Laboratorul 1

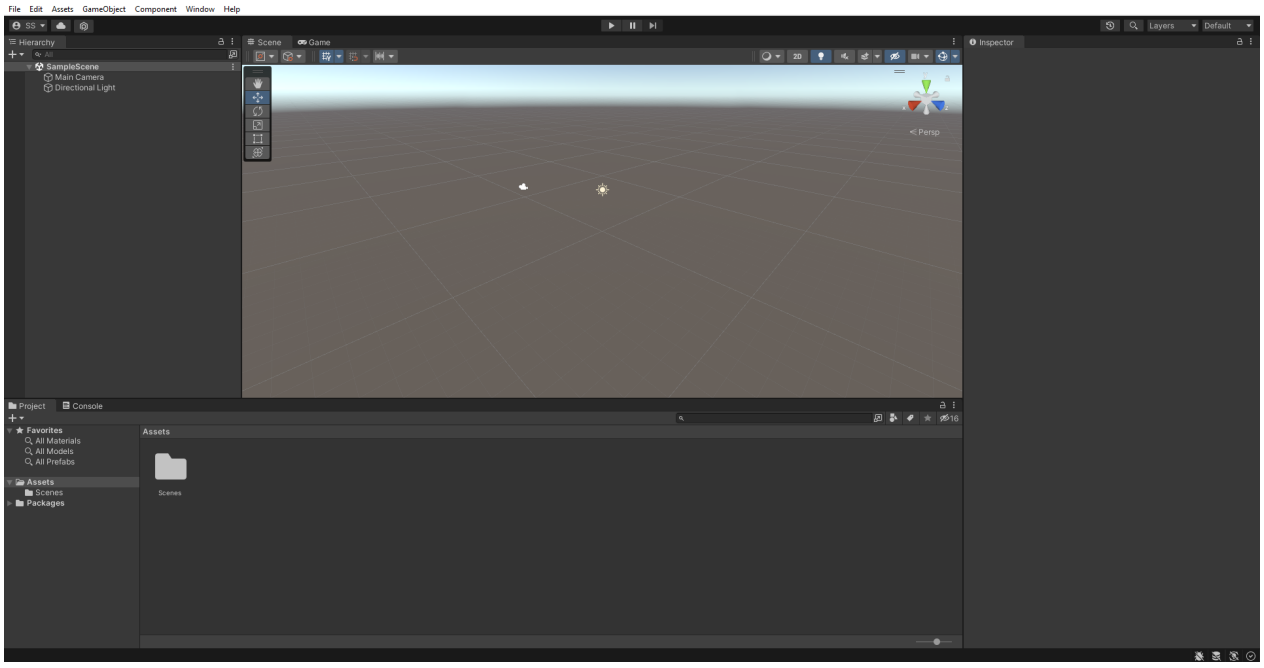
1 Crearea unui proiect

1.1 Proiect nou

În cadrul laboratoarelor vom folosi *Unity*. Preferabil este să se folosească ultima versiune de *Unity* care este *LTS* (*Long Term Support*). Aceste versiuni sunt recomandate și de către *Unity*.

Pentru a instala *Unity* și a crea proiecte în *Unity* se folosește de regulă *Unity Hub*. După ce ați instalat o versiune de Unity, dați click pe *New Project* pentru a crea un proiect nou. În partea stângă a *Unity Hub* va apărea un meniu de unde putem selecta template-ul folosit în crearea proiectului. În cadrul acestor laboratoare vom folosi template-ul $3D$ (*Core*). În partea dreaptă a ferestrei trebuie specificate numele și calea unde va fi salvat proiectul. După ce detaliile au fost selectate/ completate, apăsați pe *Create Project* pentru a crea proiectul.

După ce ați creat un proiect, ar trebui să vă apară o ferastră care arată în felul următor:

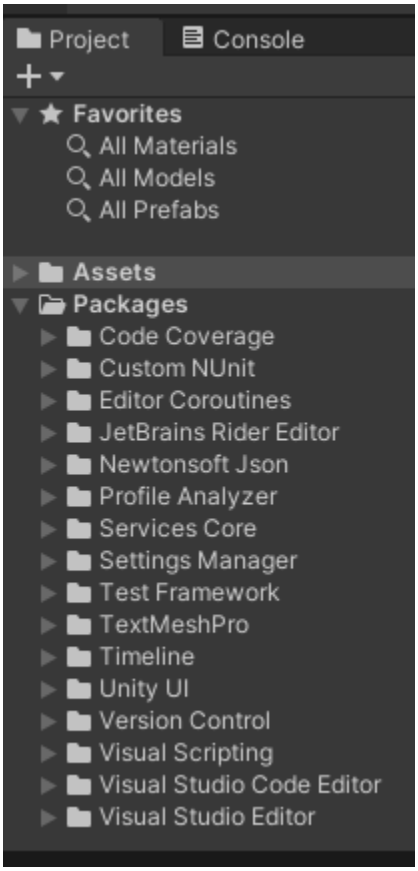


Vom examina layout-ul acestui editor după ce vom termina de configurat noul proiect. De reținut este faptul că layout-ul este cusomizabil. Ferestrele ce alcătuiesc layout-ul pot fi redimensionate, mutate și închise/ deschise. Există mai multe layout-uri predefinite în *Unity* (se pot schimba din Window/Layouts).

