UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Generare Procedurală Folosind Modele Markov

propusă de

Alexandru Loghin

Sesiunea: iulie, 2019

Coordonator științific

Conf. Dr. Anca Vitcu

UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

Generare Procedurală Folosind Modele Markov

Alexandru Loghin

Sesiunea: iulie, 2019

Coordonator științific

Conf. Dr. Anca Vitcu

Avizat,

Îndrumător lucrare de licență,

Conf. Dr. Anca Vitcu.

Data: ..... Semnătura: .....

Declarație privind originalitatea conținutului lucrării de licență

Subsemnatul **Loghin Alexandru** domiciliat în **România, jud. Iași, mun. Iași, calea Buzăului, nr. 25, bl. A, et. 5, ap. 45**, născut la data de **01 ianuarie 2018**, identificat prin CNP **1234567891234**, absolvent al Facultății de informatică, **Facultatea de informatică** specializarea **informatică**, promoția 2019, declar pe propria răspundere cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art. 143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență cu titlul **Generare Procedurală Folosind Modele Markov** elaborată sub îndrumarea domnului **Conf. Dr. Anca Vitcu**, pe care urmează să o susțin în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului ei într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diplomă sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Data: ..... Semnătura: .....

Declarație de consimțământ

Prin prezenta declar că sunt de acord ca lucrarea de licență cu titlul **Generare Procedurală**
Folosind Modele Markov, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test, etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de informatică.

De asemenea, sunt de acord ca Facultatea de informatică de la Universitatea ”Alexandru-Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Absolvent **Alexandru Loghin**

Data: ..... Semnătura: .....

Cuprins

Motivație **2**

Introducere **3**

1 Aspecte teoretice **4**

1.1 Introducere 4

1.2 Modele Markov cu stări invizibile 5

1.3 Cele mai importante probleme computaționale 5

1.3.1 Probabilitatea de a observa o secvență 5

1.3.2 Găsirea celor mai bune stări ce descriu secvența observată 6

1.3.3 Estimare de parametri 6

1.4 Limitări ale modelului 6

1.5 Algoritmi ce tratează problema estimării de parametri 7

1.6 Complexitate și metode de optimizare 7

2 Tehnologii **8**

2.1 Structura proiectului 9

2.2 Componente 10

2.2.1 MenuController 12

2.2.2 GameController 13

2.2.3 PlatformBuilder 14

2.2.4 PlatformController 14

3 Arhitectura aplicației **16**

Concluzii **17**

Bibliografie **18**

Motivație

Datorită pasiunii mele pentru *gamedev* și algoritmică am decis să realizez o modalitate de a genera procedural un mediu cât mai diversificat. Folosirea modelelor Markov a venit din necesitatea de a scăpa cumva de metoda clasică prin care se generează un șir de obiecte într-un mod aleator iar apoi se încearcă validarea acesteia după anumite constrângeri. Folosind acest model stocastic putem controla apariția unei anumite subsecvențe prin stabilirea unei anumite probabilități de a se produce așadar eliminând cu totul pasul de validare.

Scopul principal al proiectului a fost dezvoltarea unei librări ce ușurează procesul de generare. De asemenea a fost nevoie și de un sistem ce facilitează plasarea obiectelor în mediul 3D , modelul Markov fiind folosit doar ca și generator. Odată funcționale mi-am îndreptat atenția asupra interfeței grafice și a interacțiunii dintre jucător și joc.

Datorită motorului grafic folosit și a instrumentelor de care dispune , timpul necesar pentru crearea jocului a fost îmbunătățit substanțial. De asemena cu prilejul acestei aplicații am avut posibilitatea de a aprofunda și testa diferite tehnici de randare și postprocesare. Câteva dintre aceste elemente includ SSR [1] , lumina volumetrică , reflexii în timp real , corectare de culoare și multe altele.

