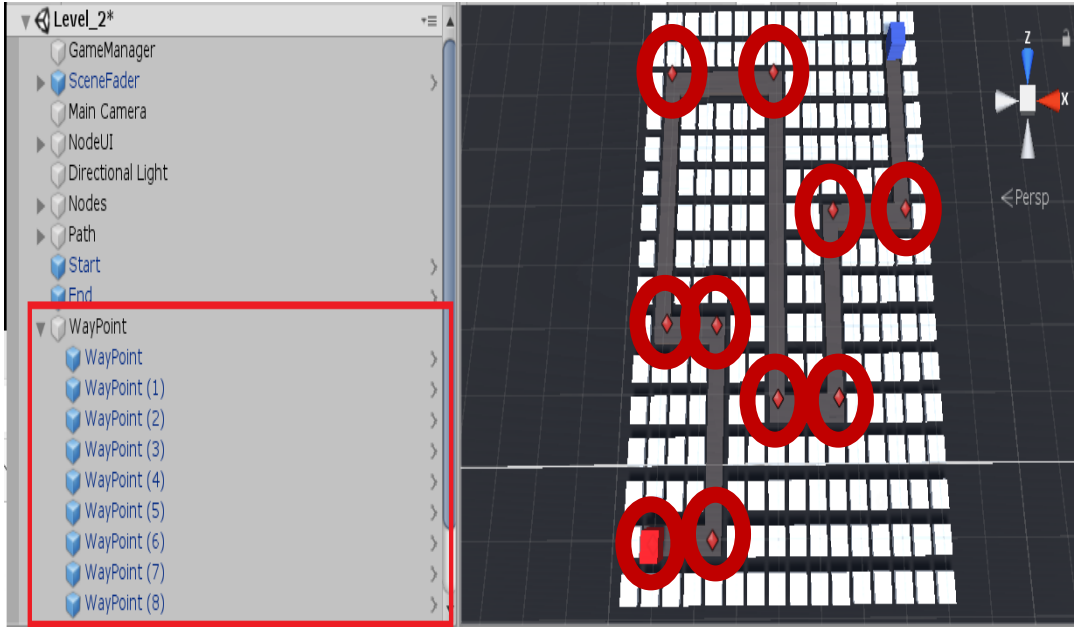


---

## 소스 설명

## 1. 적 이동 관련 스크립트 (WayPointMGR)



```
public class WayPointMGR : MonoBehaviour
{
    public static Transform[] WayPointList = null;
    Unity 메시지 참조 0개
    private void Awake()
    {
        WayPointList = new Transform[transform.childCount];
        for (int i = 0; i < transform.childCount; i++)
        {
            WayPointList[i] = transform.GetChild(i);
        }
    }
}
```

- 적 오브젝트가 이동 할 위치 정보가 있는 WayPoint 오브젝트를 생성 함
- 적 오브젝트들을 WayPoint에 순차적으로 이동 시킬 예정 이기 때문에 각각의 WayPoint 담은 배열을 생성함

## 1. 적 이동 관련 스크립트 (EnemyMovementMGR)

```
Ⓢ Unity 메시지 | 참조 0개
private void Start()
{
    m_Target = WayPointMGR.WayPointList[0];
    m_EnemyMGR = gameObject.GetComponent<EnemyMGR>();
}

Ⓢ Unity 메시지 | 참조 0개
private void Update()
{
    Vector3 Dir = m_Target.position - transform.position;
    transform.Translate(Dir.normalized * m_EnemyMGR.m_fSpeed * Time.deltaTime, Space.World); //normalized : 일정한 속도로 가기 위해서
    if (Vector3.Distance(transform.position, m_Target.position) <= 0.5f)
    {
        NextWayPoint();
    }
}
```

