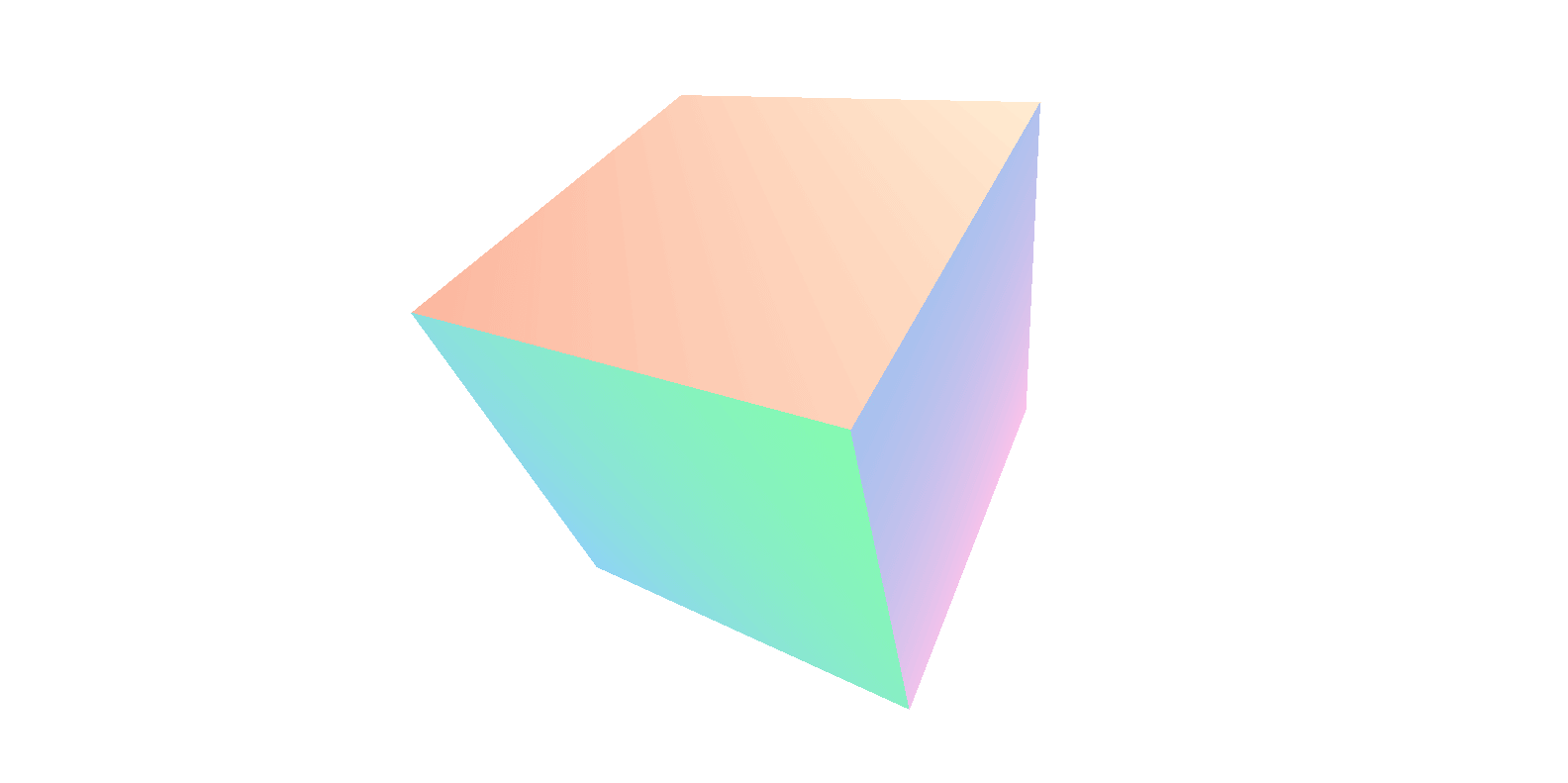
**COLEGIUL NAŢIONAL “IOSIF VULCAN” ORADEA**

**Profil matematică – informatică, intensiv informatică**

**LUCRARE DE ATESTAT**

**CubeLab**

****

**PROF. COORDONATOR:**

**ADRIANA SAVU**

**ELEV:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**CLS. a XII-a. E**

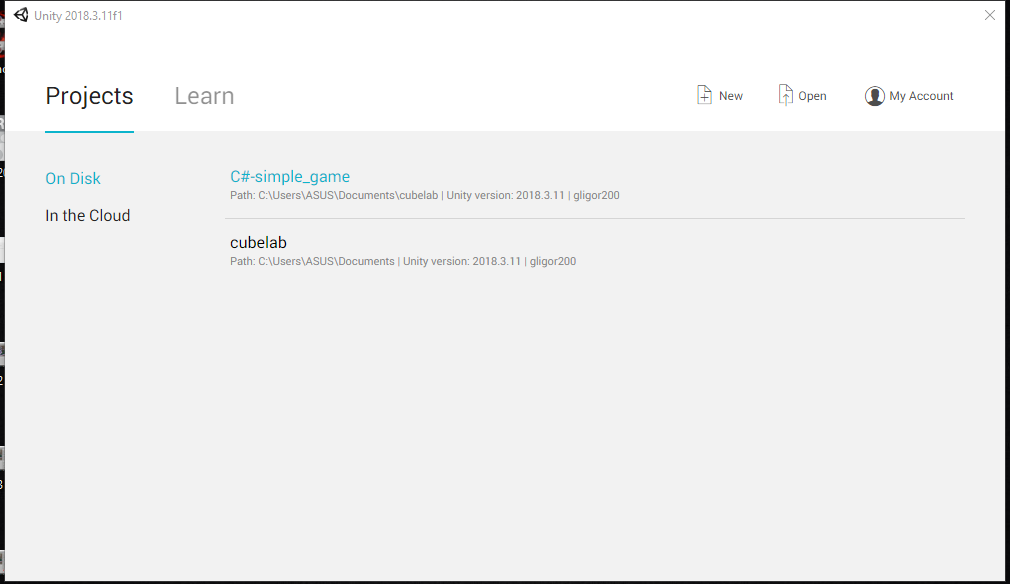
**ORADEA 2017**

Motivatie

Acest proiect a fost inspirat de pasiunea amândurora pentru jocurile video și modul în care sunt acestea făcute,dar motivul principal pentru care am ales să creăm un joc a fost pentru a învață și alte limbaje de programare și deoarece considerăm că programarea vizuală este un domeniu în continuă expansiune și folositor în diferite domenii.

Proiectul a fost o provocare în special din cauza faptului că am început totul de la zero și nu toate ideile au fost implementate

în joc

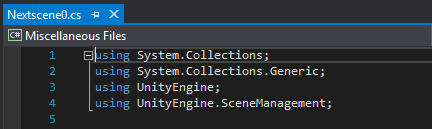


Acesta este de fapt proiectul cu numărul 4.Primul proiect a fost început tot în Unity dar din cauza lipsei de idei am renunțat la el.Al doilea a fost început în Game Maker și trebuia să fie un joc 2D însă a trebuit să renunțăm și la acea idee.Al treilea a fost început în Unreal Engine 4 însă acolo am întâmpinat dificultăți tehnice.Cu toate că nu am reușit din prima încercare am învățat din eșecurile noastre și am devenit din ce în ce mai buni la dezvoltarea acestui joc ,făcut în cele din urmă în Unity și astfel am experimentat și alte Engine-uri pentru jocuri ceea ce ne-a dat o nou perspectiva în ceea ce privește programarea vizuală.

Tema jocului nostru “CubeLab” este distracția cu forme simple.Am vrut să arătăm că jocul poate fi distractiv și provocator în același timp chiar dacă arată relativ simplu.

Am făcut designul fiecărui nivel astfel încât să aducă ceva

