UNIVERSITATEA NAȚIONALĂ DE ȘTIINȚĂ ȘI TEHNOLOGIE POLITHENICA BUCUREȘTI

Escape Room

Studenți : Andrei Mihail-Alexandru

Nicolae Gabriel

Stoica George-Cristian

Grupa : 444C

Prof. coordonator : ing. Anamaria Dumitrescu

An universitar 2024-2025

Cuprins

[1.Introducere 3](#_Toc187772257)

[1.1. Contextul proiectului 3](#_Toc187772258)

[1.2. Obiectivele lucrării 4](#_Toc187772259)

[2. Descrierea jocului 5](#_Toc187772260)

[2.1. Concept și tematică 5](#_Toc187772261)

[2.2. Public țintă 5](#_Toc187772262)

[3 .Tehnologii utilizate 6](#_Toc187772263)

[3.1. Unity și limbajul C# 6](#_Toc187772264)

[3.2. Alte resurse utilizate 6](#_Toc187772265)

[4. Design și implementare 7](#_Toc187772266)

[4.1. Structura și mecanicile jocului 7](#_Toc187772267)

[4.2. Grafică și scripting 7](#_Toc187772268)

[5. Testare și optimizare 9](#_Toc187772269)

[5.1. Metode de testare 9](#_Toc187772270)

[5.2. Îmbunătățiri și performanță 9](#_Toc187772271)

[6. Concluzii 9](#_Toc187772272)

[6.1. Rezultate și contribuții personale 9](#_Toc187772273)

[6.2. Perspective de dezvoltare 10](#_Toc187772274)

[7. Bibliografie 10](#_Toc187772275)

[8. Anexă 11](#_Toc187772276)

# 1.Introducere

Industria jocurilor video a evoluat rapid în ultimii ani, reprezentând una dintre cele mai dinamice și inovatoare forme de divertisment, a secolului XXI, aleasă de oameni. Din multitudinea de tipuri de jocuri existente pe piață, unul dintre cele mai interesante îl reprezintă jocurile de tip Escape Room, care necesită dexteritate, atenție la detalii, prezentând o atmosferă captivantă.

Aceste jocuri pun accent pe gândirea critică, rezolvarea de probleme și colaborare, în cazul jocurilor Escape Room de tip multiplayer, fiind folosite atât pentru divertisment și relaxare, cât și în contexte educaționale sau de training.

Crearea unui joc de acest fel în Unity reprezintă o oportunitate excelentă de a explora și analiza procesul de dezvoltare al unui joc 3D interactiv. Unity este una dintre cele mai utilizate platforme pentru dezvoltarea jocurilor datorită accesibilității și flexibilitășii sale, oferind intrumente pentru scripting, modelare grafică și optimizare.

Acest proiect a fost conceput cu scopul de a combina creativitatea, logica și tehnologia întru-un joc captivant care să testeze abilitățile jucătorilor.

În capitolele următoare, se vor prezenta aspecte tehnice, cât și cele creative ale designului jcului, urmărind să creeze o experiență cât mai captivantă, plăcută, pentru utilizatori.

## 1.1. Contextul proiectului

În ultimii ani, conceptul de escape room a devenit din ce în ce mai popular, fiind adoptat atât în format fizic, cât și în cel virtual. Camerele de evadare din lumea reală au reprezentat o sursă de inspirație pentru dezvoltarea de jocuri video care simulează aceste experiențe într-un mediu digital. Astfel, jucătorii pot experimenta aceleași provocări intelectuale, dar cu posibilități extinse oferite de tehnologie, cum ar fi interacțiunile complexe și scenariile creative.

În paralel, industria dezvoltării de jcuri a fost marcată de evoluția constantă a tehnologiilor 3D și a platformelor precum Unity, care au facilitat accesul dezvoltatorilor independenți la instrumente profesionale. Unity oferă funcționalități avansate pentru design, programare și optimizare, fiind ideal pentru crearea de ocuri care combină provocările logice cu elemente vizuale imersive.

Prin acest proiect ne popunem să profităm de oportunitățile tehnologice și de popularitatea în creștere a jocurilor de tip escape room, explorând cum un astfel de joc poate fi realizat, optimizat și adaptat pentru a oferi o experiență captivantă jucătorilor.

## 1.2. Obiectivele lucrării

Acest proiect are ca scop pricipal documentarea procesului de dezvoltare a unui joc de tip escape room în Unity, precum și evidențierea aspectelor thenice și creative implicate. Proiectul își propune să îmbine creativitatea cu tehnologia pentru a crea un produs captivant și funcțional. Obiectivele specifice ale lucrării sunt :

* Conceperea unui design atractiv pentru joc
* Stabilirea unei tematici captivante și coerente pentru joc.
* Crearea unor puzzle-uri logice care să ofere o provocare echilibrată jucătorilor.
* Dezvoltarea mecanicilor de joc
* Implementarea interacțiunilor dintre jucător și obiectele din mediul virtual.
* Realizarea unui sistem de progresie în joc, care să ghideze jucătorul de-a lungul scenariului.
* Utilizarea platformei Unity pentru dezvoltarea jocului
* Scripting în C# pentru a implementa logica jocului.
* Modelarea și optimizare a elementelor grafice pentru o performanță optima pe diverse dispozitive.
* Testarea și optimizarea jocului
* Evaluarea experienței utilizatorilor prin teste funcționale și de performanță.
* Îmbunătățirea elementelor grafice, mecanicilor și experienței generale pe baza feedback-ului.
* Contribuții personale și inovație
* Evidențierea abordării unice în designul și implementarea proiectului.
* Propunerea unor soluții tehnice pentru provocările întâmpinate în timpul dezvoltării.

