PATRONES DE DISEÑO

PATRÓN UTILIZADO: SINGLETON
USO (SCRIPT): GAME MANAGER

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

SE INSTANCIA UNA VEZ

```
private void Awake()
{
    if (Instance != null && Instance != this)
    {
        Destroy(gameObject);
        return;
    }
    Instance = this;
```

SI DETECTA OTRA INSTACNCIA LA DESTRUVE

¿PARA QUÉ ESTÁ SIENDO USADO?

GARANTIZAR QUE EXISTA SOLO UNA INSTANCIA GLOBAL DEL GAMEMANAGER. PERMITIR ACCEDER DESDE CUALQUIER SCRIPT A COSAS COMO:

SCORE
INVOCACIÓN DEL BOSS STORM
SPAWNERS
SISTEMA DE ALIADOS
HUD
ESTADOS DEL JUGADOR

¿CUMPLE CON SOLID? NO

LO UTILIZO PARA HACER VARIAS COSAS

PATRÓN UTILIZADO: OBJECT POOL
USO (SCRIPT): POOL BULLET PLAYER

CREA UNA CANTIDAD FIJA DE OBJETOS (POOLSIZE) AL INICIO.
MANTIENE UNA LISTA DE OBJETOS DISPONIBLES:

```
private readonly Queue<GameObject> availableBullets = new Queue<GameObject>();
```

MANTIENE UN CONJUNTO DE OBJETOS EN USO:

```
private readonly HashSet<GameObject> inUse = new HashSet<GameObject>();
```

NO DESTRUYE LAS BALAS, SINO QUE LAS REACTIVA Y REUTILIZA:

```
private GameObject CreateBullet()
{
    var bullet = Instantiate(bulletPrefab, poolParent);
    bullet.SetActive(false);
    availableBullets.Enqueue(bullet);
    return bullet;
}
```

DEVUELVE LA BULLET

```
bullet.SetActive(false);
bullet.transform.SetParent(poolParent, false);
availableBullets.Enqueue(bullet);
```

¿PARA QUÉ ESTÁ SIENDO USADO?

EVITAR INSTANCIAR/DESTRUIR BALAS EN TIEMPO REAL (MUY COSTOSO).
EVITAR PICOS DE CPU Y GC (GARBAGE COLLECTOR).
MANTENER FPS ESTABLES EN SHOOTERS.

¿CUMPLE CON SOLID? SI

PATRÓN UTILIZADO: SINGLETON Y OBJECT POOL.
USO (SCRIPT): LASERPOOLBOSS

EL PATRÓN SINGLETON ASEGURA QUE EXISTA UNA ÚNICA INSTANCIA DEL POOL EN TODO EL JUEGO, ACCESIBLE DESDE CUALQUIER OTRO SCRIPT MEDIANTE UNA REFERENCIA ESTÁTICA. (UNA INSTANCIAD DE BOSS)

¿PARA QUÉ ESTÁ SIENDO USADO?

POR SU PARTE, EL PATRÓN OBJECT POOL GESTIONA LA CREACIÓN Y REUTILIZACIÓN DE OBJETOS TIPO LÁSER, EVITANDO EL USO CONTINUO DE INSTANTIATE() Y DESTROY()

¿CUMPLE CON SOLID? NO

```
private void Prewarm()
{
    for (int i = 0; i < initialSize; i++)
    {
        var go = Instantiate(pooledLaser, transform);
        go.SetActive(false);
        lasers.Add(go);
    }
}
```

INICIA EL POOL

```
public GameObject GetLaser()
{
    for (int i = 0; i < lasers.Count; i++)
    {
        if (!lasers[i].activeInHierarchy)
            return lasers[i];
    }
}
```

PIDE EL OBJETO

```
public void ReturnToPool(GameObject laser)
{
    if (laser == null) return;
    if (!lasers.Contains(laser))
    {
        laser.transform.SetParent(transform);
        lasers.Add(laser);
    }
    laser.SetActive(false);
}
```

DEVUELVE AL POOL

LA CLASE MANTIENE UNA LISTA DE OBJETOS DISPONIBLES, LOS ACTIVA AL SOLICITARLOS Y LOS DESACTIVA AL DEVOLVERLOS

```
public void ClearPool()
{
    for (int i = 0; i < lasers.Count; i++)
    {
        if (lasers[i] != null && lasers[i].activeInHierarchy)
            lasers[i].SetActive(false);
    }
}
```

