### **Sinteza proiectului**

## 1. Prezentare Generală a Jocului

Jocul nostru 3D horror, "Asylum Escape", este conceput pentru a oferi jucătorilor o experiență intensă, în care aceștia trebuie să exploreze o clădire întunecată și bântuită, rezolvând puzzle-uri și evitând o entitate malefică. Obiectivul principal al jocului este să scape din clădire înainte ca entitatea să îi captureze.

## 2.Obiective inițiale

* Mediu 3D interactiv.
* NavMesh pentru navigația entității.
* Puzzle-uri dinamice cu dificultate progresivă.
* Atmosferă tensionată generată de efecte vizuale și audio.

## 3. Caracteristici Cheie

* **Explorare și Puzzle-uri:** Jucătorul trebuie să navigheze într-un mediu 3D complex pentru a găsi obiecte și rezolva puzzle-uri care îi permit accesul în zone noi.
* **Entitate Malefică:** Demonul care vânează jucătorul utilizează algoritmi de pathfinding avansați, devenind mai rapid și mai periculos odată cu progresul în joc.
* **Atmosferă Tensionată:** Luminile dinamice, efectele sonore și evenimentele declanșate mențin suspansul.
* **Dificultate Dinamică:** Pe măsură ce puzzle-urile devin mai complexe, entitatea devine mai greu de evitat.

## 4. Tehnologii Utilizate

* **Unity:** Motorul principal de dezvoltare datorită capabilităților sale pentru jocuri 3D.
* **C#:** Limbajul de scripting utilizat pentru logica jocului, AI și gestionarea evenimentelor.
* **Blender:** Crearea și animarea modelelor 3D.
* **FMOD:** Gestionarea sunetelor de ambianță și sincronizarea audio cu evenimentele din joc.
* **Shader Graph (Unity):** Crearea de efecte vizuale speciale pentru creșterea tensiunii vizuale.

## 5. Modul de implementare

* **Mediu 3D interactiv:** Jocul are locația într-un azil abandonat cu trei niveluri(subsol, parter și etaj) și este descarcat ca asset de pe internet. Fiecare cameră este mobilată și decorată pentru a crea o atmosferă tensionată și sinistră. Unele camere sunt capcane care forțează jucătorul să rezolve puzzle-uri pentru a ieși și a găsi anumite obiecte în ele.

**Implementare uși**: avem un script special atașat ușilor prin care jucatoul daca apasa tasta E deschida usa, in cazul in care o ușa este formată din doua bucati, ele se deschid amandoua si mereu în sensul opus jucătorului.

