UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DOMENIUL: Calculatoare și Tehnologia Informației
SPECIALIZAREA: Tehnologia Informației

LUCRARE DE DIPLOMĂ

Coordonator științific:
prof.univ.dr.ing. Vasile Ion Manta

Absolvent:
Grama George

Iași, 2024

UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DOMENIUL: Calculatoare și Tehnologia Informației
SPECIALIZAREA: Tehnologia Informației

Joc serios despre înțelegerea algoritmilor

LUCRARE DE DIPLOMĂ

Coordonator științific:
prof.univ.dr.ing. Vasile Ion Manta

Absolvent:
Grama George

Iași, 2024

DECLARAȚIE DE ASUMARE A AUTENTICITĂȚII PROIECTULUI DE DIPLOMĂ

Subsemnatul GRAMA GEORGE,
legitimat cu CI seria ZC nr. 735331, CNP 5010927226777
autorul lucrării JOC SERIOS DESPRE ÎNȚELEGEREA ALGORITMILOR

elaborată în vederea susținerii examenului de finalizare a studiilor de licență, programul de studii CTI-SPECIALIZAREA TEHNOLOGIA INFORMAȚIEI organizat de către Facultatea de Automatică şi Calculatoare din cadrul Universității Tehnice „Gheorghe Asachi" din Iaşi, sesiunea IUNIE-IULIE a anului universitar 2023-2024, luând în considerare conținutul Art. 34 din Codul de etică universitară al Universității Tehnice „Gheorghe Asachi" din Iaşi (Manualul Procedurilor, UTI.POM.02 - Funcționarea Comisiei de etică universitară), declar pe proprie răspundere, că această lucrare este rezultatul propriei activități intelectuale, nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române (legea 8/1996) şi a convențiilor internaționale privind drepturile de autor.

Data
30.06.2024

Semnătura

Cuprins

Joc serios despre înțelegerea algoritmilor

Grama George

Rezumat

În contextul actual al industriei IT, creșterea popularității jocurilor video pentru calculator, telefoane mobile, etc., a determinat o dorință și o nevoie crescute de integrare a acestora în procesul tradițional de învățare, considerat de mulți demodat și care nu se aplică nevoilor tuturor oamenilor. Pentru rezolvarea acestei probleme, a apărut industria jocurilor serioase.

Jocurile serioase sunt proiectate nu doar pentru divertisment, ci și pentru a educa, a antrena sau a rezolva probleme specifice. Acestea combină elementele captivante ale jocurilor video cu obiective educaționale, oferind o metodă interactivă și eficientă de învățare. Spre deosebire de metodele tradiționale de educație, jocurile serioase oferă experiențe dinamice și motivante care pot să îmbunătățească semnificativ înțelegerea și retenția informațiilor.

Soluția proiectată în cadrul acestei lucrări are ca scop depășirea constrângerilor impuse de abordările tradiționale, care pot fi percepute ca fiind statice și plictisitoare. Produsele educative actuale adesea trec cu vederea beneficiile semnificative ale combinării teoriei cu practica într-un mod interactiv și captivant.

Lucrarea de diplomă își propune dezvoltarea unui joc educativ destinat programatorilor începători și persoanelor care doresc să învețe informatică, dar li se pare prea complicat. Jocul integrează explicații teoretice detaliate și oferă utilizatorilor exerciții practice și exerciții de testare a cunoștințelor prin intermediul unor nivele de tip puzzle interactive, astfel oferind o experiență de învățare unică, distractivă și eficientă. În cadrul soluției dezvoltate, propunem utilizarea unui motor de jocuri precum Unity pentru construirea acestei aplicații.

Introducere

În ultimii ani, odată cu avansarea tehnologiei informațiilor, oamenii au început să integreze în viața cotidiană tot mai multe metode pentru ușurarea traiului acestora. Astfel, algoritmii au început să joace un rol crucial în aproape toate domeniile. De la motoarele de căutare și rețelele sociale, până la comerțul electronic și sistemele de navigație, algoritmii stau la baza tehnologiilor care ne fac viața mai ușoară și mai eficientă. Din acest motiv, viitorul nostru depinde de noile generații de ingineri, programatori, etc. Astfel, pentru a putea înțelege acești algoritmi complexi și pentru a putea dezvolta noi tehnologii, este crucial ca viitorii programatori să înțeleagă modul de funcționare al algoritmilor simpli și să își creeze o fundație solidă pe baza căreia să clădească viitoarele cunoștiințe.

În era digitală modernă, educația tradițională evoluează rapid, integrând tehnologii inovatoare pentru a facilita învățarea eficientă și atractivă. Un domeniu emergent în acest context este cel al jocurilor serioase (termenul in engleză-"serious game"), care combină divertismentul cu scopuri educative pentru a crea experiențe de învățare captivante. Acest joc serios se concentrează pe înțelegerea algoritmilor, oferind utilizatorilor o modalitate interactivă și plăcută de a explora concepte complexe. Prin intermediul provocărilor și scenariilor simulate, jucătorii vor învăța să aplice principii algoritmice în situații reale, dezvoltându-și abilitățile de rezolvare a problemelor și gândirea critică. Astfel, acest joc nu doar educă, ci și motivează utilizatorii să aprofundeze subiecte esențiale într-un mod plăcut și memorabil.

Jocurile serioase acoperă o gamă vastă de domenii precum sănătate (există jocuri care sunt folosite ca tratament pentru copii cu vârste între 8-12 ani care au anumite tipuri de ADHD [1]), instrumente științifice (de exemplu, în domeniul automobilistic se folosește BeamNG.tech pentru simularea condusului și accidentelor[2]), terapie prin exerciții (de exemplu, Wii Sports este folosit pentru coordonarea ochi-mână[3]), politică/cultură/publicitate (jocuri care simulează crearea unei civilizații, cum ar fi jocurile Civilization, sau jocuri care simulează crearea unei afaceri, precum OpenTTD, care simulează administrarea unei afaceri feroviare), securitate (jocurile din domeniul securității au ca scop controlul dezastrelor. Instituțiile publice și private beneficiază de acest tip de jocuri (Poliție, Pompieri, Agenția Federală pentru Ajutor Tehnic - THW Germania)), jocuri Militare (jocuri precum America's Army sunt folosite ca simulări de antrenament pentru pregătirea și recrutarea soldaților), jocuri de recrutare (anumite companii încearcă să se prezinte și să își creeze un profil prin acest tip de jocuri pentru a atrage ucenici și candidați), educație pentru adulți (simulări reale și jocuri de simulare care oferă utilizatorului oportunitatea de a căpăta experiență și cunoștințe[4]. De exemplu, The News Game, cu 100 de titluri și povești, în care trebuie să ghicești dacă sunt știri reale sau false, testându-ți capacitatea de deducție și cunoștințele despre actualități[5]), educație pentru tineri: jocuri educative care ajută la dezvoltarea cunoștințelor și abilităților în rândul tinerilor, jocuri de artă: acestea folosesc jocuri pe calculator pentru a crea artă interactivă[6], etc.

Deoarece această lucrare își propune să implementeze un joc serios destinat programatorilor începători, algoritmii abordați sunt selectați pentru a fi cât mai accesibili și ușor de înțeles. Astfel, se va pune accent pe algoritmi simpli și fundamentali, cum ar fi algoritmul de sortare prin selecție (Selection Sort) și algoritmul de sortare prin bule (Bubble Sort),etc. Acești algoritmi nu doar că oferă o bază solidă pentru învățarea principiilor de bază ale programării, dar sunt și esențiali pentru înțelegerea conceptelor mai avansate pe care utilizatorii le vor întâlni pe măsură ce își dezvoltă abilitățile de programare.

Alegerea temei acestei lucrări a fost determinată de nevoia de a combina educația în domeniul algoritmilor cu metode inovatoare de învățare, într-o manieră accesibilă și atractivă pentru noua generație de programatori. Jocurile serioase oferă o platformă ideală pentru a transforma

conceptele teoretice în experiențe practice, stimulând astfel interesul și curiozitatea utilizatorilor. În contextul în care algoritmii devin tot mai esențiali în diverse industrii, este vital să dezvoltăm metode eficiente de predare a acestora.

Tema aleasă este deosebit de relevantă în contextul actual, în care tehnologia informației se dezvoltă rapid, iar competențele în domeniul algoritmilor sunt din ce în ce mai solicitate pe piața muncii. Jocurile serioase reprezintă o metodă modernă de educație care poate îmbunătăți semnificativ procesul de învățare, făcându-l mai atractiv și mai eficient. În acest fel, utilizatorii nu doar că învață algoritmi, ci și dezvoltă abilități critice de gândire și rezolvare a problemelor, esențiale în cariera lor profesională.

Metodologia folosită în dezvoltarea acestui proiect include cercetarea teoretică asupra algoritmilor și a jocurilor serioase, proiectarea și dezvoltarea jocului folosind instrumente de programare specifice, găsirea unor modalități de ilustrare ale algoritmilor in joc și testarea acestuia pentru a asigura o experiență de utilizare optimă. Instrumentele utilizate includ limbajul de programare C#[1] , platforme de dezvoltare a jocurilor precum Unity[2] și diverse resurse educaționale pentru documentarea algoritmilor.

În această lucrare, vom aborda subiecte precum domeniul și contextul abordării temei propuse, precum și o prezentare comparativă privind tipurile de aplicații deja existente în Capitolul 1, intitulat "Fundamentarea teoretică și documentarea bibliografică". În Capitolele 2 și 3 vom discuta despre modul de proiectare al aplicației și implementarea acesteia, acoperind tehnologiile utilizate, modulele generale și felul în care acestea interacționează între ele, probleme întâmpinate și rezolvarea acestora. În final, sunt prezentate rezultatele testării aplicației (Capitolul 4) și concluziile acestei lucrări, împreună cu câteva gânduri personale.

[1] https://dotnet.microsoft.com/en-us/languages/csharp
[2] https://unity.com/

Capitolul 1. Fundamentarea teoretică și documentarea bibliografică

1.1. Domeniul și contextul abordării temei

În ultimele decenii, avansul tehnologic a adus schimbări semnificative în diverse domenii, iar educația nu face excepție. În acest context, jocurile serioase au apărut ca o metodă inovatoare de învățare, combinând aspectele ludice cu scopuri educative. Aceste jocuri sunt proiectate nu doar pentru divertisment, ci și pentru a oferi utilizatorilor oportunități de învățare și dezvoltare a unor competențe specifice.(Figura 1.1)

Figura 1.1. The context of serious games. Image: Hydra Interactive, 2021

Un segment important al acestui domeniu este reprezentat de jocurile serioase destinate educației în programare, care au scopul de a facilita învățarea algoritmilor și a structurilor de date. Algoritmii reprezintă fundamentele programării, fiind esențiali pentru rezolvarea eficientă a problemelor și pentru dezvoltarea de software de calitate. Începătorii în programare trebuie să își formeze o bază solidă în înțelegerea și aplicarea algoritmilor pentru a progresa către concepte mai complexe.

Deoarece această lucrare își propune să implementeze un joc serios destinat programatorilor începători, algoritmii abordați sunt selectați pentru a fi accesibili și ușor de înțeles. Astfel, se va pune accent pe algoritmi simpli și fundamentali, cum ar fi algoritmul de sortare prin selecție (Selection Sort), algoritmul de sortare prin bule (Bubble Sort), algoritmul de sortare rapidă (Quicksort) și algoritmul de sortare prin inserție (Insertion Sort). Acești algoritmi nu doar că oferă o bază solidă pentru învățarea principiilor de bază ale programării, dar sunt și esențiali pentru înțelegerea conceptelor mai avansate pe care utilizatorii le vor întâlni pe măsură ce își dezvoltă abilitățile de programare.

În cadrul contextului educațional actual, învățarea prin intermediul jocurilor serioase oferă numeroase avantaje. Aceste jocuri stimulează implicarea activă a utilizatorilor, oferindu-le oportunitatea de a învăța prin practică și de a aplica cunoștințele teoretice într-un mediu interactiv. De asemenea, jocurile serioase pot adapta nivelul de dificultate în funcție de progresul utilizatorului, asigurând astfel o curba de învățare optimizată.

Documentarea bibliografică a arătat că există deja numeroase studii și cercetări care susțin eficiența jocurilor serioase în educație. De exemplu, un studiu realizat de L. A. Annetta, J. Minogue, S. Y. Holmes, si M.-T. Cheng [7] a demonstrat că utilizarea jocurilor serioase în predarea științelor computaționale a dus la o îmbunătățire semnificativă a performanțelor academice ale ele-

vilor. De asemenea, un alt studiu realizat de M. Prensky[8] a evidențiat faptul că jocurile video pot îmbunătăți abilitățile de rezolvare a problemelor și gândirea critică.Gomes și Mendes [9] au specificat că învățarea programării este dificilă din mai multe motive, printre care nivelul ridicat de abstractizare necesar, cerința de cunoștințe teoretice și abilități practice de rezolvare a problemelor, precum și natura foarte practică și intensă a studiului necesar. De asemenea, metodologiile folosite de profesori nu iau întotdeauna în considerare stilurile de învățare ale studenților, iar limbajele de programare au o sintaxă complexă, concepută mai mult pentru uz profesional decât pentru scopuri pedagogice.

Prin urmare, implementarea unui joc serios pentru învățarea algoritmilor simpli se aliniază cu tendințele actuale din educație și oferă o metodă inovatoare și eficientă pentru dezvoltarea competențelor de programare ale începătorilor. Această lucrare va explora în detaliu fundamentele teoretice ale algoritmilor selectați și va descrie procesul de dezvoltare a jocului.

1.2. Tema propusă (formularea exactă a temei, obiective, justificarea abordării)

Tema propusă in această lucrare este implementarea unui joc serios pentru învățarea algoritmilor fundamentali de sortare destinat programatorilor începători. Proiectul are ca obiective dezvoltarea unei platforme interactive de învățare care implică crearea unui joc serios care să faciliteze învățarea algoritmilor de sortare pentru programatorii începători printr-o metodă interactivă și captivantă, acoperirea unor algoritmi fundamentali de sortare și stimularea gândirii critice și a abilităților de rezolvare a problemelor (acest oviectiv necesită proiectarea unor scenarii și provocări care să permită utilizatorilor să aplice algoritmii în contexte variate și să își dezvolte abilitățile de gândire critică).

În contextul actual al educației în domeniul tehnologiei informației, metodele tradiționale de predare nu sunt întotdeauna suficiente pentru a capta interesul și atenția noilor generații de programatori. Jocurile serioase reprezintă o soluție inovatoare care combină elementele ludice cu scopuri educative, oferind astfel o modalitate atractivă și eficientă de învățare.

Implementarea unui joc serios pentru învățarea algoritmilor de sortare este justificată de mai multe aspecte, acestea fiind acoperite în urmatoarele rânduri.

