

팀 프로젝트
담당 방어 시스템 구현

김재완

방어 구현 목록

- 지형 만들기
- 방어용 타워 제작
- 방어용 내 타워 위치 저장 및 로드
- 게임플레이시 상대방 타워 위치 설정

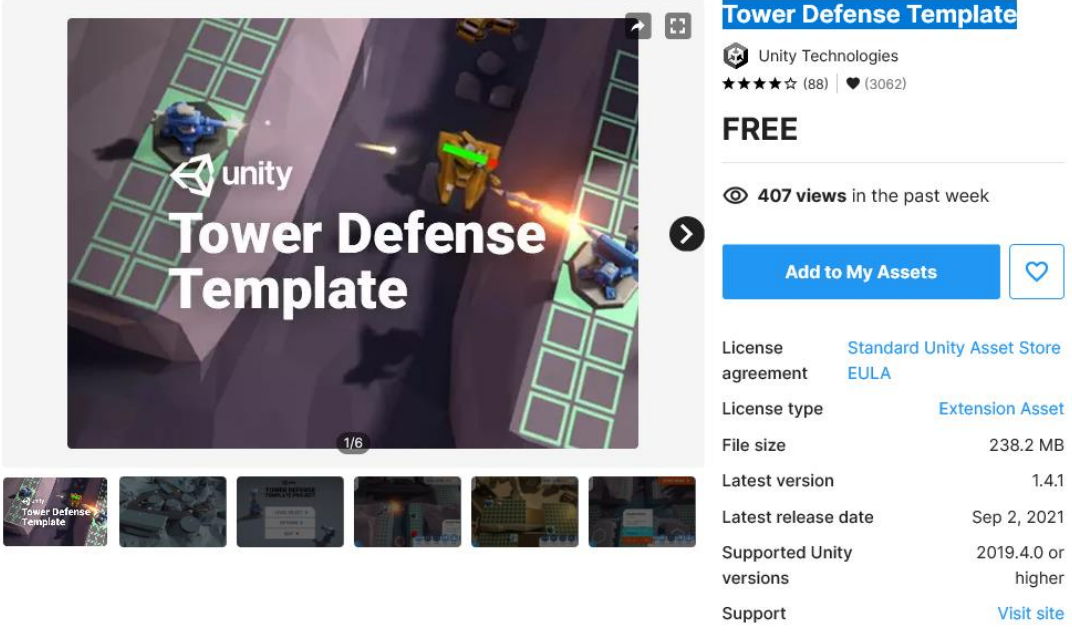
구현한 주요 코드 설명

- 타워 정보
- 방어편성 방법
- DB 저장 로드

지형 제작

사용 에셋
출처 : 에셋스토어

Home > Essentials > Tutorial Projects > Tower Defense Template



Tower Defense Template

Unity Technologies
★★★★☆ (88) | ♥ (3062)

FREE

👁 407 views in the past week

[Add to My Assets](#)

License agreement [Standard Unity Asset Store EULA](#)

License type [Extension Asset](#)

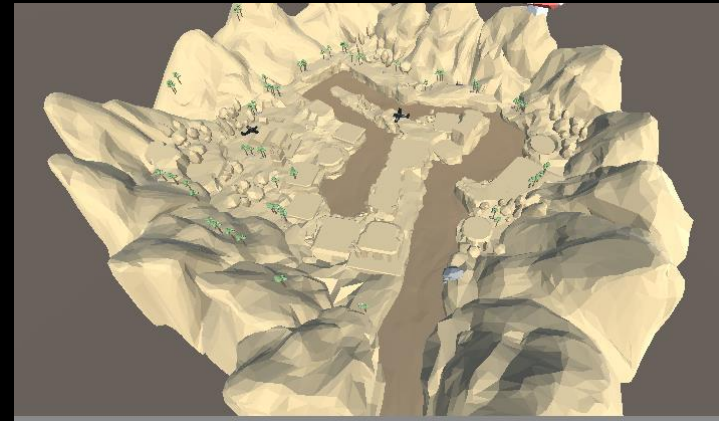
File size 238.2 MB

Latest version 1.4.1

Latest release date Sep 2, 2021

Supported Unity versions 2019.4.0 or higher

Support [Visit site](#)



패키지 안에 있는 지형을 가져와 Material의 색깔을 조절하여 알맞게 만들어 보았다.

