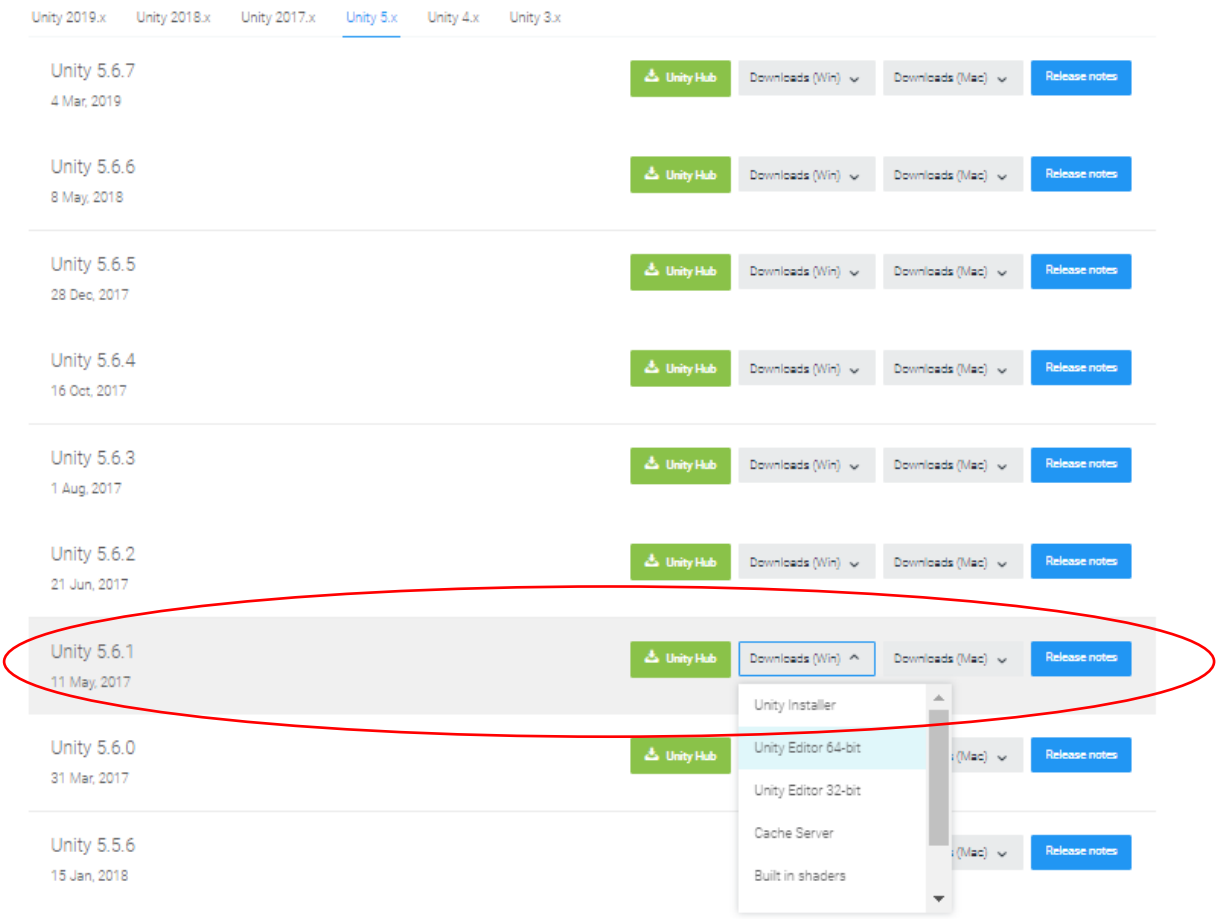
Laborator TM

Crearea unui joc 3D în Unity

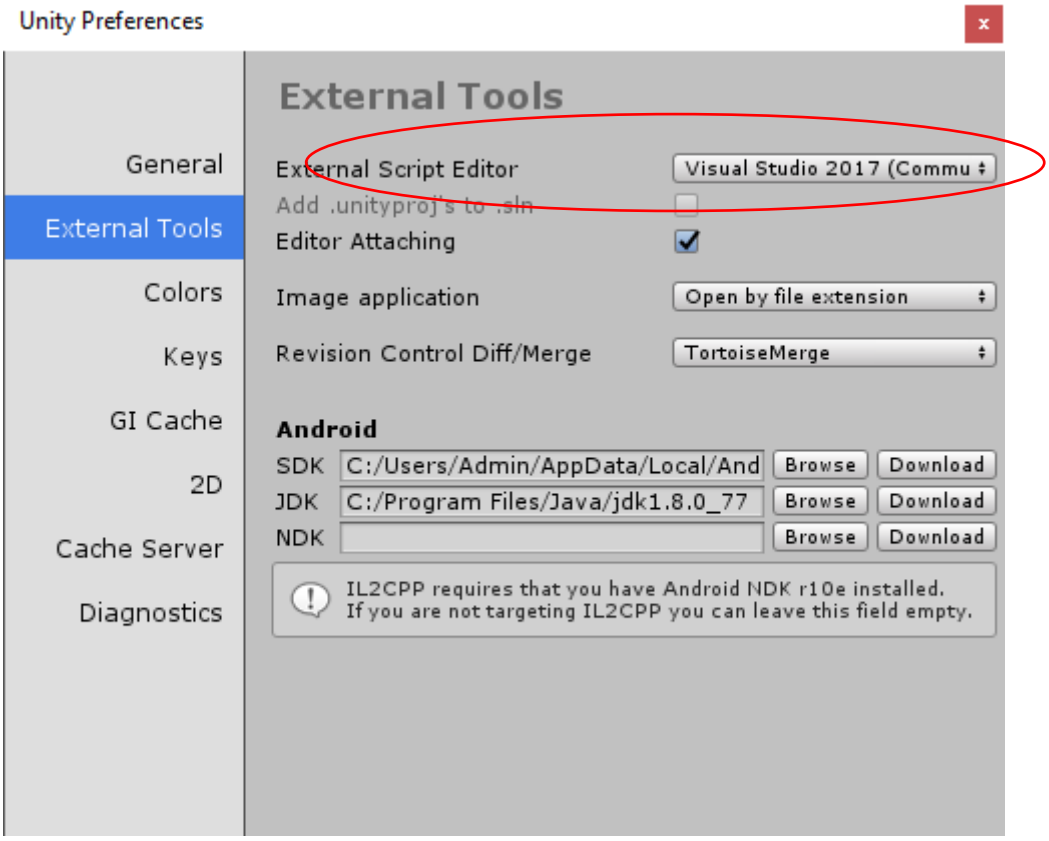
În aces laborator ne propunem să învățăm cum se creează un joc în editorul Unity 3D.

Descărcarea și instalarea editorului Unity.

Unity 5.6.1
https://unity3d.com/get-unity/download/archive?_ga=2.254299848.1121081116.1536051613-1502415783.1484743196



Setarea editorului de scriere a codurilor C# (***Edit – Preferences – External Tools***)



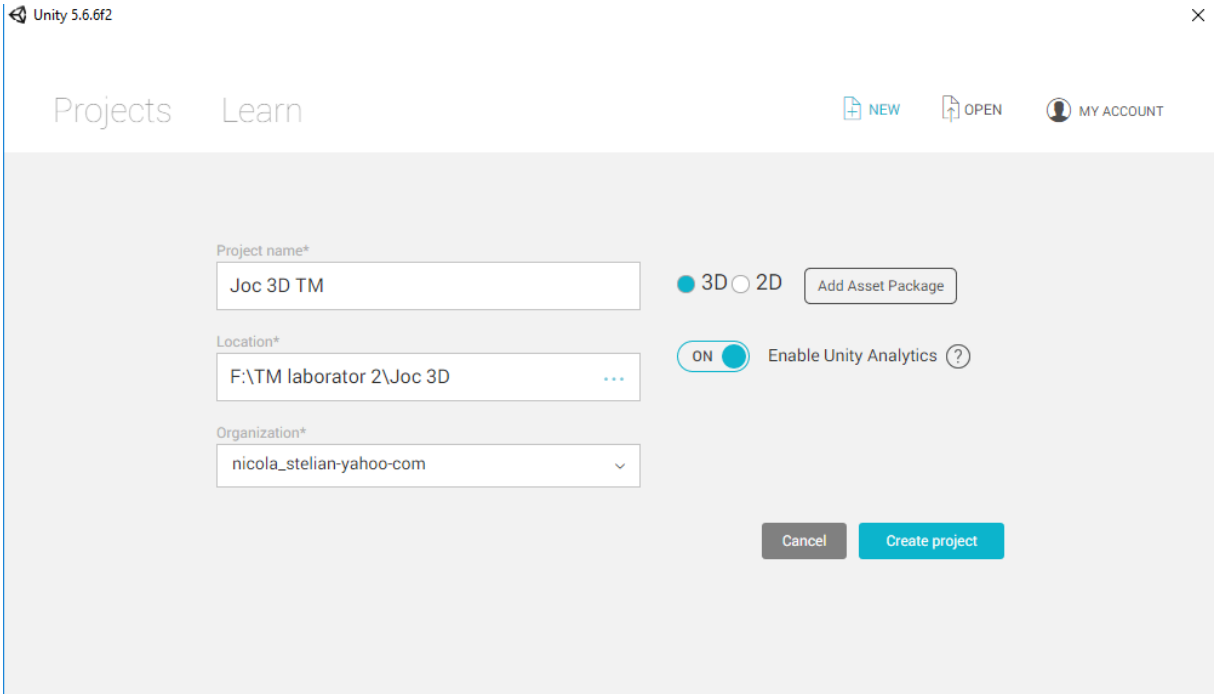
Principalele funcționalități învățate vor fi:

- folosirea proprietăților obiectelor 3D din editorul Unity
- folosirea de obiecte 3D gratis din magazinul online Unity
- importarea obiectelor 3D în Unity
- mișcarea automată a unui caracter pe o axă (x, y, z)
- urmărirea caracterului de către camera care arată sccena
- controlarea caracterului de la tastatură
- crearea dinamică de obstacole 3D și/sau recompense aleatoriu în joc
- activarea unei animații pe un caracter
- evenimente care implică coliziunea dintre două obiecte (scor, terminare joc, etc)
- salvarea scorului într-un fișier text
- cronometrarea jocului
- navigarea prin sccenele jocului (butoane)
- Salvarea jocului într-un fișier executabil pentru PC

Crearea unui nou proiect 3D în Unity

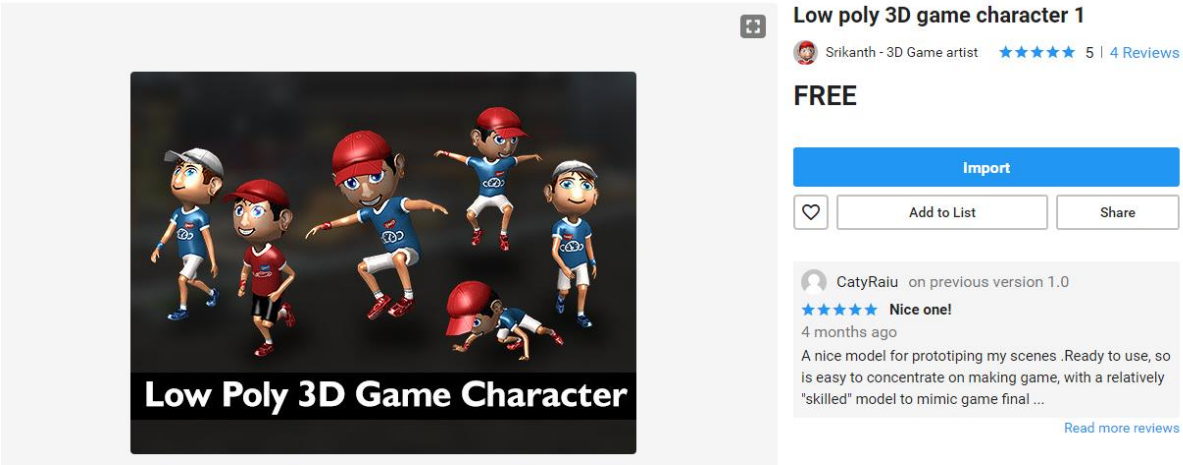
Pentru a crea un nou joc Unity vom parcurge următorii pași:

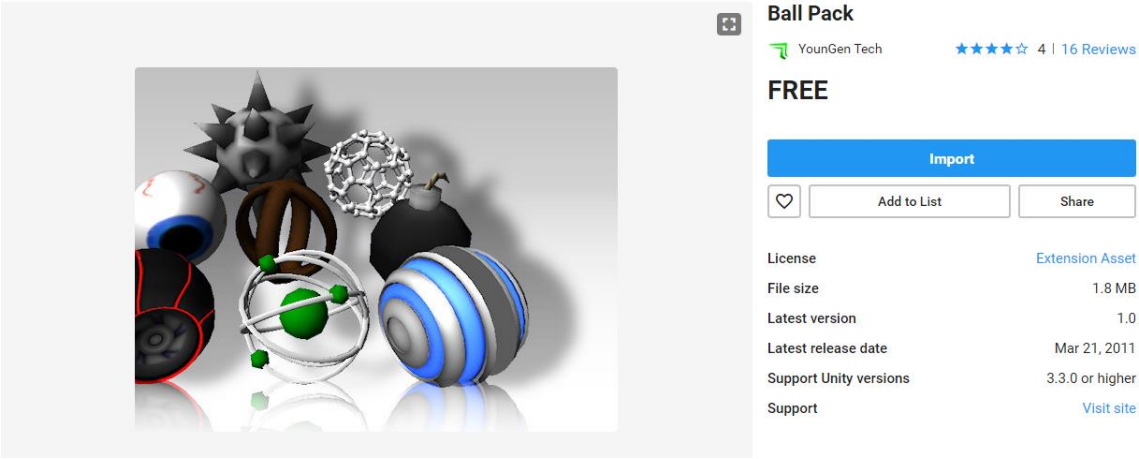
- Vom porni aplicația Unity și vom face setările ca în figura de mai jos și vom crea proiectul



La crearea proiectului se va crea în fișierul acestuia un fișier cu numele **Assets**. În acesta vom crea alte fișiere (ex. fișier cu script-uri c#, fișiere cu obiecte 3D))

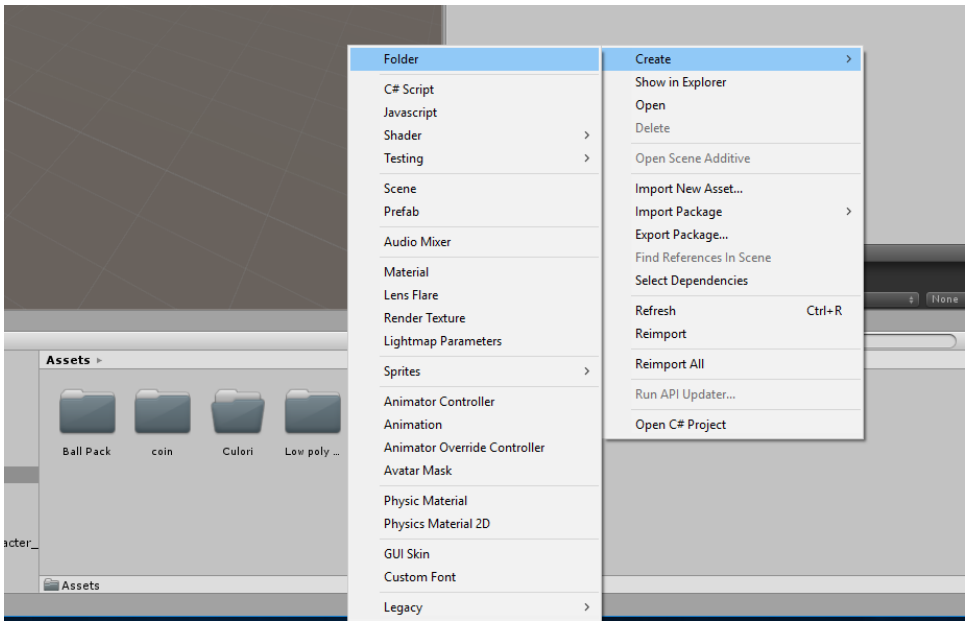
Vom căuta în magazinul online ***(din meniul Unity vom accesa Window – Asset Store***) unity obiectele necesare pentru jocul nostru: *Low poly 3D game character 1 și Ball Pack.*





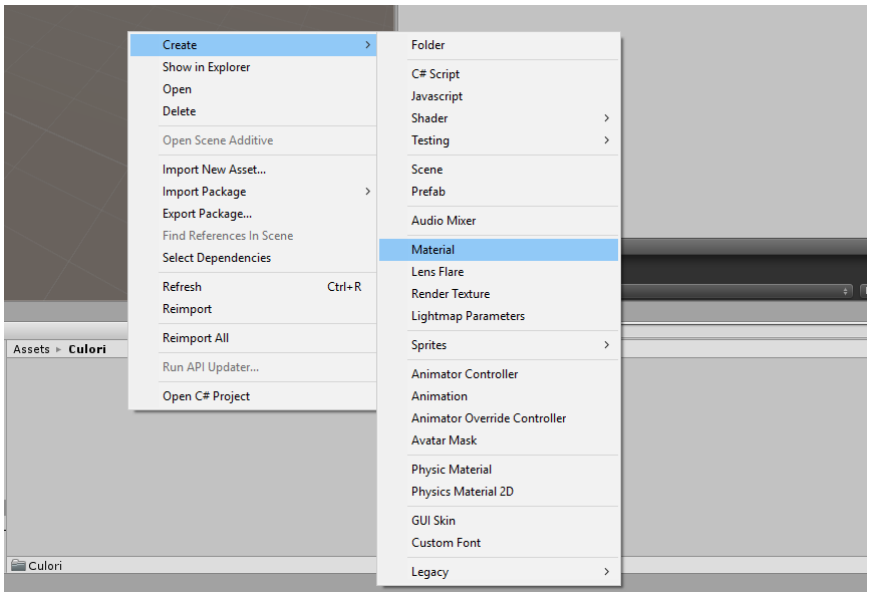
Copiem fișierul pus la dispoziție (***coin***) – la link-ul (https://drive.google.com/drive/folders/1BL_yzKRtajetCi_8hTzO2y41V-xBVHmi?usp=sharing) în fișierul *Assets* care s-a creat când am creat proiectul.

Creem trei fișiere (***Click dreapta în Assets, Create - Folder***) Culori, Script-uri și Salvări în care vom pune culorile și imaginea pentru jocul executabil, script-urile C# folosite, respectiv vom salva scorul într-un fișier text.

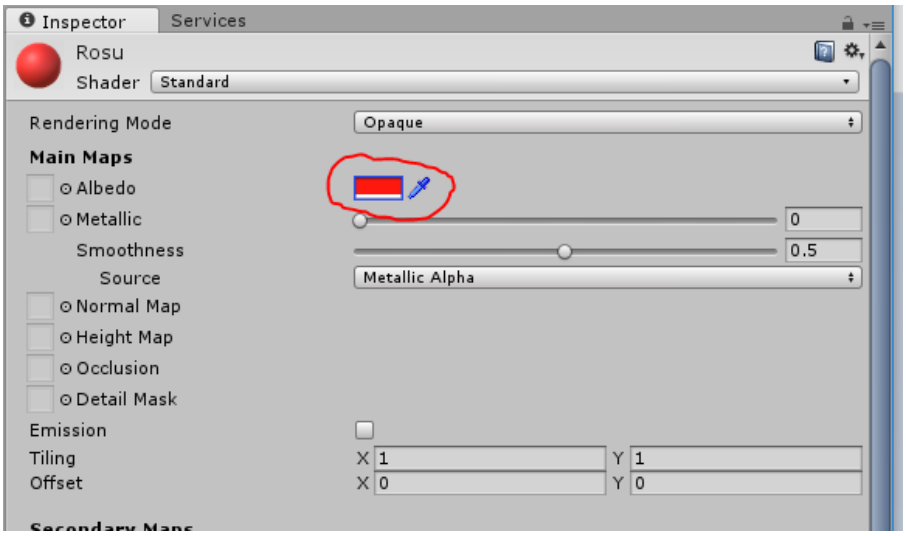


Aceste culori vor fi volosite pentru a le pune peste diferite obiecte din scena noastră. (ex. dacă se dorește ca un obiect din scenă să aibă o anumită culoare, una dintre culorile create va fi pusă peste acel obiect, drag and drop)

În fișierul Culori creem noi culori (***Click dreapta în fișierul Culori – Create - Material***)

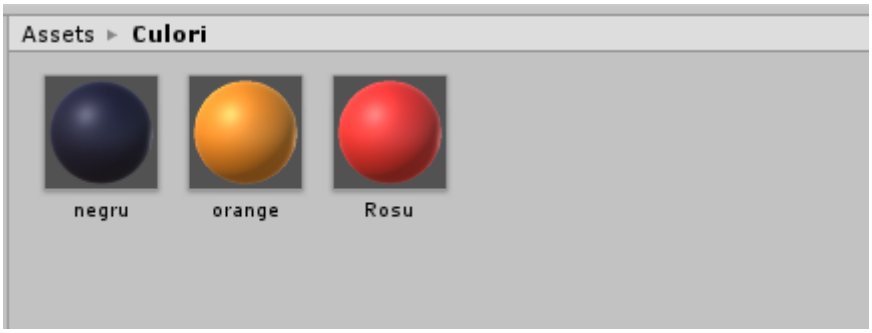


Dăm un nume materialului creat și setăm culoarea din proprietățile acestuia.



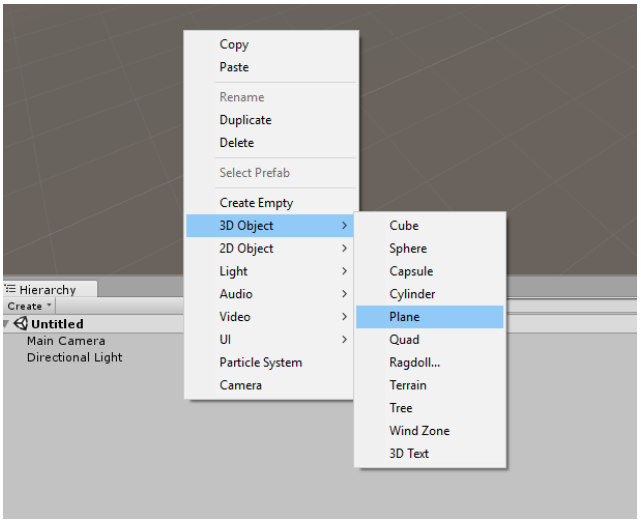
Pentru a pune o culoare pe un obiect 3D trebuie să tragem (drag and drop) culoarea creată peste acesta!

În fișierul culori s-au creat 3 materiale cu 3 culori.

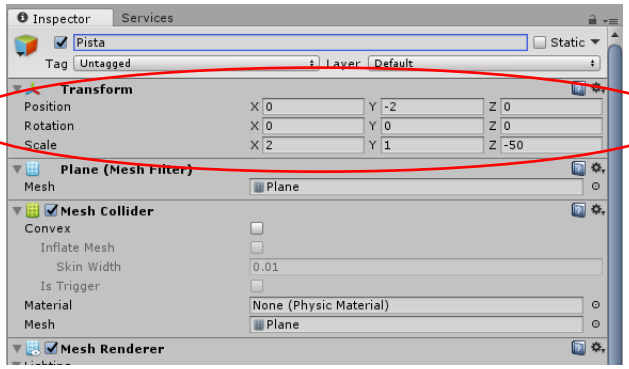


Creem un Plan 3D, cu numele *Pista*, (***Click dreapta – 3D Object – Plane***) în scena noastră.

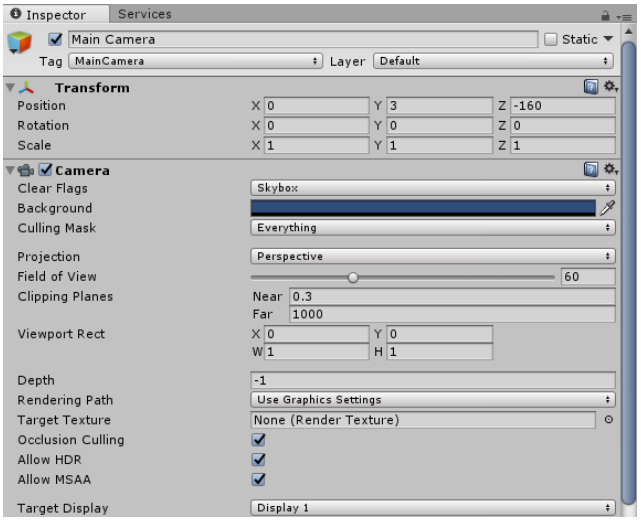
Fiecare obiect folosit în aplicație este descris de proprietăți: poziție, mărime, rotire, textură, etc.



