Atestat Profesional

Tema: Mecanica în jocurile video

Profesor coordonator:

VELICU ANTONETA

Elev:

DOBRE MARIA-ADINA

Clasa 12D

- 2022 -

Contents

Argument

Am ales acesta tema pentru ca jocurile video sunt foarte populare in lumea adolescentilor, dar nu toata lumea înțelege cum sunt acestea create și cum funcționeaza mișcarea “playerului” (partea din joc controlata de catre cel care joaca).

Mini-jocul creat este un bun exemplu in care se poate observa mecanica unui joc simplist (modul cum este controlat de jucator, modul in care se schimba scenele, dinamica jocului si inteligenta artificiala).

Capitolul 1: Prezentarea softului folosit

1.Unity

Unity este un gameengine scris în C și C++ lansat inițial în 2005 (versiunea 1.0) al cărui ultim release stabil a avut loc acum câteva luni (martie 2016, versiunea 5.3.4). Unity se bucură de popularitatea pe care o are mulțumită mai multor factori: oferă soluții pentru dezvoltarea aplicațiilor atât 2D cât și 3D ce pot fi rulate pe un număr mare de platforme (Linux, Mac, Windows, Android, iOS, WebGL, BlackBerry, Xbox, Play Station, Nintendo WiiU & 3DS, etc.). Portabilitatea nu se limitează doar la mediile în care aplicațiile Unity pot fi rulate. Astfel, dezvoltatorii nu sunt forțați să opteze pentru o anumită platformă în vederea dezvoltării, și au posibilitatea de a utiliza atât Windows și MacOS, cât și Linux (Ubuntu încă în beta). Unul dintre aspectele care sau dovedit a fi foarte utile în procesul de dezvoltare al aplicației mele este posibilitatea de a scrie și utiliza scripturi pentru aproape orice, cele mai utilizate limbaje fiind C# și Javascript. Personal, am ales să scriu în C# (în condițiile în care nu mai folosisem niciodată C# până atunci și găsisem chiar și posibiliatatea de a folosi python), principalele motive fiind ușurința aplicării unei gândiri faptul că marea majoritate a comunității încurajează utilizarea acestor limbaje pentru a ușura rezolvarea (cu sau fără ajutor extern) a potențialelor probleme ce pot apărea dea lungul procesului de dezvoltare.

Un alt aspect esențial care contribuie la experiența dezvoltării în Unity este prezența Asset Storeului. Asset Store este o platformă de cumpărare a unor assets create în majoritatea timpului de către comunitate. Un asset poate fi un grup de modele, sunete, scripturi, texturi, etc ce formează un tot unitar ce poate fi importat cu ușurință întro aplicație Unity. Pentru o mai ușoară exemplificare a noțiunii de asset, poate fi observată importanța asseturilor în proiectul de față. În această aplicație sunt folosite două asseturi: FPSController și Staff of Pain. FPSController este un asset care crează un personaj controlabil de către utilizator cu o perspectivă “firstperson” (persoana I utilizatorul observă universul jocului prin ochii personajului). Acest asset este un pachet care include scripturi pentru detectarea și tratarea inputului utilizatorului, sunete pentru pași, un collider (un obiect cu proprietăți fizice, folosit în cazul de față pentru a detecta coliziuni cu suprafața podelei) și o cameră atașată acestui collider (camera reprezintă perspectiva din care poate privi un utilizator universul aplicației). Majoritatea asseturilor, precum acesta, conțin și unul sau mai multe

“prefabs” care reprezintă un obiect al jocului prefabricat ce include toate caracteristicile menționate mai sus și le configurează pentru a lucra împreună ca un totunitar.

Unity oferă utilizatorilor posibilitatea de a crea jocuri și experiențe în ambele 2D și 3D, iar motorul oferă un API principal de scriptare în C #, atât pentru editorul Unity sub formă de plugin-uri, cât și pentru jocuri, precum și tragere și plasare funcționalitate. Înainte ca C # să fie principalul limbaj de programare utilizat pentru motor, acesta era acceptat anterior Boo, care a fost eliminat odată cu lansarea Unity 5,[29] și o versiune de JavaScript numit UnityScript, care a fost depreciat în august 2017, după lansarea Unity 2017.1, în favoarea lui C #.

În cadrul jocurilor 2D, Unity permite importul de sprite și un renderer mondial avansat 2D. Pentru jocurile 3D, Unity permite specificarea compresia texturii, mipmaps, și setările de rezoluție pentru fiecare platformă acceptată de motorul de joc,[31] și oferă suport pentru cartografiere cu umflături, cartografierea reflexiei, cartografierea paralaxei, spațiul ecranului ocluzia ambientală (SSAO), umbre dinamice folosind hărți umbre, redarea texturii și efecte de post-procesare pe ecran complet.

Creatorii pot dezvolta și vinde active generate de utilizatori către alți producători de jocuri prin Unity Asset Store. Aceasta include activele și mediile 3D și 2D pe care dezvoltatorii le pot cumpăra și vinde.[62] Unity Asset Store a fost lansat în 2010. Până în 2018, au existat aproximativ 40 de milioane de descărcări prin intermediul magazinului digital.

Interfața

(A) Bara de instrumente oferă acces la cele mai esențiale funcții de lucru. În stânga conține instrumentele de bază pentru manipularea vizualizării scenei și GameObjects în cadrul acestuia. În centru sunt comenzile de redare, pauză și pași. Butoanele din dreapta vă oferă acces la Unity Collaborate, Unity Cloud Services și contul dvs. Unity, urmate de un meniu de vizibilitate a stratului și, în final, de meniul de aspect Editor (care oferă câteva aspecte alternative pentru ferestrele Editor și vă permite să salvați propriile machete personalizate).

(B) Fereastra Ierarhie este o reprezentare text ierarhică a fiecărui GameObject din scenă. Fiecare element din Scenă are o intrare în ierarhie, astfel încât cele două ferestre sunt legate în mod inerent. Ierarhia dezvăluie structura modului în care GameObjects se atașează unul de celălalt.

(C) Vizualizarea Joc simulează cum va arăta jocul final redat prin camerele dvs. de scenă. Când faceți clic pe butonul Redare, începe simularea.

(D) Vizualizarea Scenă vă permite să navigați vizual și să editați Scena. Vizualizarea Scenă poate afișa o perspectivă 3D sau 2D, în funcție de tipul de proiect la care lucrați.