- 이동해야 할 위치를 WayPointList에서 받아 Translate 함수 이용하여 이동시킴

## 1. 적 이동 관련 스크립트 (EnemyMovementMGR)

```
private void NextWayPoint()
{
    m_nWavepointIndex++;

    if (m_nWavepointIndex >= WayPointMGR.WayPointList.Length)
    {
        EndPath();
        return;
    }

    m_Target = WayPointMGR.WayPointList[m_nWavepointIndex];
}

참조 1개
void EndPath()
{
    PlayerStats.nLives--;
    WaveMGR.nEnemyAlive--;
    GameObject.Destroy(gameObject);
}
```

- WayPoint와 일정 거리가 되면 도착한걸로 판단하여 WayPointList에서 새로운 위치를 받아 이동 시킴
- 만약 WayPointList의 마지막 위치에 도착 한다면 END에 도착 했기 때문에 오브젝트를 Destroy 시킨 후 플레이어의 Lives를 깎음

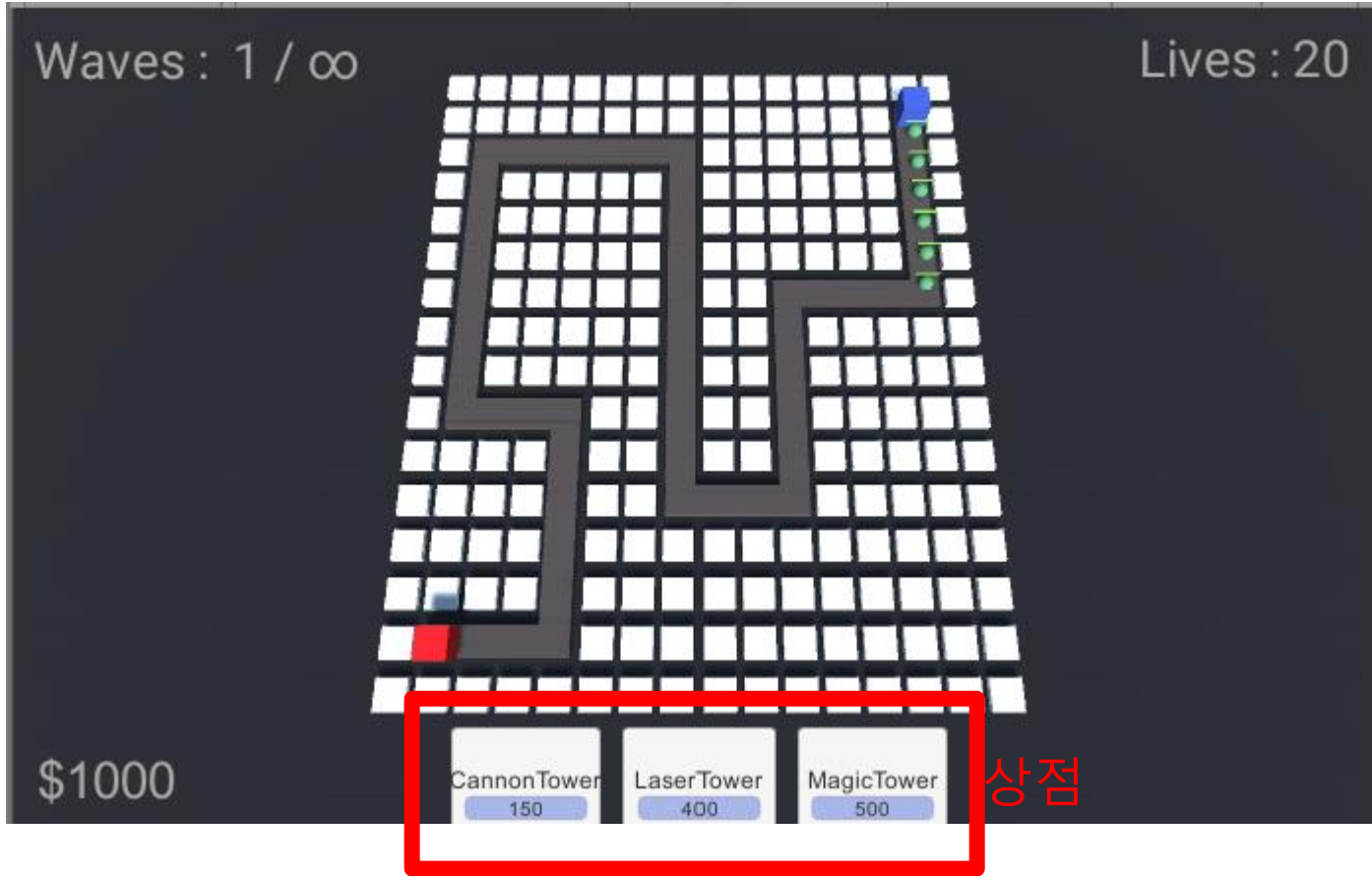
## 2. 타워 생성 관련 스크립트

```
//타워 타입
참조 60개
public enum E_TOWERTYPE
{
    E_ID ,
    E_CANNON ,
    E_LASER ,
    E_MAGIC ,
    E_MAX
}

[System.Serializable] //공통 속성을 가진 변수들을 그룹지어 클래스를 만든다. [Serializable] 속성을 부여한다.
//타워 기본 정보
참조 10개
public class TowerBuildPrint
{
    public GameObject m_prefab = null;
    public GameObject m_Upgradeprefab = null;
    public int m_nCost = 0;
    public int m_nUpgradeCost = 0;
    public int m_nSellPrice = 0;
    public int m_nSellUpgradePrice = 0;
}
```

