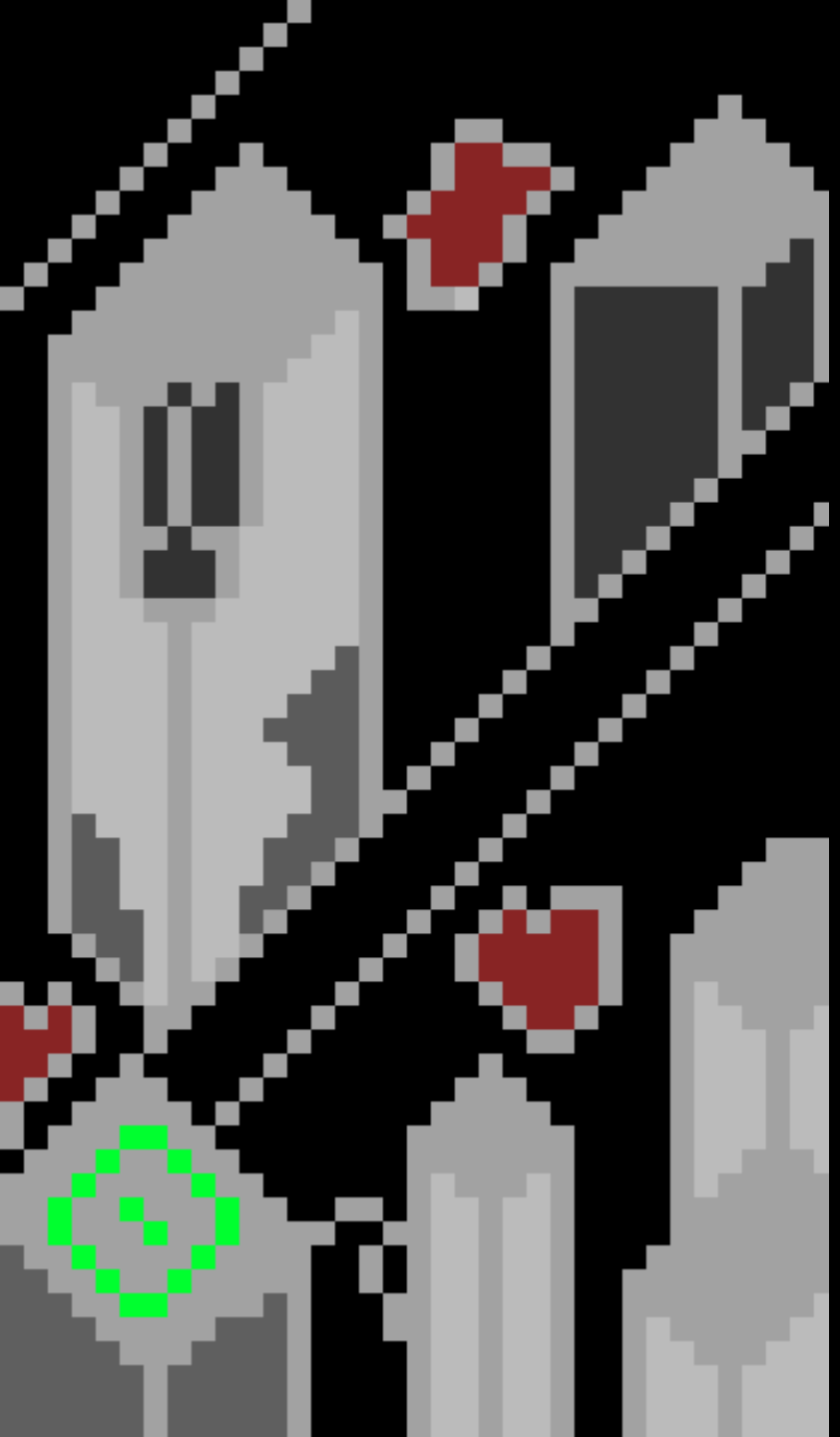


C++ 콘솔 포트폴리오 기획서

1999 ANOMIE

1999: ANOMIE

오가을

게임소개

1999: ANOMIE

1999년 1월 1일, 좀비 아포칼립스 세상의 대통령인 당신.
이 나라를 이끌 수 있을 것인가?

좀비가 가득한 아포칼립스 세상에서 시민, 국방, 종교의 통치를 통해 나라를 이끌어가는
경영 건설 시뮬레이션 게임

레퍼런스

SimCity NES
(Maxis, 1989)

건설 경영 시뮬레이션 게임의 원조라 불리는 심시티 시리즈의 첫 번째 시리즈,
심시티(1989)를 통해서 아이디어를 얻었음

레퍼런스

Yuppie Psycho
(Baroque Decay, 2019)

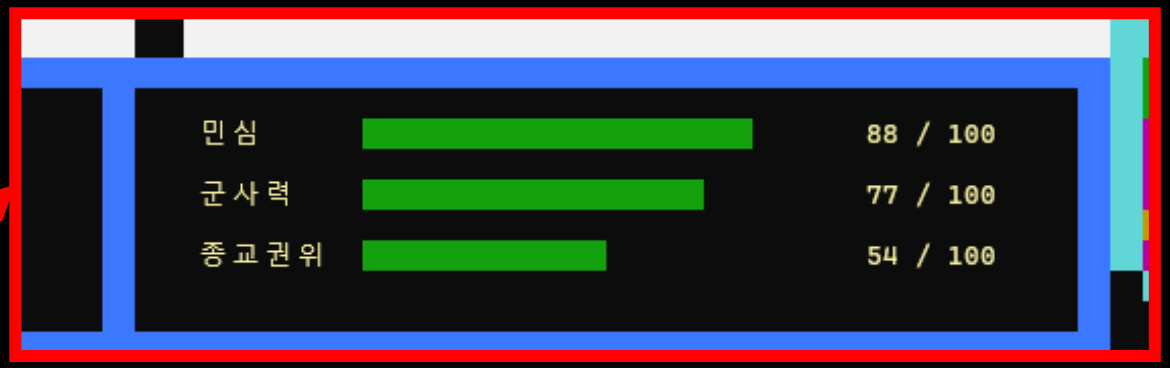
90년대 사이버펑크 디스토피아 분위기를
내고 싶어서 참고한 작품

女神転生
(ATLUS, 1987)

실제 90년대 전 후 게임 중
어두운 분위기를 가진 게임
(인게임 음악 사용)

파워

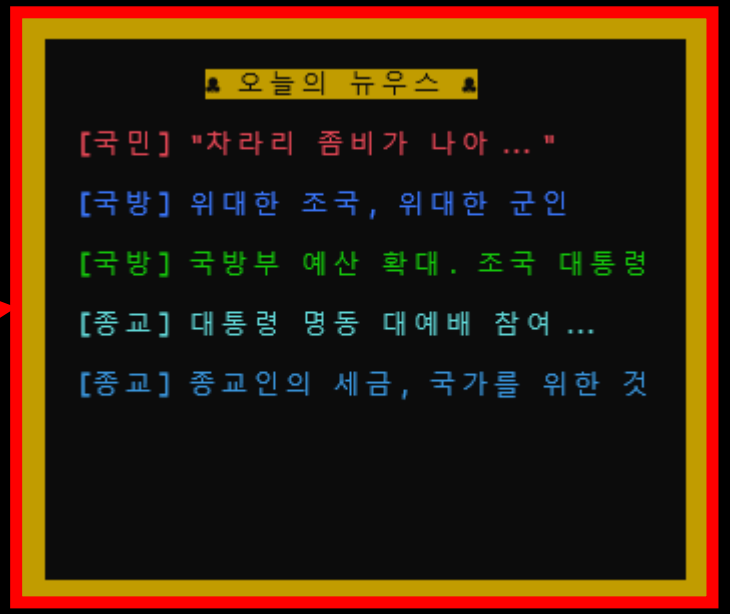
1999: ANOMIE 인게임 화면



- "파워"라고 불리는 민심, 군사력, 종교권위가 존재함
- 정부 예산(돈)과 파워의 밸런스를 유지하는 것이 핵심
- 초기 파워는 모두 50

뉴스탭

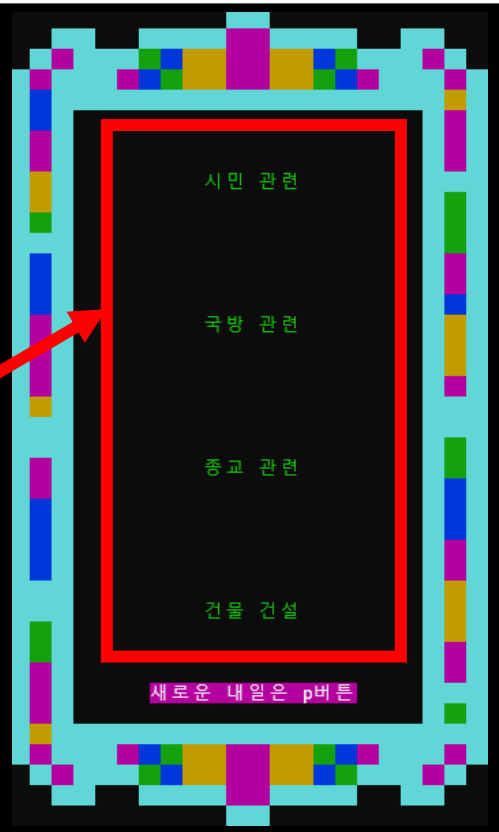
1999: ANOMIE 인게임 화면



- 전날의 행동, 엔딩 경고, 좀비 발생 등의 정보를 알려주는 뉴스탭
- 플레이어는 특정 행동의 결과, 엔딩 경고, 좀비 발생 등의 정보는 뉴스탭을 통해서 알 수 있음