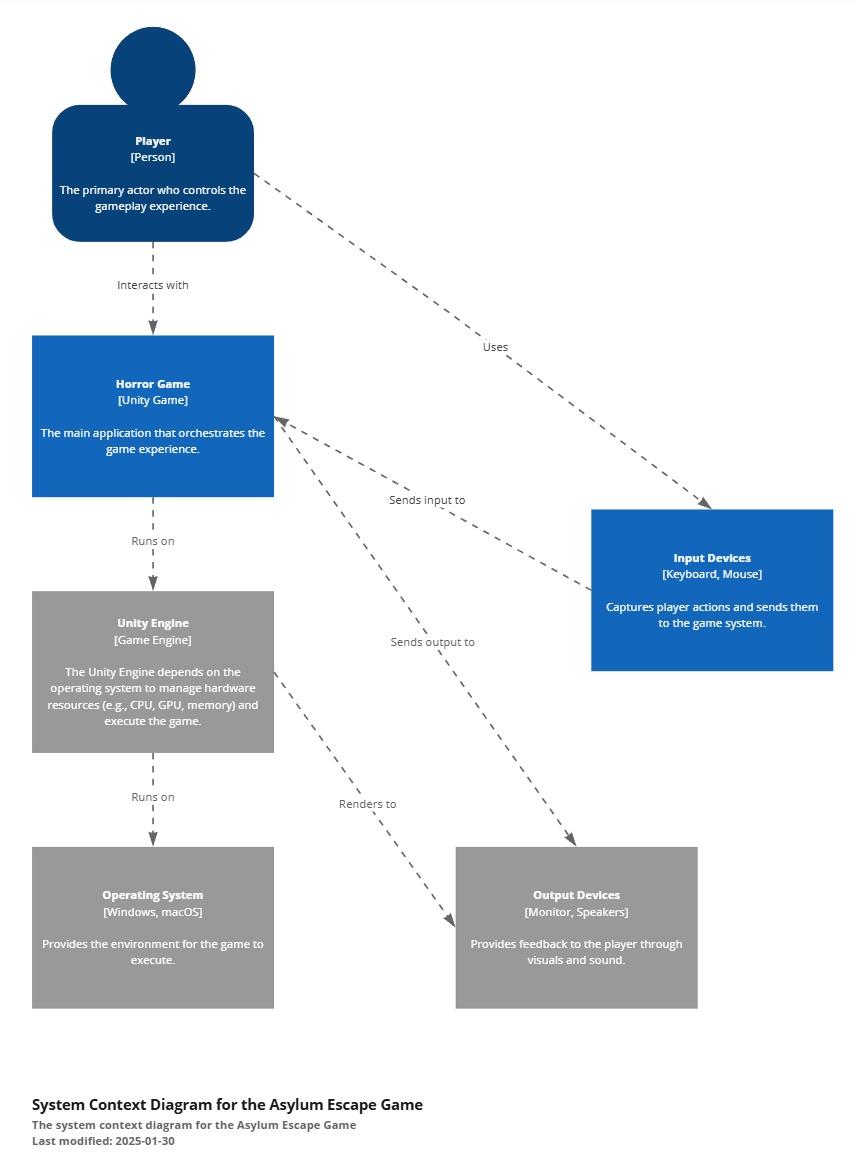
**Implementare keypad-uri**: avem un script atașat acestui asset prin care facem conexiunea cu tastatura și îi spunem ce cod are are acesta. Mai exista doua triggere, unul pentru keypad care este pus in fata keypad-ului si poti sa introduci codul doar dacă te afli pe acesta și unul în fața ușii, după ce intri în camera, pentru a se închide ușa în spatele tău.

**Implementarea inventarului:** avem un script atașat acestui obiect în care sunt implementate funcțiile de adaugare in inventar (tasta E) și de selectarea unui obiect din acesta. Dacă avem obiectul “cheie aurie” în inventar, putem sa deschidem ușa de la intrarea în azil și așa am castigat jocul. Obiectele se găsesc prin camere.

