# HOW GAME DEVS SEE

OVERWATCH

DESTINY

TOM CLANCY'S RAINBOW SIX | SIEGE

# INPUT REBINDING SYSTEM

```
public static void StartRebinding(RebindData rebindData,
                  System.Action<bool> updateUICallback,
                  System.Action onComplete)
{
    if (rebindData == null || rebindData.actionReference?.action == null)
        return;

    var action = rebindData.actionReference.action;
    action.Disable();

    // Tell UI we're starting
    updateUICallback?.Invoke(true);

    action.PerformInteractiveRebinding(rebindData.bindingIndex)
          .OnMatchWaitForAnother(0.1f)
          .OnComplete(operation =>
          {
              action.Enable();
              operation.Dispose();
              SaveBindingOverrides(rebindData.actionReference.asset);
              updateUICallback?.Invoke(false);
              onComplete?.Invoke();
          })
          .Start();
}
```

# HOW GAME DEVS SEE


## COMBO ATTACKS

```
namespace Outscal.GameSchool.Combo
{
    public enum ComboName
    {
        SuperKick,
        SuperPunch,
        SuperSlap,
    }
}
```

```
namespace Outscal.GameSchool.Combo
{
    [CreateAssetMenu(fileName = "NewCombo",
                     menuName = "Combat/Combo")]
    public class ComboSO : ScriptableObject
    {
        public ComboName comboName;
        public List<KeyCode> sequence;
    }
}
```

**Write desginer friendly code to make great games**

```
private void CheckCombo()
{
    foreach (var combo in comboSettings.combos)
    {
        if (currentInputs.Count ⩾ combo.sequence.Count)
        {
            var lastInputs = currentInputs.Skip(currentInputs.Count - combo.sequence.Count).ToList();

            if (lastInputs.SequenceEqual(combo.sequence))
            {
                ExecuteCombo(combo.comboName);
                currentInputs.Clear();
                return;
            }
        }
    }
}
```

# HOW GAME DEVS SEE


## Projectile SPAWNING

```
public class ProjectilePool
{
    private void Initialize()
    {/*Code to inialize projectile's object pool*/}

    private void GetProjectile()
    {/*Code to get projeticle from pool*/}

    // Return a projectile to the pool
    public void ReturnToPool(Projectile projectile)
    {
        ProjectileType type = projectile.ProjectileType;

        projectile.gameObject.SetActive(false);
        projectile.transform.SetParent(poolContainer);

        inactiveProjectiles[type].Enqueue(projectile);
    }

    // Create a new projectile
    private Projectile CreateNewProjectile(ProjectileType type)
    {
        ProjectileData data = projectileDataMap[type];
        Projectile newProjectile =
            Instantiate(data.Prefab, poolContainer);
        newProjectile.gameObject.SetActive(false);
        newProjectile.SetPool(this);
        inactiveProjectiles[type].Enqueue(newProjectile);

        return newProjectile;
    }
}
```

# HOW GAME DEVS SEE

# ANGRY BIRDS BIRD SPAWNING

```
namespace Outscal.GameSchool.Birds
{
    public enum BirdType
    {
        Red,
        Blue,
        Yellow,
        Bomb,
        Silver,
        Eagle,
        // Add more bird types as needed
    }
}
```

```
namespace Outscal.GameSchool.Birds
{
    public interface IBirdAbility
    {
        void ActivateAbility(Bird bird);
    }
}
```

## Create 1 Scriptable Object Script

## & Have Countless Birds

```
using UnityEngine;

namespace Outscal.GameSchool.Birds
{
    [CreateAssetMenu(fileName = "NewBirdData", menuName = "ScriptableObjects/BirdData")]
    public class BirdScriptableObject : ScriptableObject
    {
        [Header("Bird Identification")]
        public BirdType BirdType;
        public GameObject BirdPrefab;

        [Header("Flight Parameters")]
        [Min(0f)] public float BaseLaunchSpeed = 15f;
        public float Mass = 1f;

        [Header("Damage & Collision")]
        public int BaseDamage = 20;
        public float ImpactRadius = 0.5f;

        [Header("Ability & Behavior")]
        public ScriptableObject AbilityObject; // Must implement IBirdAbility
    }
}
```

# Code for: DOUBLE JUMP

```
using UnityEngine;

public class DoubleJumpController : MonoBehaviour
{
    // Core settings
    [SerializeField] private float jumpForce = 5f;
    [SerializeField] private int maxJumpCount = 2;
    private int currentJumpCount;
    private bool isGrounded;
    private Rigidbody rb;

    private void Start()
    {
        rb = GetComponent<Rigidbody>();
        rb.useGravity = false;
    }

    private void Update()
    {
        CheckGround();
        if (Input.GetKeyDown(KeyCode.Space) && currentJumpCount < maxJumpCount)
        {
            rb.velocity = new Vector3(rb.velocity.x, 0f, rb.velocity.z);
            rb.AddForce(Vector3.up * jumpForce, ForceMode.Impulse);
            currentJumpCount++;
        }
    }

    private void FixedUpdate() => rb.AddForce(Physics.gravity * 2.5f, ForceMode.Acceleration);

    private void CheckGround()
    {
        if (Physics.Raycast(transform.position, Vector3.down, 0.2f))
        {
            isGrounded = true;
            currentJumpCount = 0;
        }
        else
            isGrounded = false;
    }
}
```