대통령의 권한

1999: ANOMIE 인게임 화면



- 시민, 국방, 종교와 관련하여 대통령이 행할 수 있는 권한이 존재
- 이는 돈과 파워를 조절하는 역할을 함

대통령의 권한

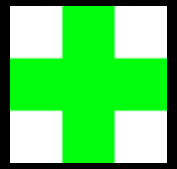
집단	제목	파워	비용(원)	특이사항
시민	강한 세금 징수	-10	+인구수 * 0.08	뉴스에 반영
	약한 세금 징수	-5	+인구수 * 0.05	뉴스에 반영
	국민 복지 정책 시행	+1이상 2이하 (랜덤)	-10,00이상 30,000미만 (랜덤)	뉴스에 반영
국방	군인 월급 삭감	-20	+인구수 * 0.5	뉴스에 반영
	국방 프로파간다	3/10 확률로 (프로파간다 실패) -20 (군사력), -20(민심)	-10,000이상 30,000미만 (랜덤)	뉴스에 반영 (실패 뉴스)
		7/10 확률로 (프로파간다 성공) +7이상 11이하 (랜덤)		뉴스에 반영 (성공 뉴스)
	국방부 예산 확대	+ 6이상 8이하 (랜덤)	-60,000원이상 90,000미만 (랜덤)	뉴스에 반영
종교	종교활동 금지(국민)	랜덤값(3이상 6이하)만큼 +민심, -종교권위	0	뉴스에 반영
	종교활동 금지(군인)	랜덤값(3이상 6이하)만큼 +군사력, -종교권위	0	뉴스에 반영
	종교활동 참여	+2이상 5이하	-10,000이상 30,000미만 (랜덤)	뉴스에 반영

건물 건설

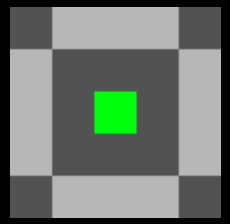
1999: ANOMIE 인게임 화면

- 시민, 국방, 종교와 관련된 건물을 건설 및 파괴할 수 있음
- 건물 건설 시, 관련 집단의 파워가 상승하고 파괴 시, 상승만큼 감소함
- 건물 파괴 시, 건설 금액의 절반만 돌려받음

건물 건설



병원



아파트

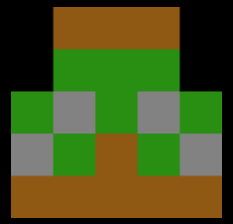
공원

(좀비) = 좀비 발생 시 해당

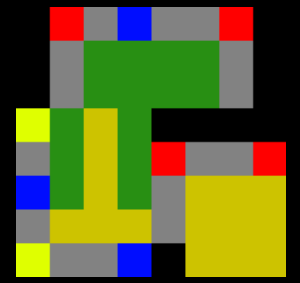
집단	건물명	파워	비용(원)	특이사항
시민	병원	+1	10,000	(좀비)건물 개수에 따라 뉴스반영 (좀비)병원 개수 * 0.001 * 전체인구수 만큼 추가 생존
	아파트	+3	70,000	(좀비)건물 개수에 따라 뉴스반영 (좀비)아파트 개수 * 0.002 * 전체인구수 만큼 추가 생존 (좀비) 종합(아파트의 중심좌표와 각 출입구와의 거리) * 0.001만큼 추가 사망
	공원	+5	30,000	(좀비)건물 개수에 따라 뉴스반영 (좀비)공원 개수 * 0.003 * 전체인구수 만큼 추가 사망

건물 건설

부대(소)



부대(중)



부대(대)

(좀비) = 좀비 발생 시 해당

집단	건물명	파워	비용(원)	특이사항
국방	부대(소)	+1	50,000	(좀비)건물 개수에 따라 뉴스반영 (좀비) 부대(소) + 부대(중) * 2 + 부대(대) * 5를 ArmyGauge라고 했을 때, ArmyGauge * 0.0007 * 인구수만큼 추가 생존
	부대(중)	+3	120,000	(좀비)건물 개수에 따라 뉴스반영
	부대(대)	+5	200,000	(좀비)건물 개수에 따라 뉴스반영

건물 건설

교회

성당

사이비

(좀비) = 좀비 발생 시 해당

집단	건물명	파워	비용(원)	특이사항
종교	교회	+2	70,000	(좀비) 건물 개수에 따라 뉴스에 반영 (좀비) 종교 건물 개수 * 0.0001 * 인구수만큼 추가 생존 (좀비) 최대 거리(163) - 아파트의 중심좌표와 종교 건물 중심좌표의 거리 * 인구수 * 0.001만큼 추가 생존
	성당	+2	70,000	
	사이비	-10(민심) -10(종교권위)	-100,000	사이비 생성 시, 뉴스에 반영 종교건물개수 * 2 /100의 확률로 종교 건물 하나가 사이비가 됨

좌표 간의 거리 공식

- 각 출입구의 중심 좌표를 (x_n,y_n)이라 하고 총 아파트의 중심좌표를 (p,k)라고 했을 때, 총 거리의 합 S를 다음과 같이 정의함

$$S = \sum_{i=1}^{n} \sqrt{(x_n - p)^2 + (y_n - k)^2}$$

- 총 아파트의 중심좌표를 (p,k)라고하고 총 Religion의 중심좌표를 (r,t)라고 했을 때, 두 좌표사이의 거리는 D 다음과 같이 정의함

$$D = \sqrt{(r - p)^2 + (t - k)^2}$$

■ 좀비 발생 시에는 해당되지 않음

파워 종류	특이사항
민심	민심이 절반보다 높을 때 전체 인구수 * 0.01 * 민심률 만큼 인구수 증가 민심이 절반보다 낮을 때 전체 인구수 * 0.01 * 민심률 만큼 인구수 감소 인구수 * 0.1만큼 세금 납부
종교권위	종교권위가 50 이상일 때, 인구수 * 종교권위 * 100만큼 돈 증가 세금 납부 여부가 뉴스에 반영됨

좀비 발생일

- 10일 주기로 100% 확률로 발생
- 1999년 1월 6일 이후부터 발생
- 10일 주기 외의 날에는
  10 / (군사력 + 1)의 확률로 발생

비통함

- 비통함 = 인구수 / 전체 사망자 수
- 비통함이 5 미만일 때, 파워에 영향을 주지 않음
- 비통함이 5 이상일 때, 비통함 *2만큼
  민심, 군사력, 종교권위가 감소됨
- 비통함은 뉴스에 반영됨