Interactivitate crescută: Prin intermediul jocului, utilizatorii sunt implicați activ în procesul de învățare, ceea ce conduce la o înțelegere mai profundă a conceptelor teoretice.

Metodă de învățare adaptivă: Jocurile serioase permit adaptarea nivelului de dificultate în funcție de progresul utilizatorului, asigurând astfel o curba de învățare optimizată.

Dezvoltarea competențelor esențiale: Algoritmii de sortare sunt fundamentali în programare, iar învățarea lor prin intermediul unui joc serios contribuie la formarea unei baze solide pe care utilizatorii pot construi cunoștințe avansate.

Feedback instantaneu: Sistemul de evaluare din cadrul jocului oferă feedback imediat, ajutând utilizatorii să identifice și să corecteze greșelile într-un mod eficient.

Motivație și angajament: Elemente de gamificare, cum ar fi punctele, nivelele și recompensele, sporesc motivația și angajamentul utilizatorilor, transformând învățarea într-o activitate plăcută și satisfăcătoare.

Prin urmare, această abordare inovatoare nu doar că facilitează învățarea algoritmilor de sortare, dar și pregătește utilizatorii pentru provocările complexe ale programării, dezvoltându-le abilități critice și gândirea analitică necesară pentru succesul în domeniul tehnologiei informației.

1.3. Prezentare succintă și comparativă privind realizările actuale și analiza tipurilor de aplicații existente

În domeniul educației asistate de jocuri serioase, numeroase studii și aplicații au demonstrat eficiența acestei abordări în învățarea programării. Mai jos sunt prezentate câteva realizări notabile și tipurile de aplicații dezvoltate pentru învățarea algoritmilor de sortare și a altor concepte de programare.

- AlgoBot[3]: AlgoBot este un joc educativ conceput pentru a învăța utilizatorii principiile de bază ale programării și algoritmilor de sortare printr-o serie de puzzle-uri și provocări. Utilizatorii sunt ghidați prin diverse scenarii în care trebuie să aplice diferiți algoritmi pentru a rezolva probleme, beneficiind de explicații detaliate ale algoritmilor și feedback imediat. Jocul prezintă nivele progresive de dificultate, ceea ce permite o adaptabilitate la nivelul de cunoștințe al utilizatorului. Interfața sa atractivă și ușor de utilizat contribuie la o experiență plăcută de învățare. (Figura 1.2)

Figura 1.2. AlgoBot

- Sort Visually[4]: Sort Visually este o aplicație concepută pentru a demonstra algoritmii de sortare prin reprezentări vizuale detaliate. Utilizatorii pot seta diferiți parametri și pot observa în timp real modul în care funcționează algoritmii. Aplicația se remarcă prin reprezentările grafice detaliate și posibilitatea de personalizare a parametrilor, oferind și explicații textuale pentru a facilita înțelegerea. Punctele sale tari includ explicațiile vizuale excelente și nivelul ridicat de personalizare. Cu toate acestea, aplicația se concentrează mai mult pe vizualizări decât pe interactivitatea utilizatorului în procesul de învățare, ceea ce poate limita implicarea activă a acestuia.(Figura 1.3)

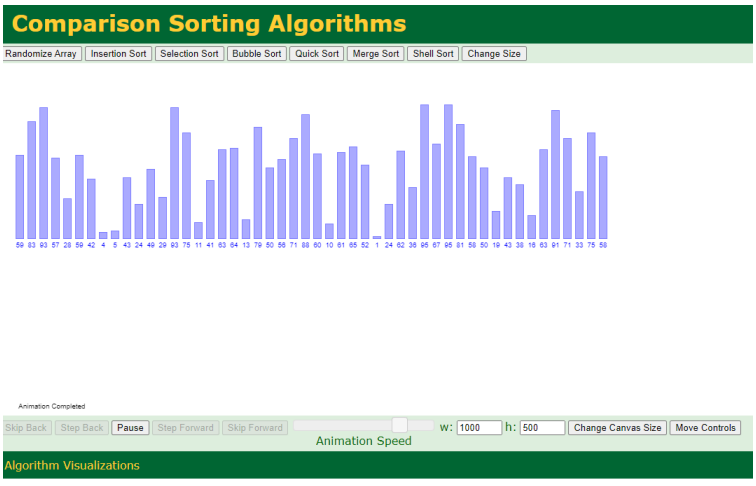


Figura 1.3. Sort Visually

Analizând tipurile de aplicații existente pentru învățarea programării prin jocuri serioase, putem observa câteva tendințe și caracteristici comune, acestea fiind menționate și detaliate în urmatoarele paragrafe.

[3] https://store.steampowered.com/app/286300/Algo_Bot/?l=romanian
[4] https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html

Aplicații vizuale și interactive: Majoritatea aplicațiilor pun un accent deosebit pe vizualizarea algoritmilor și pe interacțiunea utilizatorului cu materialul de învățare. Aceste aplicații ajută utilizatorii să înțeleagă conceptele abstracte prin reprezentări vizuale și activități practice.

Gamificare[5]: Elemente de gamificare, cum ar fi punctele, nivelele și recompensele, sunt utilizate pentru a motiva și a angaja utilizatorii. Aceste elemente fac procesul de învățare mai atractiv și mai plăcut.

Explicații detaliate și feedback: Oferirea de explicații detaliate și feedback imediat este esențială pentru a ajuta utilizatorii să înțeleagă și să corecteze greșelile pe parcursul procesului de învățare.

Adaptabilitate: Aplicațiile eficiente sunt cele care pot adapta nivelul de dificultate și conținutul în funcție de progresul și nevoile utilizatorului, asigurând astfel o curba de învățare optimizată.

1.4. Elaborarea specificațiilor privind caracteristicile așteptate de la aplicație

Jocul propus în această lucrare de licență urmărește să ofere o platformă de învățare interactivă și progresivă pentru programatorii începători. Jocul este structurat pe mai multe nivele, mai multe dintre acestea fiind dedicate câte unui algoritm de sortare: sortare prin selecție (Selection Sort), sortare prin bule (Bubble Sort), sortare rapidă (Quicksort) și sortare prin inserție (Insertion Sort).

Aplicația dezvoltată oferă următoarele funcții și caracteristici:

Explicații detaliate: Fiecare algoritm de sortare este explicat în detaliu, utilizatorii având posibilitatea de a vizualiza și câte un exemplu.

Nivele de practică: După fiecare explicație, utilizatorii pot exersa aplicarea algoritmului într-un mediu de joc interactiv, consolidându-și astfel cunoștințele. De asemenea, dificultatea nivelelor crește progresiv pentru a oferi o experiență de învățare completă și provocatoare. Această abordare permite utilizatorilor să-și dezvolte abilitățile în mod incremental, adaptându-se la complexitatea algoritmilor pe măsură ce avansează.

Explicații detaliate și feedback: Oferirea de explicații detaliate și feedback imediat este esențială pentru a ajuta utilizatorii să înțeleagă și să corecteze greșelile pe parcursul procesului de învățare.

Nivele de testare: La finalul jocului, utilizatorii sunt supuși unor nivele de test în care trebuie să recunoască algoritmul de sortare utilizat fără a fi informați în prealabil despre care algoritm este folosit. Aceste nivele de testare permit evaluarea înțelegerii și capacității de recunoaștere a algoritmilor în diferite contexte.

1.4.1. Punctele tari și punctele slabe ale aplicației

Jocul se remarcă prin interactivitatea crescută, implicând utilizatorii în mod activ prin intermediul nivelelor de practică și testare, asigurând astfel o învățare practică și aplicată. Utilizatorii primesc feedback instantaneu pe parcursul nivelelor, facilitând corectarea erorilor și învățarea eficientă. De asemenea, nivelele de testare unice permit utilizatorilor să-și evalueze cunoștințele și să identifice algoritmii în mod independent. Unul din punctele slabe ale proiectului este faptul că, acesta fiind destinat începătorilor, se concentrează pe algoritmi fundamentali și nu abordează algoritmi mai complexi sau aplicații avansate ale acestora.

1.4.2. Caracteristicile așteptate de la aplicație

Interactivitate și Usabilitate: Aplicația trebuie să aibă o interfață intuitivă, ușor de navigat și estetic plăcută. Navigarea trebuie să fie simplă și eficientă, permițând utilizatorilor să se deplaseze

[5]Gamificarea una dintre cele mai recente tendințe în educație, prin care elevii și studenții dobândesc cunoștințe prin joc. Această metodă presupune utilizarea mecanismelor jocurilor și elementelor de proiectare a jocului în procesul de predare-învățare-evaluare.[10]

cu ușurință între diferitele secțiuni ale aplicației, inclusiv nivelele de practică, testare și secțiunile explicative.

Conținut Educațional: Fiecare algoritm de sortare (Selection Sort, Bubble Sort, Quicksort, Insertion Sort) trebuie să fie explicat în detaliu, utilizând atât text cât și exemple pentru a ilustra funcționarea acestora. Exemplele practice trebuie să urmeze explicațiile teoretice, permițând utilizatorilor să aplice algoritmii în scenarii concrete și să înțeleagă mai bine principiile din spatele fiecăruia.

Nivele de Practică și Dificultate: După fiecare explicație, utilizatorii trebuie să aibă posibilitatea de a exersa aplicarea algoritmului într-un mediu de joc interactiv. Dificultatea nivelelor trebuie să crească progresiv, asigurând o experiență de învățare completă și provocatoare, adaptată la progresul utilizatorului. Aceasta va ajuta utilizatorii să își consolideze cunoștințele și să se adapteze treptat la concepte mai avansate.

Feedback și Evaluare: Aplicația trebuie să ofere feedback imediat utilizatorilor pe parcursul nivelelor de practică, facilitând corectarea erorilor și îmbunătățirea învățării. Utilizatorii trebuie să fie evaluați în mod continuu prin intermediul testelor și provocărilor, care le permit să-și evalueze cunoștințele și să identifice algoritmii fără a fi informați în prealabil despre care algoritm este folosit. Aceasta va asigura o evaluare obiectivă a înțelegerii și abilităților lor.

Performanță și Fiabilitate: Aplicația trebuie să fie performantă, oferind timpi de răspuns rapizi și o funcționare fluidă, fără întreruperi. De asemenea, trebuie să fie robustă și să funcționeze corect în diverse condiții de utilizare, asigurând integritatea datelor utilizatorilor. Aceasta va oferi o experiență de utilizare fără probleme, contribuind la satisfacția și încrederea utilizatorilor în aplicație.

1.5. Algoritmii ilustrați în joc

În cadrul jocului educativ destinat programatorilor începători, au fost selectați patru algoritmi fundamentali de sortare, fiecare fiind explicat și ilustrat în detaliu pentru a facilita învățarea. Utilizatorii au posibilitatea de a experimenta cu fiecare algoritm prin intermediul nivelelor de practică, consolidându-și astfel înțelegerea și abilitățile de aplicare a acestor concepte fundamentale. Pe măsură ce progresează, dificultatea nivelelor crește, oferind o experiență de învățare completă și provocatoare.

Acești algoritmi sunt:

1.5.1. Sortarea prin selecție (Selection Sort)

Sortarea prin selecție este un algoritm simplu și intuitiv, care funcționează prin găsirea repetată a celui mai mic element din lista nesortată și plasarea acestuia la începutul listei. Acest proces se repetă pentru fiecare element până când întreaga listă este sortată. Deși nu este cel mai eficient algoritm pentru liste mari, sortarea prin selecție este ideală pentru începători datorită simplității sale.(Figura 1.4[6])

[6] https://www.geeksforgeeks.org/c-program-for-selection-sort/

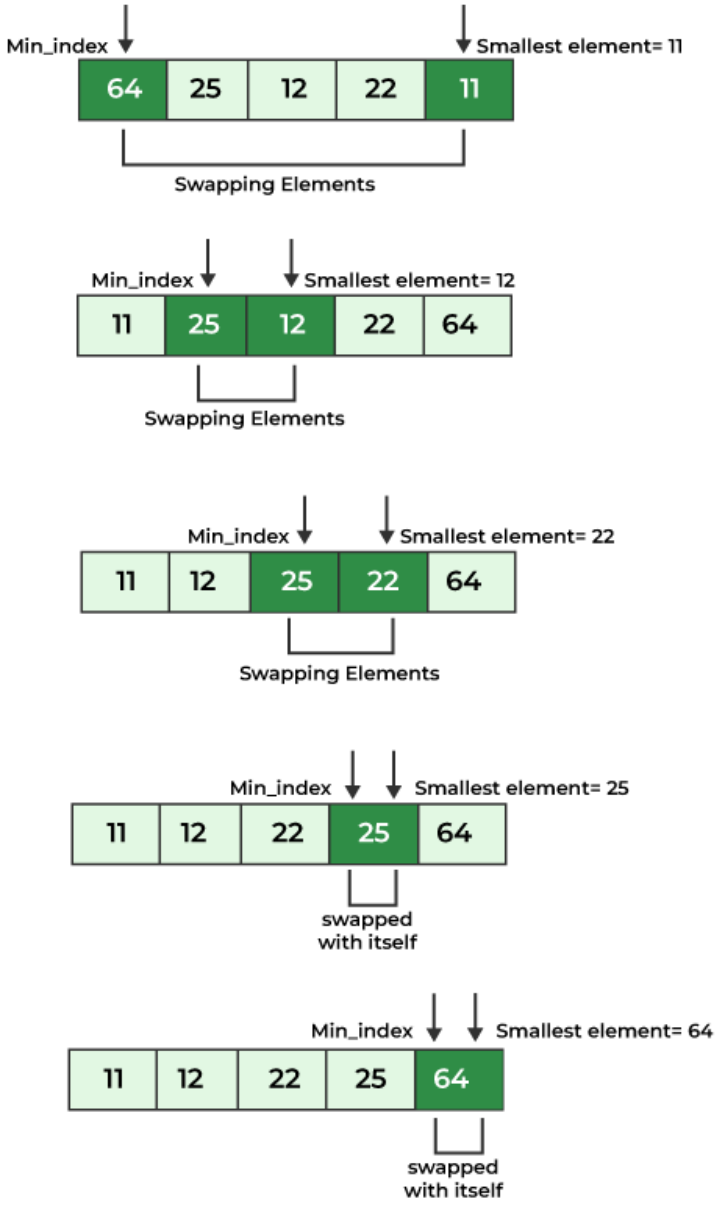


Figura 1.4. Exemplu de Sortare prin selectie. Exemplu luat de pe GeeksforGeeks

1.5.2. Sortarea prin bule (Bubble Sort)

Sortarea prin bule implică repetarea parcurgerii listei de mai multe ori, comparând elementele adiacente și schimbându-le dacă sunt în ordine greșită. Acest proces continuă până când lista este sortată. Deși este unul dintre cele mai lente algoritmi de sortare, sortarea prin bule este utilă pentru novici deoarece ilustrează bine conceptul de comparare și schimbare a elementelor.(Figura 1.5[7])

