느 게임을 시작할 지 사용자가 선택할 수 있다. **영상 00:12 시점의** 뿌요뿌요 실행, **01:57** 시점의 테트리스 실행, **05:05**의 뿌요뿌요테트리스 실행을 확인할 수 있다.



게임 종료 조건을 만족시켰을 경우, 게임은 완전히 정지되며 게임이 종료되었음을 사용자에게 알림. 사용자가 확인하면 프로그램을 종료되어야 함

영상 01:53 시점에서 게임 오버 상황을 연출하며 OK버튼을 눌러 종료하는 것을 확인할 수 있다.

<<뿌요뿌요>>



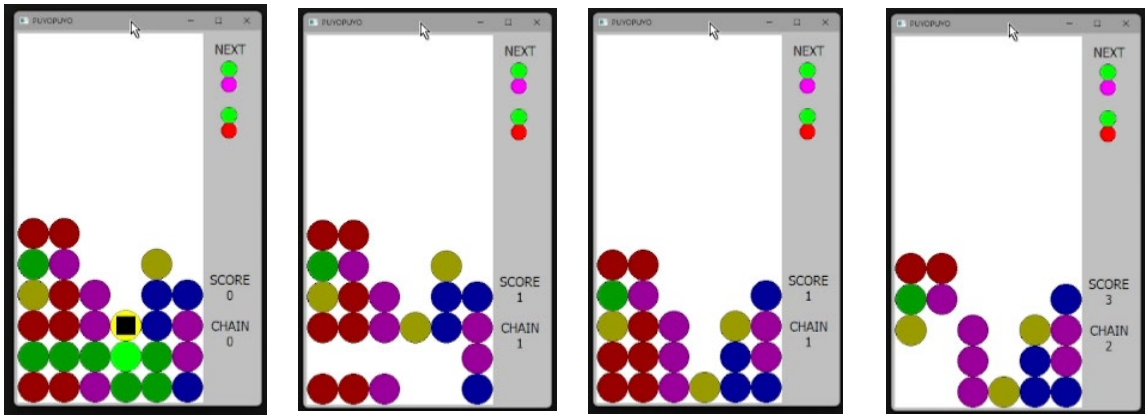
게임판 표시

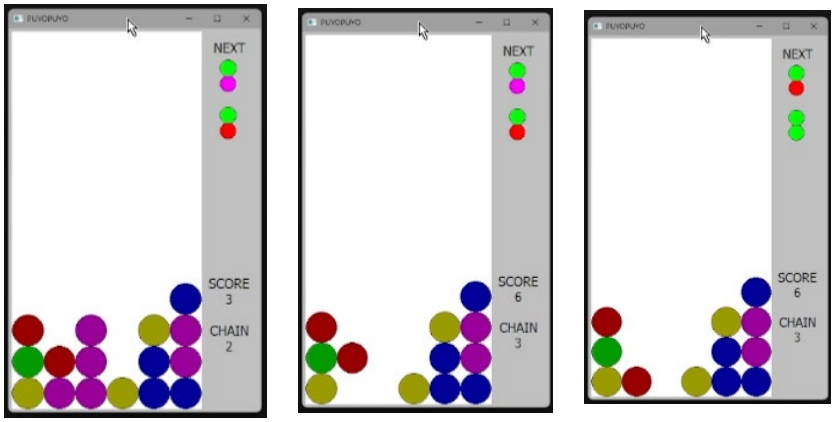
영상의 01:52 시점,

맨 위 빈 행을 포함하면 12개의 행, 6개의 열로 이루어진 게임 판이 시각화 되어 있다.

화면의 오른쪽에 순차적으로 생성 될 두 뿌요 쌍(본 사진에서는 빨강-초록, 초록-초록)이 시각화 되어 있다.

빨강, 초록, 파랑, 보라, 노랑색의 뿌요가 게임판에 다르게 시각화 되어 있다.

