Ministerul Educaţiei al Republicii Moldova
Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică şi Microelectronică
Departamentul ISA

Proiect de an

Disciplina : Analiza Proiectarea și Programarea Orientată pe Obiecte
Tema: Elaborarea unui joc 3D in Unity3D

A efectuat: st. gr. TI-142 Comanda Artur

A verificat: lector asistent Gavrisco Alexandru

Chișinău 2017

Cuprins:

Introducere

Unity3D este vorba despre o tehnologie care este atât utilizabilă în mod unic, cât și extrem de puternică. Concentrându-se pe o interacțiune clară între caracteristici și funcționalitate - din perspectiva dvs. Ca o companie, suntem cu toții să construim acea tehnologie, să o conducem înainte și să o susținem.

Unitatea vizează, de asemenea, reunirea unora dintre cei mai buni oameni din lume și lăsându-i să se despartă de rezolvarea unor probleme interesante și grele pentru clienții noștri.

Investigăm agresiv în dezvoltare pentru a menține Unitatea în mișcare și creștere într-un ritm radical, extinderea capacității de utilizare, a puterii și a platformei.

Facem munca grea, astfel încât să nu fie nevoie și să faceți durerea, astfel încât să nu fie nevoie.

Această practică este deopotrivă o sursă de calm şi relaxare, dar şi o sursă de stimulare a creierului şi a corpului. Jocul este descris adeseori ca modalitatea de petrecere a timpului cea mai vioaie, deşi de multe ori îl luăm în serios şi putem uita în totalitate de el.

Un studiu similar, desfasurat la York University, a descoperit că jocurile pe calculator pregătesc creierul pentru activități diverse ce presupun o coordonare mîini-ochi (ochii văd, mîinile execută).

Un studiu a demonstrat la dislexici îmbunătățirea înțelegerii de lectură în urma sesiunilor de jocuri grele pe acțiune. Motivul, cercetătorii cred, este faptul că jocurile au schimbat în mod constant mediile care necesita concentrare intensă.

În ziua de azi crearea jocurilor 2D-3D este facilitată datorită limbajelor de programare superioare(C++,C#, Java) și motoarelor grafice (Unity3D,Unreal Engine) care ușurează calea spre dezvoltarea acestora.

Un motor grafic este un sistem conceput pentru crearea și dezvoltarea de jocuri video. Există mai multe motoare de joc, care sunt proiectate să funcționeze pe console de jocuri video și calculatoare personale. Funcționalitatea de bază oferită de obicei de un motor grafic include un motor de randare (engleză renderer) pentru grafică 2D sau 3D, un motor de fizică sau de detectare a coliziunilor (și răspunsul la coliziune), sunet, scripting, animație, inteligență artificială, în rețea, streaming, memorie de

management, suport de localizare. Dezvoltatorul este predispus să aleagă motorul care îi convine pentru a dezvolta jocuri de înaltă calitate.

Scopul aplicației create este deopotrivă o sursă de calm și relaxare în timpul liber, dar în același timp este o sursă de stimulare a atenției, reacției.

Jocul a fost creat cu ajutorul motorului grafic Unity3D, și limbajului de programare orientat pe obiecte C#.

1 De ce am folosit Unity3D ?

Pentru dezvoltarea jocurilor și a aplicațiilor, Studio Pepwuper îi place să folosească Unity 3D ca software principal. Dacă v-ați gândit să intrați în dezvoltarea de jocuri sau aplicații sau dacă sunteți un client care caută dezvoltarea de aplicații și vă întrebați ce software pentru care a fost construită aplicația / jocul, iată primele 9

Figura 1 – Logo lui Unity 3D

motive pentru care alegem Unity:

1. Este GRATUIT să începeți cu unitatea

Unity3D vine cu o versiune gratuită și o versiune Pro, dar spre deosebire de majoritatea software-urilor cu ambele opțiuni de plată, versiunea gratuită a Unity3D este complet completă. Există avantaje clare pentru plata pentru versiunea Pro odată ce ați avansat cu programul (de exemplu: filtru audio, redare video și streaming, suport textură 3D, ecran de pornire personalizat și multe altele), dar între timp Unity permite jocurilor de noroc Jocuri fără obstacole de preț.

2. Este Multi Platform

IOS, Android, telefoane Windows, Mac, PC, Steam, Playstation, Xbox, Wii U ... etc. Există multe platforme pe care le poți publica jocul tău, iar Unity îți ușurează jocul de la o platformă la alta. Portarea unui joc într-o altă platformă care utilizează un set diferit de tehnologii folosite pentru a implica un efort masiv - de multe ori era externalizată unei alte companii și a durat luni de timp de dezvoltare. Cu Unitatea, portarea la o nouă platformă este mult mai simplă. Totuși, doriți să luați în considerare caracteristicile unice ale platformei atunci când construiți un joc pentru aceasta, dar Unity o face mult mai ușor să porți.

3. Comunitatea înfloritoare și susținătoare

Dezvoltarea jocului Indie poate deveni uneori singuratic, dar cu 2 milioane + dezvoltatori care folosesc software-ul Unity (un număr care crește în fiecare zi), este minunat să ai mai multe resurse online pentru a împărtăși dragostea și frustrarea programului. Dacă vreodată v-ați blocat într-o problemă în curs de dezvoltare, doriți să discutați cu persoane asemănătoare sau chiar căutați un artist sau un dezvoltator care să colaboreze cu următoarea idee mare, există tone de forumuri acolo unde fanii dornici de Unity se unesc. Și vorbind despre Unite, există conferința anuală pe care Unity o pune pe (Unite), unde vă puteți întâlni prietenii online Unity în persoană, fie în Europa, fie în America de Nord în fiecare vară.

În plus, există mai multe întâlniri de unitate din întreaga lume care nu sunt afiliate la unitate, dar sunt recunoscute și susținute de acestea. Dacă vă aflați în Seattle, veniți și alăturați-vă la Seattle Unity Meetup, pe care o organizează lunar pentru o prezentare și o rețea. La momentul scrisului, avem peste 600 de membri iubitori de 3D Unity!

4. Magazinul de bunuri

Unitatea Asset Store este un loc minunat pentru a) găsi ceea ce aveți nevoie pentru jocul dvs. fără să îl faceți de la zero (un personaj, o clădire etc) sau b) un loc frumos de a face un mic venit suplimentar dacă sunteți un artist, Muzician sau modeler.

Există un proces de depunere prin care trebuie să treceți pentru a vă vinde activele în magazinul Unity, dar odată ce sunteți aprobat, veți primi 70% redevențe pentru fiecare achiziție, ceea ce poate fi o modalitate fantastică de a vă finanța următorul joc!

5. Limbi scrise

Aveți posibilitatea să script-ul în Unity folosind Javascript sau C #, două dintre cele mai populare limbi și ambele sunt foarte ușor de a începe cu.

6. Capacitatea de a crea jocuri 2D

Deși Unity este excelentă pentru animația 3D, există, desigur, încă un loc pentru dezvoltarea 2D. Cu cea mai recentă versiune Unity 4.3, există un motor 2D încorporat care vă permite să creați jocuri 2D. Se ocupă de animația sprite, de fizica 2D, foaia de animație de animație ... etc. Și multe alte bunătăți.

7. Abilitatea de a crea jocuri multiplayer

Unele dintre cele mai mari jocuri multiplayer de pe web și mobile sunt construite cu Unity (Marvel Superhero Squad, Solstice Arena). Construirea unui joc multiplayer este o masivă acțiune, iar cu uneltele oferite de Unity și cu sprijinul comunității, suntem capabili să creăm jocul nostru multiplayer, Giants Me, așa cum ne-am dorit - o sarcină care ar fi fost imposibilă fără!

8. Tutoriale Online / Clase facilitează învățarea

Lucrul foarte frumos despre Unitate este cât de ușor este să înveți. Sigur, există o curbă de învățare la început, dar având în vedere ce puteți face cu software-ul, este incredibil de ușor. Cu câteva cursuri online și tutoriale care învață elementele de bază ale unității disponibile, puteți învăța cum să începeți cu aceasta pentru un cost foarte scăzut - și din confortul propriei case.

Predau un curs online de 4 săptămâni prin intermediul site-ului de învățământ Skillshare care te învață să faci primul tău joc de la zero. Nu este absolut nici o cunoaștere prealabilă a dezvoltării necesare și nici o codificare.

9. Unite - Conferința Unității

Unite a fost un eveniment anual în care adoptivi timpurii ai Unității s-au adunat și au vorbit cu tipii de la Unitate despre toate subțire

Figura 1.1 - Ponderea pe piața mondială a motoarelor de crearea jocurilor

2 Modelarea funcțională a sistemului

Scopul realizării acestei lucrări este de a implementa unui demo joc 3D din categoria Arcade unde va fi prezent un character principal ca o red panda („Pucin”) care va trebui să parcurgă din punctul initial în punctul final evitând un șir de obstacole totodată colectând elemente colectabile care oferă personajului diferite caracteristice. Jocul va dispune din mai multe nivele , fiecare nivel se deosebește unul de celălalt cu diferite questuri interesante

2.1 Modelarea comportamentului sistemului

Diagramele cazurilor de utilizare (use case diagram) descriu destinația funcțională a sistemului sau cu alte cuvinte descrie ceea ce sistemul va executa în procesul său de funcționare. Acestea reprezintă un model inițial conceptual al unui sistem în procesul de proiectare şi exploatare.

În figura 2.1 se definesc principalele funcționalități ale userului odată cu lansarea aplicației. Utilizatorul poate accesa Start Game,Resume,QuitGame,Main Menu, precum și utilizează TouchControale

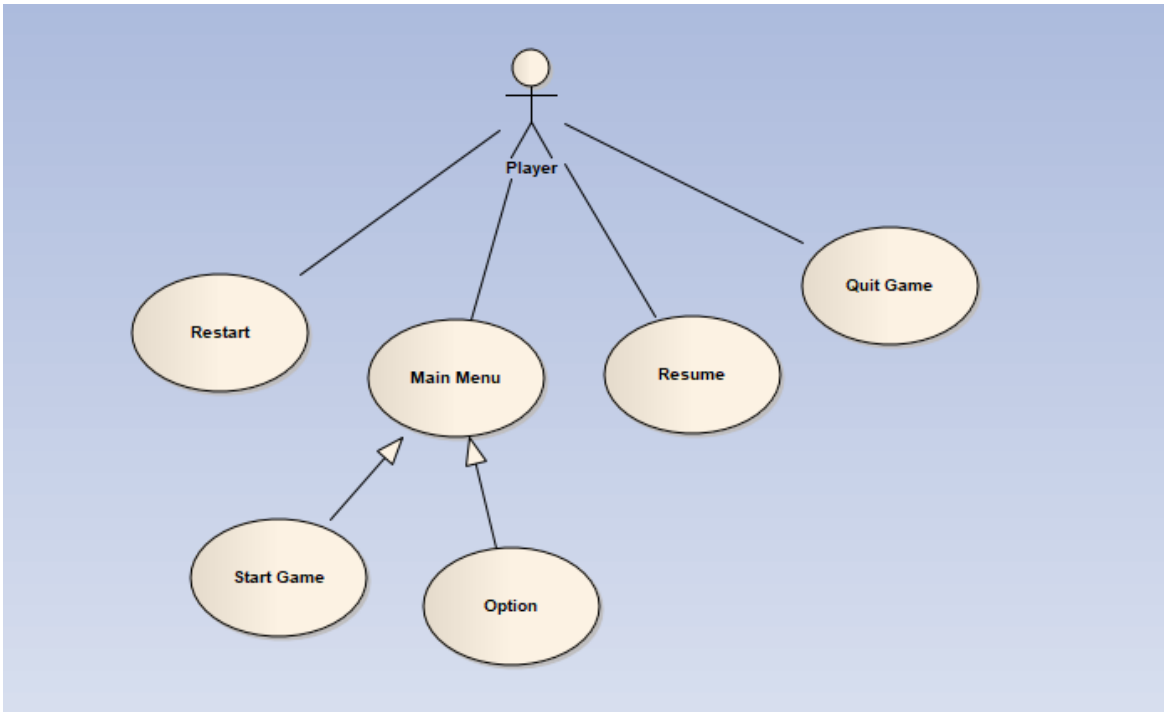

Figura 2.1- Diagrama Use Case, funcționalitățile predispuse utilizatorului

În figura nr 2.2 sunt descrise caracteristicile de bază a erouluui principal(Pucin). Acestea sunt evitarea obstacolelor, colectarea monedelor,efectuare săriturilor,obținerea unui scor,atacarea inamicilor.

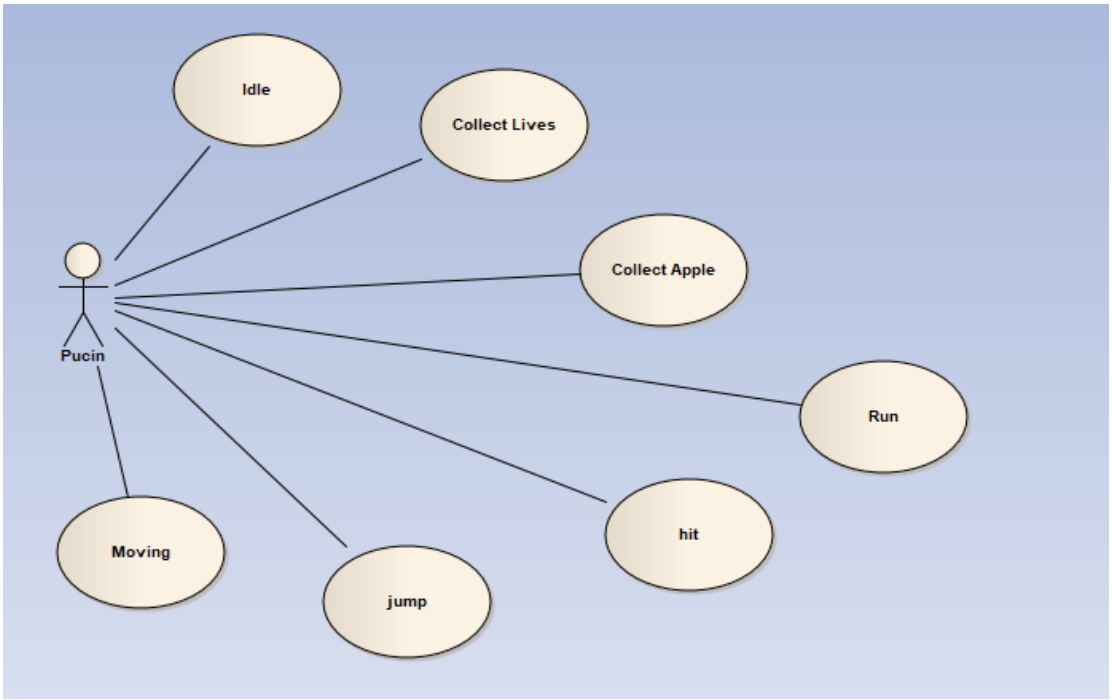

Figura 2.2- Diagrama use case, caracteristicile eroului principal

2.2 Reprezentarea particularităților temporale în cadrul diagramelor de secvențe/colaborare

Diagramele de secvenţă ilustrează interacţiunile dintre obiecte sau actori si obiecte din punct de vedere temporal. În figura 2.2.1 este reprezentat metoda de obținere a scorului. Eroul principal întîlnește în preajma sa monede(obiecte) pe care le ridică prin metoda PickUp(), după care acestea sunt destruse, și adaugă puncte la scorul personajului . O altă metodă de a obține scor este distrugerea inamicilor prin metoda OnCollider(), care la fel va apela metoda AddScore() adăugînd puncte la scorul personajului.

