



톡톡 과일팡!

4조 - 2D 클릭어 게임 구현

Contents



01 팀 구성 및 역할

02 게임 시연

03 핵심 기능

04 트러블 슈팅

05 협업 방식 및 마무리



1. 팀 구성 및 역할

1. 팀 구성 및 역할



이유진(팀장)

1 - 게임 매니저

+ 발표, 피피티



김여진

2 - 클릭 이벤트

+ UI, 로고



김지환

3 - 적 및 스테이지

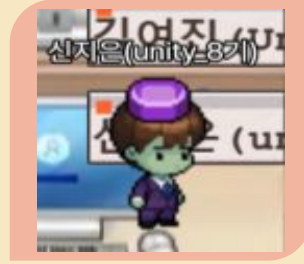
+애니메이션



천지훈

4 - 플레이어 능력치 및 업그레이드

+ 스타트썬, 자동공격 추가



신지은

5 - 무기 및 업그레이드

+ 상점구현

2. 게임 시연

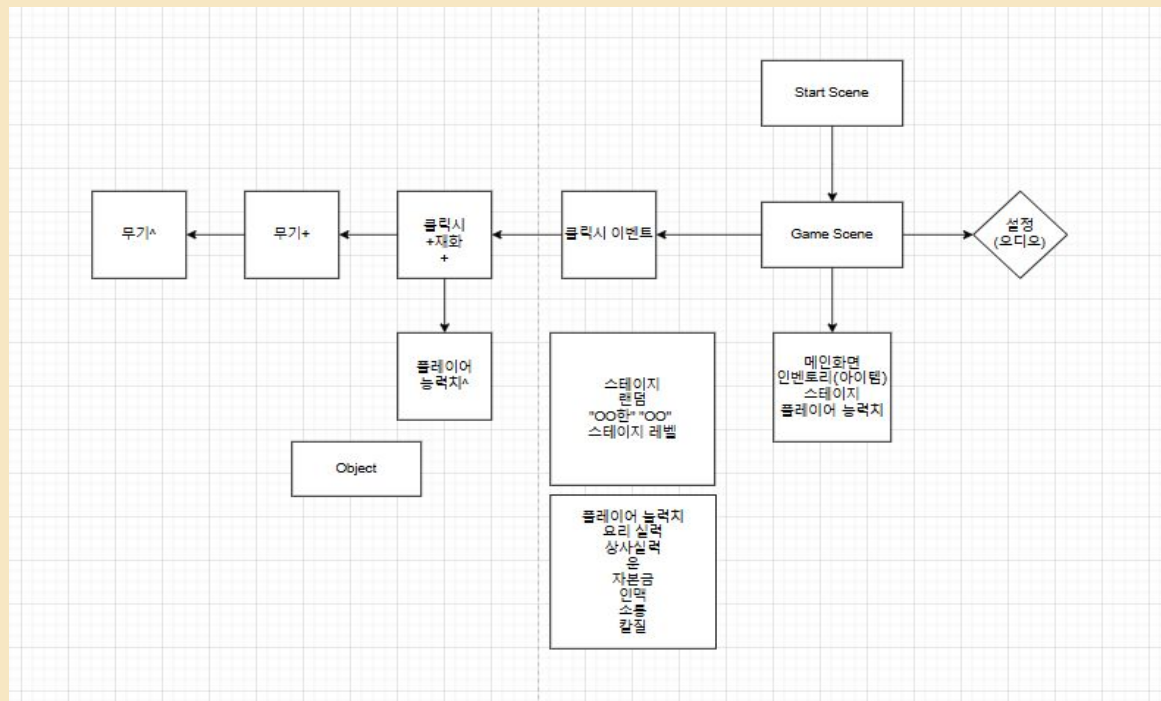
2. 게임 시연



3. 핵심 기능

3. 핵심 기능

와이어 프레임



적 및 스테이지

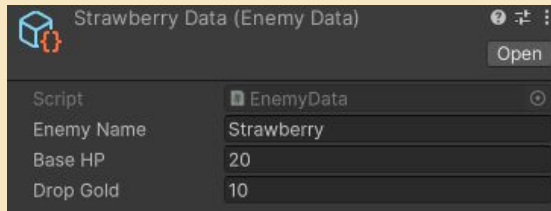
```
Unity 스크립트(자산 참조 12개) | 참조 2개
public class StageManager : MonoBehaviour
{
    public int currentStage = 1;
    public int currentround = 1;
    [SerializeField] TMP_Text stage;
    [SerializeField] TMP_Text round;
    참조 1개
    public void StageCount()
    {
        if(currentround < 10)
        {
            currentround++;
            round.text = currentround + "/" + 10;
        }
        else
        {
            ChangeStage();
            currentround = 1;
            round.text = currentround + "/" + 10;
        }
    }

    참조 1개
    public void ChangeStage()
    {
        currentStage++;
        stage.text = "스테이지" + currentStage;
    }
}
```

StageManager

화면에
나오는스테이지와
라운드를 계산

Data 를 받아 체력과 드랍되는
돈 설정



Strawberry Data (Enemy Data)	
Script	EnemyData
Enemy Name	Strawberry
Base HP	20
Drop Gold	10

ScriptableObjects EnemyData 와
Enemy 생성할 적 프리팹에 넣어 값을 설정

```
[CreateAssetMenu(fileName = "NewEnemyData", menuName = "ScriptableObjects/EnemyData")]