추가적으로 돌, 나무, 소품 등을 추가 하여 꾸며 보았다.

지형 위치 설정

특정 마크가 있는 곳에 만
타워를 설치 할 수 있게 하였다



적절한 위치에 타워 설치 포인트를 잡았다.

타워 제작

타워 컨트롤 제작 -> 타워컨트롤러를 상속받아 여러 종류의 타워를 만든다.

```
참조 11개
public enum towerCondition
{
    idle,
    attack
}

@ Unity 스크립트 (자산 참조 1개) | 참조 11개
public class TowerController : MonoBehaviour
{
    [Header("UI 관련 변수")]
    public GameObject UICanvas = null;
    public Image HPBar = null;
    [HideInInspector] public GameObject MainCamera;

    [Header("타워 모델링 관련 변수")]
    public GameObject towerHead = null;
    public GameObject SpawnPoint = null;

    [Header("사망시 나올 이펙트")]
    public GameObject bombObj;

    //----- 타워 정보 관련 변수
    protected towerCondition TwCond = towerCondition.idle;
    protected float delta = 0;
    [HideInInspector] public float TowerHP = 0;

    //List 자료구조를 통해 공격범위에 들어온 순서대로 공격한다
    protected List<GameObject> enemyList = new List<GameObject>();
    protected GameObject target;

    //----- 타워 공격 관련 변수
    public TowerInfo twInfo = new TowerInfo();
    protected Vector3 dic = Vector3.zero; //타워헤드 몸 방향 계산점

    // Start is called before the first frame update
    @ Unity 메시지 | 참조 0개
    void Start()
    {
        UICanvas.gameObject.SetActive(false);
    }

    참조 1개
    public void SetStat(TowerInfo info)
    {
        twInfo = info;
        TowerHP = twInfo.maxHP;
    }
}
```

```
@ Unity 메시지 | 참조 0개
void Update()
{
    TurretAI();
}

//포탄의 AI
참조 2개
protected virtual void TurretAI()
{
    //타워가 공격 상태가 아닐 때 포탄이 회전한다.
    if (TwCond == towerCondition.idle && towerHead != null)
    {
        Effectoff();
        towerHead.transform.Rotate(0, 60f * Time.deltaTime, 0);
    }
    //타워가 공격상태일 때
    else if (TwCond == towerCondition.attack)
    {
        if (!target)
        {
            //리스트 내 가장 먼저 들어온 유닛을 타겟으로 지정(null 이면 삭제)
            if (enemyList.Count != 0)
            {
                for (int i = 0; i < enemyList.Count;)
                {
                    if (enemyList[i].Equals(null) || enemyList[i].gameObject.activeSelf.Equals(false))
                        enemyList.RemoveAt(i);
                    else
                        i++;
                }
            }
            else
            {
                TwCond = towerCondition.idle;
                towerHead.transform.rotation = Quaternion.identity;
                return;
            }
            if (enemyList.Count >= 1)
                target = enemyList[0];
        }
        else
        {
            dic = target.transform.position;
            towerHead.transform.LookAt(dic);
            //공격 범위 내에 유닛이 없으면 -> 다시 idle 상태로 바꾼다.
            if (enemyList.Count <= 0)
            {
                TwCond = towerCondition.idle;
                return;
            }
        }
        delta += Time.deltaTime;
        if (delta >= twInfo.atkCycle)
        {
            Attack();
            delta = 0f;
        }
    }
}
```

```
참조 7개
protected virtual void Attack()
{
    //공격
    //추가적으로 적용이 필요하면 사용
}

참조 6개
protected virtual void Effectoff()
{
    //사용 이펙트 및 효과 끄기
    //추가적으로 적용이 필요하면 사용
}

참조 2개
protected virtual void Dead()
{
    //죽음
    //추가적으로 적용이 필요하면 사용
}

참조 1개
public void TakeDamage(float value = 10)
{
    TowerHP -= value;
    HPBar.fillAmount = TowerHP / twInfo.maxHP;
    UICanvas.gameObject.SetActive(true);

    if (TowerHP <= 0)
    {
        GameObject BombEffect = Instantiate(bombObj);
        BombEffect.transform.position = transform.position;
        Dead();
        // 타워 오브젝트를 구성하는 가장 상위 오브젝트를 destroy한다
        Destroy(gameObject);
        Destroy(BombEffect.gameObject, 2.0f);
        GameManager.Inst.AddGold(twInfo.gold);
    }
}

//공격 범위 감지
@ Unity 메시지 | 참조 0개
void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("Unit"))
    {
        //공격 범위 내에 적대 유닛이 들어오면 리스트에 추가하고 공격 상태로 전환
        if (!enemyList.Contains(other.gameObject)) //중복 체크
        {
            enemyList.Add(other.gameObject);
            TwCond = towerCondition.attack;
        }
    }
}

@ Unity 메시지 | 참조 0개
private void OnTriggerExit(Collider other)
{
    //공격 범위 밖으로 적이 나갈경우 리스트에서 제거
    if (enemyList.Contains(other.gameObject)) //중복 체크
    {
        enemyList.Remove(other.gameObject);
        target = null;
    }
}

참조 20개
public class TowerInfo
{
    public int towerType = 0; //타워 타입
    public int lv = 0; //타워 레벨
    public int price = 0; //타워 가격

    public float damage = 0; //타워 공격력
    public float atkCycle = 0; //공격 주기
    public float maxHP = 0; //타워 체력
    public int gold = 10; //타워 격파 시 골드
    public int cost = 1; //코스트
}
```