Odată cu finalizarea acestei aplicații , din cauza naturii sale *open source* , se va putea folosi librăria creată în cadrul acesteia pentru orice tip de proiect fără constrăngeri , fiind ușor adaptabilă pentru orice sarcină ce necesită generare procedurală , de la muzică până la construcția unui mediu virtual, librăria find ușor de folosit și axata pe eficiență.

BAGĂ CE ADUCE NOU APLICAȚIA ȘI CE AM INCERCAT!!!

RESCALEAZĂ BUCĂȚILE DE COD ȘI IMAGINILE

[1] Screen Space Reflection

Introducere

Având în vedere necesitatea creării unor medii cât mai versatile ce respectă anumite condiții sau restricții fizice s-au dezvoltat anumite tehnici pentru a facilita construcția automată și randomizată a oricărui element ce intră în componența unui joc , de la modele tridimensionale până la coloana sonoră.

Am decis așadar să încerc o abordare nouă , folosindu-mă de un model Markov ce a fost antrenat să respecte anumite reguli pentru a genera spațiul 3D. Scopul acestei lucrări este de a aduce o nouă perspectivă asupra generării procedurale și de a demonstra validitatea acestei abordări.

Tema lucrării s-a ivit din cauza necesității de adaptare rapidă a condițiilor folosite pentru generarea obiectelor din componența unui joc. Unul din avantajele acestei abordări este flexibilitatea oferită de modelul Markov fiind capabil să adapteze constrăngerile folosite la generare cu o simplă reantrenare a modelului.

Aplicația prezentă este construită folosind un motor grafic peste care am dezvoltat o librărie capabilă să reprezinte un model Markov și să îl antreneze. Folosindu-mă de această librărie , pot genera o secvență de obiecte ce încearcă să respecte fidel constrângerile folosite la antrenare.

Procesul de generare procedurală este compus din două etape. Generarea obiectelor și plasarea lor convenabil în spațiul virtual. Am încercat în special să decuplez cele două etape pentru o mai bună modularizare și o coeziune ridicată. Cele două sisteme lucrează independent unul față de celălalt , relația dintre cele două find una de agregare.

Cele două sisteme sunt puse în funcțiune pentru a genera în timp real mediul pentru jucător, mediu ce este împărțit în trei zone de interes , urbana , rurală și deșertică. De asemenea pentru a adauga complexitate se alternează între trei tipuri de platforme ce simulează un drum drept, un drum cu viraj la stănga sau un drum cu viraj la dreapta.

Capitolul 1

Aspecte teoretice

1.1 Introducere

În domeniul probabilităților un model Markov este folosit pentru a modela un sistem ce se schimbă într-un mod aleator. De obicei acesta ține cont doar de starea curentă și nu depinde de evenimentele aleatoare , acest lucru numindu-se și prorietatea Markov. Ulterior s-au dezvoltat modele ce au un așa numit ordin , extinzând astfel numărul de stări de care se ține cont în cadrul modelului.

De obicei există o separație clară între tipurile de modele Markov, criteriul folosit este disponibilitatea de a observa sau nu stările modelului. Așadar rezultă două mari categorii, sisteme cu stări complet observabile , din care fac parte lanțurile Markov și sisteme cu stări parțial observabile, cel mai cunoscut sistem fiind modelul Markov cu stări invizibile sau HMM [1]. Pe langă aceste modele prezentate, s-au dezvoltat și anumite derivate fiecare cu avantajele sale demonstrând astfel flexibilitatea și capacitatea de exprimare a acestor sisteme.

În mare parte această teză se axează în jurul parții discrete a acestor modele, numărul de stări/observații find finit numărabile și ușor de caracterizat dar există și o ramură ce se ocupă cu partea continuă a acestor modele, fiecare abordare având avantajele și dezavantajele ei.

Legat de partea practică, de-a lungul timpului aceste modele au fost folosite în foarte multe domenii de la bioinformatică până la lingvistică computațională. O aplicație foarte importantă a acestor modele este recunoașterea vocală, modelele Markov fiind standardul folosit în industrie pentru asistenții personali ca *Siri* și *Alexa*.