Prin îndeplinirea acestor obiective, lucrarea își propune să ofere o perpectivă clară asupra procesului de dezvoltarea unui joc în Unity, sprijinind atât progresul personal al autorilor, cât și comunitatea de dezvoltatori prin împărtășirea experiențelor și a soluțiilor aplicate.

# 2. Descrierea jocului

## 2.1. Concept și tematică

Jocul de tip escape room creat în Unity combină elemente diverse și captivante din mai multe tematici, oferind o experiență unică jucătorilor. Acesta este structurat în jurul unui lobby central, care servește ca punct de plecare pentru toate nivelurile jocului. Lobby-ul este decorat cu elemente inspirate din tematicile nivelurilor, fiecare contribuind la atmosfera generală : Egipt Antic – elemente care evocă misterul piramidelor și civilizația egipteană. Corabia Piraților, având o tematică nautică, completată de tunuri specifice, palete care duc la corabia piraților, necesitând dexteritate pentru a complete nivelul, deoarece acestea se mișcă. Sherlock Holmes- obiecte și decoruri inspirate din epoca victoriană și univerusl detectivului. Lobby-ul are câteva elemente moderne, ascunzând iteme de tip colecționabile, care nu sunt ușor de găsit.

În mijlocul lobby-ului se află un obelisk impunător, care simbolizează progresul jucătorului. Acesta este înconjurat de cinci locuri speciale pentru flăcări,. Scopul principal al jocului este competarea celor trei niveluri, fiecare reprezentat de un cufăr ascuns. Odată ce un chest a fost găsit, o flacără este aprinsă pe obelisk. Când toate cele patru flăcări sunt activate, a cincea flacără, simbolizând victoria, se aprinde automat.

Jocul pune accent pe dezvoltarea puzzle-urilor, explorare și interacțiune cu mediul, fiecare nivel aducând un set unic de provocări. Această structură creează o experiență dinamică și captivantă, menținând jucătorii implicați pe tot parcursul jocului.

## 2.2. Public țintă

Jocul este destinat unui public larg, cu vârste de 14 ani și peste, fiind ideal pentru adolescenți și tineri care apreciază provocările logice și jocurile interactive. Grupul țintă include persoane aflate la vârsta liceului sau studenți, care își doresc să se relaxeze în timp ce își dezvoltă gândirea critică și spiritul de observație. De asemenea, jocul atrage pasionații de puzzle-uri și escape room-uri virtuale, oferindu-le posibilitatea de a explora nivele captivante și de a rezolva enigme bine gândite. Prin design-ul său accesibil, dar provocator, acesta reușește să fie potrivit și pentru jucătorii casual, care caută o experiență distractivă, dar inteligentă. Fiind conceput cu dificultate progresivă, jocul garantează o experiență captivantă atât pentru utilizatorii cu experiență în acest gen de jocuri, cât și pentru cei aflați la început de drum.

# 3 .Tehnologii utilizate

Dezvoltarea acestui joc de tip escape room a implicat utilizarea unor tehnologii și intrumente mmoderne, care au facilitate crearea unui mediu 3D interactiv și captivant. Alegerea acestor tehnologii a fost determinate de flexibilitatea și performanța lor, precum și de posibilitatea de a implementa mecanici complexe și elemente vizuale de înaltă calitate.

## 3.1. Unity și limbajul C#

Motorul de joc Unity a reprezentat principala platformă de dezvoltare datorită versatilității și resurselor extinse pe care le pune la dispoziția dezvoltatorilor. Unity oferă suport pentru :

Grafică 3D și 2D – Crearea decorurilor detaliate și modelarea elementelor din joc. Nivelurile jocului, inclusiv temele Egipt, corabia de pirați și epoca victoriană, au fost realizate folosind modele importate și personalizate în Unity.

Scripting în C# - Limbajul de programare C# a fost utilizate pentru implementarea logicii jocului, cum ar fi interacșiunile cu obiectele, rezolvarea puzzle-urilor, mișcarea jucătorului, rezolvarea puzzle-urilor și aprinderea flăcărilor pe obelisk.

Sistemul de fizică – funcțiile fizice integrate au permis crearea interacțiunilor realiste între obiecte, cum ar fi mișcarea paletelor sau manipularea cuferelor.

## 3.2. Alte resurse utilizate

Pentru a economisi timp și a îmbunătăți calitatea designului, o parte din elementele grafice și texturile utilizate au fost achiziționate din Unity Asset Store. Printre acestea se numără modele 3D pentru mobilier, obiecte antice și decoruri maritime, materiale și texturi pentru podele, pereți și elemente tematice.