사망자 수

- 디폴트 사망자 수는 인구수 * 0.3명
- 그 외 건물 건설에 의하여
  추가 사망자와 추가 생존자가 결정됨

건물 파괴

- 10 / (종교 건물 개수 + 1) 확률로 건물 1개가 파괴됨
- 건물이 파괴될 시, 뉴스에 반영됨

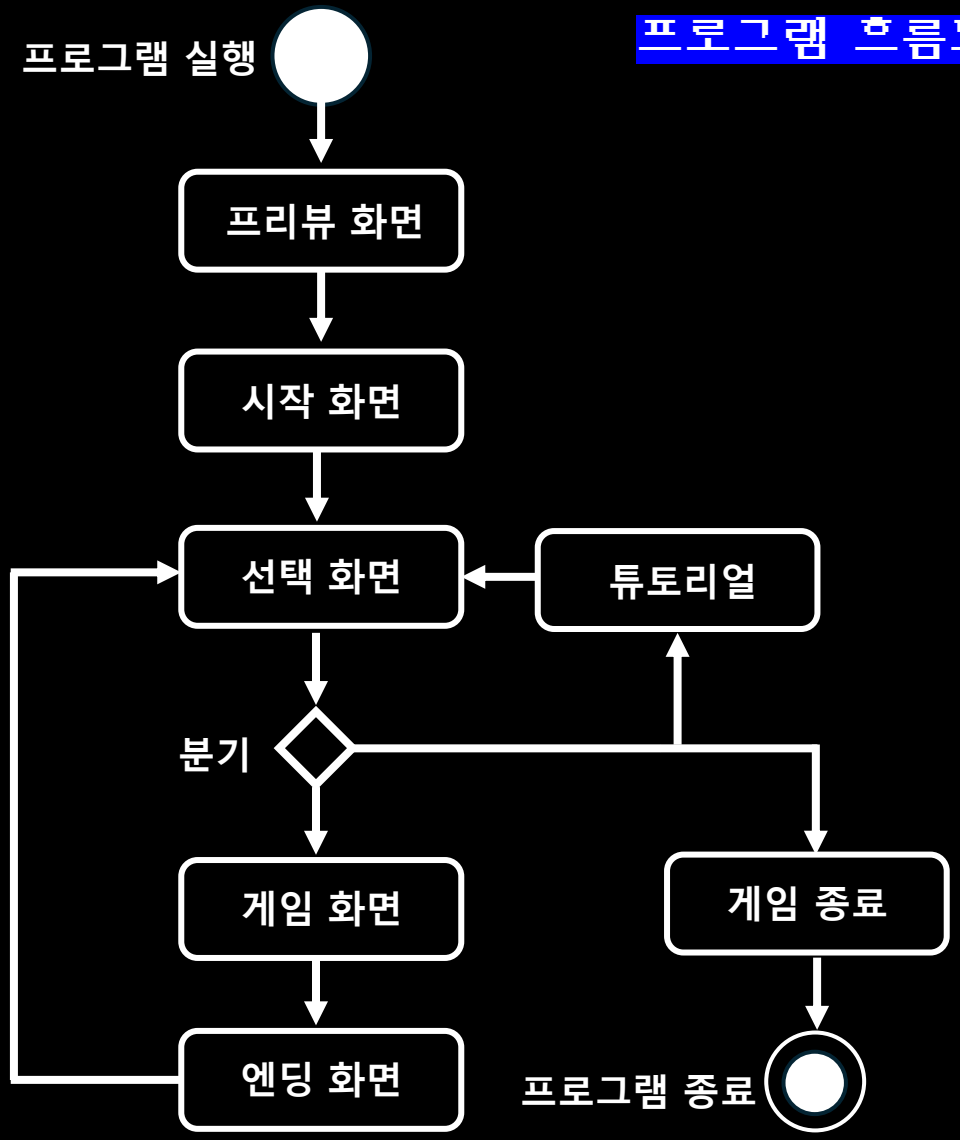
엔딩 조건

엔딩명	엔딩 조건	우선순위
탄핵 엔딩	민심이 10이하로 떨어졌을 시, 엔딩 조건 만족	1
쿠데타 엔딩	민심이 20이하이고 군사력이 50이상일 시, 쿠데타 엔딩 조건 만족	2
종교 개입 엔딩	민심과 군사력이 모두 20이하일 시, 종교 개입 엔딩 조건 만족	3
전멸 엔딩	모든 파워가 0일 시, 전멸 엔딩 만족	4

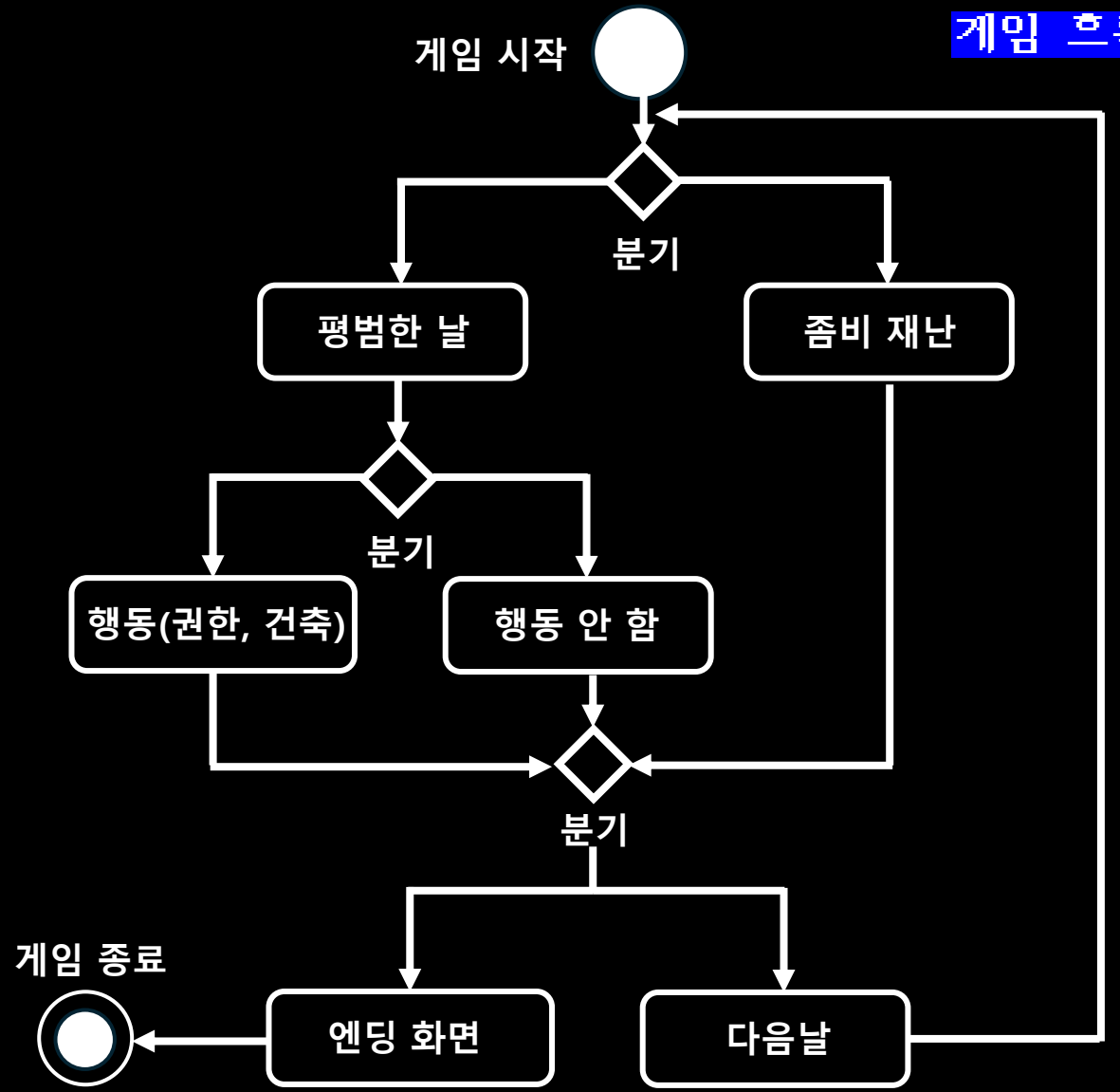
- 모든 엔딩은 조건 만족 횟수가 1번 일 시, 뉴스탭에 경고 뉴스가 표시됨
- 모든 엔딩은 조건 만족 횟수가 2번 일 시, 조건에 맞는 엔딩이 진행됨
- 엔딩 조건 만족 횟수가 1번 일 시, 엔딩 조건을 벗어나면 조건 카운팅이 0으로 초기화 됨
- 여러 엔딩의 조건이 만족되어도 우선 순위에 따라 엔딩이 결정됨
- 좀비 데이에는 전멸 엔딩만 카운팅하며, 전멸 엔딩은 조건 만족 횟수가 1번이어도 엔딩이 진행됨