[1] Hidden Markov model

1.2 Modele Markov cu stări invizibile

Acest model statistic este caracterizat în principal de faptul că stările interne sunt ascunse de un privitor exterior. Tot ce se poate observa în cadrul acestui model este emisia unor etichete sau obiecte $\{v_1, v_2, v_3, \ldots, v_n\}$ dintr-o mulțime notată pe scurt V. Acest lucru complică în general structura modelului și algortimii ce folosesc acest sistem. Un avantaj direct este creșterea capacității de expresivitate, putând modela mai fidel datele, datorită eliminării necesității de a cunoaște toate stările evenimentului.

Pentru a caracteriza concret și complet acest model avem nevoie de un *5-uplu* $HMM = (\mathbf{S}, \mathbf{V}, \mathbf{A}, \mathbf{B}, \pi)$ ce desemnează astfel :

- $\mathbf{S} = \{S_1, S_2, \ldots, S_N\}$ mulțimea ce desemnează numărul de stări ascunse din model, avand un numar de N elemente. Starea relativa la timpul t se va nota ca si q_t.
- $\mathbf{V} = \{V_1, V_2, \ldots, V_M\}$, cele M etichete observabile.
- $\mathbf{A} = \{a_{ij}\}$ unde $a_{i,j} = P(q_{t+1} = S_j | q_t = S_i), 1 \leq i, j \leq N$.
- $\mathbf{B} = \{b_i(k)\}$ unde $b_i(k) = P(v_k \; la \; timpul \; t \; | q_t = S_i), 1 \leq i \leq N, 1 \leq k \leq M$.
- $\pi = \{\pi_i\}$ unde $\pi_i = P(q_1 = S_i), 1 \leq i \leq N$.

1.3 Cele mai importante probleme computaționale

Aplicația prezentă este împărțită pe diferite module ce controlează sunetul, generarea, *UI-ul* , instanțierea și multe altele. În **Unity** acest lucru se realizeză în mare parte prin scripturi atașate de obiectele din scena curentă.

Am optat să construiesc jocul în jurul unei singure scene, ceea ce necesită un numar mai mare de controalere și scripturi. O consecință directă a folosirii unei singure scene este integrarea meniului principal în același mediu ca și jocul în sine. De aici derivă necesitatea de două scripturi ce controlează tranziția de la meniu la joc intitulate sugestiv *MenuController* și *GameController*.

1.3.1 Probabilitatea de a observa o secvență

Aplicația prezentă este împărțită pe diferite module ce controlează sunetul, generarea, *UI-ul* , instanțierea și multe altele. În **Unity** acest lucru se realizeză în mare parte

prin scripturi atașate de obiectele din scena curentă.

Am optat să construiesc jocul în jurul unei singure scene, ceea ce necesită un numar mai mare de controalere și scripturi. O consecință directă a folosirii unei singure scene este integrarea meniului principal în același mediu ca și jocul în sine. De aici derivă necesitatea de două scripturi ce controlează tranziția de la meniu la joc intitulate sugestiv *MenuController* și *GameController*.

1.3.2 Găsirea celor mai bune stări ce descriu secvența observată

Aplicația prezentă este împărțită pe diferite module ce controlează sunetul, generarea, *UI-ul* , instanțierea și multe altele. În **Unity** acest lucru se realizeză în mare parte prin scripturi atașate de obiectele din scena curentă.

Am optat să construiesc jocul în jurul unei singure scene, ceea ce necesită un numar mai mare de controalere și scripturi. O consecință directă a folosirii unei singure scene este integrarea meniului principal în același mediu ca și jocul în sine. De aici derivă necesitatea de două scripturi ce controlează tranziția de la meniu la joc intitulate sugestiv *MenuController* și *GameController*.