[7] https://medium.com/karuna-sehgal/an-introduction-to-bubble-sort-d85273acfcd8

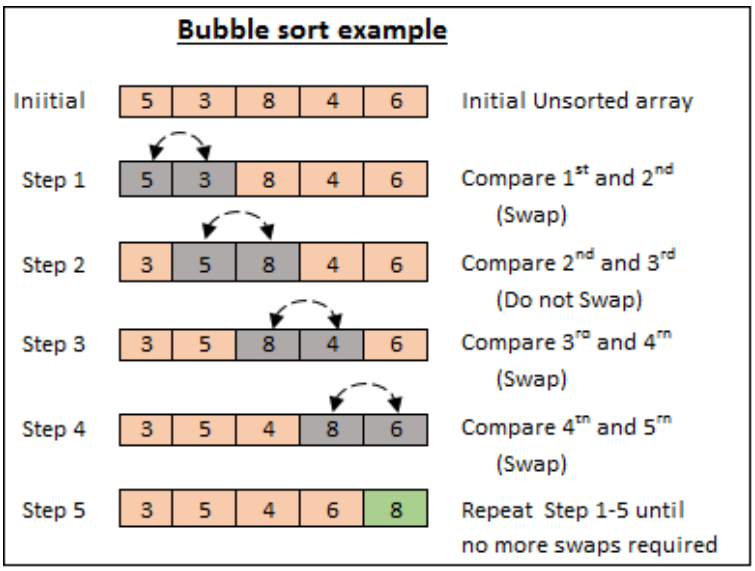


Figura 1.5. Exemplu de Sortare prin bule.Exemplu creat de Karuna Sehgal

1.5.3. Sortarea rapidă (Quicksort)

Sortarea rapidă este un algoritm eficient care utilizează o abordare de divizare și cucerire. Algoritmul alege un pivot și împarte lista în două subliste: una cu elemente mai mici decât pivotul și alta cu elemente mai mari. Acest proces se repetă recursiv pentru fiecare sublistă până când întreaga listă este sortată. Sortarea rapidă este importantă deoarece introduce concepte avansate de recursivitate și divizare a problemei.(Figura 1.6[8])

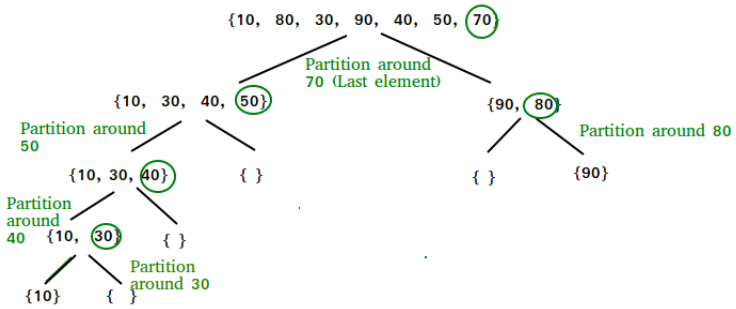


Figura 1.6. Exemplu de Sortare rapidă. Exemplu luat de pe GeeksforGeeks

[8] https://www.geeksforgeeks.org/quick-sort-algorithm//

1.5.4. Sortarea prin inserție (Insertion Sort)

Sortarea prin inserție funcționează prin construirea graduală a listei sortate prin adăugarea unui element la un moment dat, plasându-l în poziția corectă în cadrul listei sortate parțial. Acest algoritm este eficient pentru liste mici sau pentru liste aproape sortate și ajută începătorii să înțeleagă conceptul de inserare și menținere a unei liste ordonate.(Figura 1.7[9])

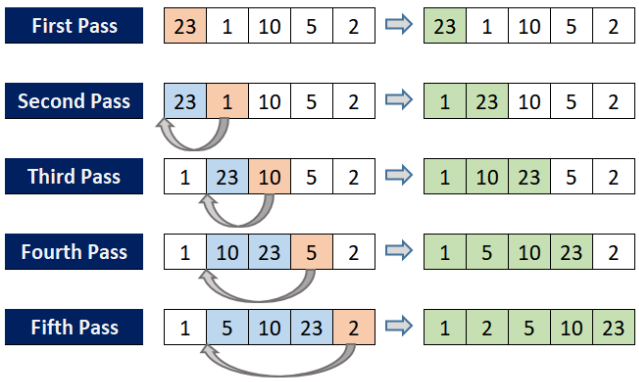


Figura 1.7. Exemplu de Sortare prin inserție. Exemplu creat de Austin Stanley.

[9] https://medium.com/austins-software-engineering-journey/insertion-sort-ea0645cc5a23

Capitolul 2. Proiectarea aplicației

În vederea proiectării unei aplicații educative eficiente și interactive pentru învățarea algoritmilor de sortare, soluția propusă în această lucrare implică utilizarea motorului de jocuri Unity pentru dezvoltare, cu scripturile scrise în limbajul C# folosind IDE-ul Visual Studio. Implementarea a fost realizată pe un calculator personal cu sistem de operare Windows, asigurând astfel un mediu de dezvoltare robust și accesibil.Această soluție permite o dezvoltare multiplatformă eficientă și garantează o performanță optimă.

2.1. Tehnologii utilizate

Pentru realizarea acestei aplicații, au fost utilizate mai multe tehnologii esențiale care contribuie la dezvoltarea și funcționarea optimă a proiectului:

2.1.1. Unity

Unity este un motor de jocuri multiplatformă foarte popular, care permite dezvoltarea de aplicații interactive și atractive. Alegerea Unity se datorează capacității sale de a oferi suport pentru multiple platforme, inclusiv Windows, macOS și Linux, precum și instrumentelor avansate pentru crearea de animații și vizualizări grafice complexe. Unity oferă un mediu integrat de dezvoltare (IDE[10]) și o gamă largă de unelte preconstruite, resurse și plugin-uri, care facilitează procesul de dezvoltare și pot reduce drastic timpul necesar. În loc să creeze totul de la zero, dezvoltatorii pot folosi magazinul de resurse al Unity și funcționalitățile integrate pentru fizică, iluminare, audio și altele[11][12].

2.1.2. C#

Scripturile aplicației sunt scrise în limbajul C#, un limbaj de programare modern și versatil, ideal pentru dezvoltarea de aplicații în Unity. C# este cunoscut pentru sintaxa sa clară și ușor de învățat, precum și pentru performanțele sale ridicate. Alegerea C# facilitează dezvoltarea și întreținerea codului, oferind în același timp o integrare perfectă cu Unity.

2.1.3. Visual Studio

Visual Studio este IDE-ul utilizat pentru scrierea și testarea scripturilor în C#. Acesta oferă o gamă largă de instrumente și funcționalități care sprijină dezvoltarea eficientă a aplicației, inclusiv debugare avansată, integrare cu sistemele de control al versiunilor și suport pentru multiple limbaje de programare. Visual Studio este bine integrat cu Unity, facilitând astfel fluxul de lucru al dezvoltatorilor.

2.1.4. Platforma Hardware

Dezvoltarea aplicației a fost realizată pe un calculator personal cu sistem de operare Windows. Specificațiile hardware ale acestui calculator includ un procesor 13th Gen Intel Core i7-13700HX, 16 GB de memorie RAM și o placă grafică NVIDIA RTX 4060. Aceste specificații asigură o performanță optimă în timpul dezvoltării și testării aplicației, permițând rularea fluidă a Unity și a Visual Studio.

Aceste specificații asigură o performanță optimă în timpul dezvoltării și testării aplicației, permițând rularea fluidă a Unity și a Visual Studio. Procesorul de ultimă generație și placa grafică

[10]Un mediu de dezvoltare este un set de programe care ajută programatorul în scrierea programelor. Un mediu de dezvoltare combină toți pașii necesari creării unui program într-un singur soft, care, de regulă, oferă o interfață cu utilizatorul grafică, prietenoasă.

puternică oferă capabilități avansate pentru simularea și vizualizarea grafică, iar memoria RAM de 16 GB garantează că aplicația poate gestiona sarcini complexe fără întârzieri.

2.2. Modulele generale ale aplicației și interacțiunile dintre ele

Aplicația este structurată în mai multe module generale, fiecare având funcționalități specifice. Această secțiune descrie modulele principale ale aplicației și interacțiunile dintre ele, asigurând astfel o funcționare coerentă și eficientă a sistemului.

2.2.1. Modulul Interfață Utilizator (UI)

Modulul UI este responsabil pentru afișarea elementelor grafice și pentru interacțiunea cu utilizatorul. Acesta include meniuri, butoane și alte componente vizuale care permit utilizatorilor să navigheze prin aplicație și să acceseze diferitele funcționalități. Interacțiunile dintre utilizator și aplicație sunt gestionate prin acest modul, care transmite comenzile utilizatorilor către celelalte module ale aplicației.

2.2.2. Modulul de Logica a Jocului

Modulul Logica Jocului gestionează regulile jocului, progresul nivelelor și aplicarea algoritmilor de sortare. Acesta include logica de control pentru fiecare nivel, determină comportamentul jocului în funcție de acțiunile utilizatorului și gestionează starea jocului. Modulul Logica Jocului interacționează cu Modulul UI pentru a actualiza afișarea în funcție de progresul utilizatorului și pentru a oferi feedback instantaneu.

2.2.3. Modulul Audio

Modulul Audio este responsabil pentru gestionarea sunetelor și muzicii de fundal în aplicație. Acesta include efecte sonore pentru interacțiuni și feedback audio pentru acțiunile utilizatorului . Modulul Audio contribuie la crearea unei atmosfere imersive și poate fi configurat pentru a oferi feedback auditiv care completează interacțiunile vizuale.

2.2.4. Modulul de Rendering

Modulul de rendering[11] din Unity este responsabil pentru procesarea și afișarea graficelor în joc. Acesta folosește o varietate de tehnici și optimizări pentru a asigura că jocul arată bine și rulează eficient pe diferite platforme hardware. Unity utilizează un pipeline de rendering flexibil care poate fi configurat și extins în funcție de nevoile jocului. Există două pipeline-uri principale de rendering în Unity:

- Built-in Render Pipeline: Este pipeline-ul implicit și tradițional din Unity, care oferă un echilibru între performanță și calitatea vizuală. Acesta suportă o gamă largă de platforme și este ușor de folosit pentru proiectele standard.

- Scriptable Render Pipeline (SRP): Introduce flexibilitate și control sporit asupra procesului de rendering. Unity oferă două implementări principale ale SRP. Universal Render Pipeline (URP): Oferă un echilibru excelent între performanță și calitate, fiind ideal pentru jocuri 2D, 3D și aplicații AR/VR. High Definition Render Pipeline (HDRP): Este destinat proiectelor care necesită grafică de înaltă calitate, precum jocurile AAA și aplicațiile care rulează pe hardware performant[13].

[11] Termenul "rendering" provine din limba engleză și se referă la procesul de generare a unei imagini sau a unui cadru video dintr-un model 2D sau 3D prin intermediul unui program de computer.

2.2.5. Modulul pentru Hitboxes

Hitboxes[12] sunt componente esențiale pentru detectarea coliziunilor în jocuri. Unity oferă un sistem robust pentru gestionarea coliziunilor și a trigger-elor prin utilizarea componentelor Collider și Rigidbody.

Colliders: Definirea zonelor de coliziune pentru obiecte. Unity oferă diverse tipuri de colliders, inclusiv BoxCollider, SphereCollider, și MeshCollider.

Rigidbodies: Adaugă fizică obiectelor, permițându-le să fie afectate de forțe și coliziuni[14].

2.2.6. Prelucrarea intrărilor utilizatorului (termenul în engleză - input)

Aplicația dezvoltată folosește noul flux de lucru(termenul în engleză - workflow)/sistem de prelucrare al input-ului, oferit de Unity, modul general de funcționare al sistemului putând fi observat în Figura 2.1[13]. Acesta colectează informația de la dispozitivele conectate (tastatură, mouse, etc.) și trimite evenimentele corespunzătoare la Input System. Sistemul de input traduce aceste evenimente in acțiuni (bazat pe informația de acțiune și informația de legătură, în engleză binding, stocată în Input Action Assets). El trimite mai apoi acțiunile la scriptul PlayerInput care apelează metodele corespunzătoare. În cazul concret al acestei lucrări, jocul necesită și prelucrează doar informația de la un singur dispozitiv, acesta fiind mouse-ul.

(Figura 2.1[14])

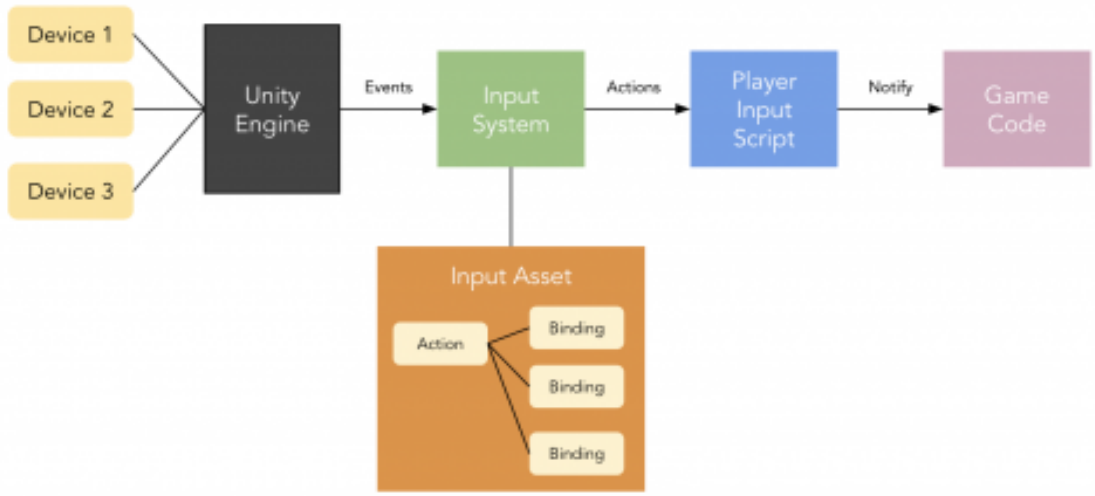


Figura 2.1. Input Workflow

Interacțiunile dintre module sunt esențiale pentru funcționarea corectă și eficientă a aplicației. Modulul UI interacționează cu toate celelalte module pentru a afișa informațiile relevante și pentru a prelua inputul utilizatorilor. Modulul Logica Jocului coordonează activitățile între diferitele nivele de practică și testare, asigurându-se că progresul utilizatorilor este monitorizat și că fiecare nivel este adaptat corespunzător.