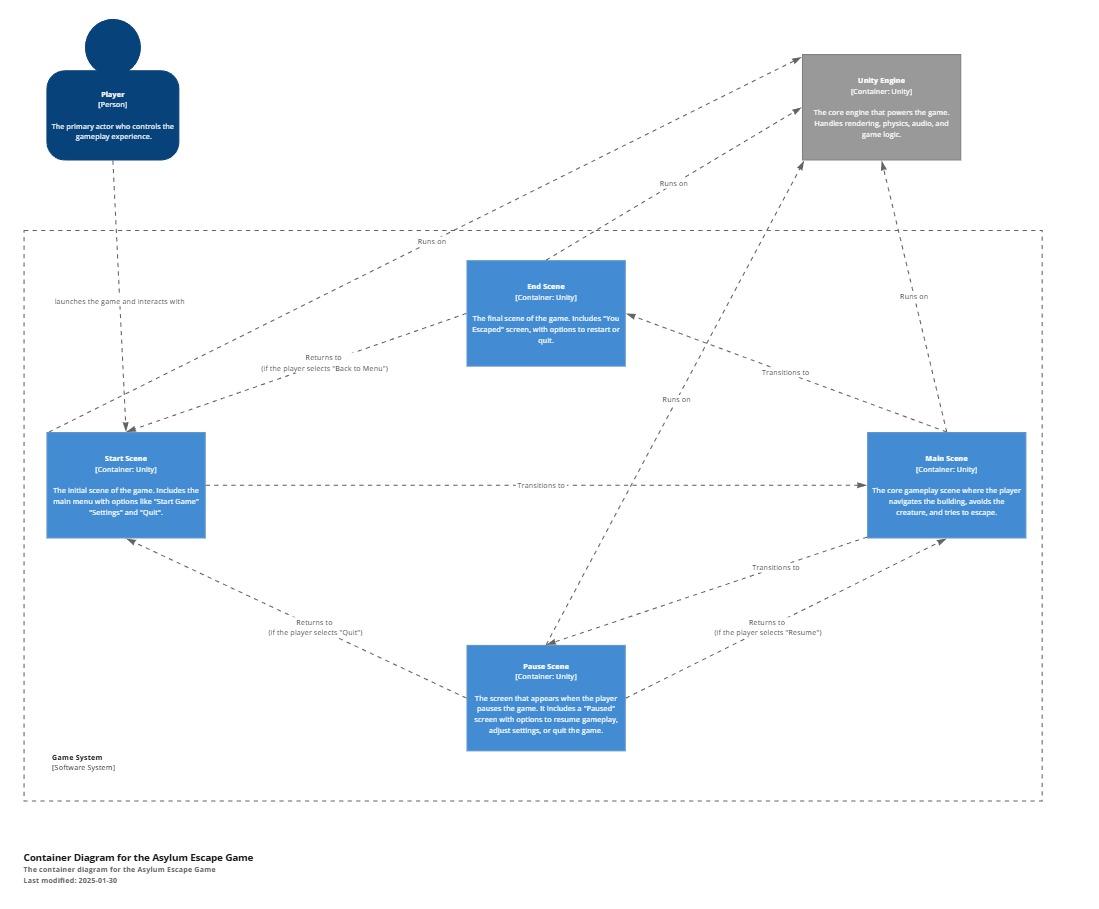
* **NavMesh pentru entitate:** s-a generat un NavMesh pentru întreaga clădire (subsol, parter, etaj), acoperind zonele navigabile. Obstacolele statice (pereți, mobilă) au fost marcate ca **NavMesh Obstacles** pentru a preveni trecerea entității prin ele.
* **Puzzle-uri dinamice:** fiecare puzzle se afla într-o camera. În camera este pus un keypad, cu toate functionalitatile descrise mai sus, și sunt niște ecuații/cifre pe pereți prin care jucătorul deschide ușa. În camera mai pot exista și obiecte pe care trebuie sa le ia ca să-și îndeplinească scopul final.
* **Atmosferă tensionată:** atmosfera tensionată este dată de aspectul clădirii, monstrul care urmărește jucătorul și dacă îl atinge de doua ori, jucătorul pierde, sunetul de fundal și imaginea întunecată.
* **AI adaptiv pentru entitate:** folosește componentele Unity **NavMeshAgent** pentru a naviga spațiul de joc. Aceasta poate ajusta ruta în timp real în funcție de poziția jucătorului. Dacă jucătorul a trecut la alt nivel și creatura nu, se transporta singura la nivelul jucătorului.

