Ministerul Educației și Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică și Microelectronică

Departamentul Ingineria Software și Automatică

**Proiect de an**

Disciplina: **Tehnici și Mecanisme de Proiectare Software**

**Tema:** Proiectarea și realizarea unui sistem de studiere a șabloanelor de proiectare

A efectuat: student, gr: TI-202

Sandu Corneliu

A verificat: asistent universitar

Gaidau Mihai

Chișinău – 2023

**Cuprins:**

Introducere ................................................................................................................................. 3

Analiza domeniului de studiu ..................................................................................................... 4

1. Tehnologii utilizate ............................................................................................................. 4
   1. Platforma Unity 3d .................................................................................................. 5
   2. Limbajul de programare C# .................................................................................... 6
   3. Legătura dintre Unity 3d și C# ............................................................................... 8
   4. Visual Studio .......................................................................................................... 9
   5. Asset Store ............................................................................................................ 11
2. Realizarea proiectului și pattern-urile utilizate.................................................................. 12

1.6 Implementarea șablonului Prototype ....................................................................... 12

1.7 Implementarea șablonului Observer ......................................................................... 15

1.8 Implementarea șablonului Mediator ......................................................................... 17

1.9 Implementarea șablonului Greutatea Muștei ............................................................. 20

1. Documentarea produsului realizat .................................................................................... 22

Concluzie ................................................................................................................................. 26

Bibliografie .............................................................................................................................. 28

**Introducere**

Stagiul de practică reprezintă activitatea desfăşurată de studenţi în conformitate cu planul de învăţământ, care are drept scop verificarea aplicabilităţii cunoştinţelor teoretice însuşite în cadrul programului de pregătire profesională.

Obiectivul fundamental al acestei activităţi este de a dobîndi cunoștințe practice printr-o implicare directă alături de specialiştii din domeniul Tehnologiilor Informaționale.

Cu scopul de a demonstra efortul depus în timpul Practicii Tehnologice și de a obține o remunerare adecvată, am selectat o temă de cercetare specifică: "Crearea unui joc în 3D". Prezentul raport reflectă rezultatele obținute în urma acestui proces de dezvoltare.

În cadrul Practicii Tehnologice, pe lângă tema selectată, am dedicat timp pentru a studia și analiza două tehnologii principale, Unity3D și Visual Studio, pe care le-am utilizat pentru realizarea jocului meu.

Această alegere a practicii la companie a fost deosebit de benefică, deoarece, pe lângă jocul dezvoltat pentru a putea studia în Unity3D, am dobândit cunoștințe suplimentare privind crearea și optimizarea jocului.

Astfel, acest raport nu va include o descriere detaliată pas cu pas a dezvoltării jocului, ci se va concentra pe teoria nou învățată, limbajului de programare studiate în profunzime, diagrama UML a aplicației și interfața aplicației de gestionare a cheltuielilor.

Raportul nu va fi util doar pentru a evidenția munca depusă și materialele studiate în vederea realizării produsului final, ci și pentru cei interesați să studieze și să-și aprofundeze cunoștințele în domeniul Unity3D și Visual Studio.

În concluzie, voi oferi o evaluare generală a efortului depus, voi împărtăși impresiile rezultate din desfășurarea practicii la companie și, desigur, voi oferi sfaturi utile.

Raportul se va încheia cu o bibliografie, în care voi include link-uri de unde am extras informațiile care mi-au fost utile în studierea și dezvoltarea jocului.

**Analiza domeniului de studiu**

Domeniul de studiu abordat în cadrul acestui proiect se încadrează în tehnologiile informaționale și are la bază specializarea în domeniul dezvoltării jocurilor 3D.

Domeniul tehnologiilor informaționale, care este bine dezvoltat și atrage studenți interesați să-și dezvolte competențele în industria game-dev-ului. Având în vedere această specializare și pasiunea mea pentru acest domeniu, am decis să dezvolt un joc 3D în cadrul Practicii Tehnologice.

Am ales să dezvolt un joc 3D deoarece acesta reprezintă o modalitate captivantă de a explora potențialul tehnologiilor informaționale și de a oferi o experiență interactivă utilizatorilor. Jocurile 3D sunt populare și atrag interesul unui public divers, oferind o gamă largă de posibilități și provocări în implementarea lor.

Scopul jocului dezvoltat este de a oferi o experiență distractivă și captivantă utilizatorilor, prin intermediul unui mediu tridimensional interactiv. Am pus accent pe utilizarea platformelor Unity3D și Visual Studio, care sunt tehnologii renumite și potențiale puternice pentru dezvoltarea jocurilor 3D.

Acest raport nu va include doar detalii despre procesul de dezvoltare pas cu pas, ci și teoria și conceptele învățate în timpul cercetării, precum și limbajele de programare utilizate în implementarea jocului.

Raportul nu este doar o demonstrație a muncii depuse și a materialelor studiate pentru a ajunge la produsul final, ci și o resursă valoroasă pentru persoanele interesate să studieze și să-și aprofundeze cunoștințele în domeniul dezvoltării jocurilor 3D folosind platformele Unity3D și Visual Studio.

În concluzie, prin acest proiect, am avut oportunitatea de a explora tehnologiile informaționale într-un mod captivant și de a aduce contribuții în domeniul dezvoltării jocurilor 3D. Am învățat multe aspecte noi și am obținut experiență practică în implementarea unui joc tridimensional. De asemenea, am avut ocazia să lucrez în cadrul.

