5조 이인 이동민

테트리스 use case

액터

* 플레이어

시스템

* 게임 자체

Use case 제목

* 게임 시작

설명

* 메인 화면에서 게임 시작을 위해 누르는 버튼.

사전 조건

* 게임이 (정상적으로)실행 중이다.

발생 조건(trigger)

* 플레이어가 게임 시작 버튼을 누른다.

케이스의 흐름

* 플레이어가 게임 시작 버튼을 누른다
* In-Game Scene으로 전환된다.

예외

* 게임 시작 버튼을 1회 이상 누를 수 있다.
* 게임 시작 버튼을 누른 직후 다른 버튼을 누를 수 있다.
* 게임 시작 버튼을 누른 직후 게임을 강제 종료를 할 수 있다.

예외 처리

* 게임 시작 버튼을 누르면 다른 행동을 취할 수 없게 한다.
* 강제 종료는 어쩔 수 없다. 변태 같은 플레이어..

끝나는 조건

* 게임이 In-Game Scene으로 전환될 때 끝난다.

결과

* Scene이 바뀌며 본 게임이 시작된다.

Use case 제목

* 게임 종료

설명

* 메인 화면에서 게임 종료를 위해 누르는 버튼.

사전 조건

* 게임이 (정상적으로)실행 중이다.

발생 조건(trigger)

* 플레이어가 게임 종료 버튼을 누른다.

케이스의 흐름

* 플레이어가 게임 종료 버튼을 누른다
* 게임이 종료된다.

예외

* 게임 종료 버튼을 1회 이상 누를 수 있다.
* 게임 종료 버튼을 누른 직후 다른 버튼을 누를 수 있다.
* 게임 종료 버튼을 누른 직후 게임을 강제 종료를 할 수 있다.

예외 처리

* 게임 종료 버튼을 누르면 다른 행동을 취할 수 없게 한다.
* 강제 종료는 어쩔 수 없다. 변태 같은 플레이어…

끝나는 조건

* 게임이 종료된다.

결과

* 게임이 종료된다.

Use case 제목

* 블록 이동

설명

* 블록이 화면에서 이동한다.
* 플레이어의 입력에 따라 좌우로 이동하거나 아래로 더 빨리 이동한다.

사전 조건(아래 항목이 충족되지 않으면 실행되지 않는다)

* 게임이 실행 중 이다.
* 게임 오버 상태가 아니다.
* 이 블록을 제외한 다른 이동하는 블록이 없어야 한다.
* 블록이 이동할 수 있는 공간이 있어야 한다. (좌 우 하단)

발생 조건(trigger)

* 플레이어가 (위를 제외한)화살표키를 눌렀을 때

케이스의 흐름

* 플레이어가 화살표키를 누른다.
* 블록이 해당 방향으로 이동을 한다.
* 아래 키를 눌렀을 시 누르고 있는 동안 블록의 낙하 주기를 50% 감소시킨다

예외

* 블록이 이동할 수 있는 공간이 없는 상태에서 화살표키를 누른다

예외 처리

* 시스템이 채워져 있는 공간을 판별하여 이동에 제한을 둔다..

끝나는 조건

* 플레이어가 눌렀던 화살표 키에서 손가락을 땐다.
* 블록이 바닥이나 다른 블록 위에 닿았을 때 블록의 낙하 주기가 지났을 시.

결과

* 블록이 플레이어의 입력대로 좌우 혹은 아래로 이동을 한다.

제목

* 블록 회전

설명

* 블록이 자기 자신의 회전 기준점을 기준으로 90도 반 시계 방향으로 회전한다.

사전 조건(아래 항목이 충족되지 않으면 실행되지 않는다)

* 게임이 실행 중 이다.
* 게임 오버 상태가 아니다.
* 이 블록을 제외한 이동하는 블록이 없어야 한다.

발생 조건(trigger)

* 플레이어가 키보드의 화살표 위 키를 누른다.

케이스의 흐름

* 화면에서 낙하중인 블록이 있다.
* 플레이어가 키보드의 화살표 위 키를 누른다.
* 블록이 자기 자신의 회전 기준점을 기준으로 90도 시계 방향으로 회전한다.
* 블록은 계속 낙하한다.

케이스의 흐름2

* 바닥 혹은 다른 블록 위에 붙은 상태이면서 고정이 되지 않은 상태이다.
* 플레이어가 키보드의 화살표 위 키를 누른다.
* 블록이 자기 자신의 회전 기준점을 기준으로 90도 시계 방향으로 회전한다.
* 블록이 회전된 상태로 바닥이나 다른 블록 위에 서있게 된다.