nou de fiecare dată.Am experimentat mult cu diferite funcții pentru a determina care este cea mai potrivită și astfel am observat diferențele și asemănările dintre c-sharp și c++. De exemplu structurile sunt în mare parte aceleași , cea mai folosită de noi fiind ‘if’.Însă c-sharp se bazează mult mai mult pe funcții predefinite având o multitudine de librării.



**C#**este un limbaj de programare orientat pe obiecte, simplu, modern, ce derivă din C++ și Java.

**Numele** limbajului C# a fost inspirat din notaţia ♯ (diez) din muzică, care indică faptul că nota muzicală urmată de ♯ este mai înaltă cu un semiton. Este o similitudine cu numele limbajului C++, unde ++ reprezintă atât incrementarea unei variabile cu valoarea 1, dar şi faptul că C++ este mai mult decât limbajul C.

**Limbajul** C# a fost dezvoltat în cadrul Microsoft. Principalii creatori ai limbajului sunt Anders Hejlsberg, Scott Wiltamuth şi Peter Golde. Prima implementare C# larg distribuită a fost lansată de către Microsoft că parte a iniţiativei .NET în iulie 2000. Din acel moment, se poate vorbi despre o evoluţie spectaculoasă. Mii de programatori de C, C++ şi Java, au migrat cu uşurinţă spre C#, graţie asemănării acestor limbaje, dar mai ales calităţilor noului limbaj. La acest moment, C# şi-a câştigat şi atrage în continuare numeroşi adepţi, devenind unul dintre cele mai utilizate limbaje din lume.

**Simplitatea** limbajului este dată de absența pointerilor, moștenirea facilităților de „garbage collection” și managementul automat al memoriei datorită integrării pe platforma .NET, iar operațiile nesigure precum manipularea directă a memoriei nu sunt premise.

**Creatorii C#** au intenţionat să înzestreze limbajul cu mai multe facilităţi. Succesul de care se bucură în prezent, confirmă calităţile sale:

        Este un limbaj de programare simplu, modern, de utilitate generală, cu productivitate mare în programare.

        Este un limbaj orientat pe obiecte.