1.3.3 Estimare de parametri

Aplicația prezentă este împărțită pe diferite module ce controlează sunetul, generarea, *UI-ul* , instanțierea și multe altele. În **Unity** acest lucru se realizeză în mare parte prin scripturi atașate de obiectele din scena curentă.

Am optat să construiesc jocul în jurul unei singure scene, ceea ce necesită un numar mai mare de controalere și scripturi. O consecință directă a folosirii unei singure scene este integrarea meniului principal în același mediu ca și jocul în sine. De aici derivă necesitatea de două scripturi ce controlează tranziția de la meniu la joc intitulate sugestiv *MenuController* și *GameController*.

1.4 Limitări ale modelului

Aplicația prezentă este împărțită pe diferite module ce controlează sunetul, generarea, *UI-ul* , instanțierea și multe altele. În **Unity** acest lucru se realizeză în mare parte prin scripturi atașate de obiectele din scena curentă.

Am optat să construiesc jocul în jurul unei singure scene, ceea ce necesită un numar mai mare de controalere și scripturi. O consecință directă a folosirii unei singure scene este integrarea meniului principal în același mediu ca și jocul în sine. De aici derivă necesitatea de două scripturi ce controlează tranziția de la meniu la joc intitulate sugestiv *MenuController* și *GameController*.

1.5 Algoritmi ce tratează problema estimării de parametri

Aplicația prezentă este împărțită pe diferite module ce controlează sunetul, generarea, *UI-ul* , instanțierea și multe altele. În **Unity** acest lucru se realizeză în mare parte prin scripturi atașate de obiectele din scena curentă.

Am optat să construiesc jocul în jurul unei singure scene, ceea ce necesită un numar mai mare de controalere și scripturi. O consecință directă a folosirii unei singure scene este integrarea meniului principal în același mediu ca și jocul în sine. De aici derivă necesitatea de două scripturi ce controlează tranziția de la meniu la joc intitulate sugestiv *MenuController* și *GameController*.

1.6 Complexitate și metode de optimizare

Aplicația prezentă este împărțită pe diferite module ce controlează sunetul, generarea, *UI-ul* , instanțierea și multe altele. În **Unity** acest lucru se realizeză în mare parte prin scripturi atașate de obiectele din scena curentă.

Am optat să construiesc jocul în jurul unei singure scene, ceea ce necesită un numar mai mare de controalere și scripturi. O consecință directă a folosirii unei singure scene este integrarea meniului principal în același mediu ca și jocul în sine. De aici derivă necesitatea de două scripturi ce controlează tranziția de la meniu la joc intitulate sugestiv *MenuController* și *GameController*.

Capitolul 2

Tehnologii

În cadrul acestui proiect s-au folosit următoarele tehnologii :

- **Unity 2019.1b** : motorul grafic folosit pentru construirea și randarea jocului.
- **Blender** : program *open source* folosit pentru modelarea 3D și contrucția obiectelor ce au fost ulterior importate în motorul grafic.
- **Math.NET** : librărie *open source* folosită pentru realizarea algoritmului de învățare automată deoarece acesta se bazează foarte mult pe lucru cu matrici.
- **C#** : limbajul nativ folosit de Unity pe lângă JavaScript , decizia fiind luată pur din motive de viteză.

Din multitudinea de motoare grafice disponibile am ales **Unity** doarece e o tehnologie cu care am mai lucrat, fiind conștient de capabilitățile acestuia. Multitudinea de unelte îl fac foarte ușor de recomandat și împreună cu magazinul de *assets* m-au ajutat foarte mult în procesul de construcție al acestui proiect. De asemenea **Unity** dispune de o documentație foarte bogată și bine pusă la punct , calități cheie pentru aprofundarea acestui motor grafic.

Câteva exemple de unelte *in-house* de care dispune motorul grafic ar fi , motorul de simularea a fizicii , motorul audio , sistemul de prefabricate , sistemul de iluminare și randare etc. Fără folosirea unei tehnologii ca **Unity** , acest proiect nu ar fi fost posibil într-un timp atăt de scurt , timpul de dezvoltare putând fi chiar triplat. Singurul dezavantaj imediat al acestui motor grafic este natura sa *closed source*.