게임 흐름도

프로그램 흐름도



게임 흐름도



Map 관련 클래스

BuildingInfo

건물의 개수,
건물의 중심 좌표 등
건물의 정보를
담고 있는 클래스

MapInfo

해당 좌표의 건물 건설 여부,
색깔 등
좌표의 정보에 관한 변수를
담고 있는 클래스

BuildingImage

건물 이미지를
담고 있는 클래스

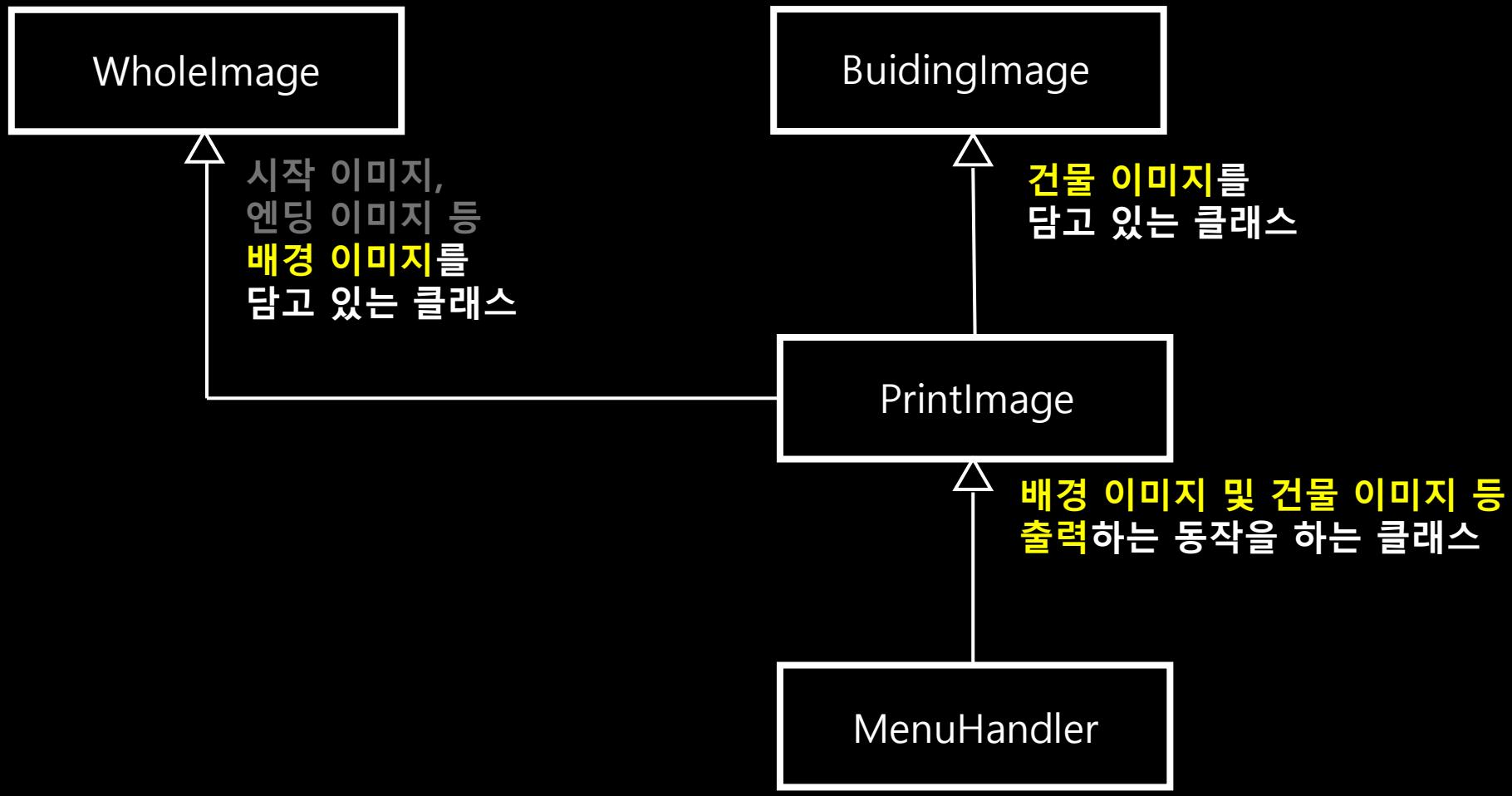
Map

**MapInfo 클래스의
객체 포인터 형의 벡터**를
담고 있는 클래스

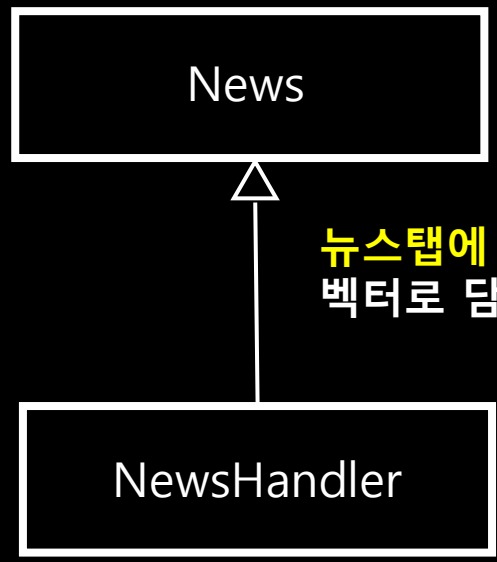
ConstructBuilding

랜덤맵 생성, 건물 건설 및 파괴 등
지도와 관련된 동작을 하는 클래스

이미지 관련 클래스


WholeImage
BuidingImage
시작 이미지,
엔딩 이미지 등
배경 이미지를
담고 있는 클래스
건물 이미지를
담고 있는 클래스
PrintImage
배경 이미지 및 건물 이미지 등
출력하는 동작을 하는 클래스
MenuHandler
건물 이미지를 메뉴에
출력하는 동작을 하는 클래스