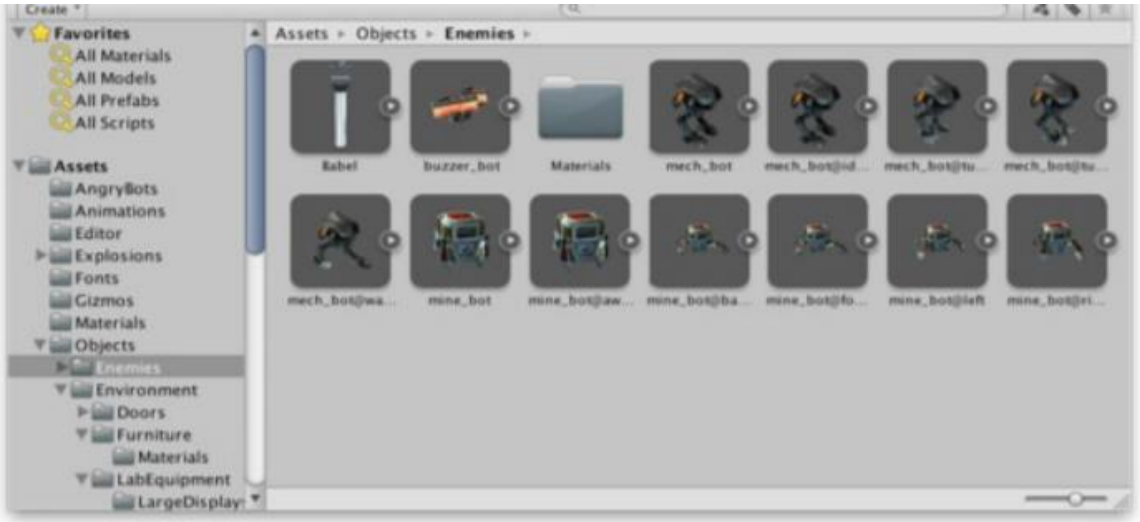
(E) Fereastra Inspector vă permite să vizualizați și să editați toate proprietățile GameObject-ului selectat în prezent. Deoarece diferite tipuri de GameObjects au seturi diferite de proprietăți, aspectul și conținutul Inspectorului fereastra se schimbă de fiecare dată când selectați un GameObject diferit.

(F) Fereastra Proiect afișează biblioteca dvs. de active care sunt disponibile pentru utilizare în proiect. Când importați active în proiect, acestea apar aici.

(G) Bara de stare oferă notificări despre diferite procese Unity și acces rapid la instrumentele și setările asociate.

Browser-ul proiectului

În aceasta fereastra vedeți mulțimea componentelor care formează proiectul dumneavoastră Partea din stânga a browserului vă prezintă proiectul structurat sub formă de listă de liste de elemente, ca nişte directoare. Atunci când alegeți un director, un dosar din listă şi dați un click asupra lui, el va fi deschis în partea din dreapta prezentându-i-se acolo conținutul. El poate include alte directoare, alte foldere, alte elemente. (n.n. Utilizatorii familiarizați cu Dolphin sau Nautilus de sub Linux vor observa asemănarea.) Bunurile indivizibile sunt arătate sub formă de iconițe care indică tipul lor: scripturi, materiale, dosare incluse în dosare etc.) Mărimea iconițelor -astfel ca să fie de o mărime convenabila pe display-ul dmneavoastră – se reglează din cursorul din stânga jos. Documentația spune că dacă mutați cursorul în extrema stângă nu mai apar iconițe



ci doar o listă de denumiri. Spațiul din stânga cursorului este folosit pentru a se scrie acolo, de

către sistem, numele ultimului element selectat, ba chiar şi calea pînă la el (de mare folos dacă ați efectuat o căutare pentru a-l găsi). O secțiune de elemente favorite numită chiar Favorites vă ține la îndemână obiectele cele mai des utilizate economisind astfel timpul care s-ar pierde pentru căutarea şi (re)căutarea lor. Partea de deasupra panoului de afişare îți arată traseul, calea – şirul de directoare parcurs – până la elementul vizat. Puteți da click pe oricare element de pe aceasta cale (este ramură în arborele de directoare) pentru a naviga mai departe pornind de la nivelul respectiv de imbricare al dosarelor. (ca şi cum ai sări într-un copac de la o ramură la alta.) În timpul căutării cu Search a unui obiect, a unui bun, sistemul va arăta aici unde s-a ajuns cu căutarea la momentul curent, dar şi numărul de obiecte gratuite şi negratuite disponibile în magazinul virtual de bunuri al comunității, numerele fiind separate cu o bară "/". În meniul Preference -> General al mediului Unity există o opțiune pe care o puteți alege pentru a dezactiva afişarea acestor căutări făcute pe internet, în magazinul virtual. E util mai ales dacă nu doriți să se comunice pe internet ce elemente apar în viitorul dumneavoastră joc, înaint e de lansarea acestuia.

Managerul de comenzi

Managerul de comenzi rapide vă permite să vizualizați și să gestionați comenzile rapide de la tastatură.

O comandă rapidă este orice tastă sau combinație de taste care este legată de o comandă Unity. O

Tasta	Semnificația
F	Cadrul (*frame)* selectat (adică arăta bunul ales plasat în dosarul ce-l conține)
Tab	Mută punctul de interes din coloana I în cea de- a II-a. (Cu două coloane, ca la N.C.)
Ctrl/Cmd + F	Câmp de căutare după *focus*.
Ctrl/Cmd + A	Selectează toate obiectele afişate pe listă.
Ctrl/Cmd + D	Duplică obiectele selectate.
Delete	Ştergere precedată de dialog de confirmare.
Delete + Shift	Ştergere neprecedată de dialog de confirmare.

comandă este o acțiune care este executată în Editor. De exemplu, tasta R este legată de comanda care activează instrumentul Scalare în vizualizarea Scenă

Accesați Managerul de comenzi rapide din meniul principal al Unity:

Pe Windows și Linux, selectați Editare > Comenzi rapide.

Pe macOS, selectați Unity > Comenzi rapide.

Tasta	Semnificația
Backspace + Cmd	Ştergerere neprecedată de dialog de confirmare. (OS X)
Enter	Începerea redenumirii elementului selectat. (OS X)
Cmd + down arrow	Deschide bunurile alese. (OS X)
Cmd + up arrow	Înapoi în folderul/dosarul părinte (OS X, două coloane)
F2	Începe redenumirea elementului selectat (Win)
Enter	Deschide bunurile alese. (Win)
Backspace	Înapoi în folderul/dosarul părinte (Win, două coloane)
Right arrow	Expandează elementul, bunul selectat (atât în mod *tree views* cât şi din rezultatele căutării). Dacă el este deja expandat, apăsarea pe săgeata dreaptă va selecta primul descendent.
Left arrow	Compactează elementul, bunul selectat (atât în mod *tree views* cât şi din rezultatele căutării). Dacă el este deja compactat, apăsarea pe săgeata stângă va selecta elementul său părinte (predecesorul).
Alt + right arrow	Expandează elementul, bunul selectat (în modul *showing assets as previews -prezentarea bunului înainte de achiziție*)
Alt + left arrow	Compactează elementul, bunul selectat (în modul *showing assets as previews -prezentarea bunului înainte de achiziție*)

Fereastra Inspectorului

Utilizați fereastra Inspector pentru a vizualiza și edita proprietăți și setări pentru aproape tot ce se află în Editorul Unity, inclusiv pentru GameObjects, componente Unity, Active, Materiale și setări și preferințe în Editor.