예외

* 좌우 벽에 붙어서 회전을 시도할 때.
* 다른 블록들에 의해 회전할 공간이 없는 상태에서 회전을 시도할 때
* 바닥이나 다른 블록 위에서 무한하게 회전을 시도할 때.

예외 처리

* 좌우 벽에 붙어서 회전을 시도할 때 벽의 안쪽에 현재 블록이 회전할 공간이 있으면 블록을 한 칸 벽 반대편으로 강제로 이동하면서 회전시킨다.
* 회전했을 때 현재 블록이 차지할 공간을 미리 예측하여 다른 블록이 차지하는 공간과 판별하여 서로 같은 공간을 공유하게 되면 회전을 못하게 막는다.
* 바닥이나 다른 블록 위에서 회전을 할 때 다시 위로 한 칸 뜨게 되는 경우가 생긴다. 이때, 무한 회전을 방지하기 위해 회전 횟수에 제한을 둔다. (5회) 회전 횟수를 다 채우면 플레이어의 의한 회전 불가 상태가 되며 바닥이나 블록 위에 고정된다.

끝나는 조건

* 1회 회전이 끝났을 때
* 5회 이상 회전해서 회전이 강제로 끝났을 때
* 회전 키를 눌렀지만 공간이 없다고 판별하여 회전이 안됐을 때

결과

* 블록이 90도 시계방향으로 회전한 상태가 된다.(안됐거나)

Use case 제목

* 블록 홀드

설명

* 현재 내려오는 블록을 보류하는 기능.

사전 조건

* 게임이 (정상적으로)실행 중이다.
* 게임 오버 상태가 아니다.
* 낙하중인 블록이 있다.

발생 조건(trigger)

* 플레이어가 홀드키 (shift 혹은 tab)를 누른다.

케이스의 흐름

* 플레이어가 홀드 키를 누른다
* 현재 내려오던 블록이 홀드 칸에 저장이 되고 목록에 있던 다음 블록이 낙하를 시작한다.
* 홀드 하는 시점에서 현재 블록의 위치와는 상관없이 교체된 블록은 다시 최 상단 에서 낙하하기 시작한다

케이스의 흐름2

* 플레이어가 홀드 키를 누른다.
* 현재 내려오던 블록이 홀드 칸에 있던 블록과 교체가 된다.
* 교체된 블록이 낙하를 시작한다.

예외

* 무한 홀드를 시도한다.

예외 처리

* 한번 홀드하면 하나의 블록이 고정되기 전에 다시 시도할 수 없다.

끝나는 조건

* 홀드키를 누른 후 홀드가 이루어졌든 안됐든.

결과

* 현재 블록이 다른 블록으로 변한다! 혹은 안변하거나.

Use case 제목

* 게임 재 시작

설명

* 게임 오버 화면에서 게임 재 시작을 위해 누르는 버튼.

사전 조건

* 게임이 (정상적으로)실행 중이다.
* 게임 오버 상태이다.

발생 조건(trigger)

* 플레이어가 게임 재 시작 버튼을 누른다.

케이스의 흐름

* 플레이어가 게임 재 시작 버튼을 누른다
* In-Game Scene을 다시 호출한다.

예외

* 게임 재 시작 버튼을 1회 이상 누를 수 있다.
* 게임 재 시작 버튼을 누른 직후 다른 버튼을 누를 수 있다.
* 게임 재 시작 버튼을 누른 직후 게임을 강제 종료를 할 수 있다.

예외 처리

* 게임 재 시작 버튼을 누르면 다른 행동을 취할 수 없게 한다.
* 강제 종료는 어쩔 수 없다. 변태 같은 플레이어…

끝나는 조건

* 게임이 In-Game Scene으로 전환될 때 끝난다.

결과

* Scene이 바뀌며 본 게임이 재 시작된다.

Use case 제목

* 게임 종료

설명

* 게임 오버 화면에서 게임 종료를 위해 누르는 버튼.

사전 조건

* 게임이 (정상적으로)실행 중이다.
* 게임 오버 상태이다.

발생 조건(trigger)

* 플레이어가 게임 종료 버튼을 누른다.

케이스의 흐름

* 플레이어가 게임 종료 버튼을 누른다
* 게임이 종료된다..