LIMPIA EL POOL

PATRÓN UTILIZADO: FACTORY (IMPLICITO)
USO (SCRIPT): PHASATRON SPAWNER

```
private void TrySpawn()
{
    int score = GameManager.Instance != null ? GameManager.Instance.TotalScore : 0;
    if (score < minScoreThreshold || score >= maxScoreThreshold) return;
    if (currentEnemy >= maxEnemies) return;
    if (enemyPrefab == null) return;

    float spawnPosX = Random.Range(spawnXRange.x, spawnXRange.y);
    Vector2 spawnPosition = new(spawnPosX, spawnY);

    GameObject newEnemy = Instantiate(enemyPrefab, spawnPosition, Quaternion.identity);

    if (newEnemy.TryGetComponent<Phasatron>(out var enemyScript))
    {
        enemyScript.moveSpots = moveSpots;
    }

    currentEnemy++;
}
```

¿PARA QUÉ ESTÁ SIENDO USADO?

CREAR ENEMIGOS DINÁMICAMENTE SEGÚN EL SCORE ES UNA ACCIÓN DEPENDIENTE DEL GAMEPLAY. CENTRALIZAR LA CREACIÓN EVITA DUPLICAR LÓGICA EN OTROS SCRIPTS.

¿CUMPLE CON SOLID? SI, EXCEPTO LA DEPENDECIA

PATRÓN UTILIZADO: FACADE
USO (SCRIPT): LEVELPROGRESSTORE

```cs
using UnityEngine;

public static class LevelProgressStore
{
    private const string KeyLastCompletedName = "LastCompletedLevelName";

    public static void SetLastCompletedName(string sceneName)
    {
        PlayerPrefs.SetString(KeyLastCompletedName, sceneName);
        PlayerPrefs.Save();
        Debug.Log($"[LevelProgressStore] Guardado último nivel completado: {sceneName}");
    }

    public static string GetLastCompletedName()
    {
        return PlayerPrefs.GetString(KeyLastCompletedName, string.Empty);
    }

    public static void Clear()
    {
        PlayerPrefs.DeleteKey(KeyLastCompletedName);
    }
}
```

¿PARA QUÉ ESTÁ SIENDO USADO?

LA CLASE ENCAPSULA PLAYERPREFS OFRECIENDO MÉTODOS SIMPLES Y CLAROS PARA UN PROPÓSITO ESPECÍFICO DESACOPLA LÓGICA DE GUARDADO DEL RESTO DEL JUEGO.

ES FACADE POR:

POR QUÉ:
OCULTA EL SISTEMA COMPLEJO "REAL" (PLAYERPREFS).
SIMPLIFICA SU USO MEDIANTE UN CONJUNTO MÍNIMO Y COHERENTE DE MÉTODOS
LA CLASE SE PRESENTA COMO INTERFAZ SIMPLE Y EXTERNA, MIENTRAS TODA LA LÓGICA INTERNA (KEY, SAVE, DELETE) QUEDA ENCAPSULADA.

¿CUMPLE CON SOLID?

SI, EXCEPTO DEPENDENCIA, PORQUE USA PLAYER PREFS

PATRÓN UTILIZADO: STATE
USO (SCRIPT): PAUSEMANAGER

```
private void Update()
{
    if (Input.GetKeyDown(KeyCode.Escape))
    {
        if (ControlsBox.activeSelf)
        {
            CloseControls();
            return;
        }

        if (onPause)
            Continue();
        else
            PauseGame();
    }
}
```

¿PARA QUÉ ESTÁ SIENDO USADO?

GESTIONAR EL MENU Y EL ESTADO DE PAUSA CUANDO EL PLAYER LO DESEE

EL SCRIPT GESTIONA DOS ESTADOS

MAQUINA DE ESTADOS BASICA

¿CUMPLE CON SOLID?