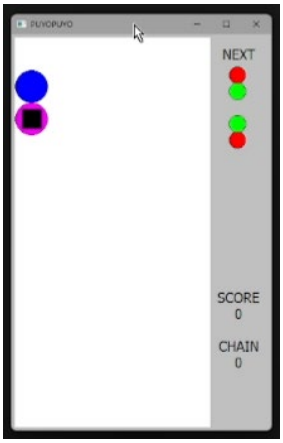




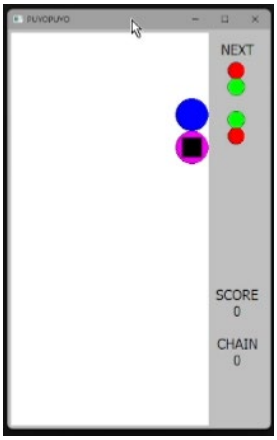
영상의 01:19 시점

첫 번째 사진과 같이 뿌요를 배치하고 순차적으로 화면이 진행된다.

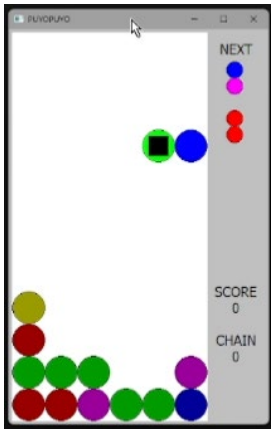
연속으로 뿌요가 제거됨에 따라 오른쪽 아래의 CHAIN의 숫자가 늘어나 연쇄를 시각화한다. 또한 다음 부분에서 새로 놓는 뿌요에 의해 연쇄가 다시 0으로 초기화되는 것을 확인할 수 있다.



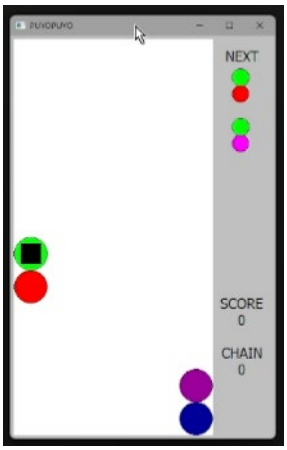
좌측 이동
(00:12)



우측 이동
(00:16)

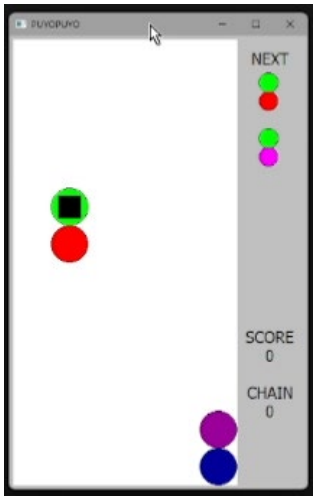


아래 이동
(00:48)



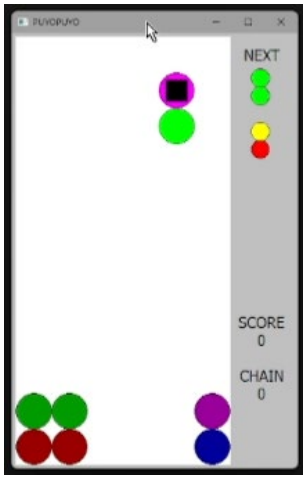
하드 드랍
(00:32)

이동은 요구 사항에 따라 4종류 구현되어 있다. 화면에는 표시되고 있지 않지만, 요구사항에 기록된 키보드 입력을 통해 해당 이동을 수행할 수 있다.



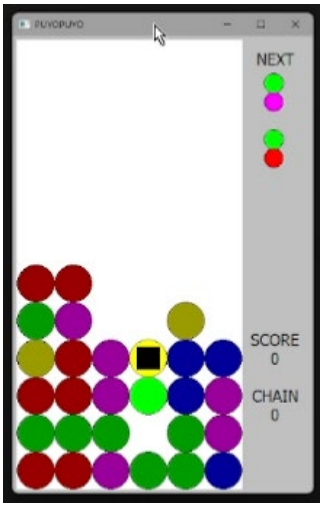
영상 00:28 시점,
X를 눌러 시계 방향으로 회전하는 것을 확인할 수 있다.

영상 00:36 시점
Z를 눌러 반시계 방향으로 회전하는 것을 확인할 수 있다.