        Permite dezvoltarea de aplicaţii industriale robuste, durabile.

        Oferă suport complet pentru dezvoltarea de componente software, foarte necesare de pildă în medii distribuite. De altfel, se poate caracteriza C# ca fiind nu numai orientat obiect, ci şi orientat spre componente.

 **La aceste caracteristici generale** se adaugă şi alte trăsături, cum este de pildă suportul pentru internaţionalizare, adică posibilitatea de a scrie aplicaţii care pot fi adaptate cu uşurinţă pentru a fi utilizate în diferite regiuni ale lumii unde se vorbesc limbi diferite, fără să fie necesare pentru această schimbări în arhitectură software.

**Modernitatea limbajului**: C# este conceput pentru construirea de operații scalabile, robuste și interoperabile. Include un suport “built în” pentru transformarea componentelor în serviciu web pentru un acces mai ușor.

**Orientat pe obiecte**: suportă încapsularea datelor. Interfețele, moștenirea și polimorfismul. A introdus structurile (structs) pentru a permite tipurilor primitive să devină obiecte.

**Type safe**: nu se execută cast-uri nesigure, tablourile sunt neindexate și verificate, depășirea tipurilor este verificabilă, tipurile primitivelor sunt initializate cu zero și tipurile de referință cu null în mod automat de către compilator.

**Interoperabilitate**: C# include suport pentru COM, permite utilizarea cu restricții a pointerilor, componente din VB.net și din alte limbaje pot fi folosite direct în C#.

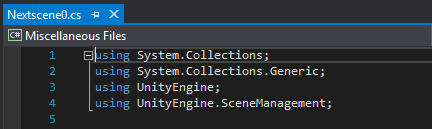
**Portabilitate**: Codul C# poate rula pe mai mult de 2,2 miliarde de dispozitive cu Windows, iOS, Android, Linux. C# este portabil într-o gama largă de medii cum ar fi telefonia mobilă, Embedded, desktop și servere de calcul.

2) Avantaje

**Avantajele față de C++:**

Sistemul garbage-collection e nativ

Biblioteca standard foarte bogată cu lucruri bine implementate și ușor de folosit



·       Clasele și metodele pot fi setate să fie interne în ansamblul în care sunt declarate

·       Permite tratarea semnăturilor de clasa-metode că funcții libere( ignoră argumentul pointer this) și creează astfel relații mai dinamice și flexibile între clase

·       Permite folosirea blocurilor de cod atât native, cât și gestionate

**Avantajele față de Java:**

·       În loc de implementarile de clase și EJB, C# are constructori proprii ușor de folosit precum Properties sau Events

·       Suportă expresia ‘using’ pe când Java o va implementa în Java 7

 ·       Este integrat în Windows

·       Are Lambda și LINQ suportând o parte de programare funcțională

·       Are variabile dinamice

Permite definirea de tipuri noi

**Programarea** vizuală trebuie privită ca un mod de proiectare a unui program prin operare directă asupra unui set de elemente grafice (de aici vine denumirea de programare vizuală). Această operare are ca efect scrierea automată a unor secvenţe de program, secvenţe care, împreună cu secvenţele scrise textual vor forma programul.

**Spunem** că o aplicaţie este vizuală dacă dispune de o interfaţă grafică sugestivă şi pune la dispoziţia utilizatorului instrumente specifice de utilizare (drag, clic, hint etc.)

**Realizarea** unei aplicaţii vizuale nu constă doar în

desenare şi aranjare de controale, ci presupune în principal stabilirea unor decizii arhitecturale, decizii ce au la bază unul dintre modelele arhitecturale de bază.