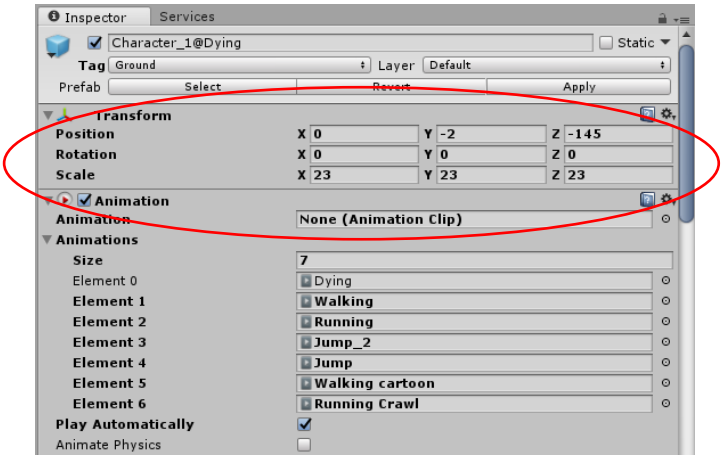
Setăm planul la poziția și scala corespunzătoare, apoi punem culoarea neagră creată în fișierul *Culori* peste acesta.Planul va fi folosit având rolul de pistă, unde caracterul controlat de utilizator va alerga. Poziția și scala acestuia a fost setată în spațiul 3D.



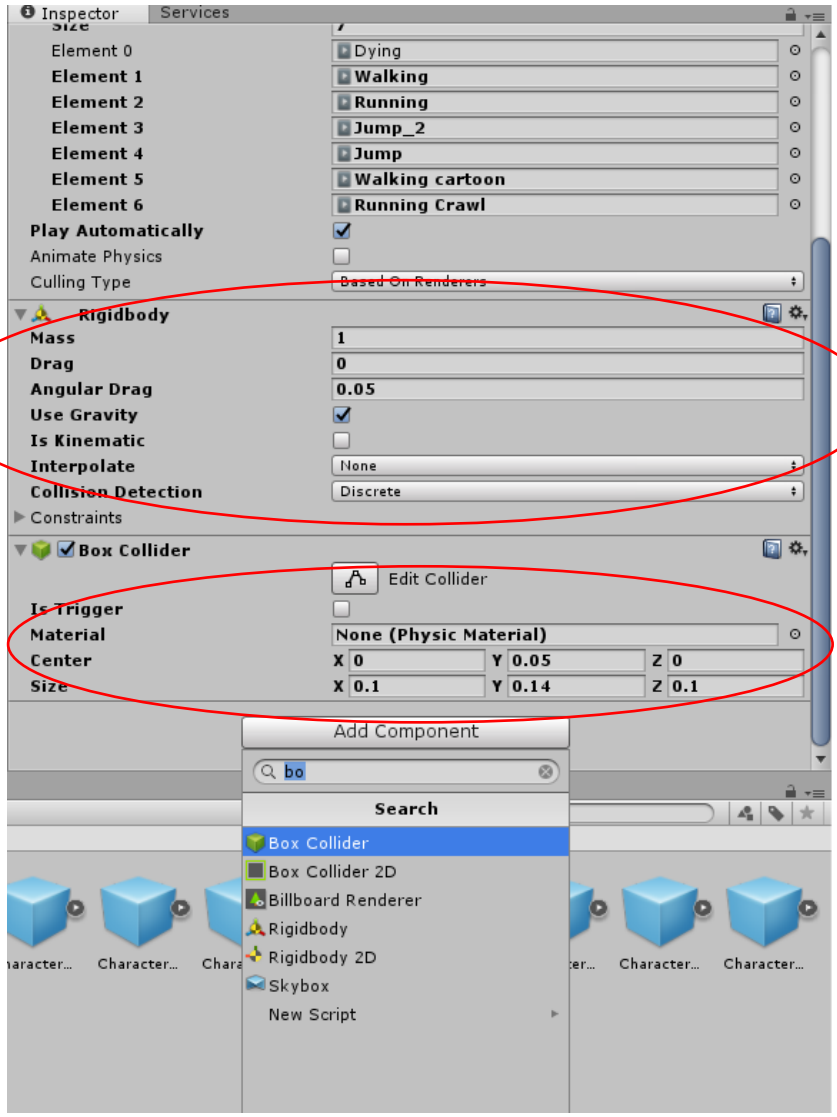
Setăm *Main Camera* la poziția corespunzătoare din proprietățile acesteia. Aceasta are rolul de a afișa sau filma toate obiectele din spțiul 3D care se află în fața acesteia. Camera a fost repoziționată în spațiul 3D (figura de mai jos).



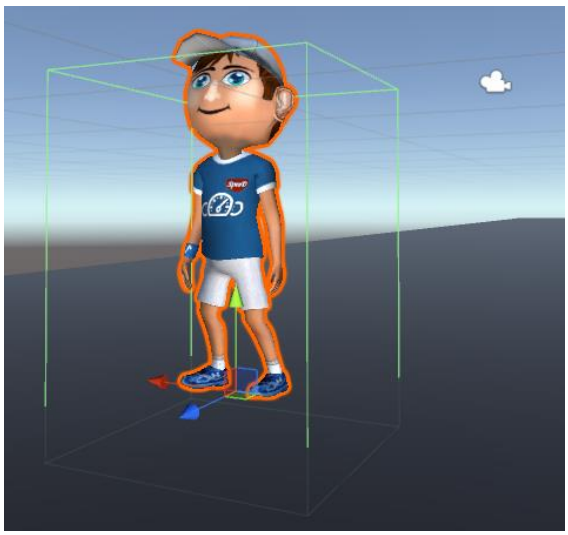
Punem în scenă caracterul pe care dorim să-l folosim din fișierul importat din magazinul online Unity (Fișierul: ***Low poly 3D game character 1 –> Prefabs -> Character_1@Dying***). Setăm caracterul la poziția dorită și animația acestuia o punem pe *None.*



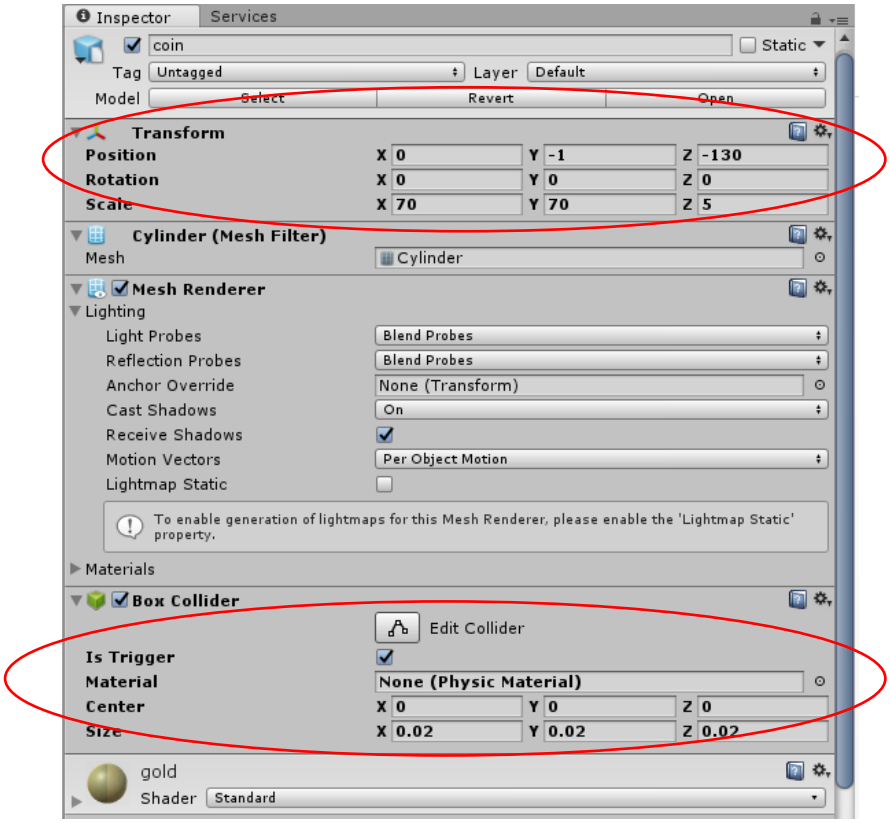
Adăugăm la proprietățile acestuia ***Rigidbody*** și ***Box Colider***. Important este să setăm aceste două noi proprietăți adăugate ca în figura de mai jos! Proprietatea ***Rigidbody*** permite ca utilizatorul să dea un efect de gravitație obiectului 3D. Proprietatea ***Box Colider*** este utilă în jocul nostru, deoarece se va folosi atunci când două obiecte 3D se ciocnesc. Box Colider arată la fel ca un cub (vezi figura caracterului 3D). Acesta trebuie să aibă poziția și scala asemănătoare obiectului 3D.



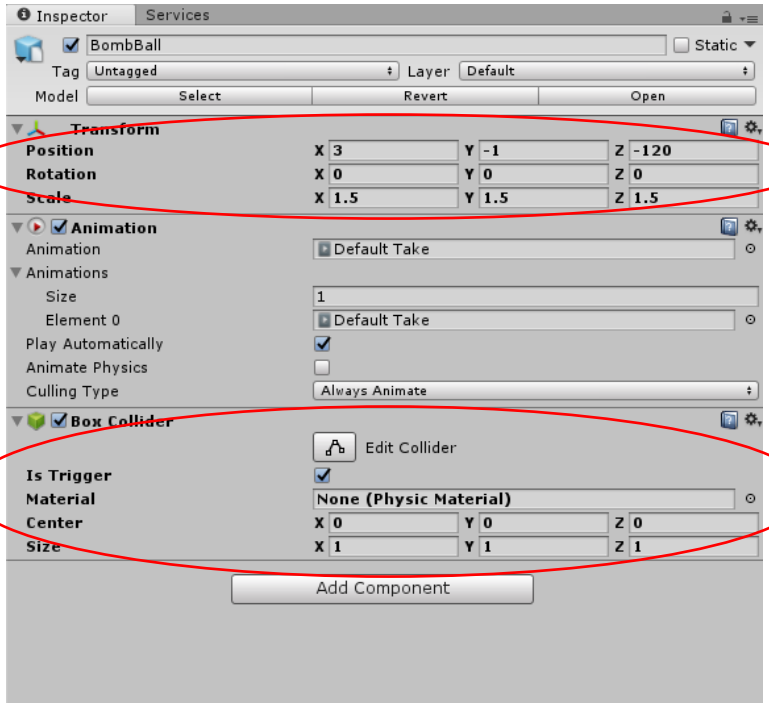
După setarea proprietăților caracterului, acesta va arăta ca în figura de mai jos:



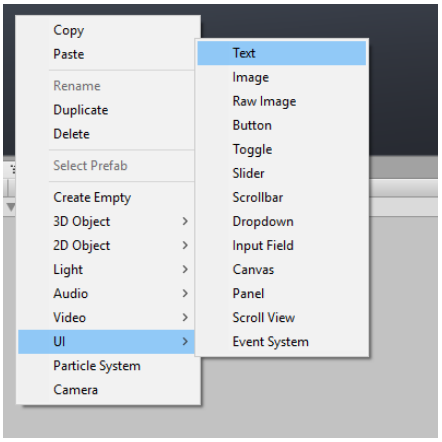
Adăugăm în scenă obiectul ***coin*** din fișierul cu același nume. (fișierul ***coin –> coin***). Alegem obiectul care are textură. Setăm obiectul coin la poziția dorită și adăugam Box Colider setat ca în figură.



Adăugăm în sccenă obiectul ***BombBall*** din fișierul cu numele Ball Pack. (***Ball Pack – Models - BombBall***). Setăm obiectul BombBall la poziția dorită și adăugam Box Colider setat ca în figură.