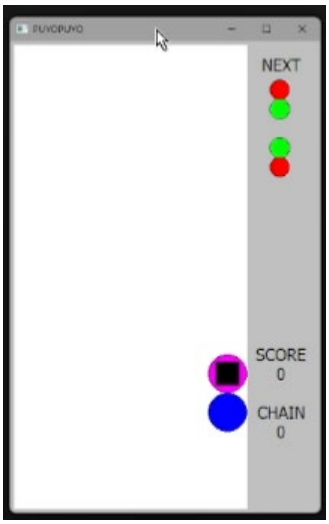
뿌요뿌요의 경우, 화면에 검은색 상자로 표시된 부분이 회전축이 되는 뿌요로, 회전 버튼을 누를 때는 해당 위치를 기준으로 회전함을 확인할 수 있다.

위 회전의 경우, 회전 하였을 때 현재 요소의 형태를 저장하는 곳에서만 데이터를 변경하고, draw 함수가 현재 위치에 조작 중인 뿌요 쌍의 현재 형태를 그리고 있을 뿐이기에 구현에 크게 어려움이 없었다.



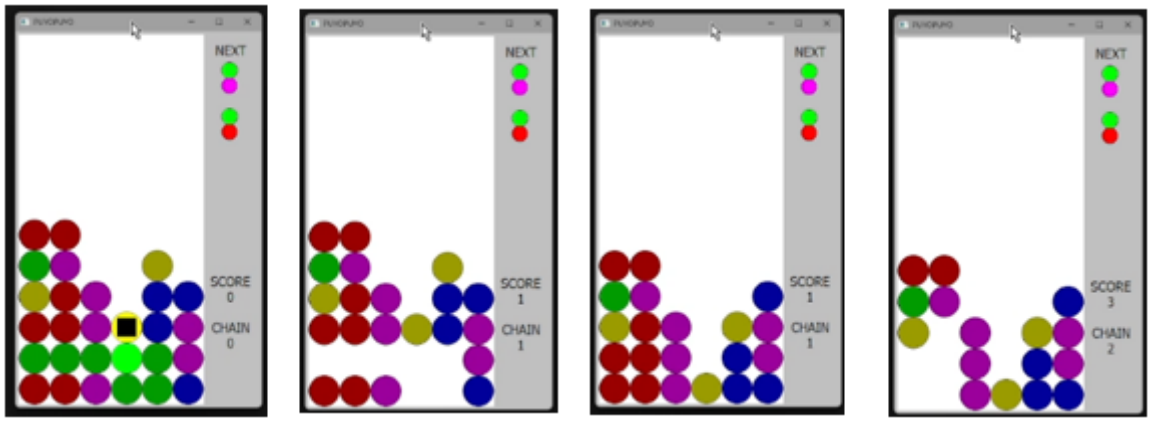
영상 01:18 시점,
조작하고 있는 뿌요 쌍은 왼쪽 이동 버튼과 오른쪽 이동 버튼을 누르고 있지만, 왼쪽과 오른쪽에 이미 다른 뿌요가 쌓여있기 때문에, 유효하지 않은 움직임으로 처리되어 움직여지지 않는다.

또한, 반시계 회전 버튼과 시계 회전 버튼 또한 누르고 있지만, 회전 시 위치에 다른 뿌요가 이미 존재하기에 유효하지 않은 움직임으로 처리되어 회전하지 않는다.



영상의 00:21 시점,
조작하고 있는 뿌요 쌍은 반시계 회전을 지속적으로 하는데, 벽에 부딪힌 시점에서 이동하지 않는다. 이는 회전을 수행하게 되면 게임 판 바깥으로 뿌요가 이동하기에 유효하지 않은 움직임으로 처리되어 회전하지 않는 장면이다.

Game 클래스의 isValid함수가 이를 담당하고 있으며, 조작키로 인해 이미 현재 위치 자체는 한 번 변경이 이루어지고, draw 함수를 통해 그려지지 않을 뿐이다. 이후 isValid함수를 실행하고, 이가 유효하지 않다고 반환할 경우, 해당 움직임을 취소하는 방식으로 유효하지 않은 움직임에 대한 차단이 구현되어 있다.



체인의 시각화에서 사용했던 자료이다.

영상의 01:19 시점에서 확인할 수 있다.