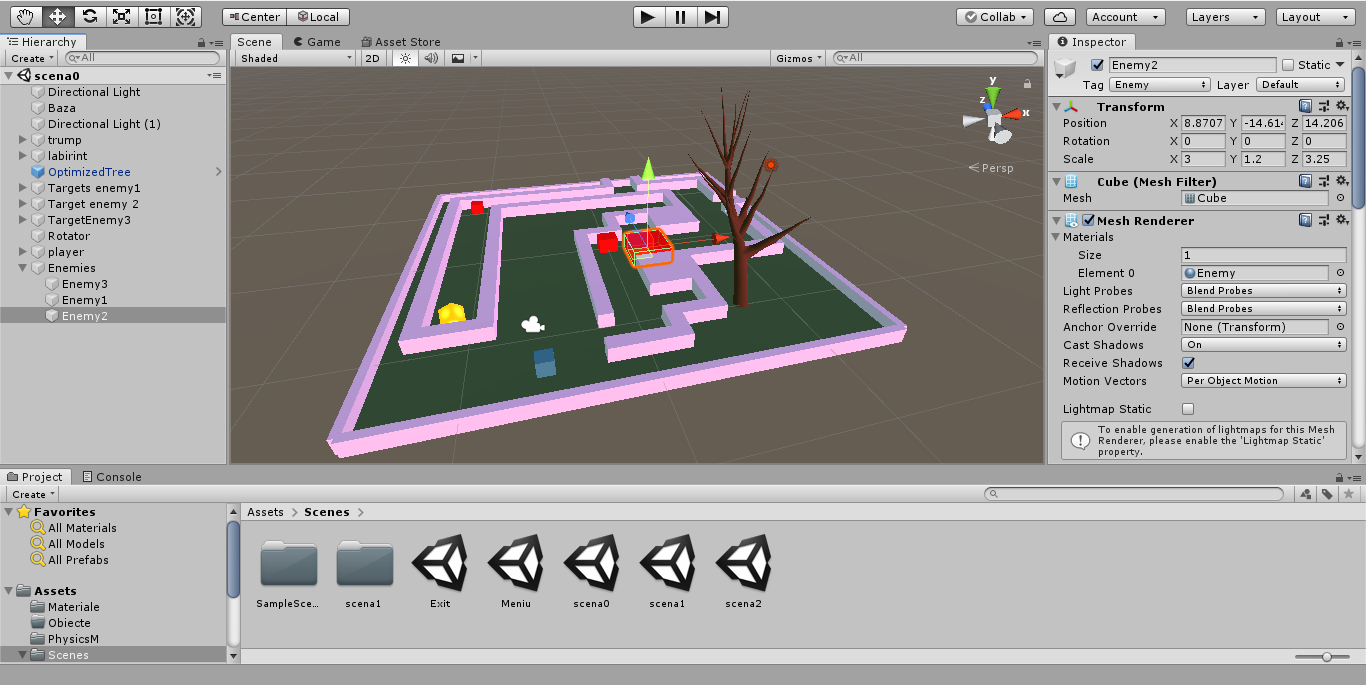
**În realizarea aplicaţiei** mai trebuie respectate şi principiile proiectării interfeţelor:

 Simplitatea: Interfaţa trebuie să fie cât mai uşor de înţeles şi de învăţat de către utilizator şi să permită acestuia să efectueze operaţiile dorite în timp cât mai scurt. În acest sens, este vitală culegerea de informaţii despre utilizatorii finali ai aplicaţiei şi a modului în care aceştia sunt obişnuiţi să lucreze.

  Poziţia controalelor: Locaţia controalelor dintr-o fereastră trebuie să reflecte importanţa relativă şi frecvenţa de utilizare. Astfel, când un utilizator trebuie să introducă nişte informaţii – unele obligatorii şi altele opţionale – este indicat să organizăm controalele astfel încât primele să fie cele care preiau informaţii obligatorii.

  Consistenţa: Ferestrele şi controalele trebuie să fie afişate după un design asemănător („template”) pe parcursul utilizării aplicaţiei. Înainte de a implementa interfaţa, trebuie decidem cum va arăta aceasta, să definim „template”-ul.

  Estetica: Interfaţa trebuie să fie pe cât posibil plăcută şi atrăgătoare.



**Unity** este un motor în timp real dezvoltat de Unity Technologies , anunțat și lansat pentru prima dată în iunie 2005 la Conferința Mondială a dezvoltatorilor de la Apple Inc., ca motor exclusiv pentru jocul OS X. Începând cu anul 2018, motorul a fost extins pentru a susține 27 de platforme.



**Motorul** poate fi folosit pentru a crea atât jocuri tridimensionale cât și bidimensionale, precum și simulări pentru numeroasele sale platforme.

**Unity** oferă utilizatorilor posibilitatea de a crea jocuri și experiențe interactive atât în versiunile 2D cât și în 3D, iar motorul oferă un API primar de scripting în C#, atât pentru editorul Unity sub formă de pluginuri, cât și pentru jocurile în sine, precum și funcționalitatea drag and drop . Înainte ca C# să fie limba de programare primară utilizată pentru motor, a susținut anterior Boo, care a fost eliminat în versiunea Unity 5, și o versiune de JavaScript numită UnityScript, care a fost depreciată în august 2017 după lansarea Unity 2017.1 în favoarea C #.

**Motorul** are suport pentru următoarele API-uri: Direct3D pe Windows și Xbox One; OpenGL pe Linux, MacOS și Windows; OpenGL ES pe Android și iOS; WebGL pe web; și API-urile proprietare pe consolele de jocuri video. În plus, Unity acceptă API-urile de nivel inferior Metal pe iOS și macOS și Vulkan pe Android, Linux și Windows, precum și Direct3D 12 pe Windows și

Xbox One.

**În cadrul jocurilor 2D**, Unity permite importul de sprites și un avansat renderer 2D din lume.

**Pentru jocuri și simulări 3D**, Unity permite specificarea compresiei texturilor, mipmaps și setările de rezoluție pentru fiecare platformă pe care suportă motorul jocului  și oferă suport pentru cartografierea bruiei, cartografierea reflecțiilor, maparea paralaxelor, ocluzia ambientală a spațiului ecranului (SSAO) , umbre dinamice care utilizează hărți umbrite, efecte       rendering-to-texture și efecte post-procesare pe tot ecranul.