Raportul se va încheia cu o bibliografie, în care voi include sursele de informație utilizate pentru studiul și dezvoltarea jocului.

**Tehnologii utilizate**

* 1. **Platforma Unity 3D.**

Platforma Unity 3D este un motor de jocuri 3D și unelte de dezvoltare, cunoscută pentru versatilitatea și popularitatea sa în industria jocurilor video. A fost creată de către Unity Technologies și a fost lansată pentru prima dată în 2005. De atunci, Unity 3D a devenit unul dintre cele mai folosite motoare de jocuri, fiind utilizată pentru dezvoltarea jocurilor pe multiple platforme, inclusiv PC, console de jocuri, dispozitive mobile și realitate virtuală.

Principala caracteristică a Unity 3D este abilitatea de a dezvolta jocuri cross-platform, ceea ce înseamnă că poți crea un joc și să-l distribui pe mai multe platforme, fără a fi nevoie să rescrii codul de la zero pentru fiecare platformă în parte. Acest lucru facilitează dezvoltarea și lansarea jocurilor pe multiple dispozitive, economisind timp și efort dezvoltatorilor.

Unity 3D oferă un mediu de dezvoltare intuitiv, care permite programatorilor să scrie cod în limbajul C# sau în limbajul de programare visual scripting, numit UnityScript. De asemenea, platforma dispune de o gamă largă de unelte și funcționalități, inclusiv un editor grafic, suport pentru grafică avansată, fizică, animații, sunet, rețea și multe altele. Acestea permit dezvoltatorilor să creeze jocuri complexe și captivante.

Unity 3D oferă, de asemenea, suport pentru dezvoltarea jocurilor în realitate virtuală (VR) și realitate augmentată (AR). Cu ajutorul uneltelor speciale și al integrării cu diferite dispozitive VR și AR, dezvoltatorii pot crea experiențe imersive și captivante pentru utilizatori.

Datorită popularității sale și a comunității mari de dezvoltatori, Unity 3D dispune de o gamă largă de tutoriale, documentație și resurse disponibile online. Acest lucru face mai ușoară învățarea și dezvoltarea de jocuri cu ajutorul platformei Unity 3D.

În concluzie, Unity 3D este o platformă puternică și versatilă de dezvoltare a jocurilor 3D, care oferă dezvoltatorilor un mediu de dezvoltare accesibil și unelte puternice pentru crearea de jocuri captivante pe multiple platforme.

* 1. **Limbajul de programare C#.**

C# (pronunțat "C sharp") este un limbaj de programare modern, puternic și orientat pe obiect, dezvoltat de către Microsoft. A fost lansat pentru prima dată în anul 2000 și face parte din familia de limbaje C, fiind influențat în special de C++ și Java. C# este un limbaj de programare versatil, care este folosit în principal pentru dezvoltarea de aplicații Windows, aplicații web, aplicații mobile și jocuri.

Câteva caracteristici cheie ale limbajului de programare C#:

Orientat pe obiect: C# este un limbaj de programare orientat pe obiect, ceea ce înseamnă că programele sunt structurate în jurul obiectelor, care reprezintă entități cu caracteristici și comportamente specifice. Aceasta facilitează dezvoltarea de aplicații modulare, ușor de întreținut și de extins.

Siguranță la tipuri de date: C# este un limbaj strict de tip, ceea ce înseamnă că toate variabilele și obiectele trebuie să fie declarate cu un tip specific și că verificarea tipului este realizată în timpul compilării. Aceasta ajută la prevenirea erorilor de tip și oferă o mai mare siguranță în timpul execuției programului.

Gestionarea automată a memoriei: C# utilizează un sistem de gestionare automată a memoriei cunoscut sub numele de "garbage collection". Acest sistem se ocupă de eliberarea automată a memoriei ocupate de obiectele care nu mai sunt utilizate, ceea ce reduce riscul de erori de memorie și simplifică dezvoltarea aplicațiilor.

Sintaxă clară și expresivă: C# oferă o sintaxă clară și expresivă, ușor de înțeles și de scris. Acest lucru facilitează dezvoltarea rapidă și productivă a aplicațiilor.

Interoperabilitate cu platforma .NET: C# este un limbaj de programare care rulează pe platforma .NET, ceea ce oferă acces la un set vast de biblioteci și unelte pentru dezvoltarea de aplicații. Acest lucru facilitează integrarea cu alte tehnologii și extinde capabilitățile limbajului.

Suport pentru programare paralelă: C# oferă suport nativ pentru programarea paralelă și concurentă, permițând dezvoltatorilor să creeze aplicații eficiente și scalabile, care pot beneficia de avantajele sistemelor multi-core și distribuite.

C# este un limbaj popular și larg utilizat în industria software. Este folosit pentru dezvoltarea de aplicații desktop, aplicații web cu ASP.NET, aplicații mobile cu Xamarin și jocuri cu ajutorul platformei Unity 3D. Datorită capacității sale de a crea aplicații robuste și performante, C# rămâne un limbaj de programare preferat pentru mulți dezvoltatori.