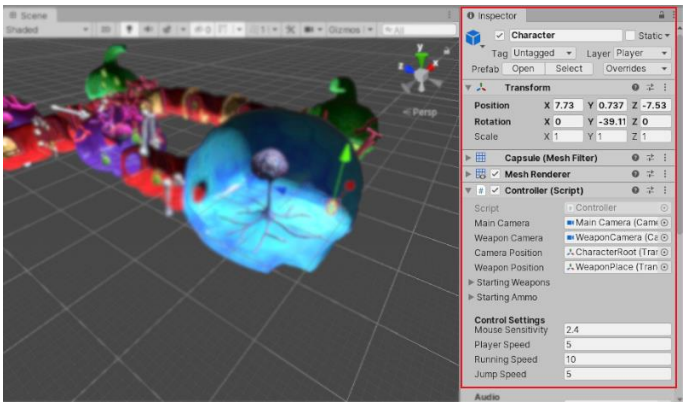
Deschiderea unei ferestre Inspector

Pentru a deschide o fereastră Inspector, efectuați una dintre următoarele:

Din meniu, selectați Windows > General > Inspector pentru a deschide o fereastră plutitoare Inspector.

Din meniul Mai multe articole din orice fereastră (⋮), selectați Adăugare filă > Inspector pentru a deschide un Inspector într-o filă nouă.Puteți deschide câte ferestre Inspector doriți și le puteți repoziționa, andoca și redimensiona în același mod în care puteți face orice altă fereastră.

Controlul focalizării ferestrei Inspector



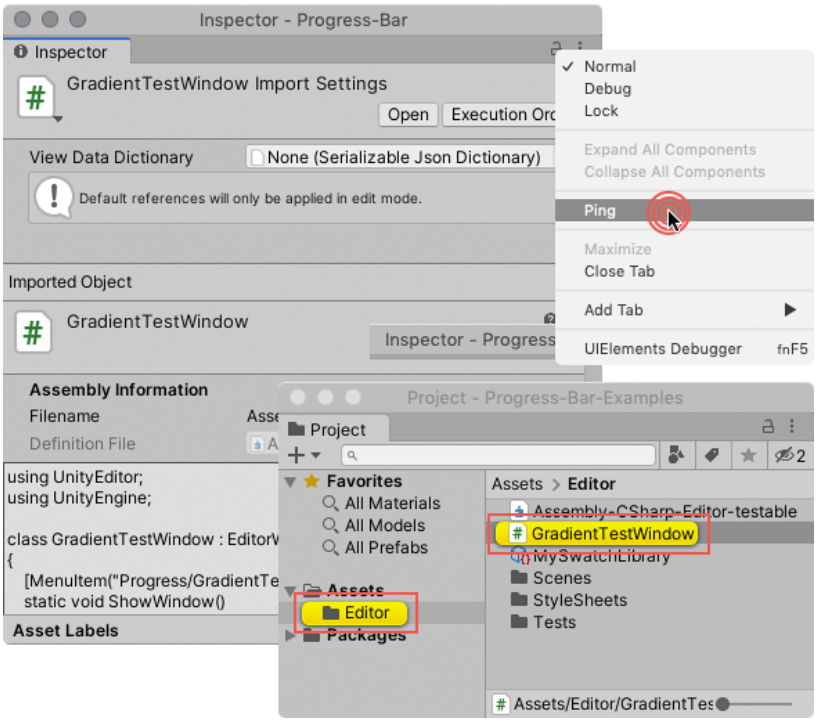
În mod implicit, o fereastră Inspector afișează proprietăți pentru selecția curentă. Conținutul Inspectorului se modifică ori de câte ori selecția se schimbă. Pentru a menține deschis același set de proprietăți, indiferent de selecția curentă, efectuați una dintre următoarele:

Blocați fereastra Inspector la selecția curentă. Când blocați o fereastră Inspector, aceasta nu se mai actualizează dacă modificați selecția.

Deschideți un Inspector concentrat pentru un GameObject, Asset sau componentă. Inspectorii concentrați afișează întotdeauna numai proprietățile elementelor pentru care le-ați deschis.

Inspectarea articolelor

Ceea ce puteți vedea și edita într-o fereastră Inspector depinde de ceea ce selectați. Această secțiune descrie ce afișează o fereastră Inspector pentru diferite tipuri de elemente pe care le puteți selecta.



Inspectarea GameObjects

Când selectați un GameObject (de exemplu, în vizualizarea Ierarhie sau Scenă), inspectorul afișează proprietățile tuturor componentelor și materialelor sale. Puteți edita proprietățile

Inspectarea componentelor de script personalizate

Când GameObjects au atașate componente de script personalizate, inspectorul afișează scripturile' variabile publice. Puteți edita variabilele de script în același mod în care editați orice alte proprietăți, ceea ce înseamnă că puteți seta parametrii și

valorile implicite în scripturile dvs. fără a modifica codul.Pentru mai multe informații, consultați Variabile și Inspector în secțiunea Scripting.

Inspectarea Activelor

Când selectați un activ (de exemplu, din fereastra Proiect), inspectorul afișează setări care controlează modul în care Unity importă și utilizează Asset-ul în timpul execuției.Fiecare tip de activ are propriile setări. Exemple de setări de import de materiale pe care le editați într-o fereastră Inspector includ:

- Fereastra Setări pentru importul modelului.
- Fereastra Setări de importare a clipurilor audio.
- Fereastra Setări import text.

Inspectarea setărilor și preferințelor

Când deschideți Setările proiectului

(meniu: Editor > Setări proiect), Unity le afișează într-o fereastră Inspector.

Inspectarea prefabricatelor

Când lucrați cu prefabricate, fereastra Inspector afișează câteva informații suplimentare și oferă câteva opțiuni suplimentare. De exemplu:

Când editați o instanță prefabricată, fereastra Inspector oferă opțiuni pentru lucrul cu materialul prefabricat și pentru aplicarea înlocuirilor.

Inspectarea mai multor articole

Când aveți două sau mai multe elemente selectate, puteți edita toate proprietățile pe care le au în comun într-o fereastră Inspector. Unity copiază valorile pe care le furnizați tuturor articolelor selectate. Fereastra Inspector afișează un număr de elemente selectate.

Mai multe GameObjects

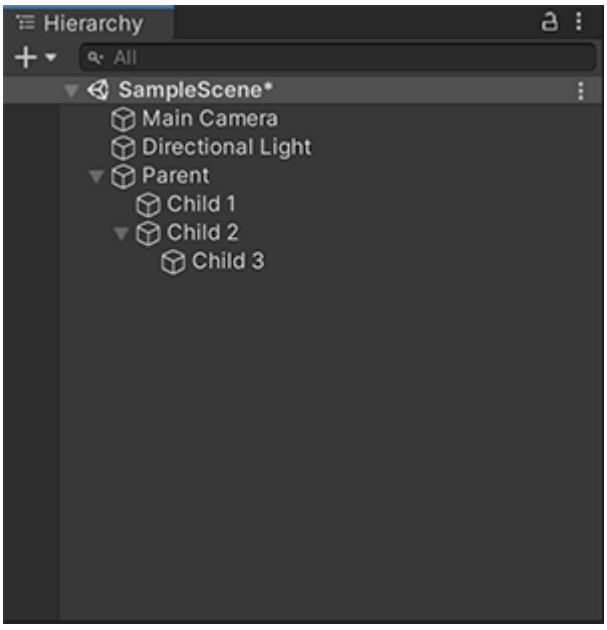
Când selectați mai multe GameObjects, fereastra Inspector afișează toate componentele pe care le au în comun.

