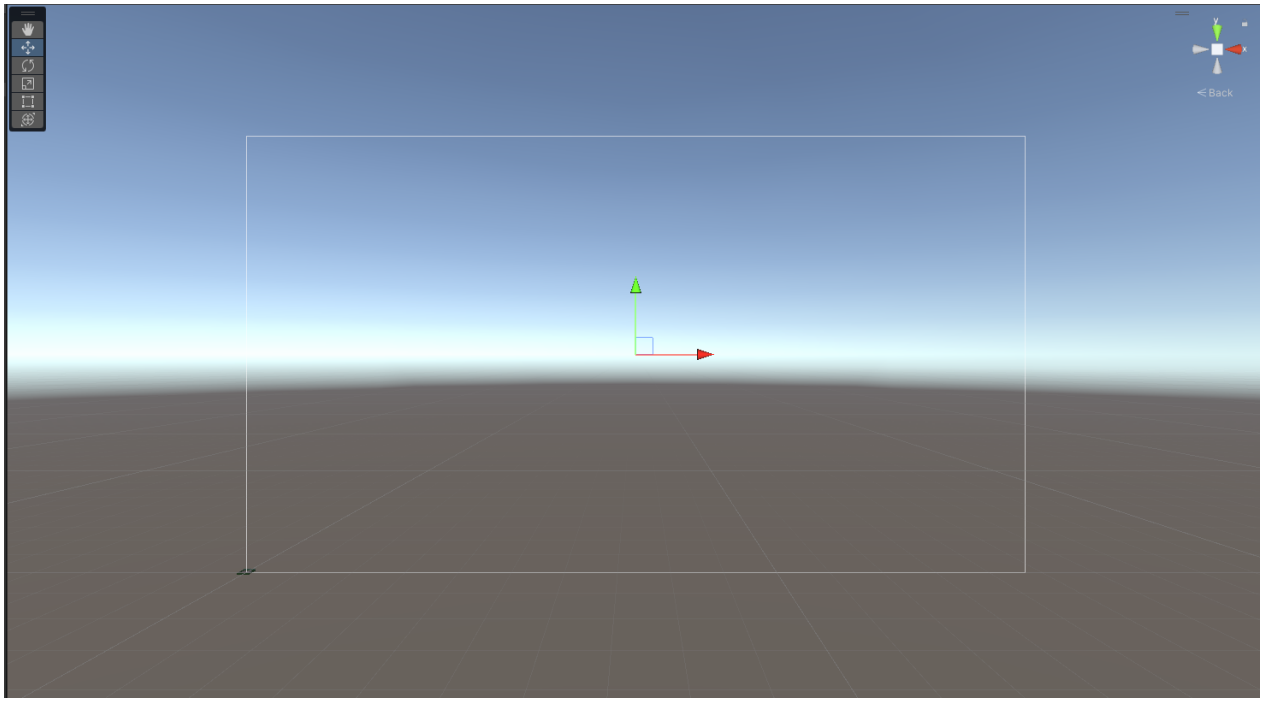
Laboratorul 8

În acest laborator vom adăuga o interfață grafică jocului la care am lucrat în laboratoarele anterioare. Vom porni de la această versiune a proiectului.

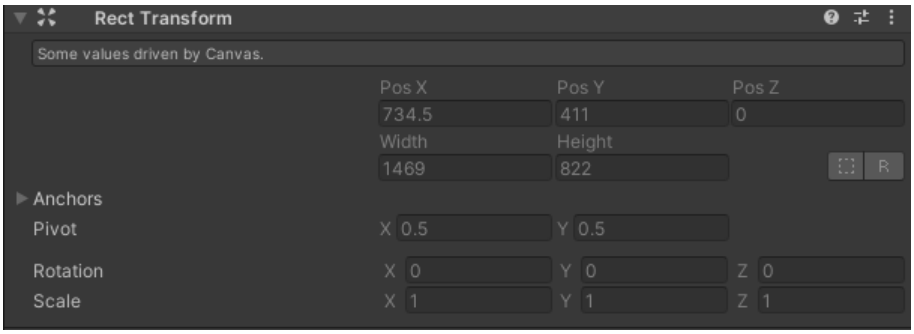
1 *UI*

1.1 *Canvas*

Pentru a putea adăuga elemente de *UI* în scenă avem nevoie de un obiect de tip *Canvas*. Obiectele de tip *Canvas* reprezintă suprafețe plane în interiorul cărora se pot adăuga elemente de *UI* care vor apărea pe ecran. Vom adăuga în scena *GameScene* un obiect de tip *Canvas* (*click dreapta in Hierarchy/UI/Canvas*). În scenă se va adăuga un obiect numit *Canvas* care conține componenta *Canvas* și un obiect numit *EventSystem*, care conține componenta *Event System*. Acest obiect din urmă este necesar pentru ca interacțiunile cu interfața grafică să funcționeze. Vom schimba numele obiectului de tip *Canvas* în *GameCanvas*.



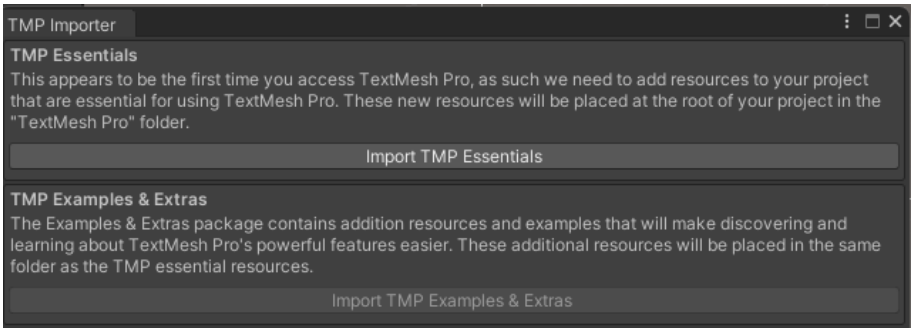
În inspector, putem observa că obiectul de tip *Canvas* are o componentă de tip *Rect Transform* în locul componentei *Transform* cu care eram obișnuiți în cazul celorlalte obiecte. Această componentă face mai ușoară manipularea obiectelor de UI decât o componentă *Transform* normală. În *Unity*, toate elementele de UI conțin componenta *Rect Transform*.



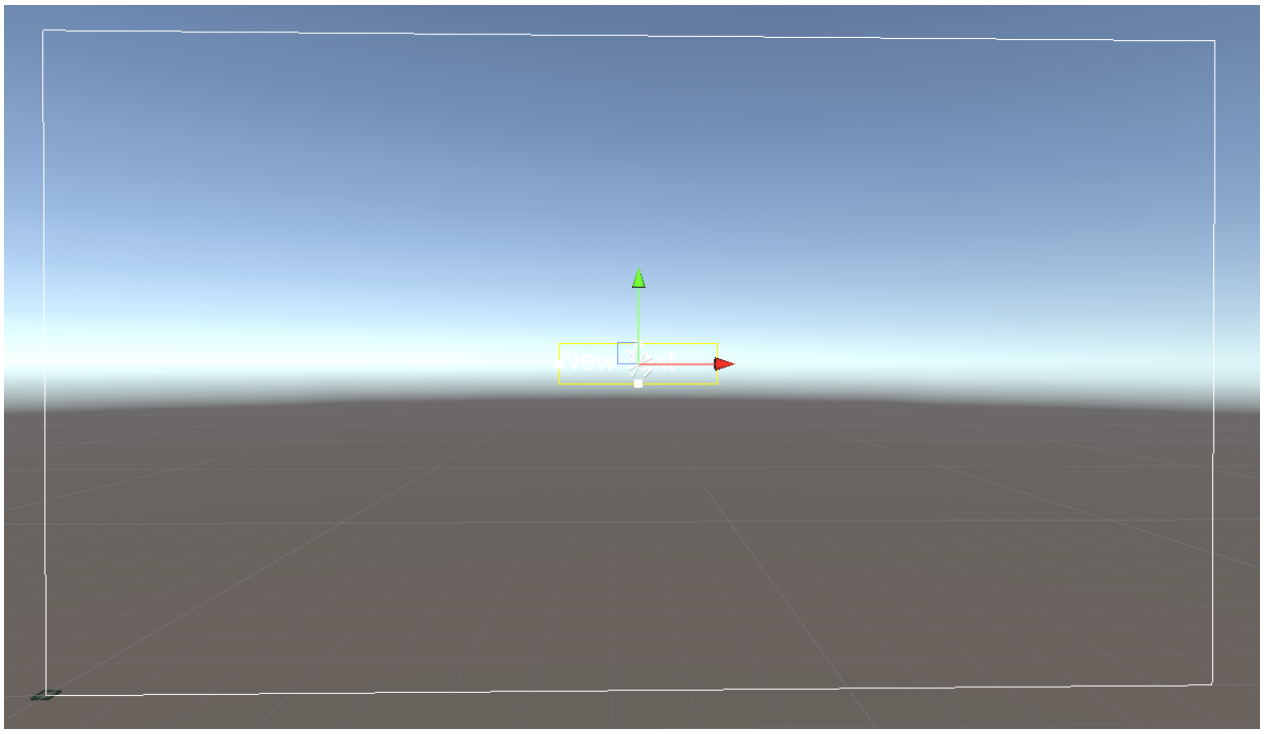
1.2 Scor

Vom dori să afișăm scorul jucătorului în partea din stânga-sus a ecranului. Pentru asta, vom adăuga un obiect de tip $Text - TextMeshPro$ în interiorul obiectului $GameCanvas$ creat anterior (*click dreapta pe obiectul* $GameCanvas/UI/Text - TextMeshPro$).