Iată un exemplu simplu de cod C# pentru un joc 3D folosind platforma Unity 3D:

using UnityEngine;

public class PlayerController : MonoBehaviour

{

public float speed = 5f;

private Rigidbody rb;

private void Start()

{

rb = GetComponent<Rigidbody>();

}

private void Update()

{

float moveHorizontal = Input.GetAxis("Horizontal");

float moveVertical = Input.GetAxis("Vertical");

Vector3 movement = new Vector3(moveHorizontal, 0f, moveVertical);

rb.AddForce(movement \* speed);

}

}

**Explicație:**

Acest script este responsabil de controlul jucătorului într-un joc 3D. El adaugă o forță la componenta rigidbody a obiectului (jucătorului) pe baza inputului utilizatorului.

Pentru a utiliza acest script, trebuie să adăugați un obiect jucător în scena Unity și să atașați acest script la obiectul respectiv. Apoi, puteți configura viteza jucătorului în inspectorul Unity.

Acest exemplu utilizează metoda Start() pentru a obține referința la componenta Rigidbody a jucătorului în momentul începerii jocului. Apoi, în metoda Update(), obținem inputul utilizatorului prin metoda Input.GetAxis(), care returnează o valoare între -1 și 1 în funcție de tastele direcționale apăsate.

Vectorul movement este creat pe baza inputului utilizatorului și multiplicat cu viteza jucătorului. Apoi, această valoare este adăugată la componenta rigidbody a jucătorului folosind metoda AddForce(), ceea ce va face ca jucătorul să se miște în scena în funcție de inputul utilizatorului.

Acesta este doar un exemplu simplu de cod pentru controlul jucătorului într-un joc 3D folosind Unity 3D. Într-un joc mai complex, veți avea nevoie de logica suplimentară pentru gestionarea coliziunilor, animațiilor, interacțiunilor cu alte obiecte și multe altele.

* 1. **Legătura dintre Unity 3d și C#.**

Unity 3D este un motor de jocuri foarte popular și puternic care permite dezvoltatorilor să creeze jocuri și aplicații interactive pentru o varietate de platforme, cum ar fi PC, console, dispozitive mobile și realitate virtuală/aumentată. Pentru a programa în Unity 3D, se folosește în principal limbajul de programare C# (C Sharp).

C# este un limbaj de programare orientat pe obiect, dezvoltat de Microsoft, care oferă o sintaxă clară și puternică. Unity 3D utilizează C# ca limbaj de scripting principal pentru dezvoltarea jocurilor și a aplicațiilor. Prin intermediul C#, poți controla comportamentul obiectelor, interacțiunea utilizatorului, grafica, fizica și multe alte aspecte ale jocului.

Unity 3D oferă un mediu de dezvoltare integrat (IDE) numit Unity Editor, care are suport nativ pentru C#. Acest editor îți permite să scrii și să editezi scripturi C# direct în interfața sa. Acolo poți crea scripturi pentru a defini comportamentul obiectelor și pentru a răspunde la evenimente precum apăsarea unei taste sau coliziunea cu alte obiecte.

Unity 3D oferă o serie de funcționalități și clase predefinite în limbajul C#, care te ajută să interacționezi cu motorul de joc și să controlezi jocul. De exemplu, poți accesa componente ale obiectelor în scenă, poți crea și distruge obiecte, poți manipula animații și grafică, poți gestiona sunete și multe altele, folosind clasele și metodele disponibile în Unity 3D.

În concluzie, legătura dintre Unity 3D și C# este strânsă și esențială pentru dezvoltarea jocurilor și a aplicațiilor în Unity. Prin programarea în C# în cadrul Unity 3D, poți controla și manipula toate aspectele jocului și poți crea experiențe interactive complexe.

* 1. **Visual Studio.**

Visual Studio este un mediu integrat de dezvoltare (IDE) puternic și extensibil utilizat în mod obișnuit pentru dezvoltarea aplicațiilor Unity 3D. Este dezvoltat de Microsoft și oferă un set complet de instrumente și facilități pentru crearea, testarea, depanarea și implementarea aplicațiilor Unity 3D.

Iată câteva caracteristici cheie ale Visual Studio pentru dezvoltarea în Unity 3D:

Editor de cod avansat: Visual Studio oferă un editor de cod puternic, cu funcționalități precum evidențierea sintaxei, completarea automată a codului, navigarea rapidă și refactorizarea codului. Aceste caracteristici facilitează scrierea și gestionarea codului sursă în limbajul C# și permit dezvoltatorilor Unity 3D să lucreze într-un mediu de dezvoltare eficient.

Gestionarea soluțiilor și proiectelor: Visual Studio permite dezvoltatorilor Unity 3D să organizeze și să gestioneze soluțiile și proiectele lor într-un mod eficient. Aceasta include crearea

și gestionarea proiectelor multiple, gestionarea dependențelor și configurarea proiectului, precum și lucrul cu diverse componente și module ale aplicației Unity 3D.

Debogare avansată: Visual Studio oferă instrumente puternice pentru debogarea aplicațiilor Unity 3D. Dezvoltatorii pot utiliza puncte de oprire, urmărirea execuției pas cu pas, analiza valorilor variabilelor și multe altele pentru a identifica și rezolva erorile și problemele de cod în aplicațiile lor Unity 3D.

Extensibilitate: Visual Studio este un IDE extrem de extensibil, permițând dezvoltatorilor să adauge extensii și plugin-uri pentru a extinde funcționalitățile IDE-ului în contextul dezvoltării de aplicații Unity 3D. Aceasta oferă posibilitatea de a integra unelte suplimentare și suport pentru tehnologii specifice sau platforme suplimentare.