타워 제작

타워 제작 예시) 머신건 타워

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

// Unity 스크립트(자산 참조 1개) | 참조 0개
public class MachineGunTowerController : TowerController
{
    //----- 타워 공격 관련 변수
    [Header("타워 공격 관련 변수")]
    public ParticleSystem Particle = null;

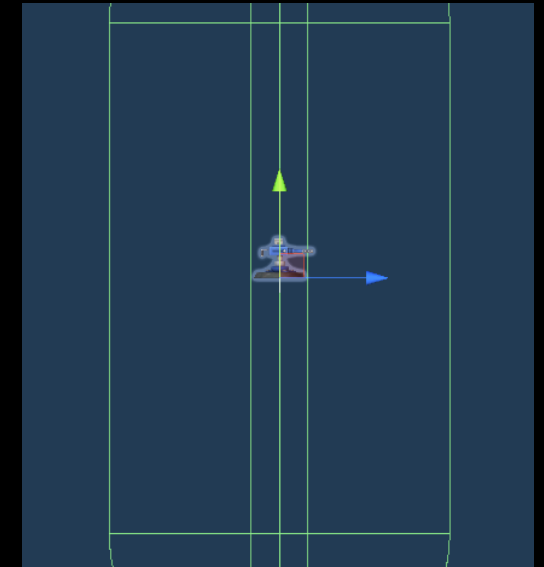
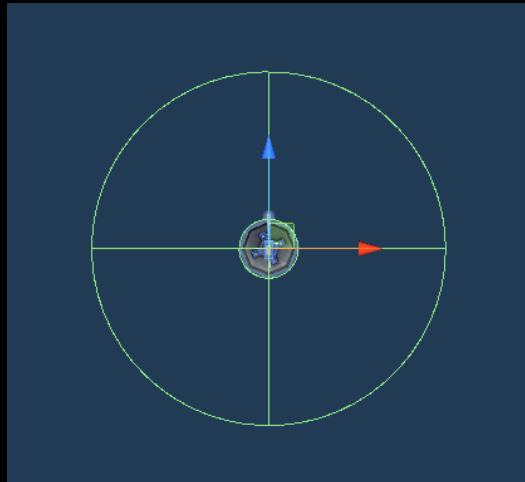
    참조 2개
    protected override void Attack()
    {
        //이펙트
        Particle.Play();
        target.GetComponent<UnitDamage>().TakeDamage(twInfo.damage);
        SoundManager.Instance.PlayEffSound("MachineGunTower", this.transform.position);
    }

    참조 2개
    protected override void Effectoff()
    {
        Particle.Stop();
    }
}
```

```
// Unity 스크립트(자산 참조 6개) | 참조 3개
public class TowerDamage : MonoBehaviour
{
    TowerController tc;
    // Start is called before the first frame update
    // Unity 메시지 | 참조 0개
    void Start()
    {
        tc = GetComponentInParent<TowerController>();
    }