Unity 스크립트 | 참조 2개
public class EnemyData : ScriptableObject
{
    public string enemyName;
    public float baseHP;
    public int dropGold;
}
```

적 및 스테이지

Enemy 스크립트

Die()

적이 죽을 때 돈을 지급하는 메서드와 라운드를 넘기는 메서드

```
참조 2개
public void Initialize()
{
    // 현재 = baseHP x 현재 스테이지 +4 + baseHP x 현재 라운드
    maxHP = enemyData.baseHP * stageManager.currentStage +4 + enemyData.baseHP * stageManager.currentRound;
    currentHP = maxHP;

    //골드 보상 트랩골드 x 현재 스테이지 + 3
    dropMoney = enemyData.dropGold * stageManager.currentStage +3;
    UpdateHPBar();
}

참조 2개
public void TakeDamage(float dmg)
{
    currentHP -= dmg;
    if (currentHP < 0) currentHP = 0;

    UpdateHPBar();

    if (currentHP <= 0)
    {
        Die();
    }
}

참조 2개
void UpdateHPBar()
{
    if (hpFillImage != null)
        hpFillImage.fillAmount = currentHP / maxHP;
}

참조 1개
void Die()
{
    //골드 지급로직
    PlayerStats.Instance.playerData.AddGold(dropMoney);
    Debug.Log($"적 처치! {dropMoney} 골드 획득. 현재 골드: {PlayerStats.Instance.playerData.gold}");

    // 골드 UI 업데이트 추가
    GoldManager goldManager = GameObject.FindObjectOfType<GoldManager>();
    if (goldManager != null)
    {
        goldManager.UpdateGoldUI();
    }
    else
    {
        Debug.LogError("GoldManager를 찾을 수 없습니다 UI 업데이트 실패");
    }

    stageManager.StageCount(); // 죽으면 스테이지 라운드 증가
    spawner.OnEnemyKilled();
    Destroy(gameObject);
}
```