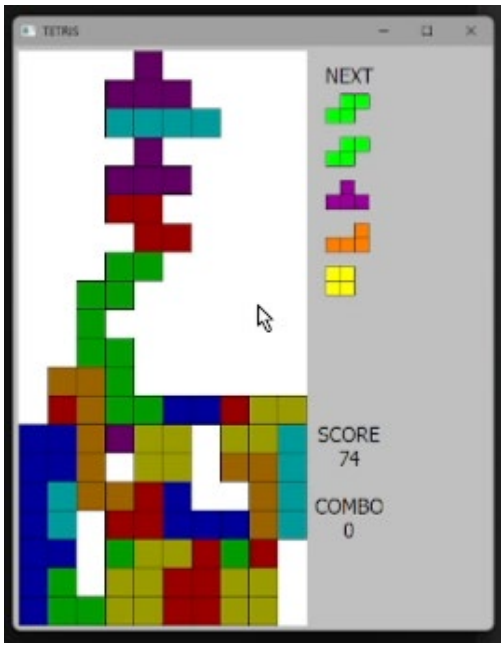
뿌요가 제거되자 이후 중력에 의해 모든 뿌요가 아래로 떨어지는 것을 확인할 수 있다. 떨어진 뿌요에서 다시 완전 탐색이 수행되며, 다시 뿌요를 제거한다.

본 기능은 Puyopuyo 클래스의 gravity 함수에서 중력의 적용을, connected 함수에서 완전 탐색을 통한 연결된 뿌요 확인은 하고 있다. connected함수의 경우, 재귀적으로 수행되며, 완전탐색을 수행한다.

영상의 01:52 시점에서는 게임 종료 위치가 (11,1)로 되어 있지만, 현재는 (11,2)로 수정해 둔 상태이다.

게임 종료 위치인 (11,2)에 뿌요가 쌓이고, 제거되지 않을 경우 게임이 종료된다.

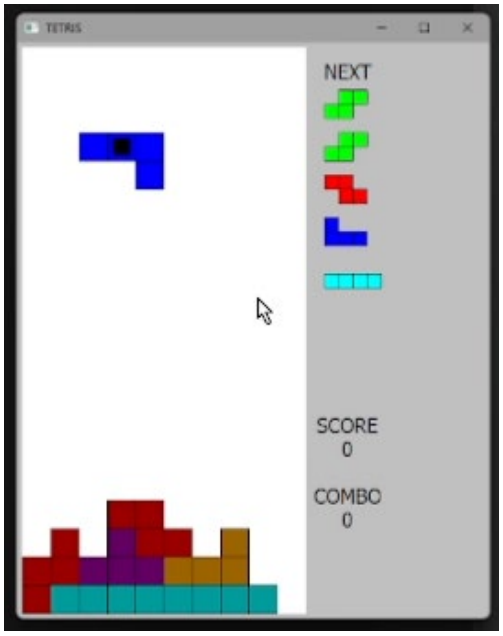
<<테트리스>>



게임판 표시

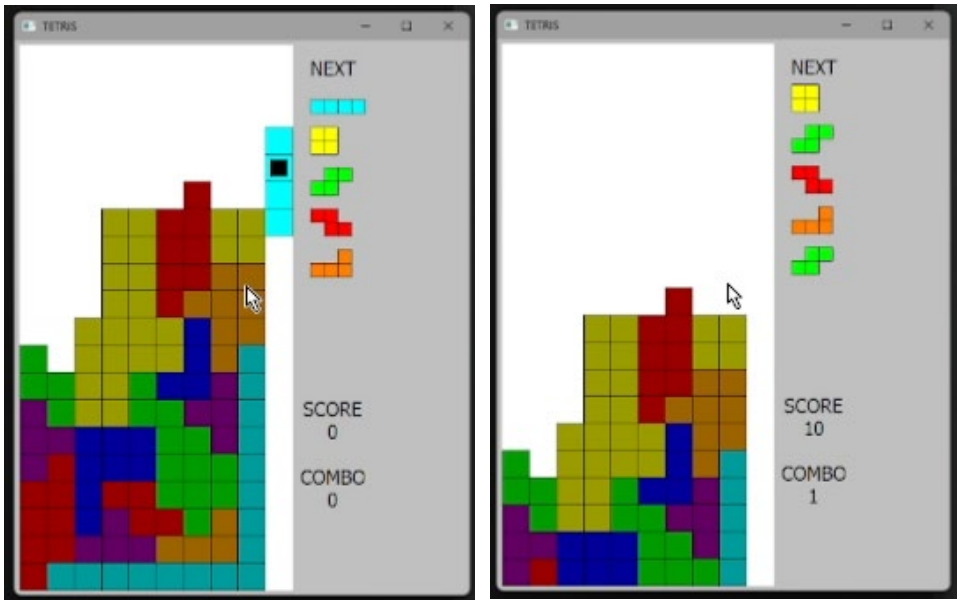
영상의 04:56 시점,
20 * 10의 게임판이 표시되고 있으며, 5개의 다음으로 출력될 테트로미노가 시각화되고 있다.