Dacă este prima dată când adăugăm un element $Text$ în UI vom fi întâmpinați cu următoarea fereastră:

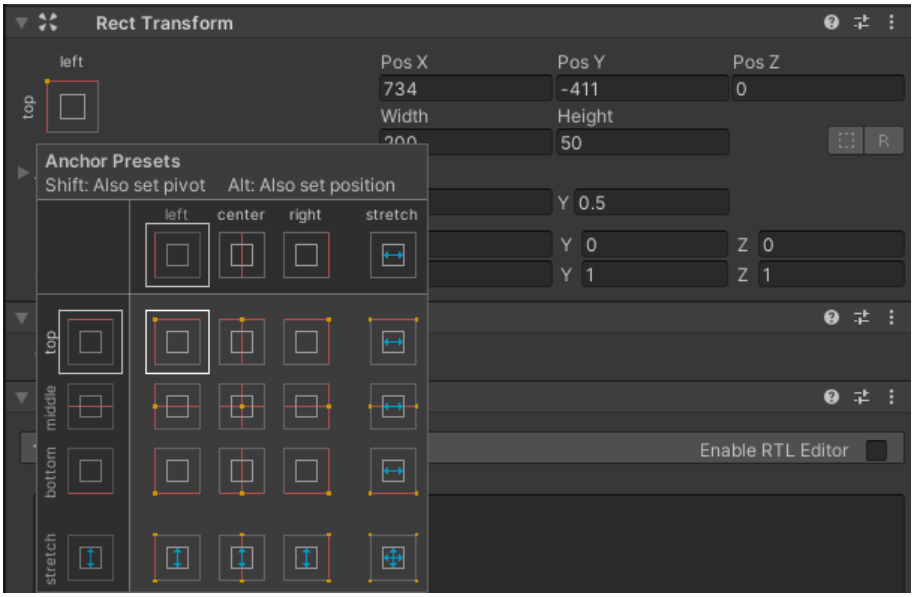


Vom selecta $Import\ TMP\ Essentials$, iar dupa ce pachetele pentru $TextMeshPro$ se vor fi importat, vom închide această fereastră (nu este necesar să apăsăm și pe butonul $Import\ TMP\ Examples\ \&\ Extras$ după ce acesta devine activ).

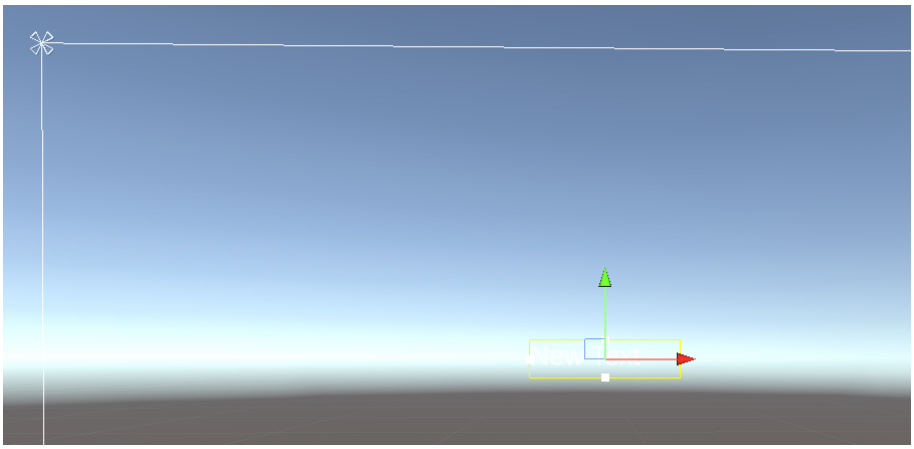


Vom numi acest obiect *ScoreText*.

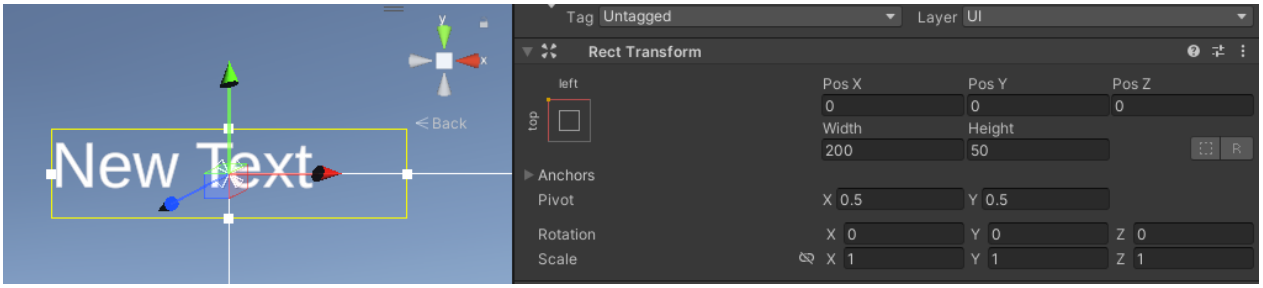
Vom dori să plasăm acest element de UI în partea din stânga-sus a ecranului. Am putea să îl mutăm direct la poziția bună, dar acesta nu va rămâne la locul potrivit în cazul în care fereastra este redimensionată. Asta deoarece momentan, originea sistemului de coordonate folosit de acest element de UI este în centrul ecranului. Orice mișcare a acestuia reprezointă doar o deplasare relativă la centrul ecranului, care nu ia în calcul redimensionarea ferestrei. Pentru a face ca originea sistemului de coordonate a acestui sistem de UI să fie colțul din stânga-sus, vom specifica acest lucru prin intermediul componentei *Rect Transform*.



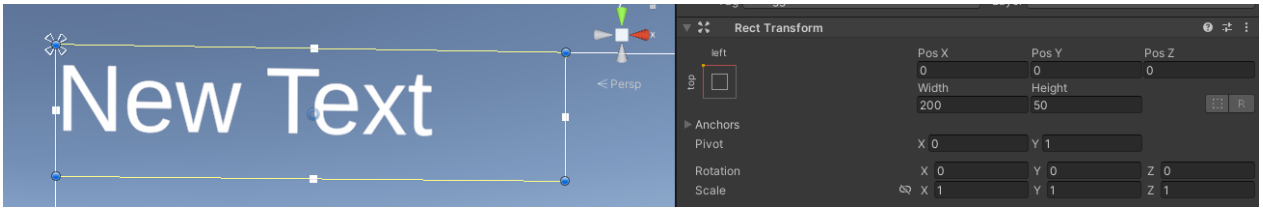
Și în scenă putem observa că originea după care se ghidează acest element de *UI* este colțul din stânga-sus al *Canvas*-ului.



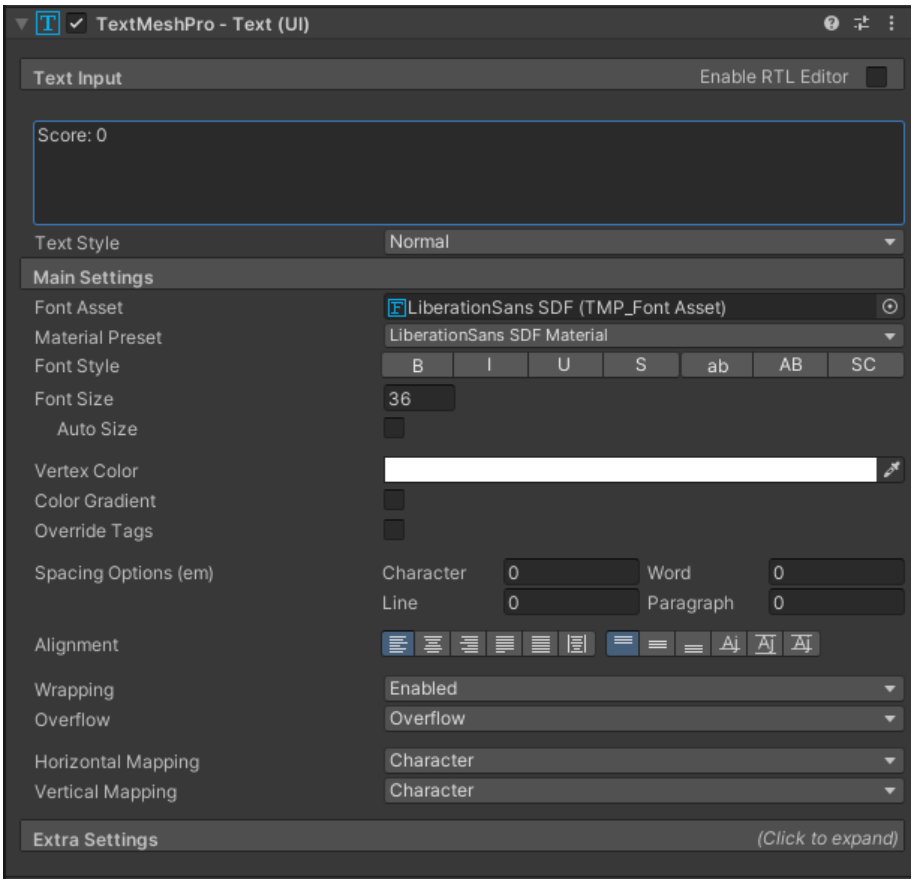
În acest moment, dacă setăm poziția acestui element de *Text* să fie $(0, 0)$ (*Pos X* și *Pos Y* în componenta *Rect Transform*), acesta va fi centrat în colțul din stânga-sus al *Canvas*-ului.