2.3. Analiza avantajelor și dezavantajelor metodei alese

[12]Termenul "hitbox" provine din limba engleză și se referă la o zonă invizibilă definită în jurul unui obiect într-un joc video, folosită pentru a detecta coliziunile cu alte obiecte. Hitbox-urile sunt esențiale pentru determinarea interacțiunilor precise între obiectele de joc.

[13]https://www.kodeco.com/9671886-new-unity-input-system-getting-started

[14]https://www.kodeco.com/9671886-new-unity-input-system-getting-started

2.3.1. Avantajele metodei alese

Utilizarea Unity permite dezvoltarea unei aplicații care poate rula pe multiple platforme, inclusiv Windows, macOS și Linux, fără a necesita modificări majore ale codului. Aceasta oferă flexibilitate și accesibilitate, permițând utilizatorilor să acceseze aplicația pe diverse dispozitive și sisteme de operare.

Unity oferă un mediu de dezvoltare integrat cu numeroase resurse și plugin-uri care facilitează procesul de dezvoltare. Utilizarea C# ca limbaj de programare principal, împreună cu IDE-ul Visual Studio, permite scrierea unui cod clar și ușor de întreținut. Acestea reduc timpul necesar pentru dezvoltare și actualizare, permițând adăugarea rapidă de noi funcționalități.

Unity este cunoscut pentru capacitățile sale grafice avansate și pentru suportul pentru animații complexe. Aceasta permite crearea unei interfețe grafice atractive și a unor vizualizări interactive care îmbunătățesc experiența utilizatorilor și facilitează învățarea. De asemenea, folosirea motorului de jocuri Unity reduce complexitatea si timpul de lucru necesar pentru dezvoltarea si crearea mai multor componente si module(componentă de randare, modulul UI, modulul Audio, etc.)(În Figura 2.2[15] se prezintă fluxul de lucru general al resurselor, termentul în engleză - assets)

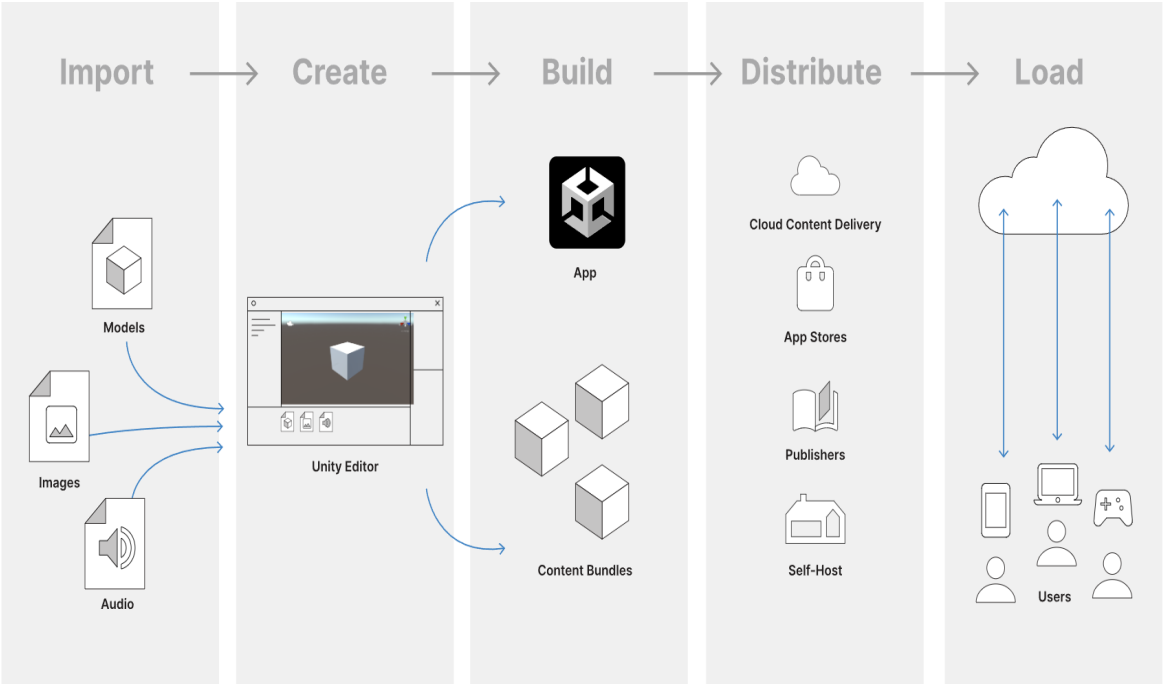


Figura 2.2. Asset Workflow. Preluat din documentatia Unity

Unity beneficiază de o comunitate vastă de dezvoltatori și de un suport tehnic solid, incluzând documentație detaliată, forumuri de discuții și tutoriale. Acest suport extensiv poate fi extrem de util în rezolvarea problemelor tehnice și în optimizarea aplicației.

2.3.2. Dezavantajele metodei alese

Performanța aplicației poate varia în funcție de specificațiile hardware ale dispozitivelor utilizate. Pe dispozitive cu specificații hardware mai slabe, aplicația poate rula mai lent sau poate întâmpina probleme de stabilitate.

Optimizarea aplicației pentru a funcționa eficient pe toate platformele suportate poate necesita eforturi suplimentare. Ajustarea setărilor grafice și a performanței pentru a asigura o experiență fluidă pe dispozitive variate poate fi complexă și consumatoare de timp.

[15] https://docs.unity3d.com/Manual/AssetWorkflow.html

Utilizarea Unity și a limbajului C# implică o dependență de acest framework specific. În cazul în care apar schimbări majore în Unity sau dacă este necesară migrarea către un alt framework, pot apărea dificultăți semnificative în adaptarea și menținerea codului existent.

Metoda aleasă funcționează optim pe dispozitive cu specificații hardware medii și înalte. Pe dispozitivele cu specificații hardware minime, performanța poate fi afectată, iar unele funcționalități grafice avansate pot fi limitate. Compatibilitatea aplicației este garantată pentru sistemele de operare Windows, macOS și Linux, iar utilizarea altor sisteme de operare poate necesita ajustări suplimentare.

În plus, utilizarea Unity și C# presupune că dezvoltatorii au cunoștințele necesare pentru a lucra eficient cu acest framework și limbaj de programare. De asemenea, este important să se țină cont de actualizările și modificările aduse de Unity, care pot influența modul în care aplicația funcționează sau este întreținută.

2.4. Componente software

Diagrama UML(Figura A.1)

2.4.1. Clasa PauseMenu

Clasa PauseMenu gestionează funcționalitățile meniului de pauză al jocului, permițând utilizatorului să pună pauză, să reia jocul, să revină la meniul principal și să salveze progresul curent. (vezi Anexa 3)

Funcționalități principale:

- Pause(): Activează meniul de pauză.
- Home(): Încărcă scena meniului principal.
- Resume(): Dezactivează meniul de pauză.
- Restart(): Reîncarcă scena curentă.
- saveProgress(): Salvează progresul curent al jocului într-un fișier text.

2.4.2. Clasa PillarSlot

Clasa PillarSlot gestionează comportamentul sloturilor în care trebuie plasați stâlpii, verificând corectitudinea plasării acestora.

Funcționalități principale:

- Placed(): Marchează slotul ca fiind ocupat de un stâlp plasat corect.

2.4.3. Clasa Pillar

Clasa Pillar gestionează comportamentul stâlpilor în joc, inclusiv interacțiunile utilizatorului cu aceștia și plasarea lor în pozițiile corecte.(vezi Anexa 4)

Funcționalități principale:

- Initialize(PillarSlot): Inițializează stâlpul cu slotul asociat.
- OnMouseDown(): Gestionează acțiunea de preluare a stâlpului de către utilizator.
- OnMouseUp(): Gestionează acțiunea de eliberare a stâlpului de către utilizator și verifică plasarea acestuia.
- Update(): Actualizează poziția stâlpului în timpul interacțiunii.

2.4.4. Clasa MainMenu

Clasa MainMenu gestionează funcționalitățile meniului principal al jocului, inclusiv navigarea către diferitele niveluri și ajustarea setărilor de volum.(vezi Anexa 5)

Funcționalități principale:

- PlayGame(): Încarcă primul nivel al jocului.
- QuitGame(): Închide aplicația.
- ChangeVolume(): Ajustează volumul muzicii în joc.
- continueButton(): Continuă jocul de la ultimul nivel salvat.

2.4.5. Clasa PuzzleManager

Clasa PuzzleManager gestionează logica jocului pentru nivelurile de puzzle, inclusiv poziționarea stâlpilor și verificarea corectitudinii plasării acestora.(vezi Anexa 2)

Funcționalități principale:

- Spawn(): Inițializează stâlpii și sloturile la începutul nivelului.
- restrictMovement(): Restricționează mișcarea stâlpilor care au fost plasați corect.
- countSwaps(): Verifică numărul de schimburi corecte realizate de utilizator.
- changeStage(): Trecerea la următorul pas al algoritmului după realizarea corectă a schimburilor.

2.5. Componente hardware:

Pentru a rula jocul educativ dezvoltat, cerințele hardware sunt destul de accesibile și nu necesită un sistem foarte performant. Jocul poate fi controlat în întregime folosind un mouse, ceea ce face interacțiunea simplă și intuitivă. În ceea ce privește specificațiile minime, jocul a fost testat și rulat cu succes pe un sistem echipat cu o placă grafică GTX 1050, un procesor Intel i5 de generația a 9-a (Kaby Lake), 8 GB de memorie RAM și un hard disk (HDD). Aceste cerințe minime asigură că jocul va funcționa fluent și va oferi o experiență de utilizare plăcută pentru utilizatori.

Capitolul 3. Implementarea aplicației

3.1. Descrierea generală a implementării

Una dintre componentele fundamentale ale aplicației este modulul de logică a jocului, care este în mare măsură reprezentat de clasa PuzzleManager (vezi Anexa 2). Această clasă joacă un rol esențial în orchestrarea dinamicii jocului, asigurând buna funcționare și interactivitate a aplicației. PuzzleManager este responsabilă pentru inițializarea tuturor elementelor din fiecare nivel, stabilind pozițiile inițiale ale stâlpilor și sloturilor. Această inițializare este realizată prin citirea unei matrici care specifică pozițiile corecte ale pieselor la începutul fiecărui nivel, precum și în fiecare pas al algoritmului până la final. Aceasta permite modulului să cunoască nu doar pozițiile inițiale, ci și tranzițiile necesare pe parcursul algoritmului de sortare, oferind astfel un cadru clar și structurat pentru utilizatori. În timpul jocului, aceasta monitorizează și verifică corectitudinea acțiunilor utilizatorului, asigurându-se că piesele sunt plasate corect conform regulilor algoritmilor de sortare predați. Mai mult, PuzzleManager impune restricții asupra mișcării pieselor, blocând stâlpii care au fost deja plasați corect, pentru a preveni modificările incorecte și a menține integritatea procesului de învățare. În momentul în care utilizatorul finalizează corect un nivel, clasa gestionează tranziția către scena următoare, facilitând astfel progresul continuu prin secvențele educative ale jocului.

```
1       void Update()
2   {
3       CheckLevelCompleted();
4       restrictMovement();
5       countSwaps();
6       changeStage();
7
8   }
9       void Spawn()
10  {
11      var Set = slotPrefabs.ToList();
12      for(int i = 0; i < Set.Count(); i++)
13      {
14          pillarSlots[i] = Instantiate(Set[i],
15          slotParent.GetChild(i).position,Quaternion.identity);
16
17          pillars[i]= Instantiate(pillarPrefab,
18          pieceParent.GetChild(i).position,Quaternion.identity);
19          pillars[i].transform.localScale = slotParent.GetChild(i).localScale;
20          pillars[i].Initialize(pillarSlots[i]);
21      }
22
23  }
```

Listing 3.1. Funcții importante din clasa PuzzleManager

Clasele Pillar este responsabilă pentru manipularea stâlpilor în joc. Aceasta include gestionarea evenimentelor de preluare și eliberare a stâlpilor de către utilizator, precum și actualizarea poziției acestora în timpul interacțiunii. De asemenea, în această clasă este programată și o mecanică prin care, dacă stâlpul este suficient de aproape de slotul său desemnat, acesta se fixează automat în locul corect, facilitând plasarea exactă și ușurând sarcina utilizatorului. Clasa PillarSlot gestionează sloturile în care stâlpii trebuie plasați. Aceasta verifică dacă stâlpii au fost plasați corect și marchează sloturile ca fiind ocupate.Prin intermediul acestor două clase, aplicația reușește

să creeze o interacțiune fluidă și intuitivă între utilizator și elementele din joc.

```
     public void Initialize(PillarSlot slot)
 {
     _renderer.sprite=slot.Renderer.sprite;
     _slot = slot;

 }
 void OnMouseDown()
{
    if (stopMoving) return;
    dragging = true;
    source.PlayOneShot(pickUpClip);
    offset = GetMousePosition() - (Vector2)transform.position;
}

void OnMouseUp()
{
    if (stopMoving) return;
    if (Vector2.Distance(transform.position, _slot.transform.position) < 1)
    {
        _slot.Placed();
        dragging = false;
        transform.position = _slot.transform.position;
    }
    else
    {
        source.PlayOneShot(dropClip);
        dragging = false;
    }
}
```

Listing 3.2. Funcții importante din clasa Pillar

Modulul UI este reprezentat de clasele MainMenu și PauseMenu. Aceste clase gestionează interfața utilizatorului, permițând navigarea prin meniurile jocului. Clasa MainMenu gestionează meniul principal al jocului, oferind utilizatorului opțiuni pentru a începe jocul, a continua de unde a rămas, a ajusta setările de volum și a ieși din joc. Aceasta asigură o navigare ușoară și intuitivă pentru utilizatori, facilitând accesul rapid la funcțiile principale ale jocului. Prin intermediul acestei clase, utilizatorii pot începe o nouă sesiune de joc, pot continua progresul salvat anterior sau pot ajusta volumul jocului in meniul Setari. De asemenea, clasa aceasta se ocupă si de trecerea de la o scenă la alta atunci cand scena respectivă nu este un nivel.

```
    [SerializeField] private AudioSource source;
[SerializeField] private AudioClip buttonClick;

public void PlayClick()
{
    source.PlayOneShot(buttonClick);
}

public void ChangeVolume()
{
    musicMixer.SetFloat("VolumeParameter",musicVol.value);
}
```

```
13
14  public void PlayGame()
15  {
16      SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
17  }
18
19  public void QuitGame()
20  {
21      Application.Quit();
22  }
23
24  public void LoadScene()
25  {
26      SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + int.Parse
27
28  }
29  public void continueButton()
30  {
31
32      LoadStringFromFile("Assets/SaveFiles/SaveFile.txt");
33      if(saveFileLevel!="")
34      {
35          LoadScene();
36      }
37      else
38      {
39          PlayGame();
40      }
41
42  }
```