Comunitate și resurse: Utilizatorii Visual Studio beneficiază de o comunitate activă și resurse bogate, inclusiv forumuri, tutoriale, documentație și resurse de învățare, dedicate dezvoltării de aplicații Unity 3D. Aceasta oferă un suport valoros pentru dezvoltatori și facilitează împărtășirea cunoștințelor și soluțiilor specifice dezvoltării în Unity 3D.

Prin utilizarea Visual Studio în contextul dezvoltării de aplicații Unity 3D, dezvoltatorii beneficiază de o suită puternică de instrumente și resurse pentru a crea și gestiona aplicații complexe și captivante în acest mediu de dezvoltare.

* 1. **Asset Store.**

Asset Store este o platformă online dezvoltată de Unity Technologies, care permite dezvoltatorilor de jocuri să găsească și să achiziționeze resurse, instrumente și alte active digitale pentru a-și dezvolta proiectele. Este disponibilă în cadrul mediului de dezvoltare Unity și oferă o gamă variată de active, inclusiv modele 3D, texturi, materiale, sunete, muzică, efecte vizuale, plugin-uri și alte componente software.

Câteva detalii importante despre Asset Store:

Accesibilitate: Asset Store este disponibil ca parte a mediului de dezvoltare Unity și poate fi accesat direct din editorul Unity. Dezvoltatorii pot naviga, căuta și vizualiza activele disponibile și pot achiziționa rapid resursele necesare pentru proiectele lor.

**Categorii de active:** Asset Store oferă o gamă largă de categorii de active, inclusiv modele 3D, texturi, animații, sunete, muzică, scripturi, plugin-uri și alte resurse utile. Aceste active pot fi utilizate pentru a îmbunătăți aspectul vizual, funcționalitatea și experiența generală a jocului.

**Active gratuite și plătite:** Asset Store include atât active gratuite, cât și active plătite. Dezvoltatorii pot alege să achiziționeze activele care se potrivesc nevoilor lor, iar prețurile variază în funcție de complexitatea și valoarea activului.

**Comunitate și recenzii:** Asset Store are o comunitate activă de utilizatori, care pot lăsa recenzii și evaluări pentru activele pe care le-au achiziționat și utilizat. Acest lucru ajută dezvoltatorii să facă alegeri informate atunci când caută active pe care să le integreze în proiectele lor.

**Integrare ușoară:** Activele achiziționate din Asset Store pot fi descărcate și integrate direct în proiectele Unity. Editorul Unity oferă un flux de lucru simplu și facilitează importul și utilizarea activelor achiziționate.

**Suport și actualizări:** Asset Store are un sistem de suport și actualizări pentru activele achiziționate. Dezvoltatorii pot primi actualizări și suport tehnic pentru a rezolva problemele sau pentru a profita de noi funcționalități adăugate.

Asset Store este un instrument valoros pentru dezvoltatorii de jocuri Unity, deoarece oferă o gamă largă de resurse și instrumente pentru a îmbunătăți calitatea și performanța jocurilor lor.

**Realizarea proiectului și pattern-urile utilizate**

**1.6 Prototype**

**Prototipul** este un model generativ care vă permite să copiați obiecte fără a intra în detaliile implementării lor.

Vă permite să creați noi obiecte pe baza prototipurilor existente, ceea ce poate economisi timp și resurse.

De exemplu, un joc poate folosi modelul Prototype pentru a crea mai multe copii ale aceluiași obiect, cum ar fi crearea mai multor monștri la același nivel.

Ca de exemplu noi modificăm distanța sau viața acestui inamic părinte, prin urmare acesta modifică la toate copiile existente pe scenă sau înafara ei.

public class EnemyStats : MonoBehaviour

{

private const string AnimAttack = "Attack", AnimWalk = "Walk", AnimIdle = "Idle";

Animator animator;

DamageCollider damageCollider;

public int soulsAwardedOnDeath = 100;

public Transform target;

public float distanceToFollow = 10f;

public float distanceToStop = 1f;

public float distanceToStopFollowing = 15f;

public Vector3 startingPosition;

private NavMeshAgent navMeshAgent;

public int currentHealth;

public int maxHealth;

public int healthLevel = 10;

public GameObject enemy;

private void Awake()

{

animator = GetComponentInChildren<Animator>();

}

private void Start()

{

navMeshAgent = GetComponent<NavMeshAgent>();

startingPosition = transform.position;

maxHealth = SetMaxHealthFromHealthLevel();

currentHealth = maxHealth;

maxHealth = enemy.GetComponent<EnemyStats>().maxHealth;

}

void Update()

{

float distanceToTarget = Vector3.Distance(transform.position, target.position);

if (target != null && distanceToTarget <= distanceToFollow)

{

Vector3 direction = (target.position - transform.position).normalized;

Vector3 targetPosition = target.position - direction \* distanceToStop;

transform.LookAt(target);

animator.SetBool(AnimWalk, true);

if (Vector3.Distance(transform.position, targetPosition) > distanceToStop)

{

navMeshAgent.SetDestination(targetPosition);

}

else

{

animator.SetBool(AnimWalk, false);

animator.SetBool(AnimAttack, true);

navMeshAgent.SetDestination(transform.position);

}

}

if (damageCollider == true)

{

currentHealth = enemy.GetComponent<EnemyStats>().currentHealth;