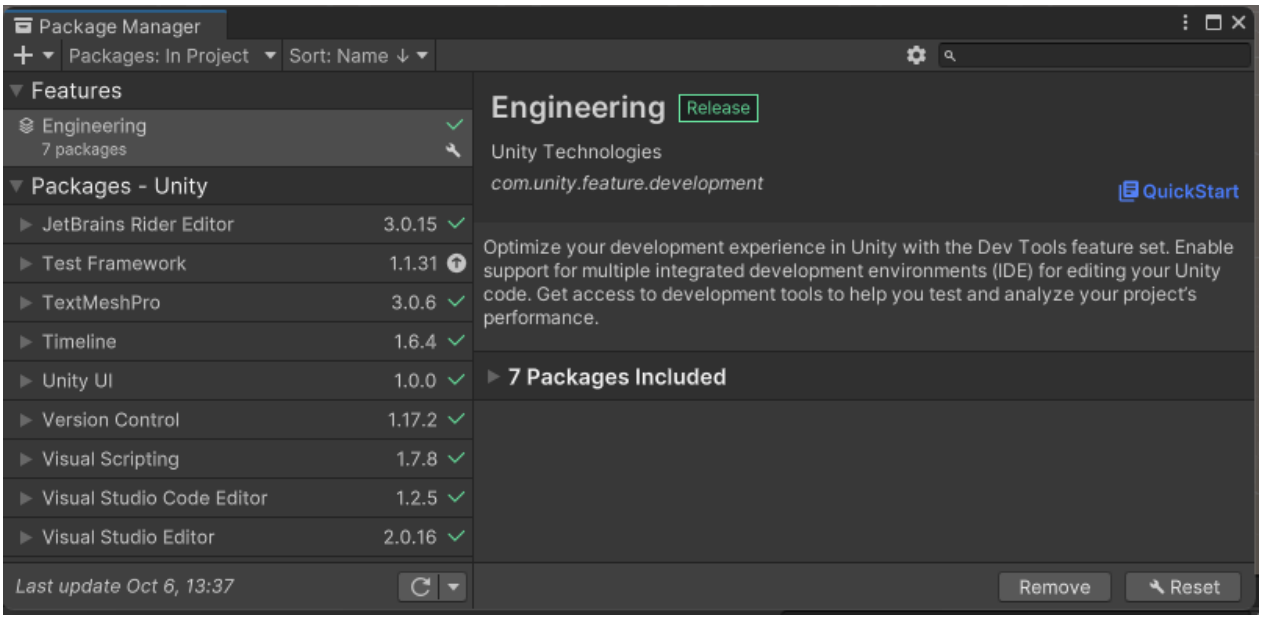
1.2 Pachete

Unity este un engine foarte modular. Are funcționalități de bază incluse, dar pentru multe task-uri avem nevoie de pachete externe. Când un proiect nou este creat, *Unity* adaugă automat câteva pachete în proiect.

Acestea se pot vedea în fereastra *Project* din partea de jos a editorului, sub categoria *Packages*.



Pentru a adăuga, elimina sau actualiza pachete se folosește *Package Manager*-ul din *Unity*, accesibil în felul următor: *Window/Package Manager*.



Când accesați *Package Manager*-ul pentru prima dată, veți vedea pachetele incluse în proiect. Ele adaugă funcționalități în plus pe lângă funcționalitățile de bază ale Unity. De exemplu, pachetul *JetBrains Rider Editor* face ca fișierele de cod din cadrul acestui proiect să poată fi editate în *IDE*-ul (*Integrated Development Environment*) *JetBrains Rider*, iar *Intellisense*-ul pentru funcțiile și clasele din *Unity* să funcționeze în acel *IDE*. În cadrul laboratorului puteți folosi orice *IDE*/ *Code Editor* vreți. Eu voi folosi *IDE*-ul *Visual Studio* 2022.

1.3 URP (Universal Render Pipeline)

Pentru a desena obiecte pe ecran, Unity urmează mai mulți pași (sortează obiectele în funcție de adâncime, îi spune plăcii video să deseneze mai multe obiecte printr-un singur apel, etc..). Acești pași se numesc *Render Pipeline*.