Listing 3.3. Funcții importante din clasa MainMenu

Clasa PauseMenu gestionează funcționalitățile meniului de pauză, permițând utilizatorului să întrerupă temporar jocul, să reia jocul, să revină la meniul principal și să salveze progresul curent. Aceasta oferă opțiuni esențiale pentru gestionarea sesiunii de joc, asigurându-se că utilizatorii pot întrerupe și relua jocul fără a pierde progresul. De asemenea, PauseMenu permite utilizatorilor să salveze progresul curent al jocului, oferind flexibilitate și control asupra experienței de joc.

```
1       public void Pause()
2   {
3       pauseMenu.SetActive(true);
4       Time.timeScale = 0;
5   }
6
7   public void Home()
8   {
9       SceneManager.LoadScene("MainMenu");
10      Time.timeScale = 1;
11  }
12
13  public void Resume()
14  {
15      pauseMenu.SetActive(false);
16      Time.timeScale = 1;
```

```
17    }
18
19    public void Restart()
20    {
21        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
22        Time.timeScale = 1;
23    }
24
25
26    public void saveProgress()
27    {
28        int currentScene = SceneManager.GetActiveScene().buildIndex;
29        ClearFileAndWriteString("Assets/SaveFiles/SaveFile.txt", currentScene.ToStrin
30
31
32    }
```

Listing 3.4. Funcții importante din clasa PauseMenu

Folosind Unity, dezvoltatorii beneficiază de un set extins de instrumente care simplifică semnificativ procesul de creare a interfeței de utilizator fără a fi necesară scrierea de cod suplimentar pentru randare. Unity oferă butoane predefinite și elemente UI care pot fi personalizate și stilizate direct în editorul său. Acest lucru elimină necesitatea de a crea funcții de randare manuală pentru butoane, texte și alte elemente grafice. De asemenea, Unity permite stilizarea textului și a fundalurilor direct în interfața sa, oferind opțiuni vizuale intuitive pentru setarea fonturilor, culorilor și altor proprietăți stilistice. Acest lucru facilitează crearea unui aspect vizual consistent și atrăgător pentru meniurile și dialogurile jocului. Pe lângă elementele grafice, Unity oferă suport integrat pentru gestionarea sunetului. Dezvoltatorii pot adăuga și configura efecte sonore și muzică de fundal fără a scrie cod suplimentar, utilizând doar instrumentele furnizate de Unity pentru gestionarea și personalizarea sunetului în joc.

3.2. Probleme speciale/dificultăți întâmpinate și modalități de rezolvare

La începutul dezvoltării, piesele și sloturile din nivel erau inițializate conform pozițiilor definite în editorul Unity. Aceasta a creat o problemă deoarece pozițiile inițiale nu se aliniau cu pozițiile ulterioare pe care piesele trebuiau să le ocupe în timpul jocului. Această problemă a fost rezolvată prin reinițializarea pieselor și sloturilor cu valori predefinite în funcția Spawn(). Astfel, piesele sunt acum plasate corect la începutul fiecărui nivel.

Inițial, logica jocului era concepută pentru a permite doar o singură interschimbare de piese în fiecare pas al algoritmului, ceea ce nu funcționa pentru algoritmul Insertion Sort, care necesită mai multe interschimbări în unele etape. Pentru a rezolva această limitare, modul de funcționare al clasei PuzzleManager a fost regândit. Am adăugat noi funcții care, pe baza unei matrici, calculau numărul de interschimbări necesare în fiecare pas al nivelului, permițând astfel integrarea algoritmului Insertion Sort în joc.

```
1        void addID(int i)
2    {
3        for(int j = 0; j < placedIDs.Length; j++)
4        {
5            if (placedIDs[j] == -1)
6            {
7                placedIDs[j] = i;
8                break;
9            }
```

```
10        }
11
12    }
13        bool isItAlreadyPlaced(int i)
14        {
15            for (int j = 0; j < placedIDs.Length; j++)
16            {
17                if (placedIDs[j] == i)
18                    return true;
19            }
20
21            return false;
22        }
23            void countTheNoOfSwapsToBeMade()
24        {
25            for (int i = 0; i < slotPrefabs.Count(); i++)
26            {
27                if (slotsPosition[i, stepStage] != 0)
28                    noSwapsToBeMade++;
29            }
30        }
```

Listing 3.5. Câteva din funcțiile adăugate pentru rezolvarea problemei respective

Un alt bug[16] identificat a apărut în situațiile în care interschimbările se întâmplau în anumite circumstanțe specifice: dacă piesa mai mare era plasată corect înaintea piesei mai mici și interschimbarea se făcea foarte rapid, jocul nu trecea la pasul următor al algoritmului. Aceasta se întâmpla deoarece algoritmul număra o interschimbare de mai multe ori, ceea ce împiedica îndeplinirea condiției necesare pentru apelarea funcției changeStage(). Problema a fost rezolvată prin adăugarea unei condiții suplimentare și modificarea uneia existente, asigurând astfel o tranziție corectă între etapele algoritmului.

```
1         void countSwaps()
2     {
3          for (int i = 0; i < slotPrefabs.Count(); i++)
4          {
5              if (pillarSlots[i].pillarPlaced == true &&
6              slotsPosition[i, stepStage] != 0 &&
7              !isItAlreadyPlaced(i) && noSwapsPerStage<noSwapsToBeMade)
8              {
9                  noSwapsPerStage++;
10                 addID(i);
11                 //Debug.Log("LastplacedID : "+ lastPlacedID);
12                 //Debug.Log("NoSwaps : "+noSwapsPerStage);
13
14             }
15         }
16     }
17
18     void changeStage()
19     {
20
```

[16]Termenul "bug" provine din limba engleză și se referă la o eroare sau defect în software sau hardware care cauzează funcționarea incorectă a unui program sau a unui sistem.

```
21        if (noSwapsPerStage >= noSwapsToBeMade && noSwapsToBeMade!=0)
22        {
23            noSwapsPerStage = 0;
24            noSwapsToBeMade = 0;
25            clearPlacedIDsMatrix();
26            //Debug.Log("LastplacedID : " + lastPlacedID);
27            //Debug.Log("NoSwaps : " + noSwapsPerStage);
28            stepStage++;
29            countTheNoOfSwapsToBeMade();
30            for (int i = 0; i < slotPrefabs.Count(); i++)
31            {
32                if (slotsPosition[i, stepStage] != 0)
33                {
34                    pillarSlots[i].transform.position =
35                    slots[slotsPosition[i, stepStage] - 1];
36
37                }
38                pillarSlots[i].pillarPlaced = false;
39                pillars[i].stopMoving = false;
40            }
41        }
42
43    }
```

Listing 3.6. Câteva din funcțiile modificate pentru rezolvarea bug-ului

3.3. Idei originale, soluții noi

Una dintre ideile originale implementate în aplicație este modul de funcționare al nivelului. Jocul cunoaște toate pozițiile viitoare ale sloturilor și pieselor, mutările corecte/interschimbările pe care utilizatorul trebuie să le facă în nivelul respectiv, starea fiecărei piese și pozițiile în care trebuie să ajungă piesele prin citirea a două matrici.

Prima matrice, denumită matrice de poziție, are un număr de linii egal cu numărul de piese din nivelul respectiv, iar coloanele sunt corespunzătoare celor trei axe/dimensiuni (x, y, z). Această matrice este folosită pentru a stoca pozițiile pe care pot ajunge sloturile/piesele.

A doua matrice este una de stare. În această matrice, numărul de linii este egal cu numărul de piese/sloturi din nivel, iar numărul de coloane este egal cu numărul de pași ai algoritmului. Dacă în matrice se găsește o valoare de 0, aceasta înseamnă că piesa cu indexul liniei respective este în poziția corectă în pasul respectiv al algoritmului și nu trebuie mutată (jocul nu lasă utilizatorul să mute o piesă care se află într-o poziție corectă). O valoare x diferită de zero reprezintă faptul că în pasul respectiv al algoritmului (în pasul egal cu indexul coloanei) slotul de pe linia respectivă este mutat pe poziția cu indexul liniei x din matricea de poziții, iar piesa de pe linia respectivă va ajunge pe poziția x la finalul pasului respectiv.

Numărul de mutări pe care trebuie să le facă utilizatorul este egal cu numărul de valori diferite de 0 din pasul respectiv. Atunci când utilizatorul a efectuat toate interschimbările, jocul trece la următorul pas (următoarea coloană din matrice). Modulul de logică al jocului știe că jucătorul a făcut toate permutările necesare rezolvării nivelului atunci când toate valorile de pe o coloană din matricea respectivă sunt 0. Această structură logică permite jocului să urmărească și să gestioneze corectitudinea mutărilor utilizatorului în timp real. În exemplul din Figura 3.1 coloana marcată cu albastru reprezinta primul pas al algoritmului.

Interpretare : În pasul 1 al algoritmului, sloturile și coloanele 2,4 și 5 rămân pe aceleași poziții și nu trebuiesc mutate. Slotul cu indexul 1 este deja mutat pe poziția 1 iar slotul 3 este mutat

pe poziția 4. Asta indică faptul că interschimbarea din acest pas se face între coloana 1(cea mai mică) si coloana 3. Din coloana respectivă se poate deduce și că în acest moment piesa 1 se află pe poziția 4 iar piesa 3 se afla pe poziția 1. Valoarea 4 din matricea de stare (marcată cu verde) corespunde cu linia 4 din matricea de poziții(marcată cu verde).

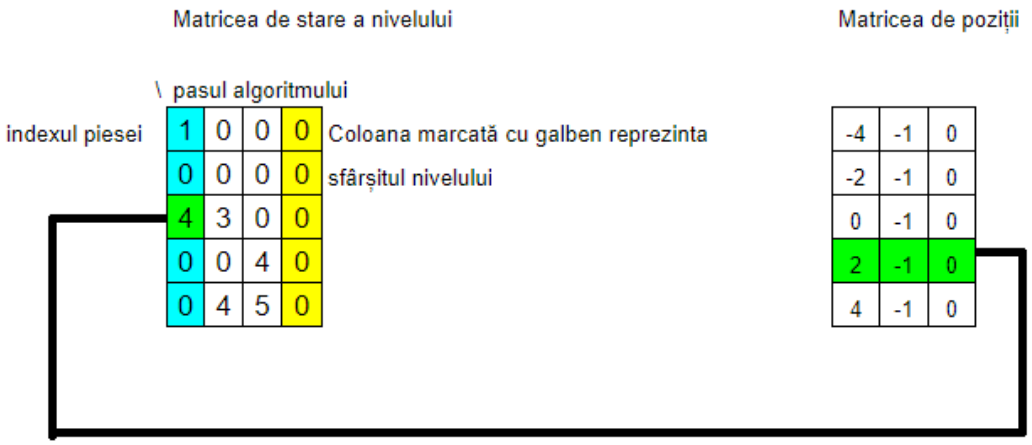


Figura 3.1. Nivelul 3

3.4. Functionarea sistemului

Jocul începe cu un ecran principal (vezi Anexa 5) (termenul în engleză : Main menu) în care utilizatorul are opțiunile de Play, Continue, Settings, Level Selector și Exit. Butonul Play începe o nouă sesiune de joc, oferind utilizatorului oportunitatea de a începe de la primul nivel. Opțiunea Continue permite reluarea jocului de unde a fost lăsat ultima dată, utilizând datele salvate anterior. Settings oferă utilizatorului posibilitatea de a ajusta setările de joc, cum ar fi volumul sunetului. Level Selector (vezi Anexa 6) permite utilizatorului să selecteze și să joace niveluri specifice, oferind astfel flexibilitate în alegerea progresului. De asemenea, jucatorul nu poate alege un nivel mai avansat față de cel care a fost ultima dată salvat. Opțiunea Exit închide aplicația.

Apăsând butonul Play, jocul începe cu o mică poveste introductivă, oferind context și imersiune în joc. După această poveste, utilizatorului i se prezintă explicația primului algoritm de sortare (vezi Anexa 7), însoțită de un scurt tutorial despre controale, pentru a se familiariza cu mecanicile jocului.

Pe parcursul nivelurilor (vezi Anexa 8), jucătorul are posibilitatea de a pune pauză, accesând meniul de pauză (vezi Anexa 9) (termenul în engleză : Pause Menu). Aici, utilizatorul are opțiuni pentru a relua jocul, a reveni la meniul principal, a salva progresul curent și a reîncepe nivelul. După completarea tutorialului și a nivelului introductiv, jucătorul continuă să treacă prin fiecare nivel, aplicând cunoștințele dobândite despre algoritmi de sortare, până la finalizarea jocului. Opțiunea de salvare a jocului memorează scena la care butonul de salvare a fost apăsat.

3.5. Interfața cu utilizatorul

Interfața cu utilizatorul (UI) a jocului este concepută pentru a oferi o experiență intuitivă și plăcută. Ecranul principal prezintă utilizatorului opțiuni clare și accesibile, permițând navigarea rapidă prin diferitele secțiuni ale jocului.

3.5.1. Ecranul principal

Ecranul principal al jocului, numit și "Main Menu", oferă utilizatorului cinci opțiuni principale: Play, Continue, Select Level, Settings și Exit. Aceste opțiuni sunt dispuse central, cu un

design atrăgător și ușor de citit.

3.5.2. Level Select

Selectorul de nivel este organizat în mod clar, prezentând nivelurile disponibile în funcție de algoritmii de sortare: Selection Sort, Bubble Sort, Quick Sort și Insertion Sort. Fiecare categorie de algoritmi este evidențiată și separată vizual, facilitând accesul utilizatorului la nivelurile dorite. De asemenea, există niveluri de testare care permit utilizatorului să își verifice cunoștințele și abilitățile acumulate.

3.5.3. Meniul de pauză

În timpul jocului, utilizatorul poate pune pauză accesând meniul de pauză. Acest meniu oferă opțiuni pentru a relua jocul, a reveni la meniul principal, a salva progresul curent și a reîncepe nivelul. Meniul de pauză este simplu și intuitiv, asigurând că utilizatorul poate gestiona cu ușurință sesiunea de joc fără a pierde progresul.

3.5.4. Interacțiunea cu butoanele

Butoanele din interfața jocului sunt concepute pentru a reacționa la acțiunile utilizatorului. Când utilizatorul trece cursorul peste un buton (hover), butonul schimbă vizual aspectul pentru a indica faptul că este activ. De asemenea, atunci când un buton este apăsat, acesta oferă un feedback vizual și auditiv pentru a confirma acțiunea utilizatorului. Aceste reacții îmbunătățesc experiența de utilizare, oferind indicii clare și imediate despre interacțiunile posibile și acțiunile efectuate.

Capitolul 4. Testarea aplicației și rezultate experimentale

4.1. Punerea în funcțiune/lansarea aplicației

Lansarea aplicației nu necesită configurări suplimentare sau proceduri complexe de instalare. Utilizatorul trebuie doar să descarce fișierul care conține resursele jocului și executabilul. După descărcare, utilizatorul poate rula direct fișierul executabil pentru a începe jocul.