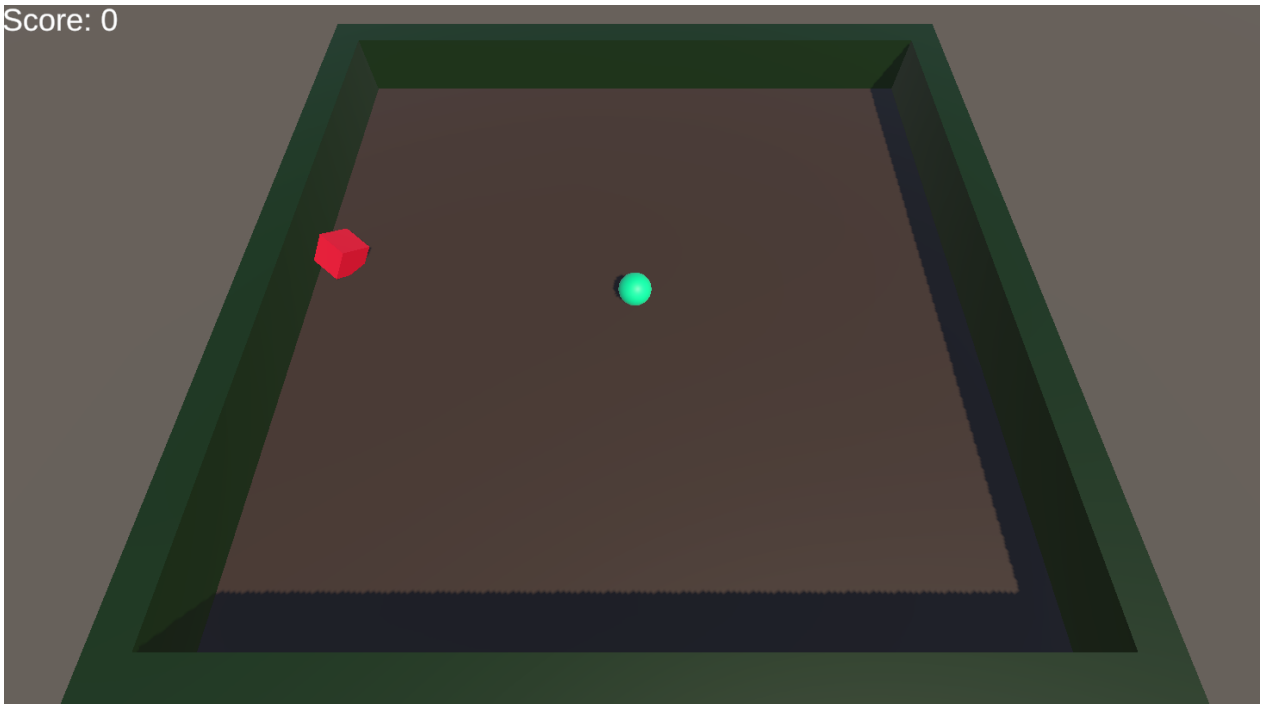
Rezultatul este aproape de cel ideal, totuși, am dori ca nu centrul elementului de *UI* să fie aliniat cu colțul din stânga-sus al *Canvas*-ului ci colțul din stânga-sus al obiectului. Pentru asta, va vi nevoie să setăm ca pivotul elementului să se afle în colțul din stânga-sus. Din *Rect Transform* vom seta valoarea fiel-ului *Pivot* în $(0, 1)$, unde 0 reprezintă marginea din stânga a axei orizontale, iar 1 reprezintă marginea din sus a axei verticale. În scenă nu se va observa nicio modificare, dar în componenta *Rect Transform* se va observa că poziția obiectului nu mai este $(0, 0)$. Dacă schimbăm din nou poziția astfel încât aceasta să fie $(0, 0)$, obiectul va fi aliniat corect.



Din componenta *TextMeshPro − Text (UI)* a acestui obiect putem schimba textul afișat. Vom schimba textul afișat în *Score* : 0.



Acum, dacă rulăm aplicația, în colțul din stânga-sus vom vedea constant scorul 0.

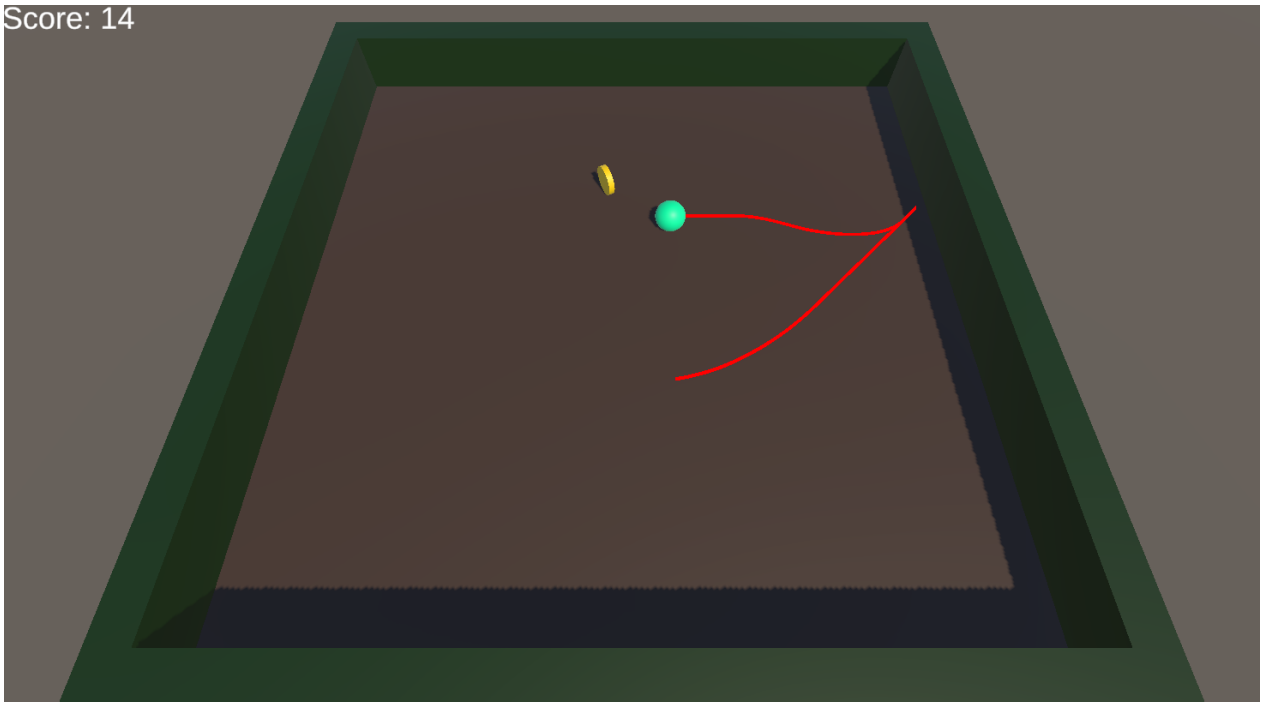


Pentru a actualiza valoarea scorului afișat, în interiorul scriptului *GameController* vom lua o referință

către obiectul de tip *Text*, căreia îi vom actualiza textul atunci când scorul este incrementat.

```
using TMPro;
...
public class GameController : MonoBehaviour
{
    ...
    public int Score
    {
        get => _score;
        set
        {
            _score = value;
            //Debug.Log("Score: " + _score);
            _scoreText.text = "Score: " + _score;
        }
    }
    ...
    [SerializeField]
    private TMP_Text _scoreText;
}
```

Acum, când rulăm jocul, va apărea scorul în partea din stânga-sus a ecranului în loc să apară prin intermediul unor mesaje în consolă.



1.3 Sfârșitul jocului

Momentan, jocul poate fi jucat oricât de mult. Vom face ca jocul să poată fi jucat doar 15 secunde, după care jucătorul să nu mai poată controla bila. Această logică o vom implementa în interiorul clasei *GameController*.