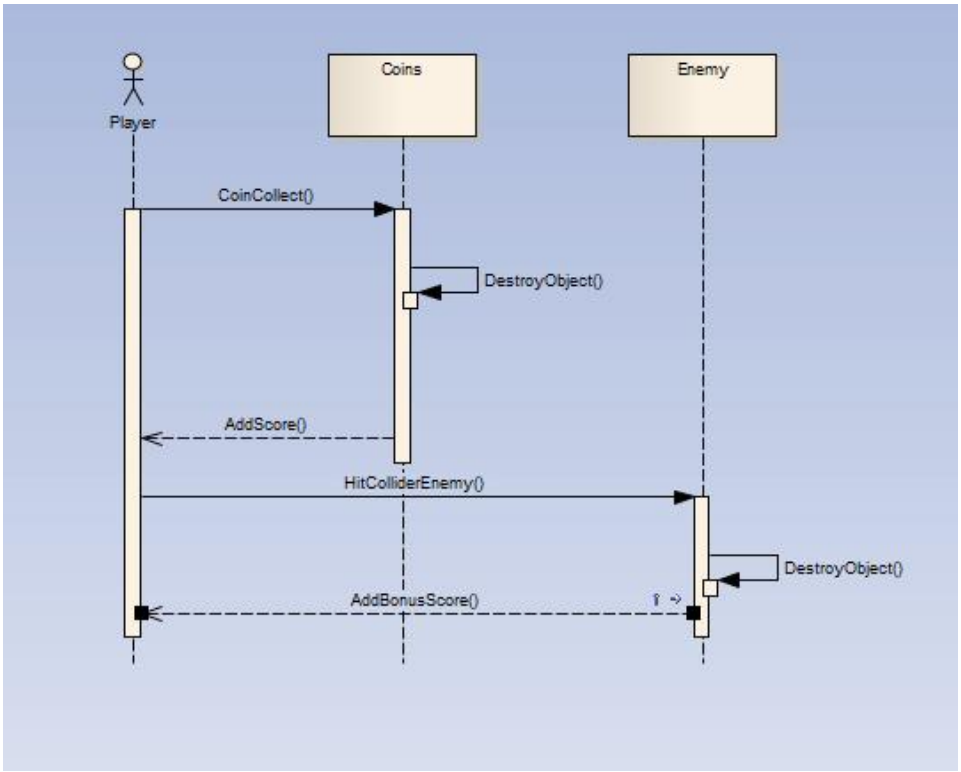

Figura 2.2.1 Adăugarea punctelor la scor

Următorul exemplu va reprezenta mișcarea Personajului doar în cazul în care el se află pe teren precum și accesarea nivelelor la interacțiunea cu ușa de intrare/ieșire. Acest exemplu se observă in figura 2.2.2.

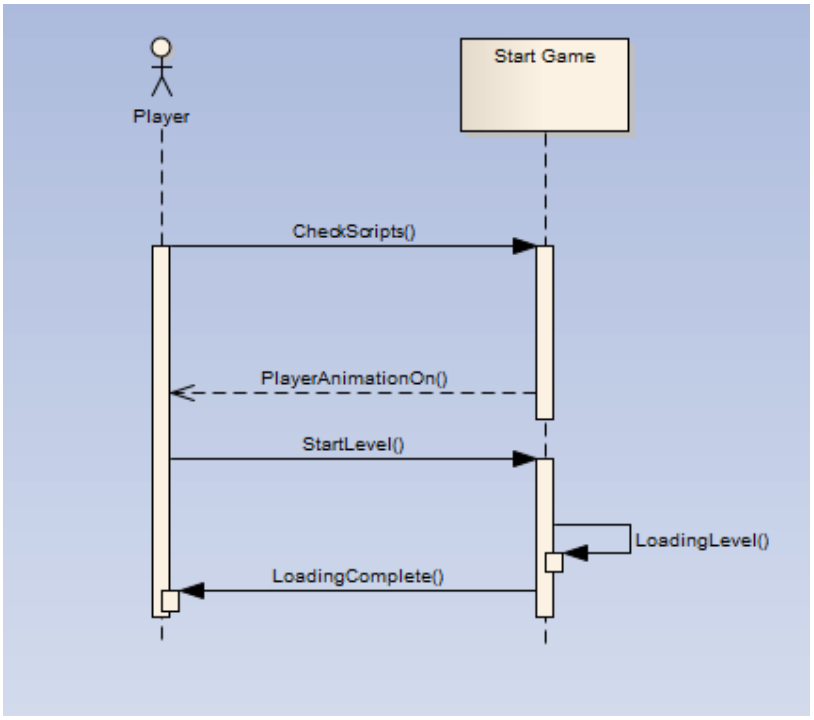

Figura 2.2.2 – Mișcarea si trecerea de la un nivel la altul

2.3 Colaborare

Particularitatea principală a diagramei de colaborare constă în posibilitatea de a reprezenta grafic nu numai consecutivitatea colaborării dar şi toate relaţiile structurale între obiecte. Spre deosebire de diagrama de secvenţă în diagrama de colaborare sunt reprezentate relaţiile între obiecte care sunt importante pentru colaborare.

Pentru atingerea unui scop sau pentru realizarea unui serviciu comportamentul unui sistem poate fi descris la nivelul obiectelor care fac schimb de mesaje, având importanță reflectarea legăturilor structurale ale obiectelor aparte. Diagrama de colaborare reflectă un fel de reprezentare statică a unui sistem ca totalitate de obiecte dependente.

În figura următoare este reprezentat colaborarea dintre erou si blocurile cu care interacționeaza și le poate distruge.

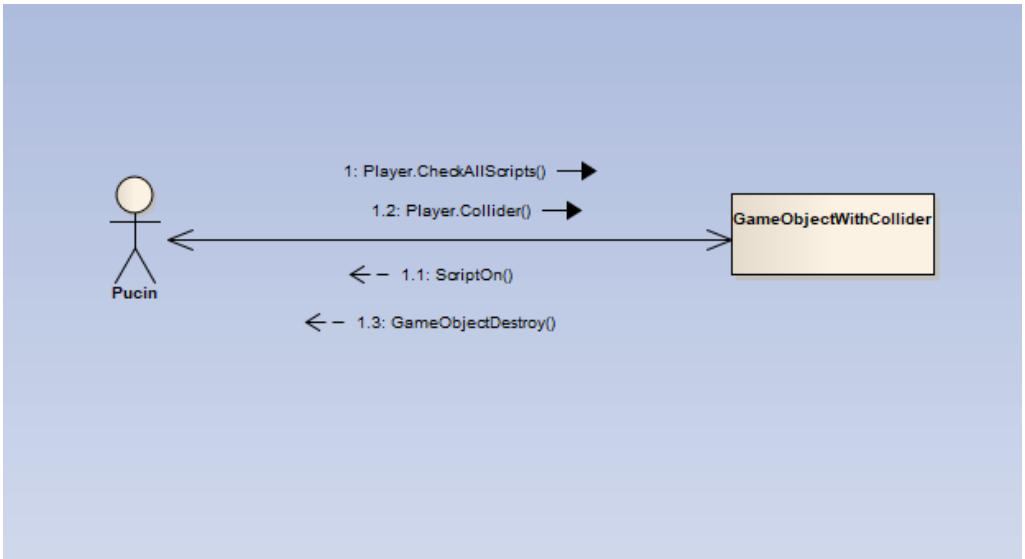

Figura 2.3.1 – Interacțiunea dintre erou și blocuri

Un alt tip de colaborare există între erou și dușmanii săi. Atunci cind eroul întilnește dușmanii , aceștia în mod automat încearcă să-l atace, la interacțiunea cu armele dușmanilor eroul pierde din scorul vieții, la rîndul său la interacțiune cu dușmanii , eroul este aruncat în spate. La atingerea inamicilor cu arma eroului aceștia sunt distruși, adăugînd puncte la scorul eroului(figura 2.3.4) în caz contrar eroul se dizolva în mai multe particule și joaca e finisată .

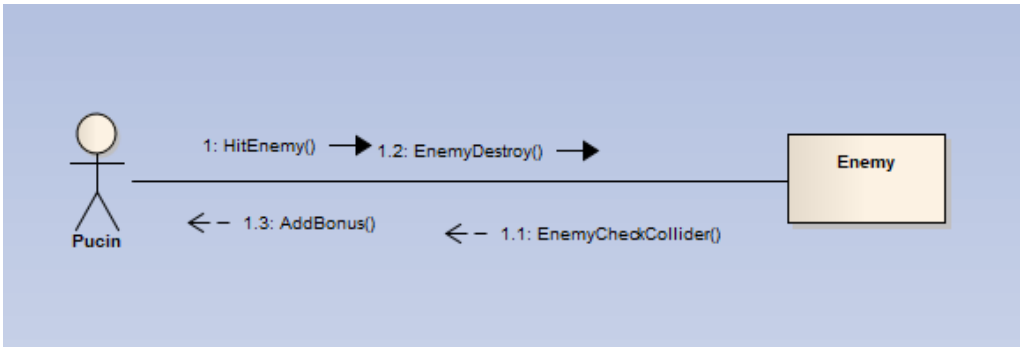

Figura 2.3.4 – Interacțiune erou-dușmani

3 Proiectarea sistemului

3.1 Proiectarea logică a sistemului. Diagrama de componente

Diagramele de componente sunt utile pentru a reprezenta structura fizică a componentelor sistemului și interdependența acestor componente. Scopul acestei diagrame este de a descrie componentele utilizate pentru oferirea gamei de funcționalități, la fel și vizualizarea componentelor fizice ale sistemului.

Pentru realizarea funcționalităților de bază, aplicația interacționează cu o serie de componente care sunt reprezentate în figura 3.1.1. O particularitate importantă în cadrul aplicației este rularea acesteia, care nu va avea loc fără interacțiunea cu fișierile de tip script , cu obiectele care le-am creat și cu care userul care va utilzia aplicația va putea manipula, precum și scenele pentru trecerea de la un nivel la altul.

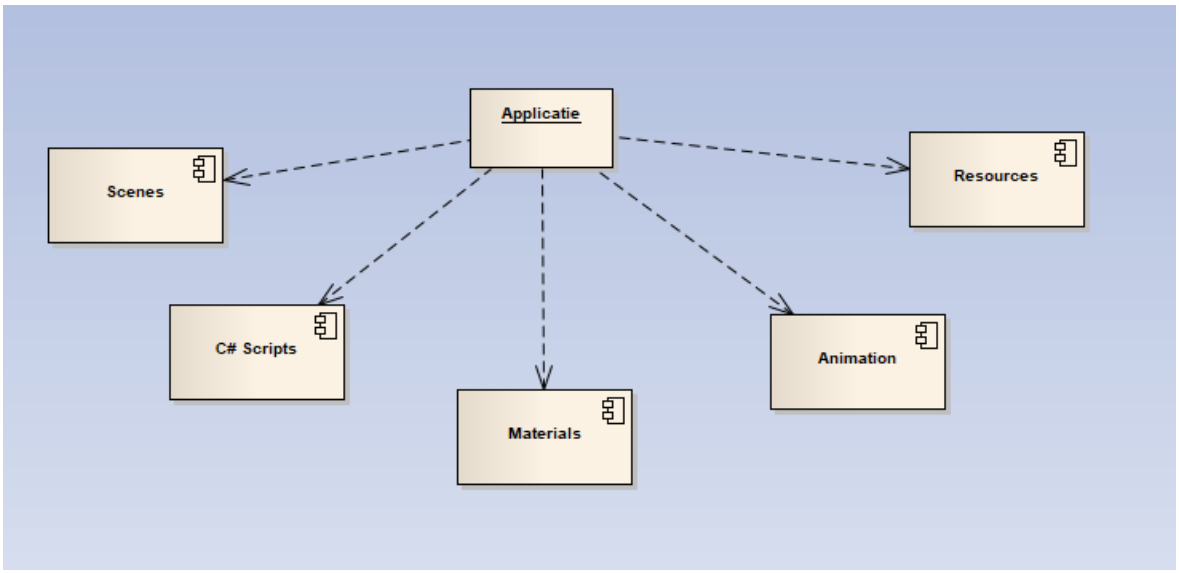

Figura 3.1.1- Diagrama de componente, principalele componente ale sistemului

3.2 Modelul structural static al sistemului proiectat. Diagrama de stări

O diagrama de stări modelează viața unui obiect prin stările sale și schimbările de stare care au loc pe parcursul vieții. Schimbările de stare sunt determinate de evenimente. O diagramă de stări reprezintă un automat cu stări finite. Diagramele de interacțiune modelează interacțiunile dintre obiecte. Diagramele de stări modelează efectul acestor interacțiuni asupra stării interne a fiecărui obiect.În figura 3.2.1 este reprezentată digrama de stări a iteme întilnite în joc . Iteme sunt obiecte ce influențează la scorul personajului atunci cînd acestea interacționează. Deci iteme vor fi prezente atîta timp cît personajul nu a interacționat cu ele

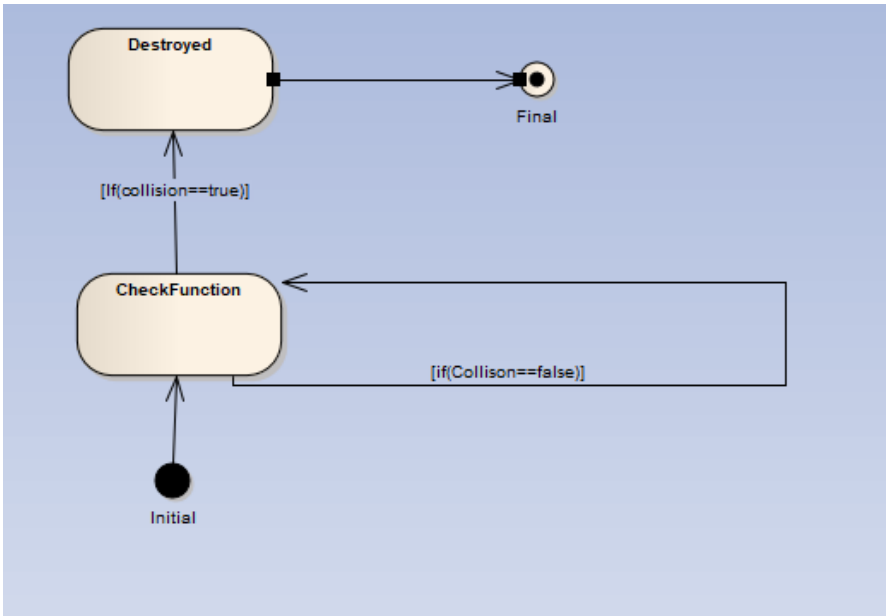

Figura 3.2.1 –Diagrama de stări a iteme

Pe lîngă iteme pesosnajul mai poate obține scor și distrugînd dușmanii. În figura 3.2.2 este reprezentată diagrama de stări a dușmanilor,care se află în stare de miscare atîta timp cît în radiusul lor nu este vazut personajul,apoi trec în starea ofensivă și sunt distruși atunci cind viața este egală cu 0 cu ajutorul unui collider.

