C:\Users\user\Downloads\unnamed.png

PROIECT PENTRU OBTINEREA ATESTATULUI DE COMPETENTE PROFESIONALE

ELEV: PROFESOR:

Munteanu Adrian-Constantin Gavril Petru-Florin

SESIUNEA MAI 2020

TEMĂ PROIECT:

PROGRAMARE C#/Unity

SAH 3D

ARGUMENT

De mic copil eram fascinate de logica matematica a jocului de șah. După ce am început să studiez informatica, am înțeles aplicabilitatea jocului de șah în modelul tablourilor bidimensionale. Lejeritatea adaptării acestui joc în model informatic este motivul pentru care am hotărât să construiesc un joc de șah tridimensional.

# CUPRINS

1. [UNITY](#_UNITY)…………………………………………………………………………………………………………………………………………..4
2. [SCENE UTILIZATE](#_CAPITOLUL_II_–) ………………………………………………………………………………………………………………………..10

1. [ELEMENTE 2D](#_CAPITOLUL_III_–)……………………………………………………………………………………………………………………………..13
2. [MODELE 3D](#_CAPITOLUL_IV_–)………………………………………………………………………………………………………………………………..16
3. [GENERAREA PIESELOR SI TABLOURI BIDIMENSIONALE](#_CAPITOLUL_V_-)……………………………………………………………….18

1. [MISCAREA PIESELOR](#_CAPITOLUL_VI_–)…………………………………………………………………………………………………………………..21

1. [CONDITII SPECIALE](#_CAPITOLUL_VII_–)……………………………………………………………………………………………………………………..31
2. [ALGORITMUL DE ȘAH](#_CAPITOLUL_VIII_–)………………………………………………………………………………………………………………….34

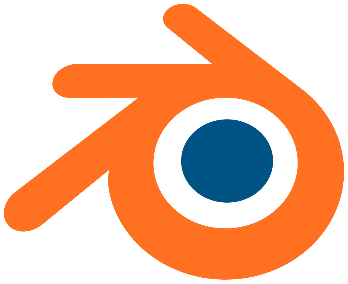
9. [BIBLIOGRAFIE](#_BIBLIOGRAFIE)……………………………………………………………………………………………………………………………..40

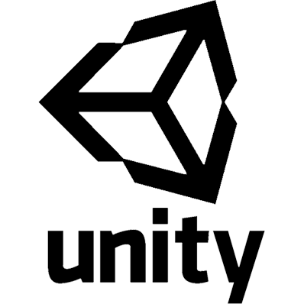
# CAPITOLUL I - UNITY

Pentru a construi jocul de șah am fost nevoit, sa folosesc un motor de joc(game engine). Un motor de joc este un program realizat special de o echipa, înainte de realizarea unui joc, pentru a putea utiliza și simula diferite acțiuni, mai ușor decât programându-le direct. Motorul de joc în sine, conține elemente preconcepute, tridimensionale sau bidimensionale, care pot fi modelate, sisteme de interacțiune, modele și sisteme fizice, efecte și particule, și componente ce acționează direct asupra celor menționate anterior cu ajutorul unui/unor limbaje de programare.

Cele mai cunoscute motoare de joc, folosite in momentul de fata sunt:

* Unreal Engine
* folosit pentru a simula interacțiuni fizice cu detalii specifice;
* folosit la jocuri precum Grand Theft Auto, Assassins’ Creed, Smite;
* limbaj de programare C++;
* Source Engine
* folosit de jocurile de tip shooter;
* dezvoltat de Valve;
* folosit la jocuri precum Counter Strike , Team Fortress 2;
* limbaj de programare C++;
* Blender
* folosit la realizarea animațiilor tridimensionale;
* folosit la realizarea filmelor animate cu mișcare fizică;
* folosit la construirea modelelor pentru diferite jocuri;
* limbaj de programare C++;
* Unity
* folosit la realizarea jocurilor cu dinamica rapida și concepute cu construcție simpla;
* folosit la jocuri precum Steep, Subnautica, Celeste;
* limbaj de programare C# și JavaScript;



Motorul de joc pe care am hotărât să-l folosesc este Unity.

Acest motor de joc a fost dezvoltat de Unity Technologies și a fost anunțat și lansat în iunie 2005, la Conferința Dezvoltatorilor Globali de la Apple. Acesta se baza in principal pe dezvoltarea jocurilor de PC, iOS și browser, însă cu începutul anului 2018, acesta a fost programat sa suporte 25 de platforme diferite. Motorul de joc a suferit de-a lungul timpului mai multe schimbări majore, ultima versiune stabila fiind Unity2019.3.12, lansata în Aprilie 2020.

Motorul de joc, Unity, poate fi folosit pentru crearea de jocuri tridimensionale, bidimensionale, realităților virtuale, precum si construcția de simulatoare virtuale. Acest motor de joc a fost adaptat de dezvoltatori pentru a putea fi folosit și în alte domenii, precum în industria cinematografica, industria de automobile, industria arhitecturii și construcțiilor, si în industria ingineriei și chimiei .

In primi zece ani platforma Unity a vândut peste 100000 de licențe. Însă din 2016 aceștia au dezvoltat un program care permite folosirea în mod gratuit a platformei de către elevi și studenți, precum și de diverși dezvoltatori de jocuri indie (jocuri create de un grup mic de persoane), care au realizat un profit uriaș. Printre aceste jocuri se numără: SuperHot, Undertale și Unturned.

Unity deține și un magazin virtual în care pot fi cumpărate piese de cod, obiecte 3d și 2d, sunete și alte asseturi care pot fi folosite de dezvoltatorii de jocuri. Acestea pot fi cumpărate prin free-lancing și prețul variază intre 5$ si 500$.

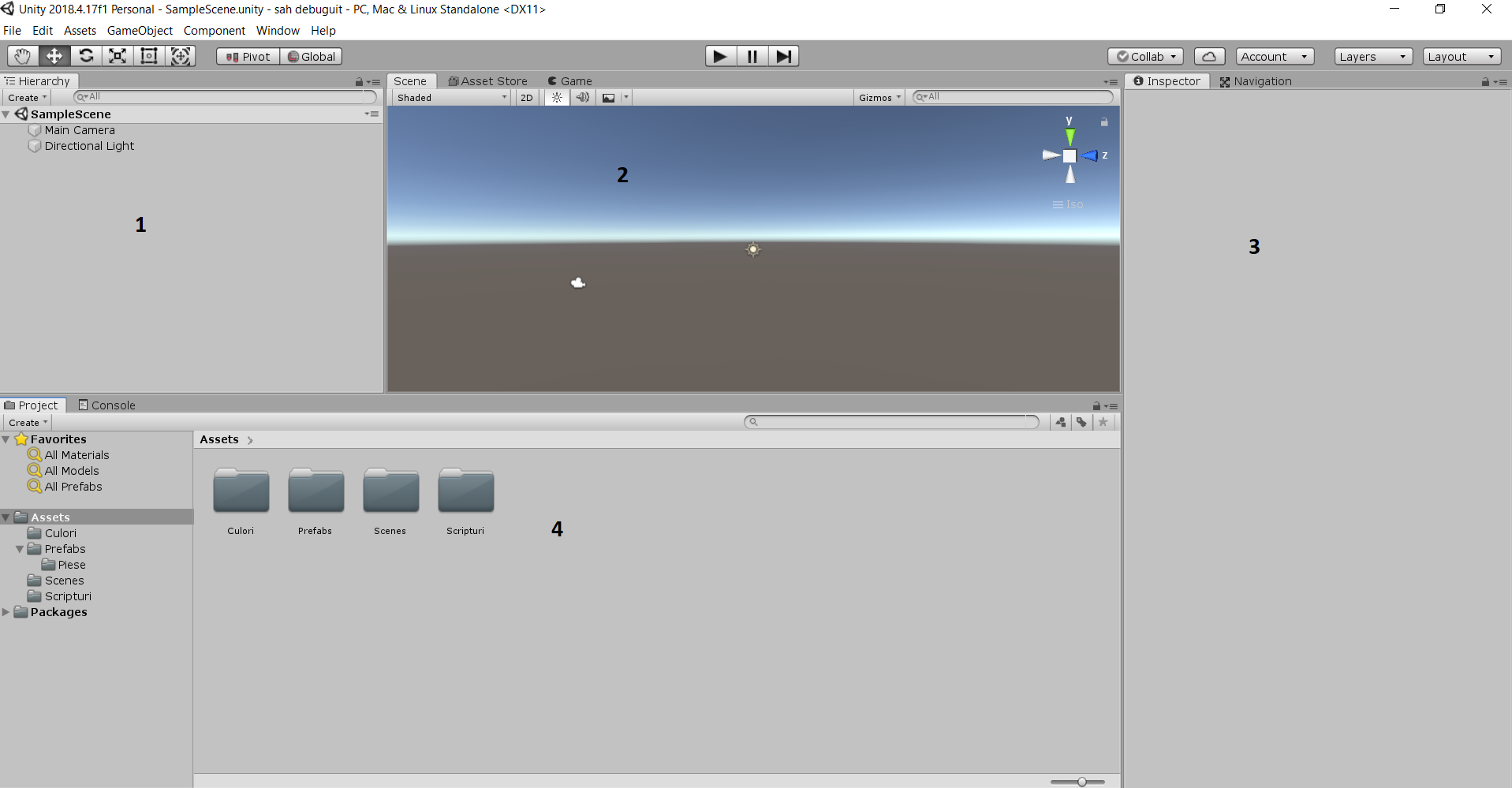
Interfața programului este simplista, fiind alcătuit din:

1 – Lista obiectelor utilizate

2 – Fereastra scenei si fereastra jocului

3 – Inspectorul de obiect

4 – Lista fișierelor utilizate și Consola



1. Lista de obiecte utilizate conține toate obiectele utilizate într-o scena. Aceste obiecte pot fi de mai mult tipuri: - obiecte 3D

- obiecte 2D

- efecte

- lumini

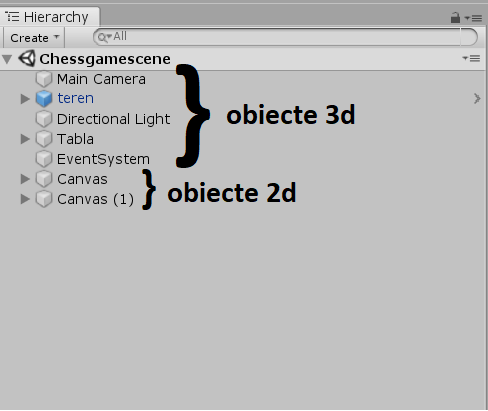
- audio

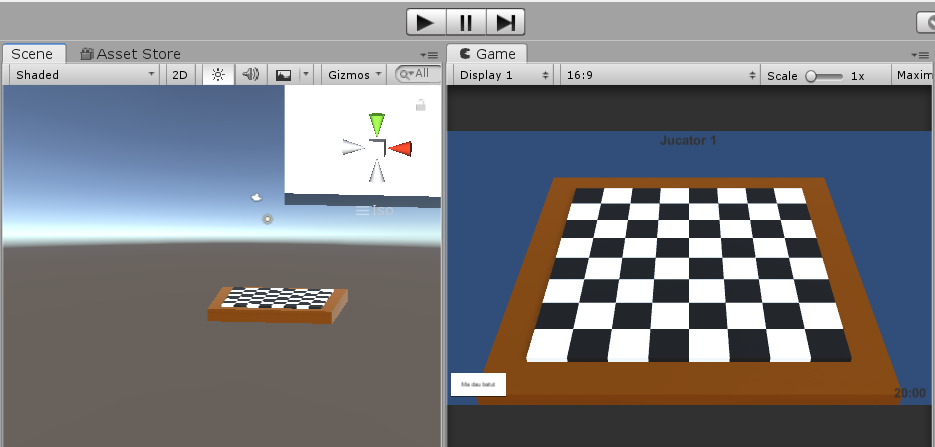
- video

- UI (unitate de interfață) – ex: pânză(canvas), buton, text, etc.