- 타워 설치 또는 판매 시 필요한 변수들이 있는 클래스를 생성함

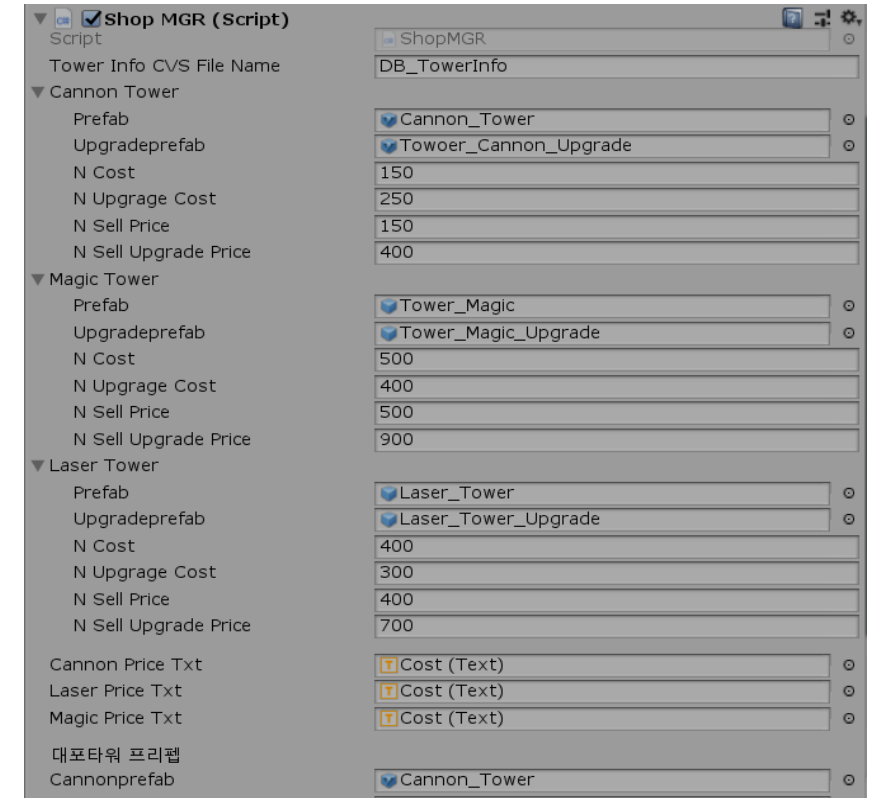
## 2. 타워 생성 관련 스크립트 (상점 부분)