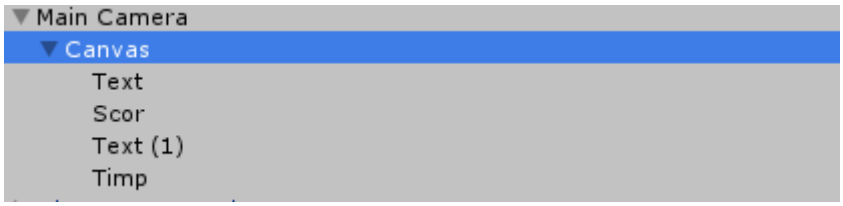


Adăugăm 4 componente de tip text în scena noastră (***Click dreapta – UI –Text***). Acestea sunt folosite pentru afișarea în scenă a diferitor texte (ex. afișarea scorului, a timpului). Când se crează un text automat se crează obiectul Canvas, care conține în interiorul acestuia textul creat.

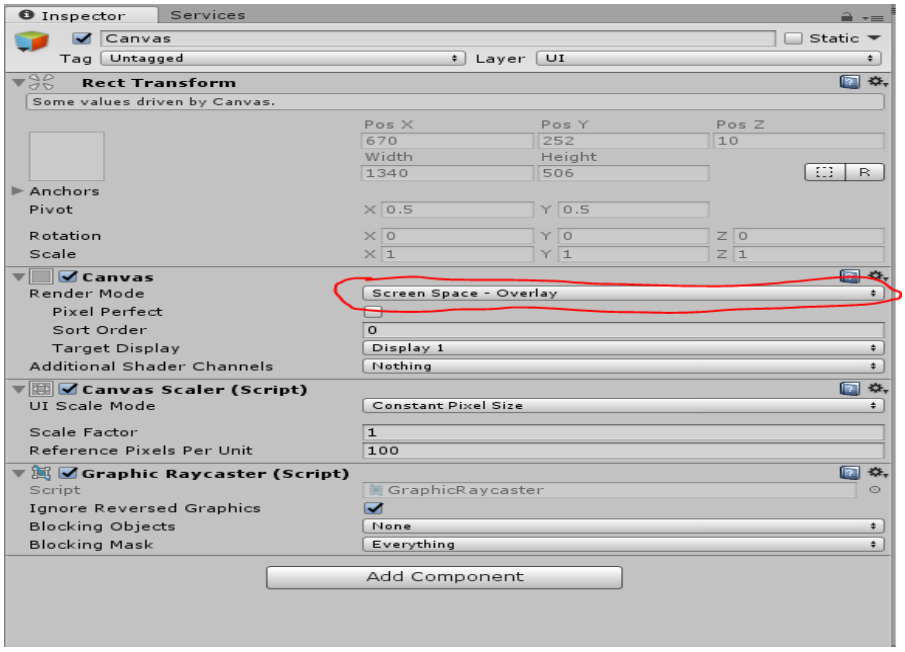


Observație:

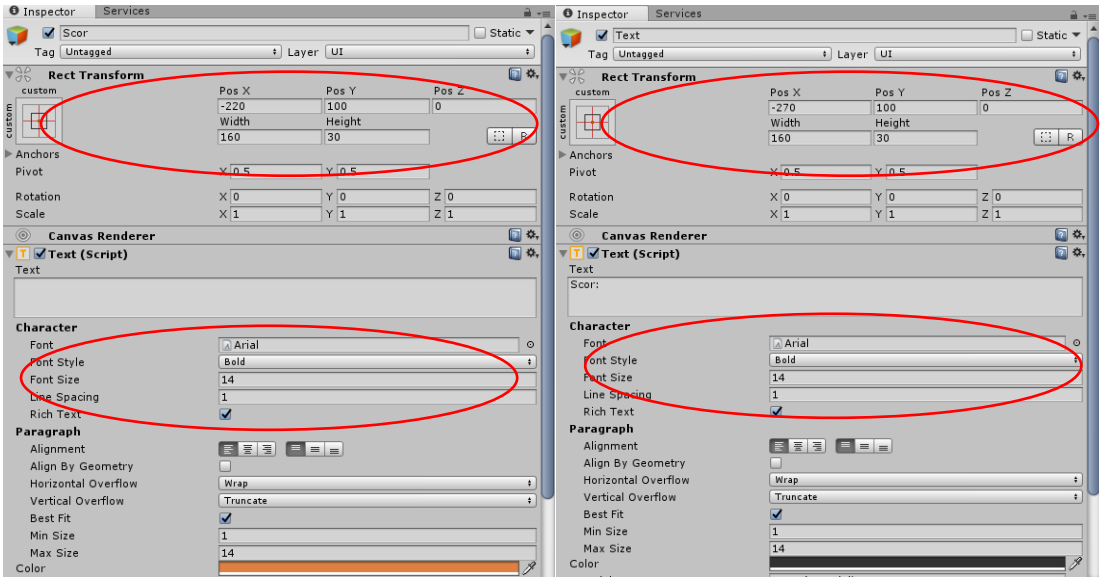
Vom trage (drag and drop) obiectul *Canvas* unde se găsesc componentele de tip text create peste obiectul *Main Camera*, astfel încât *Canvas* să devină copilul camerei. Acesta inițial apare în lista de obiecte create unde se gasește și Main Camera, iar el trebuie mutat în interiorul acesteia.

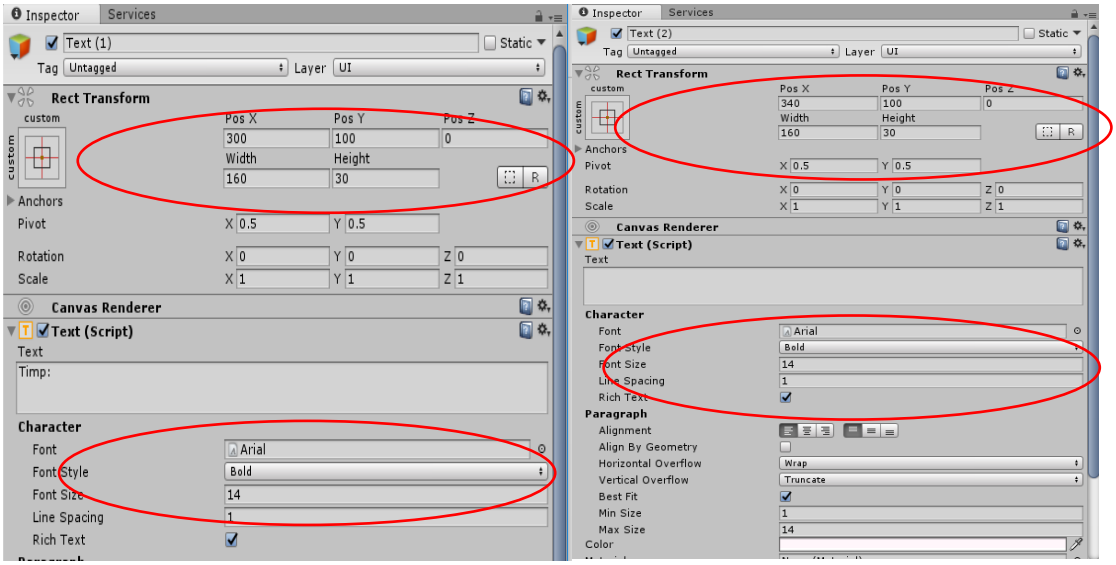


Setăm obiectul *Canvas* ca în imaginea de mai jos.

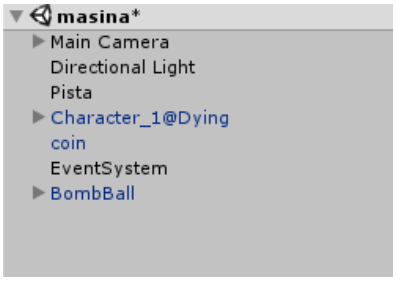


Setăm fiecare componentă de tip text astfel încât acestea să apară în partea de sus a ecranului stânga (scorul) si dreapta (timpul). Vom seta fontul și culoarea fiecărui text. În figurile de mai jos se arată cum au fost poziționat fiecare text în spațiul 3D, precum și fontul acestora. De exemplu: textul cu numele ***Scor*** a fost poziționat la ***poziția -220 pe axa x, 100 pe axa y și 0 pe axa z; având Font-ul Arial, Bold și dimensiunea 14.***





Mai jos sunt prezentate toate obiectele și componentele folosite în jocul pe care o să-l implementăm.

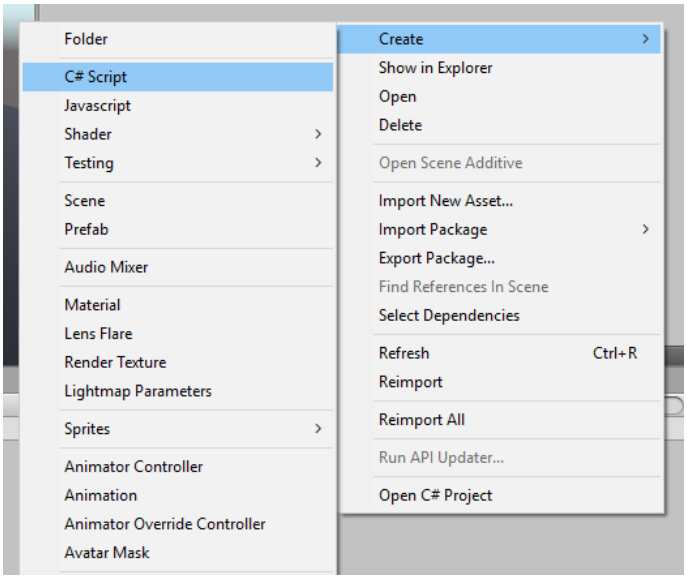


Salvarea scenei curente: ***File – Save Scene as – nume scenă.unity (în cazul nostru masina)***. De reținut *numele* dat scenei, deoarece se va folosi la navigarea între scene în joc!

În continuare se vor explica principalele funcționalități implementate împreună cu script-urile C# corespunzătoare.

1. Mișcarea automată a caracterului pe axa z:

   În fișierul Scripturi se va crea un script C# (Click dreapta în fișierul Scripturi – Create – C# Script) cu numele ***miscare_automata***



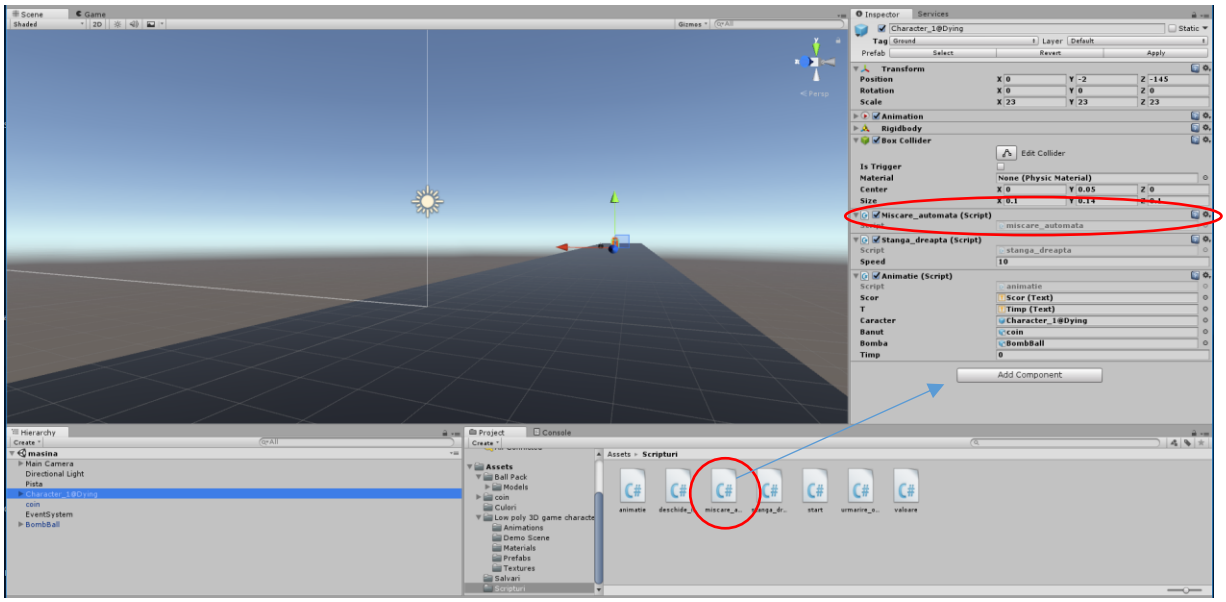
   În script-ul miscare_automata se va adăuga codul de mai jos:

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

public class miscare_automata : MonoBehaviour {

    //variabila viteza - viteza de deplasare a caracterului
    float viteza=5.0f;