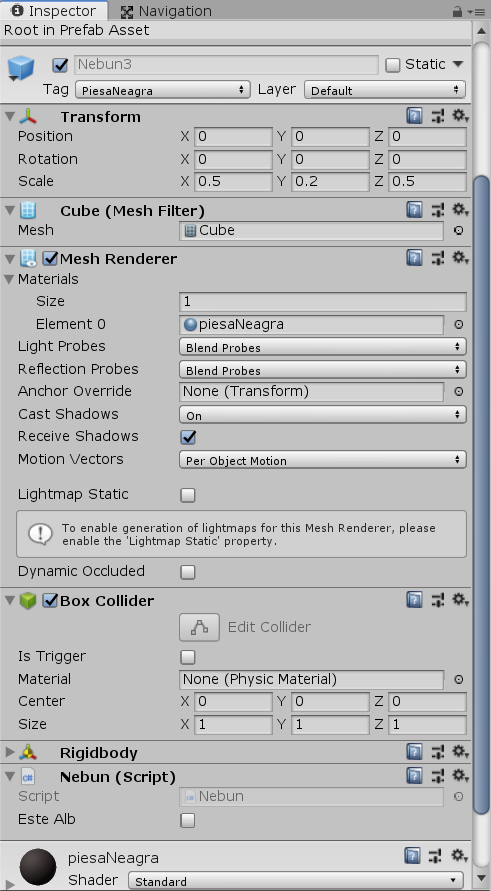
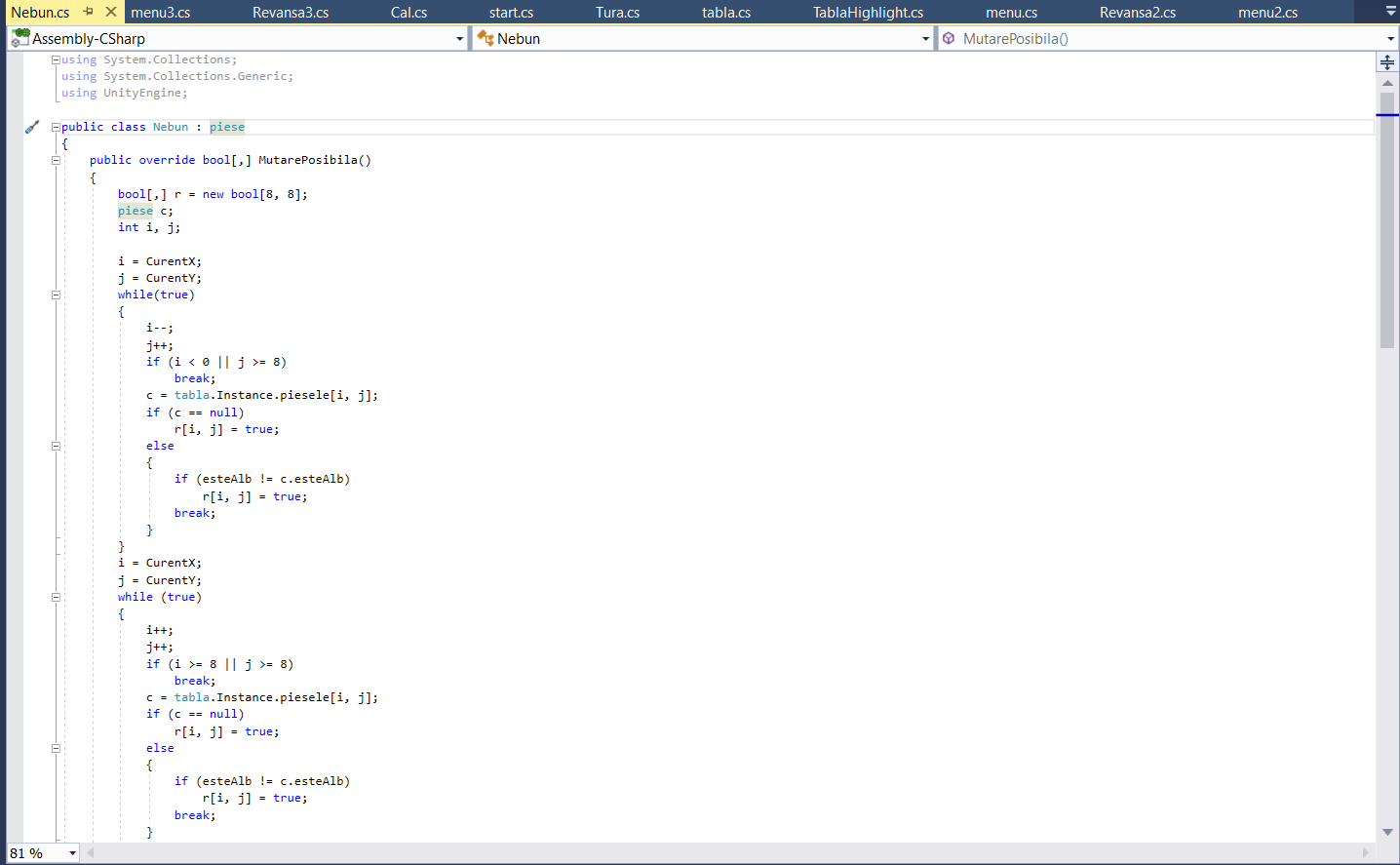
- cameră

Primele obiecte create implicit, când realizam un nou proiect sunt o lumină direcțională cu raze normale și o camera care este considerata camera principală.

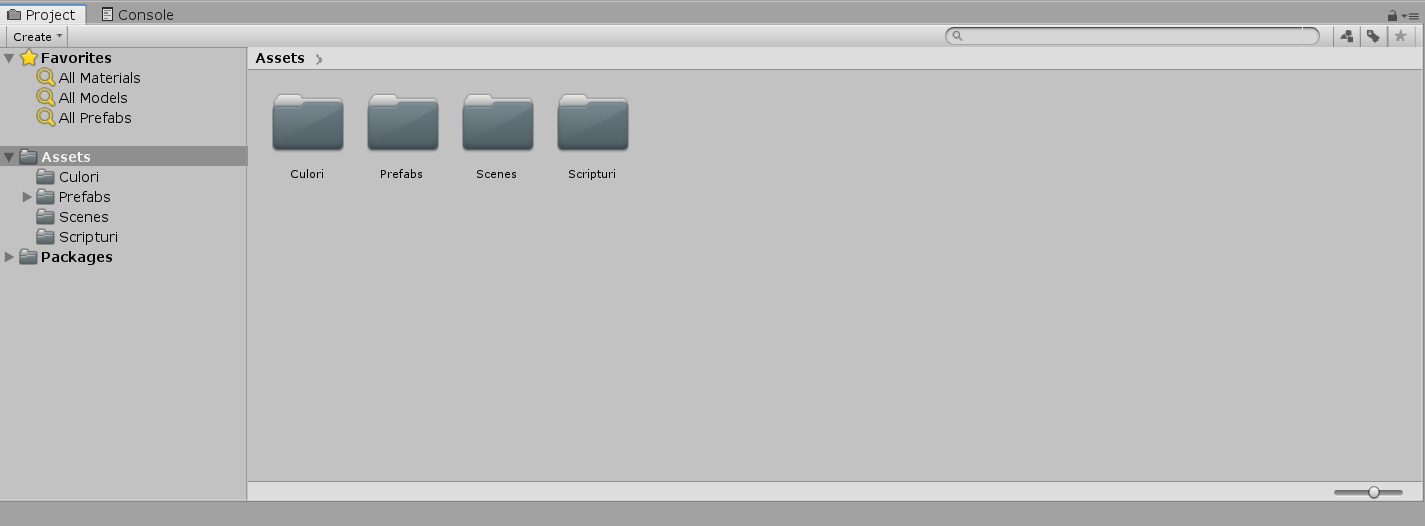


2. Fereastra scenei și fereastra jocului se afla în centrul programului și cu ajutorul acestora putem vizualiza obiectele pe care le utilizăm, orientarea lor și esteticile lor. Toate obiectele din lista obiectelor vor fi afișate în fereastra scenei cât și în fereastra jocului. Singurele obiecte care nu pot fi vizualizate sunt cele care sunt dezactivate (au funcția Mesh Render dezactivată). Diferența dintre cele 2 ferestre este aceea că fereastra scenei este mereu activă și putem vizualiza proprietățile obiectelor in timp real și fereastra jocului afișează ce se vede în timpul jocului și este activată când se apasă butonul play. Fereastra jocului afișează practic jocul, obiectele din scena respective și acțiunile pe care le poate întreprinde jucătorul. Fereastra jocului este strâns legata de poziția camerei principale, deoarece jucătorul vede prin perspectiva acesteia și toate proprietățile modificate în timpul jocului nu vor fi salvate.

3. Inspectorul de obiect este utilizat pentru a modifica proprietățile obiectului in timp real. Aceste proprietăți sunt salvate în elemente numite **componente**. Acestea pot fi implicite și/sau realizate de dezvoltator în funcție de necesitatea acestora. Unui obiect i se pot atribui culori, i se pot adăuga forțe, masă, dimensiune și așa mai departe. Componentele realizate de dezvoltator se numesc **scripturi** și sunt realizate în programul Microsoft Visual Studios, folosind limbajul de programare C#. Scripturile pot folosi funcții și sisteme din bibliotecile implicite Unity.



4. Lista fișierelor utilizate și consola, sunt situate sub lista de obiecte și sub fereastra scenei. Lista fișierelor conține toate elementele jocului, care pot fi deschise, separate și organizate in mai multe dosare. Obiectele 3d și 2d, puse în lista de obiecte, se numesc prefaburi și păstrează toate componentele care le-au fost atribuite în momentul în care au fost adăugate în listă. În lista fișierelor mai pot fi adăugate culori, texturi și scripturi create de dezvoltator, precum și scene care sunt salvate automat in listă, odată ce sunt realizate. Consola este un element opțional al programului, cu ajutorul căruia putem testa scripturile jocului – ex: afișează un mesaj când detectează o coliziune.