## 6. Diagrame

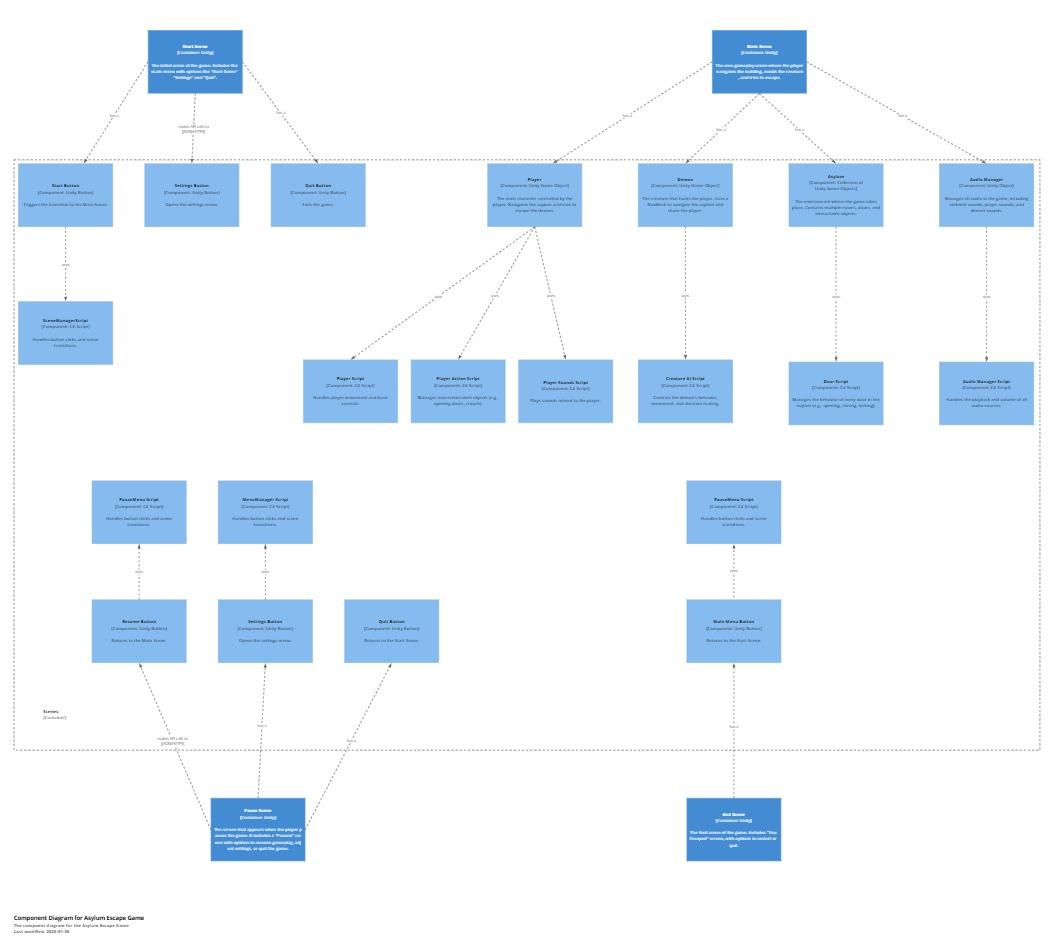
**a)System Diagram**



**b)Container Diagram**



**c)Components Diagram**



Pentru a vizualiza diagramele mai detaliat și la rezolutie mai buna puteti intra pe linkul:

<https://miro.com/app/board/uXjVLm7WQWw=/?share_link_id=597628968280>

## [7.CI/CD](http://7.ci/CD)

**a) Descrierea Environment-urilor folosite**

1. **Development (Dev)**
   * Dezvoltarea s-a realizat pe branch-uri separate pentru fiecare feature sau bug fix.
   * Codul a fost testat local înainte de a fi pus în repository.
   * Testarea inițială a fost realizată pe aceste branch-uri pentru a verifica dacă funcționalitățile adăugate funcționează corect.
2. **Staging (Testare)**
   * După ce un feature era considerat finalizat, se crea un pull request pentru a fi revizuit de echipă.
   * Se făceau testele finale pe branch-ul respectiv pentru a ne asigura că nu există erori majore.
   * Dacă testele erau trecute și codul era aprobat, se mergea mai departe.
3. **Production (Prod)**
   * Odată ce toate testele erau trecute și feature-ul era gata, se făcea un merge în branch-ul main.
   * Acesta era considerat versiunea finală, stabilă a codului.

