게임 프로그래밍 포트폴리오

이학석

목차

1. 유니티 3D
① 게임 소개
② UI
  1 아이템 슬롯
③ 환경
④ 캐릭터
  1 공격
  2 회피
  3 콤보
  4 효과음
  5 능력치
  6 장비
  7 적
  8 최종 보스
⑤ NPC
⑥ 저장
⑦ 기타

2. 유니티 2D (NGUI)
3. Cocos2d-X
4. IOCP Server

1. 유니티 3D

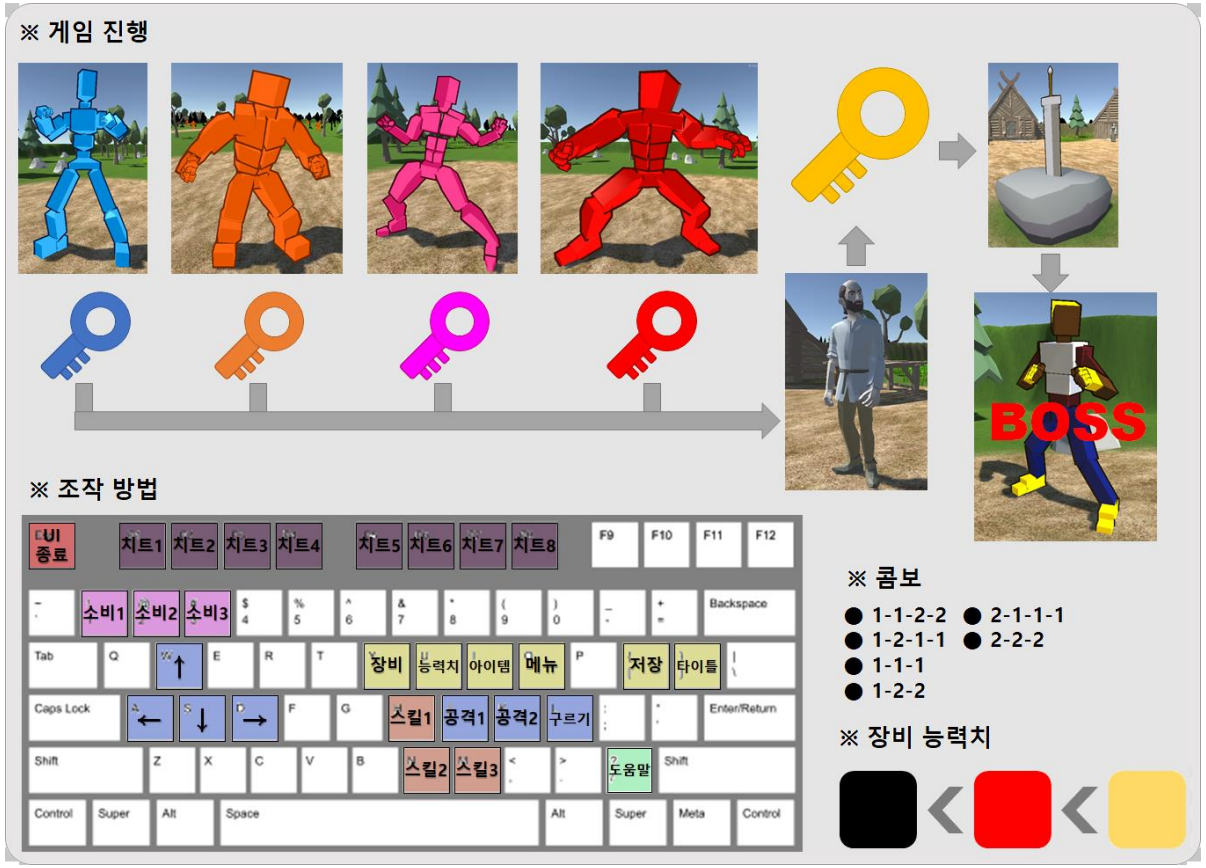
① 게임 소개

유니티 3D를 이용하여 만든 포트폴리오 입니다.
유니티 에셋 스토어에 있는 에셋을 사용하여 만든 오리지널 게임입니다.

영상 링크 : https://drive.google.com/open?id=1Wxwen8VIctL7ADDsyCxXnpRsX1Ipy09v
프로젝트 링크 : https://drive.google.com/open?id=15XJZFwS7u_ym19nM_wy0BCARvxcim9hT

1. 유니티 3D

① 게임 소개



게임은 4방향의 보스를 처치하고 얻은 열쇠를 모아 NPC에게 교환하고
이를 통해 얻은 열쇠를 사용하여 숨겨진 숲으로 이동해 최종보스를 처치하는 게임입니다.

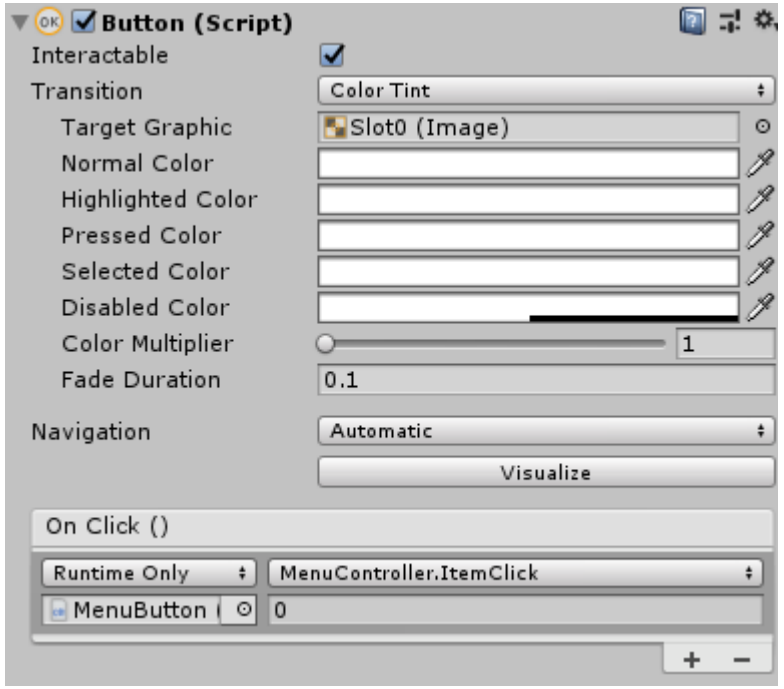
1. 유니티 3D

② UI

UI는 UGUI를 사용해 만들었습니다.
UI의 크기는 Anchor를 조절하고 텍스트는 Best Fit으로 설정하여
해상도에 따라 화면에 맞도록 달라지게 하였습니다.

1. 유니티 3D

②-1 UI (아이템 슬롯)



아이템 슬롯은 플레이어의 인벤토리의 변동이 있을 때마다 각 슬롯의 스프라이트를 변경하는 방식으로 만들었습니다.
아이템을 클릭 하였을 때 Onclick에서 인자를 받아와 클릭한 슬롯의 Index를 통해 플레이어가 사용한 아이템을 확인하고 효과를 발동시켰습니다.