오른쪽 아래에 COMBO아래에 적힌 수치로 Line이 시각화 되고 있다. 자세한 사항은 아래에서 기술하겠다.

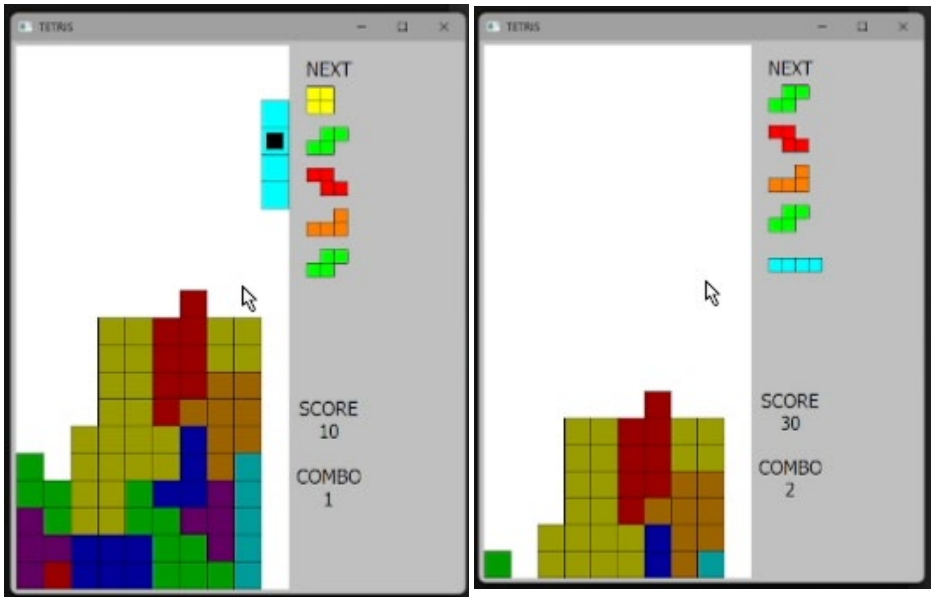


영상의 02:10 시점부터 움직임을 보면, 뿌요뿌요와 같이 좌, 우, 아래, 하드 드랍과 회전 또한 정상적으로 이루어지고 있음을 확인할 수 있다.

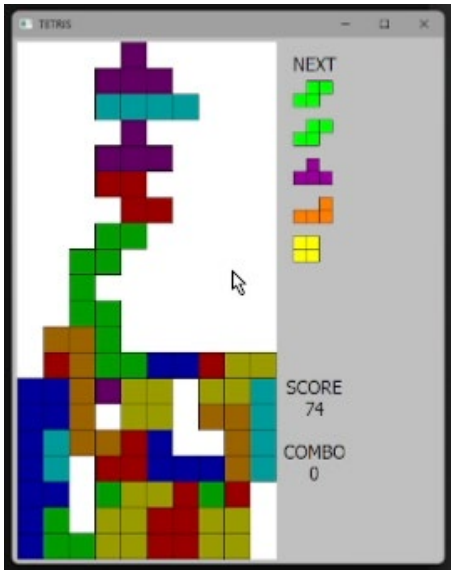
테트로미노의 회전은 3x3 블록의 경우 회전축이 중앙으로 설정되어 있어 이 지점을 기준으로 회전을 하게 된다.
I블록의 경우, 4x4 크기를 기준으로 (1,1) 지점에 축이 표시가 되어 있는데, rotation 함수를 통해 구현된 회전이 축의 -1, -1시점을 기준으로 잡고 회전하고 있어 부득이하게 이와 같이 표시하도록 하였다.
O블록의 경우, 회전이 의미가 없기 때문에 회전축은 표시되고 있으나 아무런 작용을 하지 않는다.