**Începând cu anul 2016**, Unity oferă și servicii de tip cloud pentru dezvoltatori, acestea fiind în prezent: Unity Ads, Unity Analytics, Unity Certification, Unity Cloud Build, Unity Everyplay, Unity IAP (" , Unitate multiplayer, unitate de performanță de raportare, Unity Collaborate și Unity Hub.

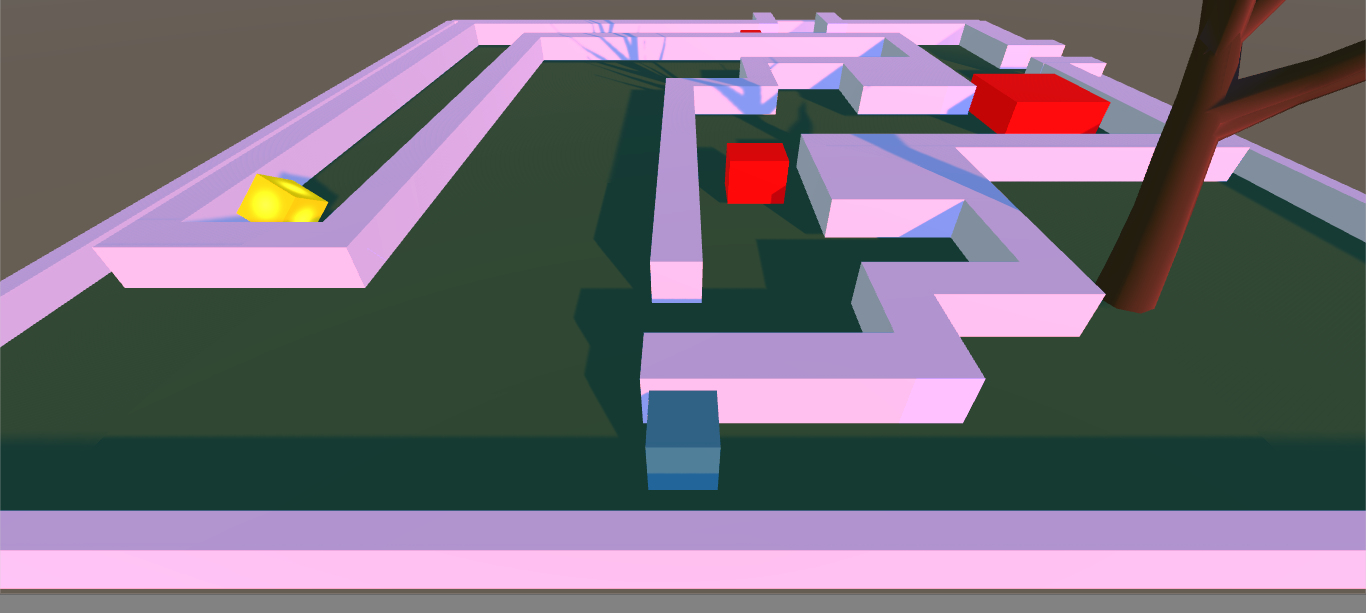
**Unity** acceptă crearea unui vârf personalizat, fragment (sau pixel), tessellation și shaders de calcul. Shaderele pot fi scrise utilizând Cg sau HLSL de la Microsoft. 

**Prezentarea generala**



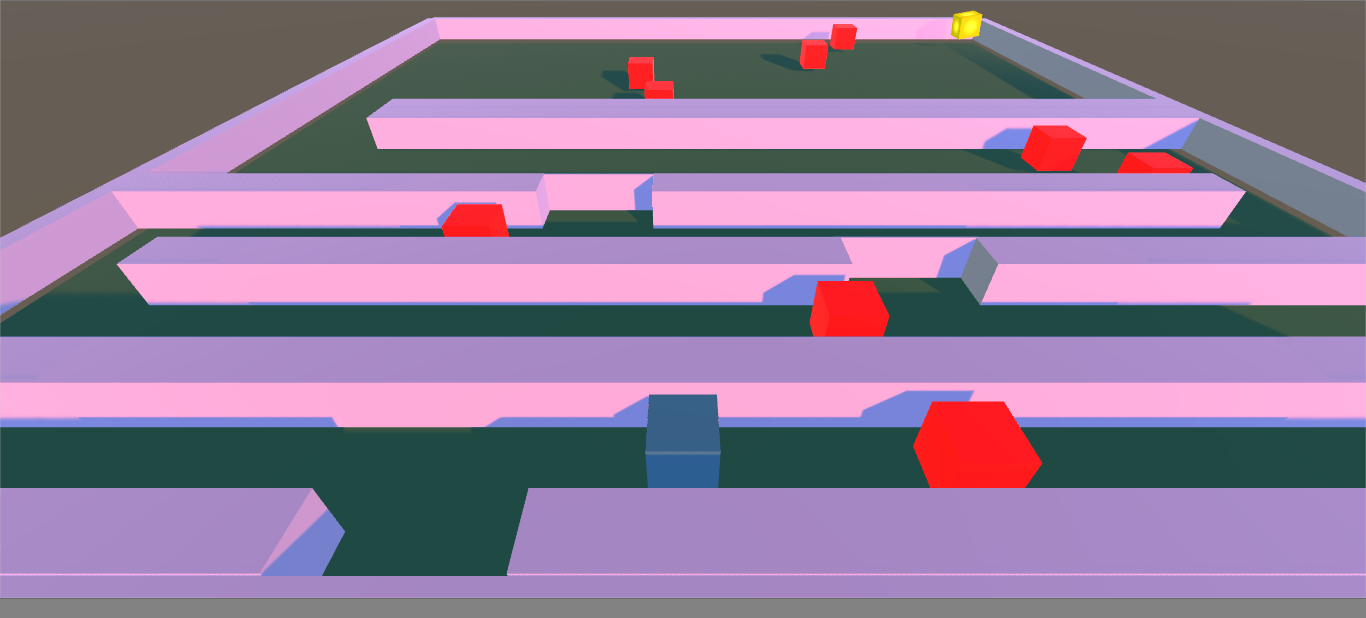
Jocul constă în trecerea în siguranță a cubului prin labirint cu scopul de ajunge la portalul de la finalul acestuia și în final acasă