# CAPITOLUL II – SCENE UTILIZATE

Pentru crearea unui joc se realizează prima data interfața și scenele folosite. Designul și utilitatea acestora sunt stabilite înainte de programare. Practic se realizează un plan de creație a scenelor. Aceste scene sunt apoi utilizate și legate între ele prin butoane și condiții. Pentru realizarea acestui joc am considerat necesare 4 scene:

1. Scena Meniului;

2. Scena Jocului;

3. Scena „Primul jucător a câștigat”;

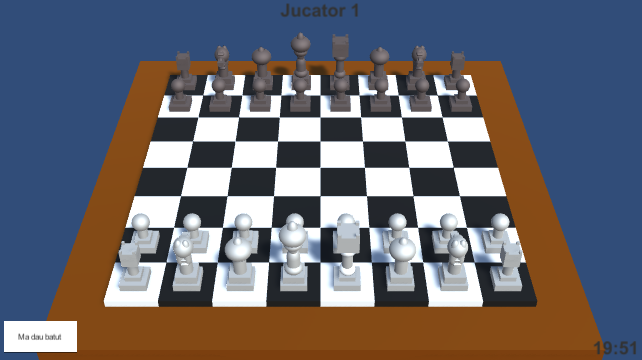
4. Scena „Al doilea jucător a câștigat”;

Pentru testul algoritmilor am folosit și o scena de test: 5. Sample Scene (auxiliara).

1. Prima scena, cea a meniului, apare în momentul în care este deschisa aplicația. Aceasta conține titlul aplicației („Sah3D”), 2 butoane („Jucător vs Jucător” si „Ieșire”), în stânga jos numele creatorului și în dreapta jos versiunea jocului. În aceasta scena au fost utilizate doua scripturi, legate de cele doua butoane, care au împreună 24 de linii de cod.



2. A doua scena, cea a jocului, presupune jocul în sine și la ea se poate ajunge din toate celelalte scene. Aceasta conține tabla de șah și piesele in centru, în vârful scenei este scris numele jucătorului care trebuie sa mute, în stânga jos se afla un buton („Mă dau bătut”), iar în dreapta jos se afla un temporizator, de 20 de minute, care reprezintă timpul standard pe care îl are fiecare jucător in timpul meciului. temporizatorul este individual pentru fiecare jucător și când ajunge la zero jocul se termina. In aceasta scena am folosit 13 componente, dintre care 6 generează mișcarea pieselor, 1 care leagă toate scripturile de mișcare a pieselor impreuna,1 care generează pozițiile de mutare pe tablă, 2 care sunt folosite pentru butonul „Mă dau bătut”, 2 care folosite pentru timer și 1 care le unește pe toate cele menționate mai sus, prin relații parent-child (relații în care anumite scripturi nu pot fi folosite decât de scriptul părinte). Toate scripturile de mai sus conțin in total 1864 de linii de cod.



3 si 4. Scenele 3 si 4 sunt similare, având diferit doar titlul din centrul fiecărei scene. Scena 3 are titlul „Jucătorul 1 a câștigat” iar scena 4 are titlul „ Jucătorul 2 a câștigat”. Cele doua scene prezinta 2 butoane („Revanșa” și „Menu”) care pot reîncepe jocul sau te duc la prima scena a aplicației și în stânga jos este din nou scris numele creatorului. Pentru aceste doua scene am folosit 4 scripturi care sunt doar legate de butoane de pe fiecare scena si care au împreuna 42 de linii de cod.



Toate cele 4 scene sunt legate intre folosind funcții din biblioteca implicita Unity.sceneManagement în fiecare dintre scripturile butoanelor, precum si in scriptul temporizatorului.

Legăturile dintre scene se reprezintă grafic astfel:

3

Scena Jucătorului 1

Meniu SahMat

1 2 Temporizator

Scena Jocului

Scena Meniului

JvJ Revanșă

Revanșă 4

Scena Jucătorului 2

Temporizator

Meniu SahMat

JvJ = Jucător vs Jucător

# CAPITOLUL III – ELEMENTE 2D

În Unity elementele 2d sunt utilizate in principal pentru a apărea în scene și se poate interacționa cu ele folosind un obiect de joc numit EventSystem și care este generat odată cu folosirea unui element **UI** (**unitate interactiva**). Unitățile interactive sunt separate de elementele 2d, acestea putând fi apăsate sau putând fi transformate.

În jocul de șah, elementele 2d folosit sunt doar **unități interactive**. Aceste unități interactive sunt de tipul:

1 – Canvas (pânză) - cel mai des folosită;

- acoperă tot ecranul;

- va conține toate elementele 2d folosite;

2 – Text - bara de text este în principal folosită pentru afișarea unui text 2d;

- bara de text poate fi inclusa sub multe alte elemente atât 2d cât și 3d;

3 – Butonul - este interacțiune de tip point-and-click;

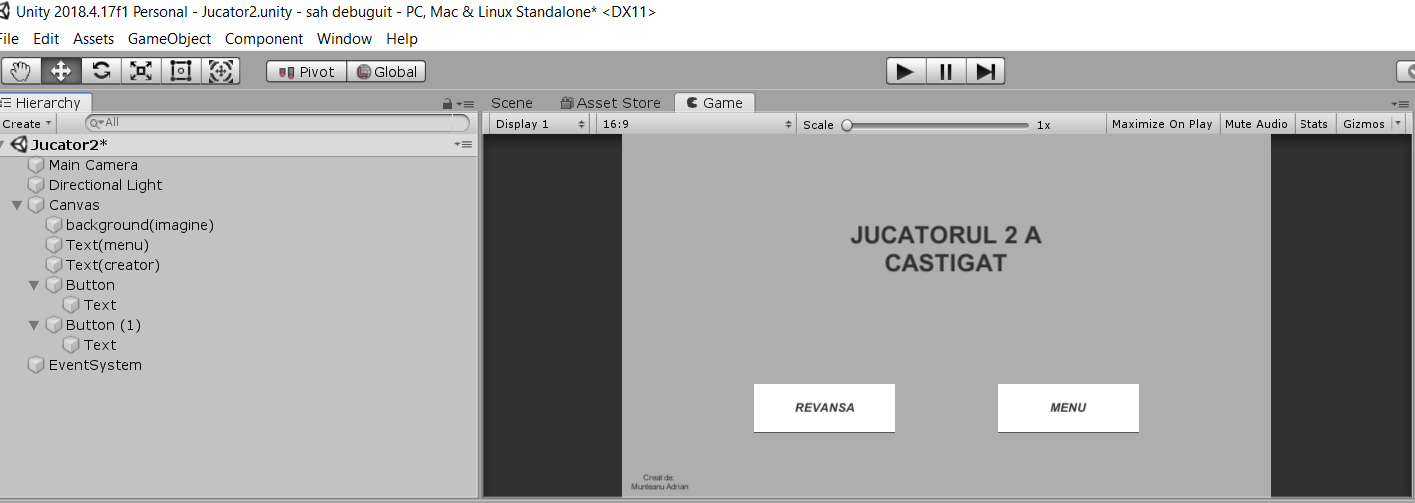
- poate conține bara de text, imagine, etc.;

- forma poate fi modelata ușor folosind unityEDIT;

4 – Imaginea - aceasta este de natura 2d și poate fi inclusă sub elemente 2d cât și 3d, la fel ca elementele text;

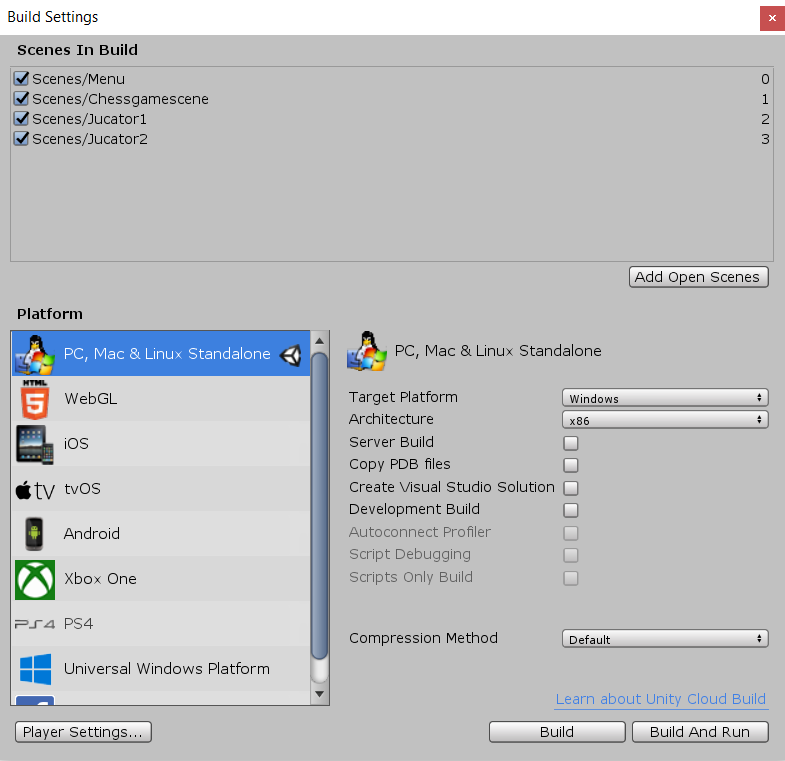
- aceasta pot varia de la poze complexe la culori și așa mai departe;

- de menționat este faptul ca imaginile sunt foarte ușor de transformat in Unity, cu ajutorul funcțiilor de modificare a textului, însă în acest proiect nu a fost necesar modificarea lor;



Scripturile folosite pe elementele 2d au constatat în cele care fac ca butoanele să poată face trecerea de la o scena la alta, cât și temporizatorul din colțul dreapta jos a scenei jocului de șah. În total au fost utilizate 10 scripturi plasate sub elementele 2d.

1. Algoritmul de apăsare a butoanelor preia numărul de indice al scenei active (care este prezentata pe ecran) și o creste/o scade schimbând scena în funcție de poziția lor in stiva scenelor din meniul de build.



Meniul Build

2. Algoritmul temporizatorului modifica practic textul acestuia în funcție de trecerea timpului. Temporizatorul primește o valoare inițială care scade. Pentru a sintetiza timpul și a utiliza temporizatorul neținând cont de specificațiile și performanța calculatorului vom folosi funcția Time.deltaTime, care asigura trecerea in mod obișnuit a secundelor. Cu fiecare secunda timpul inițial va scădea cu -1\*Time.deltaTime. Apoi folosind numărul de secunde se va calcula numărul de minute si toate valorile vor fi convertite in text cu ajutorul funcției tostring() sub forma “min:sec”.