뉴스탭, Info 관련 클래스



뉴스탭에 들어갈 뉴스를
벡터로 담고 있는 클래스

뉴스 내용을 큐로 관리하고 (이하 뉴스 큐)
뉴스 큐의 내용을 출력, Push, Pop하는
동작을 담고 있는 클래스

파워, 돈, 이름, 날짜 등
게임 운영에 관련된 변수를
담고 있는 클래스

사용자 입력에 따라
파워, 돈, 행동 횟수 등을
계산하는 동작을 담고 있는 클래스

입력 & 상수 & 음악

SetPosition

커서 위치 이동,
콘솔의 특정 부분 지우기,
색깔 지정과 같은
콘솔 동작과 관련된 클래스

EKEYBOARD

입력 받을 키보드의 값,
뉴스의 종류,
건물 번호,
엔딩 번호 등
열거형 변수를 담고 있는 헤더 파일

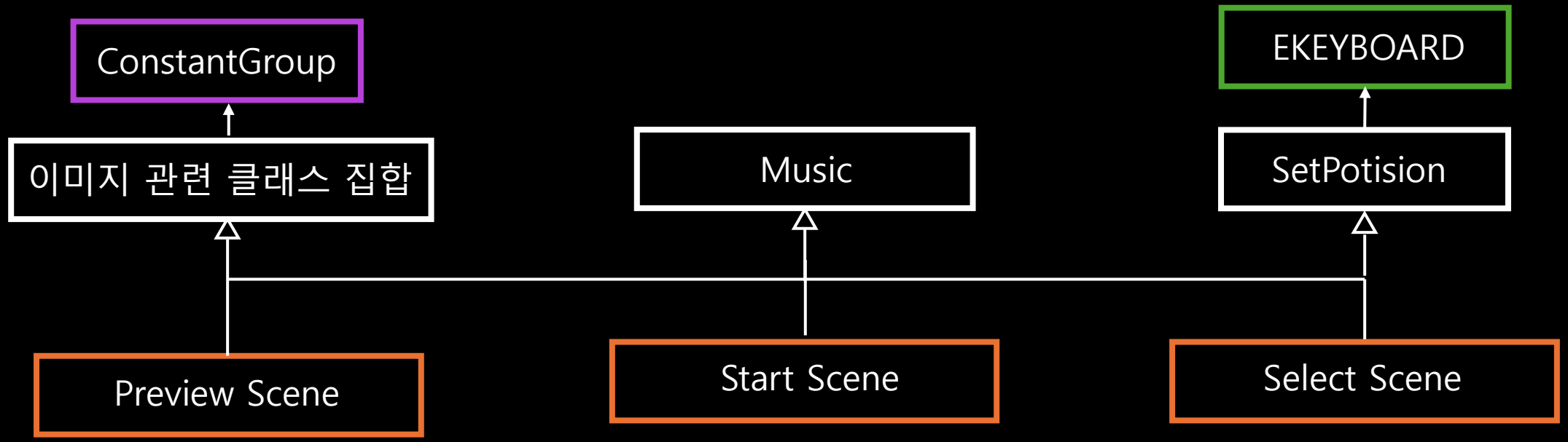
Music

배경음악, 효과음 등
음악 파일을 재생시키는 클래스

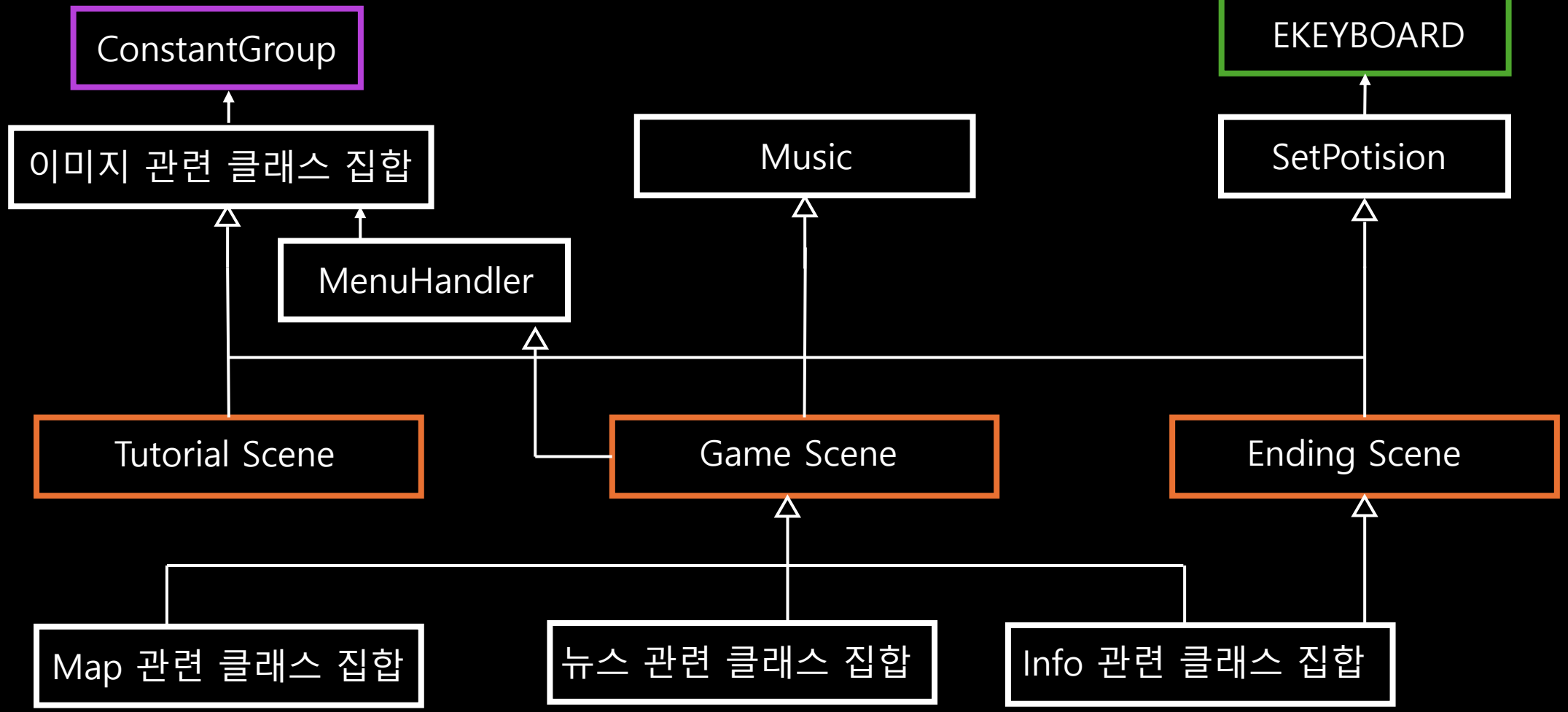
ConstantGroup

이미지의 크기,
이미지의 좌표,
텍스트의 좌표 등
매크로 상수를 담고 있는 헤더 파일

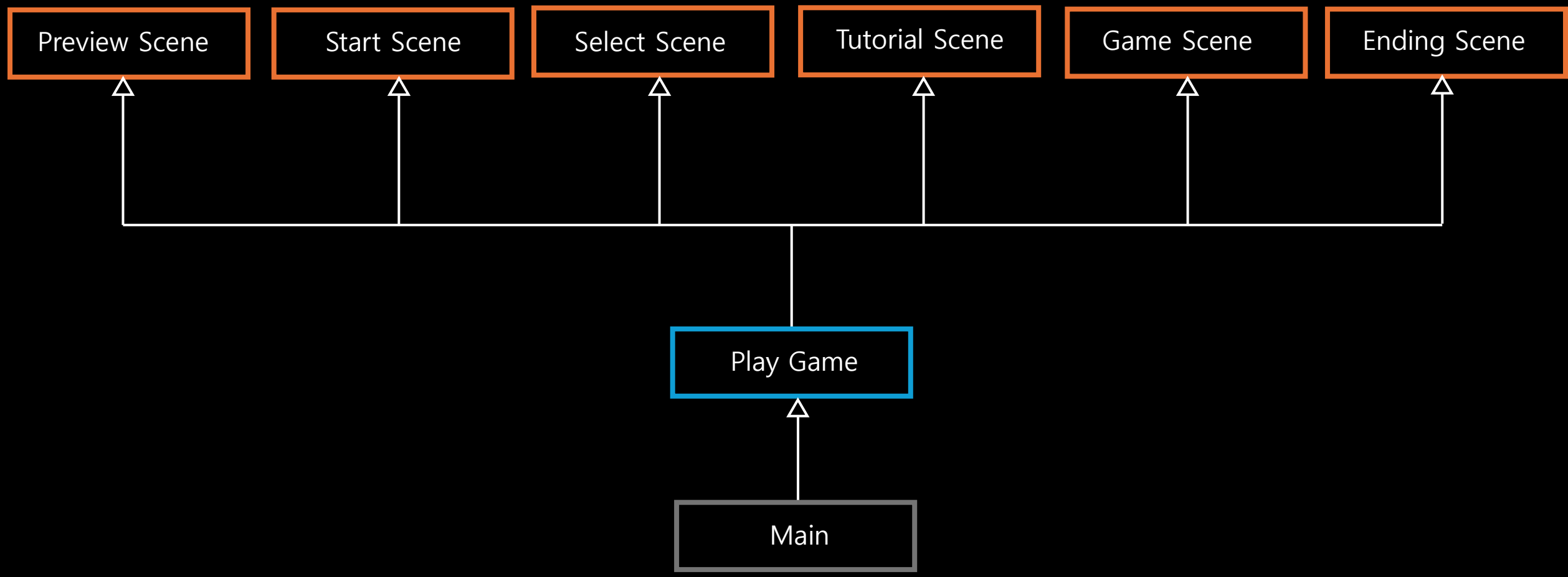
Scene 중심 코드 구조



Scene 중심 코드 구조



전체 코드 구조


Preview Scene
Start Scene
Select Scene
Tutorial Scene
Game Scene
Ending Scene
Play Game
Main

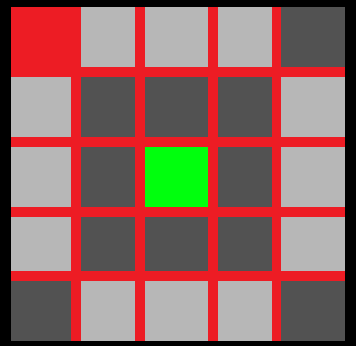
랜덤 맵 생성

맵의 구성

```
class MapInfo
{
private:
    int StartX; // Build's Start X position
    int StartY; // Build's Start Y position
    int Info; // 0: Build Avaliable, -1: Build Unvaliable, 49: hospital ~ 56: ArmyLarge
    int Size; // Building Size
    int Color; // Dot Color
```

MapInfo 객체 포인터형의 벡터인 TotalMap으로 Map을 표현

```
class Map : MapInfo
{
private:
    SetPosition* to = new SetPosition;

protected:
    std::vector<MapInfo*> TotalMap[MAP_Y];
```



건물의 맨 상단 좌측의 좌표가
StartX와 StartY가 되며 한 건물 내에는
동일한 StartX와 StartY 값이 들어감

랜덤 맵 생성

✓ 맵의 초기화

```
void Map::InitMap()
{
    for (int i = 0; i < MAP_Y; i++)
    {
        for (int j = 0; j < MAP_X; j++)
        {
            if (j == 0 || j == MAP_X-1 || j == 1 || j == MAP_X - 2 || i == 0 || i == 1 || i == MAP_Y - 2 || i == MAP_Y - 1)
            {
                TotalMap[i][j]->SetColor(240);
                TotalMap[i][j]->SetInfo(-1);
            }
            else
            {
                TotalMap[i][j]->SetColor(0);
                TotalMap[i][j]->SetInfo(0);
            }
            TotalMap[i][j]->SetSize(-1);
            TotalMap[i][j]->SetStartX(-1);
            TotalMap[i][j]->SetStartY(-1);
        }
    }
}
```

Map의 테두리만 Info를 -1(건물 건설 불가), 색깔을 하얀색으로 지정

랜덤 맵 생성

✓ 랜덤 좌표 결정

```
int index = 0;
while (index < ExitNum)
{
    // 156 x 50, but 5 is for extra
    int DecideXPart = rand() % 3; // 0: x=0, 1: x = 155(EndLine), 2: x = 5 ~ 150(Along Line)
    int DecideYPart = rand() % 2;
```

Value	DecideXPart
0	X좌표 = 0
1	X좌표 = 155
2	5 <= X좌표 <= 155

→

DecideYPart
5 <= Y좌표 <= 45
5 <= Y좌표 <= 45
1이라면 Y좌표 = 49 0이라면 Y좌표 = 0

- 구멍(출입구)를 만드는 과정을 사용자가 입력한 출입구의 개수(ExitNum)만큼 반복
- Map 테두리에 구멍(출입구)를 만들 좌표를 랜덤하게 지정함