Sistemul audio – pentru designul sonor, au fost utilizate efecte audio gratuite care contribuie la atmosfera fiecărui nivel. Acestea au fost gestionate în Unity prin utilizarea componetelor audio încorporate ( Unity assets ), pentru a asigura o imersiune auditivă completă.

Aceste tehnologii, utilizate împreună au permis realizarea unui produs bine optimizate, vizual captivant și tehnic funcțional, oferind o experiență plăcută utilizatorilor.

# 4. Design și implementare

## 4.1. Structura și mecanicile jocului

Structura jocului este compusă dintr-o singură scenă principală formată dintr-o cameră tip lobby, ce conține diferite elemente antice și moderne, o cameră cu design relaxant ce conține un puzzle bazat pe o tablă de șah și o ieșire în ocean unde jucătorul explorează o corabie. Mecanica principală constă în colectarea a patru cufere plasate strategic. Fiecare cufăr colectat aprinde o flacără din Obelisk-ul din lobby. După colectarea tuturor cuferelor, flacăra finală din mijlocul platformei este activată, însoțită de un mesaj de sfârșit de joc.

Jucătorul va avea urmatoarele butoane pentru control:

* W A S D – butoanele folosite de jucător pentru a se deplasa
* MOUSE – mouse-ul va fi folosit pentru a mișca camera jucătorului
* SPACE – buton folosit pentru a sări
* CTRL – buton folosit pentru ghemuire
* E – buton de interacțiune de diferite obiecte (cufere, uși, note, etc.)
* TAB – buton pentru a accesa inventarul si a observa obiectele colecționate
* ESC – buton pentru resetarea jocului (este recomandat să fie folosit la finalul jocului sau în cazul aparițiilor unor bug-uri)

## 4.2. Grafică și scripting

Jocul utilizează un stil grafic simplu, cu elemente 3D și 2D realizate în Unity. Flăcările sunt realizate cu ajutorul sistemului de particule din Unity, iar efectele lor sunt declanșate prin scripturi. Unele obiecte sunt obținute din Unity Asset Store pentru o eficiență mai bună a jocului. Elementele 2D sunt reprezentate de Canvas-urile si imaginile realizate pentru interfețele utilizatorului.

Script-urile sunt realizate în C# și sunt împărțite în 2 categorii:

Cateogoria Main (aici se află script-urile principale care mențin funcționalitatea jucătorului) :

* PlayerInteract.cs - este responsabil pentru gestionarea interacțiunii jucătorului cu obiectele din scenă, cum ar fi Keypad-uri, cufere, chei, uși folosind funcția Raycast
* PlayerLook.cs - script necesar pentru a gestiona mișcarea camerei jucătorului în funcție de mișcările mouse-ului folosind funcția ProcessLook și controlează rotația camerei pe axele X și Y, limitându-le cu Mathf.Clamp
* PlayerMotor.cs - controlează mișcarea jucătorului în joc, inclusiv deplasarea, săritura și ghemuirea. Este folosit împreună cu un CharacterController pentru a gestiona mișcările fizice.
* InputManager.cs - gestionează input-urile jucătorului utilizând noul sistem de input din Unity (Input System). Este responsabil pentru conectarea comenzilor (mișcare, săritură, privire) la componentele care controlează mișcarea și camera jucătorului.

Categoria Other (unde sunt incluse script-urile secundare ce nu influențează direct jucătorul):

* PlayerUI.cs – script ce afișează text pe interfața jucătorului.
* InventoryManager.cs – script care gestionează sistemul de inventar al jucătorului, permițând adăugarea și verificarea obiectelor colectate.
* Item.cs – gestionează logica pentru interacțiunea jucătorului cu un obiect colectabil. Este derivat dintr-o clasă abstractă Interactable, ceea ce înseamnă că oferă funcționalități specifice pentru obiectele cu care jucătorul poate interacționa.
* ItemSlot.cs - gestionează un slot din inventarul jucătorului, permițând afișarea, selecția și utilizarea obiectelor din inventar.
* Interactable.cs – script necesar pentru funcționalitatea interacțiunii cu obiectele.
* Door.cs – se aplică ușilor, cu scopul de a le putea deschide.
* DoorCenter.cs – folosind acest script, ușile se vor roti de la balamale, nu din centrul lor original.
* Key.cs – scriptul va fi atașat cheilor necesare pentru a putea deschide uși încuiate.
* Keypad.cs și Safe.cs – script-uri folosite pentru obiectele de tip tastatură numerică și seif.
* NoteInteract.cs – odată adăugat unei note se va putea interacționa cu aceasta, afișând conținutul ei pe ecran.
* TeleportationPlatform.cs – script folosit pentru o platforma așezată sub ocean. Odată ce jucătorul va cădea pe ea, acesta se va teleporta în lobby.
* FloatingObjects.cs – controlează paletele care duc spre corabie și le mișcă aleatoriu de la stânga la dreapta pentru a adăuga un nivel de dificultate.
* FlameManager.cs – acest script verifică dacă flăcările care se aprind la Obelisk sunt activate, iar dacă acest lucru se confirmă, se va aprinde o flacără finală și se va activa mesajul de sfârșit de joc.
* FootstepAudio.cs – adaugă sunet de pași atunci când jucătorul se deplasează.