Pentru valorile proprietăților care sunt diferite pentru două sau mai multe GameObjects selectate, inspectorul afișează o liniuță (-) (1 în captura de ecran de mai jos).Pentru valorile proprietăților care sunt aceleași pentru toate GameObjects selectate, inspectorul afișează valorile reale (2 în captura de ecran de mai jos).Pentru a aplica o valoare a proprietății de la un GameObject selectat la toate GameObjects selectate, faceți clic dreapta pe numele proprietății și selectați Set to Value of [Name of GameObject] din meniul contextual (3 în captura de ecran de mai jos).

Dacă oricare dintre GameObjects selectate are componente care nu sunt prezente pe celelalte obiecte selectate, inspectorul afișează un mesaj că unele componente sunt ascunse.

Fereastra Ierarhie

Fereastra Ierarhie afișează fiecare GameObject într-o scenă, cum ar fi modele, camere foto sau prefabricate.Puteți folosi fereastra Ierarhie pentru a sorta și grupa GameObjects pe care le utilizați într-o Scenă. Când adăugați sau eliminați GameObjects în vizualizarea Scenă, le adăugați sau eliminați și din fereastra Ierarhie.



Fereastra Ierarhie poate conține și alte Scene, fiecare Scenă conținând propriile GameObjects.

PărinteUnity folosește conceptul de ierarhie părinte-copil, sau parenting, pentru a grupa GameObjects. Un obiect poate conține alte GameObjects care își moștenesc proprietățile. Puteți lega GameObjects împreună pentru a ajuta la mutarea, scalarea sau transformarea unei colecții de GameObjects. Când mutați obiectul de nivel superior sau GameObject părinte, mutați și toate GameObjects copil. De asemenea, puteți crea GameObjects părinte-copil imbricate. Toate obiectele imbricate sunt încă descendenți ai GameObject părinte original sau GameObject rădăcină.GameObjects copil moștenește mișcarea și rotația GameObject-ului părinte.

2. Visual Studio

Visual Studio include un set complet de instrumente de dezvoltare pentru generarea de aplicații ASP.NET, Servicii Web XML, aplicații desktop și aplicații mobile. Visual Basic, Visual C++, Visual C# și Visual J# toate folosesc același mediu de dezvoltare integrat (IDE) care le permite partajarea instrumentelor și facilitează crearea de soluții folosind mai multe limbaje de programare. Aceste limbaje permit să beneficieze de caracteristicile .NET Framework care oferă acces la tehnologii cheie care simplifica dezvoltarea de aplicații web ASP și XML Web Services cu Visual Web Developer.Înainte de lansarea versiunii 4 produsele Visual Basic, Visual C + +, Visual FoxPro și SourceSafe s-au vândut separat. Visual Basic a fost disponibil în acel moment

deja la versiunea 3. Scopul a fost de introducere pentru Windows 95 îmbinarea în mod corespunzător a produselor.Visual Studio 97 poartă numele de cod Boston. Visual Studio 97 era disponibilă în ediţiile Professional Edition şi Enterprise Edition.[4] Visual Basic 5.0, Visual C++ 5.0 pentru programarea Windows; Visual J++ 1.1 pentru programarea Java; Visual FoxPro 5.0 pentru dezvoltarea de date de baze. Visual InterDev pentru crearea paginilor web utilizând Active Server Pages.Visual Studio 6.0 are numele de cod Aspen şi este ultima versiune care rulează pe platforma 9x. Include suport pentru Internet Explorer 4.0, control HTML dinamic, Windows NT 5.0 şi componente de infrastructură precum Microsoft Message Queue server, serviciul Active Directory şi tehnologia Microsoft's Zero Administration.[6] Era comercializat în două ediţii Professional şi Enterprises. Versiunea Enterprises include caracteristici care nu regăsim în versiunea Professional precum:

- Application Performance Explorer
- Automation Manager
- Microsoft Visual Modeler
- RemAuto Connection Manager
- Visual Studio Analyzer

Visual Studio 2010

Visual Studio 2010 are numele de cod Dev10şi a fost lansat pe 12 aprilie 2010 alături de .NET Framework 4. Visual Studio 2010 are un editor nou care utilizează WPF (Windows Presentation Foundation), sprijină interfaţa de tip Ribbon, suportă monitoare multiple, Windows 7 multitouch, funcţionalitatea SharePoint, instrumente de Windows Azure şi IntelliTrace, un nou produs care ajută la eradicarea bug-urilor irepetabile. Va veni furnizată împreună cu Expression Studio, Business & Enterprise Servers şi Microsoft Office şi în versiunile Ultimate şi Premium.

Noţiuni generale

Pentru gruparea fişierelor sursă şi a altor resurse utilizate în cadrul aplicaţiei, mediul Visual Studio .Net (VS) utilizează două concepte:

- Proiect: fişier care conţine toate informaţiile necesare pentru a compila un modul dintr-o aplicaţie .Net

- Soluţie: fişier care conţine lista proiectelor care compun o aplicaţie, precum şi dependinţele dintre ele

Proiectele sunt fişiere XML care conţin următoarele informaţii:

- lista fişierelor necesare, poziţia pe disc a acestora precum şi modul în care vor fi utilizate (compilate in cod executabil, incluse în executabil, …)
- lista de module externe referite
- mai multe seturi de parametri de compilare numite configuraţii (implicit sunt doar două – Debug şi Release – dar se pot defini şi alte configuraţii)
- diverse opţiuni legate de proiect

Fişierele de tip proiect pentru C# au extensia *csproj*. Principalele tipuri de proiecte sunt:

- Console Application: aplicaţii de tip linie de comandă, fără interfaţă grafică; rezultatul este un fişier executabil (.exe)
- Class Library: bibliotecă de clase care poate fi utilizată pentru construirea altor aplicaţii; rezultatul este o bibliotecă cu legare dinamică (.dll)
- Windows Application: aplicaţie windows standard; rezultatul este un fişier executabil (.exe)