예외

* 게임 종료 버튼을 1회 이상 누를 수 있다.
* 게임 종료 버튼을 누른 직후 다른 버튼을 누를 수 있다.
* 게임 종료 버튼을 누른 직후 게임을 강제 종료를 할 수 있다.

예외 처리

* 게임 종료 버튼을 누르면 다른 행동을 취할 수 없게 한다.
* 강제 종료는 어쩔 수 없다. 변태 같은 플레이어…

끝나는 조건

* 게임이 종료된다.

결과

* 게임이 종료된다.

액터

* 게임

시스템

* 7-bag 시스템

Use case 제목

* 다음 블록 결정

설명

* 다음 블록이 뭐가 나올지 결정하는 과정.

사전 조건

* 게임이 (정상적으로)실행 중이다.
* 게임 오버 상태가 아니다.

발생 조건(trigger)

* 게임 최초 실행 시(In-game Scene).
* Next의 포인터가 2일 때
* Next의 포인터가 9일 때

케이스의 흐름

* 크기가 14인 단 방향 순환되는 데이터 구조(이하 배열)를 잡는다.
* 최초 실행 시 7-bag 시스템에 의하여 0~6번 배열에 7가지의 블록 중 임의의 블록을 한가지씩 대입한다.
* 게임이 시작하자마자 배열의 0번의 블록은 게임화면에서 낙하를 시작하고 Next 포인터는 1번을 가리킨다.
* Next 포인터가 가리키는 배열의 블록은 화면의 Next 칸에 출력한다.
* Next+1, Next+2, Next+3, Next+4 의 블록들은 그 아래의 대기 칸에 차례대로 출력한다.
* 블록 하나가 고정될 때 마다 Next 포인터를 1 증가시킨다.

케이스의 흐름2

* Next 포인터가 2일 때 7-bag 시스템에 의하여 7~13번 배열에 7가지의 블록 중 임의의 블록을 한가지씩 대입한다.
* Next 포인터가 가리키는 배열의 블록은 화면의 Next 칸에 출력한다.
* Next+1, Next+2, Next+3, Next+4 의 블록들은 그 아래의 대기 칸에 차례대로 출력한다.
* 블록 하나가 고정될 때 마다 Next 포인터를 1 증가시킨다.

케이스의 흐름3

* Next 포인터가 9일 때 7-bag 시스템에 의하여 0~6번 배열에 7가지의 블록 중 임의의 블록을 한가지씩 대입한다.
* Next 포인터가 가리키는 배열의 블록은 화면의 Next 칸에 출력한다.
* Next+1, Next+2, Next+3, Next+4 의 블록들은 그 아래의 대기 칸에 차례대로 출력한다.
* 블록 하나가 고정될 때 마다 Next 포인터를 1 증가시킨다.

예외

* 내 논리에 허점이 있을지도 모른다.
* 잘 모르겠다.

예외 처리

* 해봐야 안다 ㅠㅠ

끝나는 조건

* 최초 실행 시, Next가 2일 때, 9일 때 작동을 하고 끝난다.

결과

* Next와 그 이하의 칸들에 빈틈없이 계속 블록들이 출력된다.

액터

* 게임

시스템

* 공간 판별 시스템, 난이도 관리 시스템

Use case 제목

* 블록의 낙하

설명

* 블록이 화면에서 플레이어의 입력 없이 자동으로 낙하한다.

사전 조건(아래 항목이 충족되지 않으면 실행되지 않는다)

* 게임이 실행 중 이다.
* 게임 오버 상태가 아니다.
* 이 블록을 제외한 다른 이동하는 블록이 없어야 한다.
* 블록이 이동할 수 있는 공간이 있어야 한다. (하단)

발생 조건(trigger)

* 게임 시작 시
* 이 전의 블록이 낙하를 마친 시점.(이동 불가 상태가 됐을 때)

케이스의 흐름

* 블록의 기준점을 기준으로 하여 맵의 최상 단 5번째 칸에 Next에 있던 블록을 배치한다.
* 현재 난이도를 기반하여 낙하 주기(초기 1초)에 한 칸씩 아래로 떨어진다.
* 공간 판별 시스템에 의해 더 이상 내려갈 수 없는 상태(다른 블록의 위, 바닥에 닿은 상태)이면 낙하를 중지한다.

예외

* 플레이어의 입력에 의해 다른 블록의 위에 있다가 다시 낙하를 시작할 수 있다.

예외 처리

* 정상이다.

끝나는 조건

* 블록이 바닥이나 다른 블록 위에 닿았을 때 블록의 낙하 주기가 지났을 시.