    //functia Update() se apeleaza la fiecare frame
    void Update () {
        //setarea pozitiei caracterului pe axa z; aceasta creste o data la o secunda cu 5 unitati in spatiul 3D
        transform.Translate(0, 0, Time.deltaTime*viteza);
    }
}
```

Observație!

Toate variabilele/funcțiile declarate publice vor apărea în proprităție obiectului unde este folosit script-ul.

Acest script va fi adăugat peste obiectul ***Character_1@Dying*** din scenă.



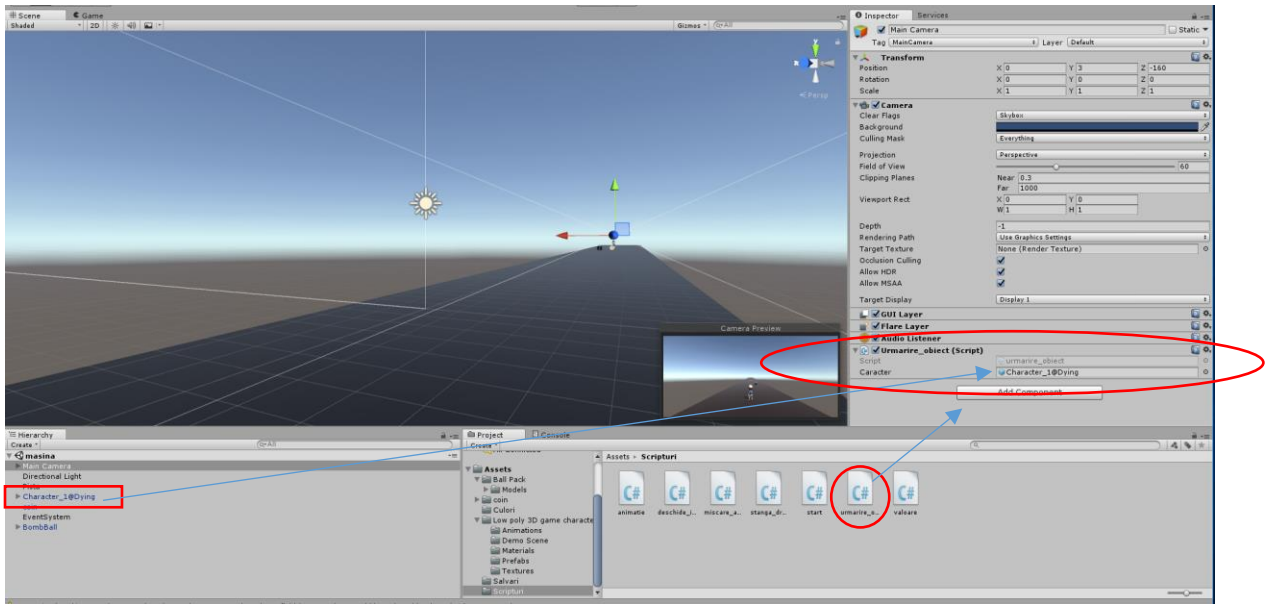
Rularea aplicației se face prin apăsarea butonului de Play (vezi imaginea de mai jos)

2. Urmărirea caracterului de către cameră.

În fișierul Scripturi se va crea un script C# (Click dreapta în fișierul Scripturi – Create – C# Script) cu numele ***urmărire_obiect***

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

public class urmarire_obiect : MonoBehaviour {

    //variabila folosita pentru setarea pozitiei caracterului
    public GameObject caracter;

    //variabila folosita pentru calcularea distantei dintre caracter si camera
    private Vector3 distanta;

    void Start () {
        //calcularea distantei dintre caracter si camera
        distanta = transform.position - caracter.transform.position;
    }

    //functia LastUpdate se apeleaza dupa functia Uptade pentru fiecare frame
    void LateUpdate () {
        //setarea pozitiei camerei fata de caracter
        transform.position = caracter.transform.position + distanta;
    }
}
```

Acest script va fi adăugat peste obiectul ***Main Camera*** din scenă.



După ce se adaugă acest script trebuie să tragem la variabila Caracter care se observă mai sus, obiectul ***Character_1@Dying.*** După cum s-a explicat mai sus variabila declarată publică în script apare și pe proprietatea obiectului unde este folosit script-ul.

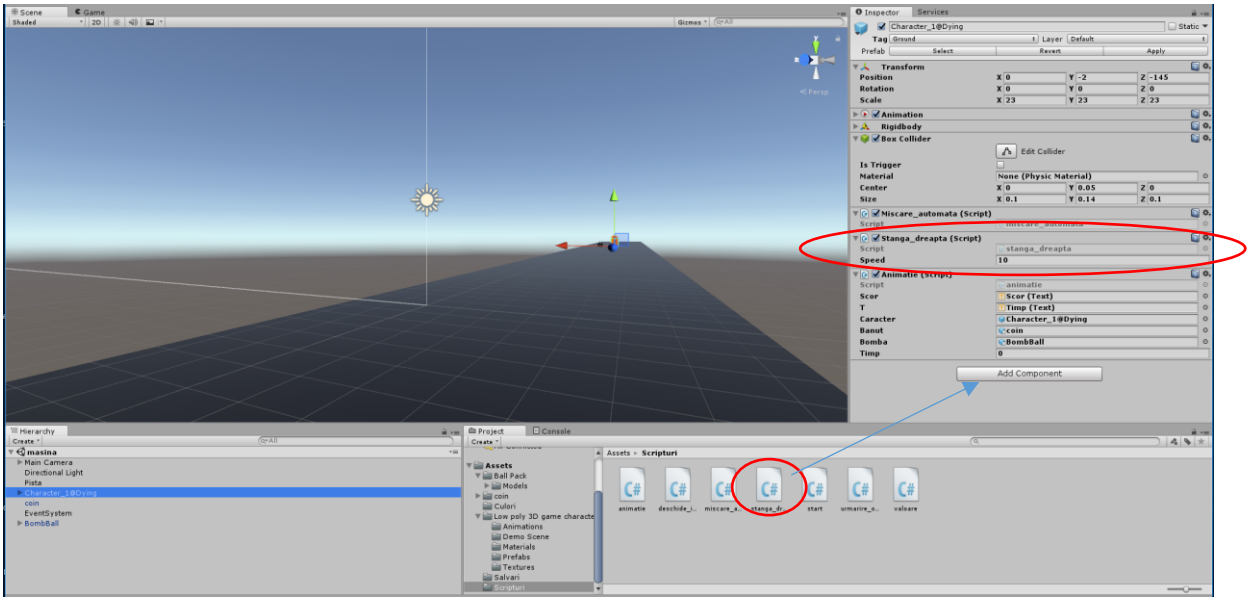
La rularea scenei caracterul se va mișca automat în scenă, iar camera va urmări caracterul.

3. Mișcarea caracterului de la tastatură

În fișierul Scripturi se va crea un script C# (Click dreapta în fișierul Scripturi – Create – C# Script) cu numele ***stanga_dreapta***

```
public class stanga_dreapta : MonoBehaviour {
    //variabila folosita pentru viteza de viscare
    public float speed = 10f;
    //variabila folosita pentru accesarea proprietatii RigidBody a caracterului
    private Rigidbody rb;

    void Start()
    {
        rb = GetComponent<Rigidbody>();
    }

    void FixedUpdate()
    {
        //variabila folosita pentru miscarea pe orizontala (stanga-dreapta) (A sau sageata-stanga pentru stanga si D sau sageata-dreapta pentru dreapta)
        float h = Input.GetAxis("Horizontal");
        //variabila folosita pentru miscarea pe verticala (fata-spate) (W sau sageata-sus pentru fata si S sau sageata-jos pentru jos )
        float v = Input.GetAxis("Vertical");
        //crearea vectorului de miscare
        Vector3 moveVector = (transform.forward * v) + (transform.right * h);
        // setarea miscarii caracterului cu o viteza setata la 10 unitati pe secunda
        moveVector *= speed * Time.deltaTime;
        //miscarea caracterului
        transform.localPosition += moveVector;
    }

}
```

Acest script va fi adăugat peste obiectul ***Character_1@Dying*** din sccenă.



4. Coliziunea dintre caracter și alte caractere, setarea scorului, a timpului. Generarea de noi obiecte dinamic, navigarea dintre scene. Activarea unei animați pe caracter. (Click dreapta în fișierul Scripturi – Create – C# Script) cu numele ***animatie***.

În script se vor adăuga următoarele biblioteci:

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

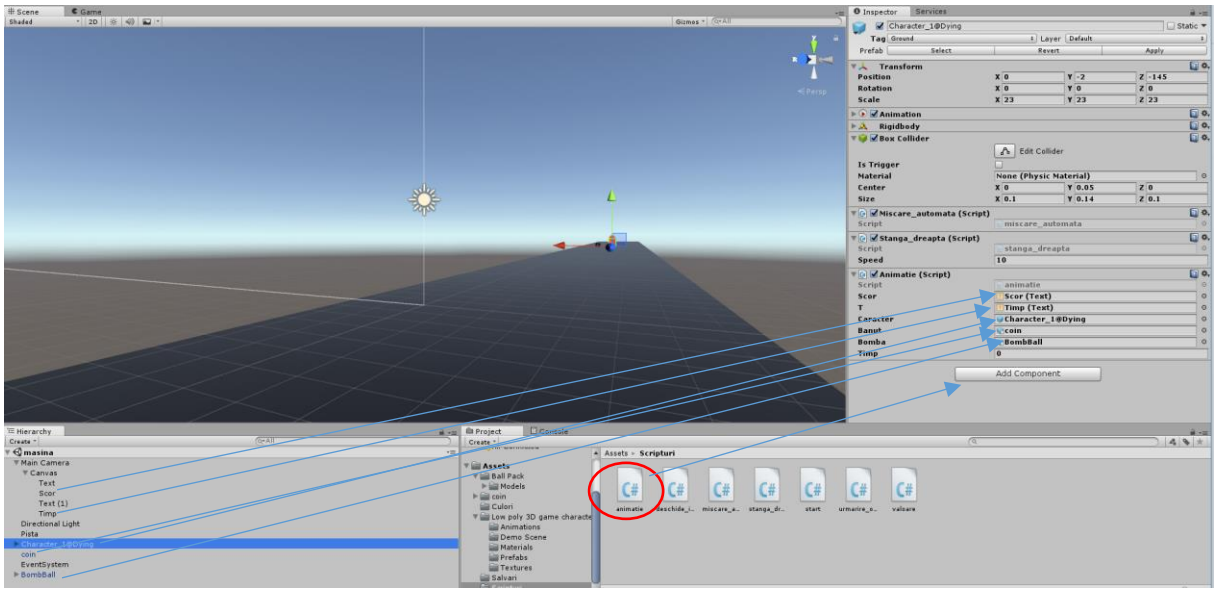
În clasa animație se vor pune următoarele linii de cod.

```
//variabile folosite pentru afisarea scorului si timpului
public  Text scor, t;
//variabila caracter folosita pentru ca acesta sa efectueze operatia de saritura
//variabila banut folosita pentru generarea dinamica de noi obiecte de acelasi tip
//variabila bomba folosita pentru generarea dinamica de noi obiecte de acelasi tip
public GameObject caracter, banut, bomba;
//variabila a folosita pentru contorizarea scorului
//variabila k folosita pentru distanta de generare de obiecte noi fata de vechiul obiect (banut, bomba)
//variabila z folosita pentru pozitia initiala de generare de noi obiecte
float  a = 0f, k = 10f, z = -130f;
//variabuila timp folosita pentru cronometrare
public float timp = 0f;
```

```
void Update()
{
    //aceesarea proprietatii Animatie a caracterului
    Animation animatie = caracter.GetComponent<Animation>();
    //daca se apasa pe tasta space
    if (Input.GetKeyDown("space"))
    {
        //activare animatie specifica sariturii
        animatie.Play("Jump_2");
        //setarea pozitiei caracterului pe axa y mai mare cu 2 unitati fata de pozitia precedenta
        caracter.transform.position = caracter.transform.position + new Vector3(0f, 2f, 0f);
    }
    //daca nu se apasa pe tasta space
    else
        //activarea animatiei specifice alergarii caracterului
        animatie.Play("Running");
    //incrementarea timpului
    timp = timp + Time.deltaTime;
    //setarea obiectului t de tip text cu valoarea variabilei timp
    t.text = string.Format("{0}", timp + "secunde");
}
```

```
private void OnTriggerEnter(Collider other)
{
    //daca se produce o coliziune intre caracter si obiectul coin sau coin(Clone)
    if (other.gameObject.name == "coin" || other.gameObject.name == "coin(Clone)")
    {
        //scorul va creste cu 1
        a = (int)(a + 1);
        ////setarea obiectului scor de tip text cu valoarea variabilei a
        scor.text = string.Format("{0}", (int)a);
        //daca scorul este mai mare sau egal decat 1
        if (a >= 1)
        {
            //generarea dinamica de obiect, banut, aleatoriu pe axa x intre valorile -6 si 6 si pe axa y intre valorile 0 si 1 la o distanta de pozitie initiala z, plus
              distanta k
            Instantiate(banut, new Vector3(Random.Range(-6f, 6f), Random.Range(0f, 1f), z + k), Quaternion.identity);
            //cresterea valorii k cu 19 unitati
            k = k + 10;
            //generarea dinamica de obiect, banut, aleatoriu pe axa x intre valorile -6 si 6 si pe axa y intre valorile 0 si 1 la o distanta de pozitie initiala z, plus
              distanta k
            Instantiate(banut, new Vector3(Random.Range(-6f, 6f), Random.Range(0f, 1f), z + k), Quaternion.identity);
            k = k + 10;
        }
    }
    //daca se produce o coliziune intre caracter si obiectul bomba (BombBall) sau (BombBall(Clone))
    if (other.gameObject.name == "BombBall" || other.gameObject.name == "BombBall(Clone)")
    {
        //deschiderea sccenei cu numele game over
        SceneManager.LoadScene("game over");

    }
    //daca scorul este mai mare sau egal decat 1
    if (a >= 1)
    {
        //generarea dinamica de obiect, bomba, aleatoriu pe axa x intre valorile -6 si 6 si pe axa y intre valorile 0 si 1 la o distanta de pozitie initiala z, plus distanta
        Instantiate(bomba, new Vector3(Random.Range(-6f, 6f), Random.Range(0f, 1f), z + k), Quaternion.identity);
        k = k + 10;
    }
}
```