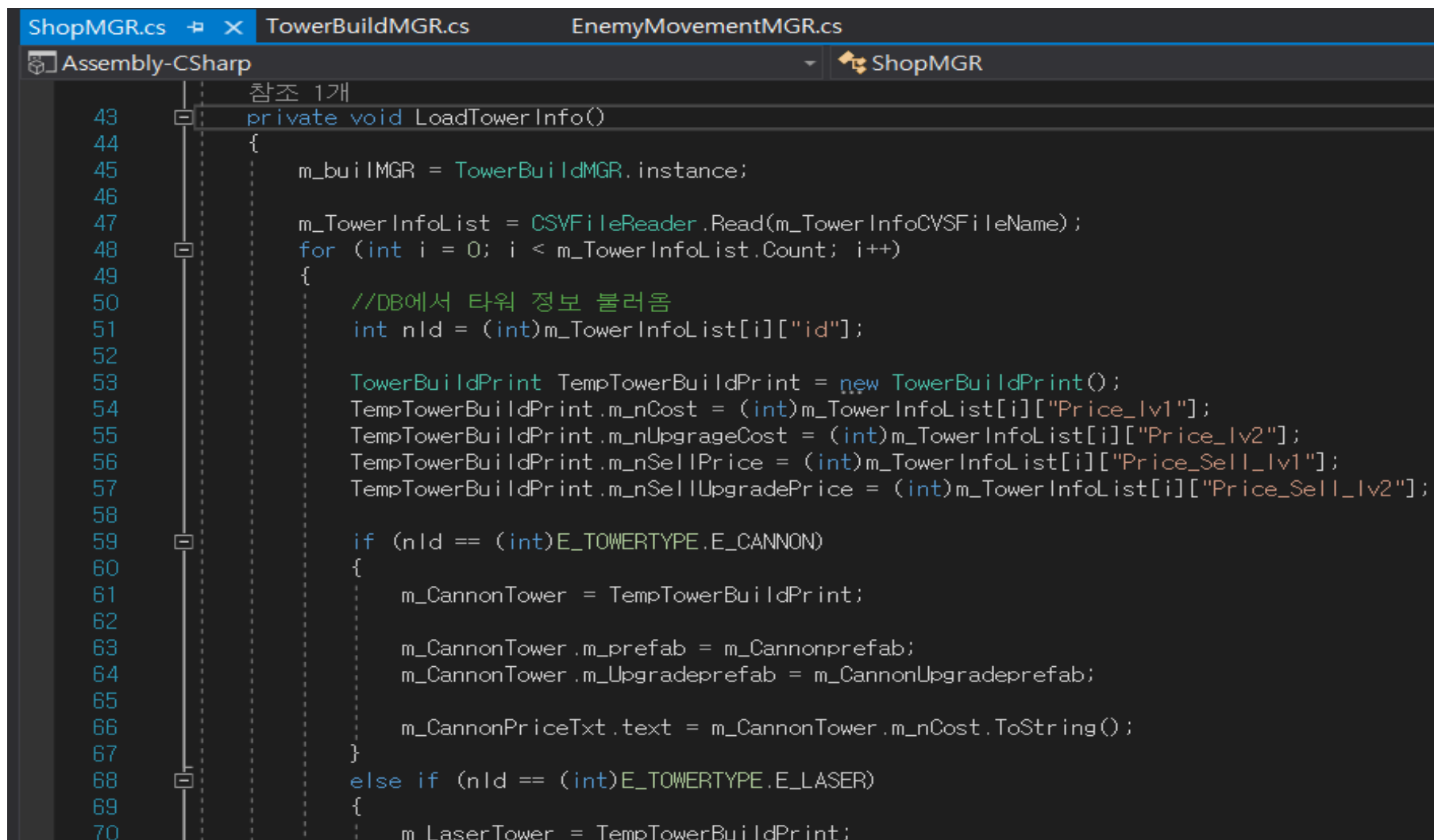
## 2. 타워 생성 관련 스크립트 (ShopMGR)