결과

* 블록이 아래로 이동을 한다.

액터

* 게임

시스템

* 공간 판별 시스템

Use case 제목

* 줄을 없앤다

설명

* 가로줄 하나 혹은 그 이상이 블록으로 가득 차 없애는 것을 판단한다.

사전 조건

* 게임이 (정상적으로)실행 중이다.
* 게임 오버 상태가 아니다.

발생 조건(trigger)

* 가로로 한 줄 10칸이 블록으로 모두 채워지면.

케이스의 흐름

* 더 이상의 조작을 받지 않는 블록이 되었을 때 판별한다
* 맨 아랫줄부터 맨 윗줄까지의 가로줄을 모두 판별한다
* 가로줄이 가득 차 있는 것이 판별된 줄의 블록을 모두 제거한다
* 제거된 줄의 위에 남은 블록과 블록의 파편을 없앤 줄 만큼 아래로 이동시킨다.
* 블록의 제거는 항상 맨 아래부터 한다.

예외

* 없다.

예외 처리

* 없을듯.

끝나는 조건

* 가로줄이 제거된 후.

결과

* 가득 찬 가로줄이 모두 제거된다.

액터

* 게임

시스템

* 공간 판별 시스템

Use case 제목

* 게임 오버를 감지한다

설명

* 게임을 더 이상 진행할 수 없는 순간을 판단한다.

사전 조건

* 게임이 (정상적으로)실행 중이다.
* 게임 오버 상태가 아니다.

발생 조건(trigger)

* 조작하던 블록이 더 이상 조작 불가능해졌을 시점.

케이스의 흐름

* 더 이상의 조작을 받지 않는 블록이 되었을 때 판별한다.
* 맨 윗줄의 가로줄 중 한 개라도 고정된 블록이 있다면 게임 오버를 발생시킨다.

예외

* 맨 윗줄을 뚫고 나가는 블록의 일부

예외 처리

* 맵의 배열을 맵의 실제 보이는 크기보다 위로 3 크게 잡아서 뻑나는 일이 없도록 한다.

끝나는 조건

* 게임오버 상태일 때

결과

* 게임오버가 된다.

액터

* 게임

시스템

* 점수 시스템(콤보 포함)

Use case 제목

* 콤보와 점수를 판별하고 출력한다

설명

* 없앤 줄에 따라 콤보와 점수를 증가시킨다.

사전 조건

* 게임이 (정상적으로)실행 중이다.
* 게임 오버 상태가 아니다.

발생 조건(trigger)

* 공간 판별 시스템에 의해 줄이 제거 됐을 때.

케이스의 흐름

* 낙하하던 블록에 바닥이나 다른 블록에 닿아 고정된 블록이 된다
* 한 개 혹은 한 개 이상의 가로줄이 가득 찬다.
* 가득 찬 가로줄이 제거된다.
* 제거된 줄 수와 콤보 에 따라 점수를 증가시킨다.
* 콤보는 연속적으로 줄을 없앨 때 1씩 증가한다.
* 콤보는 2번째 콤보부터 화면에 보여준다.

예외

* 한 주기에 여러 줄의 블록이 제거된다

예외 처리

* 한 주기에는 무조건 콤보가 1씩 증가한다.
* 한 주기에 여러 줄의 블록이 제거된다면 점수 시스템에 따라 점수가 계산된다.

끝나는 조건

* 콤보와 점수를 판별하고 화면에 출력한 직후.

결과

* 콤보 수와 점수가 증가한다.

액터

* 게임

시스템

* 난이도 관리 시스템

Use case 제목

* 게임의 난이도를 조정한다.

설명

* 게임을 진행함에 따라 난이도를 증가시킨다.

사전 조건

* 게임이 (정상적으로)실행 중이다.
* 게임 오버 상태가 아니다.

발생 조건(trigger)

* 한 가로 줄을 없앨 시.

케이스의 흐름

* 한 가로 줄을 없앤다.
* 없앤 줄의 수가 난이도 관리 시스템이 지정한 라인 수보다 크면 난이도가 올라간다.
* 이때 블록이 떨어지는 주기가 10% 감소한다. (1.00초 -> 0.9초 -> 0.81 ...)

예외

* 최고 난이도에서 더 이상 빨라질 수 없을 때

예외 처리

* 그대로 둔다.

끝나는 조건

* 난이도 조정이 일어난 직후

결과

* Level이 한 단계 오르고, 블록이 떨어지는 속도가 빨라진다.