# This SKILL TREE is a CRIME!

```
public class SkillTree : MonoBehaviour
{
    public List<string> skills = new List<string>();
    public int skillPoints = 5;

    public void UnlockSkill(string skill)
    {
        if (skill == "Fireball")
        {
            if (!skills.Contains("BasicAttack"))
            {
                Debug.Log("Can't unlock Fireball without BasicAttack");
                return;
            }
        }
        else if (skill == "IceBlast")
        {
            if (!skills.Contains("Fireball") || !skills.Contains("Shield"))
            {
                Debug.Log("Can't unlock IceBlast without Fireball and Shield");
                return;
            }
        }
        else if (skill == "Shield")
        {
            if (!skills.Contains("BasicAttack"))
            {
                Debug.Log("Can't unlock Shield without BasicAttack");
                return;
            }
        }

        if (skillPoints > 0)
        {
            skillPoints--;
            skills.Add(skill);
            Debug.Log(skill + " unlocked!");
        }
        else
        {
            Debug.Log("Not enough points!");
        }
    }
}
```

# BAD CODE

## Check the GOOD CODE!

# You're Loading Levels Wrong

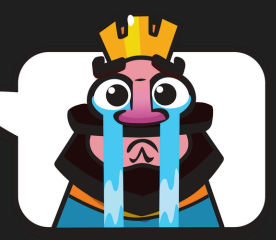
# Here's a better Way!

# NO BORING LOADING SCREENS

```
public enum SceneType { MainMenu, Forest, Village, Dungeon, Castle, Desert }

public class LevelStreamingManager : MonoBehaviour
{
    public static LevelStreamingManager Instance { get; private set; }

    private HashSet<SceneType> loadedScenes = new HashSet<SceneType>();

    private void Awake()
    {
        if (Instance == null) Instance = this;
        else Destroy(gameObject);
    }

    public void LoadScene(SceneType scene)
    {
        if (!loadedScenes.Contains(scene))
        {
            StartCoroutine(LoadSceneAsync(scene));
        }
    }

    public void UnloadScene(SceneType scene)
    {
        if (loadedScenes.Contains(scene))
        {
            StartCoroutine(UnloadSceneAsync(scene));
        }
    }

    private IEnumerator LoadSceneAsync(SceneType scene)
    {
        AsyncOperation asyncLoad = SceneManager.LoadSceneAsync(scene.ToString(), LoadSceneMode.Additive);
        while (!asyncLoad.isDone)
        {
            yield return null;
        }
        loadedScenes.Add(scene);
    }

    private IEnumerator UnloadSceneAsync(SceneType scene)
    {
        AsyncOperation asyncUnload = SceneManager.UnloadSceneAsync(scene.ToString());
        while (!asyncUnload.isDone)
        {
            yield return null;
        }
        loadedScenes.Remove(scene);
    }
}
```


SAVE CODE IN THE DESCRIPTION

# HOW GAME DEV SEE




# TROOPS

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

namespace Outscal.GameSchool.Troops
{
    [CreateAssetMenu(fileName = "TroopScriptableObject",
    menuName = "ScriptableObjects/TroopScriptableObject")]
    public class TroopScriptableObject : ScriptableObject
    {
        public TroopType Type;
        public GameObject Prefab;
        public int ElixirCost;
        public float SpawnTime;
        public int Hitpoints;
        public int Damage;
        public float AttackSpeed;
        public float MovementSpeed;
        public TargetType TargetType;
        public AttackMethod AttackType;
        public float Range;
        public bool IsFlying;
        public List<SpecialAbility> Abilities;
    }
}
```

## Create 1 Scriptable Object Script Create Troops

```cs
namespace Outscal.GameSchool.Troops
{
    public enum TroopType
    {
        Knight,
        Archer,
        Giant,
        // Add other troop types as nee
    }
}
```