	A	B	C	D	E	F	G	H
1	id	Name	Price_lv1	Price_lv2	Price_Sell	Price_Sell_	1Row_IncreaseUpgradeCost	2Row_IncreaseUpgradeCost
2	1	CannonTower	150	250	150	400	100	150
3	2	LaserTower	400	300	400	700	150	250
4	3	MagicTower	500	400	500	900	200	300

- 게임 시작 시 타워 정보가 있는 CSV 파일을 읽어와 상점에 저장시킴



## 2. 타워 생성 관련 스크립트 (ShopMGR)

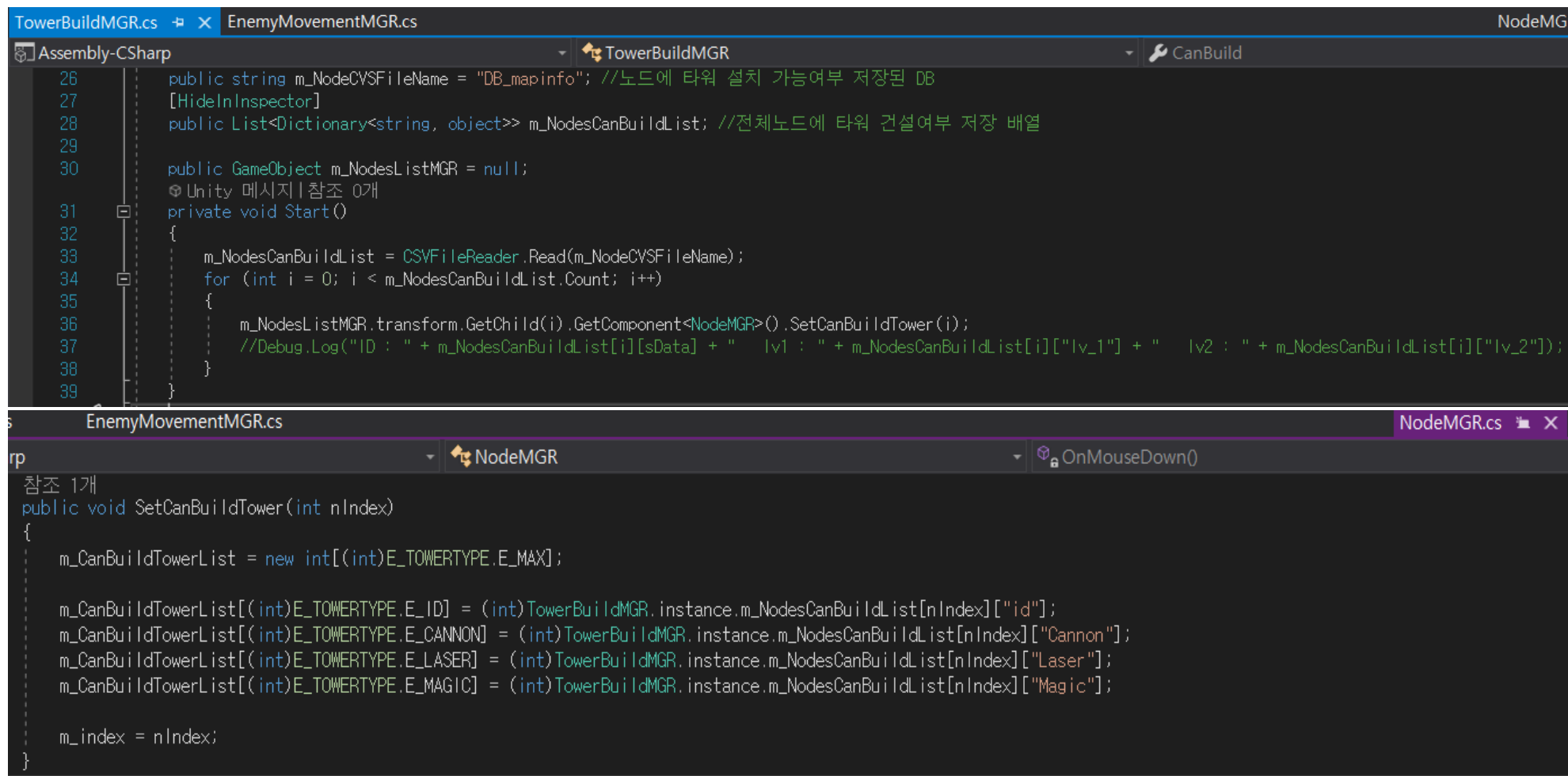


```
ShopMGR.cs TowerBuildMGR.cs EnemyMovementMGR.cs
Assembly-CSharp ShopMGR

참조 1개
43 private void LoadTowerInfo()
44 {
45     m_buiMGR = TowerBuildMGR.instance;
46
47     m_TowerInfoList = CSVFileReader.Read(m_TowerInfoCSVFileName);
48     for (int i = 0; i < m_TowerInfoList.Count; i++)
49     {
50         //DB에서 타워 정보 불러옴
51         int nId = (int)m_TowerInfoList[i]["id"];
52
53         TowerBuildPrint TempTowerBuildPrint = new TowerBuildPrint();
54         TempTowerBuildPrint.m_nCost = (int)m_TowerInfoList[i]["Price_lv1"];
55         TempTowerBuildPrint.m_nUpgradeCost = (int)m_TowerInfoList[i]["Price_lv2"];
56         TempTowerBuildPrint.m_nSellPrice = (int)m_TowerInfoList[i]["Price_Sell_lv1"];
57         TempTowerBuildPrint.m_nSellUpgradePrice = (int)m_TowerInfoList[i]["Price_Sell_lv2"];
58
59         if (nId == (int)E_TOWERTYPE.E_CANNON)
60         {
61             m_CannonTower = TempTowerBuildPrint;
62
63             m_CannonTower.m_prefab = m_Cannonprefab;
64             m_CannonTower.m_Upgradeprefab = m_CannonUpgradeprefab;
65
66             m_CannonPriceTxt.text = m_CannonTower.m_nCost.ToString();
67         }
68         else if (nId == (int)E_TOWERTYPE.E_LASER)
69         {
70             m_LaserTower = TempTowerBuildPrint;
```