랜덤 맵 생성

✓ 랜덤 좌표 결정

DecideXPart = 0 or DecideXPart = 1일때

```
// Block Check
for (int i = 0; i < 4; i++)
{
    if (TotalMap[DecideYPart - i][DecideXPart]->GetInfo() == 0 || TotalMap[DecideYPart - (3 - i)][DecideXPart]->GetInfo() == 0)
    {
        BlockValidCheck = 1;
        break;
    }
}
```

DecideXPart = 2일때

```
// Block Check
for (int i = 0; i < 4; i++)
{
    if (TotalMap[DecideYPart][DecideXPart + i]->GetInfo() == 0 || TotalMap[DecideYPart][DecideXPart - (3 - i)]->GetInfo() == 0)
    {
        BlockValidCheck = 1;
        break;
    }
}
```

- 뽑힌 좌표가 이미 뚫린 곳이 아니고 뽑힌 좌표 기준으로 위, 아래 모두 4칸씩 뚫린 곳이 아닌지 확인
- 위, 아래 4칸씩 확인하는 이유는 이미 뚫린 곳이 겹쳐져서 출입구가 생길 여지를 방지하기 위함

랜덤 맵 생성

✓ 좌표 저장

```
std::set<int> PositionX;
std::set<int> PositionY;
```

```
// Duplication Check
if (PositionX.find(DecideXPart) == PositionX.end() && PositionY.find(DecideYPart) == PositionY.end())
{
    PositionX.insert(DecideXPart);
    PositionY.insert(DecideYPart);

    PerCenterExit.push_back(std::pair<int, int>{DecideXPart, DecideYPart});
```

- 해당 좌표가 출입구가 될 수 있다면 X좌표는 PositionX에, Y좌표는 PositionY에 넣어서 중복을 방지함
- PerCenterExit는 추후 좀비 발생 시, 사망자 수 계산에 사용될 출입구의 좌표를 저장하는 vector임

랜덤 맵 생성

출입구 생성

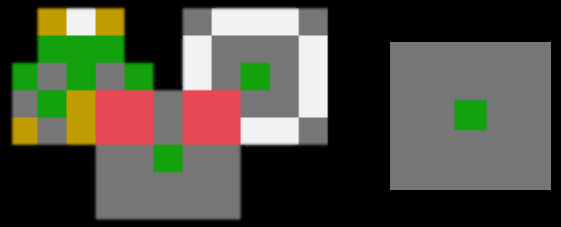
```
// Map Recoloring
if (DecideXPart == 0 || DecideXPart == 155)
{
    for (int i = 0; i < 3; i++)
    {
        if (DecideXPart == 0)
        {
            TotalMap[DecideYPart + i][DecideXPart + 1]->SetColor(0);
            TotalMap[DecideYPart + i][DecideXPart + 1]->SetInfo(0);
        }
        else
        {
            TotalMap[DecideYPart + i][DecideXPart - 1]->SetColor(0);
            TotalMap[DecideYPart + i][DecideXPart - 1]->SetInfo(0);
        }

        TotalMap[DecideYPart + i][DecideXPart]->SetColor(0);
        TotalMap[DecideYPart + i][DecideXPart]->SetInfo(0);
    }
}
```

```
else
{
    for (int i = 0; i < 3; i++)
    {
        if (DecideYPart == 0)
        {
            TotalMap[DecideYPart + 1][DecideXPart + i]->SetColor(0);
            TotalMap[DecideYPart + 1][DecideXPart + i]->SetInfo(0);
        }
        else
        {
            TotalMap[DecideYPart - 1][DecideXPart + i]->SetColor(0);
            TotalMap[DecideYPart - 1][DecideXPart + i]->SetInfo(0);
        }

        TotalMap[DecideYPart][DecideXPart + i]->SetColor(0);
        TotalMap[DecideYPart][DecideXPart + i]->SetInfo(0);
    }
}
```

- 뽑힌 좌표를 기준으로 위, 아래 3칸씩 뚫어줌 (Info = 0, Color = 검정으로 설정)
- 콘솔 창에서는 한 칸이 ½칸씩 차지하므로 가로로 두 칸씩 뚫어주었음

건물 건설 커서 출력

✓ 커서 그림자 출력




```
// 이전 Map을 다시 칠하고
if (direction == EKEYBOARD::KEY_UP)
{
    for (int i = 0; i < size * 2; i++)
    {
        to->GoToXYPosition(x + i, y + size);
        to->SetColor(TotalMap[y + size][x + i]->GetColor());
        printf(" ");
        to->SetColor(0);
    }
}
```

```
// 새 커서배경을 칠해야됨
for (int i = 0; i < size; i++)
{
    for (int j = 0; j < size * 2; j++)
    {
        to->GoToXYPosition(x + j, y + i);
        if (TotalMap[y + i][x + j]->GetInfo() != 0)
        {
            to->SetColor(193);
            printf(" ");
        }
        else if (i == (size / 2) && (j == size - 1 || j == size))
        {
            to->SetColor(32);
            printf(" ");
        }
        else
        {
            to->SetColor(129);
            printf(" ");
        }
    }
}
```

- 입력 받은 방향에 따라서 커서가 지나간 자리를 Map의 정보를 통해 다시 칠해줌
- 커서를 칠할 때는 Map의 Info가 -1인 경우 빨간색으로 칠하고 중앙은 초록, 그 외에는 회색으로 칠함

건물 건설

✓ 건설 여부 체크

```
void ConstructBuilding::NewBuild(int building, int x, int y, InfoHandler* info)
{
    int size = BuildingSize[building];
    int ValidCheck = 0;

    for (int i = 0; i < size; i++)
    {
        for (int j = 0; j < size * 2; j++)
        {
            if (TotalMap[y + i][x + j]->GetInfo() != 0)
            {
                ValidCheck = 1;
                break;
            }
        }
    }
```

```
std::map<int, int> BuildingSize;
```

```
// Size Insert
BuildingSize.insert(std::pair<int, int>(49, 3)); // Hospital
BuildingSize.insert(std::pair<int, int>(50, 3)); // Army Small
BuildingSize.insert(std::pair<int, int>(51, 5)); // APT
BuildingSize.insert(std::pair<int, int>(52, 5)); // Army Medium
BuildingSize.insert(std::pair<int, int>(53, 5)); // Curch
BuildingSize.insert(std::pair<int, int>(54, 5)); // Cathedral
BuildingSize.insert(std::pair<int, int>(55, 8)); // Park
BuildingSize.insert(std::pair<int, int>(56, 8)); // Army Large
```

- 입력 받은 건물 번호, 현재 좌표, info 정보를 들고 있는 class를 매개변수로 받음
- BuildingSize의 value를 통해 해당 건물 번호의 사이즈를 받음
- 현재 좌표를 맨 좌측 상단이라 했을 때, 빌딩 사이즈만큼 체크하여 info가 모두 0인지(건설 가능한지) 확인

건물 건설

Info Update

```
if (ValidCheck == 0)
{
    if (info->GetMoney() < BuildingPrice[building])
    {
        info->PrintMoney(64);
        Sleep(500);

        info->PrintMoney(14);

        return;
    }

    info->SetMoney(info->GetMoney() - BuildingPrice[building]);
    info->PrintMoney(14);
```

```
std::map<int, int> BuildingPrice;
```

```
//Price Insert
BuildingPrice.insert(std::pair<int, int>(49, 10000)); // Hospital
BuildingPrice.insert(std::pair<int, int>(50, 50000)); // Army Small
BuildingPrice.insert(std::pair<int, int>(51, 70000)); // APT
BuildingPrice.insert(std::pair<int, int>(52, 120000)); // Army Medium
BuildingPrice.insert(std::pair<int, int>(53, 70000)); // Curch
BuildingPrice.insert(std::pair<int, int>(54, 70000)); // Cathedral
BuildingPrice.insert(std::pair<int, int>(55, 30000)); // Park
BuildingPrice.insert(std::pair<int, int>(56, 200000)); // Army Large
```

- 건설 가능하다면 info클래스의 Money를 통해 구매 가능한지 확인
- 가능하다면 BuildingPrice의 value만큼 Money를 업데이트 후 Money부분 콘솔에 재출력
- PrintMoney의 매개변수는 출력할 텍스트의 색깔임

건물 건설

Power 및 좌표 Update

```
PlusCntBuilding(building, info);
AddCOORDData(x, y, building, BuildingSize[building]);

info->PrintPower(14);
```

```
void ConstructBuilding::PlusCntBuilding(int building, InfoHandler* info)
{
    int power;
    if (building == EKEYBOARD::NUM1_KEY)
    {
        power = info->GetCitizenPower();
        power+=1;
        info->SetCitizenPower(power);

        HospitalNum++;
    }
```

```
void BuildingInfo::AddCOORDData(int x, int y, int building, int size)
{
    if (building == EKEYBOARD::NUM3_KEY) {
        PerCenterCOORDAPT[std::pair<int,int>(x, y)] = std::pair<int,int>(CalCulCenterCOORD(size, x, y));
        CenterCOORDAPT = CalCulCenterCOORDMap(PerCenterCOORDAPT);
    }
```