적 및 스테이지

EnemySpawner 일부

리스트를 생성해서 셔플하고
리스트에 따라 적을 화면에 생성

```
참조 15개
void AddPrefabs(GameObject prefab, int count) //스테이지에 널을 적을 매개변수로 두임
{
    for (int i = 0; i < count; i++)
    {
        spawnList.Add(prefab);
    }
}

참조 2개
public void SpawnNext() //리스트의 다음적을 생성하는 메서드
{
    if (spawnList.Count == 0)
    {
        SetupStage();
    }

    if (currentEnemy != null)
        Destroy(currentEnemy);

    GameObject next = spawnList[0];
    spawnList.RemoveAt(0);

    currentEnemy = Instantiate(next, spawnParent.position, Quaternion.identity, spawnParent);
    Enemy enemyScript = currentEnemy.GetComponent<Enemy>();
    enemyScript.spawner = this;
    enemyScript.stageManager = stageManager;
    enemyScript.Initialize();
}

참조 1개
public void OnEnemyKilled() //적이 죽었을 때 호출할 메서드
{
    SpawnNext();
}

참조 1개
void Shuffle(List<GameObject> list) //배치된 리스트에서 순서를 무작위로 배치하는 기능
{
    for (int i = 0; i < list.Count; i++)
    {
        int rand = Random.Range(i, list.Count);
        GameObject temp = list[i];
        list[i] = list[rand];
        list[rand] = temp;
    }
}
```

클릭 기능

ClickDamage()

클릭을 했을 때
공격 데미지 값을 받아
태그가 **Enemy** 인 오브젝트에
데미지를 주는 메서드

GrantGold()

클릭당 골드를 획득하는 메서드

```
public class ClickToDamage : MonoBehaviour
{
    private GoldManager goldManager;

    public int clickGoldReward = 1; // 클릭시 +1골드
    // Unity 메시지 참조 0개
    private void Start()
    {
        goldManager = FindObjectOfType<GoldManager>(); // GoldManager 찾기
        if (goldManager == null)
        {
            Debug.LogError("GoldManager를 못찾았습니다. 씬에 GoldManager 오브젝트가 있는지 확인하세요.");
        }
        else
        {
            Debug.Log("GoldManager 정상적으로 할당됨.");
        }
    }
    // 참조 0개
    public void ClickDamage()
    {
        GrantGold(clickGoldReward);

        GameObject enemy = GameObject.FindWithTag("Enemy");
        if (enemy != null)
        {
            Enemy enemyScript = enemy.GetComponent<Enemy>();
            if (enemyScript != null)
            {
                // PlayerData9의 CalculateFinalDamage 반영
                float damage = PlayerStats.Instance.playerData.CalculateFinalDamage();
                enemyScript.TakeDamage(damage);
                Debug.Log($"클릭 {damage} 데미지");
            }
        }
    }
    // 참조 1개
    private void GrantGold(int gold)
    {
        if (goldManager != null)
        {
            goldManager.GetGold(gold); // GoldManager를 통해 골드 추가
            Debug.Log($"클릭으로 {gold} 골드 획득!");
        }
    }
}
```

자동 공격 시스템

StartAutoAttack()

자동 공격 시작

```
참조 2개
public void StartAutoAttack()
{
    if (!isAutoAttacking && PlayerStats.Instance.autoAttackSpeed > 0)
    {
        isAutoAttacking = true;
        StartCoroutine(AutoAttackRoutine());
    }
}
```

AutoAttackRoutine()

일정 간격마다 공격 및 골드 지급

```
참조 2개
private IEnumerator AutoAttackRoutine()
{
    while (PlayerStats.Instance.autoAttackSpeed > 0)
    {
        float attackInterval = 5f / PlayerStats.Instance.autoAttackSpeed; // 속도 반영
        yield return new WaitForSeconds(attackInterval);

        // 현재 baseDamage 값 즉시 반영 (업그레이드 적용)
        int damage = Mathf.RoundToInt(PlayerStats.Instance.baseDamage);
        int gold = Mathf.RoundToInt(10 * PlayerStats.Instance.goldBonus); // 골드 획득량 반영

        ApplyDamage();
        GrantGold(gold);
    }
}
```

자동 공격 시스템

ApplyDamage()