4.2. Încărcarea procesorului și a memoriei

În timpul dezvoltării și testării jocului, am folosit instrumentele de profilare ale Unity pentru a monitoriza utilizarea procesorului (CPU) și a memoriei. Aceste măsurători sunt esențiale pentru a asigura că jocul funcționează eficient și fără probleme pe diverse configurații hardware.Profilerul Unity arată că utilizarea procesorului se menține la valori acceptabile, fără fluctuații majore care să afecteze performanța jocului. În primele imagini de profilare (Figurile 4.1 și 4.2), se observă că utilizarea CPU-ului pentru rendering, scripturi, fizică și alte procese este distribuită uniform, cu câteva spike-uri ocazionale care sunt normale în timpul execuției anumitor sarcini intensive.În ceea ce privește memoria, profilerul indică faptul că utilizarea memoriei totale, a memoriei pentru texturi, a memoriei pentru mesh-uri și a celei pentru materiale este bine gestionată, fără scurgeri de memorie sau creșteri necontrolate (Figurile 4.1 și 4.2). Sistemul de garbage collection (GC) funcționează corect, eliberând memoria neutilizată la intervale regulate, asigurând astfel o funcționare fluidă a jocului.

Figura 4.1. Profiler Unity

Figura 4.2. Profiler Unity

4.3. Fiabilitate și securitate

Fiabilitatea jocului a fost testată prin rularea extensivă a diverselor niveluri și monitorizarea stabilității aplicației. Jocul nu a prezentat crash-uri sau comportamente neprevăzute pe parcursul testelor. În ceea ce privește securitatea, datele din fișierul de salvare și fișierele cu detaliile nivelelor pot fi alterate sau corupte de utilizator sau alte programe, întrucât ele se află în folderul cu executabilul. Aceasta reprezintă o potențială vulnerabilitate, deoarece modificarea sau coruperea acestor fișiere poate duce la comportamente neașteptate în joc.

Pentru a evalua performanța jocului, am realizat mai multe teste pe diferite configurații hardware. În testele efectuate, jocul a rulat fără probleme pe un sistem cu un procesor Intel Core i5 de generația a 9-a, 8 GB RAM și o placă video NVIDIA GTX 1050. Aceste specificații reprezintă cerințele minime pentru rularea jocului la o performanță acceptabilă. Jocul a fost testat pe cinci sisteme diferite, având următoarele configurații:

- Sistem 1: NVIDIA GTX 1050, 8 GB RAM, HDD
- Sistem 2: NVIDIA GTX 1050 TI, 8 GB RAM, HDD
- Sistem 3: NVIDIA GTX 1650, 16 GB RAM, SSD
- Sistem 4: NVIDIA RTX 2060, 16 GB RAM, SSD
- Sistem 5: NVIDIA RTX 4060, 16 GB RAM, SSD

Concluzii

Scopul lucrării a fost de a dezvolta un joc serios pentru programatorii amatori, centrat pe învățarea și aplicarea mai multor algoritmi. Obiectivele inițiale au fost atinse în mare măsură , implementând un joc interactiv care să explice și să testeze cunoștințele utilizatorilor aupra câtorva algoritmi de sortare precum Selection Sort, Bubble Sort, Quicksort si Insertion Sort. Deși inițial s-au întâmpinat anumite dificultăți legate de implementarea funcționalităților jocului, de optimizări și de găsire a unor modalități de reprezentare în nivele a algoritmilor, soluțiile dezvoltate au permis împlinirea, în mare măsură, a scopului original al acestui proiect. Cu toate acestea, jocul ar fi putut să fie mai rafinat din punct de vedere vizual și ar fi putut sa dezvolte mai mulți algoritmi. De asemenea, există anumite probleme de securitate menționate in lucrare care ar trebui și ar putea fi rezolvate in viitor.

Ideea originală a proiectului provine din dorința de a ajuta programatorii începători să înțeleagă mai bine algoritmii de început pentru a crea o fundație solidă pentru viitoarele informații care urmează să fie clădite și de a face programarea mai atractivă pentru persoanele care doresc să încerce. Această dorință a fost influențată de impactul pe care l-a avut doamna profesoară de informatică din liceu asupra mea, asupra modului în care mi-am dezvoltat gândirea inginerească și a modului în care am început să văd informatica ca domeniu și materie.

Cea mai semnificativă contribuție adusă de mine este implementarea unui sistem de gestionare a nivelurilor care utilizează matrici pentru a monitoriza pozițiile și stările pieselor. De asemenea, majoritatea elementelor vizuale au fost concepute și gândite de mine, de la modul în care arată jocul ca stil și aspect, până la desenarea de mână a coloanelor folosite în nivel. Totodată, modul în care jocul salvează progresul și modul în care utilizatorul poate selecta nivele specifice pe care să le joace au fost gândite și implementate tot de mine. Acestea au dus la apariția anumitor probleme, precum problema de securitate menționată anterior. Problema de securitate a apărut ca o soluție la o altă problemă: resursele jocului nu erau încărcate și prelucrate de Unity în fișierul cu executabilul, astfel am fost nevoit să adaug aceste resurse manual în fișierul respectiv.

În comparație cu alte proiecte similare, aplicația dezvoltată se evidențiază prin integrarea detaliată a explicațiilor teoretice cu exercițiile practice interactive și exercițiile de testare. De exemplu, alte aplicații/jocuri serioase precum AlgoBot oferă vizualizări și explicații ale algoritmilor, dar lipsesc componentele care să testeze informațiile dobândite de utilizatori, iar Sort Visually oferă doar o modalitate vizuală de funcționare a mai multor algoritmi de sortare fără a interacționa în mod direct cu utilizatorul. În concluzie, proiectul dezvoltat oferă o experiență unică de învățare care este atât eficientă, cât și distractivă, chiar dacă jocurile similare din industrie oferă o experiență mai rafinată.

Direcții viitoare de dezvoltare

Deși jocul se află într-un stadiu bun, acestuia i se pot aduce îmbunătățiri. În viitor, jocul ar trebui să acopere o gamă mai variată de algoritmi de sortare, dar ar trebui să ilustreze și algoritmi mai complexi precum algoritmul lui Lee, BFS etc. și eventual explicații grafice pentru concepte complexe cu care începătorii în programare ar putea să aibă probleme, de exemplu conceptul de pointer. Din punct de vedere al aspectului, anumite părți din joc ar putea să primească un upgrade în calitatea grafică (coloanele utilizate în fundal, meniul de pauză, etc.). De asemenea, problemele de securitate a fișierelor cu resursele nivelelor și cu salvarea jocului trebuie rezolvate.

Acestea fiind spuse, proiectul are mult potențial, făcând parte dintr-o zonă a industriei care este în dezvoltare și bazându-se pe o idee creativă, distractivă, dar și originală. Cu suficientă implicare și determinare, acesta poate deveni chiar unul dintre liderii din această industrie.

Lecții învățate pe parcusul dezvoltării lucrării de diplomă

Dezvoltarea acestei lucrări a fost o experiență de învățare complexă și valoroasă. Pe parcursul proiectului, am învățat importanța planificării detaliate și a gestionării eficiente a timpului. Stabilirea unor obiective clare și a unui plan de acțiune bine structurat au fost esențiale pentru menținerea proiectului pe drumul cel bun și pentru evitarea întârzierilor.

De asemenea, am realizat cât de crucială este testarea în crearea unui joc. Testarea frecventă a codului și a funcționalităților a permis identificarea din timp a problemelor și erorilor, ceea ce a facilitat rezolvarea rapidă a acestora. Am învățat că testarea nu este doar o etapă finală a dezvoltării, ci un proces continuu care trebuie integrat în toate fazele proiectului.

Lucrarea de diplomă mi-a oferit oportunitatea de a descoperi și a învăța o varietate de lucruri. Printre cele mai importante se numără ocazia de a lucra cu un motor de jocuri precum Unity pentru a dezvolta această aplicație și ocazia de a experimenta/învăța într-o oarecare măsură procesul necesar dezvoltării unei aplicații cu potențial, de la zero. Această experiență m-a ajutat să îmi dezvolt abilitățile tehnice și să înțeleg mai bine complexitatea și dedicarea necesare pentru a crea un software educațional de succes.

Bibliografie

[1] "Children with adhd can now be prescribed a video game, fda says," https://edition.cnn.com/2020/06/16/health/adhd-fda-game-intl-scli-wellness/index.html, 2020, Ultima accesare: 24.06.2024.

[2] "Project references," https://beamng.tech/references/, Ultima accesare: 24.06.2024.

[3] T. S. P. of Life: Inventory and P. of Serious Games (for Health), "Lampert, claudia, christiane schwinge, and daniel tolks. 2009," *Media Education: Journal for Theory and Practice of Media Education 15 (Computer Games and Video Games):1-16.* [Online]. Available: https://www.medienpaed.com/article/view/104

[4] "Website of the project coordinator cevet - centre for vocational education and training," https://netenquiry.cevet.eu/, Ultima accesare: 07.11.2013.

[5] "The news game political edition," https://web.archive.org/web/20221214114037/https://www.sowaswillichauch.de/geschenk-spiele/the-news-game-political-edition/11704806.html, 2022, Ultima accesare: 24.06.2024.

[6] T. Holmes, "Arcade classics spawn art? current trends in the art game genre," Ultima accesare: 24.06.2024. [Online]. Available: https://web.archive.org/web/20130420092835/http://hypertext.rmit.edu.au/dac/papers/Holmes.pdf

[7] L. A. Annetta, J. Minogue, S. Y. Holmes, and M.-T. Cheng, "The effectiveness of serious games: A review of the literature," *Journal of Computers in Mathematics and Science Teaching*, vol. 29, no. 2, pp. 113–133, 2010.

[8] M. Prensky, *Digital game-based learning*. Computers in Entertainment (CIE), 2003.

[9] A. Gomes and A. Mendes, "Why is it so difficult to learn programming?" *Proceedings of the 2007 International Conference on Engineering Education*, pp. 283–288, 2007.

[10] Lernero, "Ce este gamificarea și cum poate fi integrată în învățare," 2024, accesat: 20 iunie 2024. [Online]. Available: https://lernero.ro/ce-este-gamificarea-si-cum-poate-fi-integrata-in-invatare/

[11] "Unity 2d master: Game development with c# and unity," https://www.udemy.com/course/unity-2d-master-game-development-with-csharp-and-unity/, Ultima accesare: 07.06.2023.

[12] N. A. Borromeo, *Hands-On Unity 2021 Game Development*. Packt Publishing, 2021.

[13] U. Technologies, "Scriptable render pipeline," 2021, Ultima accesare: 25.06.2024. [Online]. Available: https://docs.unity3d.com/Manual/ScriptableRenderPipeline.html

[14] ——, "Collider components," 2021, Ultima accesare: 25.06.2024. [Online]. Available: https://docs.unity3d.com/Manual/CollidersOverview.html

Referințe suplimentare

- Imaginea de fundal pentru meniul principal este generată și descărcată de pe https://deepai.org/machine-learning-model/fantasy-world-generator.
- Sunetele folosite în joc sunt descărcate de pe https://pixabay.com/.
- Imaginea gradient folosită ca fundal în aplicație este descărcată de pe https://unsplash.com/photos/light-blue-to-dark-blue-gradient-pJadQetzTkI.
- Iconițele folosite în meniul de pauză sunt descărcate de pe www.flaticon.com. Iconița pentru butonul "Home" și pentru "Save" sunt făcute de Hilmy Abiyuu A. Cele pentru butonul "Play" și pentru "Pause" de Graphix's Art. Iar iconița pentru butonul "Restart" de logisstudio.
- Aplicația utilizată pentru desenarea anumitor resurse este https://pyxeledit.com/.