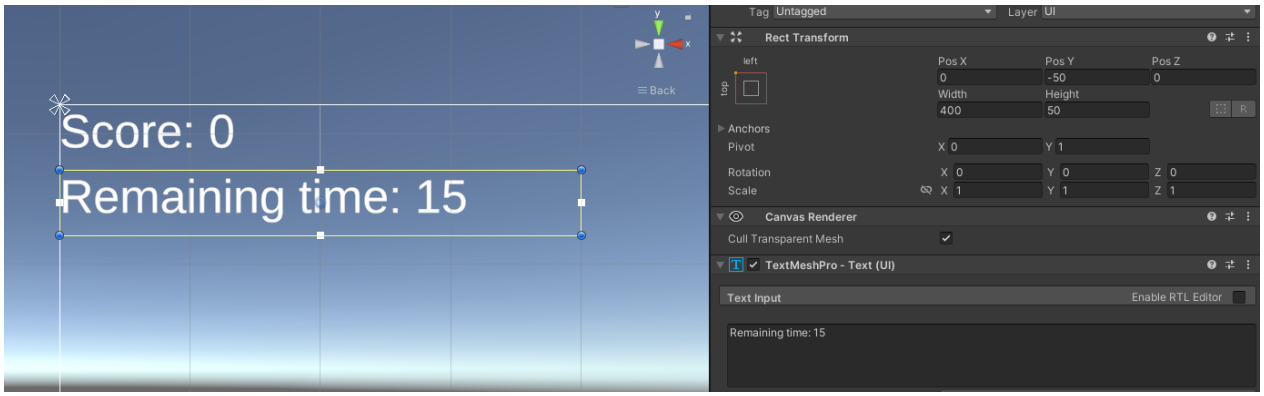
```
public bool GameRunning { get; private set; } = true;
...
[SerializeField]
private float _gameplayTime = 15.0f;
...
private void Update()
{
    _gameplayTime -= Time.deltaTime;

    if (_gameplayTime <= 0.0f)
    {
        _gameplayTime = 0.0f;
        GameRunning = false;
    }

    if (!GameRunning)
        return;

    ...
}
```

În interiorul clasei *MovingSphere* vom dezactiva orice fel de interacțiune din partea jucătorului atunci când proprietatea *GameRunning* are valoarea *false*.

```
private void Update()
{
    if (!GameController.Instance.GameRunning)
    {
        _movement = Vector2.zero;
        if (_movementCoroutine != null)
        {
            StopCoroutine(_movementCoroutine);
            _movementCoroutine = null;
        }
    }
    ...
}
```

1.3.1 Afișarea timpului rămas

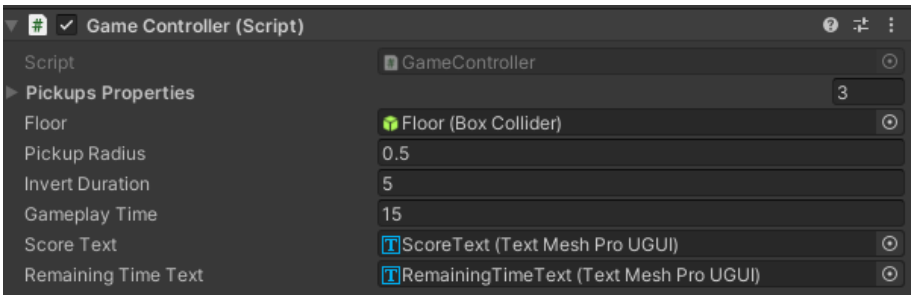
Pe ecran vom afișa timpul rămas până când jocul se va opri. Pentru a face asta vom proceda în același mod în care am procedat în cazul textului pentru scor.



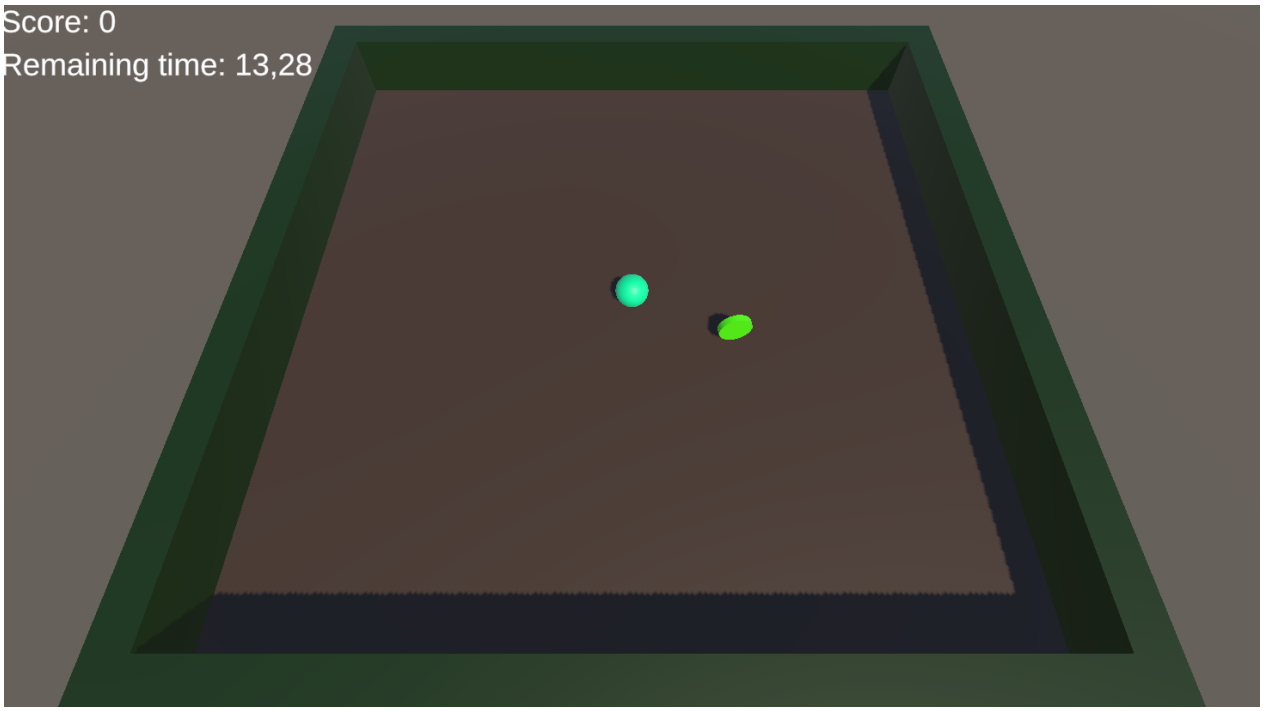
Pentru acest text am schimbat lungimea elementului de UI la 400 pentru ca întregul text să aibă loc pe o singură linie. Pe acesta l-am poziționat fix sub textul pentru scor (poziția -50 pe Y).

În *GameController* vom actualiza timul afișat de către acest element de tip *Text*. Vom afișa timpul de joc rămas cu două zecimale în interiorul clasei *GameController*.

```
...
[SerializeField]
private TMP_Text _remainingTimeText;
...
private void Update()
{
    ...
    _remainingTimeText.text = $"Remaining time: {_gameplayTime:F2}";
    ...
}
...
```



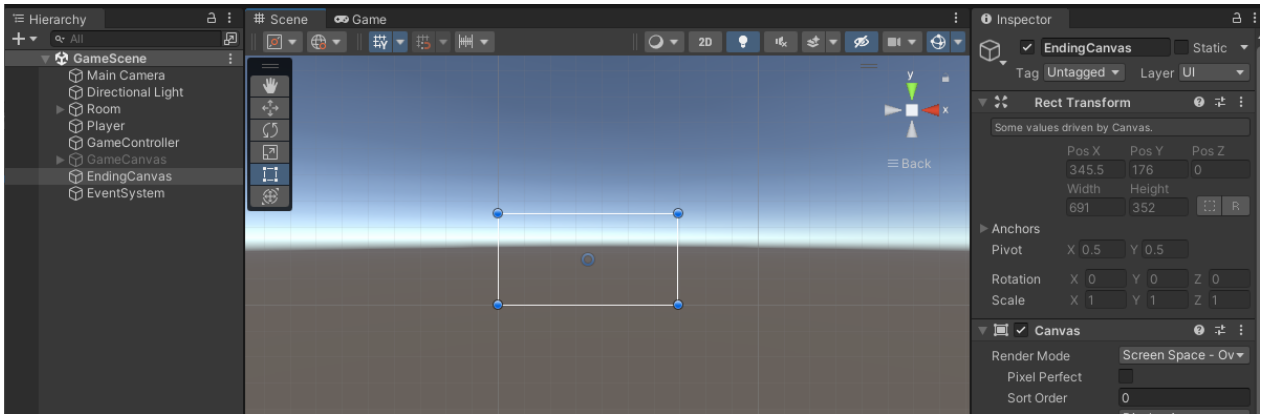
Dacă pornim jocul, putem observa că timpul rămas este afișat fix sub scor.



