레퍼런스 분석 및 설계 - Candy Crush Friends Saga

서지호

0. 과제 수행 후 정리사항

1) 실행 방법 및 조건

- 압축된 프로젝트를 유니티에서 실행 후 Assets/Scenes/Stage0.scene을 실행하면 됩니다.
- 유니티 버전: 2020.2.1 f1
- 리소스는 메일로 받은 '과제용 블록 이미지'만 사용 (에셋 사용 X)

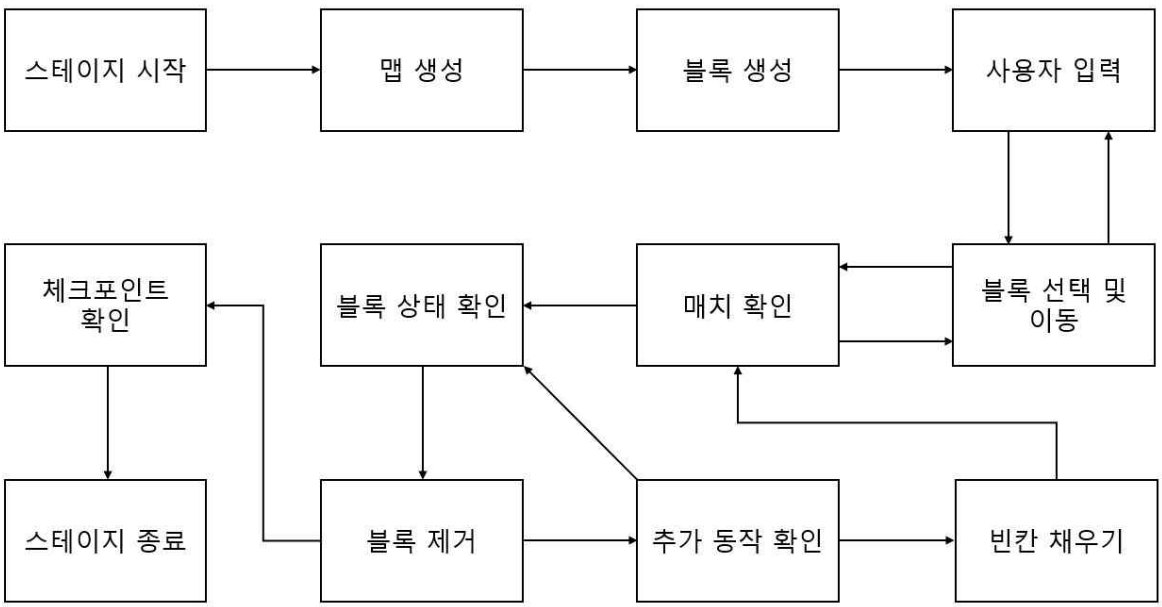
2) 구현된 내용

- 3매치 도는 사각형 매치가 없는 초기 보드 생성
- 레퍼런스 게임과 동일한 방식의 Swap 동작
    - 블록 드래그, 인접한 두 블록 선택, 블록 선택 후 보드 외부에서 드래그 시 작동
- 블록 이동 또는 파괴 도중 유저 입력 차단
- 3블록 또는 사각형 매치 시 블록 파괴
    - 4개 이상 매치 또는 사각형 매치 시 같은 색상의 특수 블록 생성
- 빈 공간에 블록 채워넣기