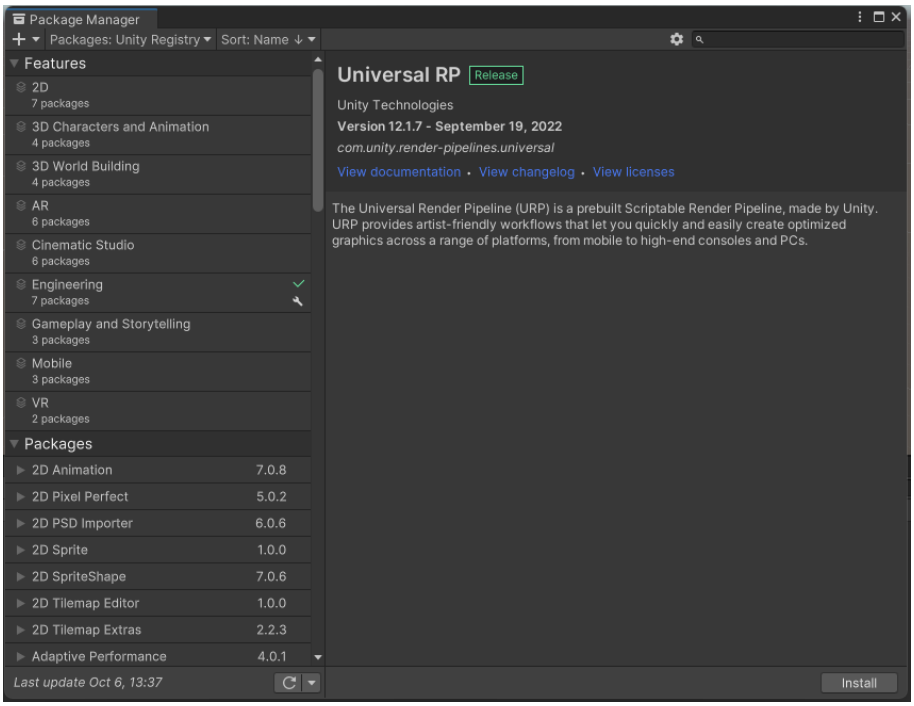
Implicit, *Unity* folosește un *Render Pipeline* care este destul de inconsistent și care nu prea mai este întreținut de către *Unity*. Recent, *Unity* a introdus posibilitatea ca programatorii să își scrie propriile *Render Pipeline*-uri. Totodată, au apărut două *Render Pipeline*-uri scrise și întreținute de către *Unity*. Cele două *Render Pipeline*-uri sunt *URP* (*Universal Render Pipeline*) și *HDRP* (*High Definition Render Pipeline*).

URP este un *Render Pipeline* lightweight care este menit să funcționeze pe orice dispozitiv, indiferent de placa video. Calitatea vizuală oferită de *URP* nu se poate ridica la nivelul jocurilor AAA, dar pentru pentru obiectivele acesui laborator, rezultatele vizuale sunt suficient de bune.

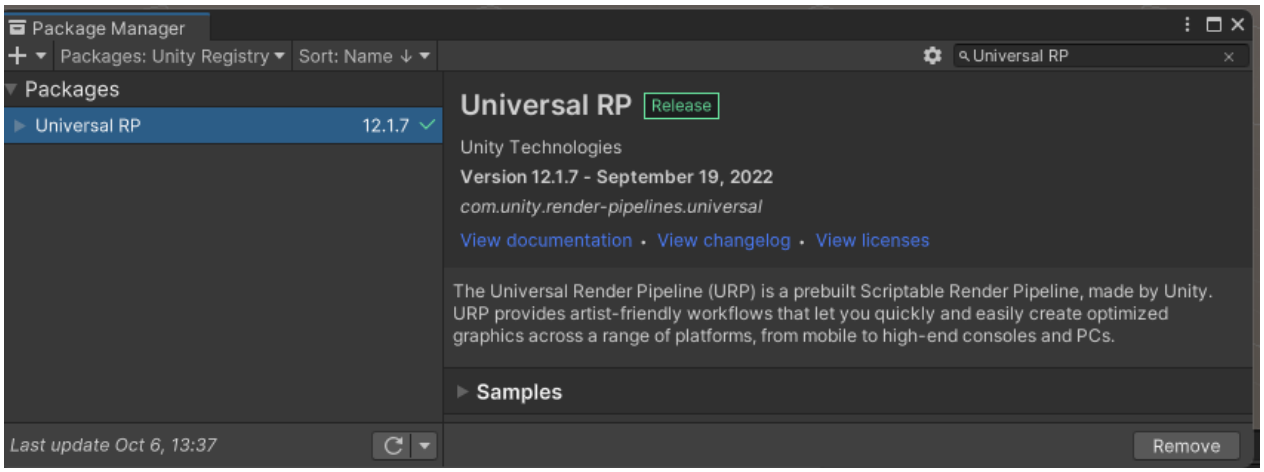
HDRP este un *Render Pipeline* complex, menit pentru jocurile care necesită o înaltă calitate vizuală. *HDRP* nu este recomandat pentru jocuri de telefon sau alte dispozitive mai puțin capabile.

La laborator se va folosi *URP*. Pentru proiecte nu este necesar să se folosească *URP*, dar este recomandat.

Pentru a instala *URP* în proiect vom folosi *Package Manager*. Din *Package Manager* trebuie selectat ca acesta să afișeze pachetele din registrul Unity. Pentru a face asta, se apasă pe butonul *Packages* : *In Project* din stânga-sus a ferestrei, iar de acolo se selectează *Unity Registry*.