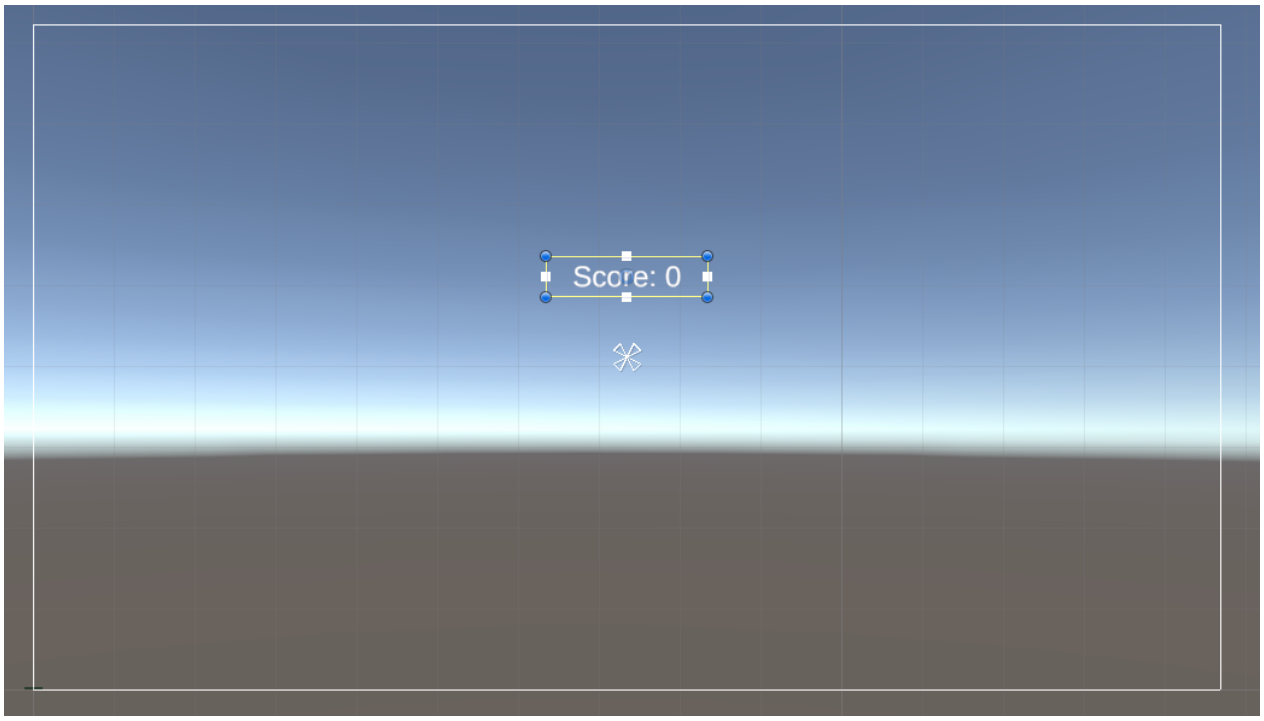
1.3.2 Ecran de sfârșit

După ce jocul se termină, vom dori să afișăm un alt ecran în interfața grafică. Un ecran care să conțină scorul jucătorului în mijlocul ecranului și alte două butoane. Un buton de *Replay* și altul de revenire la meniul principal (pe care urmează să-l adăugăm).

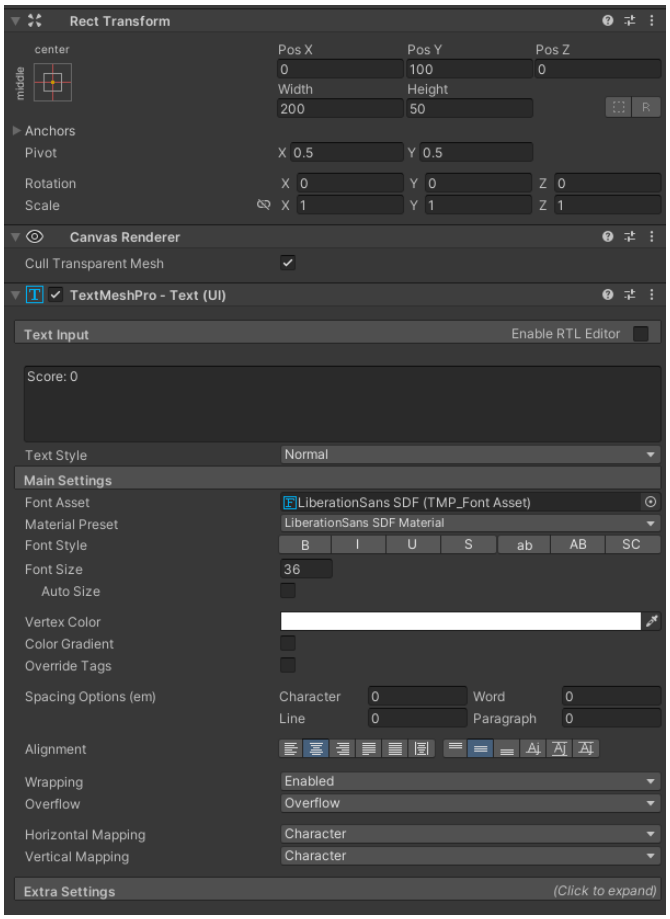
Vom începe prin a adăuga un nou *Canvas* numit *EndingCanvas* căruia îi vom defini elementele. Pentru a ne fi ușor să lucrăm cu mai multe *Canvas*-uri, vom face ca singurul canvas activ în scenă să fie cel asupra căuria lucrăm, dezactivându-le pe celelalte. În cazul acesta, vom activa *Canvas*-ul *EndingCanvas* și vom dezactiva *Camvas*-ul *GameCanvas*.



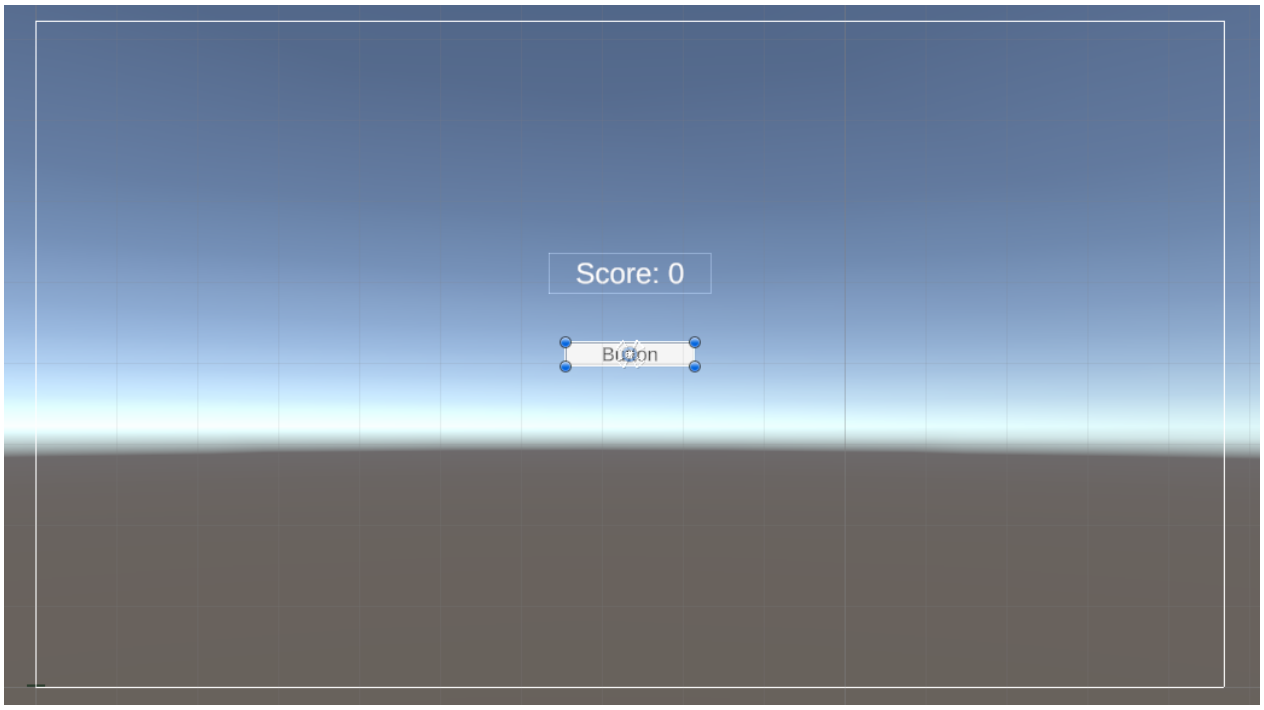
Vom defini elementele acestui *Canvas* începând cu obiectul care afișează scorul final. Acesta va fi adăugat similar cu cele adăugate anterior. Acest element de *UI* vom dori să fie poziționat relativ la centrul ecranului, deci nu îi vom modifica ancora sau pivotul.