# 5. Testare și optimizare

## 5.1. Metode de testare

Pentru a asigura funcționalitatea și calitatea jocului, au fost aplicate mai multe metode de testare. În primul rând, s-a verificat corectitudinea mecanicilor de joc, precum rezolvarea puzzle-urilor și progresul între nivele, pentru a garanta că totul funcționează conform așteptărilor. Jocul a fost testat pe diferite dispozitive și sisteme de operare, asigurând compatibilitatea și performanța optimă. De asemenea, au fost simulate scenarii de utilizare intensivă pentru a identifica eventuale probleme de stabilitate. Feedbackul obținut de la utilizatori care au testat jocul a fost esențial pentru a detecta erori și pentru a aduce îmbunătățiri semnificative experienței de joc. Aceste metode au permis rafinarea și optimizarea produsului final.

## 5.2. Îmbunătățiri și performanță

Pe parcursul testării, au fost identificate mai multe aspecte ce necesitau îmbunătățiri pentru a optimiza performanța jocului. În urma analizei, s-au realizat ajustări semnificative pentru a reduce timpii de încărcare și a îmbunătăți fluiditatea jocului pe diverse dispozitive. De asemenea, resursele grafice și audio au fost optimizate pentru a preveni suprasolicitarea procesorului și a memoriei. Pentru a îmbunătăți performanța, au fost implementate tehnici de compresie a fișierelor și optimizarea codului pentru a asigura o experiență mai rapidă și mai stabilă. În plus, s-au făcut ajustări în design-ul nivelelor, pentru a echilibra mai bine complexitatea cu performanța, asigurând astfel o experiență de joc mai fluidă și mai plăcută.

# 6. Concluzii

Jocul dezvoltat reprezintă o combinație de creativitate, logică și tehnologie, integrand tematici variate și mecanici de joc captivante. Acest proiect a demonstrate potențialul platformei Unity în crearea unui escape room virtual interactiv și a evidențiat importanța unei abordări bine planificate în designul și dezvoltarea unui joc video.

Pe parcursul proiectului, au fost învățate și aplicate multiple tehnici de dezvoltare, modelare 3D, scripting și optimizare, contribuind atât la realizarea unui produs functional, cât și la dezvoltarea personală a autorului.

## 6.1. Rezultate și contribuții personale

Rezultatele acestui poriect includ:

* Un joc complet funcțional, cu 3 niveluri distincte plus un lobby, fiecare cu o tematică unică (Egipt Antic, Corabie Pirați, Sherlock Holmes).
* Mecanisme captivante de joc, precum interacțiunile cu obiecte, puzzle-uri logice și progresul vizibil prin aprinderea flăcărilor pe obelisk.
* Un design vizual coherent și atractiv, bazat pe utilizarea unor asset-uri personalizate și a unei combinații de teme istorice.
* Optimizarea perdormanței jocului, asigurând o experiență fluidă pentru jucători.

Contribuțiile personale se reflectă în crearea și implementarea conceptului entral al obeliskului și a flăcărilor care simbolizează progresul, dezvoltarea logicii jocului în C# ( inclusiv interacțiunile dintre jucători și obiecte, precum și sisteme de completare a nivelurilor ), alegerea și integrarea tematicilor, asigurând o diversitate care să mențină interesul utilizatorilor pe tot parcursul jocului. De asemenea, gestionarea procesului de testare a jocului pe baza feesback-ului primit.

## 6.2. Perspective de dezvoltare

Proiectul are un potențial semnificativ de extindere și îmbunătățire, Perspectivele viitoare includ :

* Adăugarea unor niveluri noi cu tematici diferite, pentru a diversifica experiența jucătorilor și a prelungi durata jocului.
* Introducerea unui mod multiplayer, care să permită jucătorilor să colaboreze pentru a rezolva puzzle-urile, îmbunătățind aspectul social al jocului.
* Implementarea unor mecanici mai complexe, abilitatea de a combina elemente pentru a rezolva puzzle-uri.
* Îmbunătățirea graficii prin utilizarea unor asset-uri mai detaliate și a tehnologiilor moderne, cum ar fi iluminarea globală în timp real sau efecte vizuale avansate.
* Publicarea jocului pe platforme precum Steam pentru a atrage un public mai larg și pentru a primi feedback de la jucători din întreaga lume.