Însă drumul cubului este plin de inamici care vor să îl oprească.



Fiecare inamic are un traseu pe care îl patrulează .De aceea trebuie să fii foarte atent la locurile în care acesta nu poate să ajungă pentru a știi unde să te adăpostești .

Dacă nu sunt locuri în care te poți adapostii înseamnă că poți trece într-un fel sau altul pe lângă el .

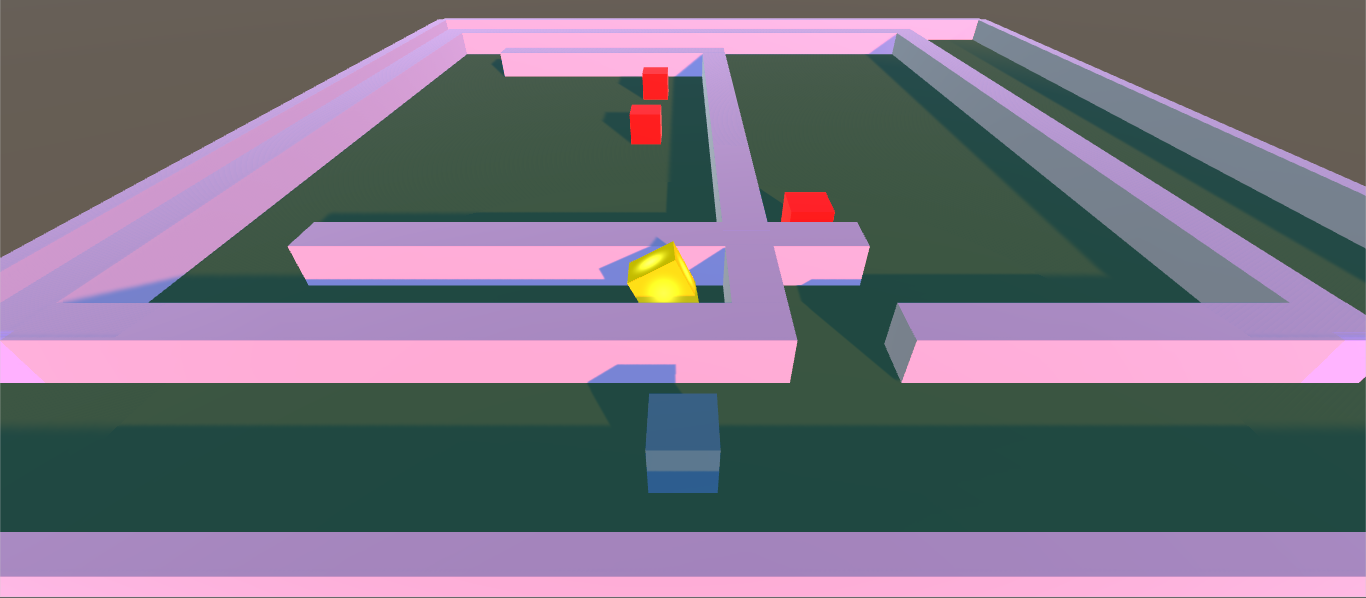


Din cauza designului compact al nivelelor poți să îți planifici de la început traseul pe care îl vei urmă,astfel jocul include și elemente de planificare și sincronizare.