}

}

private int SetMaxHealthFromHealthLevel()

{

maxHealth = healthLevel \* 10;

return maxHealth;

}

public void TakeDamage(int damage)

{

currentHealth = currentHealth - damage;

animator.Play("Damage");

if (currentHealth <= 0)

{

HandleDeath();

}

}

public void Die()

{

Collider enemyCollider = GetComponentInChildren<Collider>();

enemyCollider.enabled = false;

}

private void HandleDeath()

{

currentHealth = 0;

Die();

animator.Play("Death");

distanceToFollow = 0f;

}

public void AwardSoulsOnDeath()

{

PlayerStats playerStats = FindObjectOfType<PlayerStats>();

SoulCountBar soulCountBar = FindObjectOfType<SoulCountBar>();

if (playerStats != null)

{

playerStats.AddSouls(soulsAwardedOnDeath);

if (soulCountBar != null)

{

soulCountBar.SetSoulCountText(playerStats.soulCount);

}

}

}

}

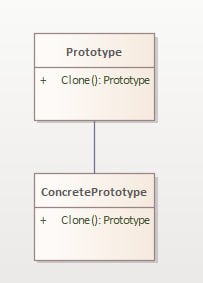


Figura 1.1 – Structura șablonului Prototype

Clasa "Prototype" definește interfața pentru obiectele care pot fi clonate și conține metoda "Clone" care returnează o instanță clonată a obiectului.

Clasa "ConcretePrototype" implementează obiectele prototip și conține o implementare concretă a metodei "Clone", prin care obiectul este clonat și returnat ca un nou obiect de tipul Prototype.

**Observer** este un model de comportament care permite obiectelor să fie notificate cu privire la schimbările în starea altor obiecte la care sunt abonate.

Jocul poate folosi acest model pentru a crea un sistem de evenimente care anunță jucătorii cu privire la diferite modificări, de exemplu, despre apariția monștrilor sau actualizarea inventarului.

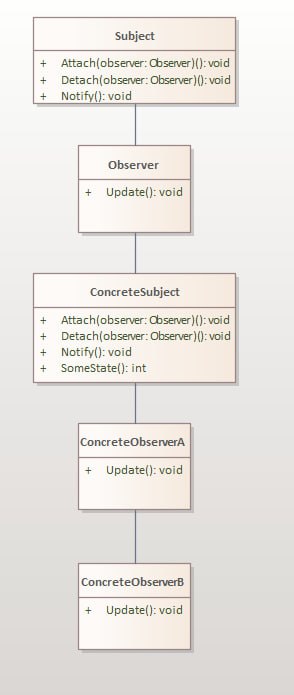


Figura 1.2 – Structura șablonului Observer.

Clasa "Subject" definește interfața pentru subiect, care are abonamente pentru obiectele observatoare și oferă metode pentru atașarea, detasarea și notificarea observatorilor.

Clasa "Observer" reprezintă o interfață pentru observatori și definește metoda "Update" care va fi apelată de către subiect atunci când acesta este notificat.

Clasa "ConcreteSubject" implementează subiectul și menține o listă de observatori înregistrați. Aceasta implementează metodele pentru atașarea, detasarea și notificarea observatorilor, precum și un anumit stare internă, cum ar fi "SomeState".

Clasa "ConcreteObserverA" și "ConcreteObserverB" reprezintă observatori specifici și implementează metoda "Update" pentru a răspunde la notificările primite de la subiect.

public class MonsterNotification : MonoBehaviour

{

[SerializeField] private GameObject monsterPrefab;

[SerializeField] private GameObject notificationPanel;

[SerializeField] private Text notificationText;

[SerializeField] private float monsterSpawnDelay = 10f;

[SerializeField] private float notificationDuration = 5f;

private void Start()

{

Invoke("SpawnMonster", monsterSpawnDelay);

}

private void SpawnMonster()

{

Instantiate(monsterPrefab, transform.position, Quaternion.identity);

notificationText.text = "Монстр появился!";

notificationPanel.SetActive(true);

Invoke("HideNotification", notificationDuration);

}

private void HideNotification()

{

notificationPanel.SetActive(false);

}

}

**Mediatorul** este un model de comportament care vă permite să reduceți cuplarea dintre obiecte prin mutarea logicii de legare într-o clasă de mediator separată.

Într-un joc, puteți folosi acest model pentru a controla interacțiunea diferitelor obiecte, cum ar fi gestionarea coliziunilor sau interacțiunile cu personajele.

public class ChestController : Interactable

{

[SerializeField] Animator animator;

public GameObject item;

public GameObject openChestPanel;

public bool isOpen = false;

public bool isNear = false;

private GameObject loot;

public override void Interact(Player playerController)

{

animator.Play("Chest Open");

isOpen = true;

loot = Instantiate(item, transform.position, Quaternion.identity);

loot.GetComponent<Collider>().enabled = false;

openChestPanel.SetActive(false);

GetComponent<Collider>().enabled = false;

}

private void OnTriggerEnter(Collider other)

{

if (other.CompareTag("Player"))

{

Debug.Log("Open Chest Collider");

openChestPanel.SetActive(true);

}

}

private void OnTriggerExit(Collider other)

{

if (other.CompareTag("Player"))

{

Debug.Log("Close Chest Collider");

openChestPanel.SetActive(false);

}

}

public void DropItems() // use animation event

{

loot.GetComponent<Collider>().enabled = true;