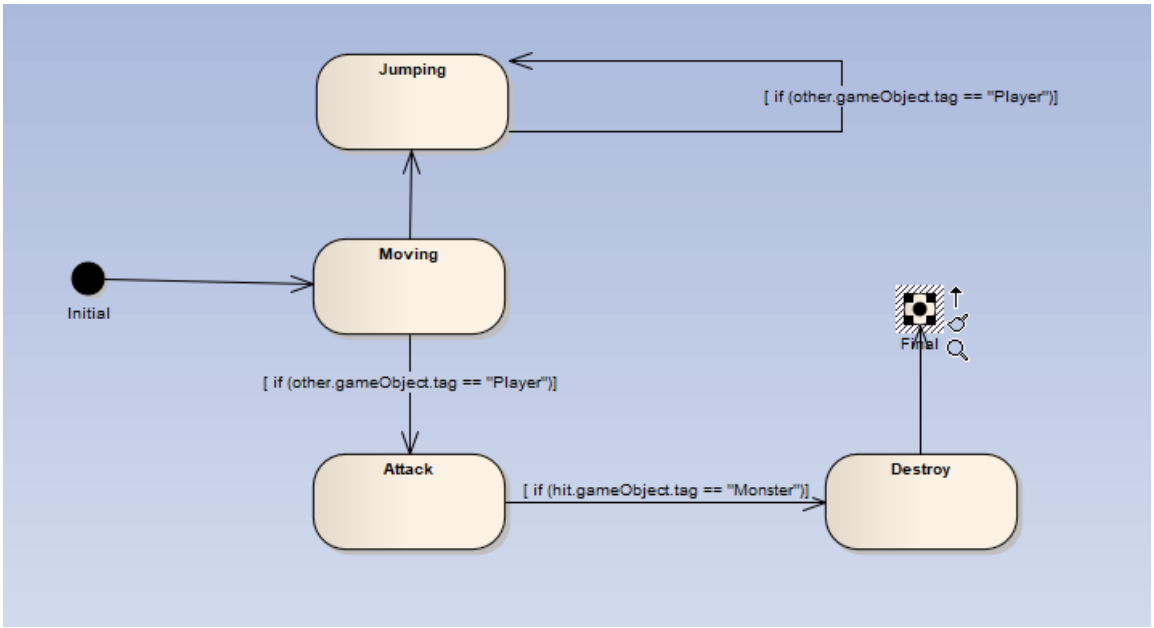

Figura 3.2.2 –Diagrama de stări a dușmanilor

3.3 Modelul structural static al sistemului proiectat. Diagrama de clase

În diagrama de clase din figurile 3.2.5 se descrie structura ierarhică a claselor și moștenirea proprietăților și comportamentul claselor părinte, având proprietățile și comportamentul său propriu, pe care nu le au clasa părinte. Controleer `CharacterController` are un rol esențial în cadrul acestei ierarhii, cu ajutorul ei ,utilizatorul poate controla eroul principal și îl pune în acțiune. Mai jos sunt reprezentate cele mai importante acțiuni realizate de către eroul principal.

```
if (moveJoystick.InputDirection == Vector3.Cross(Vector3.up, Vector3.forward))
     {
         anim.Play("run");
         transform.position += transform.forward * Time.deltaTime * movementSpeed;
         dir = moveJoystick.InputDirection;

     }

     if (moveJoystick.InputDirection != Vector3.zero)
     {
         CharacterController controller = GetComponent<CharacterController>();
         if (controller.isGrounded)
         {
```

```
                moveDirection = new Vector3(Input.GetAxis("Horizontal"), 0,
Input.GetAxis("Vertical"));
                moveDirection = transform.TransformDirection(moveDirection);
                moveDirection *= speed;

                {
                    moveDirection.y = jumpSpeed;
                    anim.Play("jump");
                }
                moveDirection.y = jumpSpeed;

                moveDirection.y -= gravity * Time.deltaTime;
                controller.Move(moveDirection * Time.deltaTime);
            }



        }
        if (moveJoystick.InputDirection == Vector3.zero)
        {

        }
        if (moveJoystick.InputDirection == new Vector3(-0.6f, 0, 0.5f))
        {
            Debug.Log("left");
        }
        if (moveJoystick.InputDirection == Vector3.right)
        {
            Debug.Log("right");
        }
        if (moveJoystick.InputDirection == Vector3.back)
        {
            Debug.Log("back");
        }


        {
            CharacterController controller = GetComponent<CharacterController>();
            if (controller.isGrounded)
            {
                moveDirection = new Vector3(Input.GetAxis("Horizontal"), 0,
Input.GetAxis("Vertical"));
                moveDirection = transform.TransformDirection(moveDirection);
                moveDirection *= speed;
                if (Input.GetKeyDown(KeyCode.Space))
                {
                    moveDirection.y = jumpSpeed;
                    anim.Play("jump");
                }
            }

            if (Input.GetKeyUp(KeyCode.Space))
                moveDirection.y = jumpSpeed;

            moveDirection.y -= gravity * Time.deltaTime;
            controller.Move(moveDirection * Time.deltaTime);
        }
        //-------------------------------------------------------------------------------------
---------------------------

        if(up==true)
        {
            anim.Play("run");
```

```
            transform.position += transform.forward * Time.deltaTime * movementSpeed;
        }
        if(left==true)
        {
            anim.Play("Armature_001Action_001");

            {
                transform.Rotate(0, Time.deltaTime * counterClockwise, 0);
            }

        }
        if(right==true)
        {
            anim.Play("Armature_001Action_001");

            {
                transform.Rotate(0, Time.deltaTime * clockwise, 0);
            }
        }

        //-------------------------------------------------------------------------------------
----------------------------
        if (Input.GetKeyDown(KeyCode.W))
        {
            anim.Play("run");
            transform.position += transform.forward * Time.deltaTime * movementSpeed;
        }
        if (Input.GetKeyUp(KeyCode.W))
        {
            anim.Play("Armature_001Action_001");
            // anim.SetInteger("run", 0);
            transform.position += transform.forward * Time.deltaTime * movementSpeed;

        }
        if (Input.GetKeyDown(KeyCode.A))
        {
            anim.Play("run");
            transform.position += transform.forward * Time.deltaTime * movementSpeed;
        }
        if (Input.GetKeyUp(KeyCode.A))
        {
            anim.Play("Armature_001Action_001");
            transform.position += transform.forward * Time.deltaTime * movementSpeed;
        }

        else if (Input.GetKey(KeyCode.S))
        {
            rb.position += Vector3.back * Time.deltaTime * movementSpeed;
        }

        if (Input.GetKey(KeyCode.E))
        {
            transform.Rotate(0, Time.deltaTime * clockwise, 0);

        }
        else if (Input.GetKey(KeyCode.Q))
        {
            transform.Rotate(0, Time.deltaTime * counterClockwise, 0);
        }
        else
        {
            // anim.Play("Armature_001Action_001");
        } }
```

În figura 3.3.1 și 3.3.2 sunt reprezentate trecerea personajului dintr-o poziție în alta,săritura și mișcare.

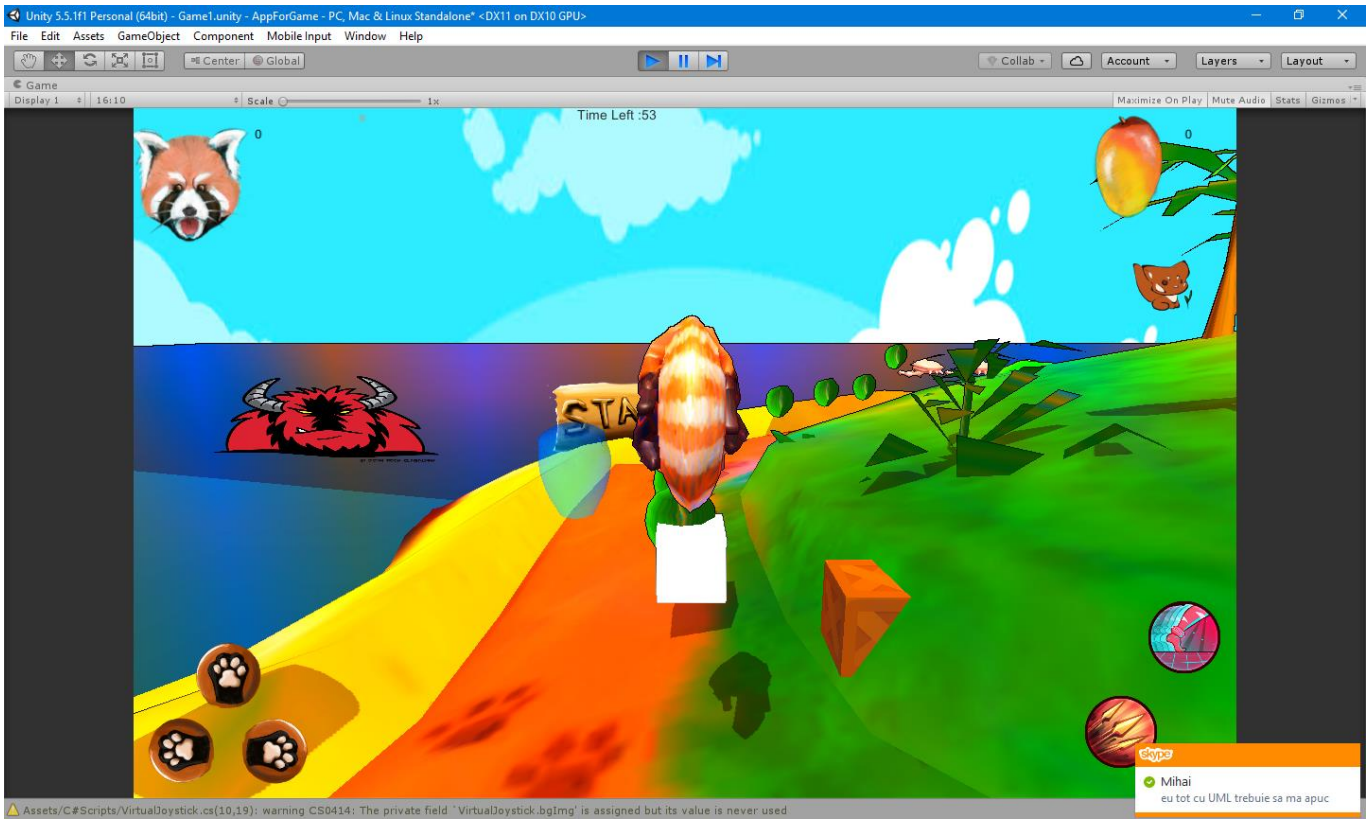

Figura 3.3.1- Săritura în înălțime a eroului principal

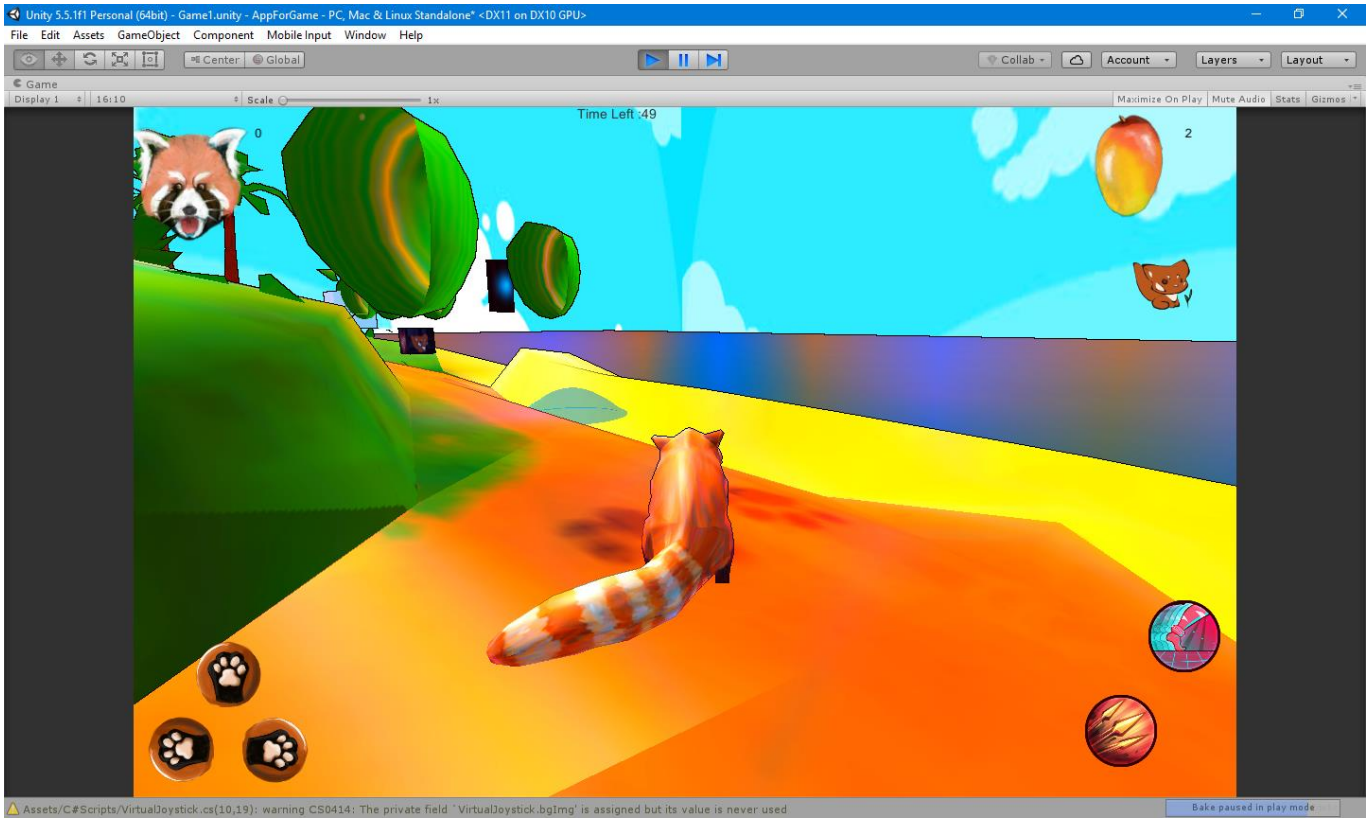

Figura 3.3.2- Mișcarea eroului

În timpul mișcărilor, personajul se poate pierde din ecranului, pentru astfel de cazuri este creată clasa CameraControll care îl va urmări pe erou pînă la fine apelînd funcția `GetComponent<CharacterController>();`, și transform.position() care va schimba poziția camerei pe axa X și Y.

```
void Start()
  {

      controller = GetComponent<CharacterController>();
      anim = GetComponent<Animator>();
      rb = GetComponent<Rigidbody>();
  }
  public void moving_foward()
  {
      anim.Play("run");
      transform.position += transform.forward * Time.deltaTime * movementSpeed;
  }

if(up==true)
      {
          anim.Play("run");
          transform.position += transform.forward * Time.deltaTime * movementSpeed;
      }
```

În cazul aparițiilor obstacolelor în timpul jocului, eroul are nevoie de implementarea anumitor acțiuni, precum ar fi lovirea obstacolelor sau lupta cu dusmani .Deci a fost creată functie, `OnTriggerEnter` și `OnControllerColliderHit` cu ajutorul caruia player-ul va putea trece peste orice obstacol. Codul este reprezentat mai jos .

```
  public void OnTriggerEnter(Collider other)
    {
        anim.Play("atk");
        controller.radius = 3f;
        //controller.radius = 4.5f;
        col = Vector3.up;
           // hit = true;
    }


  public void OnControllerColliderHit(ControllerColliderHit hit)
    {
        if (col == Vector3.up)
        {

            controller.radius = 2.4f;

            if (hit.gameObject.tag == "Monster")
            {
               // anim.Play("atk");
                Destroy(hit.gameObject);
            }
        }
        col = Vector3.zero;

           }
```

Efectul cel mai mare asupra păsirii peste obstacole o joacă atributul damage dar pe parcurs o sa codam , cu ajutorul metodei `OnControllerColliderHit` () se transmite valoarea numerică către obiectul cu Tag „ Monster”.

Un rol important în mișcarea personajului îl joaca numărul de vieți, deoarece nu putem mișca ceea ce nu există, deci mișcarea personajului depinde si de numărul de vieți de care dispune personajul. Tagul **„**`live`**”** împreuna cu atributul *lives* informează numărul de vieți atașați personajului.

```
void Start()
    {
        Player = GameObject.Find("Player");
        HH = (Texture)Resources.Load("pucinFace");
        HH1 = (Texture)Resources.Load("mango");
        texbutton = (Texture)Resources.Load("win");
        total_mango = PlayerPrefs.GetFloat("mango", (total_mango+ mango));
        onLoad();
    }



    private void OnControllerColliderHit(ControllerColliderHit hit)
    {
        if (hit.gameObject.tag == "travka" && lives < 1)
        {
            Destroy(hit.gameObject);
            SceneManager.LoadScene("GameOver");
        }
```

```
else if (hit.gameObject.tag == "travka")
{
    Destroy(hit.gameObject);
    lives--;
}

else if (hit.gameObject.tag == "live")
{
    Destroy(hit.gameObject);
    lives++;
}

else if (hit.gameObject.tag == "pizdets")
{
    lives--;
    Player.transform.position = new Vector3(0, 0, 0);

    if (lives < 1)
    { SceneManager.LoadScene("GameOver"); }



}
if (hit.gameObject.tag == "meat")
{
    Destroy(hit.gameObject);
    total_mango++;
}
if (hit.gameObject.tag == "Finish")
{
    level2open = true;
    Time.timeScale = 0f;
    finish = true;
    f1 = true;
    PlayerPrefs.SetFloat("m", total_mango);
}
if (hit.gameObject.tag == "Finish2")
{
    Time.timeScale = 0f;
    finish2 = true;
    PlayerPrefs.SetFloat("m", total_mango + mango);//6
}

   }
```

Așa cum se observă la inițierea jocului Player-ul are o variabila anumită de viața ,de aceea ce se observă și în codul de mai sus atribuind atributului lives cit mai mult o să ne jucăm

În figura 3.3.3 vor fi afișate toate legăturile dintre principalele clase ale proiectului dat.

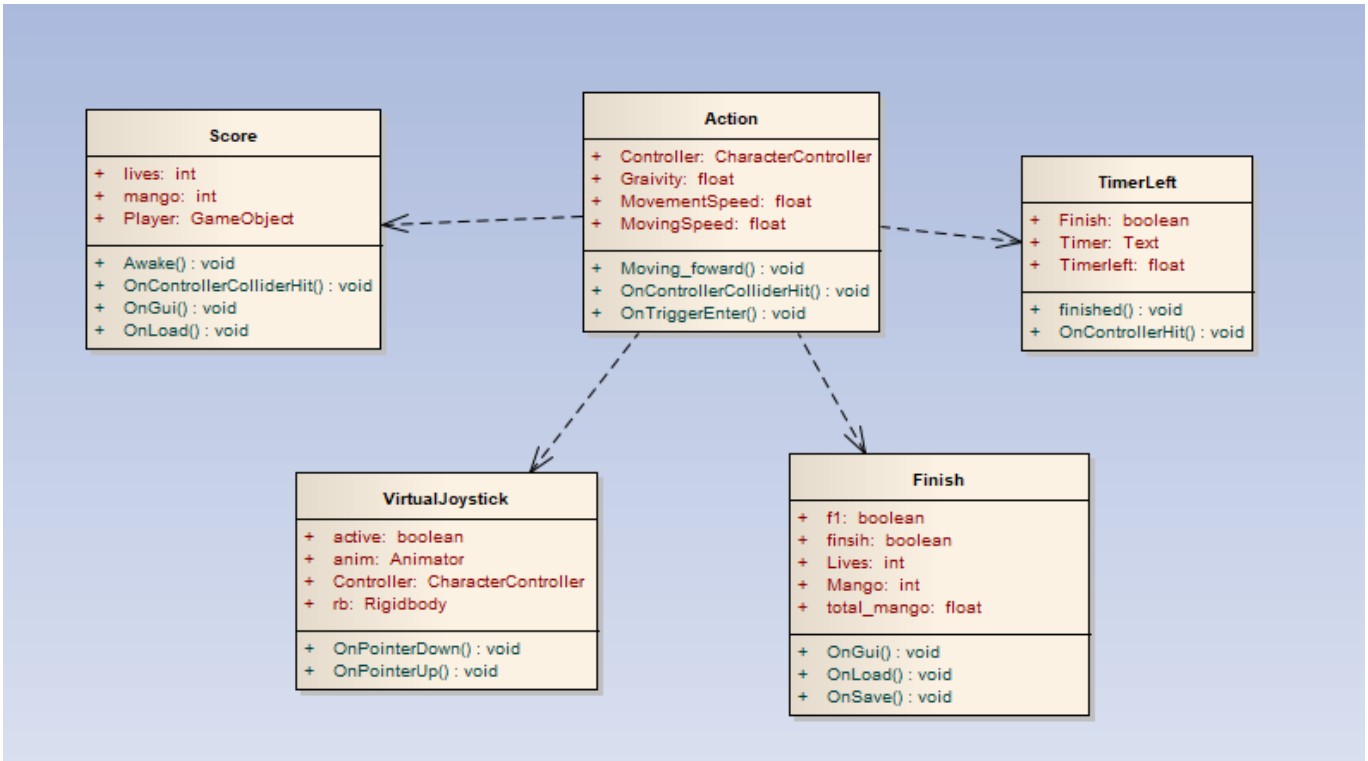

Figura 3.3.3– Relațiile PlayerController și alte clase

Cum și s-a menționat mai sus ,scopul jocului este de a obține un anumit scor și de a trece prin toate nivelele. Obținerea scorului poate avea loc prin strîngerea monedelor și distrugerea inamicilor. Clasa Score Manager va genera pe ecran scorul obținut de către erou.Sunt folosite Clasele : Score,Action,Timerleft,Virtualjoystick,Finish.