**b)Configurații Specifice**

* Am folosit GitHub pentru controlul versiunilor și colaborare.
* Am utilizat pull requests pentru revizuirea codului înainte de a ajunge în main.
* Testarea s-a făcut manual, iar doar codul stabil era integrat în main.
* Am menținut o regulă conform căreia doar codul verificat și aprobat ajunge în main, ceea ce echivalează cu un mediu de producție.

## 8. Cerințe Non-Funcționale și Soluții

### **1) Performanță**

**Optimizarea cadrelor pe secundă (FPS)**

* **Gestionarea eficientă a luminii:**
  + S-a utilizat un sistem de iluminare statică și dinamică combinată, evitând iluminarea complet dinamică pentru a reduce consumul de resurse.
  + Shader Graph a fost folosit pentru efecte vizuale optimizate.
* **Încărcarea eficientă a resurselor:**
  + Modele 3D optimizate în Blender înainte de importul în Unity.
  + Nivelurile au fost împărțite în scene distincte pentru a reduce încărcarea inițială a jocului.
* **Culling:** Obiectele care nu sunt în câmpul vizual al camerei sunt excluse de la randare.

### **2) Extensibilitate**

**Structură modulară**

* Puzzle-urile și interacțiunile sunt gestionate prin scripturi separate, fiecare cu propriile componente logice.
* Adăugarea de noi puzzle-uri este simplă datorită utilizării unei arhitecturi bazate pe **event-driven design**.
* Obiectele de tip Scriptable Object facilitează partajarea de date între componente și gestionarea stărilor de joc.

### **3) Portabilitate**

**Compatibilitate cu diferite rezoluții și specificații hardware:**

* Unity permite redimensionarea și scalarea automată a UI-ului pentru a se adapta la diverse rezoluții.
* Testarea jocului a fost efectuată pe mai multe configurații hardware pentru a asigura performanța stabilă.
* Utilizarea setărilor grafice ajustabile pentru a permite jucătorilor să reducă detaliile vizuale în funcție de capacitatea sistemului.

### **4) Securitate**

**Prevenirea exploatării mecanicilor de joc:**

* Verificarea stărilor puzzle-urilor și a interacțiunilor jucătorului pentru a preveni activarea incorectă a evenimentelor.
* Restricționarea accesului la anumite scene până la finalizarea corectă a puzzle-urilor.
* Testarea logicii pentru a evita bypass-uri de tip glitch în gameplay.

### **5) UX (User Experience)**

**Control intuitiv și feedback audio-vizual clar:**

* Schema de control a fost simplă și familiară pentru jucători (WASD, E pentru interacțiune).
* Feedback audio prin FMOD pentru confirmarea acțiunilor (de exemplu, sunet specific la rezolvarea unui puzzle).
* Efecte vizuale clare pentru elementele interactive (lumini sau highlight-uri vizuale).
* Răspuns rapid la interacțiuni datorită unui sistem eficient de gestionare a input-urilor.

**6) Gestionarea blocării demonului**

În timpul testării jocului, am observat că demonul se putea bloca în anumite situații, de exemplu, atunci când detecta jucătorul, dar acesta se ascundea brusc în spatele unei uși.

Pentru a asigura o experiență de joc fluidă și pentru a preveni situațiile în care demonul rămânea nemișcat, am implementat următoarele mecanisme:

* **Patrulare aleatorie** – Atunci când jucătorul nu mai este în câmpul vizual al demonului, acesta începe să patruleze aleatoriu prin clădire, evitând astfel blocarea într-o zonă fixă.
* **Respawn după stagnare prelungită** – Dacă demonul rămâne în același loc mai mult de 5 secunde, acesta este respawnat fie la etajul 1, fie la parter, asigurând astfel o mișcare constantă prin spațiul de joc.
* **Monitorizarea distanței parcurse** – Dacă demonul nu se deplasează pe o distanță minimă într-un interval de 30 de secunde, se aplică același mecanism de respaun, prevenind blocarea în colțuri, camere sau alte obstacole.