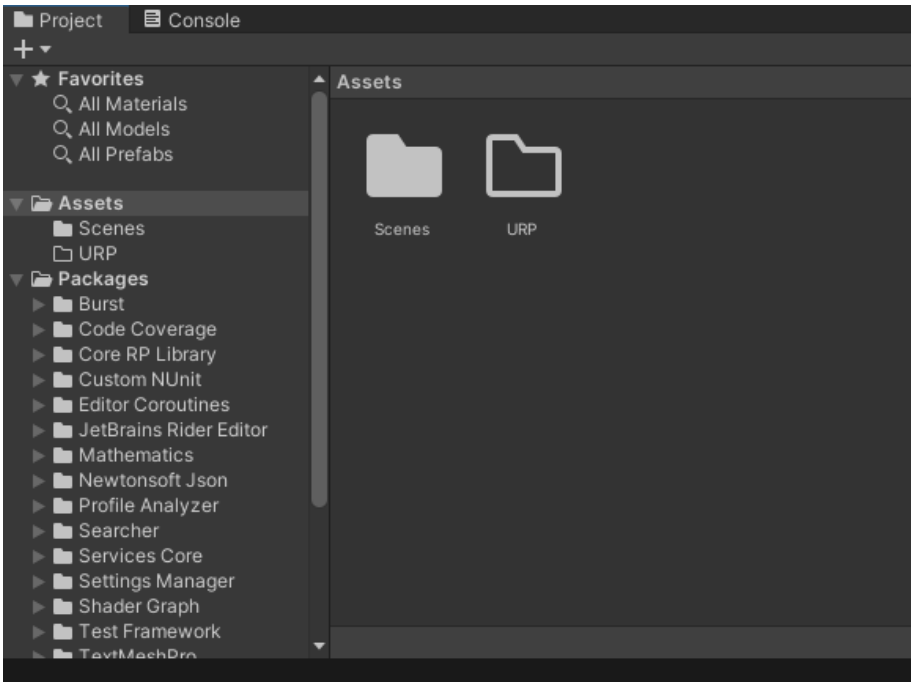


În partea din dreapta-sus a ferestrei există un buton de search. Folosiți acel buton pentru a căuta *Universal RP*. După ce apare pachetul în partea stângă a ferestrei, selectați-l și apoi apăsați pe butonul *Install* din colțul din dreapta-jos a ferestrei.

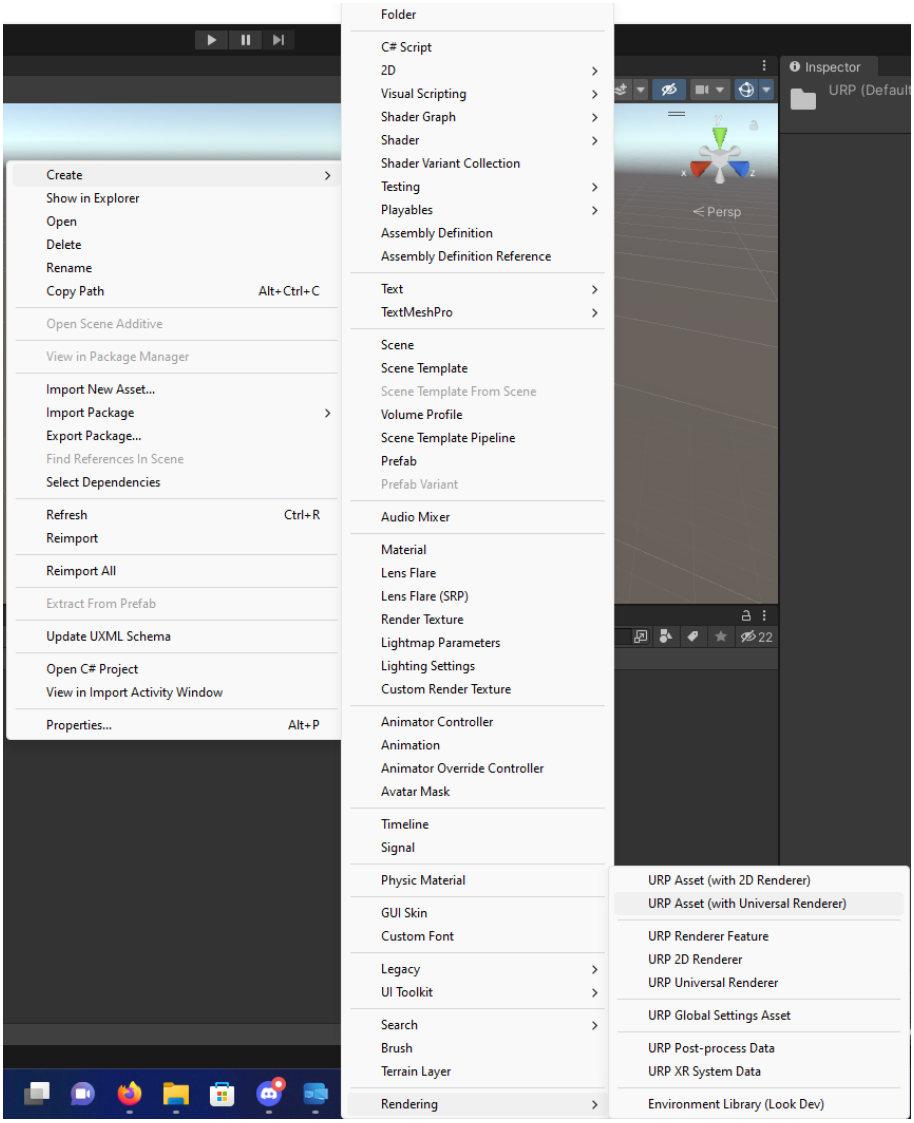


După ce *URP* s-a instalat, este nevoie de a crea un fișier care să conțină parametrii folosiți de URP în cadrul proiectului.