Așa cum obținem scor , așa și îl putem pierde atunci cînd eroul este doborît, deci clasa Score și TimerLeft ne va informa dacă pierdem sau nu din scor. Mai jos este afișată clasa Score

```
private void OnControllerColliderHit(ControllerColliderHit hit)
    {



        if (hit.gameObject.tag == "mango")
        {
            Destroy(hit.gameObject);
            //cube= (GameObject)Instantiate(cube);
            mango++;
        }
}
```

```
void onSave()
    {
       // PlayerPrefs.SetString("", Player);
        PlayerPrefs.SetFloat("mango", mango);

    }
```

Așa cum se observă metoda OnTriggerEnter2D apelează metoda AddPoints al clasei Score pentru a transmite valoarea numerica obținută în urma interacționării cu monedele. Scorul inițial este 0 așa cum este reprezentat în figura 3.3.4 .

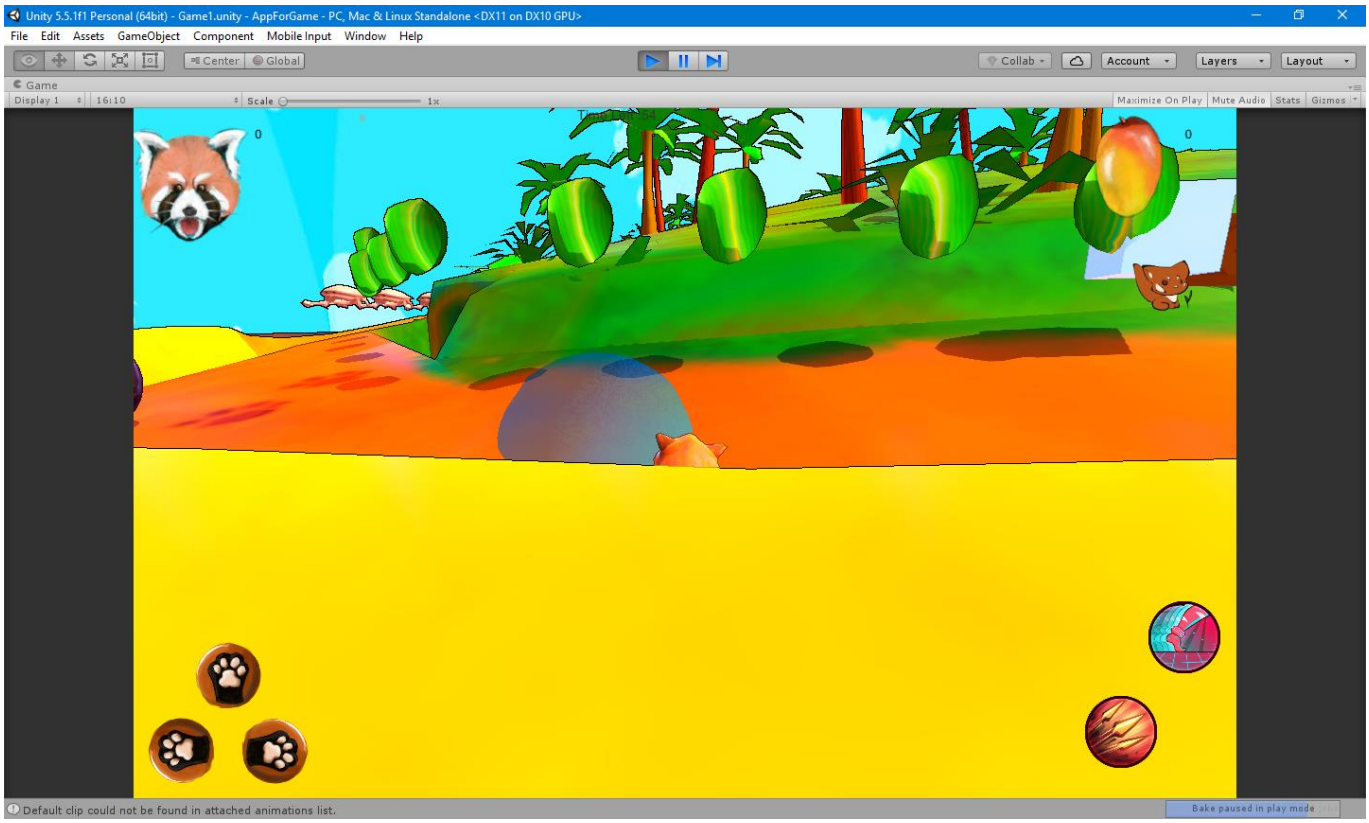

Figura 3.3.4 – Scorul inițial

În figura de mai sus ,se observă prezența gameobjecturi , care vor influența asupra scorului eroului în timpul interacțiunii iar în figura 3.3.5 , mărirea scorului în timpul strîngerii itemelor.

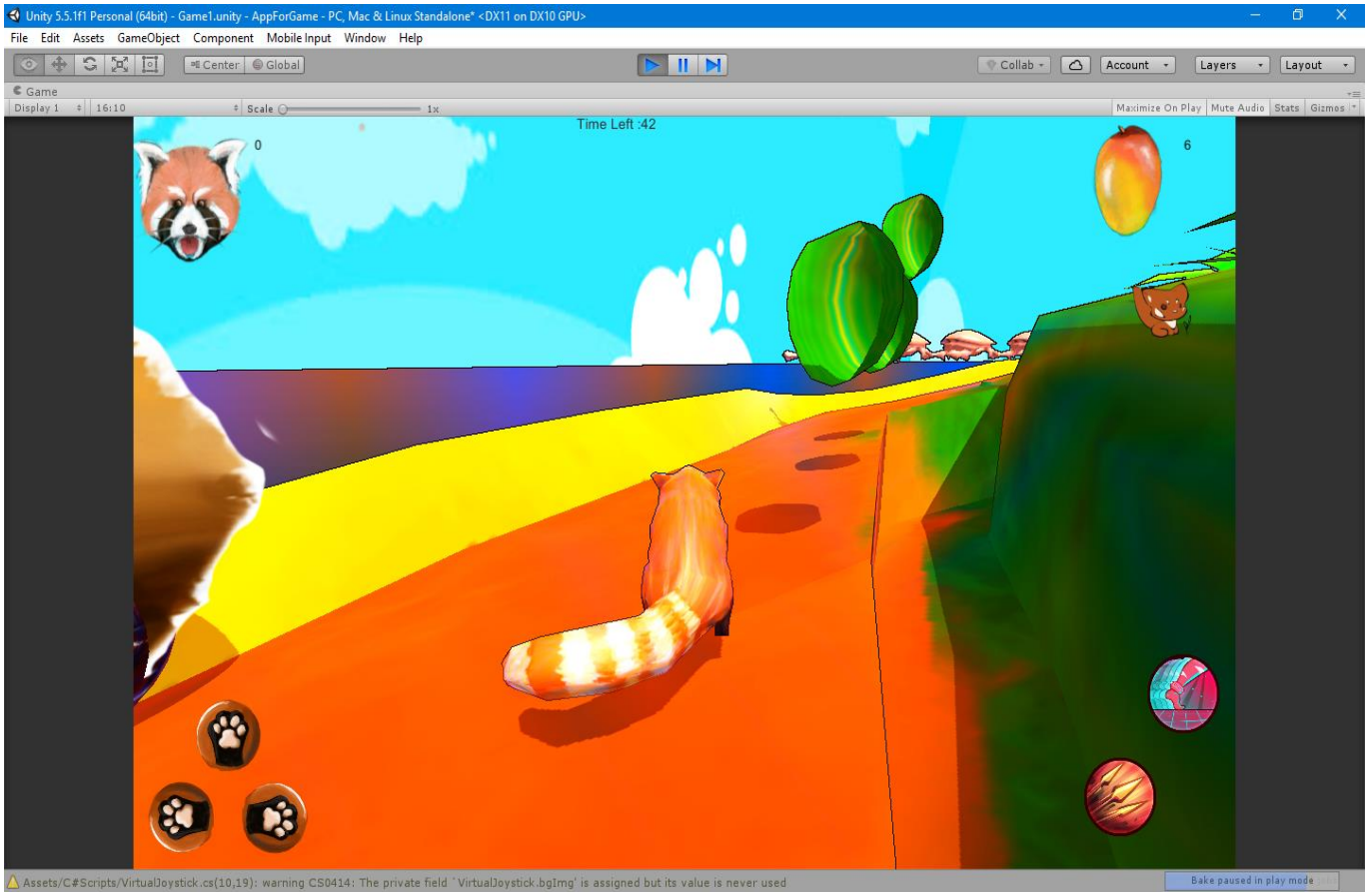

Figura 3.3.5 – Obținerea scorului cu ajutorul itemelor

În figura de mai sus ,se observă prezența celor a unitatilor colectate , care vor influența asupra scorului eroului în timpul interacțiunii iar în figura 3.3.5 , mărirea scorului în timpul strîngerii itemelor. Colectare fructelor sau monetelor este esenta jocului pentru a finisa jocul lui Pucin este nevoie se colecteze cite mai multe unitatii dintre care va fi placut si jocatorului si insa Pucinului.

3.4 Elaborarea diagramei de plasare

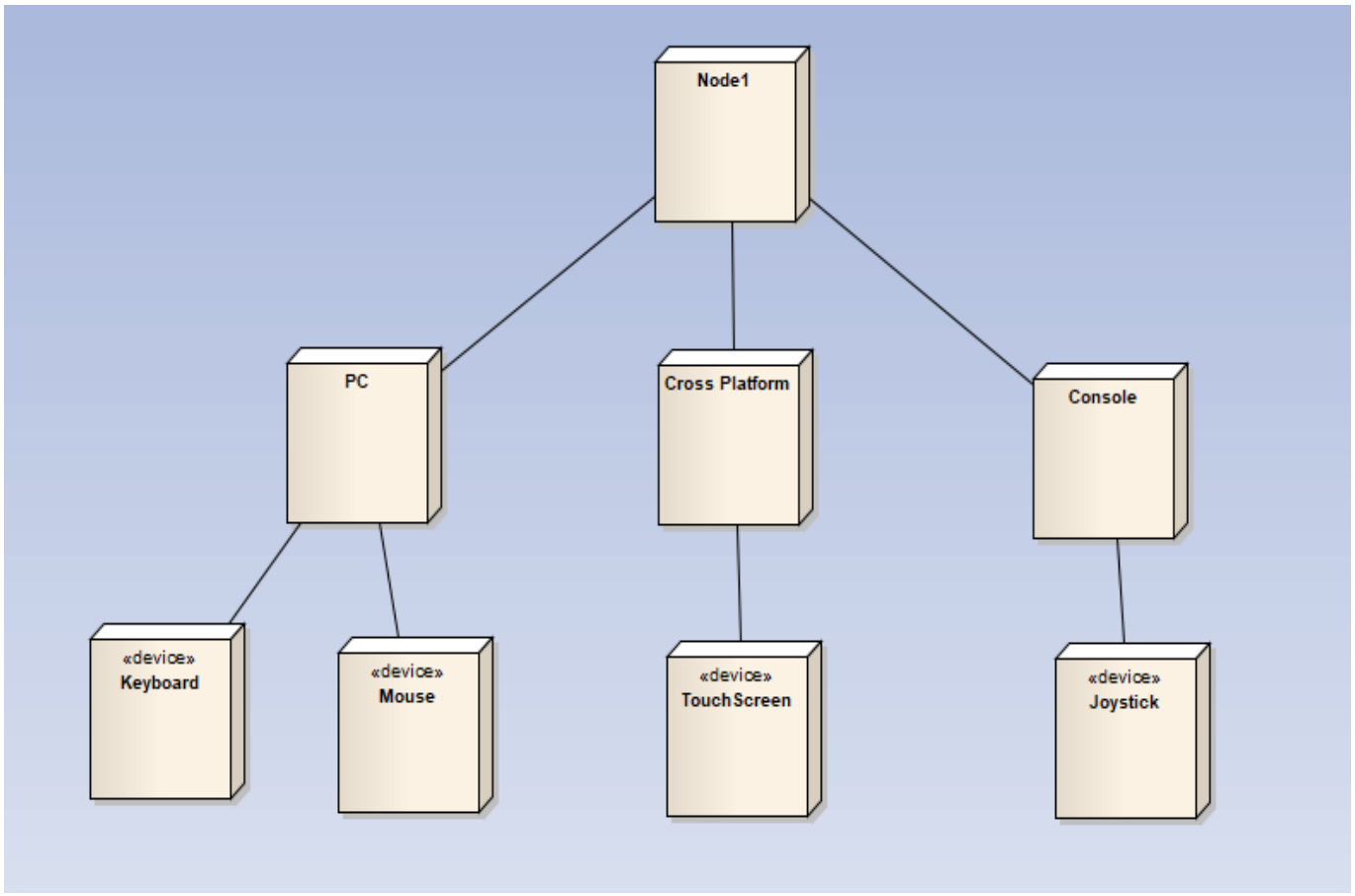

Figura 3.3.7 – Diagrama de amplasare a sistemului

În figura 3.3.7 este rerezentată diagrama de desfașurare în care se descrie accesul la server Joc pe nivele. Nivelul cel mai de jos reprezinta dispozitivele cu ajutorul carora se introduce infosrmația la nivelul 2 care reprezintă niște procesoare care au capacitatea de calcul, cum ar fi PC,Cross-Platform,Console(Sony Playstation , X-Box,etc).

3.5 Elaborarea diagramelor de activități

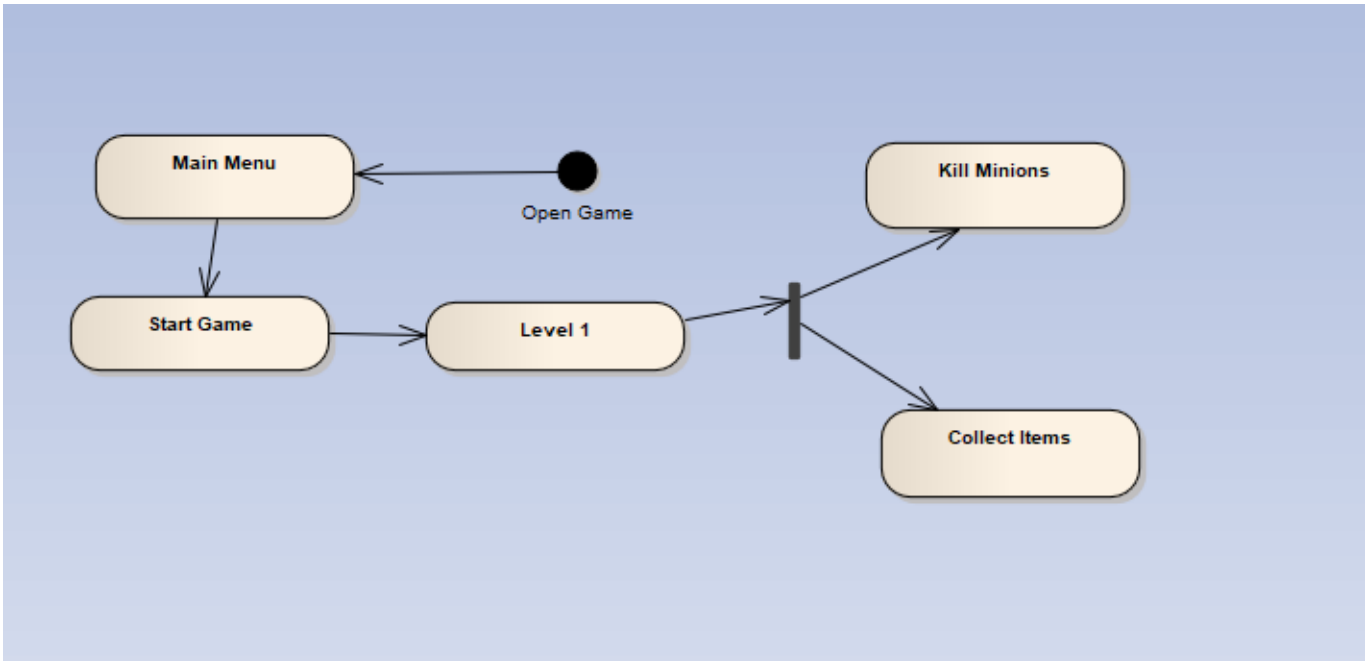

Figura 3.3.8 – Diagrama de activitate a nivelului 5

Pentru a putea Termina nivelul Pucin trebuie sa execute niste activitati care sunt reprezentate in Figura 3.3.8 , odata cu alegerea start game jucatorul nimereste intr-o lume desenata in 3d unde ai nevoie de a evita obstacole sau colecta iteme de pe nivel dar pentru a trece nivel trebuie personajul trebuie sa intre in contact cu obiectul de tip „finish” apoi se deschide access la alt nivel.