Pentru modelele 3D folosite în cadrul jocului a fost necesară folosirea unei editor și anume **Blender**. Deși mare parte din obiectele importate în joc sunt preluate de pe

magazinul integrat în **Unity** , există anumite obiecte ce au fost modelate de către mine. Cu acest prilej am aprofundat anumite tehnici ce implică modelarea 3D , unul dintre cele mai importante find *UV unwrapping*.

Am ales **Blender** datorită politcii *open source* și a documentației vaste disponibile, fiind foarte ușor de lucrat în acesta, neputând aduce aplicația la un nivel dorit de finisaj fară acestă unealtă.

Din punct de vedere al librăriilor externe am folosit **Math.NET** , importată în **Unity** folosind **NuGet**. Necesitatea acestei librări este datorat algoritmului de invățare automată ce se bazează foarte mult pe lucrul și manipularea matricilor, de la adunarea cu un scalar până la înmulțirea/împărțirea element cu element. Detaliile legate de acest algoritm împreuna cu avantajele și dezavantajele acestuia vor fi discutate în secțiunea teoretică.

Datorită motorului grafic folosit și a limitărilor acestuia întregul proiect a fost construit folosind **C#**, alternativa sa fiind **JavaScript**. Din cauza experienței cu **C#** și a performanței ridicate am luat decizia de a folosi acest limbaj.

2.1 Structura proiectului

Aplicația creată este împărțită în doua etape , generare și instanțiere. Din cauza unor motive de performanță fiecare zonă, deșertică, rurală și urbană are propriul său generator. De asemenea fiecare tip de platforma folosește propriul său model Markov însumând astlfel un total de șase generatori, ce se ocupă doar cu producerea unei secvențe specifice de obiecte. Pentru partea de instanțiere am fost nevoit să construiesc un sistem ce facilitează instanțierea obiectelor generate, plasândule pe platformă în sensul acelor de ceasornic.

Aplicația fiind facută în **Unity** fișierele și directoarele au o structură arborescentă după cum urmează:

```
Assets
    Generators
    HMM
    Sound
    Prefabs
    Scripts
    Textures
    VolumetricLights
Packages
    PostProcessing
```

Directorul de **Assets** conține toate elementele necesare aplicației, de la librăria folosită pentru generarea obiectelor până la modulele de sunet și interfață iar directorul **Packages** conține cele mai importante uneltele folosite pentru realizarea și finisajul jocului importate direct de către **Unity**.

Generators aici se află cei șase generatori folosiți pentru construirea mediului.

HMM este directorul ce conține scripturile necesare pentru construirea modelului Markov și a antrenării lui.

Sound conține toate efectele de sunet și muzica din joc.

Prefabs este directorul cu toate prefabricatele din joc, incluzând modelele pentru platforme, obiecte, jucător cât și inamici.

Scripts aici sunt stocate toate fișierele sursă ce modelează comportamentul jucatorului, al meniurilor și a sistemelui de generare/instanțiere, sumându-se în total la optsprezece fișiere.

Textures directorul ce conține toate fișierele de tip *Albedomap* , *Heightmap* și *Occlusionmap* folosite în cadrul materialelor ce au fost ulterior importate pe obiecte.

VolumetricLights directorul ce conține fișierele preluate de pe *Github* pentru lumină volumetrică.

PostProcessing modulul folosit de *Unity* pentru a putea integra efecte precum corectare de culoare, *bloom*, *motion blur* și multe altele.