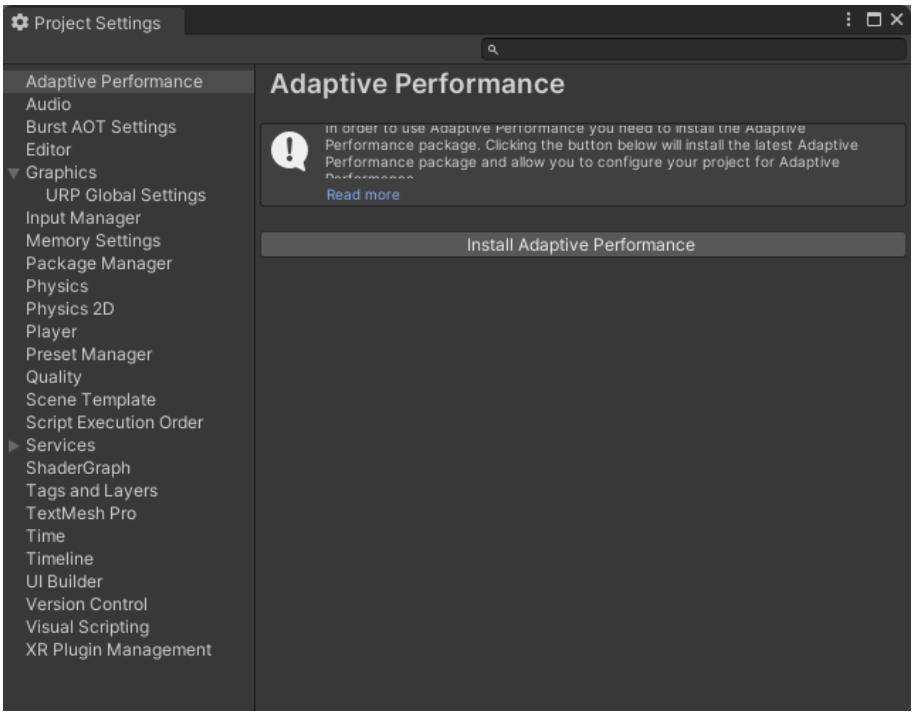
În interiorul directorului *Assets* vom crea un director nou numit *URP*. Dați click pe directorul *Assets* din fereastra *Project* pentru a accesa acel director. Pentru a crea un folder nou, din fereastra *Project* apăsați *click dreapta/Create/Folder*. Numiți acest director *URP*.



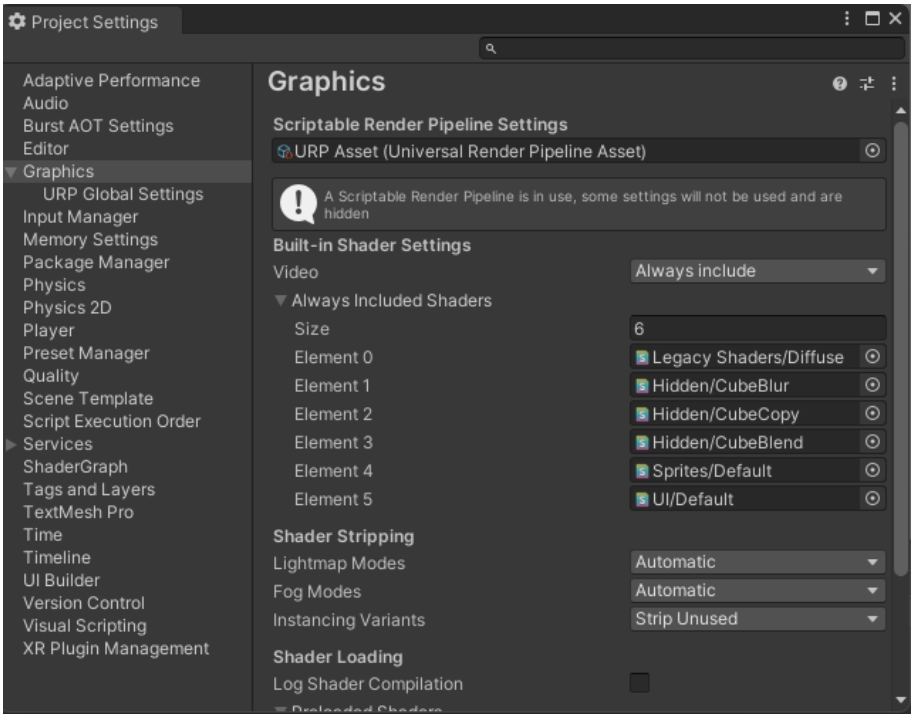
în interiorul acestui folder adăugați un fișier de tipul *URP Asset (with universal renderer)*. Pentru a face acest lucru apăsați *click dreapta/Create/Rendering/URP Asset (with universal renderer)*. Denumiți fișierul *URP Asset*.



După ce asset-ul a fost creat, acesta trebuie specificat în setările proiectului. Întrați în setările proiectului din tab-ul *Edit* (*Edit/Project Settings*). Se va deschide următoarea fereastră:



Din această fereastră vom alege categoria *Graphics*. În partea dreaptă ne va apărea un field numit *Scriptable Render Pipeline Settings*. În acel field trebuie selectat fișierul creat anterior.