# CAPITOLUL IV – MODELE 3D

Modelele 3d oferite de Unity presupun un set de elemente deformabile, care unite pot crea obiecte și structuri complexe. Acestor structuri li se pot atribui imagini, culori, materiale, scripturi și așa mai departe. Obiectele 3d oferite implicit de Unity pot fi: - cub

- sfera

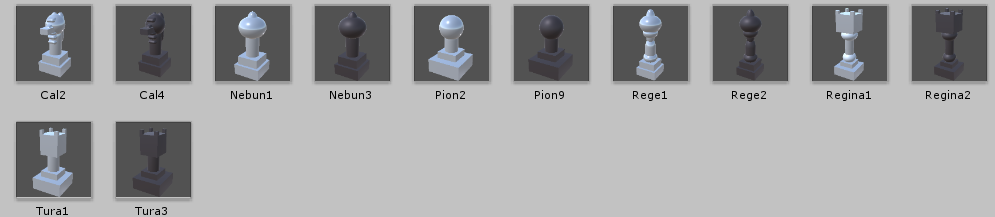
- cilindru

- plan (3d)

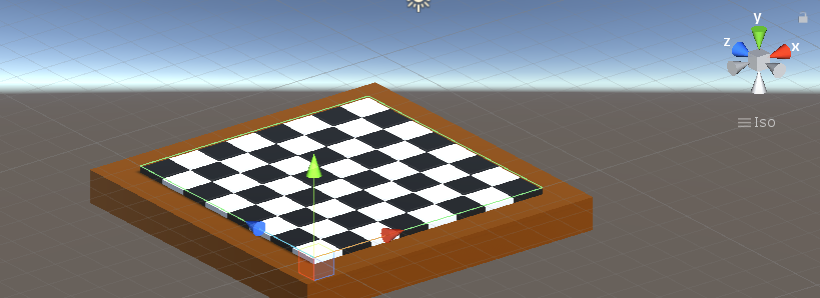
- teren

- etc.

Astfel utilizând obiectele de mai sus am construit piesele modificând component transform și combinând piesele sub o singura structura și anume baza piesei. Piesele au fost apoi salvate ca prefaburi (obiecte 3d salvate) în joc și generate succesiv, atunci când este nevoie.



După realizarea pieselor, am construit o tabla folosind mai multe cuburi pe care le-am unit sub un singur obiect mai mare (un alt cub cu dimensiunea x=9,y=1,z=9) numit tabla.



Tuturor obiectelor 3d le-am oferit culori cu ajutorul structuri de tip material care au fost salvate in dosarul culori.

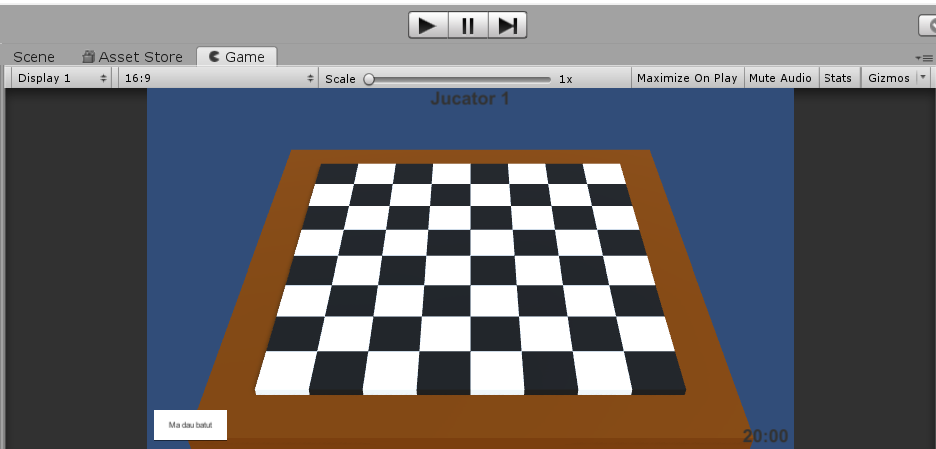
Pentru afișarea mișcărilor posibile am folosit un cub cu înălțimea de 0,01 , căruia i-am atribuit un material de culoare albastru deschis.

Toate modelele 3d sunt obiecte ce vor fi folosite doar in a doua scena a aplicației si anume scena jocului.

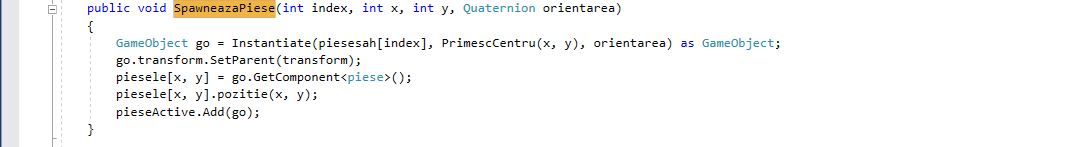
Scripturile folosite pe obiectele 3d sunt cele care generează mișcarea pieselor, folosind o funcție de override, care transforma și generează un tablou bidimensional r cu posibilitatea de mutare a pieselor pe anumite poziții, scriptul care verifica posibilitatea de rocada, schimbare pion-regina și condiția de șah folosindu-se de poziția pieselor, scriptul care afișează posibilitatea mișcării pieselor. De asemenea fiecare piesa are o proprietate numita esteAlb, care oferă informații despre echipa in care se afla piesa respectiva.

# CAPITOLUL V - GENERAREA PIESELOR SI TABLOURI BIDIMENSIONALE

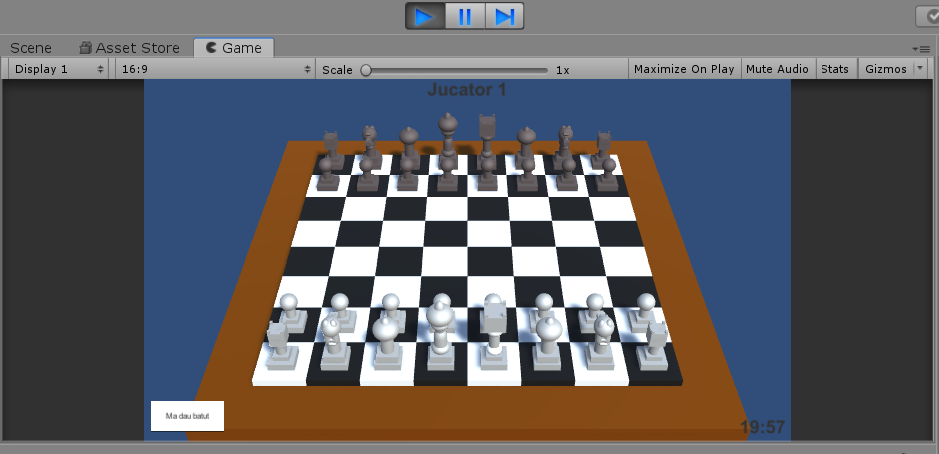
Jocul de șah consta în principiu în asocierea pieselor cu tabla și legătura intre diferite poziții și piesa respective. Astfel cu ajutorul jocului de șah se pot stabili ușor și se pot experimenta relații între obiecte și tablourile bidimensionale. Întregul joc consta în realizarea de scenarii și mutări într-un tablou bidimensional de 8 pe 8.Cu toate acestea la început scena a doua este goala.



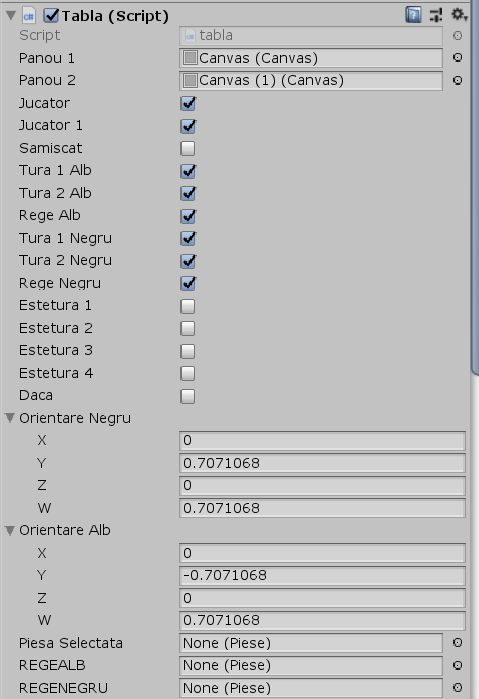
Piesele se generează la începutul intrării în scena. În plus fiecărei piesa i se atribuie un tip, aceasta putând fi: pion, cal, nebun, tura, rege, regina. Piesele prezinta poziții si rotații diferite care sunt salvate la început salvate in variabile de tip Vector3 (retine 3 variabile de tip float: x, y, z) și Quaternion (retine valori de rotație a obiectelor in 4 planuri: x ,y ,z ,w). Piesele sunt generate cu ajutorul funcției SpawneazaPiese.



În plus fiecare rege si fiecare tura sunt salvate in variabile de tip GameObject și de asemenea se salvează un bool care verifica daca turele sau regii au fost mutați precum și o variabila jucător, de tip bool, care verifica daca cel care muta este jucătorul 1 sau 2.



După ce fiecare piesa a fost generata, acestea sunt salvate într-un tablou bidimensional de tipul GameObject numit piesele si apoi se generează alte 7 tablouri bidimensionale de tip bool care verifica: dacă piesa poate fi mutată, mutările pe care le poate face piesa, pozițiile de șah ale echipei adverse, pozițiile pat in jurul regelui, pozițiile pe care le poate face piesa dacă regele este în șah (se salvează și piesa care ataca regele în variabila GameObject P), piesele active și piesele capturate. Aceste 7 tablouri bidimensionale se generează după fiecare mutare făcută de jucători.



De asemenea am folosit funcția DrawLine() pentru a genera planul tablei pe care vor fi afișate piesele.

Toate cele de mai sus sunt salvate în scriptul tabla, care este scriptul principal și fișier părinte pentru scriptul de generare a pozițiilor și algoritmul de verificare a șahului.

Scriptul generare poziții

„Piesa”

Script afișare poziții

„TableHighlights”

Script principal

„tabla”

Scriptul verificare șah

„Șahmat”

# CAPITOLUL VI – MISCAREA PIESELOR

Pentru a mișca o piesa, aceasta trebuie selectata. Am creat astfel, în scriptul principal, am creat o funcție numită SelecteazaPiesa, care folosind sistemul de raycast din Unity, va trimite o raza de pe direcția cursorului și va primi pozițiile în planul generat cu ajutorul funcției DrawLine() în variabilele CurrentX și CurrentY. Va verifica dacă există o piesă și dacă este în echipa jucătorului apoi o va trece în variabila GameObject PiesaSelectata.

După ce piesa a fost selectata se cere să se afișeze pozițiile pe care aceasta poate fi mutata. Pentru a se afișa pozițiile pieselor care sunt selectate, se va utiliza o funcție numita MutaPiesa, care va face trimitere la scriptul „TableHighlights”, fiu al scriptului „tabla”. Acesta va încerca să genereze un tablou bidimensional numit „MiscarPosibile” făcând trimitere la scriptul „Piese”, fiu al său. Cel din urma va verifica tipul piesei selectate și va căuta in scripturile fiu ale sale cel care se potrivește tipului piesei selectate. Toate scripturile sunt dependente însă de tabloul bidimensional piesele, care are salvate obiectele jocului.