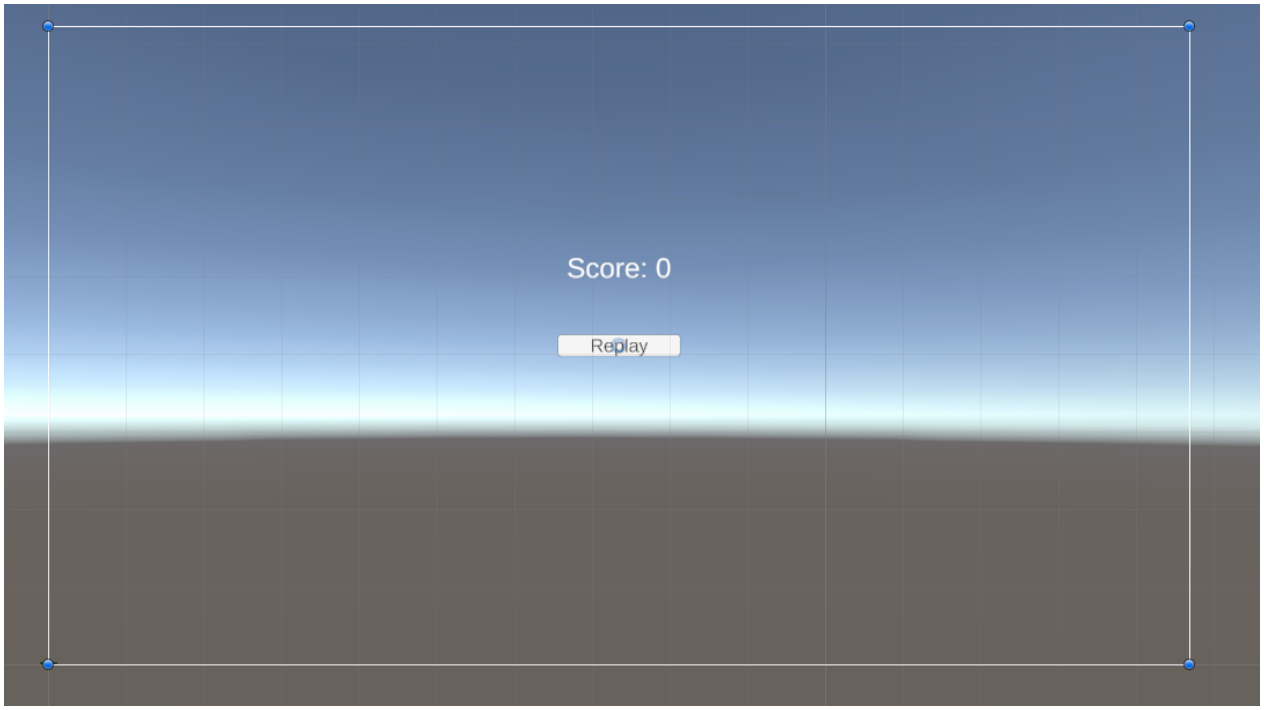
Am mutat acest obiect la poziția 100 de-alungul axei Y și am modificat modul în care este aliniat textul, astfel încât acesta să fie aliniat pe centru. Valorile din inspector pentru acest obiect:



Urmează adăugarea butoanelor în acest *Canvas*. Prima dată vom adăuga butonul de *Replay*. Pentru a adăuga acest buton, vom proceda în felul următor: *click dreapta pe EndingCanvas/UI/Button – TextMeshPro*. Vom numi acest buton *ReplayButton*.

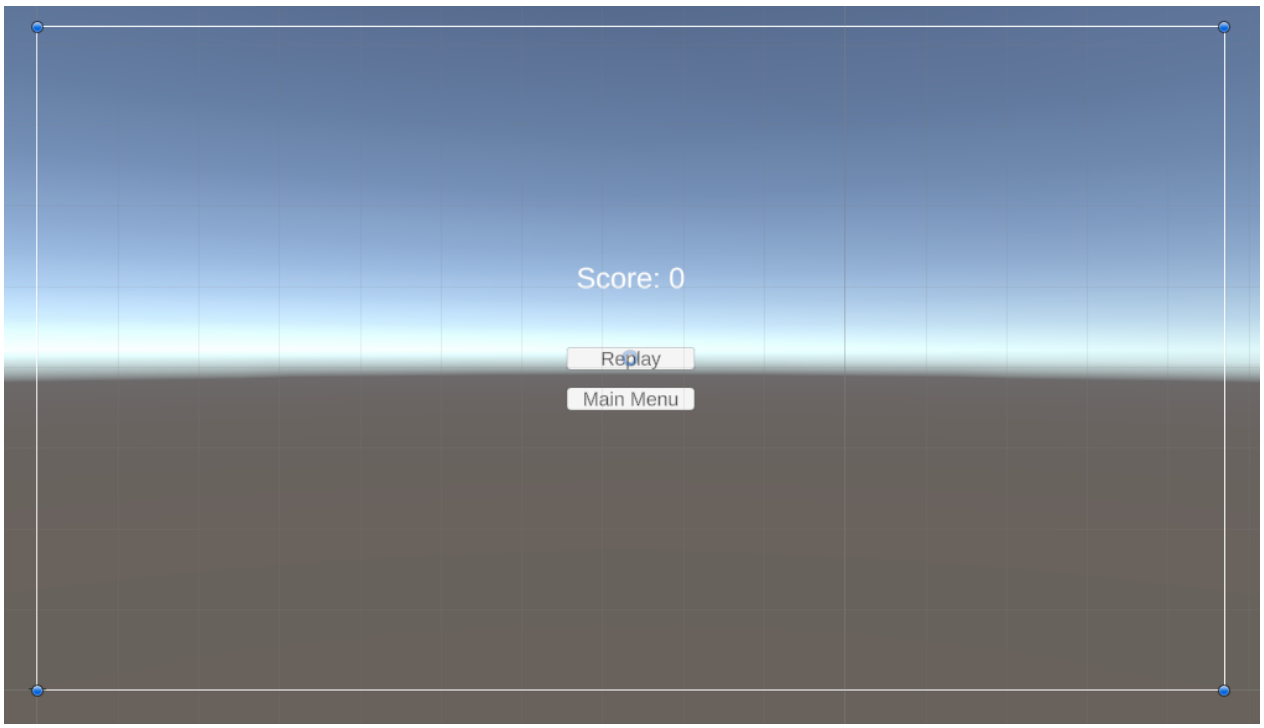


Dacă ne uităm în fereastra *Hierarchy* observăm că acest obiect are un copil care este un obiect de tip *Text*. Prin acest obiect putem schimba textul afișat deasupra butonului. Vom schimba textul în *Replay*.



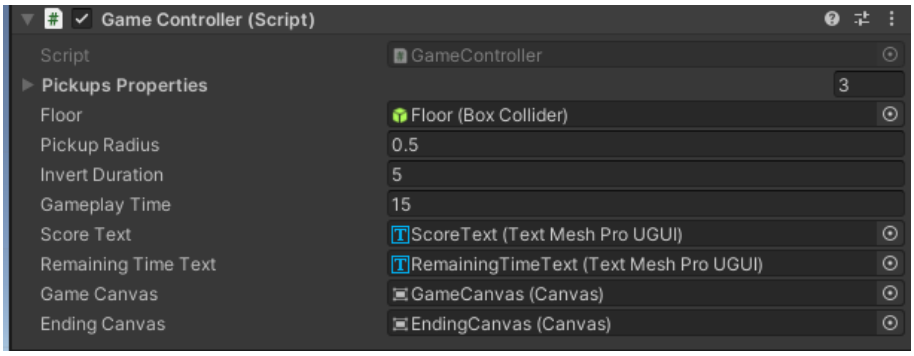
Vom mai adăuga un buton în scenă numit *MainMenuButton* căruia îi vom schimba textul în *Main Menu*

și pe care îl vom poziționa sub butonul creat anterior.