Acest script va fi adăugat peste obiectul ***Character_1@Dying*** din scenă.

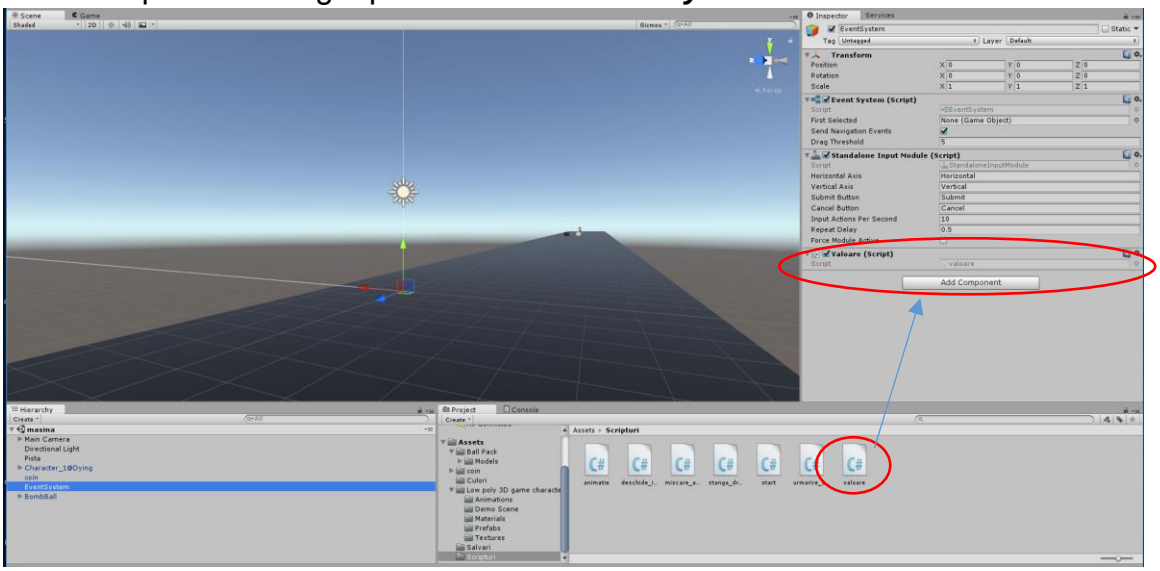


Peste fiecare variabila care apare în proprietățile caracterului unde se găsește scriptul adăugat, se vor trage (drag and drop) următoarele: *Scor* -> componenta text cu numele **Scor**, *T* -> component text cu numele **Timp**, *Caracter* -> componenta **Character_1@Dying**, *Banut* -> **coin**, *Bomba* -> **BombBall** (vezi figura de mai sus).

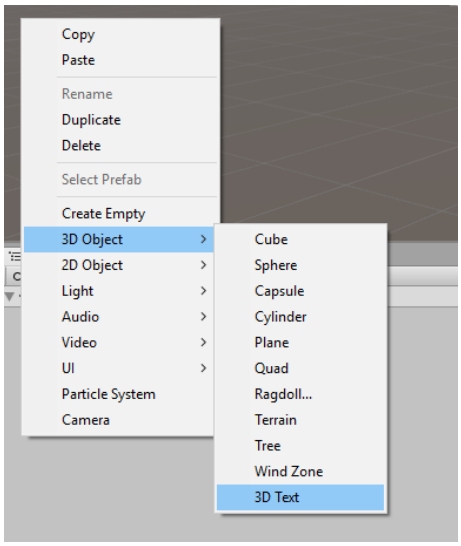
5. Transmiterea variabilelor între scene (Click dreapta în fișierul Scripturi – Create – C# Script) cu numele ***valoare***.

```
public class valoare : MonoBehaviour {

    //variabila statica pentru transmiterea valorii la o alta sccena
    public static string t;

    void Update () {
        // accesarea valorii de tip text care are cale: Canvas - Scor
        Text scor = GameObject.Find("Canvas/Scor").GetComponent<Text>();
        //initializarea variabilei t
        t = scor.text;

    }
}
```

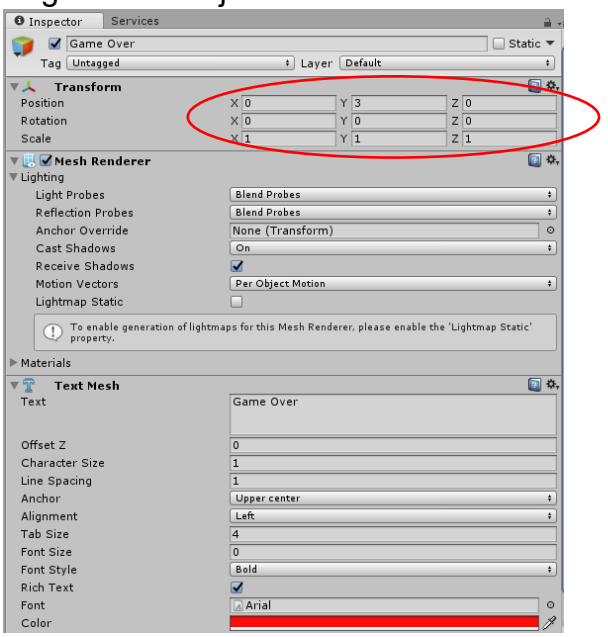
Acest script va fi adăugat peste obiectul ***EventSystem***.



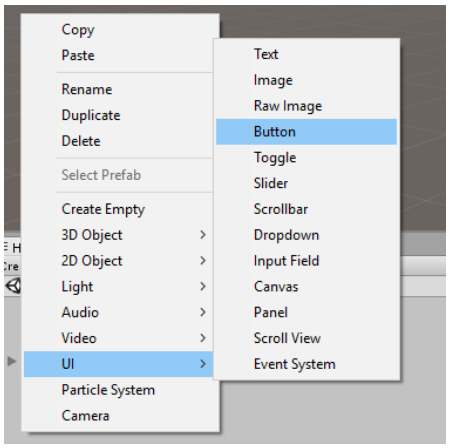
Crearea unei noi scene cu numele ***game over*** (File – New Scene).
Adăugăm în scenă un obiect de tipul *3D Text* (***Click dreapta – 3D Object – 3D Text***)



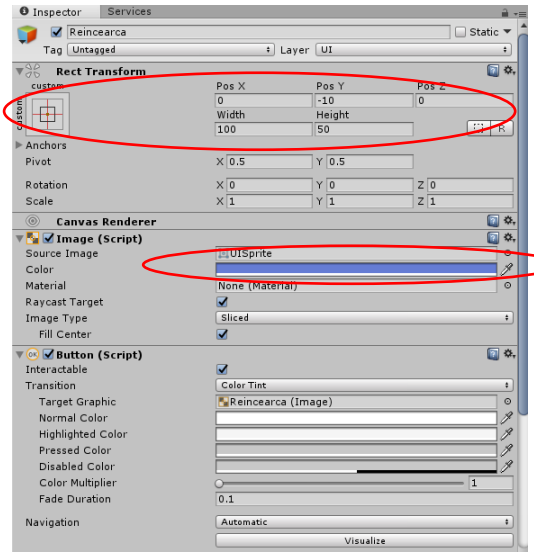
Setăm obiectul ca în figura de mai jos:



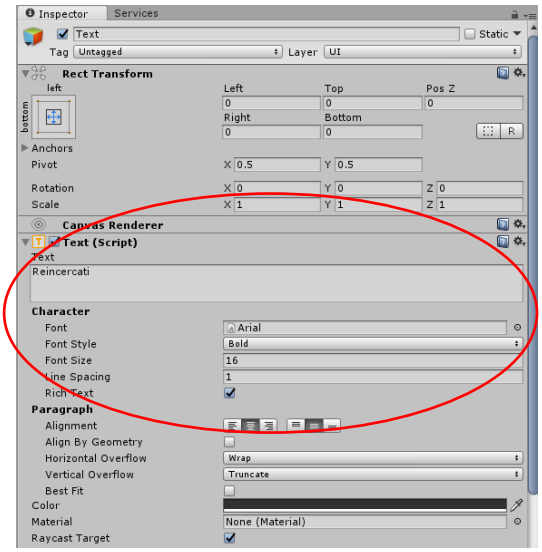
Creem două butoane, unul folosit pentru ca utilizatorul să poată reîncerca aplicația și unul pentru închiderea aplicației (***Click dreapta – UI – Button***).



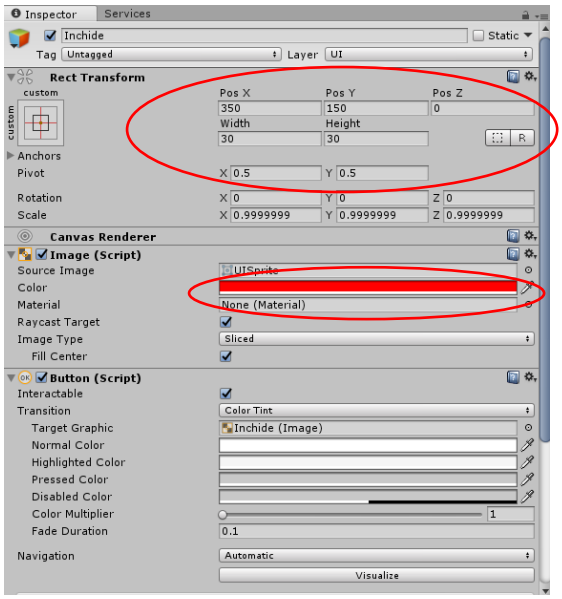
Setarea butonului de *Reincercare, poziția și culoarea acestuia*:



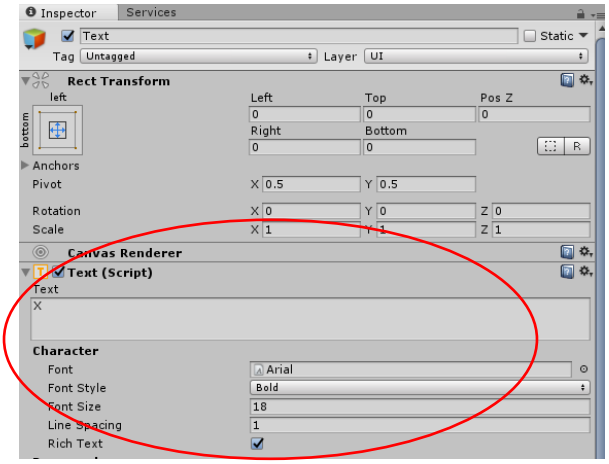
Setarea textului de pe buton: se selectează butonul, apoi se accesează textul acestuia. În proprietățile textului se setează fontul și textul.