## 2. 타워 생성 관련 스크립트 (TowerBuildMGR, NodeMGR)



```
26 public string m_NodeCSVFileName = "DB_mapinfo"; //노드에 타워 설치 가능여부 저장된 DB
27 [HideInInspector]
28 public List<Dictionary<string, object>> m_NodesCanBuildList; //전체노드에 타워 건설여부 저장 배열
29
30 public GameObject m_NodesListMGR = null;
31 // Unity 메시지 참조 0개
32 private void Start()
33 {
34     m_NodesCanBuildList = CSVFileReader.Read(m_NodeCSVFileName);
35     for (int i = 0; i < m_NodesCanBuildList.Count; i++)
36     {
37         m_NodesListMGR.transform.GetChild(i).GetComponent<NodeMGR>().SetCanBuildTower(i);
38         //Debug.Log("ID : " + m_NodesCanBuildList[i][sData] + "   lv1 : " + m_NodesCanBuildList[i][lv_1] + "   lv2 : " + m_NodesCanBuildList[i][lv_2]);
39     }
40 }

EnemyMovementMGR.cs
NodeMGR
참조 1개
public void SetCanBuildTower(int nIndex)
{
    m_CanBuildTowerList = new int[(int)E_TOWERTYPE.E_MAX];

    m_CanBuildTowerList[(int)E_TOWERTYPE.E_ID] = (int)TowerBuildMGR.instance.m_NodesCanBuildList[nIndex]["id"];
    m_CanBuildTowerList[(int)E_TOWERTYPE.E_CANNON] = (int)TowerBuildMGR.instance.m_NodesCanBuildList[nIndex]["Cannon"];
    m_CanBuildTowerList[(int)E_TOWERTYPE.E_LASER] = (int)TowerBuildMGR.instance.m_NodesCanBuildList[nIndex]["Laser"];
    m_CanBuildTowerList[(int)E_TOWERTYPE.E_MAGIC] = (int)TowerBuildMGR.instance.m_NodesCanBuildList[nIndex]["Magic"];

    m_index = nIndex;
}
```