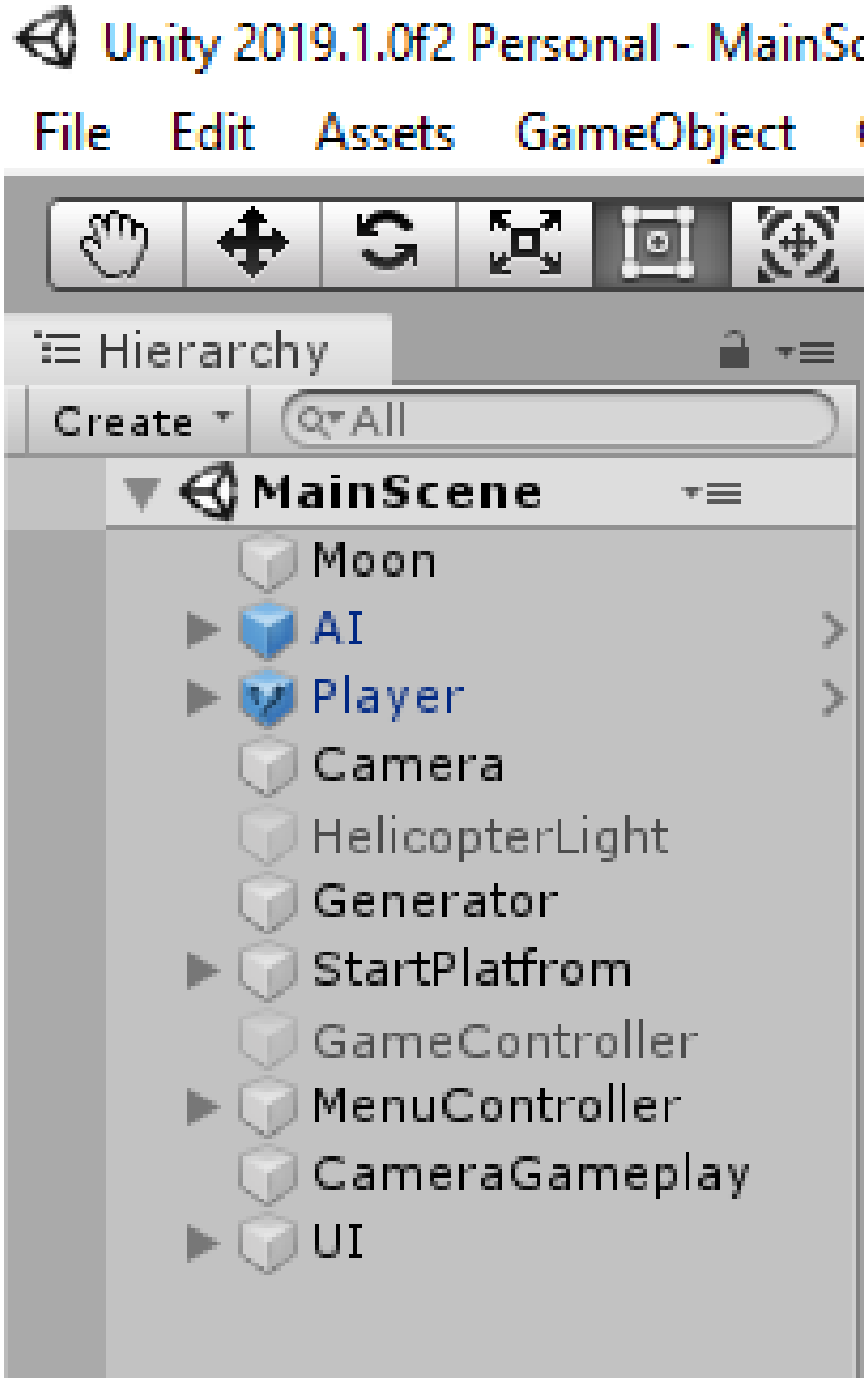
2.2 Componente

Aplicația prezentă este împărțită pe diferite module ce controlează sunetul, generarea, *UI-ul* , instanțierea și multe altele. În **Unity** acest lucru se realizează în mare parte prin scripturi atașate de obiectele din scena curentă.