적에게 데미지 적용

```
private void ApplyDamage()  
{  
    GameObject enemy = GameObject.FindWithTag("Enemy");  
    if (enemy != null)  
    {  
        Enemy enemyScript = enemy.GetComponent<Enemy>();  
        if (enemyScript != null)  
        {  
            // PlayerStats의 baseDamage를 반영  
            float damage = PlayerStats.Instance.baseDamage;  
            enemyScript.TakeDamage(damage);  
            Debug.Log($"자동 공격! {damage} 데미지");  
        }  
    }  
}
```

UpdateAutoAttackSpeed()

자동 공격 속도 변경 시 즉시 반영

```
public void UpdateAutoAttackSpeed()  
{  
    if (isAutoAttacking)  
    {  
        StopCoroutine(AutoAttackRoutine());  
        StartAutoAttack();  
    }  
}
```

자동 공격 시스템

GrantGold(gold)

적에게 데미지 적용

```
참조 1개
private void GrantGold(int gold)
{
    if (PlayerStats.Instance != null)
    {
        PlayerStats.Instance.playerData.AddGold(gold);
        GoldManager goldManager = GameObject.FindObjectOfType<GoldManager>();
        //UpgradeManager.Instance.UpdateGoldUI();
        if (goldManager != null)
        {
            goldManager.UpdateGoldUI();
        }
        else
        {
            Debug.LogError("GoldManager를 찾을 수 없습니다! UI 업데이트 실패");
        }

        Debug.Log($"자동 공격 {gold} 골드 획득! 현재 골드: {PlayerStats.Instance.playerData.gold}");
    }
    else
    {
        Debug.LogError("PlayerStats.Instance가 null. 씬에 PlayerStats 오브젝트가 있는지 확인하기.");
    }
}
참조 1개
```

4. 트러블 슈팅

4. 트러블 슈팅

업그레이드된 능력치를 올바르게 반영하는 문제

문제 상황

처음에는 업그레이드를 적용해도 자동 공격 속도, 데미지 증가, 골드 획득량 등의 값이 즉시 반영되지 않는 문제가 발생.

업그레이드 버튼을 눌러 능력치를 올려도, 실제 전투에서는 여전히 이전 값으로 계산되는 현상이 있었음.

원인 분석

1. `PlayerStats`에서 업그레이드된 값을 즉시 반영하지 못하고 있었음.
2. `AutoAttackSystem`은 기존 공격 속도를 유지한 채 실행 중이어서, 새로운 공격 속도가 반영되지 않음.
3. 골드 획득량 증가도 `AutoAttackSystem`에서 기존 값으로 계산되면서 효과가 늦게 적용됨.

해결 방법

`ApplyUpgrade()`에서 능력치를 변경한 후 즉시 **UI를 업데이트**하도록 수정.

`GrantGold()` 로직을 수정하여 업그레이드된 골드 보너스를 적용하도록 보장.

4. 트러블 슈팅

오늘 작업한 코드가 날아가는 문제 발생.

문제 상황

각자 작업한 내용을 dev에서 각자의 브랜치와 머지하는 과정에서
팀원들이 당일날 작업한 코드가 모두 날아가는 문제가 발생.

해결 방법

커밋 로그를 확인 한후 팀원의 작업이 반영된 마지막 정상 커밋을 찾고,
해당 커밋을 기준으로 새로운 dev브랜치 생성해 거기서부터 팀원들도 브랜치를 한개씩 새로 만들어 진행.

4. 트러블 슈팅

무기의 능력치가 초기화 되지않는 문제.

문제 상황

새 게임을 시작한 후, 무기의 능력치가 이전에 강화한 상태로 남아있는 현상이 발생.

원인 분석

게임의 재시작 시, 무기 강화 레벨이나 능력치가 초기화되지 않아서 발생한 문제로 추측됨.