Capitolul 2: Prezentarea Aplicatiei

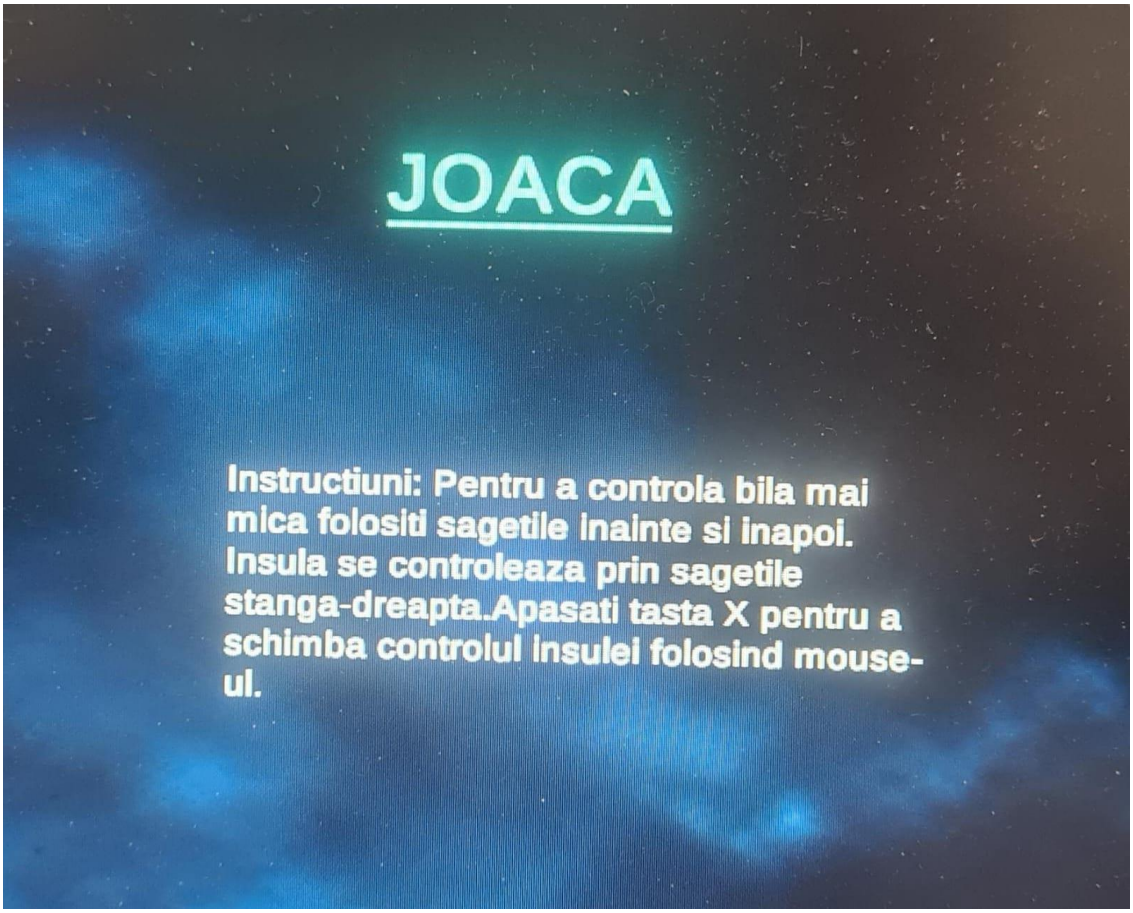
Instructiuni de untilizare:

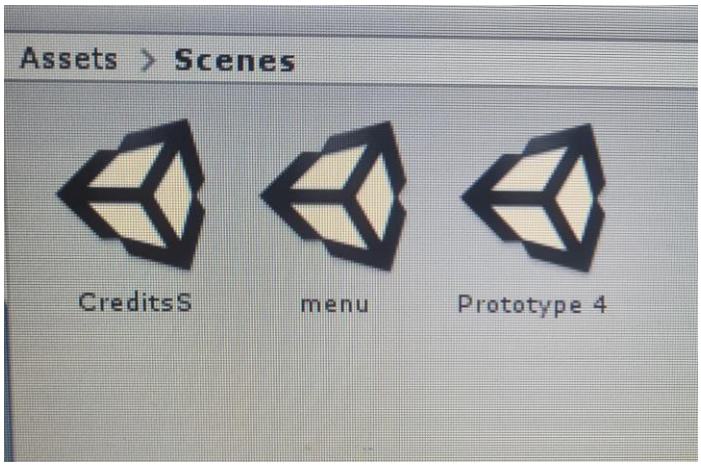
Deschideti documentatia "Atestat Informatica Dobre Maria-Adina.pdf" cu ajutorul aplicatiei Word pentru a citi documentatia.

Deschideti folderul "Proiect Atestat-Dobre Maria-Adina"in care se afla aplicatiile care ajuta la rularea programului "Prototype 4.exe".

Programul contine mai multe scene care include butoane de navigare si instructiuni precum:

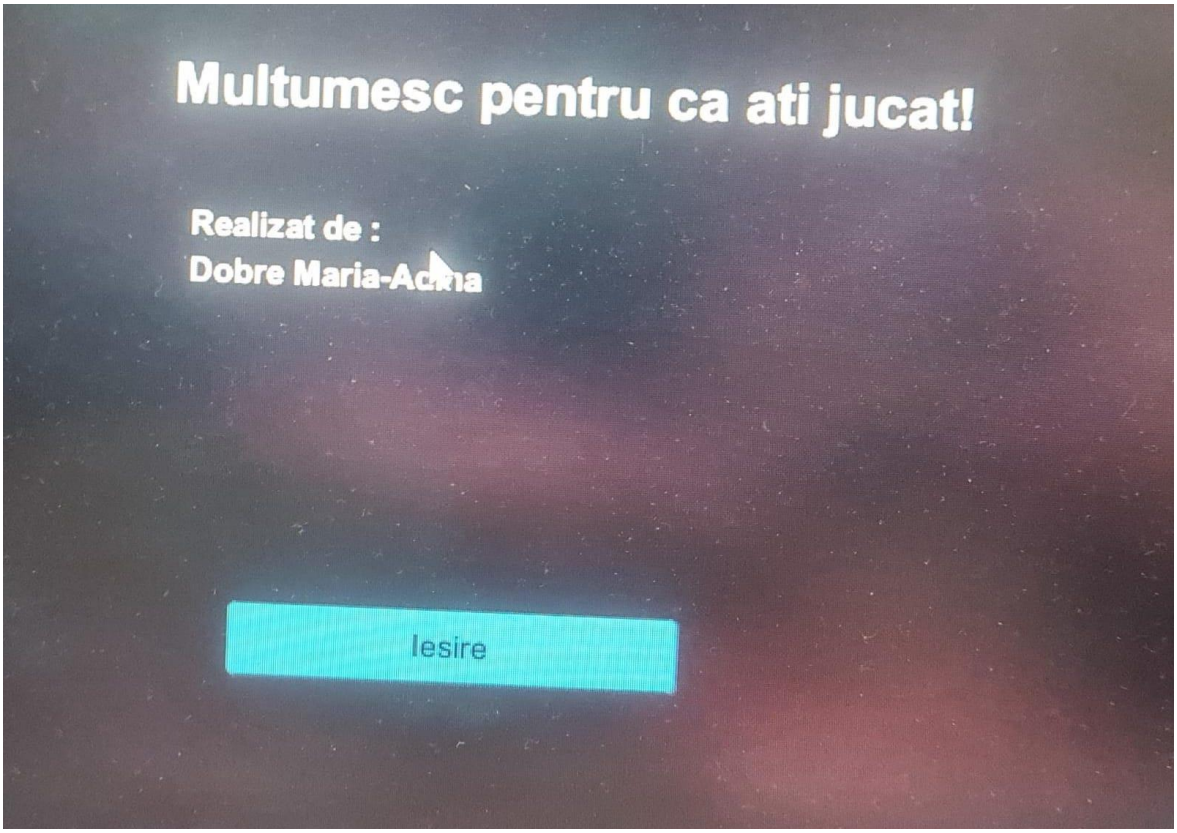
- Scena “menu” este prima pagina care include butonul “Joaca” si o serie de instructiuni de control al “playerului”.