}

}

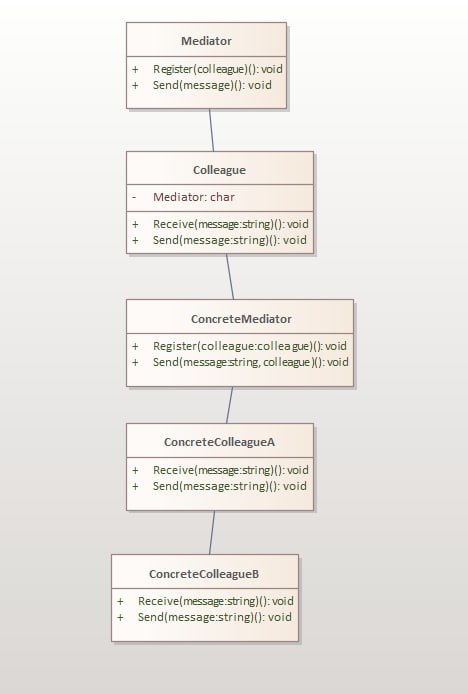


Figura 1.3 – Structura șablonului Mediator

Clasa "AbstractMediator" definește interfața pentru mediator, care gestionează comunicarea între colegi. Aceasta oferă metode pentru înregistrarea unui coleg și trimiterea unui mesaj către un coleg specific.

Clasa "AbstractColleague" reprezintă o interfață pentru colegi și conține o referință la mediator. Această clasă definește metode pentru trimiterea și primirea mesajelor.

Clasa "ConcreteMediator" implementează mediatorul și gestionează comunicarea între colegi. Aceasta conține o listă de colegi înregistrați și implementează metodele de înregistrare și trimitere a mesajelor.

Clasa "ConcreteColleagueA" și "ConcreteColleagueB" reprezintă colegi specifici și implementează metodele pentru trimiterea și primirea mesajelor.

**Greutatea muștei** este un model de design structural care vă permite să lucrați eficient cu un număr mare de obiecte mici,proprietăți și metode comune, prin separarea lor în componente separabile și inseparabile.

Acest lucru vă permite să reduceți numărul de instanțe de clasă și să reduceți cantitatea de memorie utilizată.

Exemplu: într-un joc, modelul Fly Weight poate fi folosit pentru a optimiza manipularea multor obiecte mici, cum ar fi gloanțe, explozii, schije, monștri etc.

În loc să creați un obiect separat pentru fiecare marcator, puteți utiliza un obiect partajat care va reprezenta un singur marcator și va stoca toate proprietățile și metodele acestuia.

În acest caz, fiecare marcator va avea propriile coordonate și direcție unice, care vor fi transferate la obiectul partajat atunci când este creat sau actualizat.

Astfel, puteți reduce semnificativ numărul de instanțe de clasă și puteți economisi memorie.

public class MonsterNotification : MonoBehaviour

{

[SerializeField] private GameObject monsterPrefab;

[SerializeField] private GameObject notificationPanel;

[SerializeField] private Text notificationText;

[SerializeField] private float monsterSpawnDelay = 2f;

[SerializeField] private float notificationDuration = 5f;

[SerializeField] private float respawnDelay = 10f;

private bool canSpawnMonster = true;

private int killedEnemiesCount = 0;

private GameObject currentMonster;

private void Start()

{

InvokeRepeating(nameof(SpawnMonster), monsterSpawnDelay, monsterSpawnDelay);

}

private void SpawnMonster()

{

if (canSpawnMonster)

{

notificationText.text = "Монстр появился!";

notificationPanel.SetActive(true);

currentMonster = Instantiate(monsterPrefab, transform.position, Quaternion.identity);

canSpawnMonster = false;

Invoke(nameof(KillMonster), respawnDelay);

}

Invoke(nameof(HideNotification), notificationDuration);

}

private void HideNotification()

{

notificationPanel.SetActive(false);

}

private void KillMonster()

{

if (currentMonster != null)

{

Destroy(currentMonster);

canSpawnMonster = true;

}

}

private void EnemyKilled()

{

killedEnemiesCount++;

}

}

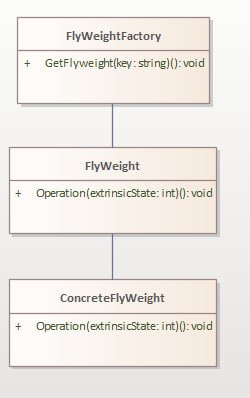


Figura 1.4 – Structura șablonului Observer

Clasa "Subject" definește interfața pentru subiect, care are abonamente pentru obiectele observatoare și oferă metode pentru atașarea, detasarea și notificarea observatorilor.

Clasa "Observer" reprezintă o interfață pentru observatori și definește metoda "Update" care va fi apelată de către subiect atunci când acesta este notificat.

Clasa "ConcreteSubject" implementează subiectul și menține o listă de observatori înregistrați. Aceasta implementează metodele pentru atașarea, detasarea și notificarea observatorilor, precum și un anumit stare internă, cum ar fi "SomeState".

Clasa "ConcreteObserverA" și "ConcreteObserverB" reprezintă observatori specifici și implementează metoda "Update" pentru a răspunde la notificările primite de la subiect.