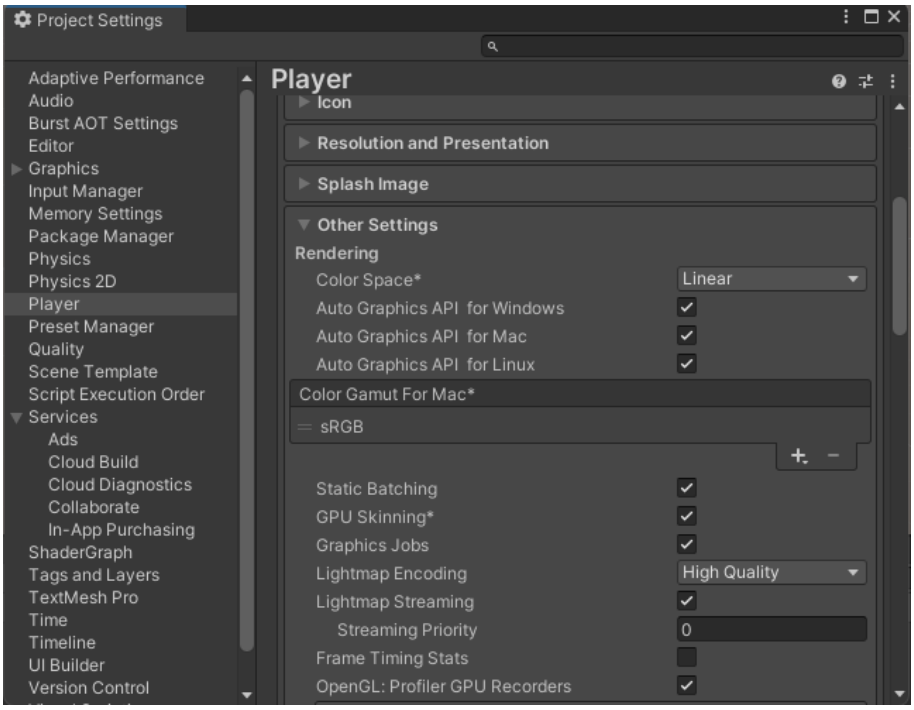


1.4 Alte setări

1.4.1 Spațiul de culori

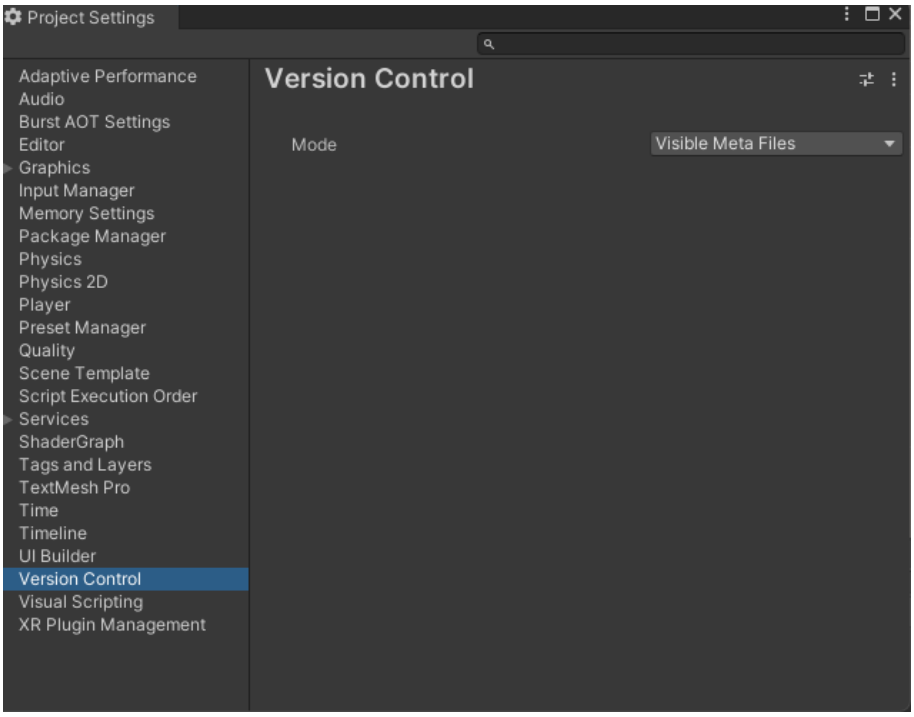
Unity suporta două tipuri de spații de culori: *Gamma* și *Linear*. Monitoarele folosesc un spațiu de culori de tip *Gamma*, deci prin urmare, atunci când scena este randată, nu se efectuează calcule în plus în acest spațiu de culori. Modul *Linear* necesită calcule în plus, dar poate produce rezultate mai bune. Pentru aceste laboratoare vom folosi spații de culoare de tip *Linear*.

Pentru a seta spațiul de culori folosit de proiect, se accesează în *Project Settings/ Player/Other Settings/Color Space*∗. Asigurați-vă că acel field este setat pe *Linear*.