1. 유니티 3D

②-1 UI (아이템 슬롯)

```
public void ItemClick(int slot)
{
    if (menu.CheckTrade())
    {
        player.SellItem(slot);
        menu.ItemSlotRefresh();
    }
    else
    {
        if (player.UseItem(slot))
            menu.ItemSlotRefresh();
    }
}
```

-아이템 클릭 시 상점UI 확인 후 사용/판매

```
public void ItemSlotRefresh()
{
    if (!itemSlot.activeSelf)
        return;
    foreach (Image s in slot)
    {
        s.sprite = null;
        s.gameObject.SetActive(false);
    }
    List<int> itemList = GameManager.instance.GetPlayer().GetComponent<MyPlayerController>().GetItemList();
    int i = 0;
    foreach (int item in itemList)
    {
        slot[i].sprite = itemSprite[item];
        slot[i].gameObject.SetActive(true);
        i++;
    }
}
```

-플레이어의 아이템을 확인 후 스프라이트 변경

```
public bool UseItem(int index)
{
    if (death)
        return false;
    if (itemList[index] < 5)
        return false;
    int equip = 0;
    switch (itemList[index])
    {
        case (int)ItemIndex.smallpotion:
            HPChange(20);
            spNum--;
            ItemChange();
            break;
        case (int)ItemIndex.bigpotion:
            HPChange(50);
            bpNum--;
            ItemChange();
            break;
        case (int)ItemIndex.biggold:
            GoldChange(100);
            break;
        case (int)ItemIndex.portal:
            ptNum--;
            GameManager.instance.DeActiveGameObjects();
            GameManager.instance.Loading((int)LoadingIndex.Game);
            break;
        case (int)ItemIndex.blackweapon:
        case (int)ItemIndex.redweapon:
        case (int)ItemIndex.goldweapon:
            equip = itemList[index];
            if (equipWeapon == 0)
                itemList.RemoveAt(index);
            else
                itemList[index] = equipWeapon;
            EquipItem(equip);
            GameManager.instance.menu.EquipOpen();
            return true;
        case (int)ItemIndex.blackarmor:
        case (int)ItemIndex.redarmor:
        case (int)ItemIndex.goldarmor:
            equip = itemList[index];
            if (equipArmor == 0)
                itemList.RemoveAt(index);
            else
                itemList[index] = equipArmor;
            EquipItem(equip);
            GameManager.instance.menu.EquipOpen();
            return true;
    }
    itemList.RemoveAt(index);
    return true;
}
```

-플레이어의 아이템 사용

1. 유니티 3D

③ 환경

마을에서 필드로 나가는 입구마다 콜라이더를 만들어 충돌 시
그 지역에 맞는 배경음악을 재생하고 그 지역명을 상단에 띄우도록 하였습니다.
NevMesh를 사용하여 캐릭터가 이동할 수 없는 장애물을 설정하였습니다.

1. 유니티 3D

③ 환경

```
private void OnTriggerEnter(Collider other)
{
    if (!other.name.Equals("Player"))
        return;
    int placeIndex=0;
    if (gameObject.name.Equals("Town"))
        placeIndex = (int)PlaceIndex.Town;
    else if (gameObject.name.Equals("East"))
        placeIndex = (int)PlaceIndex.East;
    else if (gameObject.name.Equals("West"))
        placeIndex = (int)PlaceIndex.West;
    else if (gameObject.name.Equals("South"))
        placeIndex = (int)PlaceIndex.South;
    else if (gameObject.name.Equals("North"))
        placeIndex = (int)PlaceIndex.North;
    else if (gameObject.name.Equals("Hidden"))
        placeIndex = (int)PlaceIndex.Hidden;
    GameManager.instance.ChangePlace(placeIndex);
}
```

-콜라이더를 통한 지역 변경

```
public void ChangePlace(int index)
{
    if (index == playerPlace)
        return;
    if (index == (int)PlaceIndex.Town)
    {
        PlaySound(townMusic, true);
        player.HPChange(999);
    }
    else if (index == (int)PlaceIndex.Hidden)
    {
        PlaySound(bossMusic, true);
    }
    else
    {
        PlaySound(fieldMusic, true);
    }

    playerPlace = index;
    place.ChangePlace(index);
}
```

-지역 변경 시 배경 음악 변경

```
public void ChangePlace(int index)
{
    fadeOut = true;
    timer = 0;
    canvasGroup.alpha = 1;
    switch (index)
    {
        case (int)PlaceIndex.Town:
            placetext.text = "마을";
            break;
        case (int)PlaceIndex.East:
            placetext.text = "동쪽 숲";
            break;
        case (int)PlaceIndex.West:
            placetext.text = "서쪽 숲";
            break;
        case (int)PlaceIndex.South:
            placetext.text = "남쪽 숲";
            break;
        case (int)PlaceIndex.North:
            placetext.text = "북쪽 숲";
            break;
        case (int)PlaceIndex.Hidden:
            placetext.text = "숨겨진 숲";
            break;
    }
}
```

-지역 명 UI 나타내기

1. 유니티 3D

④-1 캐릭터 (공격)

플레이어의 공격은 두가지 키의 입력을 받아 콤보로 공격하고
애니메이션 이벤트를 사용해 공격마다 공격 데미지를 다르게 하였습니다.
히트박스는 공격 애니메이션에 따라 다른 히트박스를 사용하여 공격범위를 다르게 하였고
코루틴을 사용해 0.1초간 생성하여 타격을 확인하게 하였습니다.

1. 유니티 3D

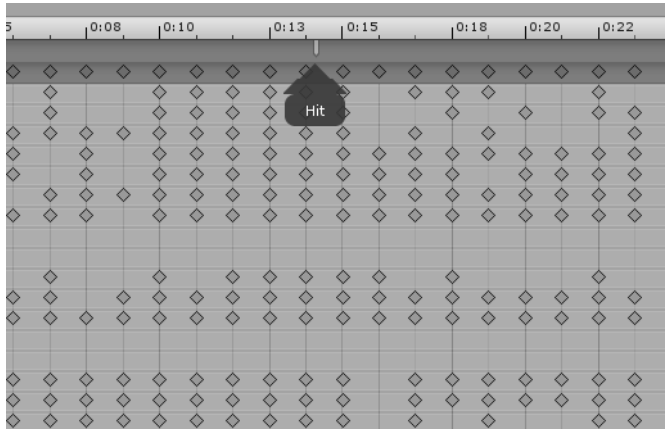
④-1 캐릭터 (공격)

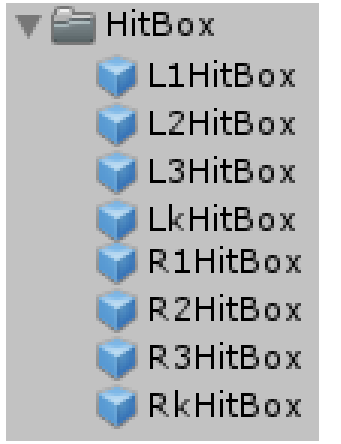
```
private void Hit()
{
    GameObject hitbox = null;
    if (animator.GetCurrentAnimatorStateInfo(0).IsTag("L1"))
    {
        hitbox = Instantiate(Resources.Load("HitBox₩₩L1HitBox"), transform.position + transform.forward + transform.up, new Quaternion(), transform) as GameObject;
        dmg = 13;
    }
    else if (animator.GetCurrentAnimatorStateInfo(0).IsTag("L2"))
    {
        hitbox = Instantiate(Resources.Load("HitBox₩₩L2HitBox"), transform.position + transform.forward + transform.up, new Quaternion(), transform) as GameObject;
        dmg = 15;
    }
    else if (animator.GetCurrentAnimatorStateInfo(0).IsTag("L3"))
    {
        hitbox = Instantiate(Resources.Load("HitBox₩₩L3HitBox"), transform.position + transform.forward + transform.up, new Quaternion(), transform) as GameObject;
        dmg = 17;
    }
    else if (animator.GetCurrentAnimatorStateInfo(0).IsTag("R1"))
    {
        hitbox = Instantiate(Resources.Load("HitBox₩₩R1HitBox"), transform.position + transform.forward + transform.up, new Quaternion(), transform) as GameObject;
        dmg = 16;
    }
    else if (animator.GetCurrentAnimatorStateInfo(0).IsTag("R2"))
    {
        hitbox = Instantiate(Resources.Load("HitBox₩₩R2HitBox"), transform.position + transform.forward + transform.up, new Quaternion(), transform) as GameObject;
        dmg = 18;
    }
    else if (animator.GetCurrentAnimatorStateInfo(0).IsTag("R3"))
    {
        hitbox = Instantiate(Resources.Load("HitBox₩₩R3HitBox"), transform.position + transform.forward + transform.up, new Quaternion(), transform) as GameObject;
        dmg = 14;
    }
    else if (animator.GetCurrentAnimatorStateInfo(0).IsTag("Lk"))
    {
        hitbox = Instantiate(Resources.Load("HitBox₩₩LkHitBox"), transform.position + transform.forward + transform.up, new Quaternion(), transform) as GameObject;
        dmg = 21;
    }
    else if (animator.GetCurrentAnimatorStateInfo(0).IsTag("Rk"))
    {
        hitbox = Instantiate(Resources.Load("HitBox₩₩RkHitBox"), transform.position + transform.forward + transform.up, new Quaternion(), transform) as GameObject;
        dmg = 19;
    }
    dmg *= (1 + str * 0.1f);
    dmg += equipDmg;
    if (atkUp)
        dmg += 5;
    StartCoroutine(ClearHitBox(hitbox));
}
```