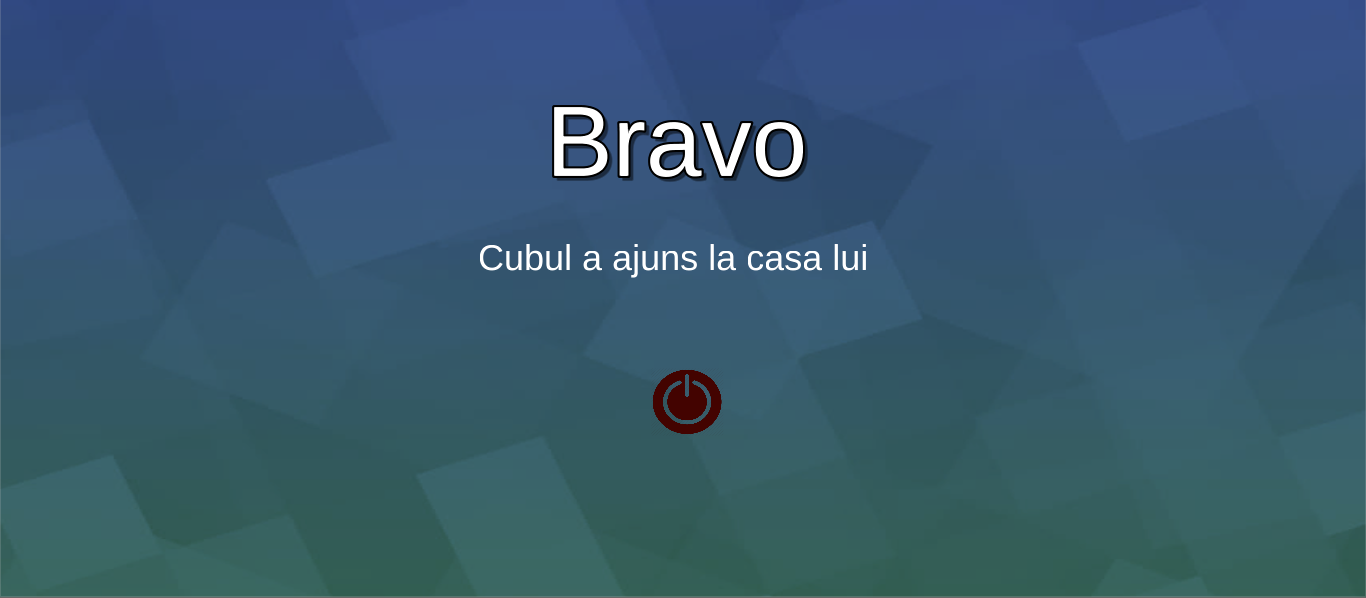
Pentru a fi mai interactive și interesant inamicii au viteze și dimensiuni diferite în funcție de traseul urmat.

De asemenea nu totul e ceea ce pare deoarece unii inamici încep că obiecte statice dar când te aștepți mai puțîn încep să te urmărească.

Și pasajele secrete apar în formulă acestui joc deci trebuie să te aștepți la orice.



Chiar dacă de multe ori designul pare simplu va putem asigura că provocarea este prezența.

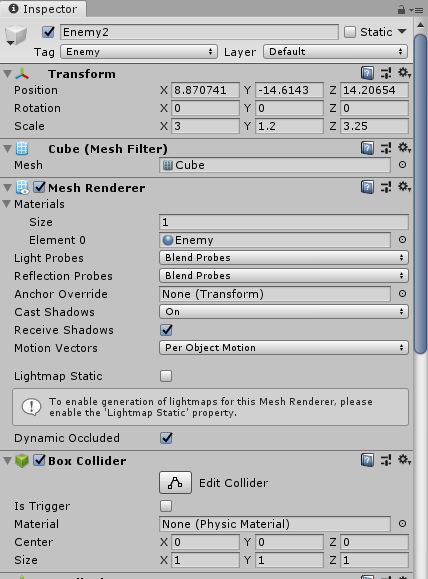


**Functiile folosite**



**În Unity fiecare platforma , fiecare inamic,fiecare “copac” ,pe scurt fiecare corp existent are proprietăți și coordinate diferite în funcție de cum le-am creeat și la ce le-am folosit.**

**Fiecare corp are un set de coordonate pentru poziție,rotație și mărime,un mesh renderer pentru render,un box collider pentru detectarea coliziunii.**



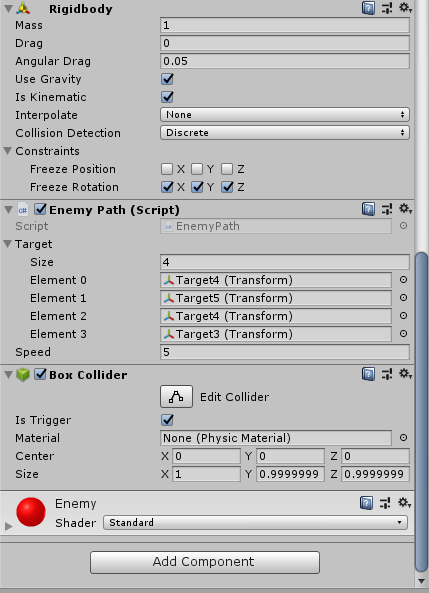
**De asemenea avem un rigidbody pentru a putea aplică forțe asupra obiectului respectiv de regulă pentru a implementa mișcare.**