3) 미구현된 내용

- 특수 블록의 동작 (한 줄 지우기 또는 랜덤 좌표로 날아가 지우기)
- 새로 생성되는 블록에 대한 반복 파괴 동작

1. 게임 로직

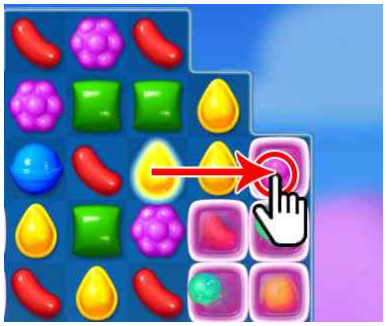


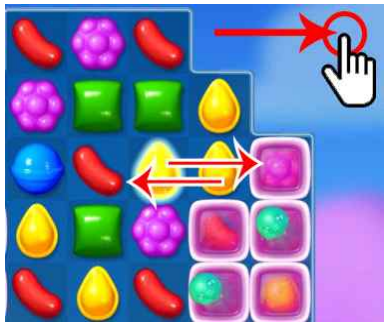
1) 단계별 요약

• 맵 생성: m,n 칸의 2차원 필드로 구성되며, 빈 칸이 존재할 수 있다.
• 블록 생성: 초기에는 3-매치 또는 사각형 매치가 일어나지 않도록 생성한다.
• 사용자 입력: 블록의 클릭 또는 스와이핑 동작을 감지한다.
• 블록 선택 및 이동: 블록을 선택 또는 선택 취소, 스왑 동작을 실행한다. 실행할 동작이 없으면 다시 입력을 대기한다.
• 매치 확인: 블록의 이동 동작 후 매치되는 요소가 있는지 검사한다.
• 블록 상태 확인: 블록이 파괴 가능한 유형 또는 상태인지 확인한다.
• 블록 제거: 블록 파괴 로직을 수행한다.
• 추가 동작 확인: 특수 블록에 파괴 로직이 입력된 경우 추가 동작을 수행하고, 파괴 유형에 따라 어라운드 매치 등으로 영향을 주는 요소를 확인합니다.
• 빈칸 채우기: 블록이 파괴된 칸이 있는지 확인하여 블록을 채워 넣습니다.
• 체크포인트 확인: 레퍼런스 게임과 같이 스테이지 클리어 조건을 확인합니다. 현재는 수행되지 않습니다.

2) 유저 입력 방식
• 같은 블록에서 마우스(터치)의 Down/Up 발생 시 선택된 상태가 된다.
• 선택이 가능한 블록일 경우 빛나는 이펙트가 나오며 선택되었음을 보여준다.

• 블록의 스왑 동작은 크게 3가지로 구분된다.





- 선택 여부와 무관한 인접한 두 블록의 드래그 동작
- 두 개 이상의 인접한 블록의 선택 동작
- 한 블록 선택 후 외부에서 상하좌우 방향으로 드래그 동작