-공격 시 애니메이션 태그를 확인하여 다른 히트박스 생성, 데미지 구현

```
private IEnumerator ClearHitBox(GameObject hitbox)
{
    yield return new WaitForSeconds(0.1f);
    Destroy(hitbox);
}
```

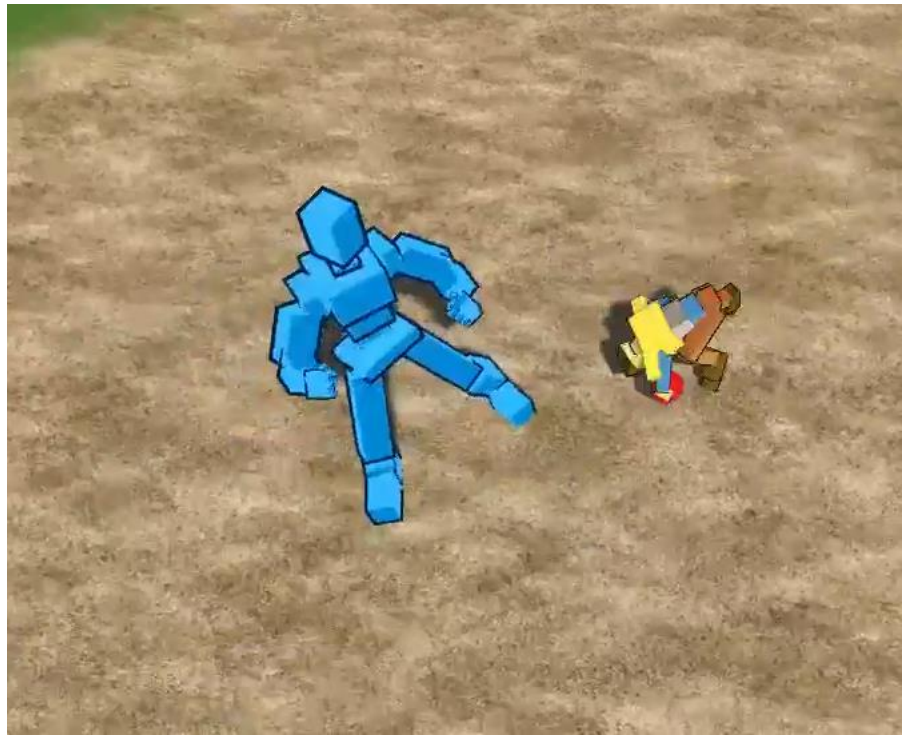
-히트박스 코루틴으로 0.1초만 유지





1. 유니티 3D

④-2 캐릭터 (회피)

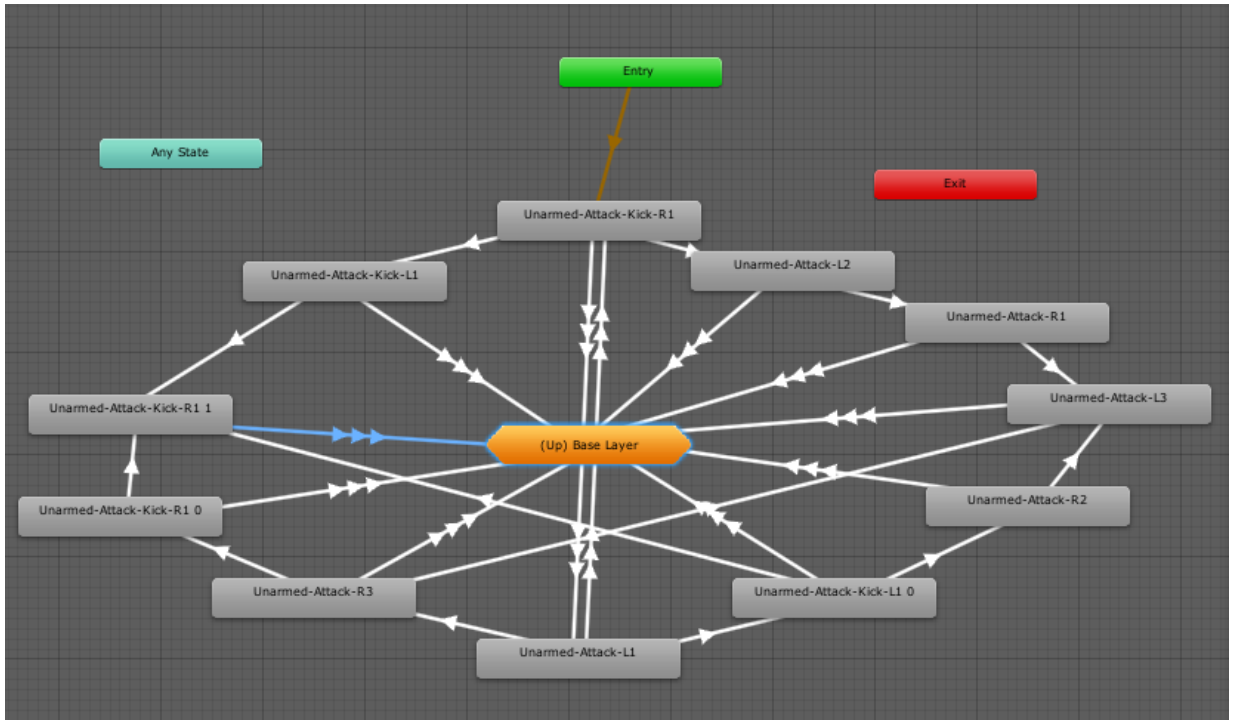


```
private void TryRoll()
{
    if (rolling)
        return;
    if (Input.GetKeyDown(KeyCode.L))
    {
        animator.SetTrigger("roll");
        StartCoroutine(RollCool());
        StartCoroutine(RollInvinc());
    }
}

private IEnumerator RollCool()
{
    rolling = true;
    yield return new WaitForSeconds(1.0f);
    rolling = false;
}

private IEnumerator RollInvinc()
{
    invincibility = true;
    invincStack++;
    yield return new WaitForSeconds(0.5f);
    invincStack--;
    if (invincStack == 0)
        invincibility = false;
}
```

플레이어는 L키를 누르면 굴러서 적의 공격을 회피할 수 있도록 만들었습니다.
코루틴을 사용하여 회피의 쿨타임과 무적 지속시간을 구현하였습니다.

1. 유니티 3D

④-3 캐릭터 (콤보)



캐릭터의 콤보 입력을 쉽게 하기위해 선입력을 구현하였습니다.
키를 입력 받았을 때 애니메이터의 매개변수를 TRUE로 만들어
콤보가 나가게하고 코루틴을 사용하여 일정 시간 후에 FALSE로 만들었습니다.
여러 번 입력 받을 시 입력 카운트를 통해 마지막 입력에서만
매개변수를 FALSE로 만들었습니다.