**Un alt element prezent la fiecare corp**

**este materialul care definește culoarea ,reflexia , netezimea și alte proprietat.**

**În funcție de necesitate corpurile pot**

**avea și coduri atașate de ele .**



Codul pentru mișcarea cubului

|  |
| --- |
| using System.Collections;  using System.Collections.Generic;  using UnityEngine;  using UnityEngine.SceneManagement;  public class KeyCode : MonoBehaviour  {  private Rigidbody rb;  // Start is called before the first frame update  void Start()  {  rb = GetComponent<Rigidbody>();  }  // Update is called once per frame  void FixedUpdate()  {  if(Input.GetKey("right"))  {  transform.Translate(0.1f, 0, 0);  }  if (Input.GetKey("left"))  {  transform.Translate(-0.1f, 0, 0);  }  if (Input.GetKey("up"))  {  transform.Translate(0, 0, 0.1f);  }  if (Input.GetKey("down"))  {  transform.Translate(0, 0, -0.1f);  }    }  } |

La început am definit un rigidbody pentru a putea aplică forțe asupra cubului nostru apoi am definit pe ce axe și în ce direcții sunt aplicate acele forțe în funcție de butonul apăsat.

EnemyPath

|  |
| --- |
| using System.Collections;  using System.Collections.Generic;  using UnityEngine;  public class EnemyPath : MonoBehaviour  {  public Transform[] target;  public float speed;  private int current;    void FixedUpdate()  {  if (transform.position != target[current].position)  {  Vector3 pos = Vector3.MoveTowards(transform.position, target[current].position, speed \*Time.deltaTime);  GetComponent<Rigidbody>().MovePosition(pos);  }  else current = (current + 1) % target.Length;  }  } |

La început am definit un rigidbody pentru a putea aplică forțe asupra cubului nostru apoi am definit pe ce axe și în ce direcții sunt aplicate acele forțe în funcție de butonul apăsat.

Rotator

|  |
| --- |
| using System.Collections;  using System.Collections.Generic;  using UnityEngine;  public class Rotator : MonoBehaviour  {  // Start is called before the first frame update    // Update is called once per frame  void Update()  {  transform.Rotate(new Vector3(15, 30, 45) \* Time.deltaTime);  }  } |

Acesta este codul pentru mișcarea de rotație a portalului de la finalul labirintului.

FollowScript

|  |
| --- |
| using System.Collections;  using System.Collections.Generic;  using UnityEngine;  public class FollowScript : MonoBehaviour  {  bool follow;  public GameObject enemy1;  public GameObject enemy2;  public GameObject player;  public float speed;  void Start()  {  follow = false;  }  void Update()  {  if (follow)  {  enemy1.transform.position = Vector3.MoveTowards(enemy1.transform.position, player.transform.position, Time.deltaTime \* speed);  if(enemy2!=null)  enemy2.transform.position = Vector3.MoveTowards(enemy2.transform.position, player.transform.position, Time.deltaTime \* speed);  }    }  void OnTriggerEnter(Collider collider)  {  follow = true;  }  } |

Acest script este elementul surpriză al nivelului 3 deoarece elimina necesitatea unui target fiindcă prin intermediul acestuia inamicii vor urmării jucătorul.

ReloadPosition

|  |
| --- |
| using System.Collections;  using System.Collections.Generic;  using UnityEngine;  public class RELOAD : MonoBehaviour  {  private Vector3 \_initialPosition;  void Start()  {  \_initialPosition = transform.position;  }  void OnTriggerEnter(Collider col)  {  if (col.gameObject.tag == "Enemy")  transform.position = \_initialPosition;  }  } |

ReloadScene

|  |
| --- |
| using System.Collections;  using System.Collections.Generic;  using UnityEngine;  using UnityEngine.SceneManagement;  public class Reload2 : MonoBehaviour  {  void OnTriggerEnter(Collider col)  {  if (col.gameObject.tag == "Enemy")  SceneManager.LoadScene("scena");  }  } |

Aceste coduri fac cubul să se întoarcă la poziția inițială în momentul contactului cu un inamic.

NextScene

|  |
| --- |
| using System.Collections;  using System.Collections.Generic;  using UnityEngine;  using UnityEngine.SceneManagement;  public class Nextscene0 : MonoBehaviour  {  // Start is called before the first frame update  void Start()  {    }  // Update is called once per frame  public void LoadScene()  {  SceneManager.LoadScene("scena0"/\*, LoadSceneMode.Additive\*/);  }  } |

Codul prin care se trece la următorul nivel.

Quit

|  |
| --- |
| using System.Collections;  using System.Collections.Generic;  using UnityEngine;  public class Quit : MonoBehaviour  {  public void doquit()  {    Application.Quit();  Debug.Log("Quit");  }  } |

Codul pentru ieșirea din joc.

Bibliografie:

<http://www.ls-infomat.ro/user/content/e9efcsharp.pdf>

<http://stst.elia.pub.ro/news/SO/Teme_SO_2013/432_Florea%20Carmen_Csharp_2.doc>

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