    //타워가 공격 받으면 호출
    참조 3개
    public void TakeDamage(float value = 10)
    {
        tc.TakeDamage(value);
    }
}
```

바깥 콜라이더 는 사거리 콜라이더로 사용
내부 콜라이더는 피격용 콜라이더로 사용 합니다.



내부 피격용 콜라이더 적용할
데미지 적용 스크립 별도 제작

방어 편성

현재 가지고 있는 타워 표시
레벨 및 설치 개수 표시



현재 상태 저장하기

현재 설치된 타워 개수
총 사용 코스트

방어 편성

선택한 노드 정보



선택한 타워가 있을 경우 사거리 표시 및 다른 UI 창 안보이게 하기



타워 셋팅 코드

타워 셋팅을 위한 변수 선언

```
@Unity 스크립트 | 차인 | 참조 1개 | 참조 5개
public class SetTowerMgr : MonoBehaviour
{
    public static SetTowerMgr inst;

    //타워 모델 프리팹
    public List<GameObject> towerModelList = new List<GameObject>();
    public Sprite[] towerSprite;
    //현재 들고 있는 타워 오브젝트
    SetTower dragobj = null;
    //위치 계산용
    private Vector3 pos;

    //전체 타워 UI
    public GameObject setTowerUI;

    [Header("UI")]
    public Text countTxt;
    public Text costTxt;

    [Header("LogBox Info")]
    public GameObject InfoPanel;
    public Image towerImg;
    public Text statusTxt;
    public Text infoTxt;

    //타워 설치 위치를 담을 배열
    GameObject[] spawnPoint;

    //타워 타입별 위치 디렉터리
    static Dictionary<TowerType, List<int>> DicttowerPos = new Dictionary<TowerType, List<int>>();

    //타워 설치용 UI
    [Header("TowerNode UI")]
    public Transform content = null;
    public GameObject nodeObj = null;

    int maxTower = 15; //최대 설치 가능 타워
    int maxCost = 99; //최대 설치 가능 코스트
    int curTowerCount = 0; //현재 설치 된 타워 갯수
    int curTowerCost = 0; //현재 설치 된 타워 코스트
    TowerType curTowerType; //현재 설치 중인 타워 타입

    //UI에 등록된 설치용 노드입니다.
    Dictionary<TowerType, TowerNode> towerNodeDic = new Dictionary<TowerType, TowerNode>();

    public bool bRetry = false; //재설치 중인지

    public Button saveBtn;
    public Button exitBtn;

    private string UpdateMyDefSetUrl = "http://pmaker.dothome.co.kr/pMaker_7Gi/UpdateMyDefSet.php";
}
```

```
@Unity 메시지 | 참조 0개
private void Awake()
{
    inst = this;
    InitTower();

    spawnPoint = GameObject.FindGameObjectsWithTag("TowerPos");
}

참조 1개
void InitTower()
{
    towerSprite = Resources.LoadAll<Sprite>("DefTeam/TowerImg");

    //DBContainer.towerInfoList; -> 타워 DB 상의 모든 정보
    //GlobalValue.TowerLvDic; -> 내가 가진 타워 Lv
    foreach (TowerInfo info in DBContainer.towerInfoList)
    {
        if (!GlobalValue.myInfo.TowerLvDic.ContainsKey(info.towerType))
        {
            GlobalValue.myInfo.TowerLvDic.Add(info.towerType, 0);
        }
        //보유 타워의 레벨과 타임이 일치하는 경우에
        //UI 노드를 인스턴스화 하고 노드를 디렉터리에 추가한다.
        //UI 노드 자체에 TowerInfo를 붙여서 바로 접근 가능.
        if (GlobalValue.myInfo.TowerLvDic[info.towerType] == info.lv)
        {
            GameObject node = Instantiate(nodeObj, content);
            TowerNode towerNode = node.GetComponent<TowerNode>();

            towerNode.SetNode(info);
            towerNodeDic.Add((TowerType)info.towerType, towerNode);
        }
    }

    // Start is called before the first frame update
    @Unity 메시지 | 참조 0개
    void Start()
    {
        LoadDefDeck();

        //버튼 초기화
        if (saveBtn != null)
            saveBtn.onClick.AddListener(() =>
            {
                GlobalValue.myInfo.mydefset = JsonMgr.DefDeckToStr(DicttowerPos);
                UpdateMyDefSet();
            });

        if (exitBtn != null)
            exitBtn.onClick.AddListener(() => { LoadingManager.Instance.LoadScene("Lobby"); });