1. 유니티 3D

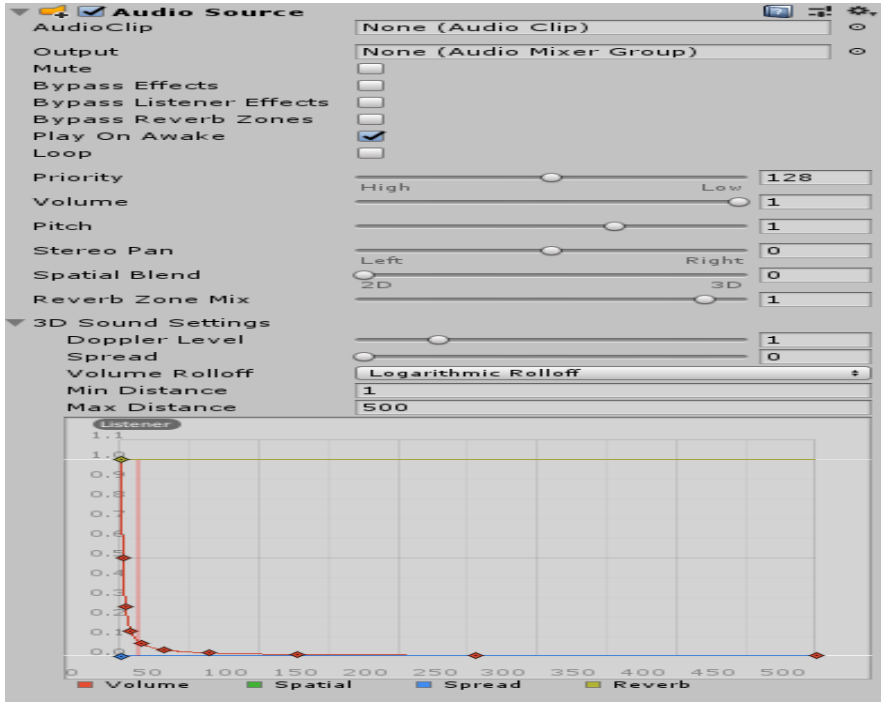
④-3 캐릭터 (콤보)

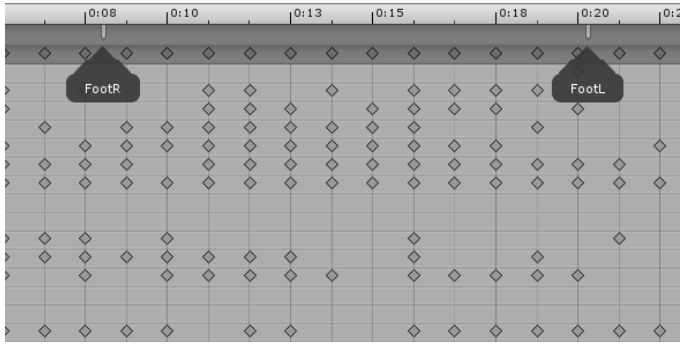
```
private void Attack()
{
    if (Input.GetKeyDown(KeyCode.J))
    {
        StartCoroutine(InputPCorutine());
    }
    if (Input.GetKeyDown(KeyCode.K))
    {
        StartCoroutine(InputKCorutine());
    }
}

private IEnumerator InputPCorutine()
{
    inputP++;
    animator.SetBool("p", true);
    yield return new WaitForSeconds(0.3f);
    inputP--;
    if (inputP == 0)
        animator.SetBool("p", false);
}

private IEnumerator InputKCorutine()
{
    inputK++;
    animator.SetBool("k", true);
    yield return new WaitForSeconds(0.3f);
    inputK--;
    if (inputK == 0)
        animator.SetBool("k", false);
}
```

-코루틴으로 선입력 구현

1. 유니티 3D

④-4 캐릭터 (효과음)





캐릭터의 효과음은 오디오 소스를 통해 구현했습니다.
발소리는 달리기 애니메이션에서 왼발과 오른발이 땅에 닿으면
애니메이션 이벤트가 발생되어 각각의 발소리를 재생합니다.
타격음은 히트박스에 따라 발차기와 주먹의 효과음을 다르게 하였습니다.
또한 타격 유무를 판단하여 타격 시 효과음과 빗나갈 시 효과음도 다르게 하였습니다.

1. 유니티 3D

④-4 캐릭터 (효과음)

```
AudioSource audioSource;
public AudioClip attack;

// Start is called before the first frame update
void Start()
{
    audioSource = GetComponent<AudioSource>();
    if (transform.parent.tag == "PlayerObject")
    {
        transform.parent.GetComponent<MyPlayerController>().PlaySound(Resources.Load<AudioClip>("Sound₩₩Miss"));
    }
    else if (transform.parent.tag == "EnemyObject")
    {
        transform.parent.GetComponent<Enemy>().PlaySound(Resources.Load<AudioClip>("Sound₩₩Miss"));
    }
    else if(transform.parent.tag == "BossObject")
    {
        transform.parent.GetComponent<Boss>().PlaySound(Resources.Load<AudioClip>("Sound₩₩Miss"));
    }
    else if (transform.parent.tag == "FinalBossObject")
    {
        transform.parent.GetComponent<FinalBossController>().PlaySound(Resources.Load<AudioClip>("Sound₩₩Miss"));
    }
}

private void OnTriggerEnter(Collider other)
{
    if (transform.parent.tag == "PlayerObject")
    {
        if (other.tag != "EnemyObject"&& other.tag != "BossObject"&& other.tag != "FinalBossObject")
            return;
        transform.parent.GetComponent<MyPlayerController>().PlaySound(attack);
    }
    else
    {
        if (other.tag != "PlayerObject")
            return;
        if (other.GetComponent<MyPlayerController>().GetInvinc())
            return;
        if (transform.parent.tag == "EnemyObject")
        {
            transform.parent.GetComponent<Enemy>().PlaySound(attack);
        }
        else if (transform.parent.tag == "BossObject")
        {
            transform.parent.GetComponent<Boss>().PlaySound(attack);
        }
        else if (transform.parent.tag == "FinalBossObject")
        {
            transform.parent.GetComponent<FinalBossController>().PlaySound(attack);
        }
    }
}
```

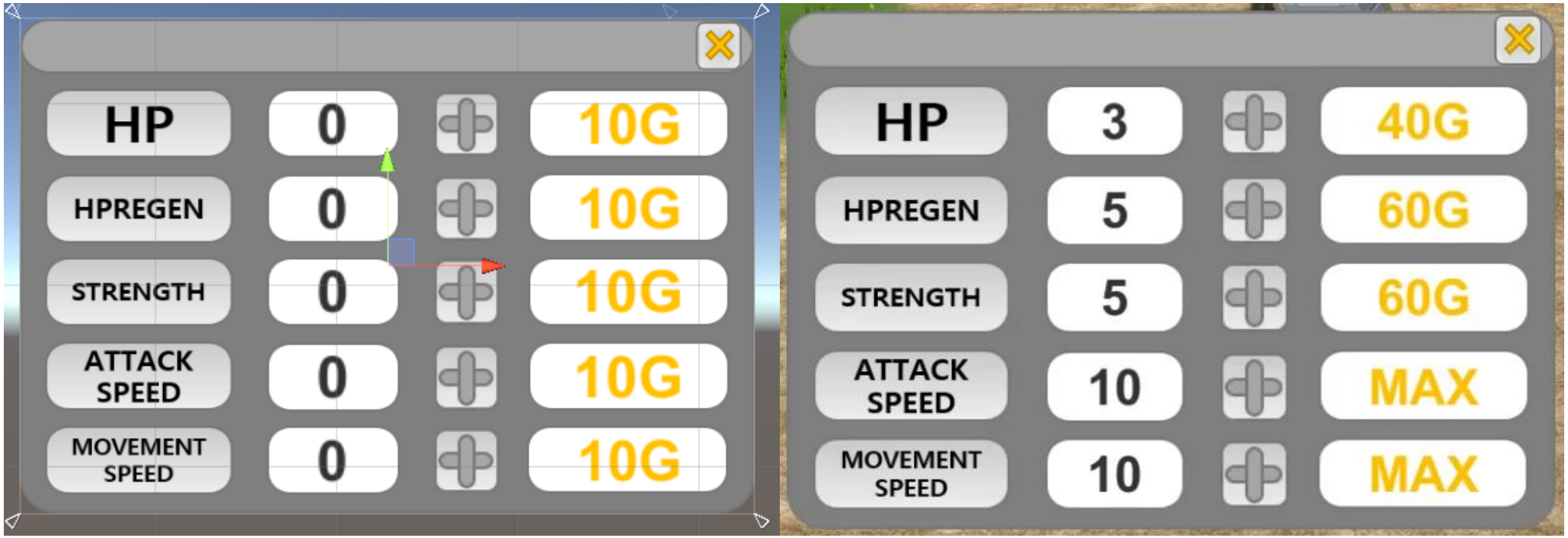
-타격 유무에 따른 효과음 재생

```
private void FootL()
{
    PlaySound(footl);
}

private void FootR()
{
    PlaySound(footr);
}

public void PlaySound(AudioClip clip)
{
    audioSource.Stop();
    audioSource.clip = clip;
    audioSource.time = 0;
    audioSource.Play();
}
```

-애니메이션 이벤트를 통한 발소리 재생

1. 유니티 3D

④-5 캐릭터 (능력치)



캐릭터의 능력치는 체력/체력 재생/힘/공격 속도/이동 속도 5가지로 만들었습니다.
능력치를 올릴 때는 골드가 필요하면 올릴 때마다 필요한 골드양이 증가합니다.
체력은 최대 체력을 증가시켜주고 체력 재생은 초당 체력 회복양을 증가시킵니다.
힘을 올리면 데미지를 퍼센트로 증가시키게 만들었고
공격 속도와 이동 속도는 애니메이터의 변수를 조절하여 구현했습니다.