4. Elementele de bază a proiectului

Cum și a fost menționat mai sus scopul proiectului dat , este elaborarea unui joc 3D. Astfel pentru a-l aduce la bun sfîrșit, îl inițializăm cu toate elementele necesare.

Din elementele necesare fac parte :

- Scenes;
- Animation;
- Collision;
- Scripting;
- Materials;
- Sprites.

4.1 Scenele și Figurile

Scenele conțin obiecte ale jocului. Ele pot fi folosite pentru a crea un meniu principal, la nivel individual, și orice altceva. Fiecare fișier Scene este unic. În fiecare scenă, plasăm mediile, obstacole, si decorațiuni, în esență, proiectarăm și construim jocul pe bucăți. Proiectul în cauză conține 5 scene : meniul și cele 4 nivele.

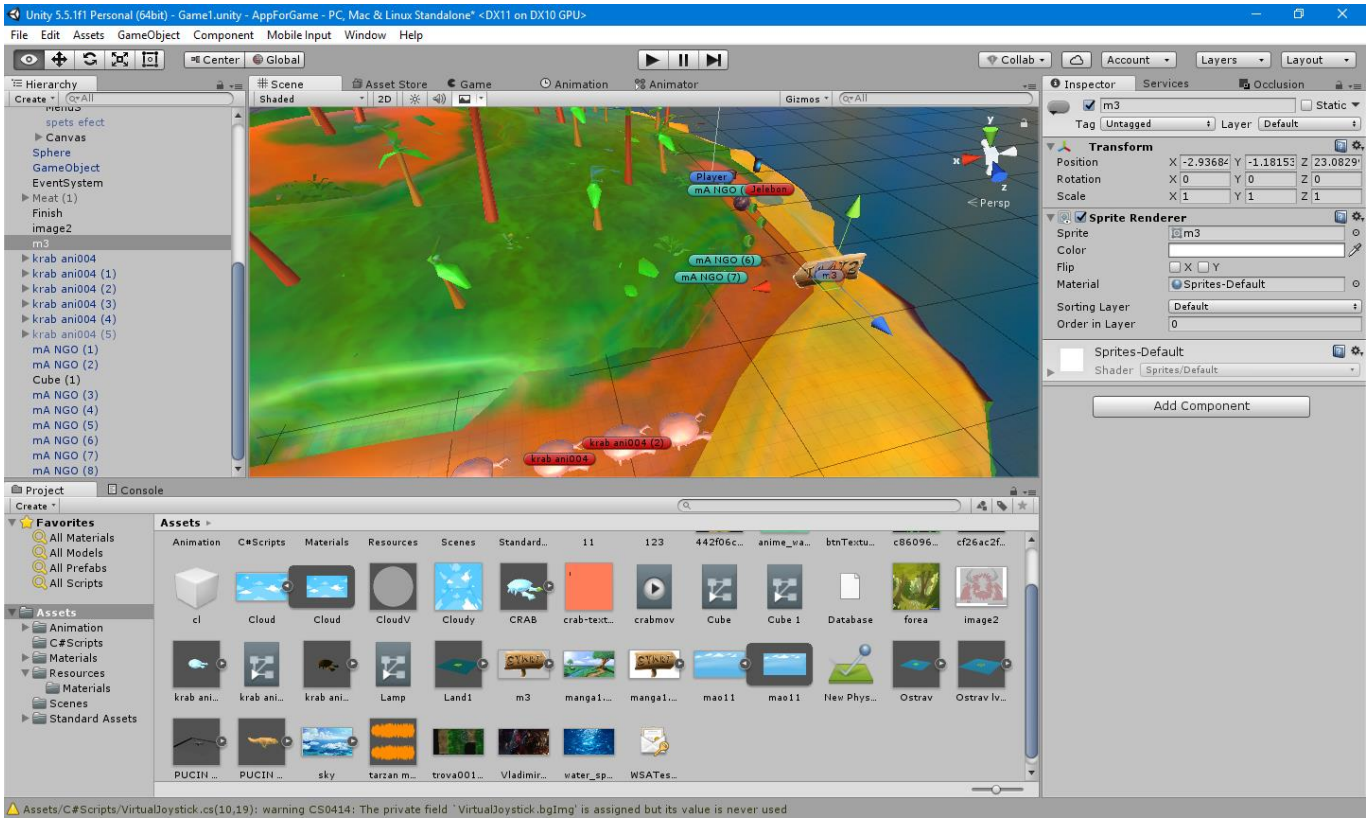

Figura 4.1.1 –Exemplu de scenă

Este foarte important de reținut , că scenele sunt unice deci nu interacționeaza automat între ele, atunci cînd sărim de la o scena la alta , nu uităm de 2 parametri foarte importanți , numele Scenei , și arhivarea acestora în submeniul Build Scene , unde fiecărei scene îi vom atribui un număr de prioritate,pentru a face legătură între ele și ca Unity Editorul să cunoască care e scena principală. În figura 4.1 sunt afișate prioritățile fiecărei scene din Submeniul Build.

Obiectele sunt elementele pe care le creeăm pentru ca user-ul să interacționeze cu ele în cadrul jocului. Eroul principal (Pucin) este un obiect 3D creat prin Blender , cu ajutorul Blender-ului am creat o animatie, un editor de Animation în Unity am adaugat o animatie la personaj.Pentru a extrage animarea la personaj am nevoie de o animare efectuate deja pe pc .

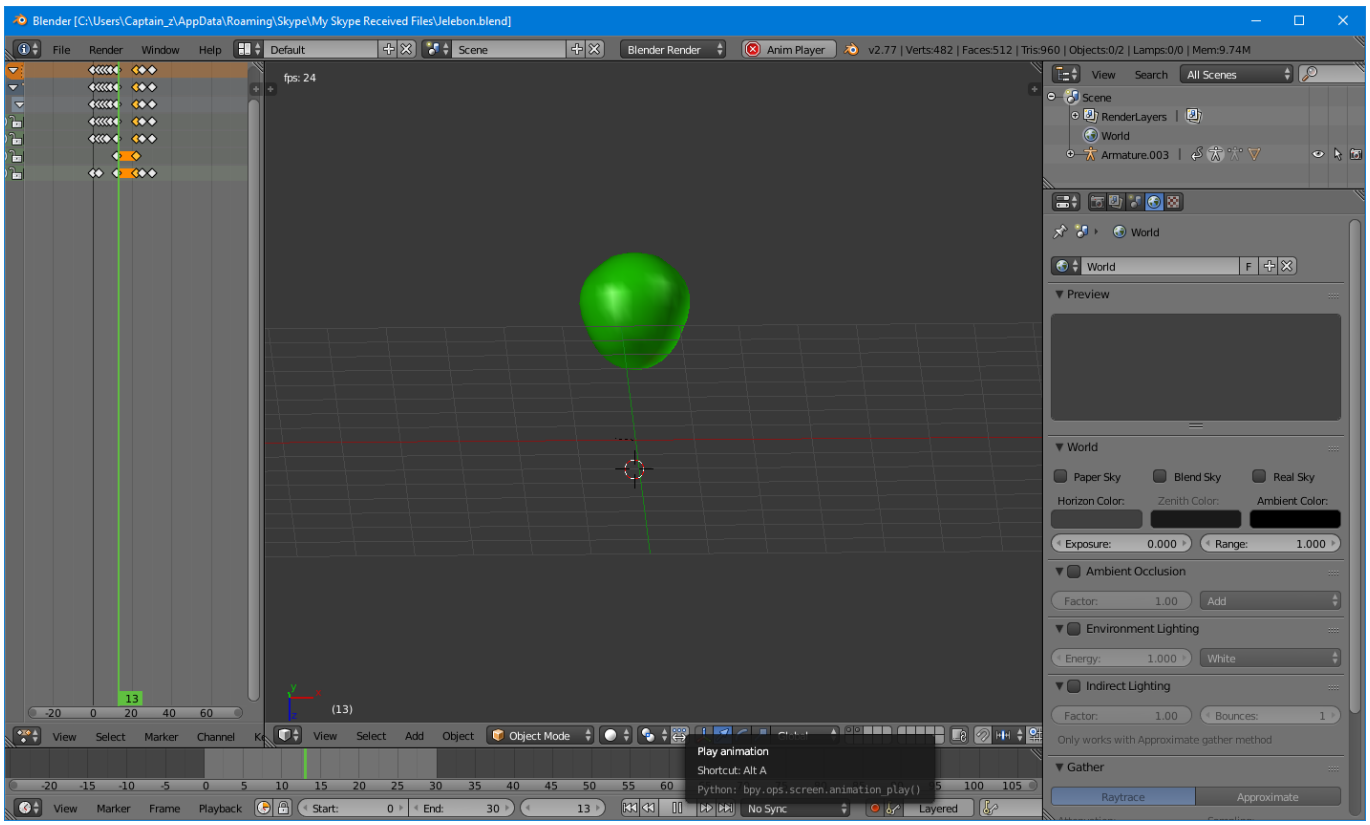

Figura 4.1.2 – Blender

Alte elemente utilizate în cadrul Blender sunt :

- Crearea eroului principal (Pucin);
- Crearea nivelelor;
- Crearea dușmanilor;
- Crearea animatiilor;
- Desenarea texturilor;
- Adaugarea scheletului la obiecte;
- Importarea Obiectelor pe Unity 3D.

În figura 4.1.3 de mai jos vor fi afișate toate elementele secundare utilizate în cadrul jocului.

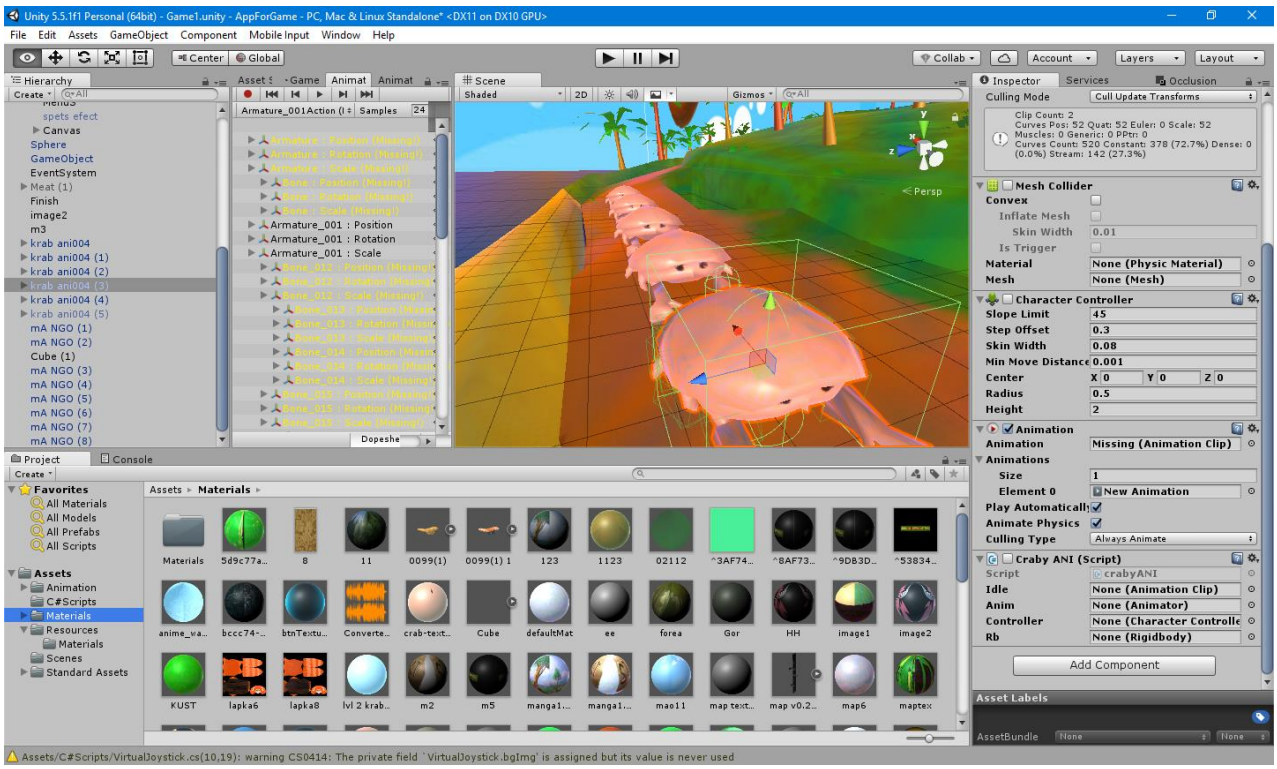

Figura 4.1.3 – Elementele secundare ale jocului

4.2 Animația

Testând joaca vom observa că accesînd tastele „w,s,d,a” sau a apasa pe buttone,eroul va începe se miste de pe loc, la fel pentru functia de attack si jump sunt folosite alte tastaturi „f” si „space” cind folosim „f” attackam dușmanii , iar pentru salt avem nevoie de „space” Toate acestea se datorează animației. A anima ceva inseamnă, literal, a-l aduce la viață, a-i da viață. Animația, în accepțiunea uzuală, cuprinde toate modificările care au un efect vizual intr-un joc 3D. Efectele vizuale pot fi de diferite feluri: poziții care variază în timp (motion dynamics - dinamica mișcării), modificări ale formei, culorilor, transparenței, structurii și a texturii suprafețelor unui obiect (update dynamics - dinamica infațișării)

Motorul Unity3D , dispune de editorul Animație, unde am folosit un obiect 3D la un anumit interval de timp . Unity va crea un fișier Anim , în care va înregistra schimbările efectuate.În figura 4.2.1 și 4.2.2 putem observa că la un anumit interval de timp și anume într-o secundă , eroul principal va realiza 4 acțiuni : jump, , run ,idle,attack.