După ce am definit structura acestui ecran, vom dezactiva *Canvas*-ul *EndingCanvas* și vom activa înapoi *Canvas*-ul *GameCanvas*. Vom păstra referințe către amândouă *Canvas*-urile în clasa *GameController* și le vom activa/ dezactiva în funcție de caz.

```
...
[SerializeField]
private Canvas _gameCanvas;
[SerializeField]
private Canvas _endingCanvas;
...
private void Update()
{
    _gameplayTime -= Time.deltaTime;

    if (_gameplayTime <= 0.0f)
    {
        _gameplayTime = 0.0f;
        if (GameRunning)
        {
            _gameCanvas.gameObject.SetActive(false);
            _endingCanvas.gameObject.SetActive(true);
            GameRunning = false;
        }
    }
    ...
}
...
```

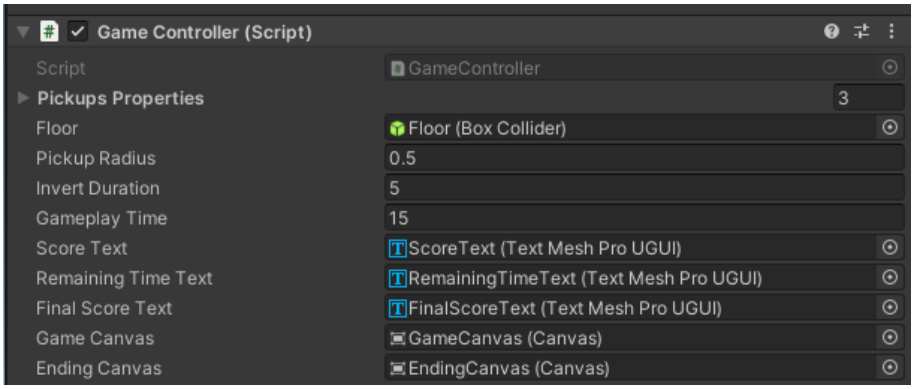


1.4 Funcționalitățile ecranului de sfârșit

1.4.1 Scorul final

Prima oară vom dori să afișăm scorul final în ecranul de sfârșit. Vom proceda la fel ca în cazul afișării scorului în joc.

```
...
[SerializeField]
private TMP_Text _finalScoreText;
...
private void Update()
{
    ...
    if (_gameplayTime <= 0.0f)
    {
        ...
        if (GameRunning)
        {
            ...
            _finalScoreText.text = "Score: " + _score;
            GameRunning = false;
        }
    }
    ...
}
...
```

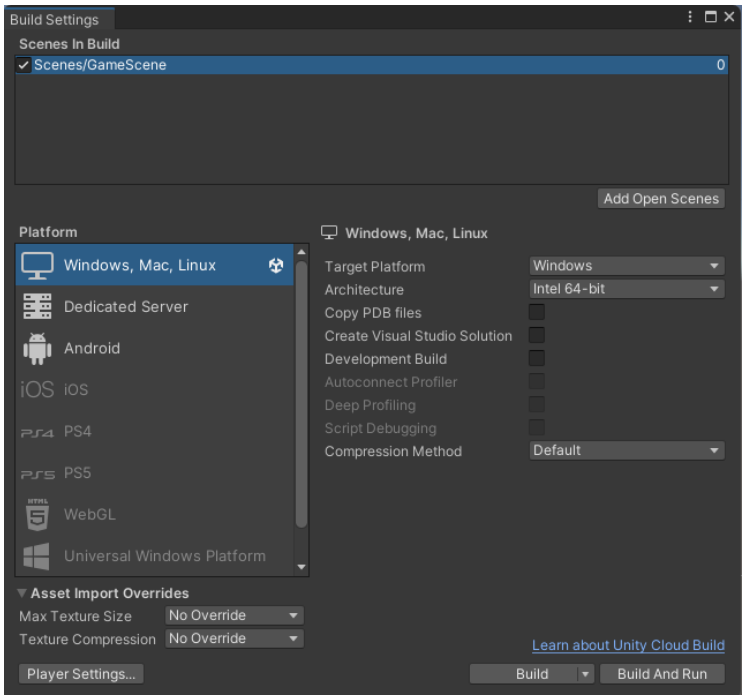


Dacă rulăm jocul, când pierdem vom observa scorul corect afișat în mijlocul ecranului.

1.4.2 Funcționalitățile butoanelor

Vom dori ca butonul *Replay* să reseteze scena curentă, iar butonul *Main Menu* să schimbe scena.

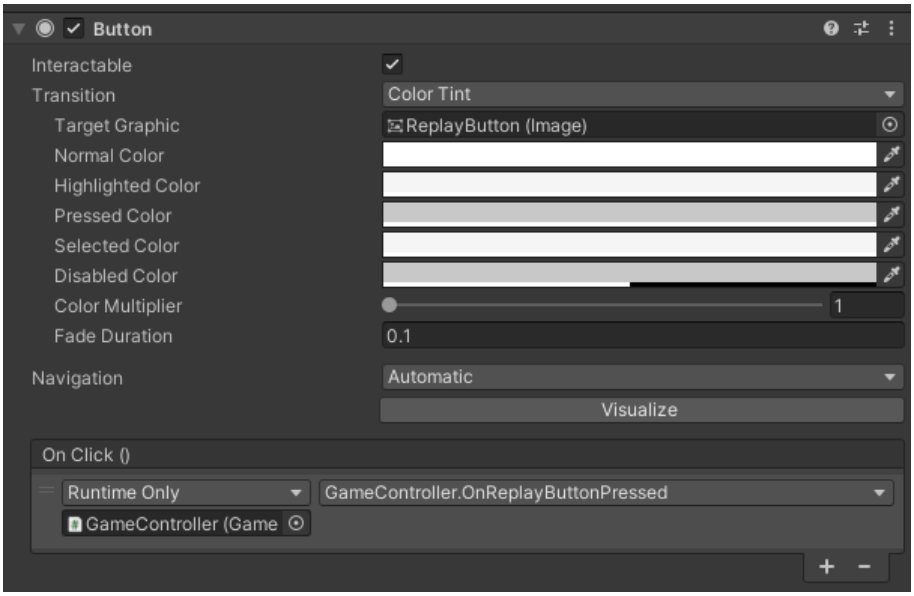
În Unity, pentru a referenția o scenă se folosește fie un ID, fie numele acesteia sub formă de șir de caractere. În primul rând vom atribui un ID scenei curente. Pentru a face asta, vom deschide scena (în cazul în care aceasta nu este deschisă), iar apoi vom urma următorii pași: *File/Build Settings/Add Open Scenes*. Asta va atribui un ID scenei curente și o va include în versiunea compilată a jocului. Dacă este prima scenă adăugată, va avea ID-ul 0.



Acum, putem defini funcționalitatea butonului *Replay*. În *GameController* vom defini o nouă metodă publică numită *OnReplayButtonPressed* care fa încărca din nou scena curentă.

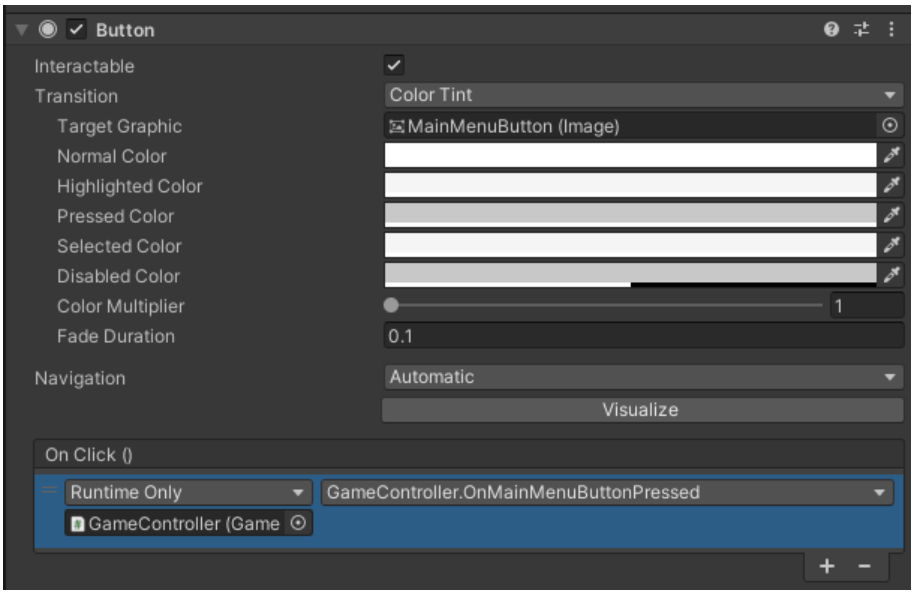
```
...
using UnityEngine.SceneManagement;
...
public class GameController : MonoBehaviour
{
    ...
    public void OnReplayButtonPressed() =>
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
    ...
}
```

Această metodă o vom referenția în componenta *Button* a obiectului *ReplayButton*, similar cu referențierile metodelor pentru *Input* din laboratoarele anterioare (se referențiază prima oară obiectul *GameController* iar apoi se selectează metoda *OnReplayButtonPressed*).