        SoundManager.Instance.PlayBGM("SetDefBGM");

        SetText();
    }

    // Update is called once per frame
}
```

```
void Update()
{
    if (!ReferenceEquals(dragobj, null))
    {
        Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition); //현재 마우스 위치 값을 입력 받습니다.
        RaycastHit rayhit;
        if (Physics.Raycast(ray, out rayhit, Mathf.Infinity))
        {
            if (Input.GetMouseButton(0))
            {
                if (rayhit.collider.CompareTag("TowerPos"))
                {
                    pos = rayhit.collider.transform.position;
                    dragobj.transform.position = pos;
                }
                else
                {
                    pos = rayhit.point;
                    pos.y += 1.0f;
                    dragobj.transform.position = pos;
                }
            }
        }

        if (Input.GetMouseButtonUp(0))
        {
            if (rayhit.collider.CompareTag("TowerPos") && (curTowerCost + towerNodeDic[curTowerType].twInfo.cost <= maxCost))
            {
                SettingTower(rayhit.collider.gameObject);
            }
            else
            {
                if (bRetry)
                {
                    curTowerCount--;
                    towerNodeDic[curTowerType].SetTowerCount(-1);
                    curTowerCost -= towerNodeDic[curTowerType].twInfo.cost;

                    DicttowerPos[curTowerType].Remove(dragobj.pointIdx);
                    if (DicttowerPos[curTowerType].Count <= 0)
                        DicttowerPos.Remove(curTowerType);

                    bRetry = false;
                }
                //기존 링이여 불러 올기
                spawnPoint[dragobj.pointIdx].layer = 0;

                //들고 있는 타워 삭제
                Destroy(dragobj.gameObject);
                dragobj = null;
                OnTowerUI();

                SetText();
            }
        }
    }
    else
    {
        if (Input.GetMouseButtonDown(0))
        {
            Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition); //현재 마우스 위치 값을 입력 받습니다.
            RaycastHit rayhit;
            if (Physics.Raycast(ray, out rayhit, Mathf.Infinity))
            {
                if (rayhit.collider.gameObject.layer.Equals(LayerMask.NameToLayer("TOWERS")))
                {
                    RetryTowerSet(rayhit.collider.gameObject);
                }
            }
        }
    }
}
```

타워 셋팅 코드

```
public void SpawnTowerModel(TowerType towertype)
{
    if (curTowerCount >= maxTower) //최대 설치 넣음
    {
        Debug.Log("최대 건설 갯수를 초과 했습니다.");
        return;
    }

    dragobj = Instantiate(towerModelList[(int)towertype]).GetComponent<SetTower>();
    dragobj.gameObject.layer = 2;
    curTowerType = towertype;

    dragobj.GetComponent<SetTower>().towertype = towertype;
    dragobj.arOn();
    OffTowerUI();
}

함프 1개
void RetryTowerSet(GameObject obj)
{
    dragobj = obj.GetComponent<SetTower>();
    dragobj.gameObject.layer = 2;
    OffTowerUI();

    dragobj.arOn();
    spawnPoint[dragobj.pointIdx].gameObject.layer = 0;
    curTowerType = dragobj.towertype;
    bRetry = true;

    SetText();
}

함프 1개
void SettingTower(GameObject posObj)
{
    pos = posObj.transform.position;
    pos.y += 1.5f;
    dragobj.transform.position = pos;
    dragobj.gameObject.layer = LayerMask.NameToLayer("TOWERS");

    int towerPosIdx = System.Array.IndexOf(spawnPoint, posObj);

    if (!bRetry) //현재 다시 설치하는게 아니라면 (위치 변경이 아니라면)
    {
        //새로 추가 하기
        if (!DictowerPos.ContainsKey(curTowerType))
        {
            List<int> pos = new List<int>();
            pos.Add(towerPosIdx);
            DictowerPos.Add(curTowerType, pos);
        }
        else
            DictowerPos[curTowerType].Add(towerPosIdx);

        dragobj.pointIdx = towerPosIdx;

        curTowerCount++;
        curTowerCost += towerNodeDic[curTowerType].twInfo.cost;
        towerNodeDic[curTowerType].SetTowerCount(1);
    }
    else
    {
        //가운 위치 변경
        for (int i = 0; i < DictowerPos[curTowerType].Count; i++)
        {
            if (DictowerPos[curTowerType][i].Equals(dragobj.pointIdx))
            {
                DictowerPos[curTowerType][i] = towerPosIdx;
                dragobj.pointIdx = towerPosIdx;
            }
        }

        bRetry = false;
    }

    posObj.layer = 2;
    dragobj.arOff();
    dragobj = null;
    OnTowerUI();
    SetText();
}
```