Scriptul generare mișcări

„Pion”

Scriptul generare mișcări

„Cal”

Scriptul generare mișcări

„Nebun”

Script afișare poziții

„TableHighlights”

Scriptul generare poziții

„Piesa”

Scriptul generare mișcări

„Rege”

„”

Scriptul generare mișcări

„Regina”

Scriptul generare mișcări

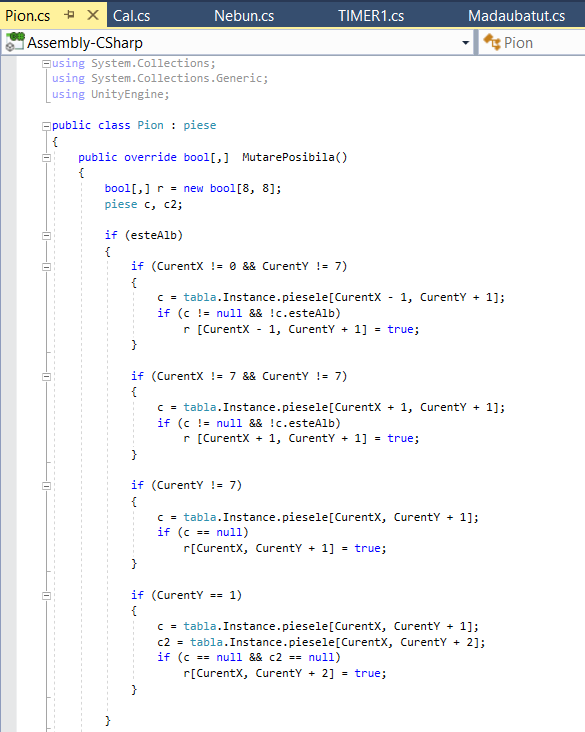
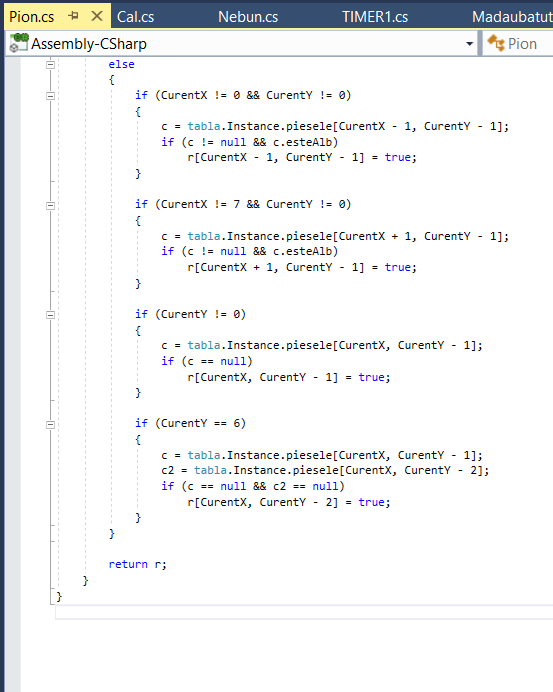
„Tura”

1. Algoritmul de generare al mișcării pionului este cel mai greu de implementat de departe, întru cat acesta este singurul algoritm, care necesită o verificare a pieselor din jurul său și este singura piesă, care poate fi transformată.

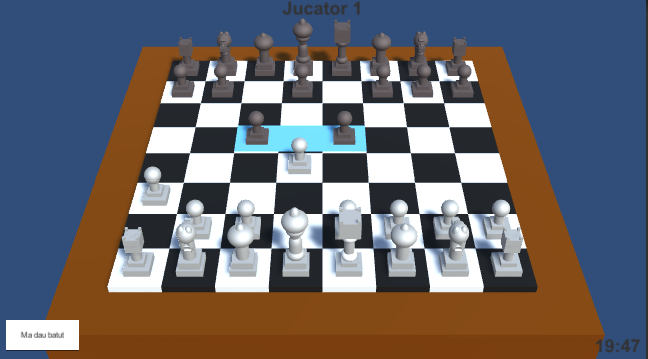
Algoritmul pionului construiește un tablou bidimensional de tip bool (r), în care vor fi salvate mutările pe care acesta le poate face. Dacă pionul nu a fost mutat de pe poziția sa inițială, acesta poate muta în fața 2 spații. Daca pionul întâlnește în stânga sau în dreapta sa o piesă inamică, acesta o poate captura. Daca pionul este blocat de o piesă in fața sa, atunci acesta nu poate fi mutat.

In plus o alta condiție care trebuie verificată în ceea ce privește pionul este condiția de interior. Trebuie verificat ca pionul să se afle mereu în interiorul tablei. Această condiție este importantă pentru algoritmul de verificare a pozițiilor de șah a pieselor adverse. Daca pionul este in margine acesta va avea o condiție speciala în care se va verifica daca prezinta o piesa adversa in dreapta sau în stânga (x>=0&&x<=7).

Pentru a implementa algoritmul pionului a fost nevoie însă de al crea la dublu, în sensul că trebuie construit un algoritm atât pentru pionul alb cât și pentru pionul negru. Dacă pionul este alb, acesta va putea fi mutat pe matrice cu y+1 sau y+2,și dacă pionul este negru acesta va fi mutat pe matrice cu y-1 sau y-2.

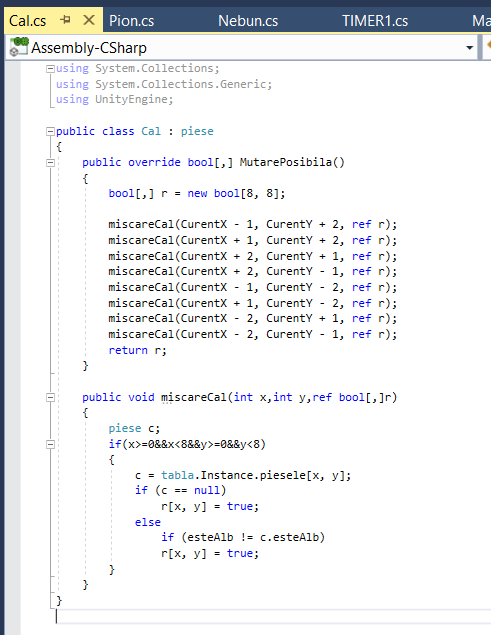


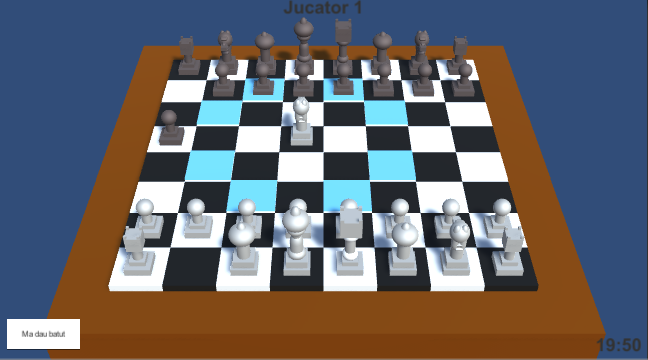
1 2



2. Algoritmul de generare a mișcării calului, presupune legătura între condiția de existență a calului, condiția de interior si o funcție care verifică mișcările calului.

Acest algoritm verifica dacă calul se afla în interiorul tablei sau pe margine, dacă poziția in care se poate muta este in interiorul tablei și dacă poziția nu este blocata de o anumită piesă. Dacă este blocată de o piesă adversa aceasta poate fi capturată.

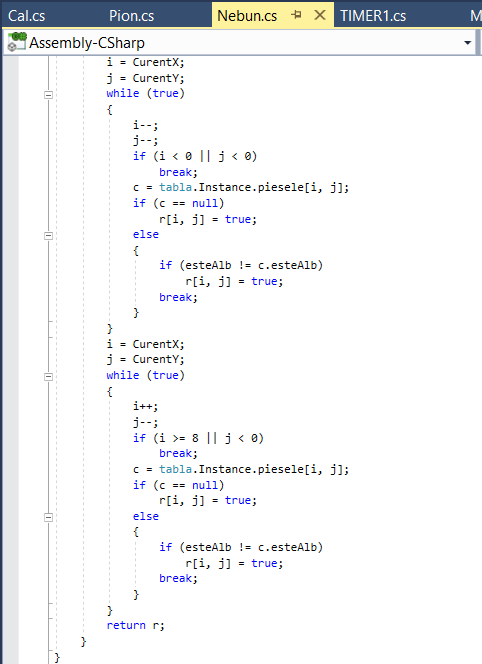
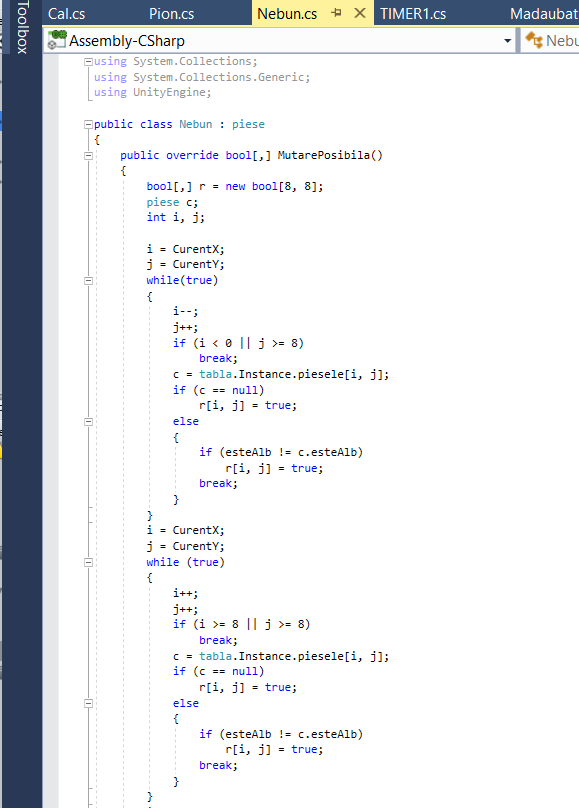
Algoritmul de generare a mișcărilor calului este cel mai eficient din punct de vedere al timpului, întru cat acesta verifica 8 poziții în jurul său, pe care poate sa fie mutat sau nu . Pentru a verifica mai ușor pozițiile am creat o funcție numita miscareCal, care va primi variabilele x, y (posibile mutări ale calului) și o referință la tabloul bidimensional r, pe care îl modifica ulterior. În final am apelat de 8 ori funcția cu fiecare mișcare posibilă a calul



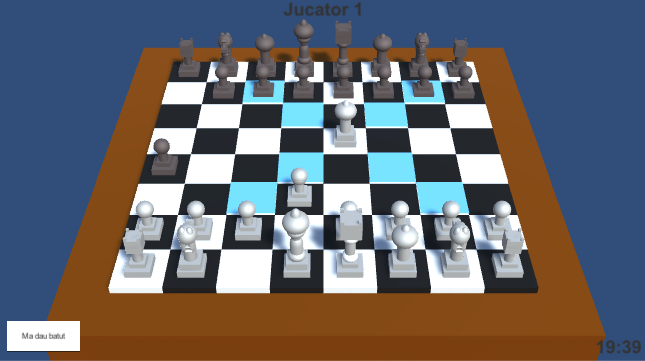
3. Algoritmul de generare a mișcării nebunului, presupune utilizarea unei condiții de existenta, a unei condiții de interior și a unei căutări pe diagonală.