영상의 02:54 시점에서 I자 테트로미노를 해당 지점에 하드드랍함으로써 아래에서 4줄이 사라지는 상태가 되어 사라지는 줄은 제거되고, 중력이 적용되어 위에 있던 줄들이 하나씩 내려오게 된다.
영상의 03:52 경에서 I자 테트로미노의 하드드랍을 통해 맨 아래가 아닌 다른 줄의 제거를 통한 중력 적용도 확인할 수 있다.



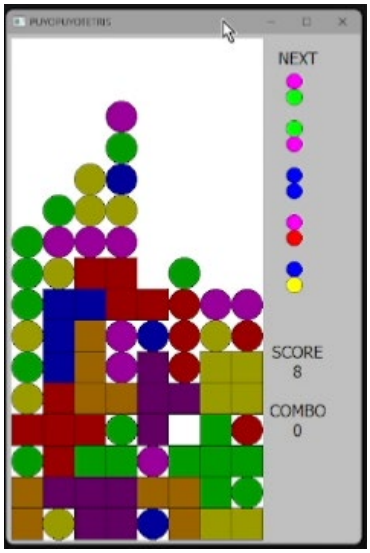
위 상태에서 COMBO가 1 상승해있고, 다음 I미노로 한 번 더 4줄을 제거하였다. 따라서, 연속으로 줄을 제거하여 COMBO가 1더 상승해 2가 된 것을 확인할 수 있다. 다음 장면에서는 제거하지 못하여 COMBO 아래 수치가 0으로 초기화되는 것 또한 확인할 수 있다.
위 기능들은 모두 Tetris 클래스의 gravity 함수에서 구현되어 있다.



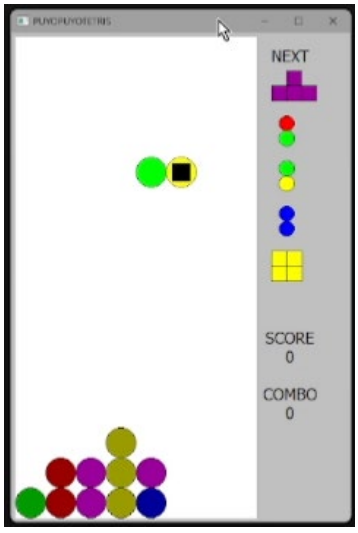
현재 상태에서는 종료 조건을 만족시키지 못한다. 다음 블록이 떨어져 T블록 위로 올라가야 조건이 만족되어 게임이 종료되게 된다.

해당 시스템의 구현을 위해 board는 bottom left부터 (0, 0)으로 좌표로 보고 있으며, 보드 위쪽에 추가로 board공간이 있으나, 화면에 표시되지 않는 방식으로 구현하였다.

<<뿌요뿌요테트리스>>

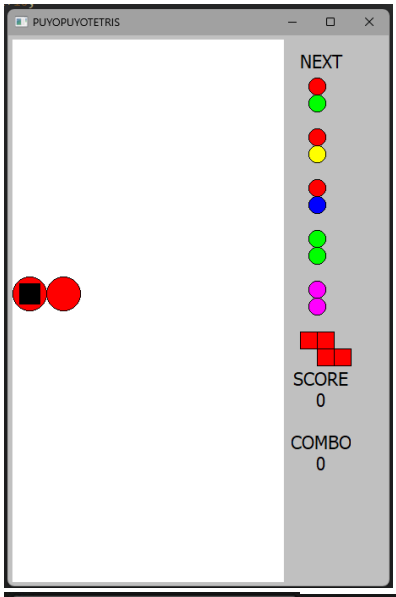


영상의 07:30 시점,
화면을 보면 8*16의 게임 판이 표시되고 있다.
또한, COMBO를 통해 현재 Chain & Line을 표시하고 있다. 뿌요는 원 모양, 테트로미노는 정사각형 모양으로 칸을 차지하며, 각각 원래 색상을 유지하며 보여지도록 되어 있다.