**Documentarea produsului realizat**

Dark Fantasy este un joc video de tip RPG (role-playing game) care se desfășoară într-un univers întunecat și plin de fantezie. Jocul pune accentul pe atmosfera gotică, elementele horror și povestea profundă.

În lumea Dark Fantasy, jucătorii preiau rolul unui erou sau a unei eroine ce trebuie să se confrunte cu forțele malefice ce amenință să distrugă întreaga lume. Aceasta este o lume plină de creaturi supranaturale, magie neagră, locuri misterioase și personaje întunecate.

Jucătorii își pot personaliza personajul, alegând dintre diferite clase, cum ar fi războinic, mag, vânător sau hoț. Fiecare clasă are abilități și stiluri de luptă specifice, permițând jucătorilor să-și creeze propria strategie în timpul confruntărilor.

Pe parcursul jocului, jucătorii vor explora locuri întunecate și periculoase, cum ar fi castele bântuite, păduri misterioase și catacombe înspăimântătoare. Aceste locuri ascund secrete și comori, dar și inamici puternici și capcane mortale. Jucătorii trebuie să fie vigilenți și să-și folosească abilitățile în mod inteligent pentru a înfrunta adversarii și a rezolva puzzle-uri complexe.

Pe măsură ce povestea se dezvoltă, jucătorii vor întâlni personaje non-jucători cu propriile lor povești și motivații. Alegerile pe care le fac jucătorii pot influența desfășurarea poveștii și relațiile cu aceste personaje.

Dark Fantasy oferă, de asemenea, și un sistem de progresie a personajului, care permite jucătorilor să-și îmbunătățească abilitățile și să obțină echipament mai puternic pe măsură ce avansează în joc. Aceasta le permite să se pregătească pentru confruntările dificile și să devină mai puternici în fața inamicilor lor.

În concluzie, Dark Fantasy este un joc captivant care îmbină atmosfera întunecată, elementele de fantezie și acțiunea dintr-un RPG clasic. Cu povestea sa profundă și lumea sa misterioasă, oferă jucătorilor o experiență unică într-un univers întunecat și plin de surprize.

**Funcționalități cheie:**

**Niveluri de dificultate:** Jocul oferă nivele de dificultate variate pentru a se potrivi cu abilitățile și preferințele fiecărui jucător. Aceste niveluri de dificultate pot influența inteligența și agresivitatea inamicilor, rezistența personajului jucătorului, precum și complexitatea puzzle-urilor și a quest-urilor.

La nivelurile de dificultate mai reduse, jocul poate fi mai accesibil și mai prietenos pentru jucătorii mai puțin experimentați sau pentru cei care doresc o experiență mai relaxată. Inamicii pot fi mai slabi, mai puțini sau mai previzibili, iar puzzle-urile pot fi mai simple.

Pe de altă parte, nivelurile de dificultate mai ridicate vor testa abilitățile și strategiile jucătorilor avansați. Inamicii pot fi mai puternici, mai rapizi și mai agresivi, cerând o abordare tactica și o gestionare eficientă a resurselor. Puzzle-urile și quest-urile pot fi mai complexe și mai provocatoare, punând la încercare inteligența și răbdarea jucătorului.

Indiferent de nivelul de dificultate ales, Dark Fantasy oferă o experiență captivantă și provocatoare, în care jucătorii trebuie să-și folosească abilitățile și cunoștințele pentru a înfrunta pericolele și a-și îndeplini misiunile în acest univers întunecat și periculos.

**Punctaje și clasamente:** Pe parcursul jocului, vei obține puncte pentru diverse realizări, cum ar fi înfrângerea inamicilor, rezolvarea puzzle-urilor sau completarea misiunilor. Cu cât obții mai multe puncte, cu atât vei avansa în clasamentul jocului și vei putea vedea cum te compari cu alți jucători.

Clasamentul îți oferă o perspectivă asupra rezultatelor tale și a poziției tale în comunitatea de jucători. Poți vedea cine sunt cei mai buni jucători și poți încerca să-ți îmbunătățești poziția prin obținerea de punctaje mai mari.

Pe lângă punctaje și clasamente, jocul Dark Fantasy poate avea și alte sisteme de recompense. De exemplu, poți câștiga obiecte speciale, abilități noi sau echipament puternic pe măsură ce avansezi în joc și obții rezultate remarcabile.

Aceste punctaje, clasamente și recompense îți oferă o motivație suplimentară pentru a-ți îmbunătăți abilitățile, a explora în profunzime jocul și a-ți atinge obiectivele personale. Cu fiecare punctaj înalt și clasare superioară, vei simți o satisfacție mai mare și vei dori să mergi mai departe în călătoria ta întunecată din Dark Fantasy.

**Explicații detaliate:** Pe parcursul aventurii tale, vei întâlni diferite elemente, personaje și evenimente care necesită o mai bună înțelegere. Jocul îți va oferi explicații detaliate pentru aceste aspecte, prezentând informații suplimentare și contextul necesar pentru a înțelege mai bine ceea ce se întâmplă în lumea Dark Fantasy.

De exemplu, atunci când descoperi un nou personaj sau un inamic puternic, jocul îți poate oferi o prezentare detaliată a caracteristicilor lor, a motivațiilor lor și a impactului pe care îl pot avea asupra povestirii. Astfel, vei putea să iei decizii informate și să te adaptezi mai bine la situații.