Algoritmul începe de pe poziția nebunului selectat și va scrie în tabloul bidimensional r pozițiile pe care acesta poate muta. Având in vedere că mișcarea nebunului este pe diagonala acesta va verifica diagonala până când ajunge în capătul tablei de joc sau până când întâlnește o piesă. Dacă piesa este din echipa adversa acesta o va putea captura și se va opri. Dacă piesa este din echipa sa acesta se va opri în fața piesei.

Algoritmul nebunului presupune căutarea și verificarea succesiva a 4 direcții. Cu fiecare verificare acesta va marca in matricea r, pozițiile pe care acesta poate muta.

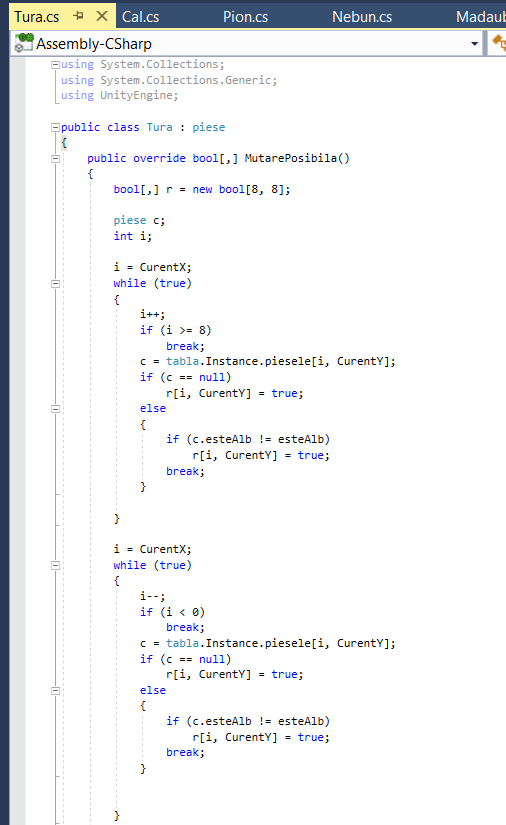


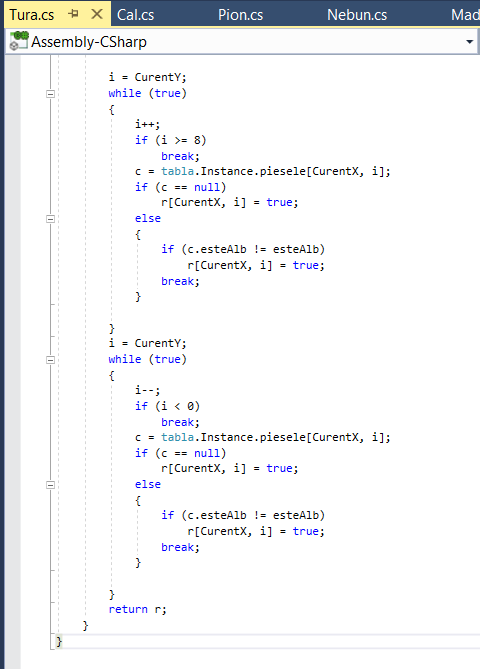
1 2



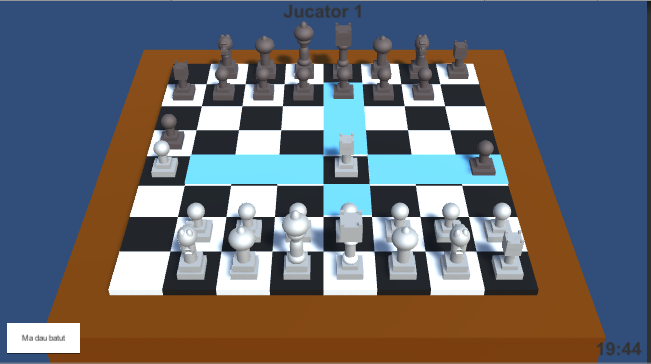
4. Algoritmul de generare a mișcării turei este asemănător cu cel al nebunului. Presupune utilizarea unei condiții de interior si a unei căutări liniare pe linie și coloană.

Asemănător cu algoritmul nebunului am salvat poziția turei și am căutat pe linia pe care se afla tura și pe coloana pe care se afla tura pozițiile pe care aceasta poate fi mutată. Am folosit o condiție de existență și o condiție de interior pentru a fi sigur că tura nu iese în afara tablei și am căutat pe linia și pe coloana turei pozițiile unde poate muta. Daca întâlnește o piesă din echipa ei aceasta se oprește în fata piesei. Daca întâlnește o piesă adversa aceasta o poate captura și se oprește pe poziția acelei piese.



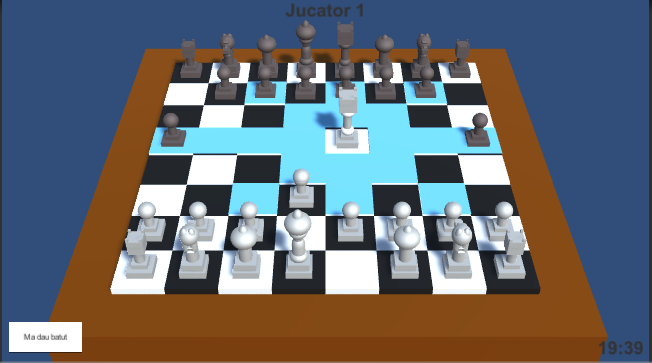


1 2



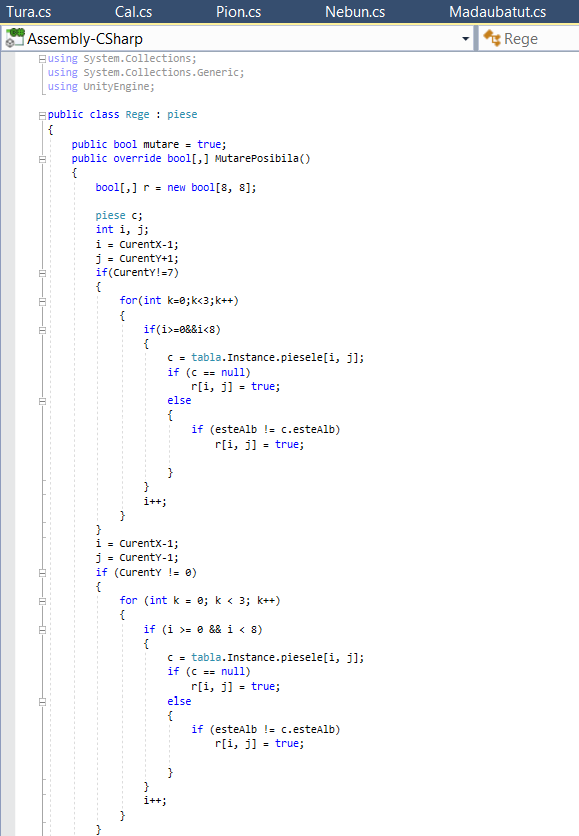
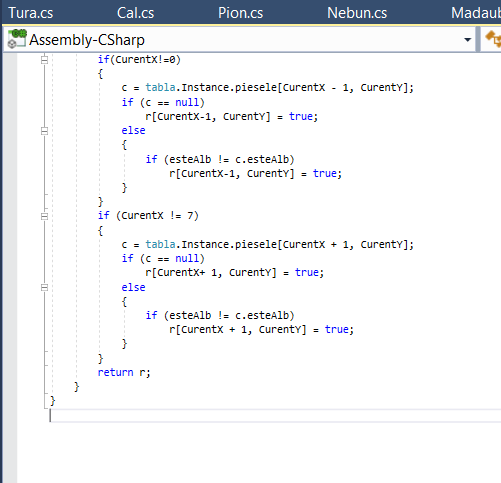
5. Algoritmul de generare a mișcării reginei, este de departe cel mai ușor algoritm de implementat dintre cele șase, întru cât reprezintă o combinație între algoritmul de generare al turei și algoritmul de generare a nebunului.

Generarea mișcărilor reginei presupune o condiție de existentă, una de interior și două căutări: una pe diagonală și una liniară pe linie și coloană. Am copiat fiecare algoritm de generare a mișcărilor nebunului și de generare a mișcărilor turei și le-am unit în unul singur.



6.Algoritmul de generare a mișcării regelui presupune o condiție de existenta, o condiție de interior și a unei verificări in jurul acestuia.

Dacă regele este selectat se verifica dacă acesta poate să se miște în una dintre direcțiile din jurul lui. Dacă regele sa află într-o margine sau într-un colț, dacă mișcările sale ies de pe tablă atunci acestea nu sunt posibile. Dacă mișcarea este blocata de o piesă a echipei sale mișcarea regelui nu este posibilă. Dacă este o piesă adversa aceasta poate sau nu sa fie mâncată, dacă regele nu va fi într-o poziție de șah după aceea.





Funcțiile SelecteazaPiesa si MutaPiesa funcționează simultan și complementar.

Daca nu exista o piesă selectată, se va utiliza funcția SelecteazaPiesa, pană când o piesă este selectata cu click stânga. Se va genera tabloul bidimensional de mișcare al piesei selectate in tabloul bidimensional MutarePosibila și va fi afișat cu ajutorul scriptului TablaHighlights. Dacă în tabloul dimensional MutarePosibila[i,j] = true, atunci pe poziția [i,j] va fi generat un plan de culoare albastru deschis.

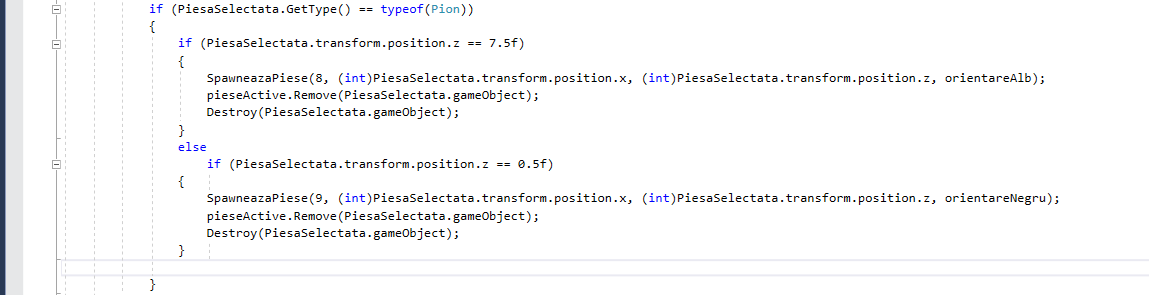
Dacă o piesa a fost selectata se utilizează funcția MutaPiesa și vor fi afișate pozițiile de mutare a piesei selectate. Dacă jucătorul selectează o poziție afișată piesa selectată va fi mutata în poziția respectivă și se va schimba jucătorul. Dacă jucătorul nu alege o poziție selectată, atunci PiesaSelectata va deveni null și va trebui selectată o altă piesă.