- PlusCntBuilding을 통해 power와 건물 개수를 업데이트 함
- AddCOORData를 통해 건물 마다의 중심좌표와 전체 건물의 중심좌표를 업데이트 함
- 좌표 계산은 아파트와 종교 건물이 좀비 발생 시, 사망자 수 계산에 필요하므로 수행함

✓ 중심 좌표 공식

- 각 건물의 중심 좌표 = 건물의 StartX - (size - 1), 건물의 StartY - (size / 2)

- 각 건물의 중심 좌표를 (x1, y1), (x2, y2) … (xn, yn)이라 했을 때, 전체 건물의 중심 좌표는 다음과 같음

$$\left(\frac{x_1 + x_2 + \ldots + x_n}{n}, \frac{y_1 + y_2 + \ldots + y_n}{n}\right)$$

✓ 중심 좌표 코드 구현

```cpp
std::pair<int, int> BuildingInfo::CalCulCenterCOORD(int size, int x, int y)
{
    return std::pair<int, int>(x + size, y + size /  2);
}

std::pair<int, int> BuildingInfo::CalCulCenterCOORDMap(std::map<std::pair<int, int>, std::pair<int, int>> map)
{
    std::map<std::pair<int, int>, std::pair<int, int>>::iterator iter;

    double MapLen = map.size();

    double TotalX = 0;
    double TotalY = 0;
    for (iter = map.begin(); iter != map.end(); ++iter)
    {
        TotalX += iter->second.first;
        TotalY += iter->second.second;
    }

    TotalX /= MapLen;
    TotalY /= MapLen;

    return std::pair<int, int>((int)TotalX, (int)TotalY);
}
```

■ 따라서 중심 좌표(CalCulCenterCOORD)와 전체 중심좌표(CalCulCenterCOORDMap)의 코드는 위와 같음

건물 건설

✓ 좌표 정보 Update

```
for (int i = 0; i < size; i++)
{
    for (int j = 0; j < size * 2; j++)
    {
        TotalMap[y + i][x + j]->SetStartX(x);
        TotalMap[y + i][x + j]->SetStartY(y);
        TotalMap[y + i][x + j]->SetSize(size);


        if (building == EKEYBOARD::NUM1_KEY)
        {
            TotalMap[y + i][x + j]->SetInfo(EBUILDING::HOSPITAL);
            TotalMap[y + i][x + j]->SetColor(Hospital[i][j]);
        }
```

```
class BuildingImage
{
protected:
    int Hospital[SMALL_Y][SMALL_X] =
    { {240, 240, 32, 32, 240, 240},
    { 32, 32, 32, 32, 32, 32},
    { 240, 240, 32, 32, 240, 240}
    };
```

StartX	선택한 X좌표
StartY	선택한 Y좌표
Size	건물의 사이즈
Color	건물의 색깔
Info	-1 (건설 불가 좌표)

- 건물의 사이즈만큼 좌표 설정을 변경해줌

건물 파괴

✓ 커서 범위 안 건물 체크

```
void ConstructBuilding::DestroyBuilding(int building, int size, int x, int y, InfoHandler* info, int Zombie)
{

    int DeleteX, DeleteY;

    std::set<std::pair<int,int>> DeleteCOOR;

    // Checking for building on choosen place
    for (int i = 0; i < size; i++)
    {
        for (int j = 0; j < size * 2 - Zombie; j++)
        {
            if (TotalMap[y + i][x + j]->GetInfo() != 0)
            {
                DeleteCOOR.insert(std::pair<int, int>{TotalMap[y + i][x + j]->GetStartX(), TotalMap[y + i][x + j]->GetStartY()});
            }
        }
    }
```

- 커서 범위 안에 있는 건물을 모두 삭제하기 위해
  커서 안에 있는 좌표 중 info가 0이 아닌 좌표를 모두 DeleteCOOR에 insert 함
- Zombie 매개변수는 좀비 발생 시, 랜덤 건물 파괴를 위한 매개변수임

건물 파괴

Power 및 좌표 Update

```
int DLen = DeleteCOOR.size();

if (DLen != 0)
{
    std::set<std::pair<int, int>>::iterator iter;
    for (iter = DeleteCOOR.begin(); iter != DeleteCOOR.end(); ++iter)
    {
        MinusCntBuilding(TotalMap[iter->second][iter->first]->GetInfo(), info);
        RemoveCOORDData(TotalMap[iter->second][iter->first]->GetStartX(), y, Tota
```

```
void ConstructBuilding::MinusCntBuilding(int building, InfoHandler* info)
{
    int power;
    if (building == EKEYBOARD::NUM1_KEY)
    {
        HospitalNum--;
        power = info->GetCitizenPower();
        power -= 1;
        info->SetCitizenPower(power);
    }
```

```
void BuildingInfo::RemoveCOORDData(int x, int y, int building, int size)
{
    if (building == EKEYBOARD::NUM3_KEY) {
        PerCenterCOORDAPT.erase(std::pair<int, int>(x, y));
        CenterCOORDAPT = CalCulCenterCOORDMap(PerCenterCOORDAPT);
    }
```

- DeleteCOOR의 사이즈가 0이 아니라면 삭제 작업을 진행함
- MinusCntBuilding에서는 Power와 건물 개수를 업데이트 함
- RemoveCOORDData에서는 삭제하는 좌표의 StartX와 StartY를 key로 두어 해당 좌표 값을 삭제 후 전체 중심좌표를 재계산 함

건물 파괴

✓ 건설비 반환

```
if (Zombie == 0 && TotalMap[iter->second][iter->first]->GetInfo() != EKEYBOARD::RELIGION_42B_KEY) // 평상시
{
    info->SetMoney(info->GetMoney() + BuildingPrice[TotalMap[iter->second][iter->first]->GetInfo()] / 2);
}
else if (TotalMap[iter->second][iter->first]->GetInfo() == EKEYBOARD::RELIGION_42B_KEY) //42B Destroy
{
    if (info->GetMoney() < 100000)
    {
        info->PrintMoney(64);
        Sleep(500);

        info->PrintMoney(14);

        continue;
    }

    info->SetCitizenPower(info->GetCitizenPower() + 10);
    info->SetReligionPower(info->GetReligionPower() + 10);
    info->SetMoney(info->GetMoney() - 100000);
}
```

- 좀비 재난이 아니라면 건설비의 절반 만큼 더하여 money를 업데이트 함
- 삭제하고자 하는 건물이 사이비 건물이라면 그에 맞게 power와 money를 업데이트 함

건물 파괴

✓ 좌표 Update

```
int CurSize = Map::TotalMap[iter->second][iter->first]->GetSize();
for (int i = 0; i < CurSize; i++)
{
    for (int j = 0; j < CurSize * 2; j++)
    {
        TotalMap[iter->second + i][iter->first + j]->SetStartX(-1);
        TotalMap[iter->second + i][iter->first + j]->SetStartY(-1);
        TotalMap[iter->second + i][iter->first + j]->SetInfo(0);
        TotalMap[iter->second + i][iter->first + j]->SetSize(-1);
        TotalMap[iter->second + i][iter->first + j]->SetColor(0);
    }
}
```

StartX	-1
StartY	-1
Size	-1
Color	0 (검정)
Info	0 (건설 가능 좌표)