1. 유니티 3D

④-5 캐릭터 (능력치)

```
public bool StatUp(int index)
{
    if (death)
        return false;
    switch (index)
    {
        case (int)StatIndex.HP:
            if (hpStat == 10)
                return false;
            if (hpStat * 10 + 10 > gold)
                return false;
            GoldChange(-hpStat * 10 - 10);
            hpStat++;
            HPChange(10);
            break;
        case (int)StatIndex.HPGEN:
            if (hpRegen == 10)
                return false;
            if (hpRegen * 10 + 10 > gold)
                return false;
            GoldChange(-hpRegen * 10 - 10);
            hpRegen++;
            break;
        case (int)StatIndex.STR:
            if (str == 10)
                return false;
            if (str * 10 + 10 > gold)
                return false;
            GoldChange(-str * 10 - 10);
            str++;
            break;
        case (int)StatIndex.ATKSPEED:
            if (atkspeed == 10)
                return false;
            if (atkspeed * 10 + 10 > gold)
                return false;
            GoldChange(-atkspeed * 10 - 10);
            atkspeed++;
            AtkSpeedChange();
            break;
        case (int)StatIndex.MOVSPEED:
            if (movspeed == 10)
                return false;
            if (movspeed * 10 + 10 > gold)
                return false;
            GoldChange(-movspeed * 10 - 10);
            movspeed++;
            MovSpeedChange();
            break;
    }
    return true;
}
```

-능력치 증가와 효과 적용

```
void Regen()
{
    if (HP == maxHP)
        return;
    regenTime += Time.deltaTime;
    if (regenTime > 1.0f)
    {
        regenTime = 0;
        HPChange(0.3f + hpRegen * 0.12f);
        //Debug.Log(HP);
    }
}
```

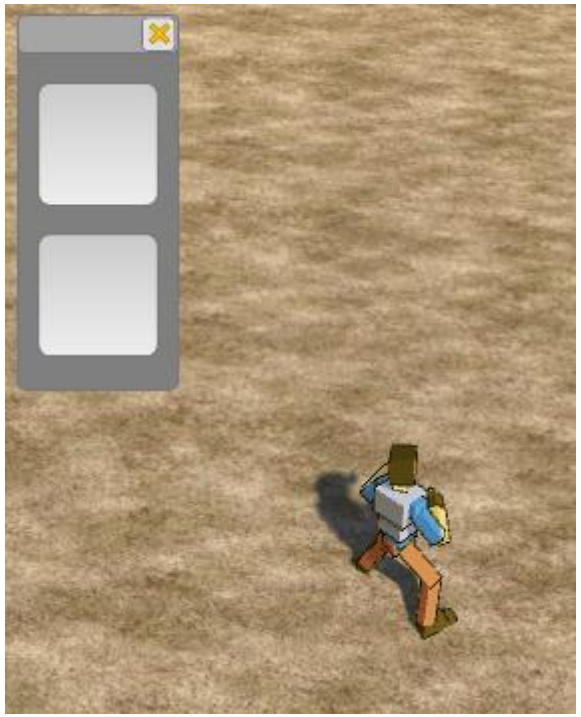
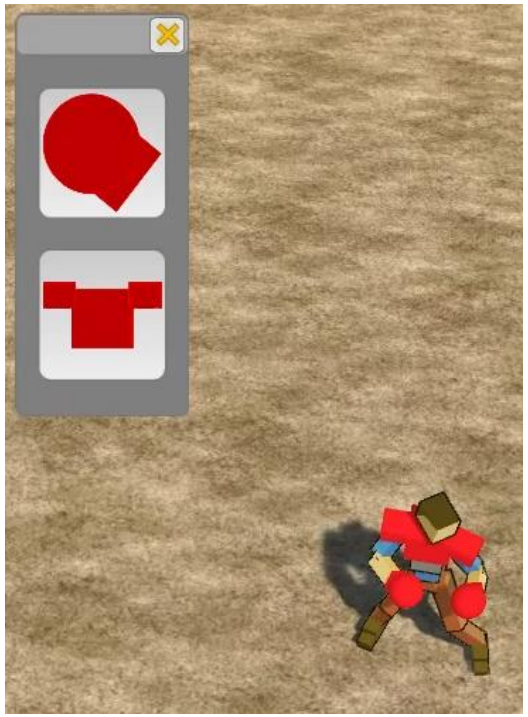
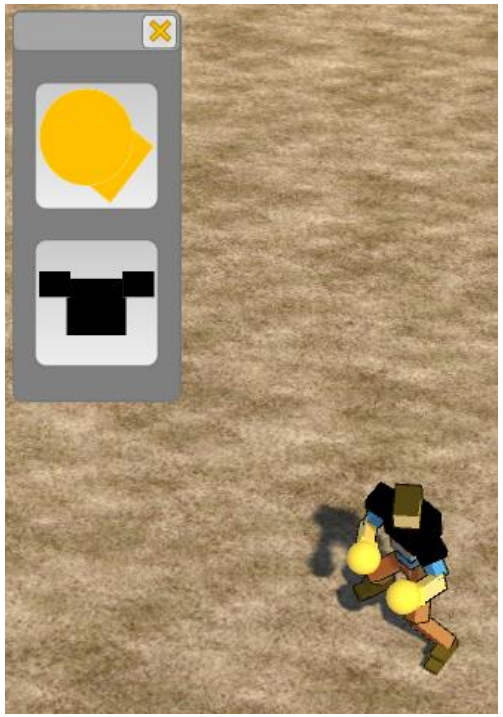
-체력 재생 구현

Motion	Unarmed-Attack-Kick-R1	
Speed	1	
Multiplier	atkspeed	☑ Parameter
Normalized Time		☐ Parameter
Mirror	☐	☐ Parameter
Cycle Offset	0	☐ Parameter
Foot IK	☐	
Write Defaults	☑	

Motion	Unarmed-Run-Forward	
Speed	1	
Multiplier	movspeed	☑ Parameter
Normalized Time		☐ Parameter
Mirror	☐	☐ Parameter
Cycle Offset	0	☐ Parameter
Foot IK	☐	
Write Defaults	☑	

1. 유니티 3D

④-6 캐릭터 (장비)

캐릭터의 장비는 무기와 갑옷 두가지로 만들었습니다.
장비는 세가지 등급이 있으며 각 등급별로 능력치가 다르게 만들었습니다.
장비를 장착하거나 해제할 때 장비 모델을 활성화, 비활성화하여
장비의 장착 유무를 나타내었습니다.