# CAPITOLUL VII – CONDITII SPECIALE

În jocul de șah exista doua condiții speciale și anume condiția de schimbare a pionului care ajunge la capătul tablei cu o regina și condiția de rocadă. Aceste doua condiții necesită îmbunătățirea funcțiilor de SelecteazaPiesa și MutaPiesa.

1. Algoritmul de schimbare a pionului cu regină, când ajunge în capătul tablei a presupus 2 condiții de verificare în funcție de culoarea pionului. Dacă pionul ajunge în capătul de sus al tablei(y=7) atunci acesta este alb și se va transforma în regină albă. Dacă pionul negru ajunge în capătul de jos al tablei (y=0) atunci acesta este negru și se va transforma în regină neagră.

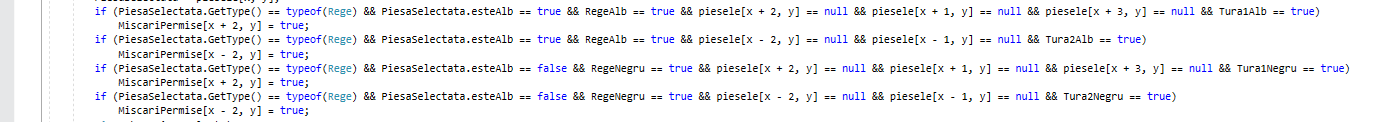
Pentru acest lucru, după ce piesa este mutata, în funcția MutaPiesa am introdus o secvență care verifică dacă piesa este pion și poziția acesteia pe tablă, în urma căreia o transforma in regină.

****

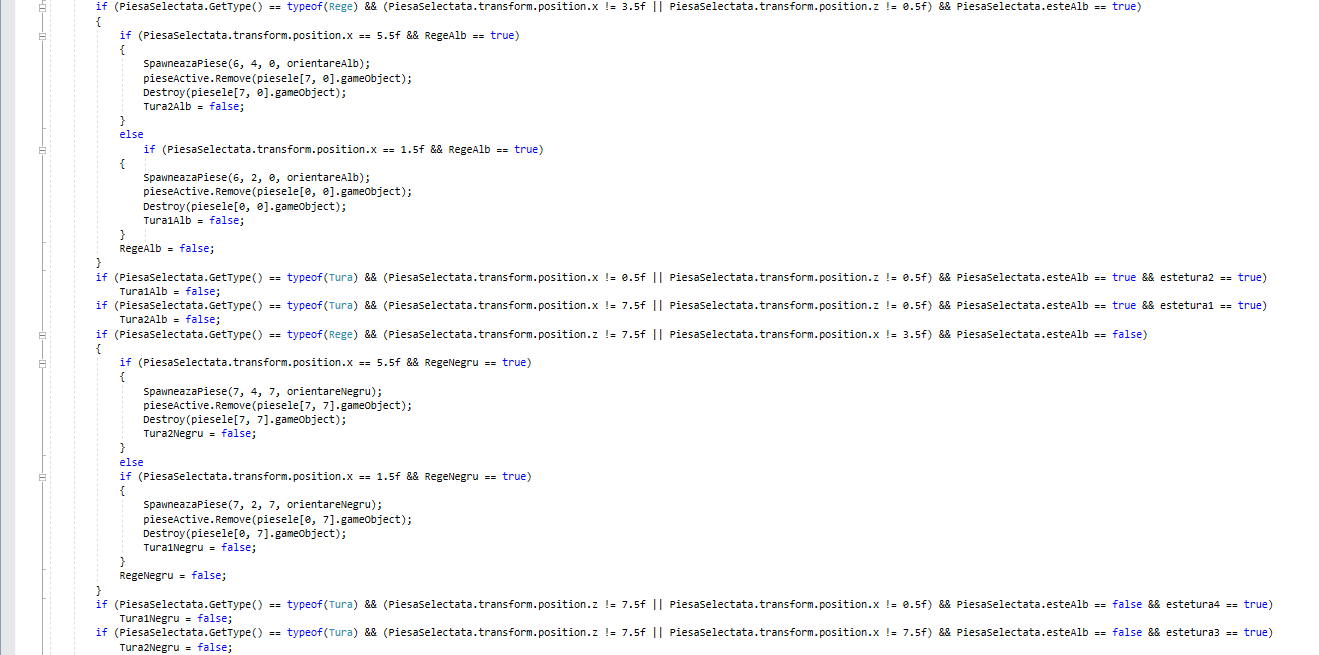


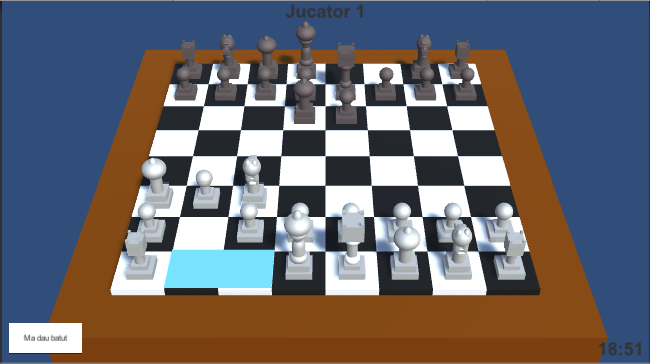
2. Algoritmul de rocadă are la baza 6 variabile de bool care verifica dacă regele sau turele au fost mutate sau au fost capturate. Dacă au fost capturate sau mutate variabilele bool devin false. Dacă piesele nu au fost mutate și între rege și ture este un spațiu gol, atunci poate avea loc rocada.

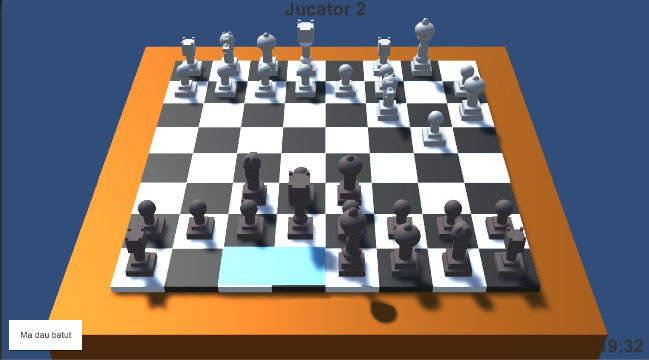
În funcția SelecteazaPiesa va fi afișată posibilitatea de rocadă, modificând tabloul bidimensional cu MiscariPosibile astfel încât regele va putea muta doua spatii.

****

În funcția MutaPiesa se va putea face schimbul între tură și rege, dacă se poate face rocada. În plus dacă cumva regele sau tura sunt mutate variabilele de tip bool ce verifica posibilitatea rocadei devin false.







# CAPITOLUL VIII – ALGORITMUL DE ȘAH

Algoritmul de șah reprezintă a doua cea mai importantă parte a scriptului „tabla”. Acesta constă într-un set de verificări, care oferă posibilitatea de a muta sau nu și de a încheia jocul. Dacă un rege este în șah mat acesta nu mai poate muta și este practic forțat să se dea bătut (să apese butonul „Ma dau batut”).

Condițiile care trebuie îndeplinite pentru ca sa se verifice condiția de șah mat sunt:

* Regele este în șah;
* Regele este în pat;
* Piesa care atacă regele poate fi blocată;
* Piesa care atacă regele poate fi capturată;
* Dacă este pat și mai sunt alte piese pe harta care pot fi mutate;

Aceste condiții impun si o serie de restricții:

* Dacă regele este atacat ești oblicat sa îl muți/ sa îl salvezi;
* Dacă o piesă poate bloca piesa care atacă regele, aceasta poate fi selectată și mutată doar ca să o blocheze pe cea care atacă;
* Dacă o piesă poate captura piesa care atacă regele, aceasta poate fi selectată și mutată doar în poziția piesei care atacă regele;

Ținând cont de cele de mai sus șahul va fi verificat în funcția SelecteazaPiesa.

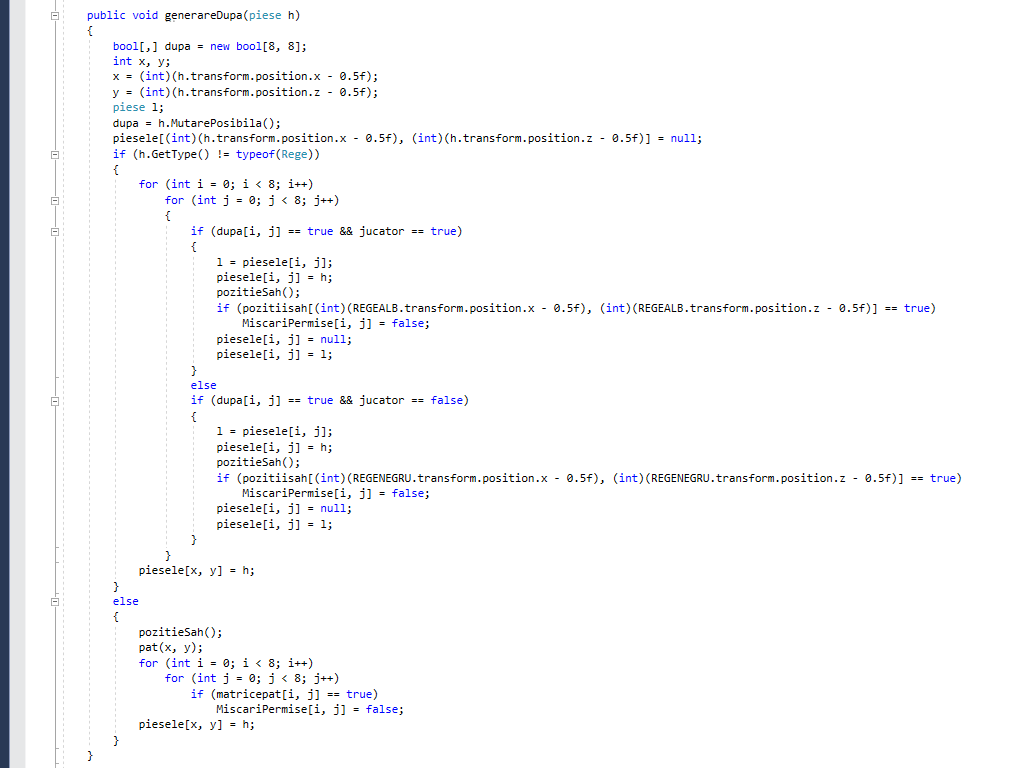
Cel mai important lucru pentru a deduce șah matul îl reprezintă pozițiile în care regele poate fi atacat. Astfel am creat funcția pozitieSah, care generează un tablou bidimensional de tip bool, numit pozitiisah, în care se salvează pozițiile în care regele poate fi atacat. Dacă regele este atacat, atunci piesa care îl ataca pe rege va fi salvata în variabila GameObject P.