- 삭제하고자 하는 건물 사이즈 만큼 좌표의 값을 업데이트 함

뉴스탭 관리

✓ 뉴스 추가 및 출력

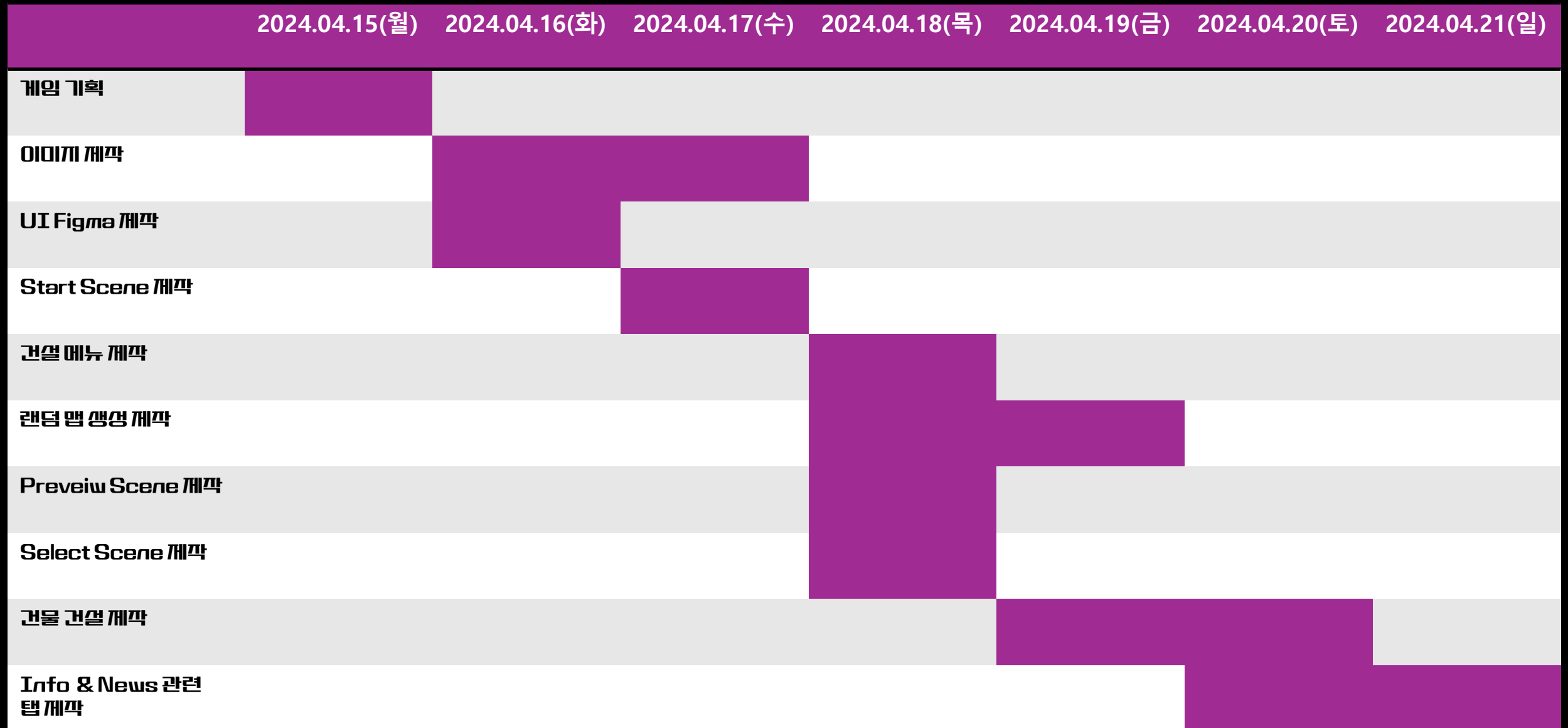
```
// Citizen
CitizenLittleForceNews.push_back(std::pair < const char*, int>("[국민] \"차라리 좀비가 나아...\"", 12));
CitizenLittleForceNews.push_back(std::pair < const char*, int>("[국민] \"라면도 비싸요.\" 흔들리...", 12));
CitizenLittleForceNews.push_back(std::pair < const char*, int>("[국민] 대통령 혼자사는 국가? ", 12));
```

```
void NewsHandler::PushNewsQueue(int category)
{
    int r;
    if (category == ENEWS_CATEGORY::CitizenLittleForceNews)
    {
        r = rand() % 3;
        if (NewsQ.size() == MAX_NEWS)
        {
            NewsQ.pop();
        }
        NewsQ.push(CitizenLittleForceNews[r]);
    }
```

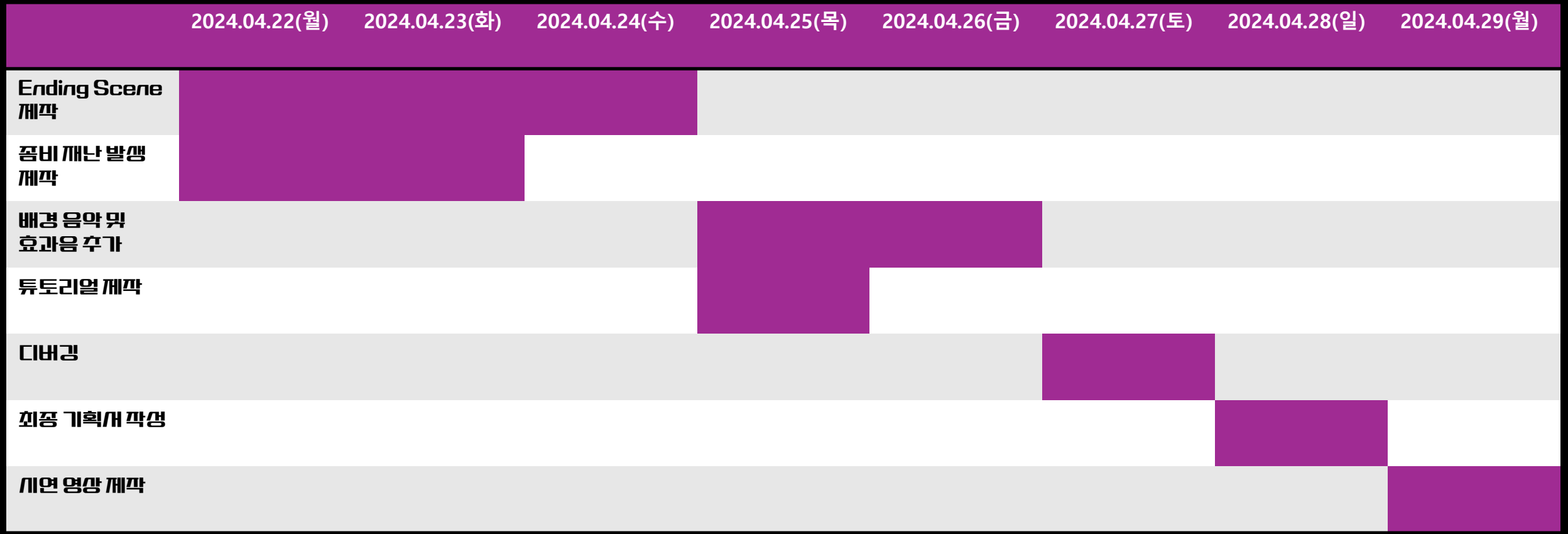
```
void NewsHandler::ShowNewNews()
{
    to->PartClean(NEWS_POSITION_X+2, NEWS_POSITION_Y+4, 40, 15);
    int Size = NewsQ.size();

    int index = 0;
    while (NewsQ.size())
    {
        to->GoToXYPosition(NEWS_POSITION_X + 4, NEWS_POSITION_Y + 4 + index);
        to->SetColor(NewsQ.front().second);
        printf("%s", NewsQ.front().first);

        NewsQ.pop();

        index+=2;
    }

}
```

- 각 분야에 맞는 뉴스를 벡터에 넣어 저장함(pair의 두 번째 인자는 텍스트 색깔을 담당함)
- 분야별 뉴스의 개수는 보통 3개이며 3개중 랜덤으로 push됨
- 모든 뉴스는 NewsQ에 넣고 새로운 날이 시작되면 모두 pop하여 뉴스탭에 출력 함

개발 일정


2024.04.15(월)
2024.04.16(화)
2024.04.17(수)
2024.04.18(목)
2024.04.19(금)
2024.04.20(토)
2024.04.21(일)
게임 기획
이미지 제작
UI Figma 제작
Start Scene 제작
건설 메뉴 제작
랜덤 맵 생성 제작
Preveiw Scene 제작
Select Scene 제작
건물 건설 제작
Info & News 관련
탭 제작

개발 일정

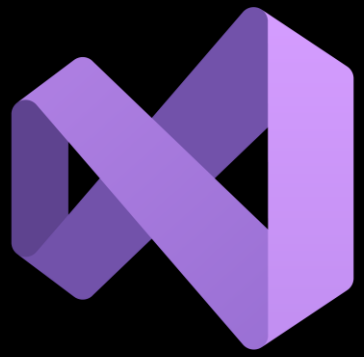

2024.04.22(월)
2024.04.23(화)
2024.04.24(수)
2024.04.25(목)
2024.04.26(금)
2024.04.27(토)
2024.04.28(일)
2024.04.29(월)
Ending Scene 제작
좀비 재난 발생 제작
배경 음악 및 효과음 추가
튜토리얼 제작
디버깅
최종 기획서 작성
시연 영상 제작

사용 언어 및 프로그램

✓ 사용 언어

C++

✓ 사용 IDE

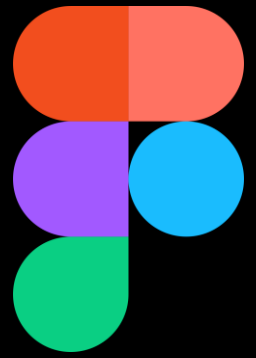


Visual Studio

✓ 형상 관리

GitHub

✓ 사용 UI툴



Figma

감사합니다

그대들은.. 1999에서.. 살아남기를...