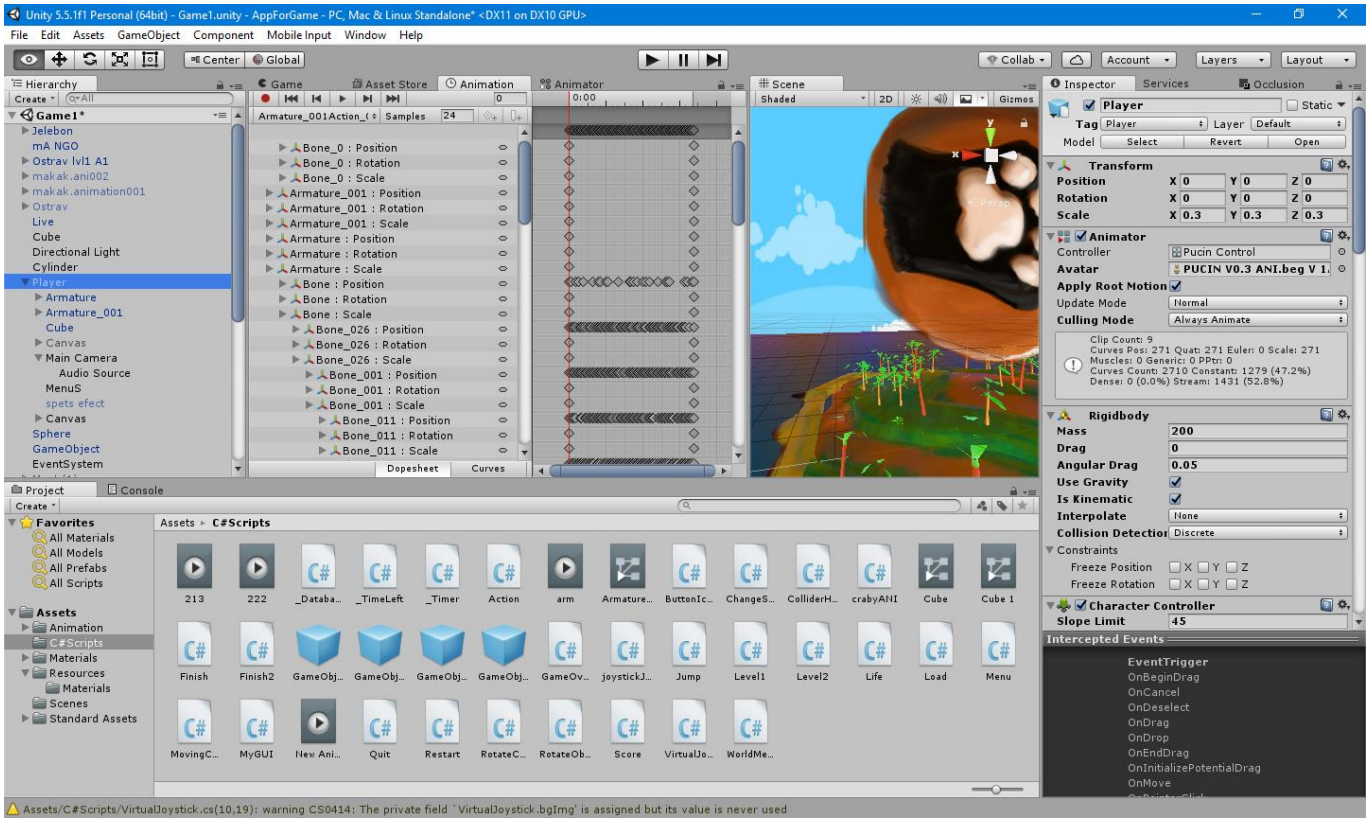

Figura 4.2.1 – Menținerea sabiei într-o poziție

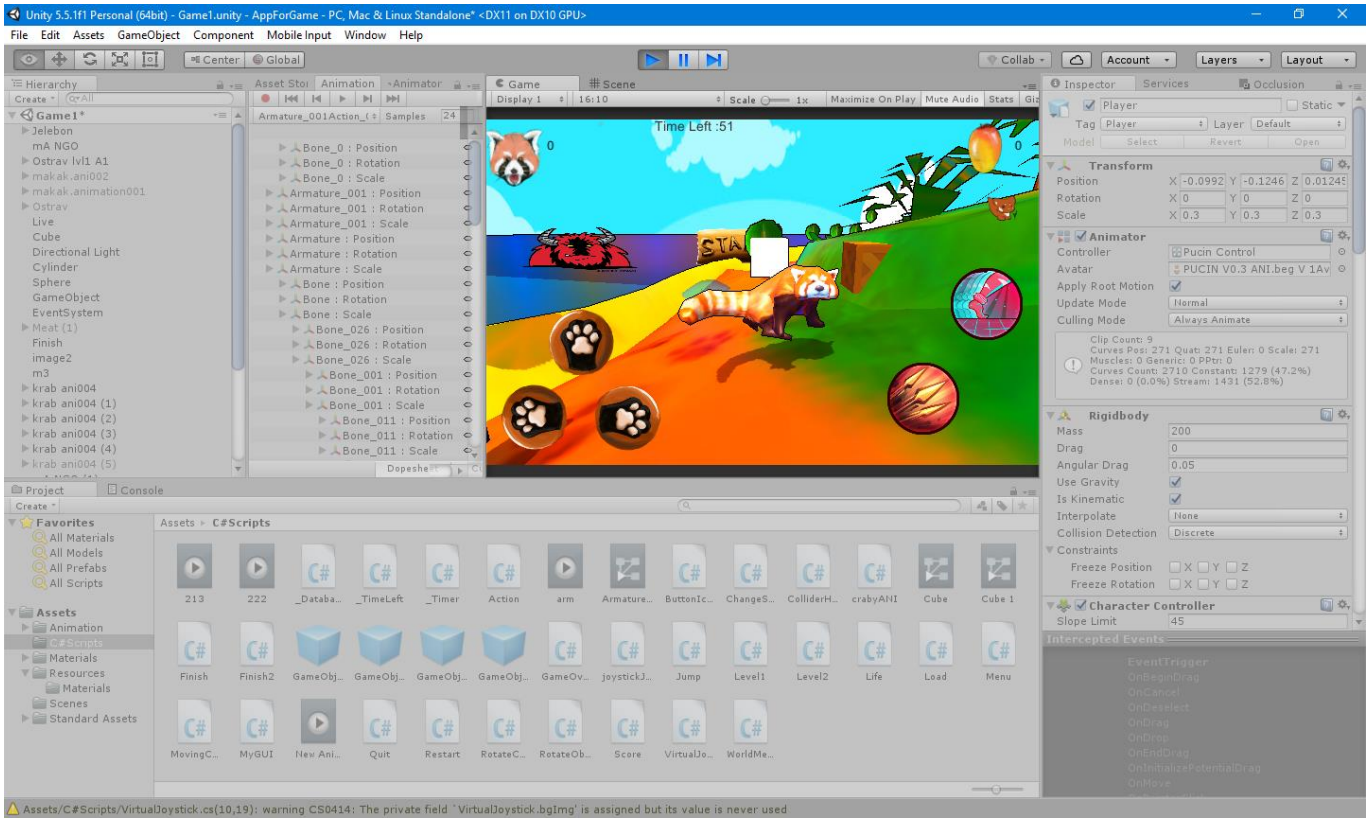

Figura 4.2.2 – Pozitie de lovire

Pe lîngă înregistrarea efectuată ,Unity mai dispune și de Animator, o interfață de control al mecanismului de animație , reprezentat în figura 4.2.3.

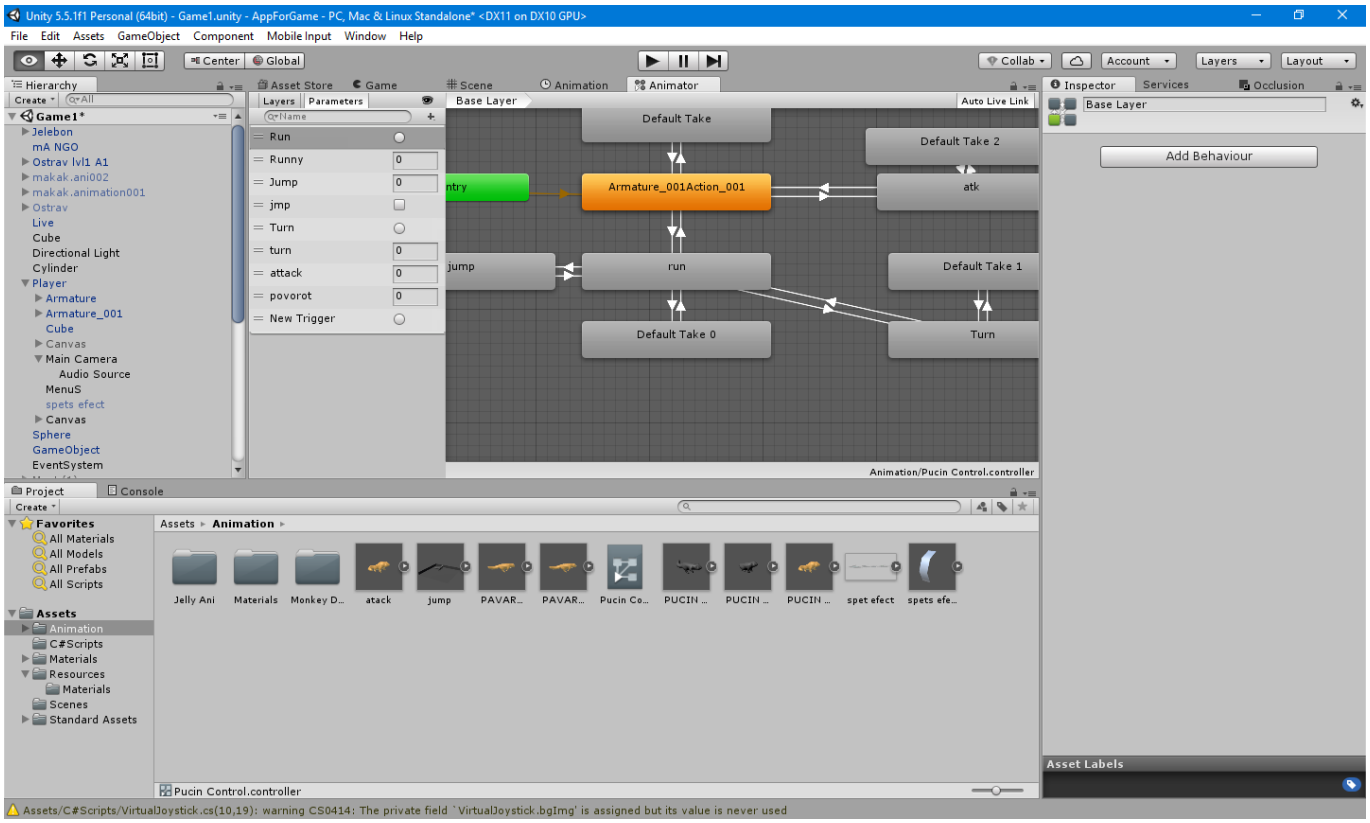

Figura 4.2.3– Animator

Așa cum observăm avem 4 dreptunghiuri , unul legat de celălalt prin referințe. Dreptunghiul Entry raspunde pentru animația la începutul pornirii jocului. jump , animație în timpul saltului, atk - animația ce corespunde in timpul unui attack și run - eroului principal v-a fuge.

Dar asta nu e de ajuns pentru a porni animațiile să lucreze , Unity3D mai cere și atașarea componentelor animației personajului , precum și anunțarea pornirii acesteia prin setarea unei variabile isBool prin True sau False .

```
if(up==true)
{
    anim.Play("run");
    transform.position += transform.forward * Time.deltaTime * movementSpeed;
}
if(left==true)
{
    anim.Play("Armature_001Action_001");

    {
        transform.Rotate(0, Time.deltaTime * counterClockwise, 0);
    }

}
if(right==true)
```

```
{
    anim.Play("Armature_001Action_001");

    {
        transform.Rotate(0, Time.deltaTime * clockwise, 0);
    }
```

4.3 Coliziunea

Coliziunea reprezintă întîlnirea particulelor sau corpurilor în care fiecare exercită o forță asupra celuilalt, provocând schimbul de energie sau impuls . Unity3D dispune de obiecte *Collider* de diferite forme (*Box Collider, Circle Collider,* etc) . Cu o poziționare atentă și dimensionare, collider-urile combinate pot ocupa obiectul în întregime.

Atunci când au loc coliziuni, motorul numește funcții cu nume specifice cu privire la orice script atașat la obiectele implicate. Avem posibilitatea să plasăm orice cod pe care dorim în aceste funcții pentru a răspunde la evenimentul de coliziune. De exemplu, să se distrugă un bloc, atunci cînd interacționăm cu sabia.

La prima actualizare a coliziunii,utilizăm funcția *OnCollisionEnter*. În timpul actualizărilor în care contactul dintre obiecte se menține pe o perioada mai lungă folosim *OnCollisionStay* și în cele din urmă, *OnCollisionExit* indică faptul că un contact a fost rupt.

5 Principii OOP

Programarea orientată pe obiect (Programare Orientată Obiectual) este unul din cei mai importanți pași făcuți în evoluția limbajelor de programare spre o mai puternică abstractizare în implementarea programelor. Ea a apărut din necesitatea exprimării problemei într-un mod mai natural ființei umane. Astfel unitățile care alcătuiesc un program se apropie mai mult de modul nostru de a gândi decât modul de lucru al calculatorului. Deși tehnica se numește "Programare Orientată Obiectual", conceptul de bază al ei este Clasa. Clasa, pe lângă faptul că abstractizează foarte mult analiza/sinteza problemei, are proprietatea de generalitate, ea desemnând o mulțime de obiecte care împart o serie de proprietăți.

Programarea orientată pe obiecte se bazează, sau este comusă și din anumite principii.

Principiile OOP :

- abstractizarea ;
- moștenirea ;
- încapsularea ;
- polimorfismul ;
- compoziție ;
- agregare .

În cadrul proiectului nostru am utilizat Încapsularea pentru fiecare clasa declarată. Încapsularea reprezintă ascunderea de informații: Asigură faptul că obiectele nu pot schimba starea internă a altor obiecte în mod direct (ci doar prin metode puse la dispoziție de obiectul respectiv), doar metodele proprii ale obiectului pot accesa starea acestuia. În codul de mai jos sunt declarate atributele clasei `CharacterController`, utilizînd Încapsularea.

```
    public GameObject Player;
        //-----------------
    public AnimationClip idle;
    public Animator anim;
    public CharacterController controller;
    public Rigidbody rb;
    public bool hit = false;
    public Vector3 InputDirection { set; get; }
           public Vector3 col { set; get; }

  void Start()
    {

        controller = GetComponent<CharacterController>();
        anim = GetComponent<Animator>();
        rb = GetComponent<Rigidbody>();
    }
```

În cadrul unui obiect, codul și datele pot fi private sau public. Cînd sunt private, ele sunt vizibile și accesibile doar în interiorul obiectului. În cazul public, celelalte părți ale programului le pot utiliza .

În cazul nostru Unity Editor ne permite să utilizăm elementele publice direct din bara de navigare așa cum vedem în figura 5.1 .

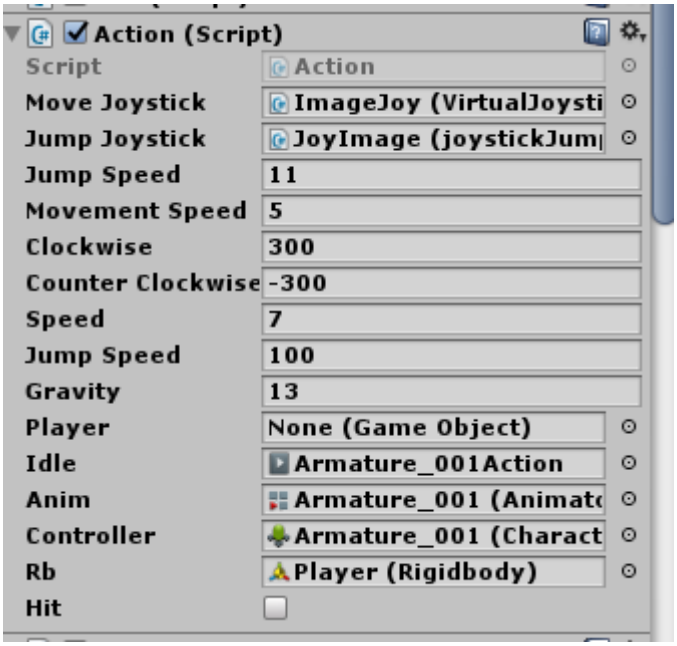

Figura 5.1 – Elementele publice ale clasei.

Pentru a utiliza metodele Unity pentru fiecare clasă aparte, am utilizat principiul Moștenirii , moștenind Clasa Action, clasa de bază de unde derivă toate scripturile.

Foarte des utilizînd o clasă avem nevoie să interacționăm cu obiectul altei clase, sau să extragem componentele acestuia , deci folosim principiul , relația de compoziție(hasA).