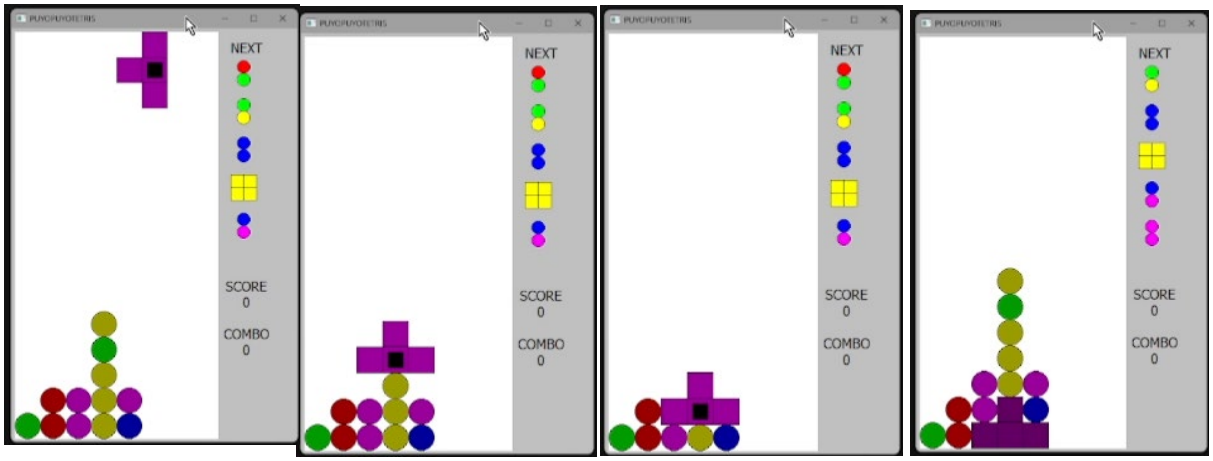


영상의 05:19 시점을 포함한 뿌요뿌요테트리스 초반부를 보면, 좌, 우, 아래, 하드 드롭 또한 정상적으로 구현되어 있음을 확인할 수 있다.

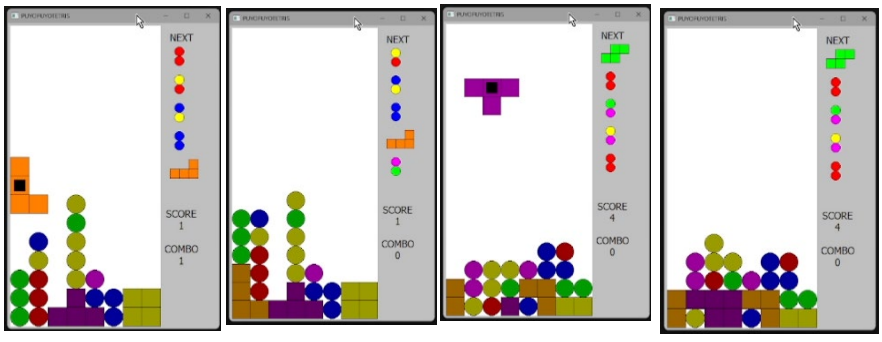
회전 또한, 기존의 방식 그대로 작동함을 알 수 있다.
뿌요뿌요, 테트리스, 뿌요뿌요테트리스의 블록의 움직임에 관련된 명령은 모두 한가지로 통일되어 있기 때문에, 유효한 움직임 처리를 제외한 나머지는 크게 달라지지 않는다.



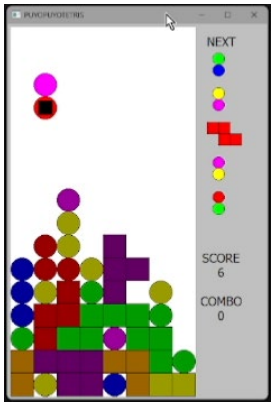
위 사진에서는 NEXT가 5개만 표시되고 있지만, 현재는 6개까지 표시하도록 정상적으로 수정된 상태이다.