# 7. Bibliografie

<https://www.youtube.com/c/Brackeys>

<https://docs.unity3d.com/>

<https://assetstore.unity.com/?clickref=1011lA8BJC65&utm_source=partnerize&utm_medium=affiliate&utm_campaign=unity_affiliate&gad_source=1>

# 8. Anexă

**PlayerMotor.cs:**   
using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class PlayerMotor : MonoBehaviour

{

private CharacterController controller;

private Vector3 playerVelocity;

private bool isGrounded;

[Header("Movement Settings")]

public float speed = 5f;

public float crouchSpeed = 2.5f;

public float gravity = -9.8f;

public float jumpHeight = 3f;

[Header("Crouch Settings")]

public float standingHeight = 2f;

public float crouchingHeight = 1f;

private float currentSpeed;

void Start()

{

controller = GetComponent<CharacterController>();

currentSpeed = speed;

}

void Update()

{

isGrounded = controller.isGrounded;

if (isGrounded && playerVelocity.y < 0)

{

playerVelocity.y = -2f;

}

playerVelocity.y += gravity \* Time.deltaTime;

controller.Move(playerVelocity \* Time.deltaTime);

}

public void ProcessMove(Vector2 input)

{

Vector3 moveDirection = new Vector3(input.x, 0, input.y);

controller.Move(transform.TransformDirection(moveDirection) \* currentSpeed \* Time.deltaTime);

}

public void Jump()

{

if (isGrounded)

{

playerVelocity.y = Mathf.Sqrt(jumpHeight \* -2.0f \* gravity);

}

}

public void StartCrouch()

{

controller.height = crouchingHeight;

currentSpeed = crouchSpeed;

}

public void StopCrouch()

{

controller.height = standingHeight;

currentSpeed = speed;

}

}

**PlayerLook.cs:**   
using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class PlayerLook : MonoBehaviour

{

public Camera playerCamera;

public float mouseSensitivity = 100f;

internal Camera cam;

private float xRotation = 0f;

private Vector2 mouseInput;

public void ProcessLook(Vector2 input)

{

mouseInput = input;

}

void LateUpdate()

{

float mouseX = mouseInput.x \* mouseSensitivity \* Time.deltaTime;

float mouseY = mouseInput.y \* mouseSensitivity \* Time.deltaTime;

xRotation -= mouseY;

xRotation = Mathf.Clamp(xRotation, -90f, 90f);

playerCamera.transform.localRotation = Quaternion.Euler(xRotation, 0f, 0f);

transform.Rotate(Vector3.up \* mouseX);

}

}

**PlayerInteract.cs:**

using System.Collections;

using UnityEngine;

public class PlayerInteract : MonoBehaviour

{

private Camera cam;

[SerializeField]

private float distance = 3f;

private PlayerUI playerUI;

[SerializeField]

private LayerMask itemLayer;

[SerializeField]

private LayerMask keypadLayer;

[SerializeField]

private LayerMask doorLayer;

private InputManager inputManager;

private NoteInteract currentNote;

[SerializeField]

private InventoryManager inventoryManager;

void Start()

{

cam = GetComponent<PlayerLook>().playerCamera;

inputManager = GetComponent<InputManager>();

playerUI = GetComponent<PlayerUI>();

if (inventoryManager == null)

{

inventoryManager = FindObjectOfType<InventoryManager>();

}

}

void Update()

{

playerUI.UpdateText(string.Empty);

if (currentNote != null)

{

if (inputManager.onFoot.Interact.triggered)

{

currentNote.CloseNote();

currentNote = null;

}

return;

}

if (inputManager.onFoot.Inventory.triggered)

{

if (inventoryManager != null)

{

inventoryManager.ToggleInventory();

}

return;

}

Ray ray = new Ray(cam.transform.position, cam.transform.forward);

Debug.DrawRay(ray.origin, ray.direction \* distance);

RaycastHit hitInfo;

if (Physics.Raycast(ray, out hitInfo, distance, keypadLayer))

{

Interactable keypad = hitInfo.collider.GetComponent<Interactable>();

if (keypad != null)

{

playerUI.UpdateText(keypad.promptMessage);

if (inputManager.onFoot.Interact.triggered)

{

keypad.BaseInteract();

}

return;

}

}

if (Physics.Raycast(ray, out hitInfo, distance, itemLayer))

{

Interactable item = hitInfo.collider.GetComponent<Interactable>();

if (item != null)

{

playerUI.UpdateText(item.promptMessage);

if (inputManager.onFoot.Interact.triggered)

{

item.BaseInteract();

}

return;

}

}

if (Physics.Raycast(ray, out hitInfo, distance, doorLayer))

{

Door door = hitInfo.collider.GetComponent<Door>();

if (door != null)

{

playerUI.UpdateText("Open/Close door");

if (inputManager.onFoot.Interact.triggered)

{

bool hasKey = inventoryManager.HasKey(door.requiredKey);

door.TryOpenDoor(hasKey, inventoryManager);

}

return;

}

}

}

}

**InputManager.cs:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.InputSystem;

public class InputManager : MonoBehaviour

{

private PlayerInput playerInput;

public PlayerInput.OnFootActions onFoot;

private Vector2 movementInput;

private PlayerMotor motor;

private PlayerLook look;

void Awake()

{

playerInput = new PlayerInput();

onFoot = playerInput.OnFoot;

motor = GetComponent<PlayerMotor>();

look = GetComponent<PlayerLook>();

onFoot.Jump.performed += ctx => motor.Jump();

onFoot.Crouch.started += ctx => motor.StartCrouch();

onFoot.Crouch.canceled += ctx => motor.StopCrouch();

onFoot.Look.performed+=ctx=>look.ProcessLook(onFoot.Look.ReadValue<Vector2>());

}

void FixedUpdate()

{

motor.ProcessMove(onFoot.Movement.ReadValue<Vector2>());