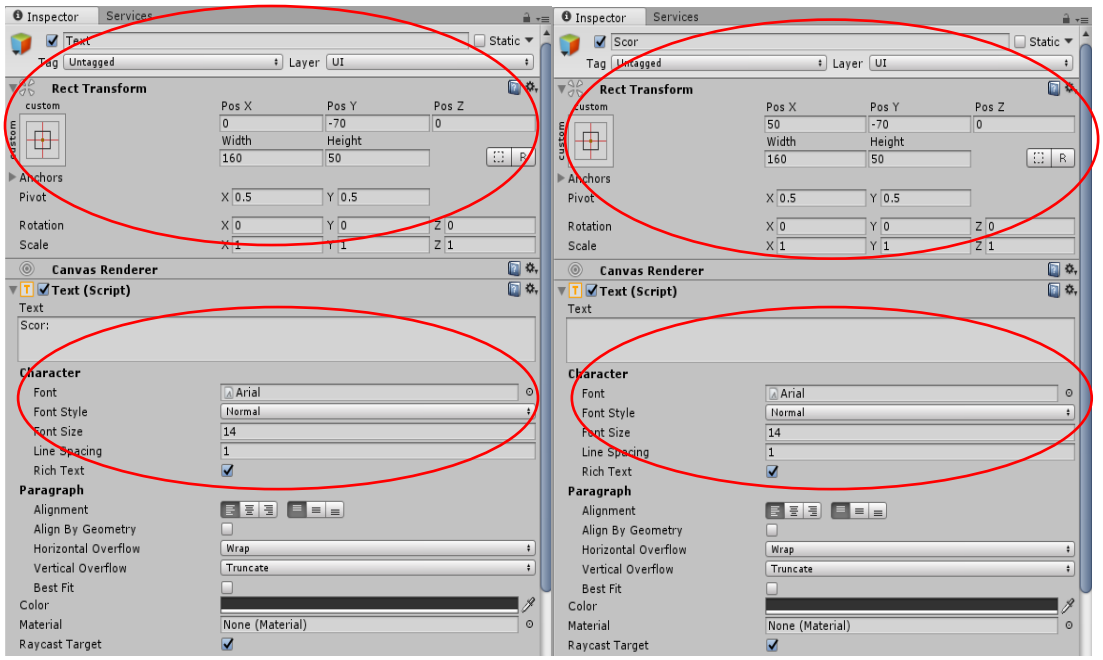
Setarea butonului de *Inchide,* poziția și culoarea acestuia:



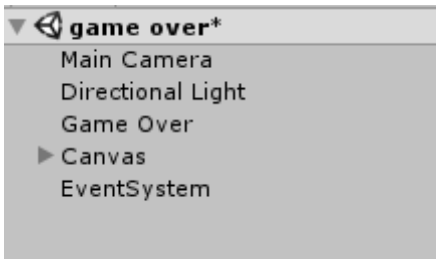
Setarea textului de pe buton: se selectează butonul, apoi se accesează textul acestuia. În proprietățile textului se setează fontul și textul.



Crearea a două componente de tip text pe scenă (***Click dreapta unde se găsesc și celelalte obiecte create –> UI –>Text***). Cele două componente vor afișa scorul utilizatorului din scena principală. Acestea sunt setate la pozițiile din spațiul 3D conform imaginilor de mai jos. Totodată pentru fiecare text s-a ales un font.



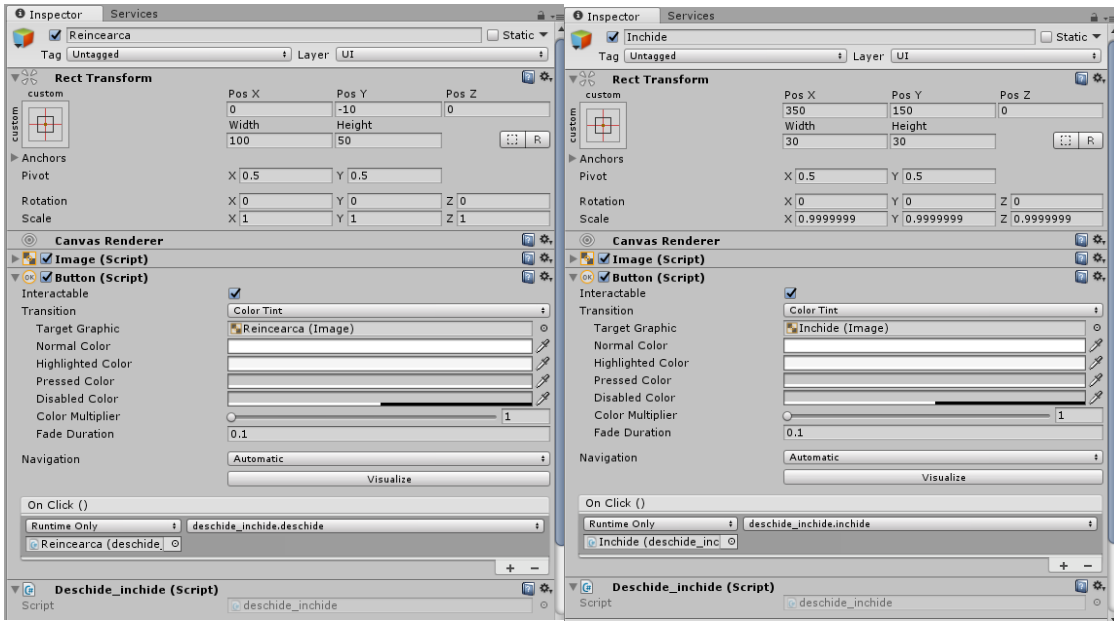
Mai jos sunt prezentate toate obiectele și componentele folosite în scenă:

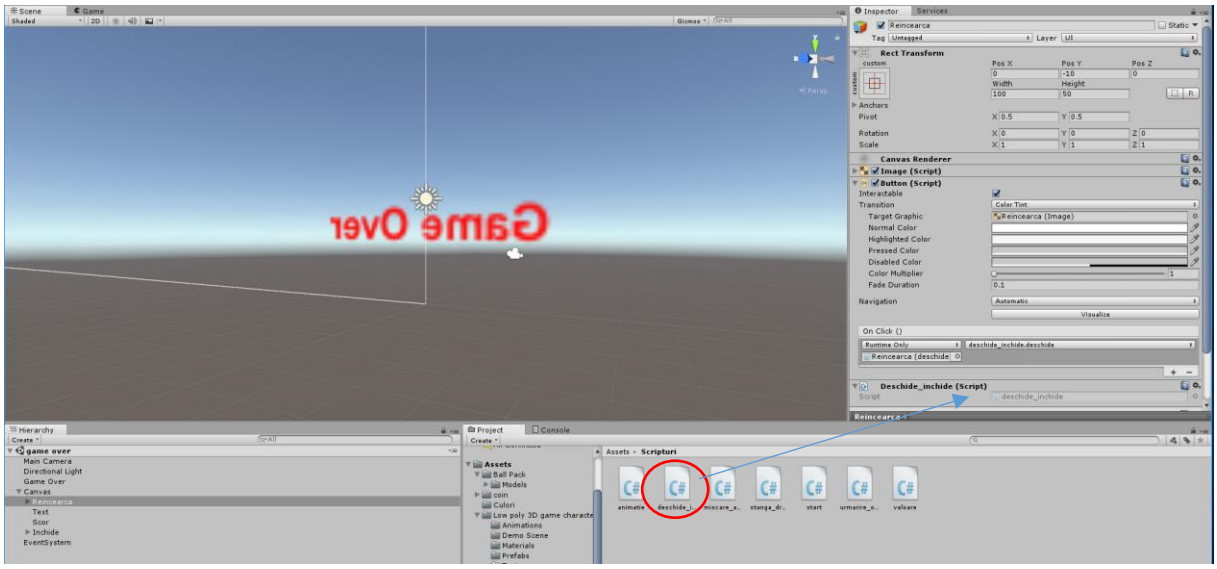


6. Navigarea la apăsarea butonului ***Reincercati*** pe sccena principală și închiderea aplicației la apăsarea butonului ***Inchide*** (Click dreapta în fișierul Scripturi – Create – C# Script) cu numele ***deschide_inchide***.

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

public class deschide_inchide : MonoBehaviour {

    public void deschide()
    {
        //redeschiderea principalei sccene cu numele masina
        SceneManager.LoadScene("masina");
    }

    public void inchide()
    {
        //inchiderea aplicatiei
        Application.Quit();
    }
}
```

Acest script va fi adăugat peste butonul Reincercati și peste butonul Inchide. La fiecare proprietate cu numele ***On Click()*** a butoanelor va fi adăugată funcția deschide pentru butonul ***Reincercati*** si inchide pentru butonul ***Inchide***.

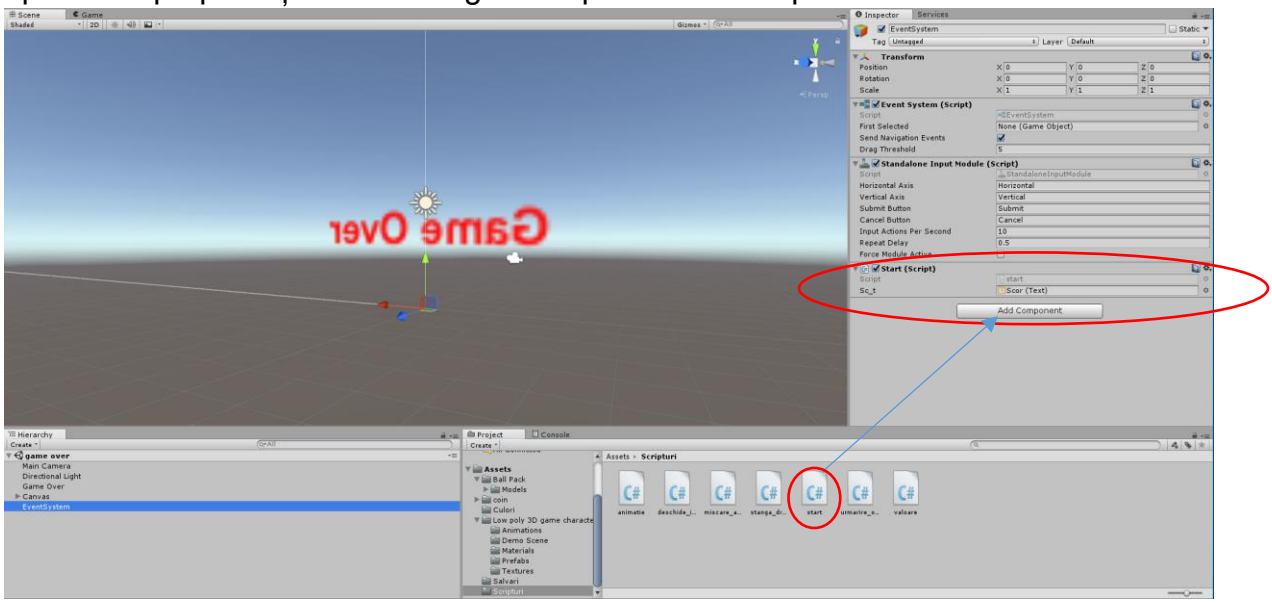




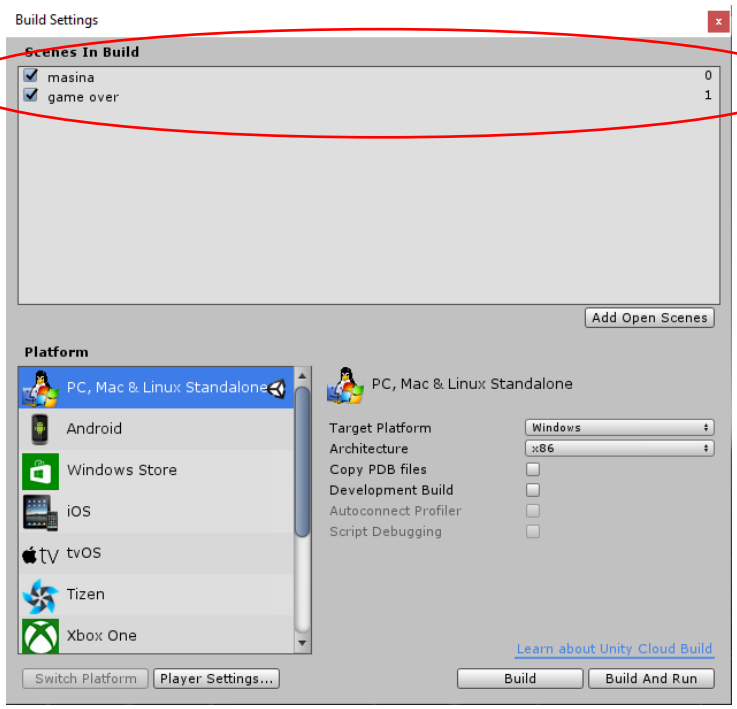
7. Setarea scorului și salvarea acestuia într-un fișier text (Click dreapta în fișierul Scripturi – Create – C# Script) cu numele ***start.***

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

public class start : MonoBehaviour {
    //variabila folosita pentru calea spre fisierul unde se va salva scorul
    string cale = "Assets/Salvari/scor.txt";
    //variabila folosita pentru afisarea scorului
    public Text sc_t;

    void Start()
    {
        //initializarea valorii variabilei sc_t cu valoarea variabilei t din clasa valoare
            sc_t.text = valoare.t;
       //initializarea conexiunii cu fisierul text
            StreamWriter f = new StreamWriter(cale, true);
       //scrierea scorului in fisierul text cu numele scor.txt
            f.WriteLine(sc_t.text);
       //inchiderea conexiunii
            f.Close();
    }
}
```