- Scena numita “Prototype 4 “ este jocul propriu-zis unde se intampla actiunea : jucatorul are control asupra unor elemente folosind tastele mentionate in instructiuni.

- Scena “CreditsS” este pagina de final care contine credite, multumiri si butonul “Iesire” care permite parasirea aplicatiei.



Jocul propriu-zis; Explicarea mecanicii din joc.

Aceasta este scena pe care o vede jucatorul. Reprezinta o platforma unde jucatorul poate controla atat bila mai mica, dar si rotatia insulei. Controlul se poate efectua din mouse sau din taste (dupa preferintele jucatorului). Jucatorul urmeaza sa se dueleze cu un adversar ‘bot’ care are inteligenta artificiala. Astfel, adversarul este greu de aruncat de pe scena din cauza unor bariere invizibile care sunt activate numai la coliziunea adversarului cu ele. Jucatorul trebuie sa darame adversarul de pe insula. Ezista tot felul de potiuni speciale care ‘dau putere’ jucatorului. Aici intervine mecanica jocului: bila mica este mai usoara decat bila mare si deci este mai usor de aruncat in afara suprafetei de joc. Odata aruncata, asupra bilei actioneaza ‘gravitatea’. Acest aspect este vizibil si cand bila sare folosind tasta ‘space’. Cele doua bile sunt ‘corpuri solide’ deci se pot impinge prin atingere. Se remarca impulsul la atingere dat de masa si viteza corpului. Astfel, in acest joc este bine exemplificata mecanica predata in liceu. La crearea acestui joc, elementele de mecanica au fost create prin cod.

Capitolul 3: Limbajele C++ si C#

C# este un limbaj de programare orientat-obiect conceput de Microsoft la sfârșitul anilor 90. A fost conceput ca un concurent pentru limbajul Java. Ca și acesta, C# este un derivat al limbajului de programare C++.C# simplifică mult scrierea de programe pentru sistemul de operare Windows, iOS, Android etc. Este un limbaj de programare cross-platform.

Exemplu de program simplu Windows scris în Managed C++ (C++/CLI) și C#:

Cod scris în Managed C++ (C++/CLI):

```
public:
int main(array<System::String ^> ^args)
{
        // Activarea efectelor vizuale Windows XP înainte de crearea oricărui control
        Application::EnableVisualStyles();
        Application::SetCompatibleTextRenderingDefault(false);

        // Crearea și rularea ferestrei principale
        Application::Run(gcnew Form1());
        return 0;
}
```

Cod scris în C#:

```
public static void Main()
{

    Form1 form1 = new Form1();
    form1.Show();
    Application.Run(form1);
}
```

C++ (pronunțat ca în engleză /ˈsi: plʌs plʌs/) este un limbaj de programare general, compilat. Este un limbaj multi-paradigmă, cu verificarea statică a tipului variabilelor ce suportă programare procedurală, abstractizare a datelor, programare orientată pe obiecte. În anii 1990, C++ a devenit unul dintre cele mai populare limbaje de programare comerciale, rămânând astfel până azi[1].

Bjarne Stroustrup de la Bell Labs a dezvoltat C++ (inițial denumit *C cu clase*) în anii 1980, ca o serie de îmbunătățiri ale limbajului C. Acestea au început cu adăugarea noțiunii de clase, apoi de funcții .virtuale, suprascrierea operatorilor, moștenire multiplă (engleză *multiple inheritance*), șabloane (engleză *template*) și excepții. Limbajul de programare C++ a fost standardizat în 1998 ca și ISO 14882:1998, versiunea curentă fiind din 2003, ISO 14882:2003. Următoarea versiune standard, cunoscută informal ca C++0x, este în lucru.

Program care afișează textul "Hello World":

```
#include <iostream>

using namespace std;

int main() {
   cout << "Hello World" << endl;
   return 0;
}
```

Tipuri de date:

- bool este unul dintre tipurile de date simple. El contine 2 valori - false sau true. Tipul bool este important sa-l intelegeti cand se folosesc operatorii logici in instructiunea if.

- int este prescurtarea pentru integer si este un tip de date folosit pentru a pastra numere fara zecimale. Cand se lucreaza cu numere, int este tipul de date folosit de cele mai multe ori. Numerele intregi au asociate cateva tipuri de date in C# in functie de marimea numarului care trebuie stocat,
- string este folosit pentru a păstra text, adica mai multe caractere. In C#, stringurile sunt immutabile, ceea ce inseamna ca stringurile nu sunt niciodata modificate dupa ce au fost create. Cand folosim metode care modifica un string, stringul initial nu este modificat - un nou sir este returnat in locul celui initial.
- char este folosit pentru a pastra un singur caracter.
- float este un tip de date folosit pentru a pastra numere care pot avea 0 sau mai multe zecimale.

Instructiunea IF

Una dintre cele mai importante instructiuni ale fiecarui limbaj de programare este instrucțiunea if. A fi capabil să construiesti secvente de cod conditionale este un principiu fundamental in a scrie cod. În C#, instrucțiunea if este foarte simplu de folosit. Daca ai folosit deja un alt limbaj de programare vei folosi if-ul in C# la fel.

```
namespace ConsoleApplication1
{
   class Program
   {
   static void Main(string[] args)
   {
      int number;

      Console.WriteLine("Please enter a number between 0 and 10:");
      number = int.Parse(Console.ReadLine());

      if(number > 10)
```

```
            Console.WriteLine("Hey! The number should be 10 or less!");
            else
            if(number < 0)
               Console.WriteLine("Hey! The number should be 0 or more!");
            else
               Console.WriteLine("Good job!");

            Console.ReadLine();
        }
        }
    }
```

Operatori

Operatori Un operator reprezinta un simbol care determina compilatorul sa realizeze o anumita operatie matematica sau logica. Limbajul C# ofera patru categorii de operatori: aritmetici, pe biti, relationali si logici. Operatori aritmetici Limbajul C# defineste urmatorii operatori aritmetici: + (adunare), - (scadere), * (inmultire), / (impartire), % (rest sau operatorul modulo), ++ (incrementare), --(decrementare). Atunci cand se aplica operatorul / asupra unor intregi, restul este trunchiat, de exemplu: 14/3 este egal cu 4. O diferenta fata de C si C++ este faptul ca operatorul modulo se poate aplica atat tipurilor intregi cat si celor in virgula mobila (Ex: 14.0 % 3.0=2, 14.2 %3.0=2.2, 14.0%3.1=1.6. Operatorii unari de incrementare si decrementare functioneaza la fel ca in C. Acestia sunt: x++ (forma postfixata), ++x (forma prefixata), x-- (forma postfixata), -- x (forma prefixata).