1. 유니티 3D

④-6 캐릭터 (장비)

```
private void EquipItem(int index)
{
    if (index == 0)
        return;
    if (index < (int)ItemIndex.blackarmor)
    {
        equipWeapon = index;
        EquipItemModel(0, true);
        if (index == (int)ItemIndex.blackweapon)
        {
            weaponMaterial.mainTexture = blackTexture;
            equipDmg = 1;
        }
        else if (index == (int)ItemIndex.redweapon)
        {
            weaponMaterial.mainTexture = redTexture;
            equipDmg = 2;
        }
        else if (index == (int)ItemIndex.goldweapon)
        {
            weaponMaterial.mainTexture = goldTexture;
            equipDmg = 4;
        }
    }
    else
    {
        equipArmor = index;
        EquipItemModel(1, true);
        if (index == (int)ItemIndex.blackarmor)
        {
            armorMaterial.mainTexture = blackTexture;
            equipHP = 10;
        }
        else if (index == (int)ItemIndex.redarmor)
        {
            armorMaterial.mainTexture = redTexture;
            equipHP = 20;
        }
        else if (index == (int)ItemIndex.goldarmor)
        {
            armorMaterial.mainTexture = goldTexture;
            equipHP = 40;
        }
        HPChange(0);
    }
    GameManager.instance.menu.EquipRefresh();
}
```

-장비 아이템 착용

```
public bool UnequipItem(int index)
{
    if (itemList.Count > 19)
        return false;
    switch (index)
    {
        case 0:
            if (equipWeapon == 0)
                return false;
            EquipItemModel(0, false);
            AddItem(equipWeapon);
            equipWeapon = 0;
            equipDmg = 0;
            break;
        case 1:
            if (equipArmor == 0)
                return false;
            EquipItemModel(1, false);
            AddItem(equipArmor);
            equipArmor = 0;
            equipHP = 0;
            HPChange(0);
            break;
    }
    return true;
}
```

-장비 아이템 해제

```
private void EquipItemModel(int index, bool equip)
{
    if (equip)
    {
        if (index == 0)
        {
            foreach (GameObject g in weapon)
            {
                g.SetActive(true);
            }
        }
        else
        {
            foreach (GameObject g in armor)
            {
                g.SetActive(true);
            }
        }
    }
    else
    {
        if (index == 0)
        {
            foreach (GameObject g in weapon)
            {
                g.SetActive(false);
            }
        }
        else
        {
            foreach (GameObject g in armor)
            {
                g.SetActive(false);
            }
        }
    }
}
```

-장비 모델 활성화/비활성화

1. 유니티 3D

④-7 캐릭터 (스킬)

캐릭터의 스킬은 슈퍼 아머/공격력 증가/이동 속도 증가 3가지로 만들었습니다.
스킬을 사용했을 때 코루틴을 실행시켜 지속시간과 쿨타임을 구현하였고
UI는 FillAmount를 조절하여 나타내었습니다.
슈퍼 아머는 지속 시간동안 피격시 경직이 발생하지 않게 하였고
캐릭터의 스프라이트를 수정하여 효과를 나타내었습니다.

1. 유니티 3D

④-7 캐릭터 (스킬)

```
private IEnumerator SuperArmorSkill()
{
    if (!saSkill)
        yield break;
    saSkill = false;
    superArmor = true;
    float timer = 0;
    playerMaterial.mainTexture = goldTexture;
    while (true)
    {
        saCool.fillAmount = 1 - timer / 50f;
        saActive.fillAmount = 1 - timer / 10f;

        if (superArmor && timer > 10f)
        {
            superArmor = false;
            playerMaterial.mainTexture = normalTexture;
        }
        if (timer > 50f)
        {
            saSkill = true;
            yield break;
        }
        yield return null;
        timer += Time.deltaTime;
    }
}
```

-슈퍼 아머 스킬 사용

```
private IEnumerator AttackUpSkill()
{
    if (!atkUpSkill)
        yield break;
    atkUpSkill = false;
    atkUp = true;
    float timer = 0;
    while (true)
    {
        atkUpCool.fillAmount = 1 - timer / 15f;
        atkUpActive.fillAmount = 1 - timer / 5f;

        if (atkUp && timer > 5f)
        {
            atkUp = false;
        }
        if (timer > 15f)
        {
            atkUpSkill = true;
            yield break;
        }
        yield return null;
        timer += Time.deltaTime;
    }
}
```

-공격력 증가 스킬 사용

```
private IEnumerator SpeedUpSkill()
{
    if (!spdUpSkill)
        yield break;
    spdUpSkill = false;
    spdUp = true;
    float timer = 0;
    MovSpeedChange();
    while (true)
    {
        spdUpCool.fillAmount = 1 - timer / 10f;
        spdUpActive.fillAmount = 1 - timer / 5f;

        if (spdUp && timer > 5f)
        {
            spdUp = false;
            MovSpeedChange();
        }
        if (timer > 10f)
        {
            spdUpSkill = true;
            yield break;
        }
        yield return null;
        timer += Time.deltaTime;
    }
}
```

-이동 속도 증가 스킬 사용

1. 유니티 3D

④-8 캐릭터 (적)

적 캐릭터는 일반 몬스터와 중간 보스 그리고 최종 보스가 있고 일반 몬스터와 중간 보스는
4종류로 만들었고 각각 능력치와 히트박스를 다르게 만들어 차이를 나타내었습니다.
적은 일정 거리안에 플레이어가 다가오면 따라가게 하였고 거리를 벗어나면 돌아가도록 만들었고
적이 따라올 때 NevMesh Agent를 사용하여 장애물을 돌아올 수 있게 하였습니다.
적의 공격 패턴은 코루틴을 사용하여 선딜레이와 후딜레이를 주었고
보스 몬스터는 피격 경직을 없애 일반 몬스터와 차이를 줬습니다.

1. 유니티 3D

④-8 캐릭터 (적)

```
public void Init(float dist, float delay, string boxRes, float maxhp, float damage)
{
    respawnPos = transform.position;
    transform.rotation = Quaternion.Euler(new Vector3(0f, Random.Range(0f, 360f), 0f));
    nav = GetComponent<NavMeshAgent>();
    player = GameManager.instance.GetPlayer();
    animator = GetComponent<Animator>();

    attackDistance = dist;
    attackDelay = delay;
    hitboxRes = boxRes;
    maxHp = maxhp;
    HP = maxHp;
    dmg = damage;
}
```

-적 캐릭터 능력치 초기화

```
private IEnumerator Attack()
{
    attackReady = false;
    yield return new WaitForSeconds(0.3f);
    if (death)
        yield break;
    if (canAttack && Vector3.Distance(player.transform.position, transform.position) < attackDistance)
    {
        canAttack = false;
        animator.SetTrigger("attack");
    }
    else
    {
        attackReady = true;
        yield break;
    }
    yield return new WaitForSeconds(0.15f);
    if (death)
        yield break;
    GameObject hitbox = null;
    hitbox = Instantiate(Resources.Load(hitboxRes), transform.position + transform.forward * 0.3f, new Quaternion(), transform) as GameObject;
    yield return new WaitForSeconds(0.1f);
    Destroy(hitbox);
    yield return new WaitForSeconds(attackDelay);
    if (death)
        yield break;
    if (hitStack == 0)
        canAttack = true;
    attackReady = true;
}
```

-적 캐릭터 공격 패턴 구현

```
void Update()
{
    if (death)
        return;
    if (!player.activeSelf)
    {
        BackToRespawn();
        return;
    }
    Vector3 target = new Vector3(player.transform.position.x, transform.position.y, player.transform.position.z);
    if (Vector3.Distance(respawnPos, player.transform.position) > 25)
    {
        BackToRespawn();
    }
    else if (Vector3.Distance(player.transform.position, transform.position) < attackDistance)
    {
        animator.SetBool("move", false);
        nav.isStopped = true;
        transform.LookAt(target);
        if (attackReady && canAttack)
        {
            StartCoroutine(Attack());
        }
    }
    else if (Vector3.Distance(player.transform.position, transform.position) < 8)
    {
        animator.SetBool("move", true);
        nav.isStopped = true;
        transform.LookAt(target);
    }
    else if (Vector3.Distance(player.transform.position, transform.position) < 15)
    {
        animator.SetBool("move", true);
        nav.isStopped = false;
        nav.SetDestination(player.transform.position);
    }
    else
    {
        BackToRespawn();
    }
}
```

-적 캐릭터의 이동 구현

1. 유니티 3D

④-9 캐릭터 (최종 보스)

최종 보스는 중간 보스의 열쇠를 모두 모아 숨겨진 숲으로 이동하면
전투를 시작하게 하였고 플레이어와 같은 콤보를 사용하게 만들었습니다.
최종 보스는 랜덤한 입력을 받아 한방향으로 콤보를 사용하도록 하였습니다.
체력이 절반이 되면 페이즈2로 넘어가고 공격 속도와 데미지가 증가하며
스프라이트를 붉은 색으로 바꿔 나타내었습니다.