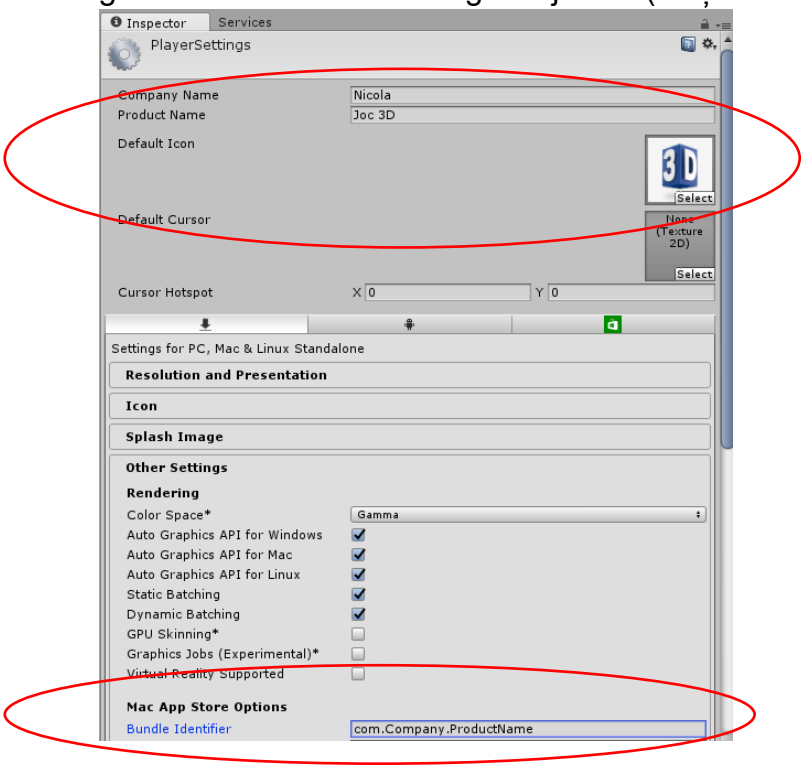
Acest script va fi adăugat peste obiectul ***EventSystem*** iar peste variabila Sc_t care apare la proprietăți se va trage componenta de tip text cu numele **Scor**.



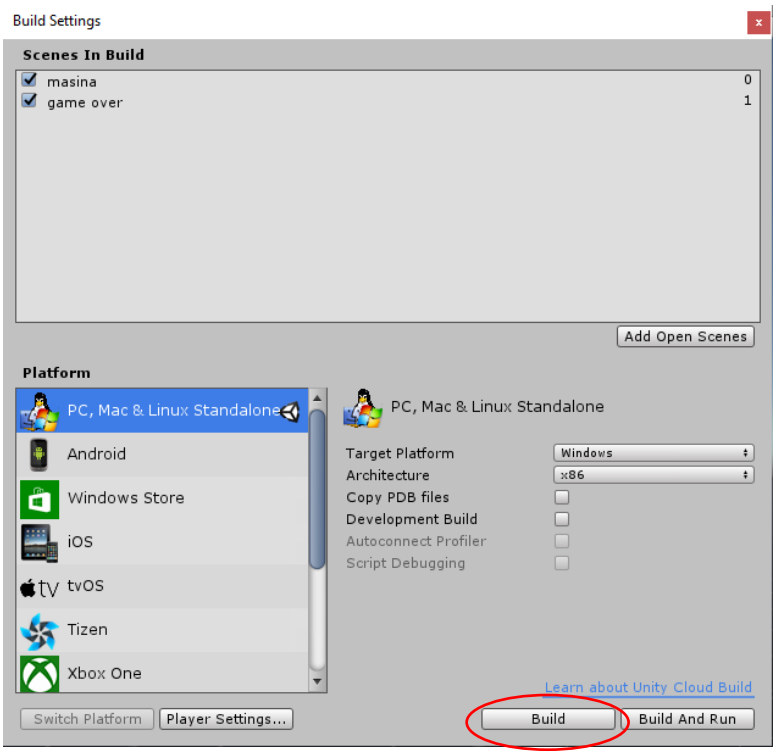
Generarea fișierului executabil: (***File -> Build Settings***). Se selectează sistemul de operare pentru care se generează fișierul, apoi se bifează scenele și ordinea lor, care se vor utiliza în generarea fișierului.

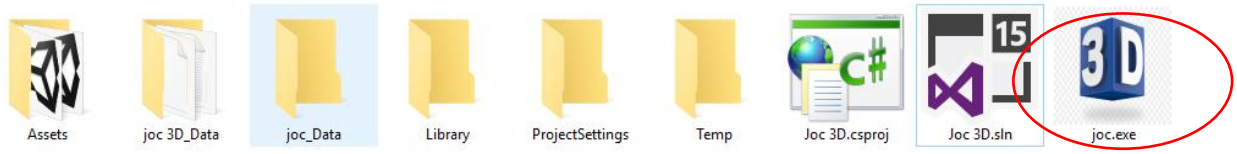


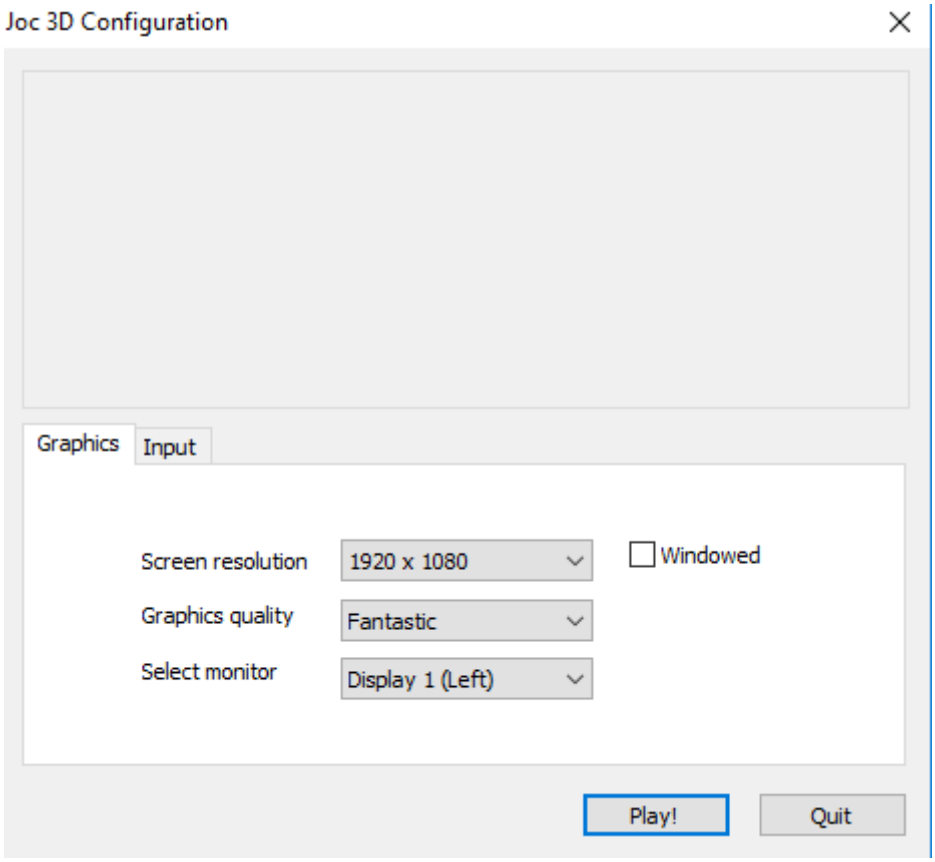
Setarea imaginii jocului si a numelui companiei. (***File -> Build Settings -> PlayerSettings***). La *Company Name* se pune un nume (ex. numele dumneavoastră), Product Name (ex. numele proiectului), iar la Default Icon se selectează o imagine care dorim să fie imaginea jocului (a fișierului executabil).

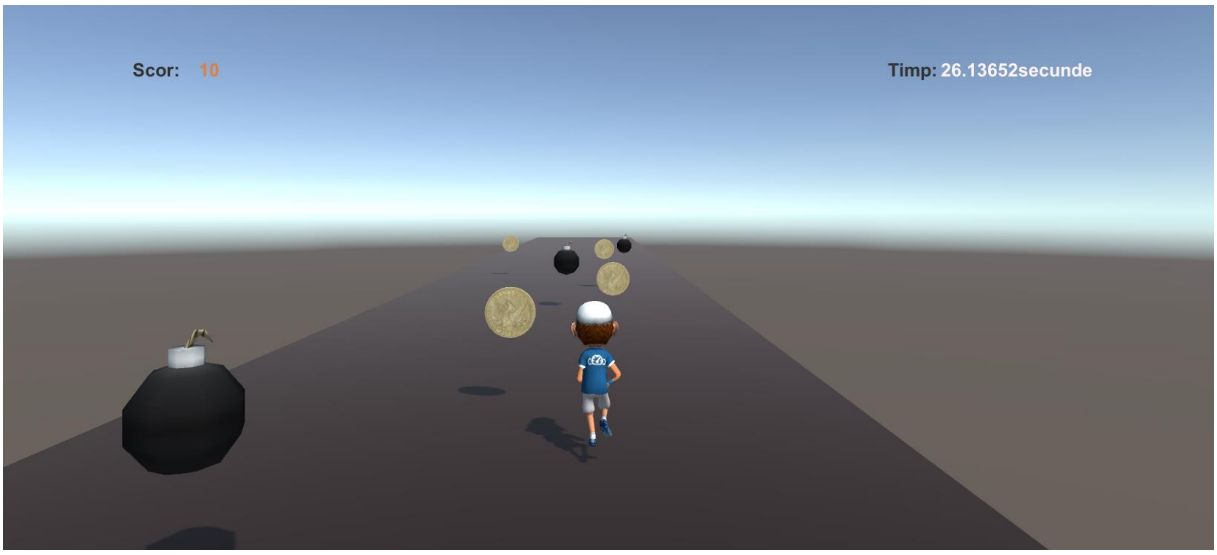


File -> Buil Settings -> Build -> Nume.exe (joc.exe)


Build Settings
Scenes In Build
masina
0
game over
1
Add Open Scenes
Platform
PC, Mac & Linux Standalone
Android
Windows Store
iOS
tvOS
Tizen
Xbox One
PC, Mac & Linux Standalone
Target Platform
Windows
Architecture
x86
Copy PDB files
Development Build
Autoconnect Profiler
Script Debugging
Learn about Unity Cloud Build
Switch Platform
Player Settings...
Build
Build And Run


Assets
joc 3D_Data
joc_Data
Library
ProjectSettings
Temp
Joc 3D.csproj
Joc 3D.sln
joc.exe


Joc 3D Configuration
Graphics
Input
Screen resolution
1920 x 1080
Windowed
Graphics quality
Fantastic
Select monitor
Display 1 (Left)
Play!
Quit


Scor: 10
Timp: 26.13652secunde


X
Game Over
Reincercati
Scor: 10