Operatori relationali si logici Operatorii relationali se refera la relatiile de ordine care pot exista intre doua valori, iar operatorii logici desemneaza modalitatile in care se pot asocia valorile de adevar true si false. Operatorii relationali se utilizeaza deseori cu cei logici. Rezultatele intoarse de operatorii relationali si logici sunt de tip bool. Operatorii relationali sunt urmatorii: == (egal cu), != (diferit de), > (mai mare decat), < (mai mic decat), >= (mai mare sau egal cu), <= (mai mic sau egal cu). Operatorii logici sunt: & (si), | (sau), ^ (sau exclusiv), || (sau scurtcircuitat), && (si scurtcircuitat), ! (non). In cazul operatorilor relationali ,<=,>= operanzii trebuie sa apartina tipurilor pe care este definita o relatie de ordine. In cazul operatorilor logici, operanzii trebuie sa

fie de tip bool. In tabelul de mai jos sunt precizate rezultatele opearatiilor logice. S-a folosit conventia: 0=false si 1=true.

Conventia tipurilor de date

Conversii implicite .Daca in atribuire sunt implicate tipuri compatibile atunci valoarea din partea dreapta este convertita automat la tipul din partea stanga. Atunci cand un tip de date este atribuit unui alt tip de variabila, se efectueaza o conversie implicita (automata) de tip daca: cele doua tipuri sunt compatibile si tipul destinatie este mai cuprinzator decat tipul sursa. Daca cele doua conditii sunt indeplinite atunci are loc o conversie prin largire. (Ex: long in double se face prin conversie automata, in schimb double in long nu se poate realiza automat). De asemenea nu exista conversii automate intre decimal si double sau decimal si float si nici intre tipurile numerice si bool.

Conversii explicite. In multe situatii (daca exista vreo legatura intre tipul sursa si tipul destinatie) se pot realiza conversii explicite de tip, utilizand un cast. Un cast este o directiva catre compilator de a converti un anumit tip in altul. Forma generala este: (tip_tinta) expr; Exemple: double x,y; int z;… z= (int) (x/y); int i=230; byte b; ... b=(byte)i; Insa de fiecare data, responsabilitatea este cea a programatorului daca rezultatul obtinut este cel convenabil. Pentru a controla rezultatul se pot utiliza comenzile: checked (expresie) ; sau unchecked (expresie).

Exemple de cod folosite:

Aceastea sunt variabilele declarate in scriprul atribuit obiectelor pe care le controleaza jucatorul. In ceea ce priveste mecanica jocului, obviectele controlate au un corp rigid care este folosit pentru a detacta coliziuni si pentrru a permite miscarea.Variabilele numerice sunt folosite pentru viteze si intervale de timp. Obiectele GameO sunt obiecte controlate sau care apar si au un anumit scop. Variabilele de tip bool sunt folosite la verificarea unor conditii.

In functia void start am impus gravitatia , am atribuit corpul rigid obiectului controlat de jucator si a adversarului. Am invocate o functie care da o putere speciala care ajuta jucatorul sa lupte impotriva adversarului.

In functia void update an pus conditiiloe pentru miscarea si controlul pe care il are jucatorul asupra obiectelor.

```
public class PlayerController : MonoBehaviour
{
    public float speed;
    private Rigidbody playerRb;
    private GameObject focalPoint;
    public float jumpForce=15f;//for hopping
    public float gravityModifier=5f;//for hopping
    public bool isOnGround = true;
    public bool hasPowerUp = false;
    public bool hasPowerUpEasy = false;
    private float powerUpStrength = 15.0f;
    private float powerUpStrengthD = 90.0f;
    public GameObject powerupIndicator;
    public GameObject powerupIndicatorEasy;
    public GameObject bullet;
    public float bulletSpeed = 7f;
    public GameObject enemy;
    public float c = 1;// we use c to switch between keyboard and mouse by pressing x
    private float startDelay=15.0f;



    public bool hasBullet=false;
    // Start is called before the first frame update
```

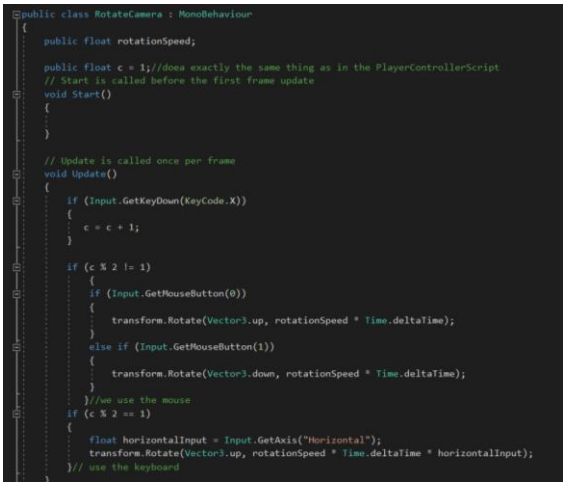
```
void Start()
{

   playerRb = GetComponent<Rigidbody>();
    focalPoint = GameObject.Find("FocalPoint");
    Physics.gravity *= gravityModifier;  //for hopping
    InvokeRepeating("RandomPowerupEasy", startDelay,Random.Range(15,30));
    enemy = GameObject.Find("Enemy");
}

// Update is called once per frame
void Update()
{
    powerupIndicator.transform.position = transform.position + new Vector3(0, 1.2f, 0);
    powerupIndicatorEasy.transform.position = transform.position + new Vector3(0, -0.5f, 0);
    if (Input.GetKeyDown(KeyCode.X))
    {
        c = c + 1;
    }//c will always increase its value by 1 every time we press x;
    if (c % 2 == 1)//if c noes not devide by 2
    {
        float forwardInput = Input.GetAxis("Vertical");
        playerRb.AddForce(focalPoint.transform.forward * speed * forwardInput);
    }//we use the keyboard
    else if (c % 2 != 1)
    {
        float forwardInput = Input.GetAxis("Mouse ScrollWheel");
        playerRb.AddForce(focalPoint.transform.forward * speed * forwardInput);
    }//we use the mouse
    if(Input.GetKeyDown(KeyCode.Space)&&isOnGround)
    {
        playerRb.AddForce(Vector3.up * jumpForce, ForceMode.Impulse);
        isOnGround = false;
    }//press space to jump
```