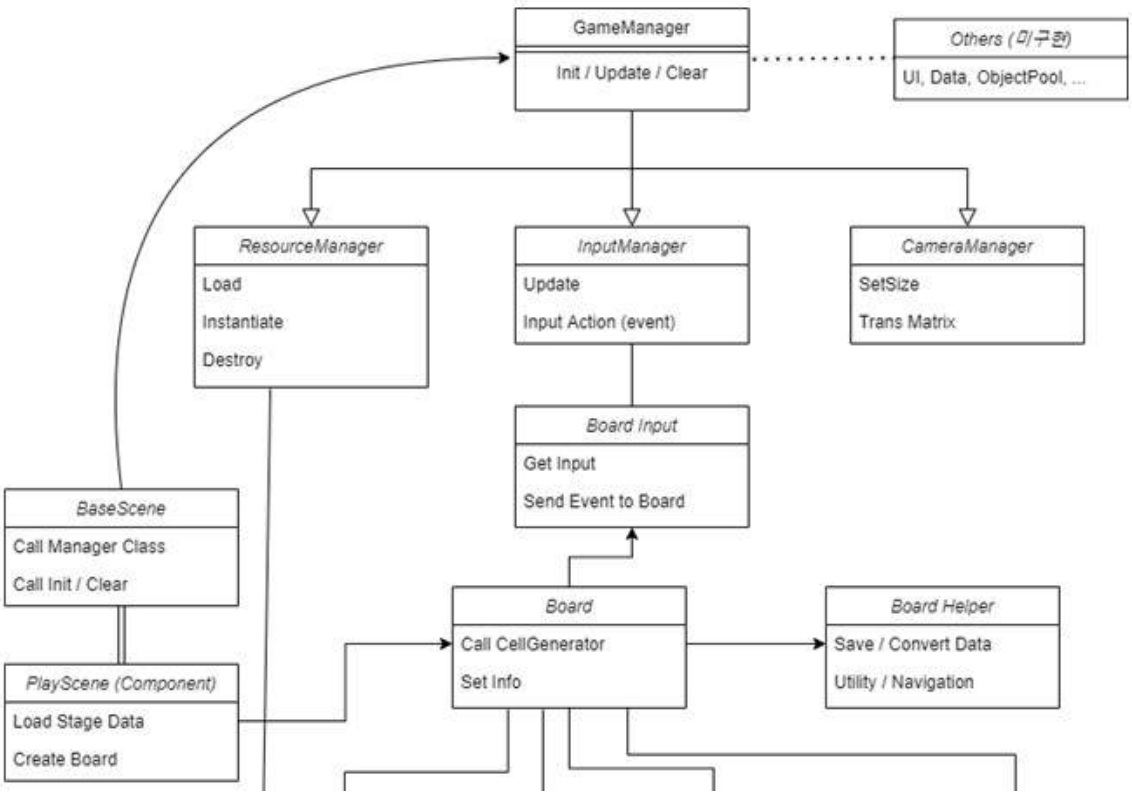
3) 블록 구성 장치들

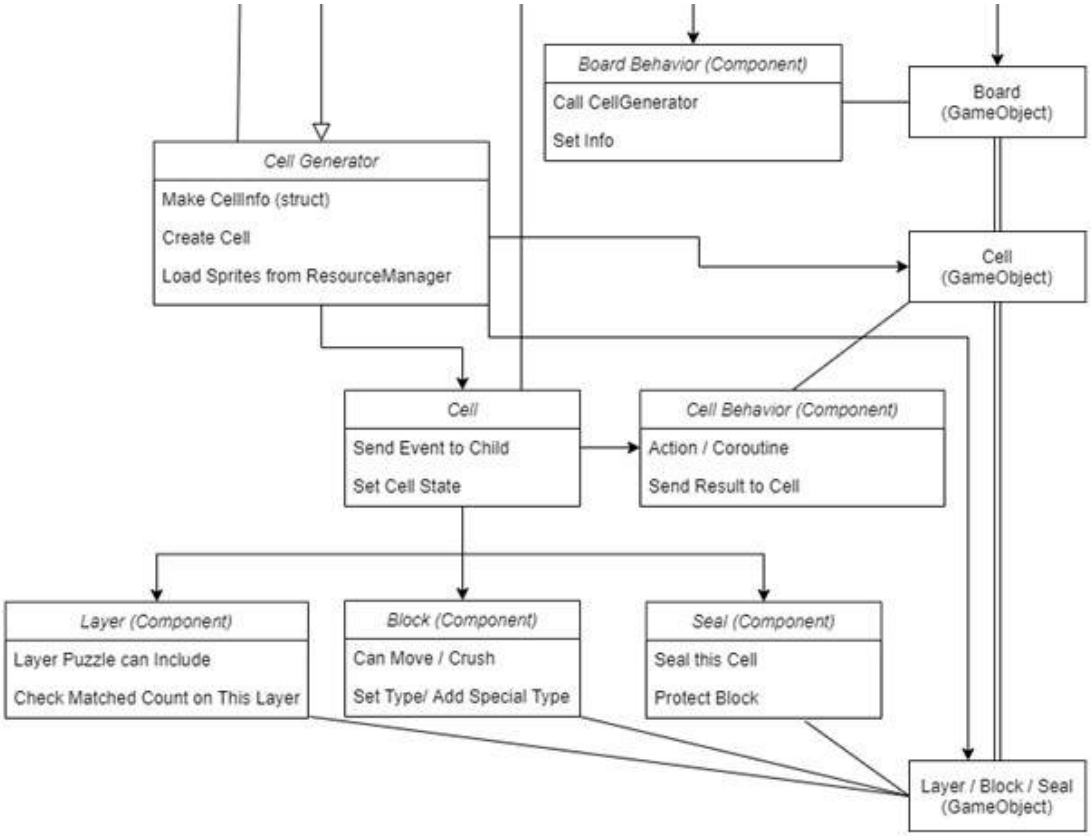
- 보드의 한 칸을 '셀(Cell)'이라고 할 때, 셀은 크게 세 가지 단계로 구성된다.
    - 블록에 영향을 주지 않지만 배경을 지우거나 칠하여 클리어 여부를 가르는 Layer
    - 유저 조작에 따라 칸 이동이 가능하며 매치 여부 확인의 기준이 되는 Block
    - 블록 위에 설치되어 파괴 또는 이동을 제한하는 Seal

- 따라서 위치가 고정된 좌표값 Cell 하나에 3가지 요소가 연결된 형태가 된다.
- 렌더링 순서는 Layer - Block - Seal 순이 된다.
    - 배경과 Cell의 기본 배경은 그 뒤쪽 레이어, 파티클 효과와 UI는 앞쪽 레이어에 위치

2. 클래스 설계

1) 스크립트 구조





2) 매니저 클래스

• GameManager: 싱글톤으로 최초 호출 시 GameObject가 생성되며, 다른 매니저 클래스를 생성하여 외부에서 쉽게 접근할 수 있게 하고, Initialize/Update/Clear(Release)를 담당한다.