Am optat să construiesc jocul în jurul unei singure scene, ceea ce necesită un numar mai mare de controalere și scripturi. O consecință directă a folosirii unei singure scene este integrarea meniului principal în același mediu ca și jocul în sine. De aici derivă necesitatea de două scripturi ce controlează tranziția de la meniu la joc intitulate sugestiv *MenuController* și *GameController*.

De asemenea există un obiect *Generator* ce conține toți generatori necesari pentru construirea scenei. Unul pentru fiecare tip de zonă de interes și tip de platformă. Legătura între cei șase generatori este facută de cele două scripturi atașate de obiectul *Camera* denumite *PlatformBuilder* și *PlatformController*. Primul se ocupă cu instanțiera

și plasarea obiectelor pe platformă în funcție de parametri de instanțiere setați de un script atașat de un *Spawn Point*. Cel de-al doilea se ocupă cu randomizarea și controlarea numărului de platforme pentru fiecare zonă rurală , urbană și deșertică.



2.2.1 MenuController

Acest obiect se ocupă de toate aspectele ce țin de meniul principal. Deoarece există o singură scenă scriptul este destul de complex , rolul său fiind activarea tuturor componentelor ce țin de jocul principal, sunet, inamici, jucător, cameră, efecte și multe altele.

În continuare se vor prezenta cele mai importante secvențe de cod ce se găsesc în fișierul **MenuController**.

Exemplu 2.1: Funcțiile din MenuController

```
1  public void OnGameExit()
2  {
3          Application.Quit();
4  }
5
6  public static void OnGameReset()
7  {
8          retry = true;
9  }
10
11 public void OnCredits()
12 {
13         this.creditsStart = true;
14         this.credits.SetActive(true);
15 }
16
17 public void OnGameStart()
18 {
19         gameStart = true;
20         gameController.SetActive(true);
21         player.constraints = RigidbodyConstraints.None;
22         AI.constraints = RigidbodyConstraints.None;
23         player.velocity = Vector3.forward * 3;
24         AI.velocity = Vector3.forward;
25         bars.SetActive(true);
26 }
```

Cele patru funcții controlează tranzițiile aplicației în funcție de interacțiunea jucatorului cu meniul principal. De remarcat în funcția *OnGameStart* linia ce activează *GameController* odată ce se pornește jocul și anume *gameController.SetActive(true)*.

2.2.2 GameController

Obiectul ce se ocupă de funcționalitatea principală din joc. În mare parte meniul de pauză și meniul de final al jocului sunt controlate de acest script.

Legat de parte de cod , fișierul sursă atașat de obiect arată astfel:

Exemplu 2.2: Funcțiile din GameController

```
1  public void SetPausedState()
2  {
3          gamePaused = gamePaused == true ? false : true;
4  }
5
6  public void SetGameOverState()
7  {
8          player.GetComponent<CarController>().enabled = false;
9          cameraScript.enabled = false;
10         enemyScript.enabled = false;
11         gameOver = true;
12         gameOverScreen.SetActive(true);
13 }
14
15 public void TogglePause()
16 {
17         ToggleSound();
18         ToggleRender();
19         TogglePauseMenu();
20         SetPausedState();
21 }
```

Meniul din joc odată activat oprește randarea jocului, lucru ce complică apelu de funcții dependente de timp.

2.2.3 PlatformBuilder

Scriptul ce realizează construirea în timp real a platformelor și le instanțiază. Datorită volumului mare de cod mai jos este prezentată doar funcția ce adaugă obiectele pe platformă. Înainte de instanțiere funcția are grija să preia datele de *spawn* , conținute de scriptul atașat *SpawnSettings* pentru fiecare loc valid de instanțiere.