Am creat functii speciale pentru p[otiunile magice pe care jucatorul le poate colecta pentru a avea o putere mai mare. Pentru a intelege mecanica : cand jucatorul loveste colliderul unei npotiuni, atunci incepe numaratoarea incersa pentru timpul in care poate folosi abilitatea capatata.Pe langa aceste potiuni, jucatorul are avantaje random de-a lungul jocului.Am creat un script pentru inserarea potiunilor in scena jocului.Acestea sunt inserate random (in locuri neasteptate si la intervale de timp mneasteptate pe platforma din joc.

```
private void OnTriggerEnter(Collider other)
{
    if(other.CompareTag("PowerUp"))
    {
        hasPowerUp = true;
        Destroy(other.gameObject);
        StartCoroutine(PowerupCountdownRoutine());
        powerupIndicator.gameObject.SetActive(true);

    }
   if(other.CompareTag("KnockItOff"))
    {
        hasBullet = true;
        Destroy(other.gameObject);
        bullet.gameObject.SetActive(true);

    }



}
IEnumerator PowerupCountdownRoutine()
{
    yield return new WaitForSeconds(7);
    hasPowerUp = false;
    hasPowerUpEasy = false;
    powerupIndicator.gameObject.SetActive(false);
    powerupIndicatorEasy.gameObject.SetActive(false);
}
private void RandomPowerupEasy()//(Collision collision)

{
    if (hasPowerUp==false)
    {
        hasPowerUpEasy = true;
        StartCoroutine(PowerupCountdownRoutine());
        powerupIndicatorEasy.gameObject.SetActive(true);
    }
```

```
public class SpawnManager : MonoBehaviour
{
    public GameObject[] enemyPrefab;
    private float spawnRange = 9.0f;
    public GameObject knockItOff;
    public GameObject player;
    public Vector3 playerPos;

    // Start is called before the first frame update
    void Start()
    {
        //Instantiate(knockItOff, GenerateSpawnPosition(), knockItOff.transform.rotation);
         InvokeRepeating("SpawnKnockItOff", 5, Random.Range(10,15));
       // InvokeRepeating("PowerupCountdownRoutine", 5, Random.Range(10, 15));

        int enemyIndex = Random.Range(0, enemyPrefab.Length);
        Instantiate(enemyPrefab[enemyIndex], GenerateSpawnPosition(), enemyPrefab[enemyIndex].transform.rotation);
    }


    // Update is called once per frame
    void Update()
    {

    }
    private Vector3 GenerateSpawnPosition ()
    {
    float spawnPosX = Random.Range(-spawnRange, spawnRange);
    float spawnPosZ = Random.Range(-spawnRange, spawnRange);
    Vector3 randomPos = new Vector3(spawnPosX, 0, spawnPosZ);

        return randomPos;
    }
    void SpawnKnockItOff()
    {
```

Am mentionat anterior ca anumite obiecte sunt sontrolate de jucator. In afara de obiectul principal (bila jucatorului), jucatorul mai poate controla si insula. Acesta spec teste inpropriu deoarece ceea ce jucatorul roteste de fapt nu este platforma ci camera. Practic, viziunea se roteste spre stanga si spre cu ajutorul sagetilor. Acest script face posibila aceasta situatie. Pe langa viteza de rotatie trebuie atribuit si timpul.



```
public class RotateCamera : MonoBehaviour
{
    public float rotationSpeed;

    public float c = 1;//doea exactly the same thing as in the PlayerControllerScript
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.X))
        {
            c = c + 1;
        }

        if (c % 2 != 1)
            {
            if (Input.GetMouseButton(0))
            {
                transform.Rotate(Vector3.up, rotationSpeed * Time.deltaTime);
            }
            else if (Input.GetMouseButton(1))
            {
                transform.Rotate(Vector3.down, rotationSpeed * Time.deltaTime);
            }
           }//we use the mouse
        if (c % 2 == 1)
        {
            float horizontalInput = Input.GetAxis("Horizontal");
            transform.Rotate(Vector3.up, rotationSpeed * Time.deltaTime * horizontalInput);
        }// use the keyboard
    }
```

Acest script contine programarea adversarului si este un exemplu de inteligenta artificiala.Adversarul are o alta greutate si putere atribuita pentru a constitui o provocare. Existenta

unor pereti invizibili de a lungul marginilor platformei este esentiala deoarece in coliziune, adversarul se mista foarte incet in cadrul acostor prereti si deci este nevoie de o forta mai mare pentru a-l face sa cada de pe platforma.ln alte cuvintr, “adversarul stie cand se apropie de marginile platformei si incetineste”.Mai mult, adversarul este programat sa se deplaseze in directia jucatorului pentru a-l dobori. Asemanator acestui cod , am creat un altul care este atribuit unui tip de adversar mai greu de doborat. La inceperea jocului, adversarul este ales aleatoriu dintre cele 2 posibilitati.

```
public class SpawnManager : MonoBehaviour
{
    public GameObject[] enemyPrefab;
    private float spawnRange = 9.0f;
    public GameObject knockItOff;
    public GameObject player;
    public Vector3 playerPos;

    // Start is called before the first frame update
    void Start()
    {
        //Instantiate(knockItOff, GenerateSpawnPosition(), knockItOff.transform.rotation);
         InvokeRepeating("SpawnKnockItOff", 5, Random.Range(10,15));
       // InvokeRepeating("PowerupCountdownRoutine", 5, Random.Range(10, 15));

        int enemyIndex = Random.Range(0, enemyPrefab.Length);
        Instantiate(enemyPrefab[enemyIndex], GenerateSpawnPosition(), enemyPrefab[enemyIndex].transform.rotation);
    }


    // Update is called once per frame
    void Update()
    {

    }
    private Vector3 GenerateSpawnPosition ()
    {
    float spawnPosX = Random.Range(-spawnRange, spawnRange);
    float spawnPosZ = Random.Range(-spawnRange, spawnRange);
    Vector3 randomPos = new Vector3(spawnPosX, 0, spawnPosZ);

        return randomPos;
    }
    void SpawnKnockItOff()
    {
```

Pentru tranzitia scenelor am programat butoanele.

public class GameManager : MonoBehaviour

{

```
 bool gameHasEnded = false;
 public float restartDelay = 1f;
public void EndGame ()
 {
    if (gameHasEnded == false)
    {
        gameHasEnded = true;
        Debug.Log("GAME OVER");

         Invoke("LoadNExtScene", restartDelay); }
 //  void Restart ()
 //   {
 //   SceneManager.LoadScene(SceneManager.GetActiveSc
 //}
 public void LoadNExtScene()
 {
    SceneManager.LoadScene(2);
}
```

Nota:

Acesta este scriptul pe care l – am folosit pentru a incheia jocul cand bila jucatorului cade de pe platforma. Aceasta secventa cheama un alt script care contine codul pentru incarcarea urmatoarei

Bibliografie:

Unity Manual

Traducator

Interfata Unity

Ghid