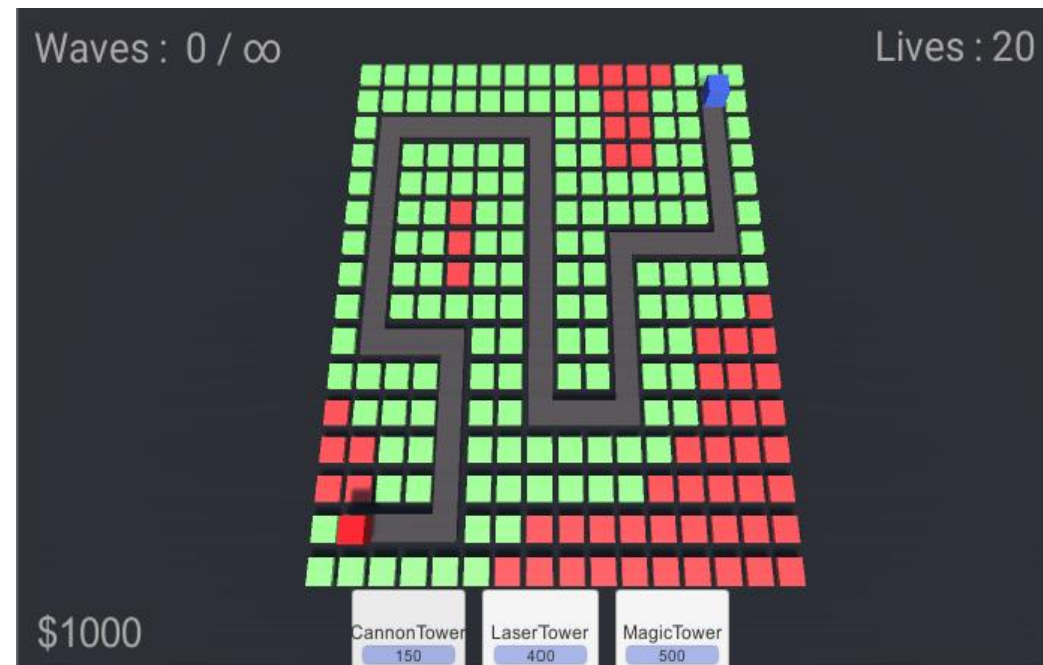
- 게임매니저에서 타워의 종류에 따라 노드에 설치 가능 여부를 저장한 CSV 파일을 읽어와 각각의 노드에 저장함  
(CSV 파일 읽는 소스 해당 링크 참고 : <https://bravenewmethod.com/2014/09/13/lightweight-csv-reader-for-unity/#comment-71111>)

## 2. 타워 생성 관련 스크립트 (TowerBuildMGR)

```
s ShopMGR.cs TowerBuildMGR.cs x EnemyMovementMGR.cs
arp
참조 3개
public void SelectTowerToBuild (TowerBuildPrint Tower, E_TOWERTYPE eTowerType)
{
    m_eTowerType = eTowerType;
    m_TowerBuildPrintMGR = Tower;

    if (HasMoney)
    {
        for (int i = 0; i < m_NodesCanBuildList.Count; i++)
        {
            m_NodesListMGR.transform.GetChild(i).GetComponent<NodeMGR>().SetNodeColor(eTowerType);
        }
    }

    DeSelectNode();
}
```