Anexe

Anexa 1. Diagrama UML

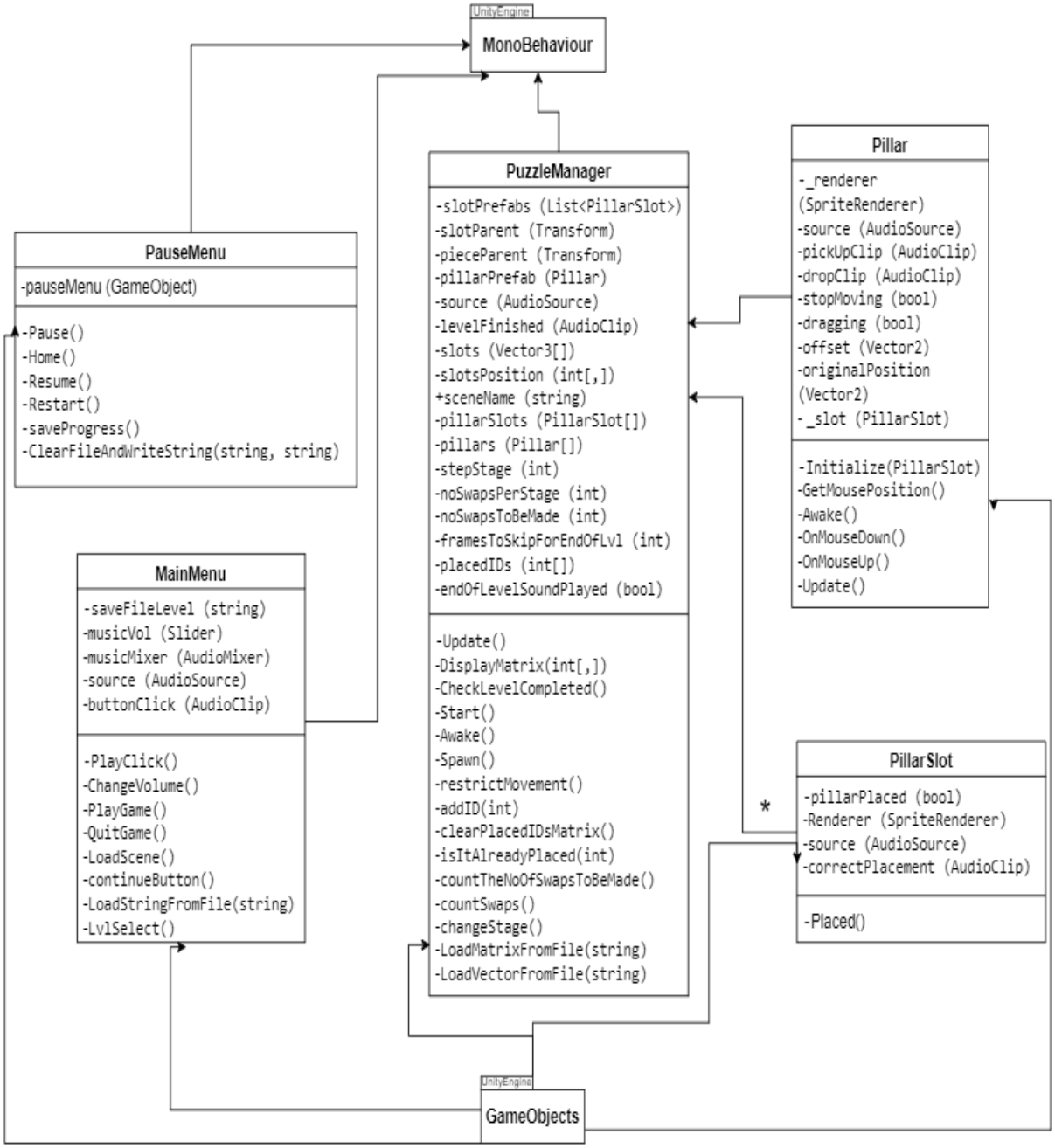


Figura A.1. Diagrama UML a proiectului

Anexa 2. Clasa PuzzleManager

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using System.IO;
4  using System.Linq;
5  using Unity.VisualScripting;
6  using UnityEngine;
7  using UnityEngine.SceneManagement;
8  using static Unity.VisualScripting.Member;
9
10 public class PuzzleManager : MonoBehaviour
11 {
12
13
14     [SerializeField] private List<PillarSlot> slotPrefabs;
15     [SerializeField] private Transform slotParent,pieceParent;
16     [SerializeField] private Pillar pillarPrefab;
17     [SerializeField] private AudioSource source;
18     [SerializeField] private AudioClip levelFinished;
19
20     private Vector3[] slots;
21
22
23     ///numarul de linii este egal cu numarul de pillere din nivel. numarul de co
24     ///numerele din matrice reprezinta pozitia in care trebuie sa fie slot-urile
25     ///slotul de pe linia i(in functie de marime-cel mai mic este primul) trebuie
26
27     private static int[,] slotsPosition;
28
29
30     string sceneName;
31
32
33     PillarSlot[] pillarSlots= new PillarSlot[5];
34     Pillar[] pillars = new Pillar[5];
35     private int stepStage = 0;
36     private int noSwapsPerStage = 0;
37     private int noSwapsToBeMade = 0;
38
39     private int[] placedIDs = { -1, -1, -1, -1, -1, -1, -1, -1, -1, -1 };
40     private bool endOfLevelSoundPlayed = false;
41
42
43
44
45     void Update()
46     {
47
48
49         CheckLevelCompleted();
50
51         //1. restrictionam ce piese pot fi mutate
52         restrictMovement();
```

```
53          //2. verificam daca jucatorul a facut swapp-ul corect
54          countSwaps();
55          //3. daca jucatorul a facut swappul-trecem la urmatorul stage/step in 
56          changeStage();
57
58      }
59
60      private void DisplayMatrix(int[,] matrix)
61      {
62          for (int i = 0; i < matrix.GetLength(0); i++)
63          {
64              string row = "";
65              for (int j = 0; j < matrix.GetLength(1); j++)
66              {
67                  row += matrix[i, j] + " ";
68              }
69              Debug.Log(row);
70          }
71      }
72
73      //Verifica daca nivelul este complet(adica daca in matricea slotsPosition 
74      //Daca toate valorile sunt 0 asteapta un numar de frameuri pentru a se auz
75      //si dupa schimba scene-ul
76      private void CheckLevelCompleted()
77      {
78          int ok = 1;
79          for (int i = 0; i < slotPrefabs.Count(); i++)
80          {
81              if (slotsPosition[i, stepStage] != 0)
82              {
83                  ok = 0;
84
85              }
86          }
87          if (ok == 1)
88          {
89              if (endOfLevelSoundPlayed == false)
90              {
91                  source.PlayOneShot(levelFinished);
92                  endOfLevelSoundPlayed = true;
93              }
94
95              StartCoroutine(WaitAndLoadNextLevel(4f));
96          }
97      }
98      private IEnumerator WaitAndLoadNextLevel(float waitTime)
99      {
100         yield return new WaitForSeconds(waitTime);
101         SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
102     }
103
104     private void Start()
105     {
```

```
106
107             Spawn();
108         }
109
110         //Load the data for each level
111         private void Awake()
112         {
113             Scene currentScene = SceneManager.GetActiveScene();
114             sceneName = currentScene.name;
115             if (sceneName == "Level1")
116             {
117                 slotsPosition = LoadMatrixFromFile("Assets/Scripts/LevelStates/slotsP
118                 slots = LoadVectorFromFile("Assets/Scripts/LevelStates/slotsLv1.txt")
119                 //DisplayMatrix(slotsPosition);//used for debugging
120             }
121             else if (sceneName == "Level2")
122             {
123                 slots = LoadVectorFromFile("Assets/Scripts/LevelStates/slotsLv2.txt")
124                 slotsPosition = LoadMatrixFromFile("Assets/Scripts/LevelStates/slotsP
125             }
126             else if (sceneName == "Level3")
127             {
128                 slots = LoadVectorFromFile("Assets/Scripts/LevelStates/slotsLv3.txt")
129                 slotsPosition = LoadMatrixFromFile("Assets/Scripts/LevelStates/slotsP
130             }
131             else if (sceneName == "Level4")
132             {
133                 slots = LoadVectorFromFile("Assets/Scripts/LevelStates/slotsLv4.txt")
134                 slotsPosition = LoadMatrixFromFile("Assets/Scripts/LevelStates/slotsP
135             }
136             else if (sceneName == "Level5")
137             {
138                 slots = LoadVectorFromFile("Assets/Scripts/LevelStates/slotsLv5.txt")
139                 slotsPosition = LoadMatrixFromFile("Assets/Scripts/LevelStates/slotsP
140             }
141             else if (sceneName == "Level6")
142             {
143                 slots = LoadVectorFromFile("Assets/Scripts/LevelStates/slotsLv6.txt")
144                 slotsPosition = LoadMatrixFromFile("Assets/Scripts/LevelStates/slotsP
145             }
146             else if (sceneName == "Level7")
147             {
148                 slots = LoadVectorFromFile("Assets/Scripts/LevelStates/slotsLv7.txt")
149                 slotsPosition = LoadMatrixFromFile("Assets/Scripts/LevelStates/slotsP
150             }
151             else if (sceneName == "Level8")
152             {
153                 slots = LoadVectorFromFile("Assets/Scripts/LevelStates/slotsLv8.txt")
154                 slotsPosition = LoadMatrixFromFile("Assets/Scripts/LevelStates/slotsP
155             }
156             else if (sceneName == "Level9")
157             {
158                 slots = LoadVectorFromFile("Assets/Scripts/LevelStates/slotsLv9.txt")
```

```
159             slotsPosition = LoadMatrixFromFile("Assets/Scripts/LevelStates/slot
160         }
161         else if (sceneName == "Level10")
162         {
163             slots = LoadVectorFromFile("Assets/Scripts/LevelStates/slotsLv10.tx
164             slotsPosition = LoadMatrixFromFile("Assets/Scripts/LevelStates/slot
165         }
166         else if (sceneName == "Level11")
167         {
168             slots = LoadVectorFromFile("Assets/Scripts/LevelStates/slotsLv11.tx
169             slotsPosition = LoadMatrixFromFile("Assets/Scripts/LevelStates/slot
170         }
171         else if (sceneName == "Level12")
172         {
173             slots = LoadVectorFromFile("Assets/Scripts/LevelStates/slotsLv12.tx
174             slotsPosition = LoadMatrixFromFile("Assets/Scripts/LevelStates/slot
175         }
176         else if (sceneName == "Level13")
177         {
178             slots = LoadVectorFromFile("Assets/Scripts/LevelStates/slotsLv13.tx
179             slotsPosition = LoadMatrixFromFile("Assets/Scripts/LevelStates/slot
180         }
181         else if (sceneName == "Level14")
182         {
183             slots = LoadVectorFromFile("Assets/Scripts/LevelStates/slotsLv14.tx
184             slotsPosition = LoadMatrixFromFile("Assets/Scripts/LevelStates/slot
185         }
186         else if (sceneName == "Level15")
187         {
188             slots = LoadVectorFromFile("Assets/Scripts/LevelStates/slotsLv15.tx
189             slotsPosition = LoadMatrixFromFile("Assets/Scripts/LevelStates/slot
190         }
191         else if (sceneName == "Level16")
192         {
193             slots = LoadVectorFromFile("Assets/Scripts/LevelStates/slotsLv16.tx
194             slotsPosition = LoadMatrixFromFile("Assets/Scripts/LevelStates/slot
195         }
196         //inumaram cate swapuri trebuie sa faca in primul stage
197         countTheNoOfSwapsToBeMade();
198 
199     }
200 
201 
202     void Spawn()
203     {
204         var Set = slotPrefabs.ToList();
205         for(int i = 0; i < Set.Count(); i++)
206         {
207             pillarSlots[i] = Instantiate(Set[i], slotParent.GetChild(i).positio
208 
209             pillars[i]= Instantiate(pillarPrefab, pieceParent.GetChild(i).posit
210             pillars[i].transform.localScale = slotParent.GetChild(i).localScale
211             pillars[i].Initialize(pillarSlots[i]);
```

```
212             }
213
214         }
215
216         //Functia restrictioneaza miscarea unei piese care a fost plasata corect sau
217         void restrictMovement()
218         {
219             for (int i = 0; i < slotPrefabs.Count(); i++)
220             {
221                 if (pillarSlots[i].pillarPlaced == true)
222                 {
223                     pillars[i].stopMoving = true;
224                 }
225                 else
226                 {
227                     if (slotsPosition[i, stepStage] == 0)
228                     {
229                         pillars[i].stopMoving = true;
230                     }
231                     else
232                     {
233                         pillars[i].stopMoving = false;
234                     }
235                 }
236
237             }
238         }
239
240         //Adauga un ID al unui slot la matricea care tine minte ID-urile tuturor pil
241         void addID(int i)
242         {
243             for(int j = 0; j < placedIDs.Length; j++)
244             {
245                 if (placedIDs[j] == -1)
246                 {
247                     placedIDs[j] = i;
248                     break;
249                 }
250             }
251
252         }
253
254         //Imi goleste matricea PlacedIDs pentru urmatorul stage
255         void clearPlacedIDsMatrix()
256         {
257             for (int j = 0; j < placedIDs.Length; j++)
258             {
259                 placedIDs[j] = -1;
260             }
261
262         }
263
264         //verifica daca pillarul respectiv a fost plasat acum sau mai demult pentru a
```

```
265         bool isItAlreadyPlaced(int i)
266         {
267             for (int j = 0; j < placedIDs.Length; j++)
268             {
269                 if (placedIDs[j] == i)
270                     return true;
271             }
272 
273             return false;
274         }
275 
276 
277         //In functia aceasta numar cate swapuri trebuie sa faca jucatorul in stage
278         void countTheNoOfSwapsToBeMade()
279         {
280             for (int i = 0; i < slotPrefabs.Count(); i++)
281             {
282                 if (slotsPosition[i, stepStage] != 0)
283                     noSwapsToBeMade++;
284             }
285         }
286 
287 
288 
289         //Functia asta inumara cate swapuri a facut jucatorul fata de cate trebuie
290         void countSwaps()
291         {
292             for (int i = 0; i < slotPrefabs.Count(); i++)
293             {
294                 if (pillarSlots[i].pillarPlaced == true && slotsPosition[i, stepSta
295                 //BUG: trebuie verificat si noSwaps sa fie mai mic decat 2 deoarec
296                 //atunci cand plasezi intai pillarul mai mare si dupa cel mai mic 
297                 //cel mai probabil inumara un swap de 2 ori asa ca nu mai intra in
298                 {
299                     noSwapsPerStage++;
300                     addID(i);
301                     //Debug.Log("LastplacedID : "+ lastPlacedID);
302                     //Debug.Log("NoSwaps : "+noSwapsPerStage);
303 
304                 }
305             }
306         }
307 
308 
309         //functia aceasta avanseaza in urmatorul pas al algoritmului atunci cand s
310         void changeStage()
311         {
312 
313             //o sa modific conditia din noSwapsPerStage==2 in noSwapsPerStage>=2 p
314             //Pt a implementa nivele cu InsertSort trebuie sa schimb noSwapsPerSta
315             if (noSwapsPerStage >= noSwapsToBeMade && noSwapsToBeMade!=0)
316             {
317                 noSwapsPerStage = 0;
```

```
318                 noSwapsToBeMade = 0;
319                 clearPlacedIDsMatrix();
320                 //Debug.Log("LastplacedID : " + lastPlacedID);
321                 //Debug.Log("NoSwaps : " + noSwapsPerStage);
322                 stepStage++;
323                 countTheNoOfSwapsToBeMade();
324                 for (int i = 0; i < slotPrefabs.Count(); i++)
325                 {
326                     if (slotsPosition[i, stepStage] != 0)
327                     {
328                         //aici se muta pozitiile slot.urilor in functie de pasul urma
329                         pillarSlots[i].transform.position = slots[slotsPosition[i, st
330 
331                     }
332                     pillarSlots[i].pillarPlaced = false;
333                     pillars[i].stopMoving = false;
334                 }
335             }
336 
337         }
338 
339         //Functia aceasta este folosita pentru a citi matricea de gestionare a nivelu
340         private int[,] LoadMatrixFromFile(string filename)
341         {
342             string[] lines = File.ReadAllLines(filename);
343             int rowCount = lines.Length;
344             int colCount = lines[0].Split(',').Length;
345             int[,] matrix = new int[rowCount, colCount];
346 
347             for (int i = 0; i < rowCount; i++)
348             {
349                 string[] values = lines[i].Split(',');
350                 for (int j = 0; j < colCount; j++)
351                 {
352                     matrix[i, j] = int.Parse(values[j]);
353                 }
354             }
355 
356             return matrix;
357         }
358 
359         //Functia aceasta este folosita pentru a citi vectorii de pozitie a slot-uril
360         private Vector3[] LoadVectorFromFile(string filename)
361         {
362             string[] lines = File.ReadAllLines(filename);
363             Vector3[] vectors = new Vector3[lines.Length];
364 
365             for (int i = 0; i < lines.Length; i++)
366             {
367                 string[] values = lines[i].Split(',');
368                 vectors[i] = new Vector3(float.Parse(values[0]), float.Parse(values[1
369             }
370 
```