În codul de mai jos voi reprezenta clasa CameraController, în care vom declara un obiect al clasei `CharacterController`, îi vom găsi tipul prin metoda `= GetComponent<CharacterController>();`, și îl vom utiliza în metoda

```
    void Update()
{

    Vector3 col = Vector3.forward;
    Vector3 dir = Vector3.zero;
    dir.x = Input.GetAxis("Horizontal");
    dir.z = Input.GetAxis("Vertical");
    if (dir.magnitude > 1)
        dir.Normalize();


    if (moveJoystick.InputDirection == Vector3.Cross(Vector3.up, Vector3.forward))
    {
        anim.Play("run");
        transform.position += transform.forward * Time.deltaTime * movementSpeed;
        dir = moveJoystick.InputDirection;
```

```
        }

        if (moveJoystick.InputDirection != Vector3.zero)
        {
            CharacterController controller = GetComponent<CharacterController>();
            if (controller.isGrounded)
            {
                moveDirection = new Vector3(Input.GetAxis("Horizontal"), 0,
Input.GetAxis("Vertical"));
                moveDirection = transform.TransformDirection(moveDirection);
                moveDirection *= speed;

                {
                    moveDirection.y = jumpSpeed;
                    anim.Play("jump");
                }
                moveDirection.y = jumpSpeed;

                moveDirection.y -= gravity * Time.deltaTime;
                controller.Move(moveDirection * Time.deltaTime);
            }
```

Concluzii

Crearea acestei aplicației pentru mine insă a fost o mare placere deoarece eu am acumulat o experiența mare dar și am creat o demo joaca arcadă destul de interesantă și frumoasă pe care sper să finizezi ,utilizând principiile programării orientate pe obiecte are o serie de avantaje importante, atîi și pentru lumea jocurilor .Am avut o mare plăcucere să lucrez cu animație și adăugarea texturilor la personaj Aceste principii permit o proiectare mai interesantă . În ceea ce privește proiectarea, se facilitează descompunerea problemelor complexe în subprobleme simple, care pot fi ușor modelate cu ajutorul obiectelor Efectuând acest proiect de curs am studiat un Editor nou, sub numele de Unity3D Editor care m-a ajutat la crearea unui joc 3D. Unity permite crearea unei applicație cu utilizarea funcților care sunt deja importate in Unity3D gata de folosire pe parcursul elaborării ce ne convin , putem implementa funcționalitățile personale. Pe lângă faptul că oferă un șir vast de funcționalități , mai are și o documentație bună și ușoară pentru începătorii în industria dată. În documentația dată se poate de găsit răspunsuri la majoritatea întrebărilor iar dacă acestea nu sunt suficientemai sunt și exemple oferite de către dezvoltatorii acestui .Sunt foarte multe tutoriale pe internet și chiar un site unde Unit3d raspunde aproape la toate intrebările necesare pentru a creia un joc de tipul acesta ,Unity este foarte ușor de folosit și este chiar genial , el convine pentru crearea jocurilor pe care le imaginezi doar că stăruință și răbdare.

Anexa A Coduri sursă

A.1 Action

```
using UnityEngine;
using System.Collections;
using UnityEngine.EventSystems;
using System;

[RequireComponent(typeof(Rigidbody))]
public class Action : MonoBehaviour {

    public VirtualJoystick moveJoystick;
    public joystickJump jumpJoystick;

    public float jumpSpeed = 5f;
    public float movementSpeed = 5.0f;
    public float clockwise = 1000.0f;
    public float counterClockwise = -500.0f;
    public float speed = 6.0F;
    public float JumpSpeed = 100.0f;
    public float gravity = 20.0F;
    private Vector3 moveDirection ;
    public GameObject Player;
        //-----------------
    public AnimationClip idle;
    public Animator anim;
    public CharacterController controller;
    public Rigidbody rb;
    public bool hit = false;
    public Vector3 InputDirection { set; get; }
    public Vector3 col { set; get; }
    bool up;
    bool left;
    bool right;
    void Start()
    {

        controller = GetComponent<CharacterController>();
        anim = GetComponent<Animator>();
        rb = GetComponent<Rigidbody>();
    }
    public void moving_foward()
    {
        anim.Play("run");
        transform.position += transform.forward * Time.deltaTime * movementSpeed;
    }

    void Update()
    {

        Vector3 col = Vector3.forward;
        Vector3 dir = Vector3.zero;
        dir.x = Input.GetAxis("Horizontal");
        dir.z = Input.GetAxis("Vertical");
        if (dir.magnitude > 1)
            dir.Normalize();


        if (moveJoystick.InputDirection == Vector3.Cross(Vector3.up, Vector3.forward))
        {
            anim.Play("run");
```

```
            transform.position += transform.forward * Time.deltaTime * movementSpeed;
            dir = moveJoystick.InputDirection;

        }

        if (moveJoystick.InputDirection != Vector3.zero)
        {
            CharacterController controller = GetComponent<CharacterController>();
            if (controller.isGrounded)
            {
                moveDirection = new Vector3(Input.GetAxis("Horizontal"), 0,
Input.GetAxis("Vertical"));
                moveDirection = transform.TransformDirection(moveDirection);
                moveDirection *= speed;

                {
                    moveDirection.y = jumpSpeed;
                    anim.Play("jump");
                }
                moveDirection.y = jumpSpeed;

                moveDirection.y -= gravity * Time.deltaTime;
                controller.Move(moveDirection * Time.deltaTime);
            }



        }
        if (moveJoystick.InputDirection == Vector3.zero)
        {

        }
        if (moveJoystick.InputDirection == new Vector3(-0.6f, 0, 0.5f))
        {
            Debug.Log("left");
        }
        if (moveJoystick.InputDirection == Vector3.right)
        {
            Debug.Log("right");
        }
        if (moveJoystick.InputDirection == Vector3.back)
        {
            Debug.Log("back");
        }


        {
            CharacterController controller = GetComponent<CharacterController>();
            if (controller.isGrounded)
            {
                moveDirection = new Vector3(Input.GetAxis("Horizontal"), 0,
Input.GetAxis("Vertical"));
                moveDirection = transform.TransformDirection(moveDirection);
                moveDirection *= speed;
                if (Input.GetKeyDown(KeyCode.Space))
                {
                    moveDirection.y = jumpSpeed;
                    anim.Play("jump");
                }
            }

            if (Input.GetKeyUp(KeyCode.Space))
                moveDirection.y = jumpSpeed;
```

```
            moveDirection.y -= gravity * Time.deltaTime;
            controller.Move(moveDirection * Time.deltaTime);
        }
        //-------------------------------------------------------------------------------------
---------------------------

        if(up==true)
        {
            anim.Play("run");
            transform.position += transform.forward * Time.deltaTime * movementSpeed;
        }
        if(left==true)
        {
            anim.Play("Armature_001Action_001");

            {
                transform.Rotate(0, Time.deltaTime * counterClockwise, 0);
            }

        }
        if(right==true)
        {
            anim.Play("Armature_001Action_001");

            {
                transform.Rotate(0, Time.deltaTime * clockwise, 0);
            }
        }

        //-------------------------------------------------------------------------------------
---------------------------
        if (Input.GetKeyDown(KeyCode.W))
        {
            anim.Play("run");
            transform.position += transform.forward * Time.deltaTime * movementSpeed;
        }
        if (Input.GetKeyUp(KeyCode.W))
        {
            anim.Play("Armature_001Action_001");
            // anim.SetInteger("run", 0);
            transform.position += transform.forward * Time.deltaTime * movementSpeed;

        }
        if (Input.GetKeyDown(KeyCode.A))
        {
            anim.Play("run");
            transform.position += transform.forward * Time.deltaTime * movementSpeed;
        }
        if (Input.GetKeyUp(KeyCode.A))
        {
            anim.Play("Armature_001Action_001");
            transform.position += transform.forward * Time.deltaTime * movementSpeed;
        }

        else if (Input.GetKey(KeyCode.S))
        {
            rb.position += Vector3.back * Time.deltaTime * movementSpeed;
        }

        if (Input.GetKey(KeyCode.E))
        {
            transform.Rotate(0, Time.deltaTime * clockwise, 0);
```

```
        }
        else if (Input.GetKey(KeyCode.Q))
        {
            transform.Rotate(0, Time.deltaTime * counterClockwise, 0);
        }
        else
        {
            // anim.Play("Armature_001Action_001");
        }
    }


    public void OnTriggerEnter(Collider other)
    {
        anim.Play("atk");
        controller.radius = 3f;
        //controller.radius = 4.5f;
        col = Vector3.up;
           // hit = true;
    }
    public void OnCollisionEnter(Collision collision)
    {
    }
    public void OnControllerColliderHit(ControllerColliderHit hit)
    {
        if (col == Vector3.up)
        {
            controller.radius = 2.4f;
            if (hit.gameObject.tag == "Monster")
            {
               // anim.Play("atk");
                Destroy(hit.gameObject);
            }
        }
        col = Vector3.zero;

    }
    public void jump_up()
    {
        CharacterController controller = GetComponent<CharacterController>();
        if (controller.isGrounded == true)
            print("eqweq");
        {
            moveDirection = new Vector3(Input.GetAxis("Horizontal"), 0,
Input.GetAxis("Vertical"));
            moveDirection = transform.TransformDirection(moveDirection);
            moveDirection *= speed;

            {
                moveDirection.y = jumpSpeed;
                anim.Play("jump");
            }
        }
    }
    public void jump_down()
    {
        if (controller.isGrounded)
        {
            moveDirection.y = jumpSpeed;

            moveDirection.y -= gravity * Time.deltaTime;
            controller.Move(moveDirection * Time.deltaTime);
```

```
        }
        anim.Play("jump");
    }
    public void _run()
    {
        up = true;
    }
    public void _offrun()
    {
        up = false;
        anim.Play("Armature_001Action_001");
    }
    public void _turnleft()
    {
        left = true;
    }
    public void _offturnleft()
    {
        left = false;
    }
    public void _turnright()
    {
        right = true;
    }
    public void _offturnright()
    {
        right = false;
    }
```

} **A.2 TimerLeft**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class _TimeLeft : MonoBehaviour {
    public bool finish=false;
    public float timeleft = 100;

    public Text Timer;

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

        timeleft -= Time.deltaTime;
        Timer.text = "Time Left :" + Mathf.Round(timeleft);
        if(timeleft<0)
        {
            Application.LoadLevel("GameOver");
        }

       }
    public void finished()
    {
        if(finish==true)
        {

        }

    }
    private void OnControllerColliderHit(ControllerColliderHit hit)
```

```
    {
        if (hit.gameObject.tag == "Finish")
        {
            finish = true;
            // finish = true;
            Timer.color = Color.blue;
            Timer.fontSize = 30;
            timeleft = 100;
        }
    }
}
```

A.3 ColliderHit

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.EventSystems;

public class ColliderHit : MonoBehaviour {
    public AnimationClip idle;
    public Animator anim;
    public CharacterController controller;
    public Rigidbody rb;
    // Use this for initialization
    void Start () {
        controller = GetComponent<CharacterController>();
        anim = GetComponent<Animator>();
        rb = GetComponent<Rigidbody>();
    }

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

       }
    public void OnControllerColliderHit(ControllerColliderHit hit)
    {
        {
            anim.Play("atk");
            {
                controller.radius = 4.5f;

                if (hit.gameObject.tag == "Monster")
                {
                    Destroy(hit.gameObject);
                }
                controller.radius = 2.4f;
            }
        }
    }
    public void OnCollisionEnter(Collision hit)
    {
        anim.Play("atk");
        {
            controller.radius = 4.5f;

            if (hit.gameObject.tag == "Monster")
            {
                Destroy(hit.gameObject);
            }
            controller.radius = 2.4f;
        }
```

```
        Destroy(gameObject);
    }
    public void OnTriggerEnter(Collider other)
    {
        if (other.gameObject.tag == "Monster")
        {
            anim.Play("atk");
            Destroy(other.gameObject);
        }
    }
    //public void OnPointerDown(PointerEventData eventData)
    //{
    //    throw new NotImplementedException();
    //}
}
```

A.4 Finish

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;


public class Finish : MonoBehaviour
{

    public Texture HH;
    public Texture HH1;
    public Texture2D texture;
    public Texture RedHP;
    public Texture2D image;
    public Texture background;
    public Texture texbutton;
    GameObject cube;

    public GUITexture img;

    public int lives;
    public int mango;
    GameObject Player;
    public GameObject imageeat;
    public GUISkin skin1;
    public GUIStyle config;
    public GUIStyle gui;
    Material material;
    bool finish;
    bool finish2;
    public float total_mango;
    public float m1;
    public bool f1=false;
    public bool level2open = false;

    // GameObject Redhp;
    // Use this for initialization
    void Start()
    {
        Player = GameObject.Find("Player");
        HH = (Texture)Resources.Load("pucinFace");
        HH1 = (Texture)Resources.Load("mango");
        texbutton = (Texture)Resources.Load("win");
        total_mango = PlayerPrefs.GetFloat("mango", (total_mango+ mango));
```

```
        onLoad();
    }

    private void OnControllerColliderHit(ControllerColliderHit hit)
    {
        if (hit.gameObject.tag == "travka" && lives < 1)
        {
            Destroy(hit.gameObject);
            SceneManager.LoadScene("GameOver");
        }
        else if (hit.gameObject.tag == "travka")
        {
            Destroy(hit.gameObject);
            lives--;
        }
        else if (hit.gameObject.tag == "live")
        {
            Destroy(hit.gameObject);
            lives++;
        }
        else if (hit.gameObject.tag == "pizdets")
        {
            lives--;
            Player.transform.position = new Vector3(0, 0, 0);

            if (lives < 1)
            { SceneManager.LoadScene("GameOver"); }



        }
        if (hit.gameObject.tag == "meat")
        {
            Destroy(hit.gameObject);
            total_mango++;
        }
        if (hit.gameObject.tag == "Finish")
        {
            level2open = true;
            Time.timeScale = 0f;
            finish = true;
            f1 = true;
            PlayerPrefs.SetFloat("m", total_mango);
        }
        if (hit.gameObject.tag == "Finish2")
        {
            Time.timeScale = 0f;
            finish2 = true;
            PlayerPrefs.SetFloat("m", total_mango + mango);//6
        }

    }

    public void OnGUI()
    {

        GUILayout.BeginArea(new Rect(Screen.width * 0.11f, 20, Screen.width, Screen.height));
        GUILayout.Label("" + lives, config);
        GUILayout.EndArea();

        GUI.DrawTexture(new Rect(Screen.width * -0.15f, Screen.height * -0.1f, Screen.width /
2.5f, Screen.height / 2.5f), HH);
```

```
        GUI.DrawTexture(new Rect(Screen.width * 0.82f, 0, Screen.width / 6, Screen.height / 6),
HH1);

        GUILayout.BeginArea(new Rect(Screen.width * 0.95F, 20, Screen.width, Screen.height));
        GUILayout.Label( ""+ (total_mango+mango), config);
        GUILayout.EndArea();


        if (finish == true)
        {

            GUI.Label(new Rect(-300, -300, Screen.width, Screen.height), background, gui);
            // GUI.TextArea(new Rect(Screen.width * 0.4f, Screen.height * 0.4f, Screen.width / 3,
Screen.height / 3), "YOU WIN ");
            GUI.Label(new Rect(Screen.width * 0.2f, Screen.height * 0.2f, Screen.width / 2.4f,
Screen.height / 2.4f), HH);
            GUI.Label(new Rect(Screen.width * 0.3f, Screen.height * 0.5f, Screen.width / 4,
Screen.height / 4), HH1);
            GUI.Label(new Rect(Screen.width * 0.48f, Screen.height * 0.31f, Screen.width / 2.4f,
Screen.height / 2.4f), "" + lives, config);
            GUI.Label(new Rect(Screen.width * 0.48f, Screen.height * 0.54f, Screen.width / 2.4f,
Screen.height / 2.4f), "" + mango, config);
            if (GUI.Button(new Rect(Screen.width * 0.7f, Screen.height * 0.6f, Screen.width / 3,
Screen.height / 3), texbutton, config))
            {
                SceneManager.LoadScene("Game2");
            }
        }
        if (finish2 == true)
        {
            GUI.Label(new Rect(0, 0, Screen.width, Screen.height), background, gui);
            // GUI.TextArea(new Rect(Screen.width * 0.4f, Screen.height * 0.4f, Screen.width / 3,
Screen.height / 3), "YOU WIN ");
            GUI.Label(new Rect(Screen.width * 0.2f, Screen.height * 0.2f, Screen.width / 2.4f,
Screen.height / 2.4f), HH);
            GUI.Label(new Rect(Screen.width * 0.3f, Screen.height * 0.5f, Screen.width / 4,
Screen.height / 4), HH1);
            GUI.Label(new Rect(Screen.width * 0.48f, Screen.height * 0.31f, Screen.width / 2.4f,
Screen.height / 2.4f), "" + lives, config);
            GUI.Label(new Rect(Screen.width * 0.48f, Screen.height * 0.54f, Screen.width / 2.4f,
Screen.height / 2.4f), "" + total_mango, config);
            if (GUI.Button(new Rect(Screen.width * 0.7f, Screen.height * 0.6f, Screen.width / 3,
Screen.height / 3), texbutton, config))
            {
                SceneManager.LoadScene("Game3");
            }
        }
    }
    void onSave()
    {
        // PlayerPrefs.SetString("", Player);
        PlayerPrefs.SetFloat("mango",total_mango+total_mango);
        PlayerPrefs.SetInt("lives", lives);
    }
    void onLoad()
    {
        PlayerPrefs.GetInt("lives", lives);
        PlayerPrefs.GetFloat("mango", total_mango + total_mango);
    }