- 상점에서 타워 버튼 누를 시 타워 설치 금액이 있다면 설치 가능 여부에 따라 색깔 변경

## 2. 타워 생성 관련 스크립트 (NodeMGR)

참조 1개

```
void BuildTower(TowerBuildPrint TowerBuildPrint)
{
    if (PlayerStats.nMoney < TowerBuildPrint.m_nCost)
    {
        return;
    }
    PlayerStats.nMoney -= TowerBuildPrint.m_nCost;

    GameObject effect = Instantiate(m_TowerBuildMGR.m_buildEffect, GetBuildPosition(), Quaternion.identity);
    Destroy(effect, 3f);

    //GameObject Tower = Instantiate(m_TowerBuildPrintMGR.m_prefab, node.GetBuildPosition(), Quaternion.identity);
    GameObject Tower = Instantiate(TowerBuildPrint.m_prefab, GetBuildPosition(), TowerBuildPrint.m_prefab.transform.rotation);
    m_Tower = Tower;
    m_TowerBuildPrintMGR = TowerBuildPrint;
    m_TowerBuildMGR.ResetSelectTower();
}
```

- 타워 설치가 가능한 노드를 클릭 했을 시 해당 함수로 타워 생성함

---

## 화면 별 스크립트 기능 설명 (메인메뉴)

- MainMenu : 메뉴 화면 이벤트 관련 스크립트
- LevleSelector : 스테이지 선택 화면에서 버튼 클릭 관련 스크립트
- SceneFader : 화면 전환 시 부드럽게 하기 위한 스크립트
- UpgradeStatsResetMGR : 업그레이드 내역 초기화 스크립트

---

## 화면 별 스크립트 기능 설명 (업그레이드)

- UpgradeButtonMGR : 업그레이드 화면에서 사용되는 오브젝트 관리 스크립트
- UpgradeConfirmMGR : 업그레이드 항목 선택 시 나오는 이벤트 관련 스크립트
- NotEnoughMoneyUIMGR : 업그레이드 시 돈이 모자를 경우 나오는 캔버스 관련 스크립트
- SceneFader : 화면 전환 시 부드럽게 하기 위한 스크립트
- UpgradeMGR : 업그레이드 항목 선택 관련 스크립트

---

## 화면 별 오브젝트 및 스크립트 기능 설명 (게임 스테이지)

- 게임 매니저

- ① PlayerStats : 게임시작 시 플레이어의 목숨, 돈 세팅 관련 스크립트
- ② WaveMGR : 게임 시 적 오브젝트 생성 스크립트
- ③ GameManager : 게임을 종료 관련 오브젝트 관리 스크립트
- ④ TowerBuildMGR : 타워 생성 시 필요한 오브젝트 관리 스크립트

- 카메라

- ① CameraMGR : PC버전으로 실행 시 마우스 이벤트 관련 스크립트
- ② CameraTouchMGR : 안드로이드 버전 실행 시 터치 관련 스크립트

- 오버레이 캔버스

- ① GameOverMGR : 게임오버 시 나오는 화면 관련 스크립트
- ② PausedMenu : 게임 중 ESC 클릭 시 나오는 캔버스 관련 스크립트
- ③ RoundSurviveMGR : 게임 종료 시 Wave 표시 효과 관련 스크립트
- ④ CompleteLevel : 스테이지 클리어 시 나오는 화면에서 사용 되는 스크립트
- ⑤ ShopMGR : 상점 관련 스크립트

---

## 화면 별 오브젝트 및 스크립트 기능 설명(게임 스테이지)

- 노드
  - ① NodeMGR : 노드에서 발생 하는 이벤트 관련 스크립트
- 노드UI
  - ① NodeUIMGR : 타워가 있는 노드 선택 시 나오는 캔버스 관련 스크립트
- 타워
  - ① TowerMGR : 타워에서 사용 되는 스크립트
  - ② BulletMGR : 타워에서 투사체 오브젝트 생성 후 투사체에서 실행 하는 스크립트
- 적
  - ① EnemyMGR : 적 오브젝트의 상태 관련 스크립트
  - ② EnemyMovementMGR : 적 오브젝트의 움직임 관련 스크립트
- 공통 클래스
  - ① TowerBuildPrint : 타워 설치 시 필요한 변수 클래스
  - ② CSVFileReader : CSV 파일 읽는 클래스