NO CUMPLE CON SOLID, DEBIDO A QUE TIENE VARIAS DEPENDENCIAS Y PARA AGREGAR UNA NUEVA OPCION SE DEBE MODIFICAR TODO EL SCRIPT

PATRÓN UTILIZADO: SINGLETON
USO (SCRIPT): GLOBALMUSICMANAGGER

```
private void Awake()
{
    if (instance == null)
    {
        instance = this;
        DontDestroyOnLoad(gameObject);
    }
    else
    {
        Destroy(gameObject);
    }
}
```

- SI NO EXISTE UNA INSTANCIA, SE GUARDA ESTA (THIS) Y SE MARCA CON DONTDESTROYONLOAD() PARA PERSISTIR ENTRE ESCENAS.

SI YA EXISTE OTRA, SE DESTRUYE EL NUEVO OBJETO DUPLICADO (EVITA MÚLTIPLES REPRODUCTORES DE MÚSICA SIMULTÁNEOS).

¿CUMPLE CON SOLID?

NO CUMPLE CON EL SEGUNDO NI EL QUINTO, DEBIDO A QUE DEBE MODIFICARSE PARTE DEL SCRIPT PARA QUE AGREGUE MUSICA A OTRO BOSS Y TAMBIEN DEPENDE DEL SCENE MANAGER

¿PARA QUÉ ESTÁ SIENDO USADO?

PARA QUE SOLAMENTE EXISTA UNA INSTANCIA DE SONIDO EN EL JUEGO, Y SI DETECTA OTRA QUE LA BORRE

```
private void PlayMusicForCurrentScene()
{
    int currentSceneIndex = SceneManager.GetActiveScene().buildIndex;
    AudioClip newClip = null;

    if (currentSceneIndex == 2) newClip = level1Music;
    else if (currentSceneIndex == 3) newClip = level2Music;
    else if (currentSceneIndex == 4) newClip = level3Music;

    if (newClip != null && audioSource.clip != newClip)
    {
        audioSource.clip = newClip;
        audioSource.Play();
        isMusicPaused = false;
    }
    else if (newClip == null)
    {
        audioSource.Stop();
        audioSource.clip = null;
    }
}

public void PauseMusic()
{
    if (audioSource.isPlaying)
    {
        audioSource.Pause();
        isMusicPaused = true;
    }
}

public void ResumeMusic()
{
    if (isMusicPaused)
    {
        audioSource.Play();
        isMusicPaused = false;
    }
}
```

PATRÓN UTILIZADO: POOL
USO (SCRIPT): OBJECTPOOL (BOSS)

CREA UNA COLECCION DE OBJETOS

```
void Start()
{
    for (int i = 0; i < poolSize; i++)
    {
        var obj1 = Instantiate(objectPrefab);
        var obj2 = Instantiate(objectPrefab2);

        obj1.SetActive(false);
        obj2.SetActive(false);

        obj1.transform.SetParent(transform);
        obj2.transform.SetParent(transform);

        objectPool.Enqueue(obj1);
        objectPool2.Enqueue(obj2);
    }
}
```

ENTREGA BAJO DEMANDA

```
public GameObject GetObject(bool useSecondPrefab = false)
{
    if (useSecondPrefab)
    {
        while (objectPool2.Count > 0)
        {
            var obj = objectPool2.Dequeue();
            if (obj)
            {
                obj.transform.SetParent(null, false);
                obj.SetActive(true);
                return obj;
            }
        }

        var created = Instantiate(objectPrefab2);
        created.SetActive(true);
        return created;
    }
```

```
public void ReturnObject(GameObject obj, bool useSecondPrefab = false)
{
    if (!obj) return; // Unity null check
```

UNA VEZ USADO LO DEVUELVE AL POOL

¿CUMPLE CON SOLID?

SI, EXCEPTO CON PRINCIPIO ABIERTO/CERRADO; SE DEBE MODIFICAR MUCHA PARTE DEL SCRIPT SI SE DESEA AGREGAR OTRO OBJETO

PATRÓN UTILIZADO: COMMAND (IMPLICITO)

USO (SCRIPT): MENU

```
public void StartGame() => LoadSceneAfterDelay(level1Scene);
public void LevelSelection() => LoadSceneAfterDelay(levelSelectScene);
public void Level1() => LoadSceneAfterDelay(level1Scene);
public void Level2() => LoadSceneAfterDelay(level2Scene);
public void Level3() => LoadSceneAfterDelay(level3Scene);
```

FUNCIONA COMO UN COMANDO EJECUTADO POR BOTONES DE UI. CADA MÉTODO "DISPARA" UNA ACCIÓN SIMPLE LO QUE HACE ES CARGAR ESCENA TRAS UN DELAY.