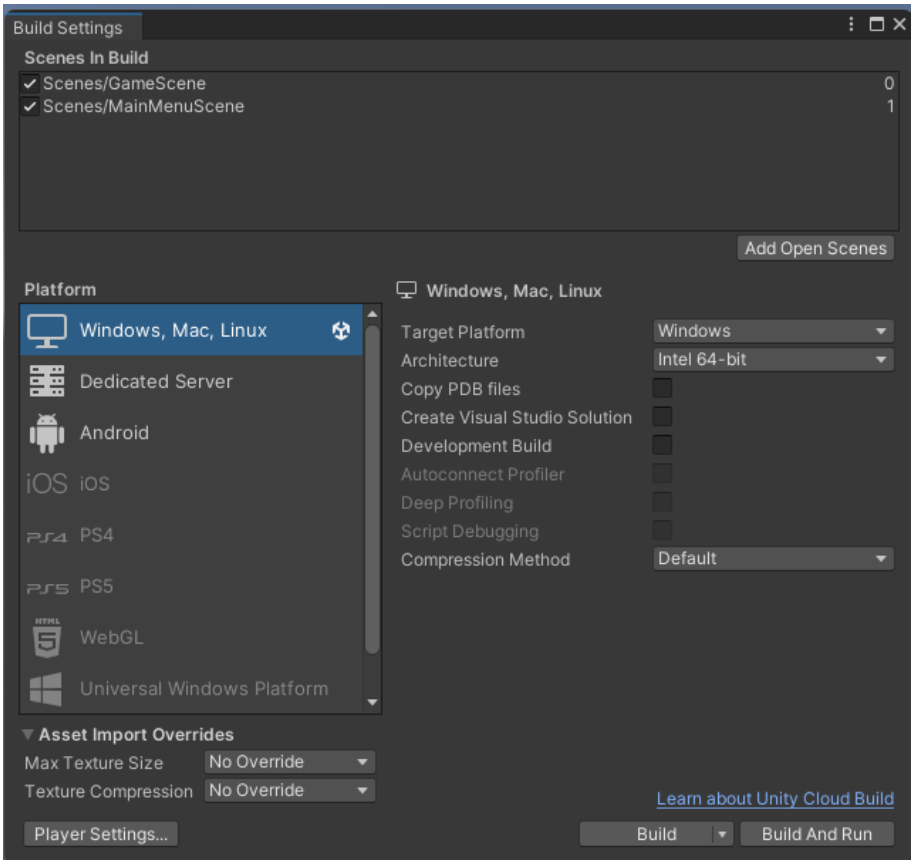
Vom adăuga o metodă și pentru butonul *Main Menu*. Acesta va încărca o scenă numită *MainMenuScene*. De data aceasta, referențierea o vom face prin nume, ci nu prin *ID* ca în cazul butonului de *Replay*.

```
...
public void OnMainMenuButtonPressed() =>
    SceneManager.LoadScene("MainMenuScene");
...
```

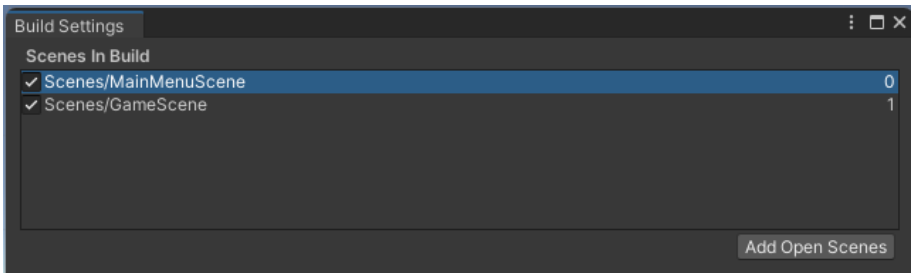


Tot ce rămâne de făcut este să definim scena *MainMenuScene* și să îi atribuim un *Build Index*.

În directorul *Scenes* vom crea o nouă scenă numită *MainMenuScene*. Vom deschide scena respectivă și îi vom atribui un *ID* cum am făcut și în cazul scenei anterioare.



Dorim ca ID-ul acestei scene să fie egal cu 0 pentru ca aplicația compilată să pornească din această scenă. Pentru a schimba ID-ul, putem schimba ordinea scenelor din *Scenes In Build* folosind drag and drop.

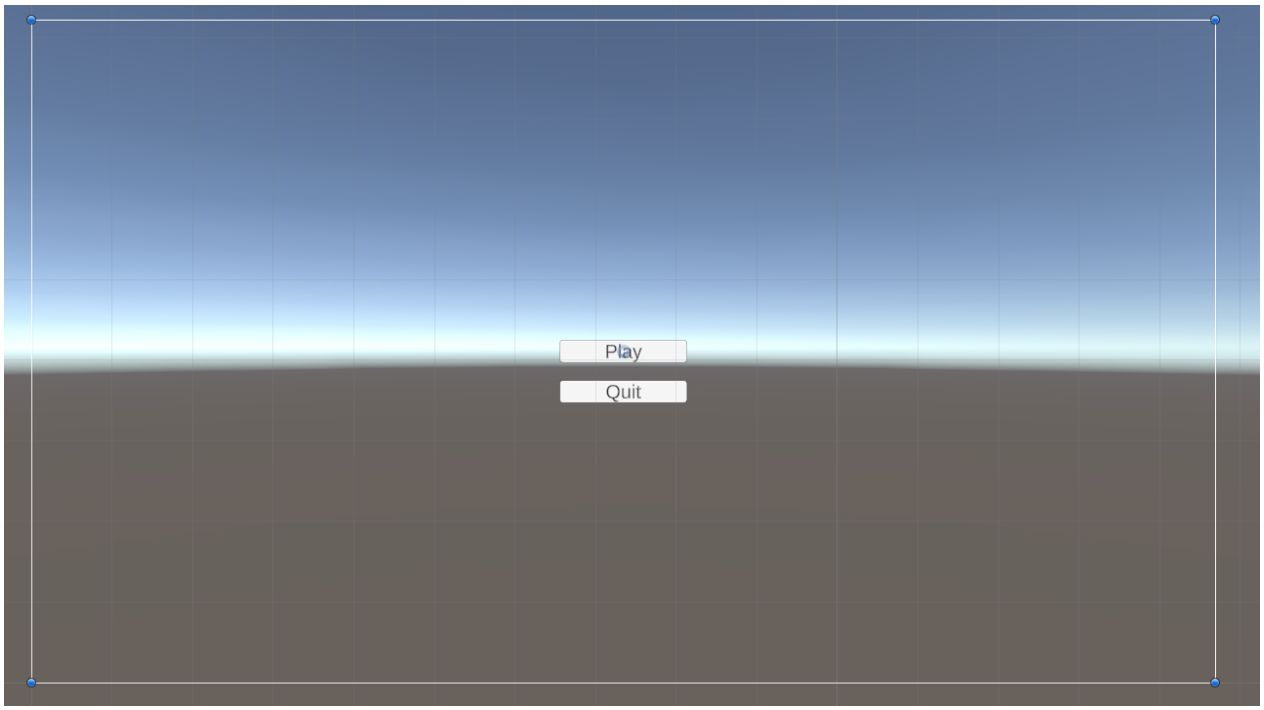


Dacă intrăm din nou pe scena *GameScene*, acum întregul meniu de sfârșit al jocului ar trebui să funcționeze.

1.5 Meniul principal

Vom intra din nou în scena *MainMenuScene* și vom adăuga în aceasta două butonae: Un buton de *Play* care să pornească jocul și un buton de *Quit* care să închidă aplicația.

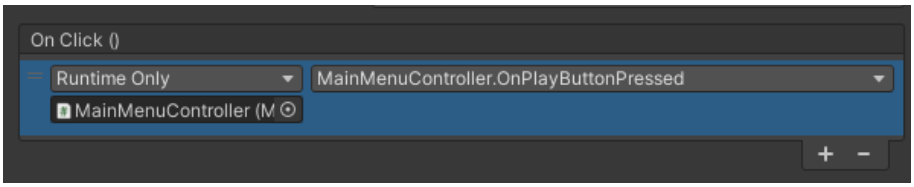
Vom proceda ca în cazul butoanelor anterioare.

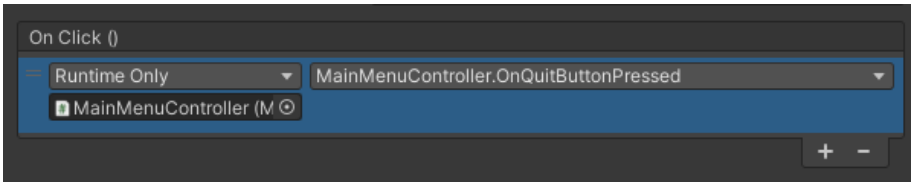


În această scenă vom adăuga un nou obiect gol numit *MainMenuController* căruia îi vom atrbui un nou script numit *MainMenuController* în interiorul căruia vom defini metodele butoanelor.

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class MainMenuController : MonoBehaviour
{
    public void OnPlayButtonPressed() =>
        SceneManager.LoadScene("GameScene");

    public void OnQuitButtonPressed() =>
        Application.Quit();
}
```

Vom referenția aceste metode.





Butonul *Play* va funcționa așa cum trebuie când dăm îl apăsăm din *Unity*. Totuși, butonul *Quit* nu va funcționa atunci când este apăsat din *Unity*. Acesta funcționează doar dacă se rulează o versiune compilată a jocului. Pentru a rula o versiune compilată a jocului se procedează în felul următor: *File/Build Settings/se selectează platforma pe care doriți să rulați/Build And Run/se alege destinatia unde se va afla executabilul*.

2 Exerciții

2.1 Implementați funcționalitatea de a pune pauză jocului.

2.2 Jucați-vă cu elementele de UI pentru a face ca interfețele grafice implementate în acest laborator să arate mai bine.