}
```

A.5 GameOver

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;


public class GameOver : MonoBehaviour
{

    GameObject pausemenu;

    [SerializeField]
    public Texture textureoption;
    public Texture teximage;
    public Texture texresume;
    public Texture texrestart;
    public Texture texmainmenu;
    public Texture texquit;
    public GUIStyle settings;
    public GUIStyle transparent;
    bool view = false;
    bool ok;

    void Start()
    {

        textureoption = (Texture)Resources.Load("RedHP");
        teximage = (Texture)Resources.Load(" ");
        texresume = (Texture)Resources.Load("res");
        Time.timeScale = 1f;

    }


     void Update()
    {

    }


    public void OnGUI()
    {
       // view = true;

        if (GUI.Button(new Rect(Screen.width * 0.9f, Screen.height * 0.2f, Screen.width / 10,
Screen.height / 10), textureoption,transparent))
        {
            view = !view;

        }

        if (view == true)
        {
            Time.timeScale = 0f;
            GUI.Label(new Rect(0, 0, Screen.width, Screen.height + 300), teximage, settings);
            if (GUI.Button(new Rect(Screen.width * 0.3f, Screen.height * 0.2f, Screen.width /
2.5f, Screen.height / 7), texresume))
```

```
            {
                Time.timeScale = 1;
                view = false;
                ok = false;

            }
            if (GUI.Button(new Rect(Screen.width * 0.3f, Screen.height * 0.35f, Screen.width /
2.5f, Screen.height / 7), textureoption))
            {
                SceneManager.LoadScene("Game1");
            }
            if (GUI.Button(new Rect(Screen.width * 0.3f, Screen.height * 0.5f, Screen.width /
2.5f, Screen.height / 7), textureoption))
            {
                SceneManager.LoadScene("UnitOne");

            }
            if (GUI.Button(new Rect(Screen.width * 0.3f, Screen.height * 0.65f, Screen.width /
2.5f, Screen.height / 7), textureoption))
            {
                Application.Quit();
            }
        }
    }




}
```

A.6 JoystickJump

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;
using System;
[RequireComponent(typeof(Rigidbody))]
public class joystickJump : MonoBehaviour, IDragHandler, IPointerUpHandler, IPointerDownHandler
{

    private Image bgImg;
    private Image joystickImg;
    public GameObject Player;

    public Vector3 InputDirection { set; get; }
    public Animator anim;
    public CharacterController controller;
    public Rigidbody rb;
    public Action act;
    public bool active;
    private void Start()
    {


        controller = GetComponent<CharacterController>();
        anim = GetComponent<Animator>();
```

```
        rb = GetComponent<Rigidbody>();

        bgImg = GetComponent<Image>();
        joystickImg = GetComponent<Image>();
        InputDirection = Vector3.zero;
    }

    public virtual void OnDrag(PointerEventData ped)
    {
        //Vector2 pos = Vector2.zero;
        //if (RectTransformUtility.ScreenPointToLocalPointInRectangle
        //    (bgImg.rectTransform,
        //    ped.position,
        //    ped.pressEventCamera,
        //    out pos))
        //{
        //    pos.x = (pos.x / bgImg.rectTransform.sizeDelta.x);
        //    pos.y = (pos.y / bgImg.rectTransform.sizeDelta.y);
        //    float x = (bgImg.rectTransform.pivot.x == 1) ? pos.x * 2 + 1 : pos.x * 2 - 1;
        //    float y = (bgImg.rectTransform.pivot.y == 1) ? pos.y * 2 + 1 : pos.y * 2 - 1;
        //    InputDirection = new Vector3(x, 0, y);
        //    InputDirection = (InputDirection.magnitude > 1) ? InputDirection.normalized :
InputDirection;

        //    joystickImg.rectTransform.anchoredPosition = new Vector3(InputDirection.x *
(bgImg.rectTransform.sizeDelta.x / 3), InputDirection.z * (bgImg.rectTransform.sizeDelta.y / 3));
        //    Debug.Log(InputDirection);
        //}
    }
    public virtual void onPointerDown(PointerEventData ped)
    {
        Debug.Log("PointerDown");
    }
    public virtual void onPointerUp(PointerEventData ped)
    {

    }

    public void OnPointerUp(PointerEventData ped)
    {
        //InputDirection = Vector3.zero;
        //joystickImg.rectTransform.anchoredPosition = Vector3.zero;
        //active = false;
        Debug.Log("dasd");

    }

    public void OnPointerDown(PointerEventData ped)
    {
        //active = true;
        //InputDirection = Vector3.zero;
        //joystickImg.rectTransform.anchoredPosition = Vector3.zero;
        //Debug.Log("ok");
        Vector2 pos = Vector2.zero;
        if (RectTransformUtility.ScreenPointToLocalPointInRectangle
            (bgImg.rectTransform,
            ped.position,
            ped.pressEventCamera,
            out pos))
        {
            pos.x = (pos.x / bgImg.rectTransform.sizeDelta.x);
            pos.y = (pos.y / bgImg.rectTransform.sizeDelta.y);
            float x = (bgImg.rectTransform.pivot.x == 1) ? pos.x * 2 + 1 : pos.x * 2 - 1;
```

```
            float y = (bgImg.rectTransform.pivot.y == 1) ? pos.y * 2 + 1 : pos.y * 2 - 1;
            InputDirection = new Vector3(x, 0, y);
            InputDirection = (InputDirection.magnitude > 1) ? InputDirection.normalized :
InputDirection;

            joystickImg.rectTransform.anchoredPosition = new Vector3(InputDirection.x *
(bgImg.rectTransform.sizeDelta.x / 3), InputDirection.z * (bgImg.rectTransform.sizeDelta.y / 3));
            Debug.Log(InputDirection);
        }
    }
}
```

A.7 Life

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;


public class Life : MonoBehaviour {

    public Texture HH;
    public Texture HH1;
    public Texture2D texture;
    public Texture RedHP;
    public Texture2D image;
    public Texture background;
    public Texture texbutton;
    GameObject cube;

   public GUITexture img;

    public int lives;
    public int mango;
    GameObject Player;
    public GameObject imageeat;
    public GUISkin skin1;
    public GUIStyle config;
    public GUIStyle gui;
   // public LightmapData map;
    Material material;
    bool finish;
    bool finish2;
    // GameObject Redhp;
    // Use this for initialization
    void Start() {
        Player = GameObject.Find("Player");
        HH = (Texture)Resources.Load("pucinFace");
        HH1 = (Texture)Resources.Load("mango");
        texbutton = (Texture)Resources.Load("win");
        cube = (GameObject)Resources.Load("mang");

       // PlayerPrefs.SetFloat("mango", mango);


        // Create a texture. Texture size does not matter, since
        // LoadImage will replace with with incoming image size.


        // GetComponent<Renderer>().material.mainTexture = tex;
```

```
}

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

}
void ss()
{
    GameObject cube = GameObject.CreatePrimitive(PrimitiveType.Cube);
    GameObject cube1 = GameObject.CreatePrimitive(PrimitiveType.Cube);


}
private void OnControllerColliderHit(ControllerColliderHit hit)
{
    if (hit.gameObject.tag == "travka" && lives < 1)
    {
        Destroy(hit.gameObject);
        SceneManager.LoadScene("GameOver");
    }
    else if (hit.gameObject.tag == "travka")
       {
        Destroy(hit.gameObject);
        lives--; }
   else if(hit.gameObject.tag== "live")
    {
        Destroy(hit.gameObject);
        lives++;
    }
    else if(hit.gameObject.tag == "pizdets"  )
    {
        lives--;
        Player.transform.position = new Vector3(-0, 0, 0);

     if (lives<1)
        { SceneManager.LoadScene("GameOver"); }



    }
    if (hit.gameObject.tag == "meat")
    {
        Destroy(hit.gameObject);
        //cube= (GameObject)Instantiate(cube);
        mango++;
    }
    if(hit.gameObject.tag=="Finish")
    {
        Time.timeScale = 0f;
        finish = true;
        PlayerPrefs.SetFloat("mango", mango);
    }
    if (hit.gameObject.tag == "Finish2")
    {
        Time.timeScale = 0f;
        finish2 = true;
        PlayerPrefs.SetFloat("mango", mango);
    }

}
```

```
    void onSave()
    {
       // PlayerPrefs.SetString("", Player);
        PlayerPrefs.SetFloat("mango", mango);
        PlayerPrefs.SetInt("lives", lives);
    }
    void onLoad()
    {
        PlayerPrefs.GetInt("lives", lives);
    }


}
```

A.8 MovingCanvas

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using UnityEngine.EventSystems;
using System;

public class MovingCanvas : MonoBehaviour
{


    public float jumpSpeed = 11f;
    public float movementSpeed = 5.0f;
    public float clockwise = 300.0f;
    public float counterClockwise = -300.0f;
    public float speed = 7.0F;
    public float gravity = 15.0F;
    public GUIStyle customButton;
    public GUIStyle up;

    public float speedx = 5;
    GameObject movingcan;
    Texture textureleft;
    Texture textureup;
    Texture textureright;
    Texture texturedown;
    Texture textureatk;
    Texture texturejmp;
    Texture textureoption;
    Animator anim;
    public Rigidbody rb;
    Touch touch;
    private Vector3 moveDirection = Vector3.zero;
    private Vector3 moveDirection1;
    bool buttonClicked = false;
    private float holdTime = 0.8f; //or whatever
    private float acumTime = 0;
    public float JumpSpeed = 100.0f;
    public bool hold;

    GameObject w;
    public CharacterController controller;
```

```
    // Use this for initialization
    void Start() {
        anim = GetComponent<Animator>();
        rb = GetComponent<Rigidbody>();
        controller = GetComponent<CharacterController>();
        movingcan = GameObject.Find("movingcan");
        textureleft = (Texture)Resources.Load("leftlapka");
        textureup = (Texture)Resources.Load("uplapka");
        textureright = (Texture)Resources.Load("rightlapka");
        texturedown = (Texture)Resources.Load("downlapka");
        textureatk = (Texture)Resources.Load("butatk");
        texturejmp = (Texture)Resources.Load("butjmp");
        textureoption = (Texture)Resources.Load("image2");

        w = GameObject.Find("w");



    }

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


    }



        if (GUI.RepeatButton(new Rect(Screen.width * 0.1f, Screen.height * 0.55f, Screen.width /
7.5f, Screen.height / 7.5f), textureup, customButton))
        {

            anim.Play("run");
            transform.position += transform.forward * Time.deltaTime * movementSpeed;
        }
        else if (Event.current.type == EventType.Repaint)
        {
            anim.Play("Armature_001Action_001");

        }






        if (GUI.RepeatButton(new Rect(Screen.width * 0.03f, Screen.height * 0.7f, Screen.width /
7.5f, Screen.height / 7.5f), textureleft, customButton))
        {
            anim.Play("Armature_001Action_001");
            transform.Rotate(0, Time.deltaTime * counterClockwise, 0);
        }

        if (GUI.RepeatButton(new Rect(Screen.width * 0.17f, Screen.height * 0.7f, Screen.width /
7.5f, Screen.height / 7.5f), textureright, customButton))
        {

            anim.Play("Armature_001Action_001");
```

```
            transform.Rotate(0, Time.deltaTime * clockwise, 0);
        }

        }




}
```

A.9 Score

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class Score : MonoBehaviour {
    public int mango;
    public GUIStyle config;
    public Texture background;
    public GUIStyle gui;
    public Texture HH;
    public Texture HH1;
    public Texture2D texture;
    public Texture RedHP;
    public Texture2D image;
    GameObject cube;
    GameObject Player;
    bool finish;
    bool finish2;
    GameObject SaveScore;
    public int lives;


    public Texture texbutton;
    // Use this for initialization
    void Start () {
        Player = GameObject.Find("Player");
        HH = (Texture)Resources.Load("pucinFace");
        HH1 = (Texture)Resources.Load("mango");
        texbutton = (Texture)Resources.Load("win");
        cube = (GameObject)Resources.Load("mang");
        SaveScore = GameObject.Find("SaveScore");
        onLoad();
        Time.timeScale = 1f;
    }

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

    }

    private void OnControllerColliderHit(ControllerColliderHit hit)
    {
        if(hit.gameObject.tag =="live")
        {
            Destroy(hit.gameObject);
```

```
            lives++;
        }
        if (hit.gameObject.tag == "mango")
        {
            Destroy(hit.gameObject);
            //cube= (GameObject)Instantiate(cube);
            mango++;
        }
        if (hit.gameObject.tag == "meat")
        {
            Destroy(hit.gameObject);
            //cube= (GameObject)Instantiate(cube);
            mango++;
        }
        if (hit.gameObject.tag == "Finish")
        {
            Time.timeScale = 1f;
            finish = true;
           // PlayerPrefs.SetFloat("mango", mango);
        }
        if (hit.gameObject.tag == "Finish2")
        {
            Time.timeScale = 1f;
            finish2 = true;
            //PlayerPrefs.SetFloat("mango", mango);
        }
    }
    public void Awake()
    {
        DontDestroyOnLoad(transform.gameObject);
    }
    public void onLoad()
    {
        PlayerPrefs.GetFloat("mango", mango);
    }
    public void OnGUI()
    {

        GUILayout.BeginArea(new Rect(Screen.width * 0.11f, 20, Screen.width, Screen.height));
        GUILayout.Label("" + lives,config);
        GUILayout.EndArea();

        GUI.DrawTexture(new Rect(Screen.width * -0.15f, Screen.height * -0.1f, Screen.width /
2.5f, Screen.height / 2.5f), HH);
        GUI.DrawTexture(new Rect(Screen.width * 0.82f, 0, Screen.width / 6, Screen.height / 6),
HH1);

        GUILayout.BeginArea(new Rect(Screen.width * 0.95F, 20, Screen.width, Screen.height));
        GUILayout.Label(" " +  mango, config);
        GUILayout.EndArea();


        if (finish == true)
        {

            GUI.Label(new Rect(0, 0, Screen.width, Screen.height), background, gui);
            // GUI.TextArea(new Rect(Screen.width * 0.4f, Screen.height * 0.4f, Screen.width / 3,
Screen.height / 3), "YOU WIN ");
            GUI.Label(new Rect(Screen.width * 0.2f, Screen.height * 0.2f, Screen.width / 2.4f,
Screen.height / 2.4f), HH);
            GUI.Label(new Rect(Screen.width * 0.3f, Screen.height * 0.5f, Screen.width / 4,
Screen.height / 4), HH1);
```

```
            GUI.Label(new Rect(Screen.width * 0.48f, Screen.height * 0.54f, Screen.width / 2.4f,
Screen.height / 2.4f), "" + mango, config);
            if (GUI.Button(new Rect(Screen.width * 0.7f, Screen.height * 0.6f, Screen.width / 3,
Screen.height / 3), texbutton, config))
            {
                finish = false;
                SceneManager.LoadScene("Game2");
            }
        }

        if (finish2 == true)
        {

            GUI.Label(new Rect(0, 0, Screen.width, Screen.height), background, gui);
            // GUI.TextArea(new Rect(Screen.width * 0.4f, Screen.height * 0.4f, Screen.width / 3,
Screen.height / 3), "YOU WIN ");
            GUI.Label(new Rect(Screen.width * 0.2f, Screen.height * 0.2f, Screen.width / 2.4f,
Screen.height / 2.4f), HH);
            GUI.Label(new Rect(Screen.width * 0.3f, Screen.height * 0.5f, Screen.width / 4,
Screen.height / 4), HH1);

            GUI.Label(new Rect(Screen.width * 0.48f, Screen.height * 0.54f, Screen.width / 2.4f,
Screen.height / 2.4f), "" + mango, config);
            if (GUI.Button(new Rect(Screen.width * 0.7f, Screen.height * 0.6f, Screen.width / 3,
Screen.height / 3), texbutton, config))
            {
                finish2 = false;
                SceneManager.LoadScene("Game3");
            }
        }

    }
}
```

A.10 Score

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class RotateCamera : MonoBehaviour
{

    public float smooth ;
    public float tiltAngle;
    void Start()
    {

    }
    void Update()
    {
        // Camera.main.transform.position.x = 33;
      //  Vector3 position = new Vector3(Random.Range(-10.0f, 10.0f), 0, Random.Range(-10.0f,
10.0f));
        float tiltAroundZ = Input.GetAxis("Horizontal") * tiltAngle;
        float tiltAroundX = Input.GetAxis("Vertical") * tiltAngle;
        Quaternion target = Quaternion.Euler(tiltAroundX + Random.Range(-5.0f, 5.0f),
Random.Range(-5.0f, 5.0f), tiltAroundZ+ Random.Range(-5.0f, 5.0f));
```

```
        // transform.rotation = Quaternion.Slerp(transform.rotation, target, Time.deltaTime *
smooth);
        //    Camera.main.transform.rotation = Quaternion.Euler(tiltAroundX, 0, tiltAroundZ);
       Camera.main.transform.rotation = Quaternion.Slerp(transform.rotation, target,
Time.deltaTime * smooth);
         transform.Rotate(Vector3.up * Input.GetAxis("Horizontal"));
      //  Camera.main.transform.rotation = Quaternion.Euler(tiltAroundX, 0,
tiltAroundZ*Time.deltaTime);
        //  Camera.main.transform.rotation = Quaternion.Slerp(transform.rotation, target2,
Time.deltaTime * smooth);


        if (tiltAroundZ > 3)
        {
            Quaternion target1 = Quaternion.Euler(tiltAroundX, 0, tiltAroundZ - 5);
            Camera.main.transform.rotation = Quaternion.Slerp(transform.rotation, target1,
Time.deltaTime * smooth);
        }



}
    void psc()
    {

    }


}
```