De asemenea, în timpul jocului, poți descoperi obiecte sau artefacte speciale. Explicațiile detaliate îți vor oferi informații despre proprietățile și utilizările acestor obiecte, permițându-ți să le folosești în mod strategic în lupte sau puzzle-uri.

Explicațiile detaliate pot fi, de asemenea, disponibile pentru puzzle-uri sau provocări specifice din joc. Acestea îți vor oferi indicii și strategii pentru a le rezolva și a înainta în poveste.

Prin intermediul acestor explicații detaliate, jocul Dark Fantasy te ajută să-ți îmbunătățești înțelegerea lumii sale și să te implici mai profund în poveste. Astfel, vei avea o experiență mai captivantă și vei putea savura fiecare aspect al acestui univers întunecat.

**Cum să joci:**

**Începe jocul:** Accesează platforma sau dispozitivul pe care este instalat Dark Fantasy. Selectează opțiunea "Start" sau "New Game" din meniul principal al jocului. Întâlnești diferiți inamici și creaturi în timpul aventurii tale. Folosește armele, abilitățile și tactici de luptă pentru a învinge acești adversari și a-ți proteja personajul. Lasă-te absorbit de povestea întunecată și de atmosfera captivantă a jocului Dark Fantasy și bucură-te de fiecare moment al aventurii tale.

**Obține punctaje:** Atunci când te lupți cu inamicii din joc, vei primi puncte pentru fiecare înfrângere reușită. Folosește-ți abilitățile și strategiile de luptă pentru a obține scoruri înalte și pentru a-ți proteja personajul.

**Explorează:** Obiecte și echipament: Pe măsură ce explorezi, fii atent la obiectele și echipamentul pe care le găsești. Acestea pot fi ascunse în cutii, la sfârșitul traseelor alternative sau în locații secrete. Colectează aceste obiecte pentru a-ți îmbunătăți echipamentul și abilitățile personajului..

**Continuă să joci:** depinde de preferințele tale și de cât de mult timp dorești să petreci în aventura întunecată, asta pute accent pe cît de pregătit v-ei fi moral și spiritual.

**Concluzie**

Pattern-ul Prototype este utilizat în joc pentru a crea copii ale obiectelor existente, oferind astfel o modalitate eficientă de a genera noi entități fără a recrea întregul obiect de la zero. Acest lucru poate fi util în cazul în care jocul are nevoie de mai multe instanțe ale aceluiași tip de obiect, dar cu proprietăți și comportamente ușor diferite.

Pattern-ul Flyweight este folosit pentru a economisi memorie prin partajarea eficientă a datelor între obiecte similare. În cazul jocului Dark Fantazy, acest pattern poate fi aplicat pentru a reduce cantitatea de memorie consumată de obiecte precum texturi, modele 3D sau sunete, prin partajarea acestor resurse între entități similare.

Pattern-ul Mediator este utilizat pentru a gestiona comunicarea între diferitele componente ale jocului. În Dark Fantazy, acest pattern poate fi aplicat pentru a facilita interacțiunile dintre entități, cum ar fi personajele jucătorului, inamicii și mediul înconjurător. Mediatorul acționează ca un intermediar care dirijează comunicarea și coordonează acțiunile între entități, menținând astfel o structură flexibilă și slab cuplată.

Pattern-ul Observer este utilizat în joc pentru a permite monitorizarea evenimentelor și reacționarea dinamică la acestea. Acest pattern facilitează comunicarea slaba între obiecte, astfel încât atunci când un eveniment are loc, entitățile observatoare pot fi notificate și pot lua măsuri corespunzătoare. În Dark Fantazy, acest pattern poate fi aplicat pentru a gestiona evenimente precum coliziuni, interacțiuni cu obiecte sau schimbări de stare.

Platforma Unity este esențială în dezvoltarea jocului Dark Fantazy. Unity oferă un mediu de dezvoltare integrat și un set complet de instrumente și funcționalități pentru crearea de jocuri. Cu ajutorul Unity, dezvoltatorii pot implementa și integra ușor pattern-urile de proiectare menționate mai sus în jocul lor, facilitând astfel crearea unui joc captivant, eficient și modular.

În concluzie, jocul Dark Fantazy utilizează pattern-uri de proiectare precum Prototype, Flyweight, Mediator și Observer pentru a îmbunătăți performanța, modularitatea și gestionarea interacțiunilor din joc. Platforma Unity oferă un mediu puternic și complet pentru dezvoltarea jocului, contribuind la crearea unei experiențe de joc captivante și bine structurate.

**Bibliografie:**

1. Informații despre platforma Unity, [Resursă electronică], [Citat 30.01.2022] - Link-ul de acces: https://unity.com/ru
2. Informații despre Microsoft Visual Studio, [Resursă electronică], [Citat 01.02.2022] - Link-ul de acces: https://visualstudio.microsoft.com/ru/
3. Limbajul de Programare C#, [Resursă electronică], [Citat 06.02.2022] - Link-ul de acces: <https://ro.wikipedia.org/wiki/C_sharp>
4. Tutorial, [Resursă electronică], [Citat 10.02.2022] - Link-ul de acces: https://www.youtube.com/@SebastianGraves/featured
5. Magazinul Unity Asset Store, [Resursă electronică], [Citat 09.02.2022] - Link-ul de acces: <https://assetstore.unity.com/>
6. Cartea cu Design Pattern-uri [Resursă electronică] - Link-ul de acces: https://refactoring.guru/design-patterns/creational-patterns