• ResourceManager: Load/Instantiate/Destroy 등 리소스의 생성과 파괴 동작을 매핑한다. 자주 로딩되는 데이터는 저장하여 중복 로드 동작을 방지하고 최적화에 도움을 준다.

• InputManager: 유저 입력을 감지하여 이벤트를 발생시킨다. 스크립트마다 Update에서 입력을 감지하지 않아도 되게 설계한다.

• CameraManager: 기기의 종횡비에 따라 카메라 크기를 제어하고, 좌표계 변환 로직을 호출할 수 있게 한다.

• 그 외: 추후 파티클 효과나 UI, 애니메이션 등 추가 시 이를 관리할 클래스가 추가될 수 있음

3) 초기 세팅

• PlayScene -> Board -> Cell Generator -> Cell 다수 생성 순으로 진행된다.

• PlayScene: 플레이 씬의 다른 요소들을 불러오는 역할을 한다. 게임 씬에 유일하게 배치해야 하는 요소로 스테이지 번호를 입력해야 한다.

- Stage Data를 로딩해서 해당 정보를 통해 Board를 생성한다.

- 레퍼런스 게임에 수천 개의 스테이지가 존재하는 만큼, 각 스테이지 정보는 별개의 씬이나 이미지가 배치된 상태보단 간결한 데이터 형태로 관리되어야 한다고 판단함.

• Board: 게임이 진행되는 보드판을 나타낸다. 각 블록에 해당하는 Cell들을 저장하고 있으며, 유

저 입력을 받아 제어한다.
 - Cell Generator를 통해 Cell의 생성을 요청한다.
 - 입력/탐색/동작을 처리할 클래스를 생성하며, 동작을 관리하는 Behavior는 컴포넌트 형태로 게임오브젝트에 연결시킨다.

• Cell Generator: 정보를 입력받아 Cell 및 구성요소를 생성하는 정적 함수들이 들어 있는 클래스. 타입에 따른 이미지를 로딩하여 세팅하며 디자인 패턴상으로 Factory의 역할을 의도했다.

4) Cell의 구성 요소
• Cell: Board의 한 칸에 해당하는 공간 자체를 나타낸다.
 - 레퍼런스 게임과 같이 다양한 장치 표현을 위해 Layer, Block, Seal로 구성된다.
 - Cell에 들어오는 동작은 Cell Behavior에서 받아서, 현재 Cell에 할당된 Layer와 Seal의 제한사항 및 Block의 타입을 고려해서 처리한다.

• Layer
 - 해당 영역에서 매칭될 경우 Layer에 매칭 횟수가 증가한다.
 - 매칭 횟수에 따라 Cell 배경이 칠해지거나 드러나는 효과를 줄 수 있다.

• Block
 - 유저가 조작/이동시킬 수 있으며 매치 대상이 되는 요소
 - 타입을 가지며, 특수 타입이 아닐 경우 동일 타입은 동일 색상으로 묘사된다.
 - 특수 타입 속성(SpecialType)은 bit mask 형태로 관리되며, 여러 속성을 함께 가질 수 있다.
 ㄴ 예를 들어 레퍼런스 게임에서 '가로축 블록 모두 파괴' + '임의의 다른 블록으로 날아가 파괴' 속성을 동시에 갖는 경우가 있음

• Seal
 - Block 위에 추가적으로 생성된 장치로, 해당 Cell의 Block 이동이나 파괴를 제한한다.
 - Seal은 hp를 가지며, 파괴 로직을 전달받으면 Block을 파괴하는 대신 Seal의 hp를 감소시킨다.

• 참고: 파괴 방식 구분
 - 단순 매치로 파괴되는 경우 주변의 Seal이 파괴되지만, 임의의 Cell에 날아가 파괴시키는 특수 블록에 의한 파괴는 주변 Seal을 파괴시키지 않는다.
 - 파괴 유형에 따라 어라운드 매치된 블록 파괴 여부를 결정할 수 있어야 한다.



<< 1칸의 Seal만 파괴