```cs
namespace Outscal.GameSchool.Troops
{
    public enum TargetType
    {
        Ground,
        Air,
        Buildings,
        GroundAndAir,
        GroundAndBuildings,
        AirAndBuildings,
        All
    }
}
```

# OBSERVER

## PROGRAMMING PATTERN

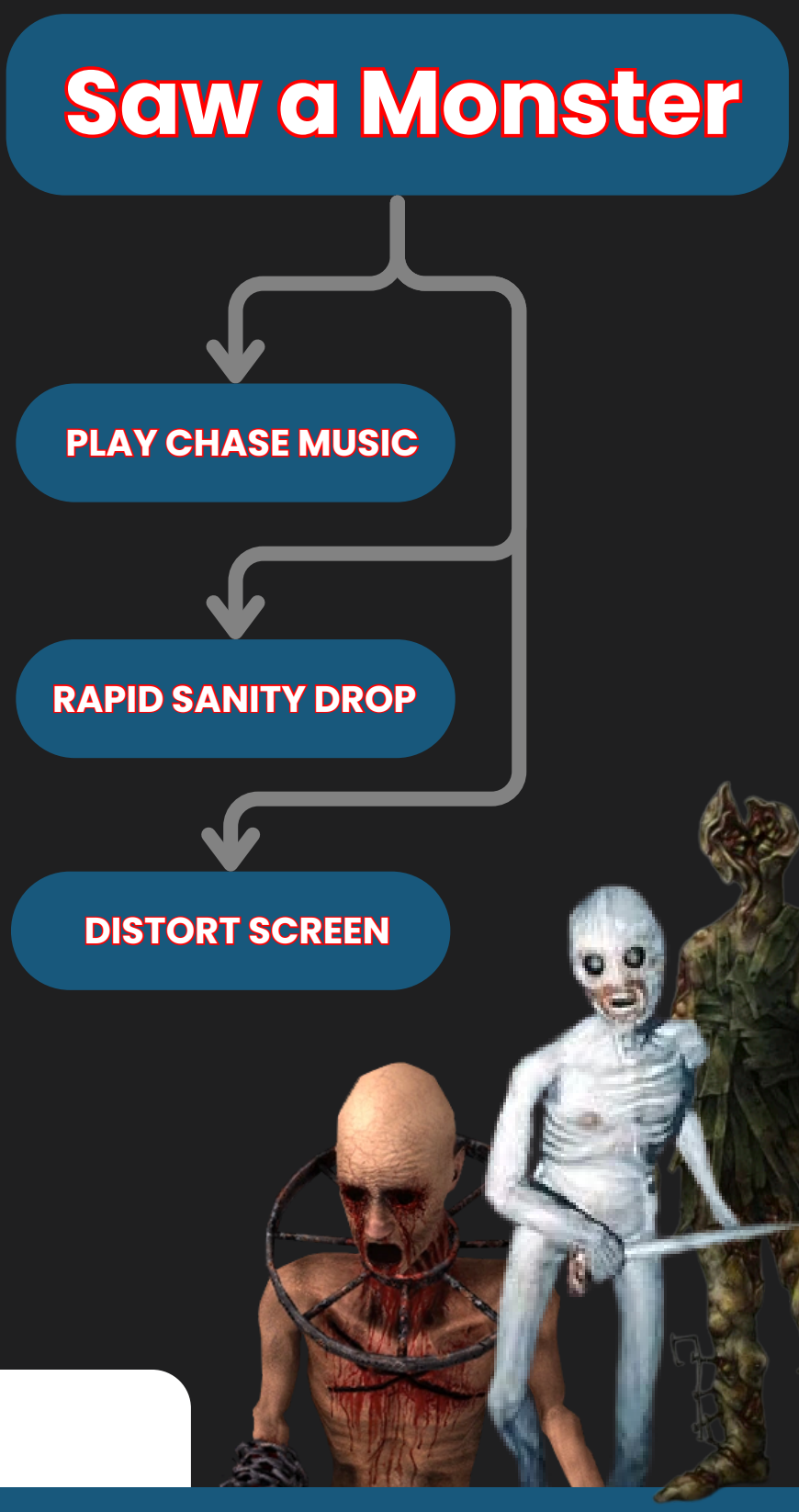
### GAME PROGRAMMER GUIDE

```cs
using System;

public class EventController
{
    public Action baseEvent;
    public void AddListener(Action listener) => baseEvent += listener;
    public void RemoveListener(Action listener) => baseEvent -= listener;
    public void InvokeEvent() => baseEvent?.Invoke();
}

public class EventController<T>
{
    public Action<T> baseEvent;
    public void AddListener(Action<T> listener) => baseEvent += listener;
    public void RemoveListener(Action<T> listener) => baseEvent -= listener;
    public void InvokeEvent(T type) => baseEvent?.Invoke(type);
}
```

```cs
public class EventService
{
    private static EventService instance;
    public static EventService Instance
    {
        get
        {
            if (instance == null)
            {
                instance = new EventService();
            }
            return instance;
        }
    }

    public EventController OnLightSwitchToggled { get; private set; }

    public EventService()
    {
        OnLightSwitchToggled = new EventController();
    }
}
```

Use it to build games like


# 1 Event
# Many Effects




**Sample Code in Description**