}

void LateUpdate()

{

look.ProcessLook(onFoot.Look.ReadValue<Vector2>());

}

private void OnEnable()

{

onFoot.Enable();

}

private void OnDisable()

{

onFoot.Disable();

}

}

**PlayerUI.cs:**  
using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using TMPro;

public class PlayerUI : MonoBehaviour

{

[SerializeField]

private TextMeshProUGUI promptText;

void Start()

{

}

public void UpdateText(string promptMessage)

{

promptText.text = promptMessage;

}

}

InventoryManager.cs:  
using UnityEngine;

using TMPro;

using System.Collections.Generic;

public class InventoryManager : MonoBehaviour

{

public GameObject InventoryMenu;

private bool menuActivated;

public ItemSlot[] itemSlots;

private ItemSlot currentlySelectedSlot;

private List<GameObject> keys = new List<GameObject>();

private void Start()

{

menuActivated = false;

}

public void ToggleInventory()

{

menuActivated = !menuActivated;

InventoryMenu.SetActive(menuActivated);

if (menuActivated)

{

Cursor.lockState = CursorLockMode.None;

Cursor.visible = true;

Time.timeScale = 0f;

}

else

{

DeselectAllSlots();

Cursor.lockState = CursorLockMode.Locked;

Cursor.visible = false;

Time.timeScale = 1f;

}

}

public void AddItem(string itemName, Sprite itemIcon, GameObject itemObject, string itemDescription)

{

foreach (ItemSlot slot in itemSlots)

{

if (!slot.isFull)

{

slot.AddItem(itemName, itemIcon, itemObject, itemDescription);

return;

}

}

}

public void SelectedSlot(ItemSlot newSlot)

{

if (currentlySelectedSlot != null)

{

currentlySelectedSlot.Deselect();

}

currentlySelectedSlot = newSlot;

}

public void DeselectAllSlots()

{

foreach (ItemSlot slot in itemSlots)

{

slot.Deselect();

}

}

public void AddKey(GameObject key)

{

if (!keys.Contains(key))

{

keys.Add(key);

}

}

public bool HasKey(GameObject key)

{

foreach (GameObject storedKey in keys)

{

if (storedKey == key)

{

return true;

}

}

return false;

}

public void RemoveKey(GameObject key)

{

if (keys.Contains(key))

{

keys.Remove(key);

}

}

}

**Item.cs:**  
using UnityEngine;

public class Item : Interactable

{

[SerializeField]

private string itemName;

[SerializeField]

private Sprite itemIcon;

[SerializeField]

private GameObject itemObject;

[TextArea]

[SerializeField]

private string itemDescription;

[SerializeField]

private GameObject fireObject;

[Header("Teleport Settings")]

[SerializeField]

private TeleportationPlatform teleportPlatform;

[SerializeField]

private Transform newTeleportPoint;

[Header("Sound Settings")]

[SerializeField]

private AudioClip pickupSound;

[SerializeField]

private AudioClip flameSound;

private AudioSource audioSource;

private InventoryManager inventoryManager;

void Start()

{

inventoryManager = GameObject.Find("Inventory").GetComponent<InventoryManager>();

audioSource = GetComponent<AudioSource>();

if (audioSource == null)

{

audioSource = gameObject.AddComponent<AudioSource>();

audioSource.playOnAwake = false;

}

if (fireObject != null)

fireObject.SetActive(false);

}

protected override void Interact()

{

if (inventoryManager != null)

{

inventoryManager.AddItem(itemName, itemIcon, itemObject, itemDescription);

PlayPickupSound();

gameObject.SetActive(false);

if (fireObject != null)

{

fireObject.SetActive(true);

PlayFlameSound(fireObject);

}

if (itemObject != null && itemObject.CompareTag("Key"))

{

inventoryManager.AddKey(itemObject);

}

if (CompareTag("Chest"))

{

FlameManager flameManager = FindObjectOfType<FlameManager>();

if (flameManager != null)

{

flameManager.CollectChest();

}

}

if (teleportPlatform != null && newTeleportPoint != null)

teleportPlatform.UpdateTeleportPoint(newTeleportPoint);

}

}

private void PlayPickupSound()

{

if (audioSource != null && pickupSound != null)

audioSource.PlayOneShot(pickupSound);

}

private void PlayFlameSound(GameObject fire)

{

AudioSource flameAudioSource = fire.GetComponent<AudioSource>();

if (flameAudioSource != null && flameSound != null)

flameAudioSource.PlayOneShot(flameSound);

}

}

**ItemSlot.cs:**

using UnityEngine;

using UnityEngine.EventSystems;

using UnityEngine.UI;

using TMPro;

public class ItemSlot : MonoBehaviour, IPointerClickHandler

{

//===ITEM DATA===//

public string itemName;

public Sprite itemSprite;

public GameObject itemObject;

public bool isFull;

public string itemDescription;

//===ITEM SLOT===//

[SerializeField]

private Image itemImage;

//===ITEM DESCRIPTION SLOT===//

public TMP\_Text ItemDescriptionName;

public TMP\_Text ItemDescriptionText;

public GameObject selectedShader;

public bool thisItemSelected;

private InventoryManager inventoryManager;

private void Start()