```
371             return vectors;
372         }
373
374     }
```

Listing 1. clasa *PuzzleManager*

Anexa 3. Clasa PauseMenu

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.SceneManagement;
5  using System.IO;
6
7  public class PauseMenu : MonoBehaviour
8  {
9
10     [SerializeField] GameObject pauseMenu;
11     public void Pause()
12     {
13         pauseMenu.SetActive(true);
14         Time.timeScale = 0;
15     }
16
17     public void Home()
18     {
19         SceneManager.LoadScene("MainMenu");
20         Time.timeScale = 1;
21     }
22
23     public void Resume()
24     {
25         pauseMenu.SetActive(false);
26         Time.timeScale = 1;
27     }
28
29     public void Restart()
30     {
31         SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
32         Time.timeScale = 1;
33     }
34
35
36     public void saveProgress()
37     {
38         int currentScene = SceneManager.GetActiveScene().buildIndex;
39         ClearFileAndWriteString("Assets/Scripts/SaveFiles/SaveFile.txt", currentS
40
41
42     }
43
44     private void ClearFileAndWriteString(string filename, string content)
45     {
46         try
47         {
48             // Functia asta creaza fisierul daca nu exista si da overwrite la con
49             File.WriteAllText(filename, content);
50             Debug.Log("Cleared file and wrote string: " + content + " into " + fi
51         }
52         catch (System.Exception e)
```

```
53              {
54                  Debug.LogError("Error writing to file " + filename + ": " + e.Messa
55              }
56          }
57
58
59      }
```

Listing 2. clasa *PauseMenu*

Anexa 4. Clasa Pillar

```
1       using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class Pillar : MonoBehaviour
6   {
7       [SerializeField] private SpriteRenderer _renderer;
8
9       [SerializeField] private AudioSource source;
10      [SerializeField] private AudioClip pickUpClip, dropClip;
11
12      //i want to use this in the future so that the pillar cant be moved after it
13      public bool stopMoving=false;
14
15      private bool dragging;
16
17      //this is used so that when u pick up the pillar the middle of the pillar doe
18      private Vector2 offset;
19
20      //this is used so that when u drop the pillar it will return to it's original
21      private Vector2 originalPosition;
22
23      private PillarSlot _slot;
24
25      public void Initialize(PillarSlot slot)
26      {
27          _renderer.sprite=slot.Renderer.sprite;
28          _slot = slot;
29
30      }
31
32      // Start is called before the first frame update
33      void Start()
34      {
35          //because we dont have a game object for every pillar here we initialize
36          transform.localScale = _slot.transform.localScale;
37      }
38
39      Vector2 GetMousePosition()
40      {
41          return Camera.main.ScreenToWorldPoint(Input.mousePosition);
42      }
43
44
45      void Awake()
46      {
47          originalPosition = transform.position;
48      }
49
50      void OnMouseDown()
51      {
52          if (stopMoving) return;
```

```
53              dragging = true;
54              source.PlayOneShot(pickUpClip);
55              offset = GetMousePosition() - (Vector2)transform.position;
56          }
57
58          void OnMouseUp()
59          {
60              if (stopMoving) return;
61              if (Vector2.Distance(transform.position, _slot.transform.position) < 1
62              {
63                  _slot.Placed();
64                  dragging = false;
65                  transform.position = _slot.transform.position;
66              }
67              else
68              {
69                  source.PlayOneShot(dropClip);
70                  dragging = false;
71              }
72          }
73
74
75          void Update()
76          {
77              if(stopMoving) return;
78              if (!dragging) return;
79
80              var mousePosition = GetMousePosition();
81
82              transform.position = mousePosition - offset;
83
84          }
85
86
87      }
```

Listing 3. clasa *Pillar*

Anexa 5. Clasa MainMenu

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using UnityEngine.UIElements;
using UnityEngine.Audio;
using System.IO;

public class MainMenu : MonoBehaviour
{
    public string saveFileLevel;
    public UnityEngine.UI.Slider musicVol;
    public AudioMixer musicMixer;
    public bool advancedSkill;

    [SerializeField] private AudioSource source;
    [SerializeField] private AudioClip buttonClick;

    public void PlayClick()
    {
        source.PlayOneShot(buttonClick);
    }

    public void ChangeVolume()
    {
        musicMixer.SetFloat("VolumeParameter",musicVol.value);
    }

    public void PlayGame()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
    }

    public void QuitGame()
    {
        Application.Quit();
    }

    public void LoadScene()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + int.Par

    }
    public void continueButton()
    {

        LoadStringFromFile("Assets/Scripts/SaveFiles/SaveFile.txt");
        if(saveFileLevel!="")
        {
            LoadScene();
        }
```

```
53          else
54          {
55              PlayGame();
56          }
57
58      }
59
60
61
62
63      private string LoadStringFromFile(string filename)
64      {
65          try
66          {
67
68              string content = File.ReadAllText(filename);
69              saveFileLevel = content;
70              Debug.Log("Loaded string from file: " + filename);
71              return content;
72          }
73          catch (System.Exception e)
74          {
75              Debug.LogError("Error reading file " + filename + ": " + e.Message
76              return null;
77          }
78      }
79
80      public void advancedLevel()
81      {
82          advancedSkill = true;
83          //Debug.Log(advancedSkill);
84      }
85
86      public void beginnerLevel()
87      {
88          advancedSkill = false;
89          //Debug.Log(advancedSkill);
90      }
91
92      //Functii pentru level Selector
93      public void Lv1Select()
94      {
95          int savedSceneIndex = int.Parse(File.ReadAllText("Assets/Scripts/SaveF
96          if (savedSceneIndex >= 2  advancedSkill==true)
97              SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 
98
99      }
100     public void Lv2Select()
101     {
102         int savedSceneIndex = int.Parse(File.ReadAllText("Assets/Scripts/SaveF
103         if (savedSceneIndex >= 5  advancedSkill == true)
104             SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 
105
```

```
106         }
107         public void Lv3Select()
108         {
109             int savedSceneIndex = int.Parse(File.ReadAllText("Assets/Scripts/SaveFile
110             if (savedSceneIndex >= 7  advancedSkill == true)
111                 SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 7);
112 
113         }
114         public void Lv4Select()
115         {
116 
117             int savedSceneIndex = int.Parse(File.ReadAllText("Assets/Scripts/SaveFile
118             if (savedSceneIndex >= 9  advancedSkill == true)
119                 SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 9);
120 
121         }
122         public void Lv5Select()
123         {
124             int savedSceneIndex = int.Parse(File.ReadAllText("Assets/Scripts/SaveFile
125             if (savedSceneIndex >= 12  advancedSkill == true)
126                 SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 12)
127 
128         }
129         public void Lv6Select()
130         {
131             int savedSceneIndex = int.Parse(File.ReadAllText("Assets/Scripts/SaveFile
132             if (savedSceneIndex >= 14  advancedSkill == true)
133                 SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 14)
134 
135         }
136         public void Lv7Select()
137         {
138             int savedSceneIndex = int.Parse(File.ReadAllText("Assets/Scripts/SaveFile
139             if (savedSceneIndex >= 16  advancedSkill == true)
140                 SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 16)
141 
142         }
143         public void Lv8Select()
144         {
145             int savedSceneIndex = int.Parse(File.ReadAllText("Assets/Scripts/SaveFile
146             if (savedSceneIndex >= 19  advancedSkill == true)
147                 SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 19)
148 
149         }
150         public void Lv9Select()
151         {
152             int savedSceneIndex = int.Parse(File.ReadAllText("Assets/Scripts/SaveFile
153             if (savedSceneIndex >= 21  advancedSkill == true)
154                 SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 21)
155 
156         }
157         public void Lv10Select()
158         {
```

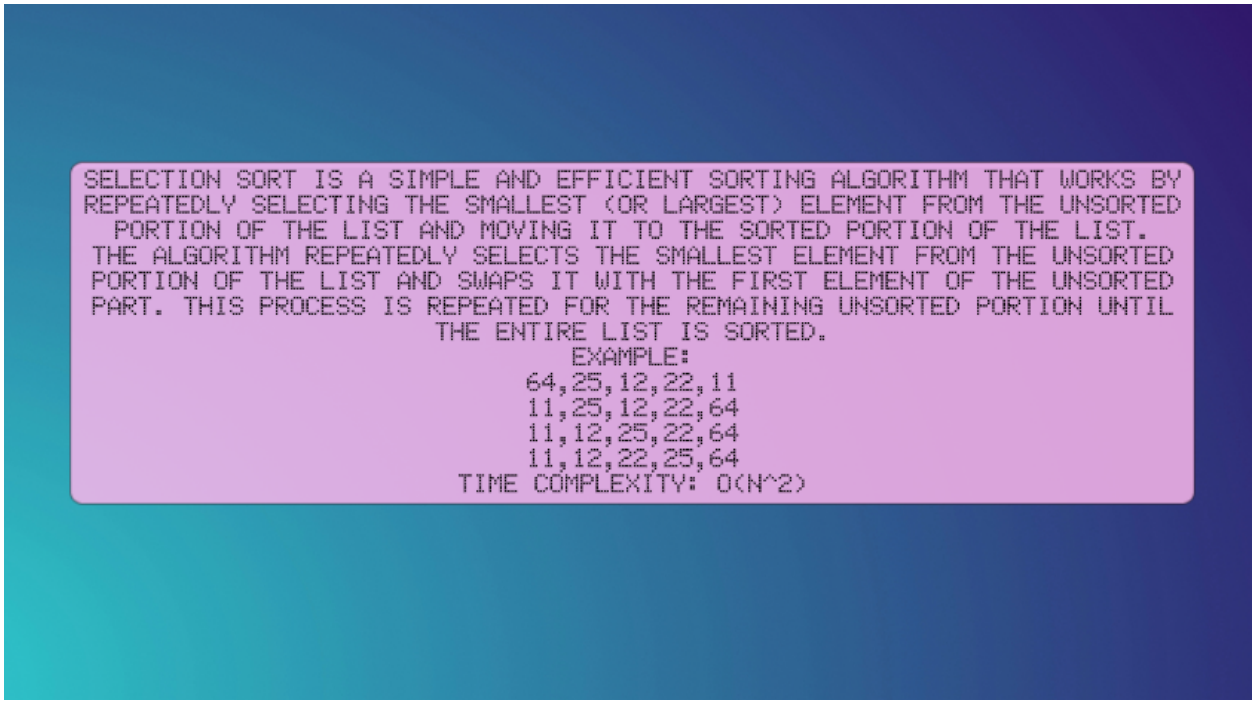
```
159             int savedSceneIndex = int.Parse(File.ReadAllText("Assets/Scripts/SaveF
160             if (savedSceneIndex >= 23  advancedSkill == true)
161                 SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 
162 
163         }
164         public void Lv11Select()
165         {
166             int savedSceneIndex = int.Parse(File.ReadAllText("Assets/Scripts/SaveF
167             if (savedSceneIndex >= 26  advancedSkill == true)
168                 SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 
169 
170         }
171         public void Lv12Select()
172         {
173             int savedSceneIndex = int.Parse(File.ReadAllText("Assets/Scripts/SaveF
174             if (savedSceneIndex >= 28  advancedSkill == true)
175                 SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 
176 
177         }
178         public void Lv13Select()
179         {
180             int savedSceneIndex = int.Parse(File.ReadAllText("Assets/Scripts/SaveF
181             if (savedSceneIndex >= 30  advancedSkill == true)
182                 SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 
183 
184         }
185         public void Lv14Select()
186         {
187             int savedSceneIndex = int.Parse(File.ReadAllText("Assets/Scripts/SaveF
188             //Debug.Log(advancedSkill);
189             if (savedSceneIndex >= 33  advancedSkill == true)
190                 SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 
191 
192         }
193         public void Lv15Select()
194         {
195             int savedSceneIndex = int.Parse(File.ReadAllText("Assets/Scripts/SaveF
196             if (savedSceneIndex >= 35  advancedSkill == true)
197                 SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 
198 
199         }
200         public void Lv16Select()
201         {
202             int savedSceneIndex = int.Parse(File.ReadAllText("Assets/Scripts/SaveF
203             if (savedSceneIndex >= 37  advancedSkill == true)
204                 SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 
205 
206         }
207 
208     }
```

Listing 4. clasa *MainMenu*

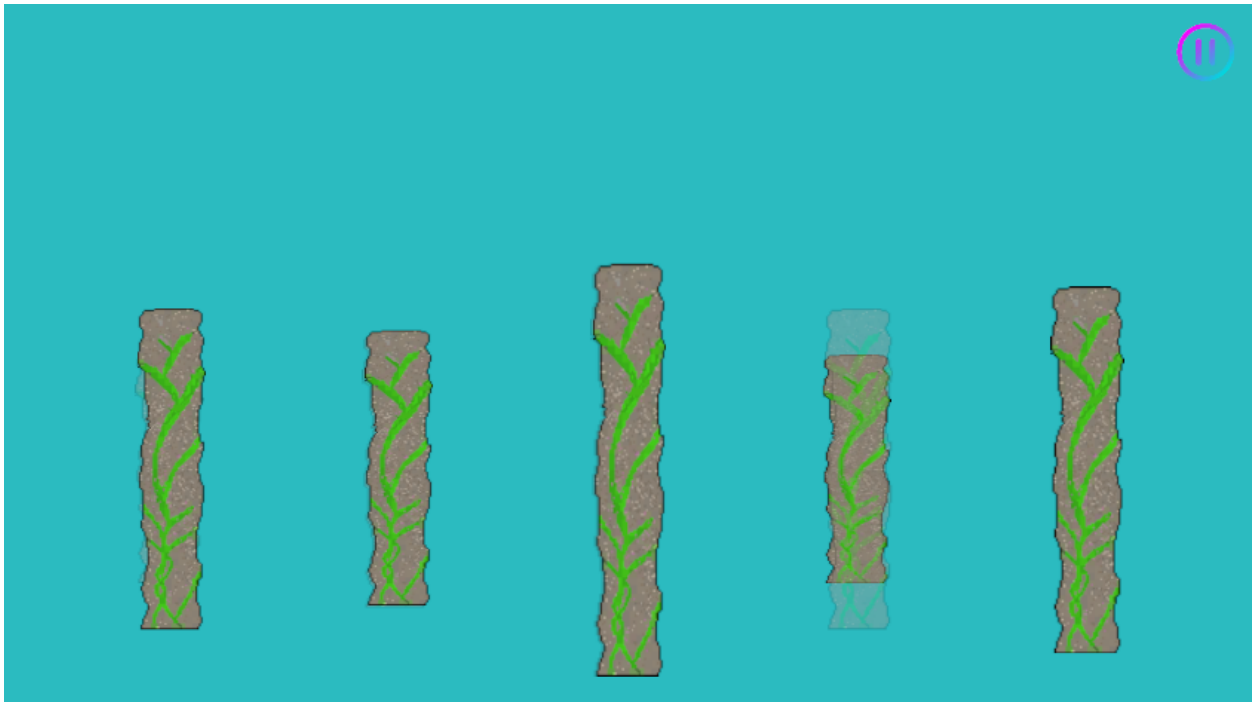
Anexa 6. Meniul Principal

Anexa 7. Select Level

Anexa 8. Explicație algoritm



Anexa 9. Exemplu de nivel



Anexa 10. Meniu pauză