CADA ACCIÓN ES UN COMANDO SEPARADO
LA UI INVOCA COMANDOS
CADA COMANDO EJECUTA UNA SOLA ACCIÓN (CARGAR ESCENA O SALIR)
ES UN COMMAND RUDIMENTARIO SIN INTERFACES NI SEPARACIÓN.

```
public void NextLevel()
{
    string lastCompleted = LevelProgressStore.GetLastCompletedName();
    Debug.Log($"[Menu] NextLevel desde: {lastCompleted}");

    if (lastCompleted == level1Scene) LoadSceneAfterDelay(level2Scene);
    else if (lastCompleted == level2Scene) LoadSceneAfterDelay(level3Scene);
    else
        LoadSceneAfterDelay(levelSelectScene);
}

private void LoadSceneAfterDelay(string sceneName)
{
    StartCoroutine(LoadAfterDelay(sceneName));
}

private IEnumerator LoadAfterDelay(string sceneName)
{
    yield return new WaitForSeconds(delayTime);
    SceneManager.LoadScene(sceneName);
}

private IEnumerator LoadAfterDelay(int buildIndex)
{
    yield return new WaitForSeconds(delayTime);
    SceneManager.LoadScene(buildIndex);
}

private IEnumerator QuitAfterDelay()
{
    yield return new WaitForSeconds(delayTime);
    Application.Quit();
}
```

¿CUMPLE CON SOLID?

NO, EL SCRIPT TIENE VARIOS PROBLEMAS, PRIMERO Y PRICIPAL DEPENDE DE OTROS MODULOS, NO UTILIZA METODOS DE ABSTRACCION COMO ISCENELOADER, IPROGRESSSERVICE O IEXITSERVICE

PATRÓN UTILIZADO: OBSERVER (IMPLICITO)
USO (SCRIPT): HUD

```
void Update()
{
    var gm = GameManager.Instance; // puede ser null si aún no existe/persistió

    if (Score != null && gm != null)
    {
        Score.text = gm.TotalScore.ToString();
    }
```

¿PARA QUÉ ESTÁ SIENDO USADO?

GESTIONAR EL MENU Y EL ESTADO DE PAUSA CUANDO EL PLAYER LO DESEE

EL HUD OBSERVA EL ESTADO DEL GAME MANAGER CADA FRAME Y REFLEJA LOS CAMBIOS EN OTRO SISTEMA (GAME MANAGER)

```
public void UpdateScore(int totalScore)
{
    if (Score) Score.text = totalScore.ToString();
}

public void UpdateLivesSlider(float lives)
{
    if (livesSlider) livesSlider.value = lives;
}
```

METODOS ESPERANDO A SER LLAMADOS DESDE AFUERA

¿CUMPLE CON SOLID?

NO, SOLO CUMPLE CON SUBTITUCION E INTERFACE. SI CAMBIO EL SISTEMA DE SCORE, SE ROMPE

PATRÓN UTILIZADO: FACTORY (LEVE)
USO (SCRIPT): ENEMYFLYTRHOUGHTSPAWNER

```
GameObject enemy = Instantiate(enemyFlyThroughPrefab, spawnPos, Quaternion.identity);

if (enemy.TryGetComponent<EnemyFlyThrough>(out var fly))
{
    fly.SetMoveDirection(moveDir);
}

activeEnemies++;
```

¿PARA QUÉ ESTÁ SIENDO USADO?

PARA CREAR UN SPAWNER DE ENEMIGOS QUE RESPONDA SEGUN CIERTAS REGLAS

FABRICA DE ENEMIGOS SEGUN REGLAS (CONDICION DE SCORE, POSICION Y LIMITE)

```
if (enemyFlyThroughPrefab == null) return;

int score = GameManager.Instance != null ? GameManager.Instance.TotalScore : 0;
if (score < minScoreThreshold || score >= maxScoreThreshold) return;
```

SI CUMPLE CON LOS REQUISITOS, ENTONCES CREA UNA INSTANCIA

¿CUMPLE CON SOLID? NO CUMPLE.