Aceste măsuri contribuie la un gameplay mai dinamic și elimină situațiile în care demonul devine inactiv, menținând astfel tensiunea și imprevizibilitatea în joc.

## 9. Comparare Obiective vs. Realizări

| **Obiectiv Inițial** | **Realizări** |
| --- | --- |
| Mediu 3D interactiv | Da, structurat pe trei niveluri cu camere interactive |
| NavMesh pentru entitate | Implementat și optimizat |
| Puzzle-uri dinamice | Da, cu dificultate progresivă și integrare în gameplay |
| Atmosferă tensionată | Da, cu shadere dinamice și efecte audio sincronizate |
| AI adaptiv pentru entitate | Implementat cu patrulare și urmărire |

## 10. QA

#### **Obiectivele Testării:**

1. **Artefactele care trebuie testate:**
   * **Gameplay-ul principal:** verificarea logicii de joc (cheile, puzzle-urile, mecanica demonului).
   * **Interfața grafică:** meniuri, HUD (afişarea cheilor găsite).
   * **Sunetul:** efectele audio contextuale (pașii demonului, sunete ambientale).
   * **Performanța:** testarea timpilor de încărcare, lag, și optimizarea scenei.
   * **Compatibilitatea:** rularea pe mai multe dispozitive/platforme.
2. **Nivelul testării:**
   * **Nivel funcțional:** verificarea mecanicilor (ex.: deschiderea ușilor, rezolvarea puzzle-urilor).
   * **Nivel integrat:** verificarea interacțiunilor dintre diferitele elemente (ex.: keypad-ul deblochează ușa, lanterna se încarcă cu baterii).
   * **Nivel experiențial:** cât de intuitiv și captivant este jocul pentru utilizator.

#### **Procesul Testării:**

Testarea a fost aplicată în diferite momente ale **Ciclului de Dezvoltare al Jocului (SDLC)**:

1. **După implementarea funcționalităților majore:**
   * Testare manuală a mecanicilor de bază (săritura, aplecarea, deschiderea ușilor, colectarea obiectelor, utilizarea inventarului).
2. **După integrarea puzzle-urilor:**
   * Verificare dacă puzzle-urile sunt logic conectate cu progresia jocului. (ex. să nu poți progresa din greșeală mai repede rezolvând puzzle-uri într-o ordine anume)
3. **Înainte de lansare:**
   * Testare finală a întregii experiențe de gameplay, incluzând bug-uri, blocaje și experiența utilizatorului.

#### **Metodele Testării:**

1. **Testare manuală (prin gameplay):**
   * **Cum a fost folosită:** Dezvoltatorii au jucat jocul pentru a identifica bug-uri, comportamente neașteptate și pentru a evalua experiența de joc după fiecare schimbare adăugată.
   * **Justificare:** Testarea manuală este ideală pentru jocuri, deoarece implică evaluarea elementelor subiective precum dificultatea puzzle-urilor, frica indusă de demon și satisfacția generală a experienței oferite de joc.

#### **Rezultatele Testării**

1. **Observații principale:**
   * Mecanica de urmărire a demonului funcționează corect, dar uneori traseul acestuia este prea previzibil.
   * Puzzle-urile sunt funcționale, însă un puzzle a fost considerat prea dificil de majoritatea testerilor.
   * Performanța este acceptabilă pe dispozitive moderne, dar apar unele sacadări pe dispozitive mai vechi.
   * Sunetele ambientale contribuie bine la atmosferă, dar uneori nu se declanșează corect (ex.: uneori pașii nu se aud, atunci când jucătorul umblă pe suprafețe care nu sunt menite a fi utilizate dar care pot fi totuși accesate).
2. **Îmbunătățiri propuse:**
   * Ajustarea dificultății puzzle-urilor în funcție de feedback-ul jucătorilor.
   * Optimizarea performanței pentru dispozitive mai vechi.
   * Corectarea problemelor legate de declanșarea efectelor audio.