{

inventoryManager = GameObject.Find("Inventory").GetComponent<InventoryManager>();

Deselect();

}

public void AddItem(string itemName, Sprite itemIcon, GameObject itemObject, string itemDescription)

{

this.itemName = itemName;

this.itemSprite = itemIcon;

this.itemObject = itemObject;

this.itemDescription = itemDescription;

isFull = true;

if (itemImage != null)

{

itemImage.sprite = itemIcon;

itemImage.enabled = true;

}

}

public void UseItem()

{

if (itemObject != null)

{

itemObject.SetActive(true);

}

ClearSlot();

}

public void ClearSlot()

{

itemName = null;

itemSprite = null;

itemObject = null;

isFull = false;

if (itemImage != null)

{

itemImage.sprite = null;

itemImage.enabled = false;

}

}

public void OnPointerClick(PointerEventData eventData)

{

if (eventData.button == PointerEventData.InputButton.Left)

{

OnLeftClick();

}

}

public void OnLeftClick()

{

inventoryManager.SelectedSlot(this);

Select();

ItemDescriptionName.text = itemName;

ItemDescriptionText.text = itemDescription;

}

public void Select()

{

selectedShader.SetActive(true);

thisItemSelected = true;

}

public void Deselect()

{

selectedShader.SetActive(false);

thisItemSelected = false;

}

}

Door.cs:  
using UnityEngine;

public class Door : MonoBehaviour

{

[SerializeField]

public GameObject requiredKey;

private Animator animator;

private bool isUnlocked = false;

private void Start()

{

animator = GetComponent<Animator>();

}

public void TryOpenDoor(bool hasKey, InventoryManager inventory)

{

if (isUnlocked)

{

ToggleDoor();

}

else if (hasKey)

{

isUnlocked = true;

inventory.RemoveKey(requiredKey);

ToggleDoor();

}

}

private void ToggleDoor()

{

bool isOpen = animator.GetBool("IsOpen");

animator.SetBool("IsOpen", !isOpen);

}

}

**Key.cs:**

using UnityEngine;

public class Key : MonoBehaviour

{

private void OnTriggerEnter(Collider other)

{

if (other.CompareTag("Player"))

{

InventoryManager inventory = other.GetComponent<InventoryManager>();

if (inventory != null)

{

inventory.AddKey(this.gameObject);

}

Destroy(gameObject);

}

}

}

**Keypad.cs:**

using UnityEngine;

using TMPro;

public class Keypad : Interactable

{

[Header("Keypad Settings")]

public GameObject keypadUI;

public TextMeshProUGUI codeDisplay;

public string correctCode;

[Header("Sound Settings")]

public AudioClip buttonPressSound;

public AudioClip correctCodeSound;

public AudioClip incorrectCodeSound;

public AudioClip safeOpenSound;

private string inputCode = "";

private bool isKeypadActive = false;

private bool isUnlocked = false;

private AudioSource audioSource;

[Header("Safe Settings")]

public Animator safeAnimator;

void Start()

{

if (keypadUI != null)

{

keypadUI.SetActive(false);

}

audioSource = GetComponent<AudioSource>();

if (audioSource == null)

{

audioSource = gameObject.AddComponent<AudioSource>();

}

}

protected override void Interact()

{

if (isKeypadActive)

{

CloseKeypad();

}

else if (!isUnlocked && keypadUI != null)

{

OpenKeypad();

}

}

private void OpenKeypad()

{

isKeypadActive = true;

keypadUI.SetActive(true);

Cursor.lockState = CursorLockMode.None;

Cursor.visible = true;

Time.timeScale = 0f;

}

public void CloseKeypad()

{

if (isKeypadActive && keypadUI != null)

{

isKeypadActive = false;

keypadUI.SetActive(false);

Cursor.lockState = CursorLockMode.Locked;

Cursor.visible = false;

Time.timeScale = 1f;

}

}

public void AddDigit(string digit)

{

if (inputCode.Length < 4)

{

inputCode += digit;

codeDisplay.text = inputCode;

PlaySound(buttonPressSound);

}

if (inputCode.Length == 4)

{

CheckCode();

}

}

private void CheckCode()

{

if (inputCode == correctCode)

{

PlaySound(correctCodeSound);

OpenSafe();

CloseKeypad();

}

else

{

PlaySound(incorrectCodeSound);

ResetCode();

}

}

private void ResetCode()

{

inputCode = "";

codeDisplay.text = "";

}

private void PlaySound(AudioClip clip)

{

if (audioSource != null && clip != null)

{

audioSource.PlayOneShot(clip);

}

}

private void OpenSafe()

{

if (!isUnlocked)

{

isUnlocked = true;

if (safeAnimator != null)

{

safeAnimator.SetTrigger("Open");

}

PlaySound(safeOpenSound);

}

}

}

**NoteInteract.cs:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class NoteInteract : Interactable

{

[SerializeField]

private Image noteImage;

private bool isNoteVisible = false;

protected override void Interact()

{

if (noteImage != null)

{

isNoteVisible = !isNoteVisible;

noteImage.gameObject.SetActive(isNoteVisible);