PATRÓN UTILIZADO: FACTORY (LEVE)
USO (SCRIPT): PHASATRON3SPAWNER

```
GameObject newEnemy = Instantiate(enemyPrefab, spawnPosition, Quaternion.identity);
Phasatron3 enemyScript = newEnemy.GetComponent<Phasatron3>();
```

```
private void TrySpawn()
{
    int score = GameManager.Instance != null ? GameManager.Instance.TotalScore : 0;
    if (score < minScoreThreshold || score >= maxScoreThreshold) return;
    if (currentEnemy >= maxEnemies) return;
    if (enemyPrefab == null) return;

    float spawnPosX = Random.Range(spawnXRange.x, spawnXRange.y);
    Vector2 spawnPosition = new(spawnPosX, spawnY);
    SI CUMPLE CON LOS REQUISITOS, ENTONCES CREA UNA INSTANCIA
    GameObject newEnemy = Instantiate(enemyPrefab, spawnPosition, Quaternion.identity);

    if (newEnemy.TryGetComponent<Phasatron>(out var enemyScript))
    {
        enemyScript.moveSpots = moveSpots;
    }

    currentEnemy++;
}
```

¿PARA QUÉ ESTÁ SIENDO USADO?

PARA CREAR UN SPAWNER DE ENEMIGOS QUE RESPONDA SEGUN CIERTAS REGLAS

- EL SPAWNER SE ENCARGA DE DECIDIR CUÁNDO Y CÓMO CREAR LOS ENEMIGOS.
- LA LÓGICA DE INICIALIZACIÓN DEL ENEMIGO (POSICIÓN, PUNTOS DE MOVIMIENTO, LÍMITE DE CANTIDAD, ETC.) ESTÁ CENTRALIZADA DENTRO DE LA CLASE.

EL RESTO DEL JUEGO SIMPLEMENTE PUEDE ACTIVAR O DETENER EL SPAWNER, SIN PREOCUPARSE DE LOS DETALLES DE CREACIÓN.

AUNQUE NO SE UTILIZA HERENCIA NI INTERFACES ABSTRACTAS, LA FUNCIÓN DEL SPAWNER COINCIDE CON LA RESPONSABILIDAD TÍPICA DE UNA FÁBRICA CONCRETA DENTRO DEL PATRÓN.

¿CUMPLE CON SOLID?

NO CUMPLE.

PATRÓN UTILIZADO: POOL
USO (SCRIPT): BULLETPOOLBOSS

```
public static BulletPoolBoss bulletPoolInstanse;

[SerializeField] private GameObject pooledBullet;
[SerializeField] private bool notEnoughBulletsPool = true;

private readonly List<GameObject> bullets = new();

private void Awake()
{
    bulletPoolInstanse = this;
}
```

CREA LA POOL

```
public GameObject GetBullet()
{
    for (int i = 0; i < bullets.Count; i++)
    {
        if (!bullets[i].activeInHierarchy)
            return bullets[i];
    }
```

PIDE LA BALA

```
public void ClearPool()
{
    for (int i = 0; i < bullets.Count; i++)
    {
        if (bullets[i] != null && bullets[i].activeInHierarchy)
            bullets[i].SetActive(false);
    }
}
```

LIMPIA LA POOL

¿PARA QUÉ ESTÁ SIENDO USADO?

REUSAR INSTANCIAS EN LUGAR DE HACER INSTANTIATE() CADA VEZ. GUARDAR OBJETOS EN UNA LISTA. ACTIVAR/DESACTIVAR SEGÚN NECESIDAD.

¿CUMPLE CON SOLID?

SI, PERO SOLO PARA UN TIPO DE BULLET.