```
참조 2개
void OnTowerUI()
{
    setTowerUI.SetActive(true);
    InfoPanel.SetActive(false);
}

참조 2개
void OffTowerUI()
{
    setTowerUI.SetActive(false);
}

참조 1개
public void SetInfoBox(TowerInfo info)
{
    InfoPanel.SetActive(true);

    statusTxt.text = Enum.GetDescription((TowerType)info.towertype) + "\n\n";
    statusTxt.text += "공격 : " + info.damage + "\n";
    statusTxt.text += "공격속도 : " + info.atkcycle + "\n";
    statusTxt.text += "체력 : " + info.maxHP + "\n";
    towerImg.sprite = towersprite[info.towertype];

    switch((TowerType)info.towertype)
    {
        case TowerType.Rocket:
            infoTxt.text = "로켓을 발사하는 가장 기본적인 타워입니다";
            break;
        case TowerType.MG:
            infoTxt.text = "공격력은 약하지만 공격 속도가 빠른 타워입니다";
            break;
        case TowerType.Laser:
            infoTxt.text = "빠른 공격속도의 레이저를 발사하는 타워입니다";
            break;
        case TowerType.Fire:
            infoTxt.text = "넓은 범위에 화염을 발사하는 타워입니다";
            break;
        case TowerType.Buff:
            infoTxt.text = "주변 타워의 공격력을 상승시키는 타워입니다";
            break;
        case TowerType.Super:
            infoTxt.text = "많은 미사일을 발사하는 가장 강력한 타워입니다";
            break;
    }
}

참조 1개
public void OffInfoBox()
{
    InfoPanel.SetActive(false);
}

참조 6개
void SetText()
{
    countTxt.text = "설치 타워 : " + curTowerCount + " / " + maxTower;
    costTxt.text = "설치 코스트 : " + curTowerCost + " / " + maxCost;
}
```

```
함프 1개
void LoadDefDeck()
{
    DictowerPos.Clear();
    DictowerPos = JsonMgr.DefDeckToDic(GlobalValue.myInfo.myDefSet);
    // 각 위치에 타워 설치
    foreach (var item in DictowerPos)
    {
        for (int i = 0; i < item.Value.Count; i++)
        {
            //타워별 프리팹 생성
            SetTower setTower = Instantiate(towerModelList[(int)item.Key]).GetComponent<SetTower>();

            //타워 지정
            setTower.towertype = item.Key;
            //지정된 포인터에서 오버래프 위치 지정
            pos = spawnPoint[item.Value[i]].transform.position;
            pos.y += 1.0f;
            setTower.gameObject.transform.position = pos;
            setTower.gameObject.layer = LayerMask.NameToLayer("TOWERS");

            setTower.pointIdx = item.Value[i];

            //사거리 표시 끄기
            setTower.arOff();

            //카운트 해주기
            curTowerCount++;
            curTowerCost += towerNodeDic[item.Key].twInfo.cost;

            towerNodeDic[item.Key].SetTowerCount(1);

            //타워의 발한 레이어 변경
            spawnPoint[item.Value[i]].gameObject.layer = 2;
        }
    }

    //텍스트 갱신
    SetText();
}

함프 1개
void UpdateMyDefSet()
{
    StartCoroutine(UpdateMyDefSetCo());
}

함프 1개
IEnumerator UpdateMyDefSetCo()
{
    WWWForm form = new WWWForm();
    form.AddField("Input_id", GlobalValue.myInfo.userID, System.Text.Encoding.UTF8);
    form.AddField("Input_mydefset", GlobalValue.myInfo.mydefset);

    UnityWebRequest a_www = UnityWebRequest.Post(UpdateMyDefSetUrl, form);
    yield return a_www.SendWebRequest();

    if (a_www.error == null)
    {
        System.Text.Encoding enc = System.Text.Encoding.UTF8;
        string sReturn = enc.GetString(a_www.downloadHandler.data);
        if (sReturn.Contains("OK_"))
        {
            LoadingManager.Instance.LoadScene("Lobby");
        }
    }
    else
    {
        Debug.Log(a_www.error);
    }
}
```

타워 셋팅 코드

생성된 노드에 들어가는 코드