}

}

public void CloseNote()

{

if (isNoteVisible != false)

{

isNoteVisible = false;

noteImage.gameObject.SetActive(false);

}

}

}

**Safe.cs:**

using UnityEngine;

public class Safe : MonoBehaviour

{

private bool isOpen = false;

public void OpenSafe()

{

if (!isOpen)

{

isOpen = true;

Debug.Log("Seiful a fost deschis!");

}

}

}

FlameManager.cs:

using System.Collections;

using UnityEngine;

public class FlameManager : MonoBehaviour

{

[Header("Flăcări")]

public GameObject[] flames;

public GameObject finalFlame;

[Header("UI Settings")]

public GameObject thankYouText;

public float displayTime = 5f;

private int collectedChests = 0;

void Start()

{

foreach (GameObject flame in flames)

{

if (flame != null)

flame.SetActive(false);

}

if (finalFlame != null)

finalFlame.SetActive(false);

if (thankYouText != null)

thankYouText.SetActive(false);

}

public void CollectChest()

{

collectedChests++;

if (collectedChests <= flames.Length && flames[collectedChests - 1] != null)

{

flames[collectedChests - 1].SetActive(true);

}

if (collectedChests == flames.Length)

{

ActivateFinalFlame();

}

}

private void ActivateFinalFlame()

{

if (finalFlame != null)

{

finalFlame.SetActive(true);

}

StartCoroutine(DisplayThankYouText());

}

private IEnumerator DisplayThankYouText()

{

if (thankYouText != null)

{

thankYouText.SetActive(true);

yield return new WaitForSeconds(displayTime);

thankYouText.SetActive(false);

}

}

}

FloatingObjects.cs:

using UnityEngine;

public class FloatingPlatform : MonoBehaviour

{

[Header("Random Movement Settings")]

public float minMoveDistance = 2f;

public float maxMoveDistance = 8f;

public float minMoveSpeed = 1f;

public float maxMoveSpeed = 4f;

private float moveDistance;

private float moveSpeed;

private Vector3 startPosition;

private Vector3 previousPosition;

private void Start()

{

startPosition = transform.position;

previousPosition = startPosition;

moveDistance = Random.Range(minMoveDistance, maxMoveDistance);

moveSpeed = Random.Range(minMoveSpeed, maxMoveSpeed);

}

private void Update()

{

float offset = Mathf.PingPong(Time.time \* moveSpeed, moveDistance) - (moveDistance / 2);

transform.position = new Vector3(transform.position.x, transform.position.y, startPosition.z + offset);

previousPosition = transform.position;

}

public Vector3 GetDeltaPosition()

{

return transform.position - previousPosition;

}

}

**GameRestart.cs:**

using UnityEngine;

using UnityEngine.SceneManagement;

public class GameRestart : MonoBehaviour

{

void Update()

{

if (Input.GetKeyDown(KeyCode.Escape))

{

RestartGame();

}

}

private void RestartGame()

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);

}

}

**DoorCenter.cs:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class DoorCenter : MonoBehaviour

{

void Start()

{

}

void Update()

{

transform.Rotate(Vector3.up \* Time.deltaTime \* 50);

}

}

**TeleportationPlatform.cs:**  
using UnityEngine;

public class TeleportationPlatform : MonoBehaviour

{

[Header("Teleport Settings")]

public Transform teleportPoint;

private void OnTriggerEnter(Collider other)

{

if (other.CompareTag("Player"))

{

TeleportPlayer(other.gameObject);

}

}

private void TeleportPlayer(GameObject player)

{

if (teleportPoint != null)

{

CharacterController controller = player.GetComponent<CharacterController>();

if (controller != null)

{

controller.enabled = false;

player.transform.position = teleportPoint.position;

player.transform.rotation = teleportPoint.rotation;

controller.enabled = true;

}

else

{

player.transform.position = teleportPoint.position;

player.transform.rotation = teleportPoint.rotation;

}

}

}

public void UpdateTeleportPoint(Transform newTeleportPoint)

{

teleportPoint = newTeleportPoint;

}

}

**Interactable.cs:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public abstract class Interactable : MonoBehaviour

{

public string promptMessage;

public virtual void BaseInteract()

{

Interact();

}

protected virtual void Interact()

{

}

}

**FootstepAudio.cs:**

using UnityEngine;

public class FootstepAudio : MonoBehaviour

{

[Header("Footstep Settings")]

public AudioClip footstepSound;

private AudioSource audioSource;

private CharacterController characterController;

void Start()

{

audioSource = GetComponent<AudioSource>();

if (audioSource == null)

{

audioSource = gameObject.AddComponent<AudioSource>();

}

characterController = GetComponent<CharacterController>();

audioSource.clip = footstepSound;

audioSource.loop = true;

audioSource.playOnAwake = false;

audioSource.volume = 0f;

}

void Update()

{

if (characterController != null && characterController.isGrounded && characterController.velocity.magnitude > 0.1f)

{

if (!audioSource.isPlaying)

{

audioSource.Play();

}

}

else

{

if (audioSource.isPlaying)

{

audioSource.Stop();

}

}

}

}