PATRÓN UTILIZADO: MEMENTO
USO (SCRIPT): PLAYERMEMENTO

DECLARA LA VARIABLE Y LLAMA A LA LOGICA

```
private PlayerMemento checkpoint;

public void SaveCheckpoint()
{
    checkpoint = SavePlayerState();
    Debug.Log("CHECKPOINT GUARDADO");
}

public void LoadCheckpoint()
{
    if (checkpoint != null)
    {
        RestorePlayerState(checkpoint);
        Debug.Log("CHECKPOINT RESTAURADO");
    }
}
```

```
private void Update()
{
    if (Input.GetKeyDown(KeyCode.Alpha1))
    {
        ToggleImmortality();
    }
    if (Input.GetKeyDown(KeyCode.Alpha2))
    {
        ActivateCheat();
    }


    if (Input.GetKeyDown(KeyCode.C))
    {
        SaveCheckpoint();
    }


    if (Input.GetKeyDown(KeyCode.R))
    {
        LoadCheckpoint();
    }
}
```

```
public PlayerMemento SavePlayerState()
{
    return new PlayerMemento(
        player.transform.position,
        lives,
        totalScore
    );
}

public void RestorePlayerState(PlayerMemento memento)
{
    player.transform.position = memento.position;
    lives = memento.lives;
    totalScore = memento.score;

    hud.UpdateLivesSlider(lives);
    hud.UpdateScore(totalScore);
}
```

LOGICA DE DATOS GUARDADOS Y DATOS RESTAURADOS AL USAR

¿PARA QUÉ ESTÁ SIENDO USADO?

CREA UN CHECKPOINT (VIDA, POSICION Y SCORE) MANUAL CADA VEZ QUE TOCO LA TECLA R Y VOY AL LUGAR DEL CHECKPOINT CON LA TECLA C

CONDICIONES PARA USAR EL CHECKPOINT MANUAL

¿CUMPLE CON SOLID?

SI, PODEMOS AÑADIR DATOS A GUARDAR MODIFICANDO POCOS DATOS EN EL SCRIPT

```
using UnityEngine;

// Memento: clase de datos (NO MonoBehaviour)
public class PlayerMemento
{
    public Vector3 position;
    public float lives;
    public int score;

    public PlayerMemento(Vector3 position, float lives, int score)
    {
        this.position = position;
        this.lives = lives;
        this.score = score;
    }
```

PATRÓN UTILIZADO: EVENT QUEUE

USO (SCRIPT): EVENT QUEUE/ GAME EVENT / PLAY SOUND

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

public class EventQueue : MonoBehaviour
{
    public static EventQueue Instance;

    private Queue<GameEvent> eventQueue = new Queue<GameEvent>();

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

    public void Enqueue(GameEvent gameEvent)
    {
        if (gameEvent == null) return;
        eventQueue.Enqueue(gameEvent);
    }

    private void Update()
    {

        if (eventQueue.Count > 0)
        {
            GameEvent ev = eventQueue.Dequeue();
            ev.Execute();
        }
    }
}
```

SE CREA COLA DE EVENTOS

```cpp
int previousScore = totalScore;
totalScore += ScoreToAdd;
hud.UpdateScore(totalScore);

int prevHundreds = previousScore / 100;
int currHundreds = totalScore / 100;
if (currHundreds > prevHundreds)
{

    if (EventQueue.Instance != null && score100Clip != null)
    {
        EventQueue.Instance.Enqueue(new PlaySoundEvent(score100Clip));
    }
    else
    {

        PlayOneShot(score100Clip);
    }
}
```

CADA VEZ QUE SUPERA 100 DE SCORE, SE ENCOLA EL EVENTO

```cpp
public void PlayOneShot(AudioClip clip, float volume = 1f)
{
    if (clip == null) return;
    if (audioSource == null)
    {

        audioSource = GetComponent<AudioSource>();
        if (audioSource == null)
        {
            Debug.LogWarning("No AudioSource en GameManager ni se encontró uno.");
            return;
        }
    }
    audioSource.PlayOneShot(clip, volume);
}
```

SE REPRODUCE SONIDO