```
Ⓢ Unity 스크립트(자산 참조 1개) | 참조 4개
public class TowerNode : MonoBehaviour, IPointerDownHandler, IPointerEnterHandler, IPointerExitHandler
{
    [HideInInspector] public TowerInfo twInfo;

    public Text towerName;
    public Image towerImg;
    [HideInInspector] public int maxCount = 5;
    [HideInInspector] public int count = 0;

    public Text countTxt;

    // Start is called before the first frame update
    Ⓢ Unity 메시지 | 참조 0개
    void Start()
    {
        countTxt.text = count + " / " + maxCount;
    }

    참조 1개
    public void SetNode(TowerInfo info)
    {
        twInfo = info;

        towerImg.sprite = SetTowerMgr.inst.towersSprite[info.towerType];
        towerName.text = "Lv." + info.lv + " " + Enum.GetDescription((TowerType)info.towerType);
    }

    참조 0개
    public void OnPointerDown(PointerEventData eventData)
    {
        if (count >= maxCount)
            return;

        SetTowerMgr.inst.SpawnTowerModel((TowerType)twInfo.towerType);
    }

    참조 0개
    public void OnPointerEnter(PointerEventData eventData)
    {
        SetTowerMgr.inst.SetInfoBox(twInfo);
    }

    참조 0개
    public void OnPointerExit(PointerEventData eventData)
    {
        SetTowerMgr.inst.OffInfoBox();
    }

    참조 3개
    public void SetTowerCount(int add)
    {
        count += add;
        countTxt.text = count + " / " + maxCount;
    }
}
```

생성된 타워에 들어가는 코드

```
1 Ⓢ using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.UI;
5
6 Ⓢ Unity 스크립트(자산 참조 6개) | 참조 6개
7 public class SetTower : MonoBehaviour
8 {
9     public TowerType towertype;
10    public GameObject attackRange;
11    public int pointIdx;
12
13    참조 2개
14    public void arOn()
15    {
16        attackRange.SetActive(true);
17    }
18
19    참조 2개
20    public void arOff()
21    {
22        attackRange.SetActive(false);
23    }
24 }
```

DB 변환 코드

서버 저장을 위한 타워 위치
JSON 변환

```
public static string DefDeckToStr(Dictionary<TowerType, List<int>> DictTowerPos)
{
    string Return = string.Empty;
    try
    {
        var jsonAllTower = new JObject();
        var jsonKeyType = new JSONArray();

        foreach (var item in DictTowerPos)
        {
            var jsonTower = new JObject();

            var json = new JSONArray();
            for (int i = 0; i < item.Value.Count; i++)
            {
                json.Add(item.Value[i]);
            }
            jsonTower.Add("P", json);
            jsonKeyType.Add((int)item.Key); //enum -> int -> string
            jsonAllTower.Add(((int)item.Key).ToString(), jsonTower);
        }

        jsonAllTower.Add("K", jsonKeyType);

        Return = jsonAllTower.ToString();
    }
    catch (Exception ex)
    {
        Debug.Log(ex.ToString());
    }
    return Return;
}
```

타워 위치(JSON)를 실제
딕셔너리로 변환

```
참조 2개
public static Dictionary<TowerType, List<int>> DefDeckToDic(string str)
{
    Dictionary<TowerType, List<int>> DictTowerPos = new Dictionary<TowerType, List<int>>();
    try
    {
        if (string.IsNullOrEmpty(str) || str == "")
            return DictTowerPos;

        DictTowerPos.Clear(); // 한번 초기화

        var json = JSON.Parse(str);
        var key = json["K"];
        for (int i = 0; i < key.Count; i++)
        {
            List<int> pos = new List<int>();
            int keyCount = json[key[i].ToString()][ "P" ].Count;
            int keyint = key[i].AsInt;
            string keystr = key[i].ToString();
            for (int ii = 0; ii < keyCount; ii++)
                pos.Add(json[keystr][ "P" ][ii]);

            DictTowerPos.Add((TowerType)keyint, pos);
        }
    }
    catch (Exception ex)
    {
        Debug.Log(ex.ToString());
    }
    return DictTowerPos;
}
```