Pentru a salva eficiența, am generat mutările de atac în tabloul bidimensional pozitiisah cu ajutorul scripturilor de generare a pieselor, dacă piesa este diferita de pion. Dacă piesa este un pion, atunci aceasta va prezenta o condiție speciala, de generare a mișcărilor de capturare e pieselor în fața sa, în partea stângă și în partea dreapta.

În plus pentru a verifica daca regele este înconjurat am folosit o funcție pat, care verifică dacă regele mai poate fi mișcat utilizând informațiile din tabloul bidimensional pozitiisah.

Cele doua funcții prezentate mai sus au fost utilizate pentru a construi alte doua funcții care sunt utilizate in permanenta înainte de fiecare selectare de piesa.

1. Prima funcție a fost denumit generareDupa și este cea care verifică dacă putem mișca o piesa și dacă această mișcare ne pune regele într-o poziție de șah.

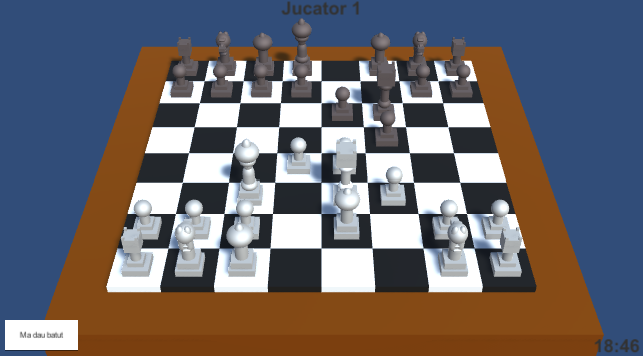


După cum sugerează și numele această funcție verifică dacă piesa selectată este sau nu rege și generează fiecare scenariu posibil, în care acea piesa este mutată. Daca piesa selectata nu este rege atunci ea este salvata in variabila GameObject L și se generează toate mutările posibile în tabloul bidimensional de tip bool, denumit „dupa”.

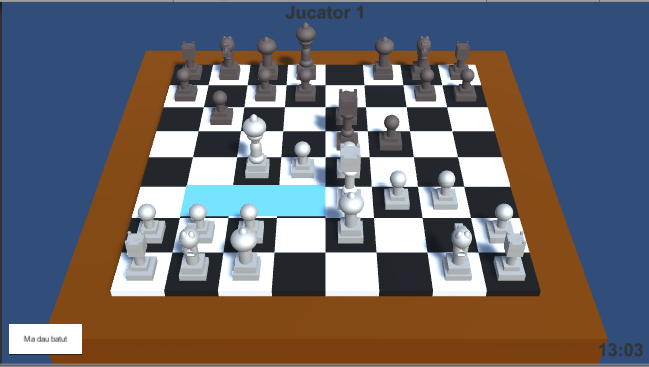
După generarea mișcărilor posibile a piesei selectate, această funcție caută în tabloul bidimensional „dupa” o poziție în care piesa poate fi mutată și o muta pe acea poziție. Apoi se realizează din nou matricea pozitiisah. În aceasta se verifică dacă regele este atacat. Dacă regele este atacat, atunci acea mutare nu este permisă, întru cât regele nu poate fi în șah.

Dacă piesa selectata este un rege, această funcție va împiedica ca acesta sa fie mutat într-o poziție în care este atacat, utilizând tabloul bidimensional matricepat.

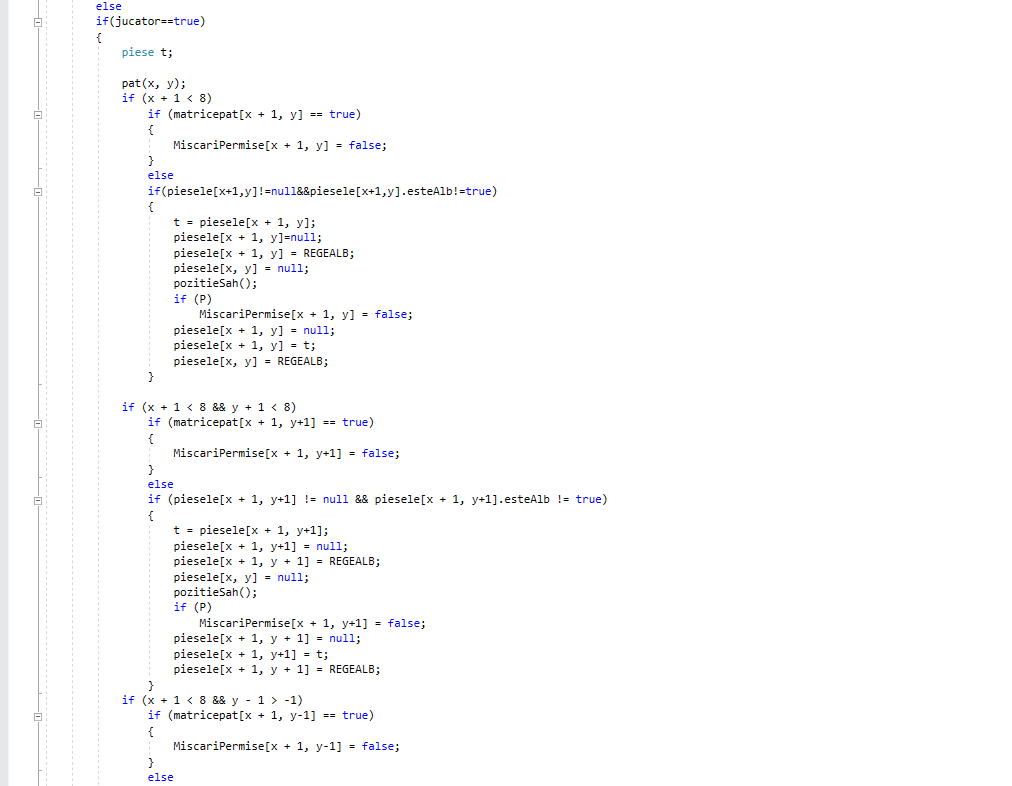
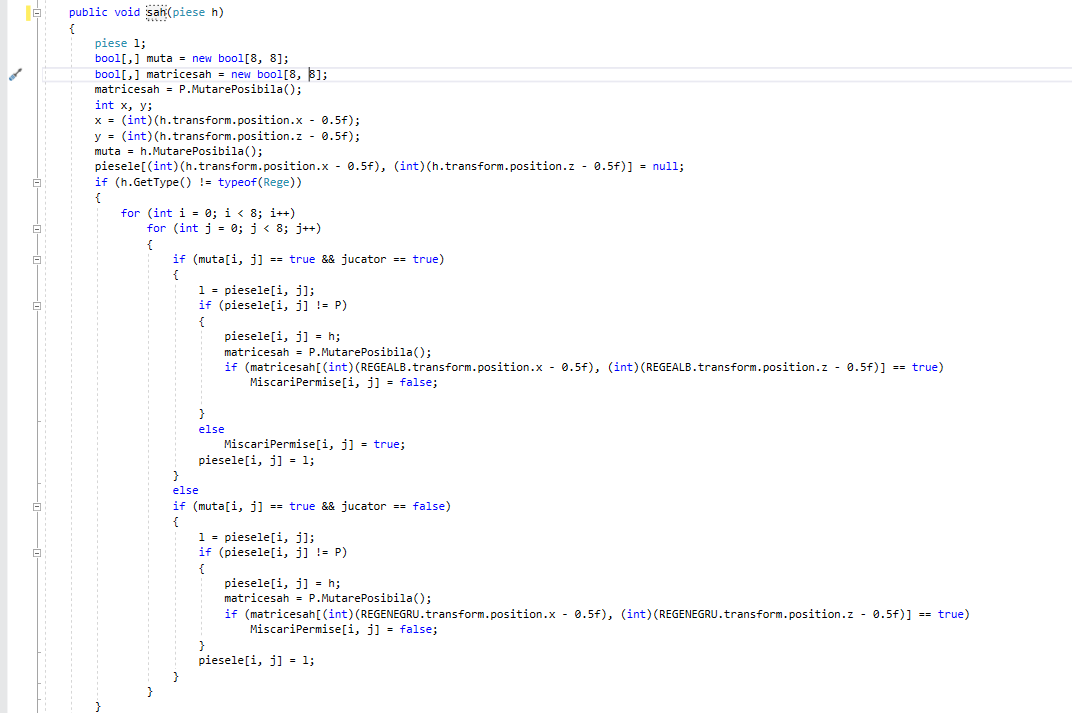
În acest exemplu pionul nu poate fi mutat întru cât regele va fi atacat de regina.



În acest exemplu regele nu poate fi mutat întru cat va fi atacat de nebun și regină.

****

2. A doua funcție a fost denumita „șah” și reprezintă funcția de șah propriu-zisă. Această funcție, asemănător cu funcția generareDupa, folosește o salvare și o generare, pentru a verifica dacă regele sau alte piese pot fi mutate, în condiția în care regele este deja atacat.



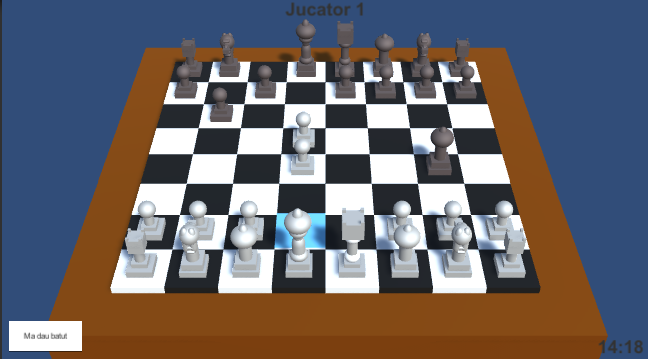
Secvența de mai sus, realizează o generare a tuturor posibilităților de mutare, a tuturor pieselor. Această funcție este apelată doar dacă regele este atacat și practic verifica dacă acesta poate fi mutat în condițiile în care este apelată.

Dacă regele este atacat, fiecare piesă este obligată practic să încerce să îl salveze. Dacă jucătorul dorește sa mute o piesă, această funcție obligă piesa să poată fi mutată doar în cazul în care poate să captureze piesa care ataca regele sau în cazul în care poate fi mutata în fața regelui, salvându-l. Dacă jucătorul dorește să mute regele acesta este obligat sa mute într-o poziție în care nu este atacat.

În exemplul de mai joc regina este forțată sa mute în fata regelui, pentru a-l salva, întru cât acest este atacat.



În exemplul de mai jos regele este forțat sa mute, întru cât acesta este atacat.



Există și alte modalități de a verifica condițiile de șah, însă cea prezentata mai sus este cea mai eficientă din punct de vedere a timpului și relativ din punct de vedere a spațiului.

Dacă nicio piesă nu poate fi mutata ca să salveze regele și nici regele nu poate fi mutat atunci jocul s-a terminat și jucătorul este forțat să apese butonul „Ma dau batut”, sau să aștepte să se termine timpul.

# BIBLIOGRAFIE

1. <https://en.wikipedia.org/wiki/Unity_(game_engine)>

2. <https://en.wikipedia.org/wiki/Checkmate>

3. <https://en.wikipedia.org/wiki/Chess>

# 