## 11. Security Analysis

### **1. Principalele Riscuri de Securitate**

#### **a) Exploatarea mecanicilor de joc:** jucătorul poate găsi metode de a ocoli puzzle-urile sau de a accesa zone restricționate.

#### **b) Manipularea datelor din inventar:** posibilitatea de a modifica sau injecta obiecte direct în inventar pentru a câștiga jocul fără a urma regulile normale.

#### **c) Integritatea Scenei și Stărilor Jocului:** forțarea schimbării scenelor fără a finaliza corect puzzle-urile.

#### **d) Performanță și stabilitate:** suprasolicitarea motorului de joc prin exploatarea anumitor evenimente (de exemplu, crearea necontrolată a instanțelor de entități).

### **2. Tactici pentru Adresarea Riscurilor de Securitate**

#### **a) Verificarea Interacțiunilor Corecte cu Puzzle-uri:** fiecare puzzle are o verificare strictă a soluției corecte înainte de a permite progresul și trigger-ele sunt plasate inteligent pentru a preveni accesul la zone restricționate fără rezolvarea puzzle-urilor.

#### **b) Gestionarea Sigură a Inventarului:** inventarul este protejat prin metode care verifică integritatea obiectelor adăugate și obiectele pot fi adăugate doar prin interacțiuni valide definite în joc.

#### **c) Controlul Scenei și Stării Jocului:** scenele sunt blocate și deblocate numai după evenimente validate și se verifică dacă toate cerințele au fost îndeplinite înainte de tranziția către următoarea scenă.

#### **d) Optimizarea Performanței:** limitarea numărului de instanțe pentru entitatea malefică si curățarea memoriei după evenimente complexe și limitarea evenimentelor de generare dinamică.

#### **e) Testare și Validare Continuă:** testarea sistematică a glitch-urilor care pot permite exploatări si validarea codului și logica de gameplay prin scenarii de testare automatizată.

## 12. Concluzii și Îmbunătățiri Viitoare

#### **Concluzii**

Proiectul *Asylum Escape* a reprezentat o experiență complexă și provocatoare, implicând implementarea unor mecanici avansate de joc și tehnologii moderne. Dezvoltarea jocului a permis explorarea unor concepte esențiale din domeniul dezvoltării software pentru jocuri, inclusiv inteligența artificială, optimizarea performanței și crearea unei atmosfere captivante.

Principalele realizări includ:

* Crearea unui mediu 3D captivant și bine structurat, care susține explorarea și rezolvarea puzzle-urilor.
* Implementarea unui AI adaptiv pentru entitatea malefică, capabilă să navigheze spațiul de joc folosind NavMesh și să reacționeze la acțiunile jucătorului.
* Integrarea unor puzzle-uri dinamice care adaugă complexitate și provocări gameplay-ului.
* Folosirea efectelor audio și vizuale pentru a crea o atmosferă tensionată și captivantă.
* Optimizarea performanței jocului pentru a asigura un framerate stabil.

Proiectul a demonstrat importanța unei arhitecturi modulare și a unui workflow bine organizat pentru gestionarea dezvoltării unui joc complet funcțional.

#### **Îmbunătățiri Viitoare**

Deși proiectul a atins majoritatea obiectivelor propuse, există numeroase aspecte care pot fi dezvoltate și perfecționate:

1. **Extinderea Gameplay-ului:**
   * Crearea unor niveluri suplimentare pentru a extinde durata jocului.
   * Adăugarea unor noi tipuri de puzzle-uri mai complexe, cu mecanici inovatoare.
2. **Experiență User-Friendly:**
   * Crearea unui tutorial interactiv pentru a ajuta jucătorii să se familiarizeze rapid cu mecanicile jocului.
   * Adăugarea de opțiuni suplimentare pentru setări grafice și audio.

Această bază solidă oferă un potențial mare pentru dezvoltarea continuă a jocului *Asylum Escape*, transformându-l într-o experiență chiar mai captivantă și memorabilă.