Exemplu 2.3: Funcțiile din PlatformBuilder

```
1  public GameObject BuildPlatform(GameObject state)
2  {
3          GameObject nextPlatform = Instantiate(state, new Vector3(currentPlatform.transform.
               position.x + xOffset, −15, currentPlatform.transform.position.z + zOffset),
               Quaternion.Euler(0, 0, 0));
4          HMM propGenerator = ChoosePropGenerator(nextPlatform.name);
5          Transform spawnPoints = nextPlatform.transform.Find("SpawnPoints");
6          foreach (Transform spawn in spawnPoints)
7          {
8              SpawnSettings spawnSettings = spawn.GetComponent<SpawnSettings>();
9              if (spawnSettings.isStackable == false && spawnSettings.maxObjectStack > 1)
10                 throw new System.Exception("Spawn is not stackable!");
11             for (int i = 0; i < spawnSettings.maxObjectStack; ++i)
12             {
13                 InstantiateProp(propGenerator, spawn, spawnSettings, i);
14             }
15         }
16         nextPlatform.transform.rotation = Quaternion.Euler(0, rotation, 0);
17         return nextPlatform;
18 }
```

Scriptul *SpawnSettings* conține doi parametri *isStackable* și *maxObjectStack* folosite pentru a specifica dacă locul de *spawn* poate să conțină mai multe obiecte și în ce cantitate.

2.2.4 PlatformController

Din cauza necesității de a controla durata unei anumite zone evitând astfel inconsistența, *PlatformController* alege un număr de platforme pentru fiecare secțiune tinând cont de

o limită superioara și inferioară.

Exemplu 2.4: Funcțiile din PlatformController

```
public GameObject GetNextPlatform()
{
        GameObject output;
        currentEmissionCount++;
        if (currentEmissionCount > maxEmissionCount)
        {
            output = transitionPlatforms[currentGeneratorIndex];
            currentEmissionCount = 0;
            currentGeneratorIndex = (currentGeneratorIndex + 1) % generators.Length;
            maxEmissionCount = Random.Range(lowerBound, upperBound);
        }
        else
        {
            output = generators[currentGeneratorIndex].NextEmission();
        }
        return output;
}
```

Capitolul 3

Arhitectura aplicației

Amet venenatis urna cursus eget. Quam vulputate dignissim suspendisse in est ante. Proin nibh nisl condimentum id. Egestas maecenas pharetra convallis posuere morbi. Risus viverra adipiscing at in. Vulputate eu scelerisque felis imperdiet. Cras adipiscing enim eu turpis egestas pretium aenean pharetra. In aliquam sem fringilla ut morbi tincidunt augue. Montes nascetur ridiculus mus mauris. Viverra accumsan in nisl nisi scelerisque eu ultrices vitae. In nibh mauris cursus mattis molestie a iaculis. Interdum consectetur libero id faucibus nisl tincidunt eget. Gravida in fermentum et sollicitudin ac orci. Suscipit adipiscing bibendum est ultricies. Etiam non quam lacus suspendisse. Leo urna molestie at elementum eu facilisis sed odio morbi. Egestas congue quisque egestas diam in arcu cursus. Amet consectetur adipiscing elit ut aliquam purus.

Concluzii

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Nunc mattis enim ut tellus elementum sagittis vitae et. Placerat in egestas erat imperdiet sed euismod. Urna id volutpat lacus laoreet non curabitur gravida. Blandit turpis cursus in hac habitasse platea. Eget nunc lobortis mattis aliquam faucibus. Est pellentesque elit ullamcorper dignissim cras tincidunt lobortis feugiat. Viverra maecenas accumsan lacus vel facilisis volutpat est. Non odio euismod lacinia at quis risus sed vulputate odio. Consequat ac felis donec et odio pellentesque diam volutpat commodo. Etiam sit amet nisl purus in. Tortor condimentum lacinia quis vel eros donec. Phasellus egestas tellus rutrum tellus pellentesque eu tincidunt. Aliquam id diam maecenas ultricies mi eget mauris pharetra. Enim eu turpis egestas pretium.

Bibliografie

- Author1, *Book1*, 2018
- Author2, *Boook2*, 2017