해결 방법

`weaponStats`클래스에 무기 능력치와 강화레벨을 기본값으로 리셋하는 `ResetToDefault()` 메서드를 추가하고 스타트씬에서 호출하여 해결.

```
// 무기 능력치를 기본값으로 복원하는 메서드  
참조 1개  
public void ResetToDefault()  
{  
    baseDamage = statUpgrades[0].damageIncrease;  
    baseSpeed = statUpgrades[0].speedIncrease;  
    baseCritChance = statUpgrades[0].critChanceIncrease;  
    baseCritDamage = statUpgrades[0].critDamageIncrease;  
  
    upgradeLevel = 0; // 강화 레벨 초기화  
}
```

5. 협업 방식 및 마무리

5. 협업 방식 및 마무리

Figma와 같은 실시간 협업 가능한 디자인 툴을 활용하여 게임 기획을 효율적으로 진행.
매일 오전/오후에 정기 회의를 통해 팀원들과 작업 진행 상황을 공유하고,
발생한 문제들은 함께 소통하여 해결.

사조참지 #픽셀 #과일

1. 이미지란

• 디지털 환경에서 진행하면 스테이지, 보우 제작, 일그래이드 전환, 스포 캠페인 등등의 유닛 데이터(인물화)
• 지출 스포츠 상품(일그래이드 및 장악 무기 등의 항목을 종합해 지출 스포츠당 산출)

• 자료화 데이터의 획득과 사용 관리, 화면에 대응중인 자료 처리, 상호작용에 필요한 처리가 부족한 경우 클로 네딩하기

• BOM용역과 같은 인터페이스를 보면 해당 화면에서 서로 다른 BOM을 통해 출력되는 데이터를 추출, 출력을 확인하기

2. 특징

• 자료, 분리(오기)한정(스케치, 레퍼, 일그래이드, 장비 변환 등 사용자 정의 진행 상황 저장, 로드)

(연습장)

2. 작업현

스케치판

공작기

공작기

공작기

공작기

공작기

공작기

공작기

공작기

공작기

공작기

공작기

공작기

공작기

공작기

공작기

공작기

공작기

공작기

공작기

공작기

공작기

공작기

공작기

공작기

공작기

공작기

공작기

공작기

4. 컨디션

ScriptableObject 로 일그래이드 구성 만들기
능력치를 일그래이드로 인한 수치 증가량 테이블을 만들고, 현재 일그래이드에 따른 능력치를 산출

- 지능이 데이터
- 자동 공격
- 체력 획득할 증가

start 인 만들기

일그래이드로 전달 및 그에 따라 현재 능력치를 화면에서 표시

컨디션

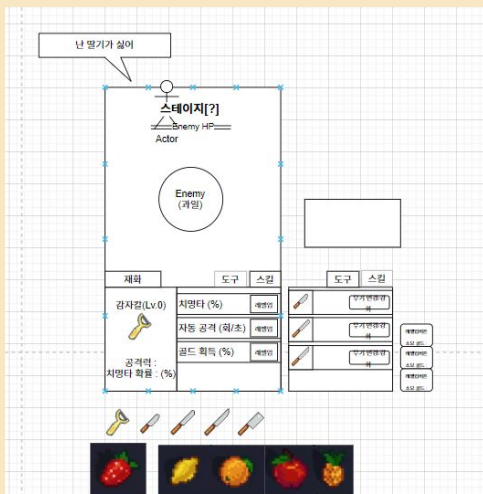
5. 스토리

스토리 및 스토리보드

- 이야기 줄거리(스토리를 만들기, 이해할 때 흐름)
- 무기 능력이 데이터 테이블 작성
- 무기 공격 & 방위
- 캐릭터 스토리 무기 능력치를 선택한다

스토리

10/10/2020



1	평가 10
2	제안 15 평가 5 (대중연결)
3	제안 3 평가 5 오연자 4 (대중연결)
4	제안 2 평가 2 오연자 3 서가 3 (대중연결)
5	제안 2 평가 2 오연자 2 서가 2 피연대물 2 (대중연결)

1	말개*10
2	개단*10
3	오래사*10
4	사개*10
5	마단*10

THANK YOU!

Q & A

TEAM 사조참치(4조)

팀장 이유진

팀원 김여진 김지환 신지은 천지훈