```cpp
public AudioClip score100Clip;
public AudioClip deathSound;
private AudioSource audioSource;
```

SE AGREGA UN SONIDO EN EL MANAGER

¿PARA QUÉ ESTÁ SIENDO USADO?

REPRODUCE UN SONIDO DE FEEDBACK AL REALIZAR 100 PTS, SUENA CADA VEZ QUE SE SUMAN 100 PTS

¿CUMPLE CON SOLID? SI.

PATRONES NO UTILIZADOS

PATRÓN NO UTILIZADO: ABSTRACT FACTORY

JUSTIFICACION

EL PATRÓN ABSTRACT FACTORY ES UN PATRÓN DE DISEÑO CREACIONAL QUE CREA FAMILIAS DE OBJETOS RELACIONADOS O DEPENDIENTES SIN NECESIDAD DE ESPECIFICAR SUS CLASES CONCRETAS.

EJEMPO Y COMO PODRÍA IMPLEMENTARLO EN MI JUEGO

SI TUVIERA UN SISTEMA DE PERSONALIZACION DE NAVES, Y PONGO NAVES DE TIPO; PESADAS, LIGERAS E INTERMEDIAS. Y LUEGO DEBERIA CREAR PARTES DE LA NAVE QUE CORRESPONDAN A CADA TIPO DE NAVE, ARMAS PESADAS, ALETAS PESADAS, ETC. ENTONCES PODRIA IMPLEMENTAR UNA FABRICA ABSTRACTA PARA CADA FAMILIA DE NAVES, POR EJEMPLO LUEGO SI EL PLAYER ELIJE " NAVE PESADA" EL JUEGO INSTACIARÁ ESA NAVE PESADA Y LE PIDE A LA FABRICA LAS PARTES DE LA NAVE PESADA.

NO SE IMPLEMENTÓ DEBIDO A MI FALTA DE ORGANIZACIÓN CON EL PROYECTO Y LA COMPLEJIDAD DE LA IDEA SOBRE EL TIEMPO QUE ME QUEDABA PARA TERMINAR EL PROYECTO.

PATRÓN NO UTILIZADO: FLYWEIGHT

JUSTIFICACION

EL PATRÓN DE DISEÑO FLYWEIGHT ES UN PATRÓN ESTRUCTURAL QUE OPTIMIZA EL USO DE MEMORIA AL COMPARTIR PARTES DE DATOS COMUNES ENTRE MÚLTIPLES OBJETOS EN LUGAR DE DUPLICAR LA INFORMACIÓN EN CADA UNO

EJEMPO Y COMO PODRÍA IMPLEMENTARLO EN MI JUEGO

CADA ENEMIGO QUE TENGO, COMPARTE CARACTERISTICAS, VIDA, DAÑO, VELOCIDAD, ETC. PODRÍA HABER UTILIZADO ESTE PATRÓN PARA CREAR UNA CLASE ENEMIGOS Y LUEGO VERTIR TODOS LOS DATOS COMUNES DE MIS ENEMIGOS. LUEGO, CADA VEZ QUE NECESITE UN NUEVO ENEMIGO, CREARE UN OBJETO Y HARÉ UNA REFERENCIA A LA INSTANCIA DE ENEMIGOS.

NO SE IMPLEMENTÓ DEBIDO A QUE UTILIZO UN METODO DE SPANWERS EN MI VIDEOJUEGO Y SE ME COMPLEJIZO UN POCO (NO QUERIA MODIFICAR MUCHO MI SISTEMA FUNCIONAL) , MÁS ALLÁ DE ESO. ES POSIBLE.

PATRÓN NO UTILIZADO: STRATEGY

JUSTIFICACION

EL PATRÓN DE DISEÑO STRATEGY ES UN PATRÓN DE COMPORTAMIENTO QUE ENCAPSULA UNA FAMILIA DE ALGORITMOS EN CLASES SEPARADAS Y LAS HACE INTERCAMBIABLES.

EJEMPO Y COMO PODRÍA IMPLEMENTARLO EN MI JUEGO

SI EN MI PLAYER TUVIERA DISTINTOS TIPOS DE PROVECTILES, QUE PUED IR INTERCAMBIANDO SEGUN POWER UPS O UN MENU DE MEJORES, Y ADMINISTRAR DISTINTOS EFECTOS PARA CADA PROVECTIL (FUEGO, VENENO, FRIO QUE RELENTICE, ETC) PODRIA IMPLEMENTAR FACILMENTE ESTE PATRON PARA QUE SEGUN EL TIPO DE ATAQUE QUE TENGO OCUPADO, UTILICE OTRO TIPO DE PROVECTIL.

MI JUEGO PERMITE APLICAR ESTE PATRON, PERO POR EL MOMENTO SE IMPLEMENTO EL SISTEMA DE DISTINTOS TIPOS DE PROVECTILES, POR ENDE NO SE HA APLICADO ESTE PATRÓN DE OTRA MANERA.

¡MUCHAS GRACIAS POR VER!