1. 유니티 3D

④-9 캐릭터 (최종 보스)

```
private IEnumerator Attack()
{
    attacking = true;
    lastAttack = RandomAttack();
    yield return new WaitForSeconds(0.3f);
    if (death)
        yield break;
    lastAttack = RandomAttack();
    yield return new WaitForSeconds(0.3f);
    if (death)
        yield break;
    lastAttack = RandomAttack();
    yield return new WaitForSeconds(0.3f);
    if (death)
        yield break;
    lastAttack = RandomAttack();
    yield return new WaitForSeconds(0.3f);
    if (death)
        yield break;
    animator.SetBool("p", false);
    animator.SetBool("k", false);
    yield return new WaitForSeconds(attackDelay);
    if (death)
        yield break;
    attacking = false;
}
```

-최종 보스 공격 패턴 구현

```
private int RandomAttack()
{
    if (lastAttack == 0)
    {
        animator.SetBool("p", false);
    }
    else
    {
        animator.SetBool("k", false);
    }
    if (Random.Range(0, 2) == 0)
    {
        animator.SetBool("p", true);
        return 0;
    }
    else
    {
        animator.SetBool("k", true);
        return 1;
    }
}
```

-랜덤 입력 콤보 구현

```
void EnterPhase2()
{
    if (phase2)
        return;
    phase2 = true;
    bossMaterial.mainTexture = redTextrue;
    attackDelay = 0.5f;
    animator.SetFloat("movspeed", 1.1f);
    animator.SetFloat("atkspeed", 1.1f);
}
```

-페이즈 2 진입

1. 유니티 3D

⑤ NPC

NPC에 콜라이더를 만들어 일정 범위에 들어오면 UI를 띄우게 하였고
범위 밖으로 나가면 NPC를 통해 만들어진 모든 UI를 사라지게 하였습니다.
NPC를 통해 상점을 열거나 보스를 잡아 모아온 열쇠를 교환할 수 있습니다.
상점에서는 포션(소)/포션(대)/포탈/랜덤 장비를 살 수 있게 만들었으며
장비는 랜덤 클래스를 사용하여 등급별로 확률을 조절하였습니다.

1. 유니티 3D

⑤ NPC

```
public class NpcController : MonoBehaviour
{
    private void OnTriggerEnter(Collider other)
    {
        if (!other.name.Equals("Player"))
            return;
        GameManager.instance.NpcOpen();
    }

    private void OnTriggerExit(Collider other)
    {
        if (!other.name.Equals("Player"))
            return;
        GameManager.instance.NpcClose();
    }
}
```

-콜라이더 진입/탈출 UI ON/OFF

```
public bool TradeItem(int index)
{
    if (itemList.Count > 19)
        return false;
    switch (index)
    {
        case (int)TradeIndex.smallpotion:
            if (gold < 20)
                return false;
            GoldChange(-20);
            AddItem((int)ItemIndex.smallpotion);
            break;
        case (int)TradeIndex.bigpotion:
            if (gold < 60)
                return false;
            GoldChange(-60);
            AddItem((int)ItemIndex.bigpotion);
            break;
        case (int)TradeIndex.portal:
            if (gold < 100)
                return false;
            GoldChange(-100);
            AddItem((int)ItemIndex.portal);
            break;
        case (int)TradeIndex.randomequip:
            if (gold < 500)
                return false;
            GoldChange(-500);
            int rand1 = UnityEngine.Random.Range(0, 2);
            int rand2 = UnityEngine.Random.Range(0, 9);
            int equip = (int)ItemIndex.blackweapon + rand1 * 3 + rand2 / 5 + rand2 / 8;
            AddItem(equip);
            break;
    }
    return true;
}
```

-아이템 구매 구현

1. 유니티 3D

⑥ 저장

```
public void Save(List<int> stat, int gold, List<int> item, List<int> equipList)
{
    saveData.Init(stat, gold, item, equipList);

    JsonData saveJson = JsonMapper.ToJson(saveData);

    File.WriteAllText(Application.dataPath + "₩₩SaveData.json", saveJson.ToString());
}

public SaveData Load()
{
    FileInfo fi = new FileInfo(Application.dataPath + "₩₩SaveData.json");
    if (!fi.Exists)
        return new SaveData();
    string jsonString = File.ReadAllText(Application.dataPath + "₩₩SaveData.json");

    JsonData saveJson = JsonMapper.ToObject(jsonString);

    List<int> stat = new List<int>();
    stat.Add(int.Parse(saveJson["HP"].ToString()));
    stat.Add(int.Parse(saveJson["HPRegen"].ToString()));
    stat.Add(int.Parse(saveJson["Str"].ToString()));
    stat.Add(int.Parse(saveJson["AtkSpeed"].ToString()));
    stat.Add(int.Parse(saveJson["MovSpeed"].ToString()));

    int gold = int.Parse(saveJson["Gold"].ToString());

    List<int> item = new List<int>();
    int itemSize = int.Parse(saveJson["ItemSize"].ToString());
    for (int i = 0; i < itemSize; i++)
    {
        item.Add(int.Parse(saveJson["ItemList"][i].ToString()));
    }

    List<int> equip = new List<int>();
    equip.Add(int.Parse(saveJson["EquipWeapon"].ToString()));
    equip.Add(int.Parse(saveJson["EquipArmor"].ToString()));

    saveData.Init(stat, gold, item, equip);
    return saveData;
}
```

```
{
  "HP": 6,
  "HPRegen": 6,
  "Str": 6,
  "AtkSpeed": 10,
  "MovSpeed": 10,
  "Gold": 270,
  "ItemSize": 6,
  "ItemList": [ 12, 12, 14, 12, 13, 12 ],
  "EquipWeapon": 10,
  "EquipArmor": 13
}
```

Json을 사용하여 저장을 구현하였습니다.
저장을 하면 플레이어의 능력치/골드/장비/아이템을 Json파일로 만들고
로드하면 파일을 다시 읽어와 플레이어에 입력하였습니다.

1. 유니티 3D

⑦ 기타

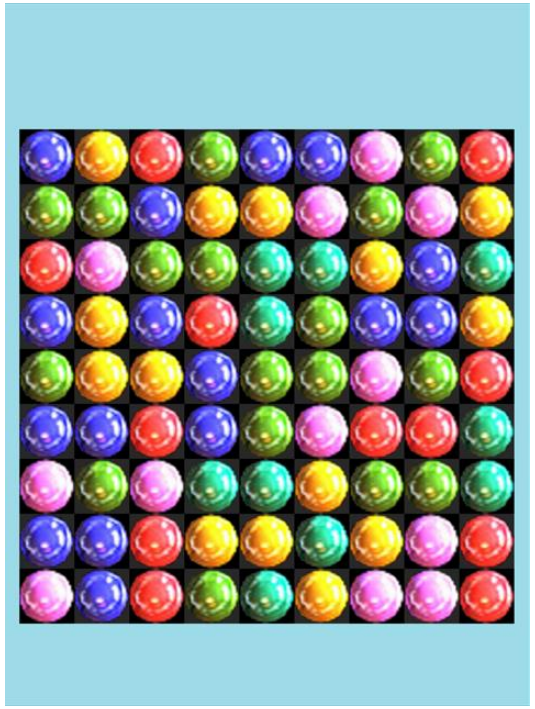
```cpp
public abstract class MonoSingleton<T> : MonoBehaviour where T : MonoSingleton<T>, new()
{
    private static T _instance = null;
    private static GameObject _gameObject = null;
    private static bool _isInitialize = false;

    protected virtual void OnInitialize() { }

    public static T instance
    {
        get
        {
            if (_instance == null)
            {
                //Scene에서 T타입의 GameObject를 찾는다.
                _instance = FindObjectOfType(typeof(T)) as T;

                //Scene내에 없다면 새로 GameObject를 생성
                if (_instance == null)
                {
                    _gameObject = new GameObject("__" + typeof(T).ToString());
                    _instance = _gameObject.AddComponent<T>();

                    //씬이 변경되어도 제거되지 않도록 설정.
                    DontDestroyOnLoad(_gameObject);

                    //GameObject를 새로 만들었음에도 _instance가 null이면 Error!!!
                    if (_instance == null)
                    {
                        Debug.LogError("Problem during the creation of " + typeof(T).ToString());
                    }
                }

                //Init.
                if (!_isInitialize)
                {
                    _instance.OnInitialize();
                    _isInitialize = true;
                }
            }

            return _instance;
        }
    }
```