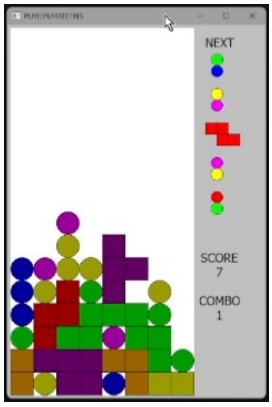


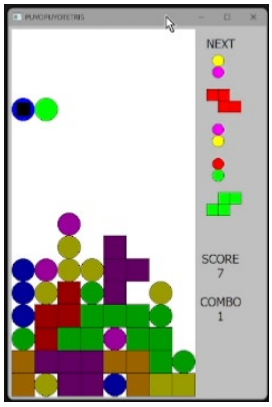
뿌요뿌요테트리스만의 유효한 이동이 있는데, 이는 테트로미노가 뿌요의 위로 올라가는 경우이다. 이 경우, 테트로미노가 "아래"로 이동하는 경우에만 허용하게 되며, 쌓여있는 뿌요 중 중간에 있는 뿌요를 제거하지 못하도록 좌, 우, 회전을 통해 뿌요가 있는 곳에 테트로미노의 일부가 자리하게 될 경우, 유효하지 않은 움직임으로 간주하고 움직임을 무효로 하도록 하였다. 이러한 과정을 통해 제거된 뿌요는 테트로미노의 중력 적용 이후 위에서 다시 내려와 테트로미노의 크기만큼 올라간 상태로 자리하게 된다.

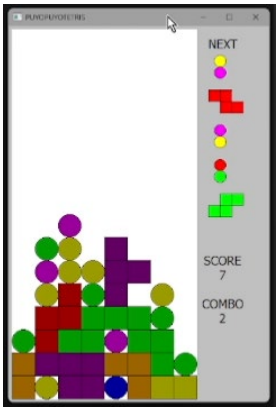


좌측 장면들과 같이 영상 중간중간에서 특수한 움직임 처리는 자주 확인할 수 있다.









위 4 장면은 영상의 06:40경의 시점에서의 연속된 장면이다. 조작으로 하드드랍 되어 적색 뿌요가 제거 되었고, 이후에 청색 뿌요로 다른 뿌요를 제거하였는데, 중간에 COMBO가 초기화되지 않는다. 뿌요뿌요테트리스는 테트리스와 같이 연속된 제거는 Chain & Line이 초기화되지 않도록 요구되어 있는데, 이 부분 또한 구현하였다. 뿌요뿌요 시스템과 테트리스 시스템을 동시에 적용하기 위해 이 점수를 다루는 부분을 일시적으로 음수로 바꾸어 뿌요뿌요측에서도 연속된 데이터 관리를 할 수 있도록 구현하였다.

영상 07:30 시점에서 뿌요가 게임 종료 지점에 도달하여 게임이 종료되는 것 또한 정상적으로 이루어짐을 확인할 수 있다.
모든 게임의 종료 조건이 다른 것은 virtual function을 통해 endCheck함수를 동일하게 실행함에도 동작하는 코드가 달라져 가능하도록 되어 있다.

E. 고찰

Reference Code확인과 예시 작동 확인 과정에서 Qt Creator사용에서 생기는 다양한 이슈들이 있었으나, 프로젝트를 새로 생성하고 해당 코드를 하나하나 붙여넣는 것으로 작동확인을 할 수 있었다. 쉽지 않은 과정이나 숙련자가 작성한 구조와 코드를 참고하는 것이 큰 도움이 된다는 사실을 알 수 있었다.

본 프로젝트로 작성한 코드의 기본적인 구조는 Reference Code에서 영감을 받았다. 이러한 클래스 구조와 객체지향 프로그래밍을 통한 코드 관리는 유지보수에서 매우 편리하다는 사실을 크게 느낄 수 있었다. 중간에 있었던 게임 종료 조건 관리 등에서 이를 크게 알 수 있었는데, 기존의 Top Left부터 생성하던 Board에 의해 테트리스의 종료 조건을 작성하기 힘들었는데, 이를 Bottom Left로 변경하는 과정이 크게 어렵지 않게 느껴질 정도였다. 하늘과 땅을 뒤집는 작업임에도 몇몇 메서드의 작동 값 변경만으로 이를 쉽게 관리할 수 있었다. 또, 게임별로 다르게 불러와야 했던 함수들도 virtual function을 사용하면 더 쉽게 관리할 수 있음을 생각하고 이용에 필요한 다양한 함수들을 virtual function으로 작성하였고, 관리의 편의성을 쉽게 확인할 수 있었다.