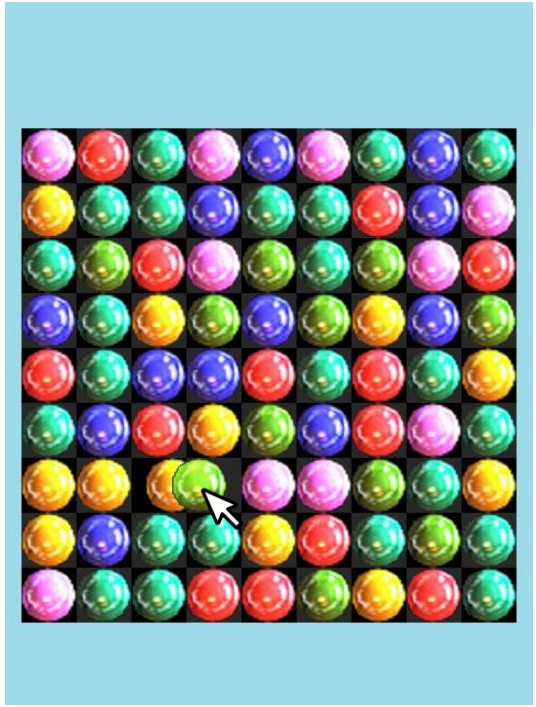
```cpp
enum ItemIndex
{
    bluekey=0,
    orangekey,
    pinkkey,
    redkey,
    goldkey,
    smallpotion,
    bigpotion,
    biggold,
    portal,
    blackweapon,
    redweapon,
    goldweapon,
    blackarmor,
    redarmor,
    goldarmor,
};

enum StatIndex
{
    HP=0,
    HPGEN,
    STR,
    ATKSPEED,
    MOVSPEED,
}
```

```cpp
enum PlaceIndex
{
    Town=0,
    East,
    West,
    South,
    North,
    Hidden,
}

enum LoadingIndex
{
    Title=0,
    Game,
    Boss,
}
```

GameManager는 싱글톤으로 구현하여 접근하기 쉽도록 만들었습니다.
변수의 값을 알아보기 쉽도록 인덱스를 만들어서 관리하였습니다.

2. 유니티 2D (NGUI)

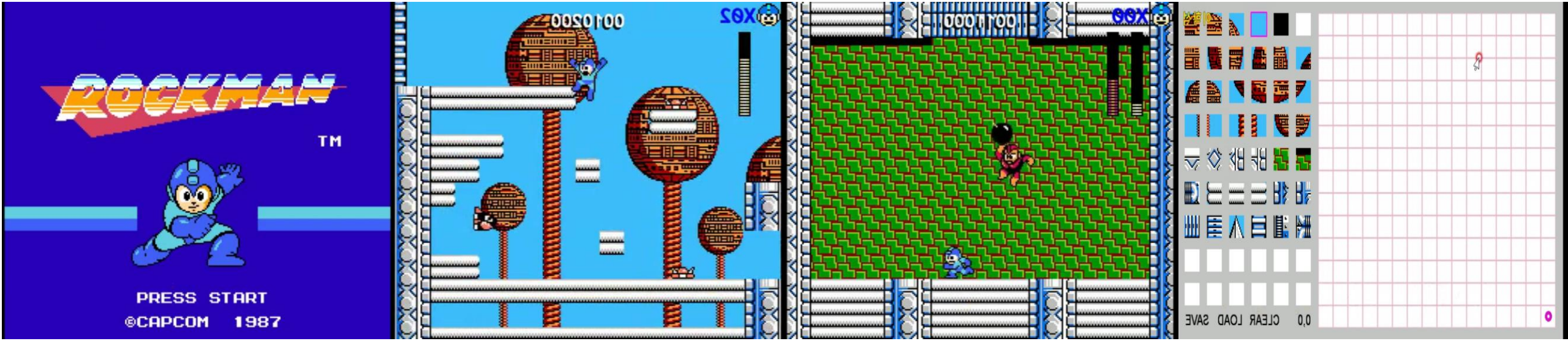




유니티의 NGUI를 사용하여 만든 간단한 퍼즐 게임입니다.
드래그를 통해 구슬을 이동시켜 3개를 맞추면 사라지고 위에서 구슬이 떨어집니다.
NGUI의 UI Event Trigger를 사용하여 On Press와 On Release의 함수를 만들어
구슬의 드래그를 구현하였습니다.

프로젝트 링크 : https://drive.google.com/open?id=112PHtoIodAtclKNJW6PX6HIcLMR_WDUK

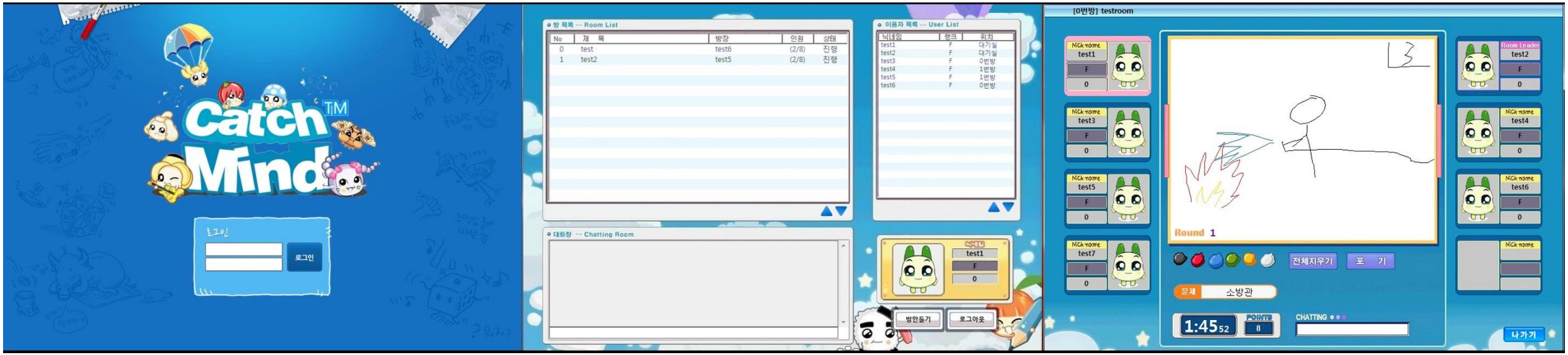
3. Cocos2d-X



Cocos2d-X를 사용하여 만든 포트폴리오입니다.
2D 스크롤 게임인 록맨1의 봄버맨 스테이지를 구현하였습니다.
맵 툴을 사용하여 스테이지를 만들었습니다.
타일과 스크롤을 사용하여 캐릭터의 조작을 구현하였습니다.

영상 링크 : https://drive.google.com/open?id=1t1v0Hcl_hZ6H5H_I-B7K3UMvd_um76D3
프로젝트 링크 : https://drive.google.com/open?id=1ck1RvRSg0QZlAkHwiUAf9FymoxASis8E

4. IOCP Server



IOCP 모델로 서버를 만들고 WinAPI를 사용하여 클라이언트를 만든 포트폴리오입니다.
그림으로 정답을 맞추는 캐치마인드를 구현하였습니다.
스트림 클래스와 패킷 클래스를 구현하여 데이터 통신을 하였고
로그인/로비/게임방을 만들어 각각 자료를 주고 받게 만들었습니다.

영상 링크 : https://drive.google.com/open?id=1GAhQDmGLiv5S4CUu9eR5prEYJ3r0WoIf
프로젝트 링크 : https://drive.google.com/open?id=1vm9ppkybY7EoiMW5vqyvi-IM0IXrJODQ