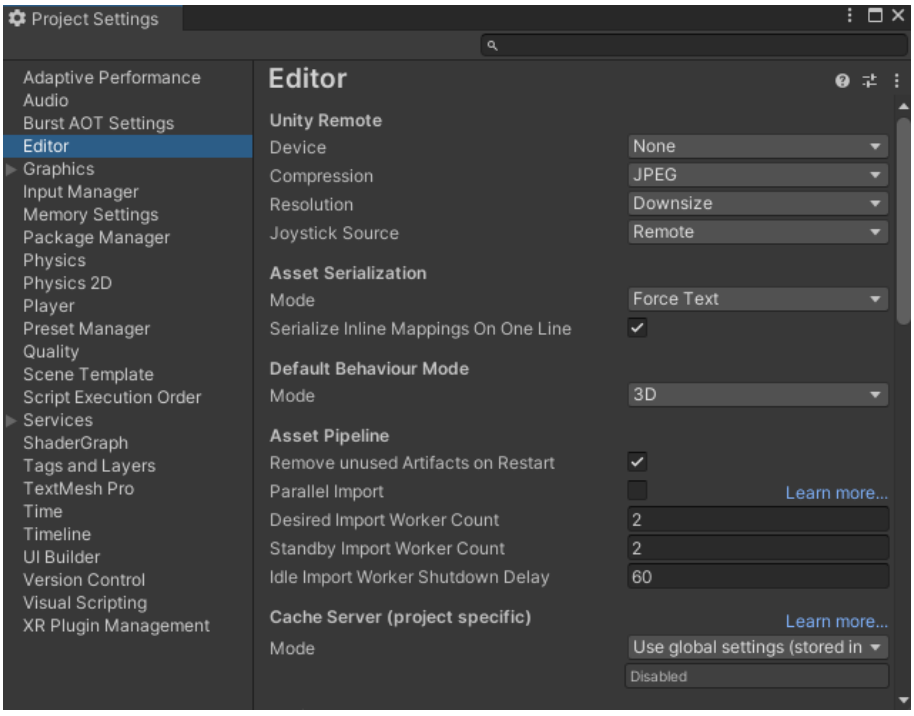


1.4.2 git

Pentru a putea folosi *git* asupra unor proiecte în *Unity*, trebuie ca anumite lucruri să fie setate în *Project Settings*. În primul rând, setarea *Version Control Mode* trebuie să aibă valoarea *Visible Meta Files* (*Project Settings/Version Control/Mode*).

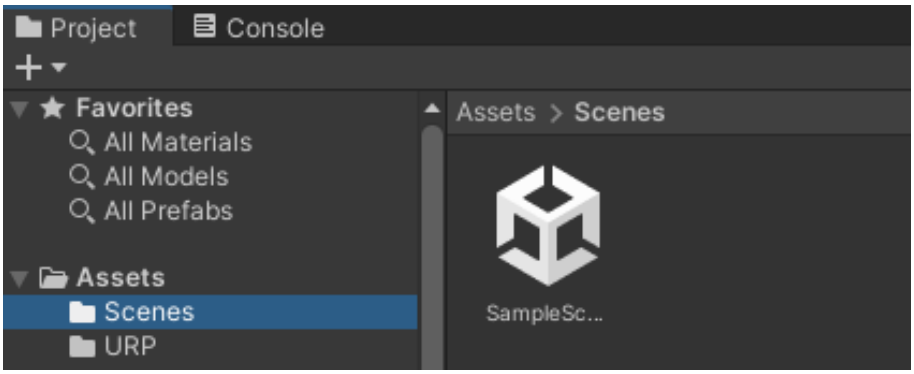


De asemenea, trebuie ca valoarea *Asset Serialization Mode* să aibă valoarea *Force Text*. Setarea este accesibilă din *Project Settings/Editor/Asset Serialization/ Mode*.



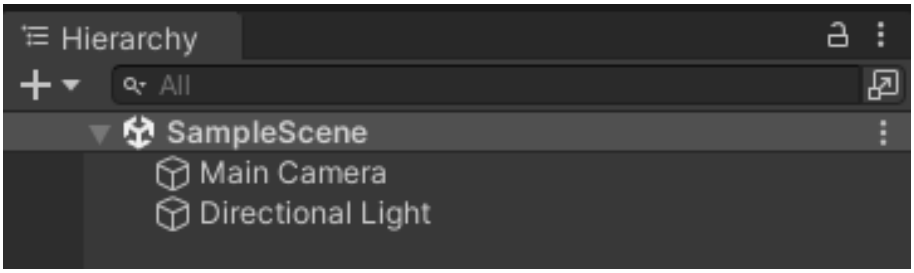
2 Scene

Când un proiect este creat, se adaugă automat și o scenă de bază. Aceasta se află în directorul *Assets/Scenes*:

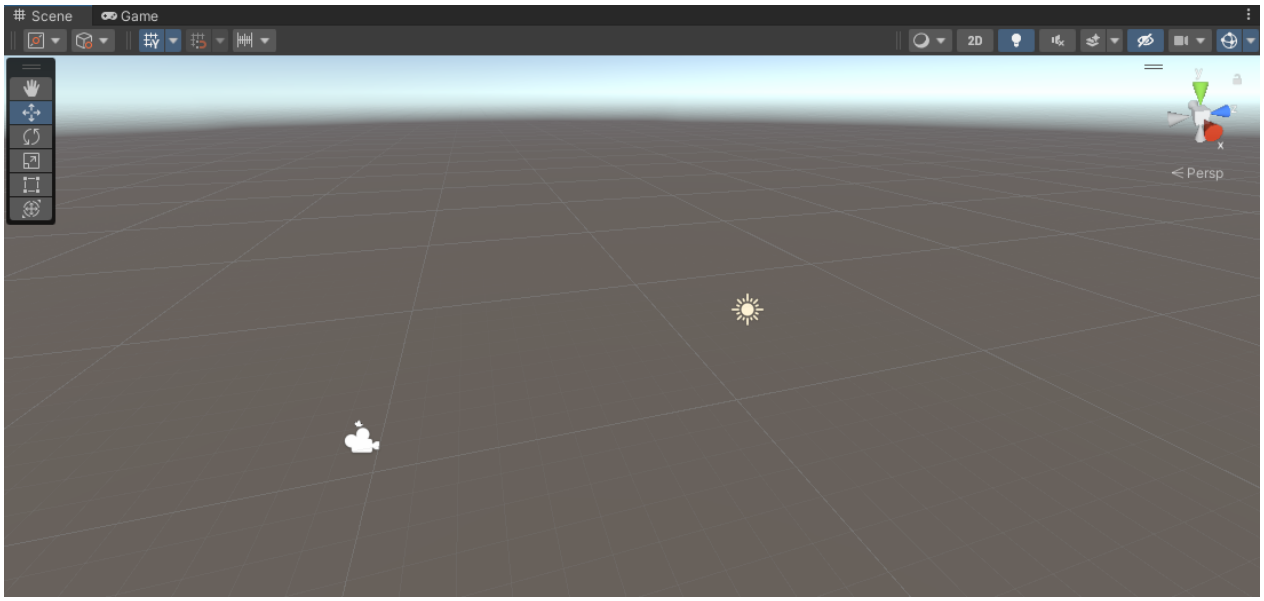


Această scenă este și deschisă automat de către editor implicit.

În partea stângă a editorului se află fereastra *Hierarchy*, aceasta conține conține toate obiectele care se află în scena deschisă.



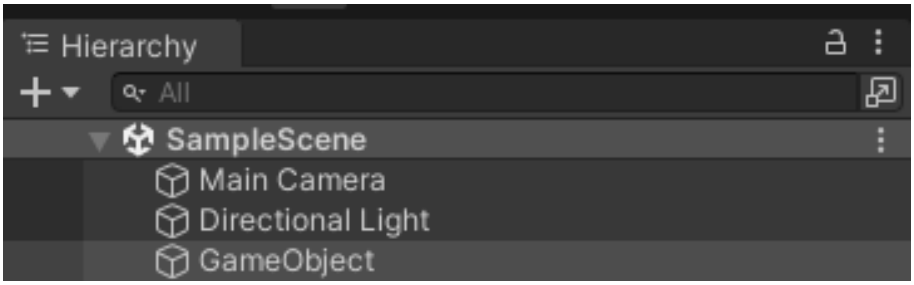
Pentru a selecta un obiect, se poate folosi atât fereastra *Hierarchy* cât și fereastra *Scene* din centrul editorului. Scena implicită conține două obiecte: *Main Camera* și *Directional Light*. Acestea sunt vizibile și în editor. *Main Camera* este un obiect de tip cameră, iar *Directional Light* este o lumină direcțională. În fereastra *Scene* acestea sunt reprezentate de o iconiță cu o cameră de filmat, respectiv o iconiță cu un soare.



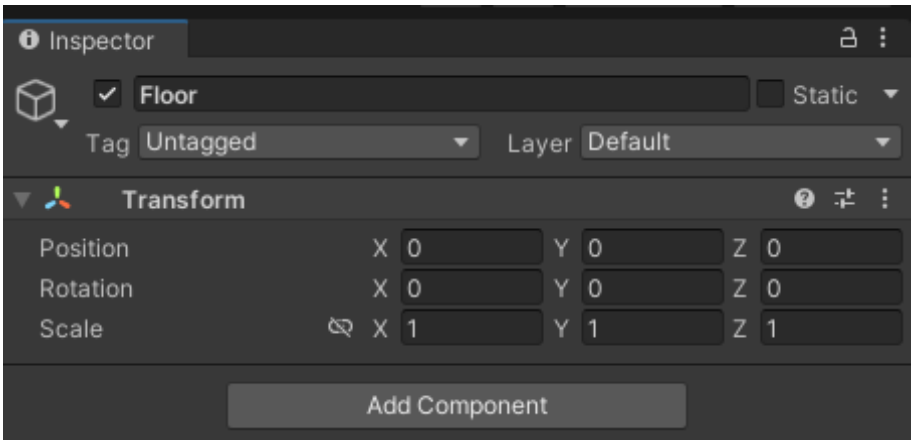
Pentru a naviga ușor în fereastra *Scene* se poate ține apăsat click dreapta peste scenă, concomitent cu folosirea butoanelor WASD de pe tastatură împreună cu mișcări ale mouse-ului. Rotița mouse-ului se poate folosi pentru a te apropia/ îndepărta în scenă.

3 Obiecte

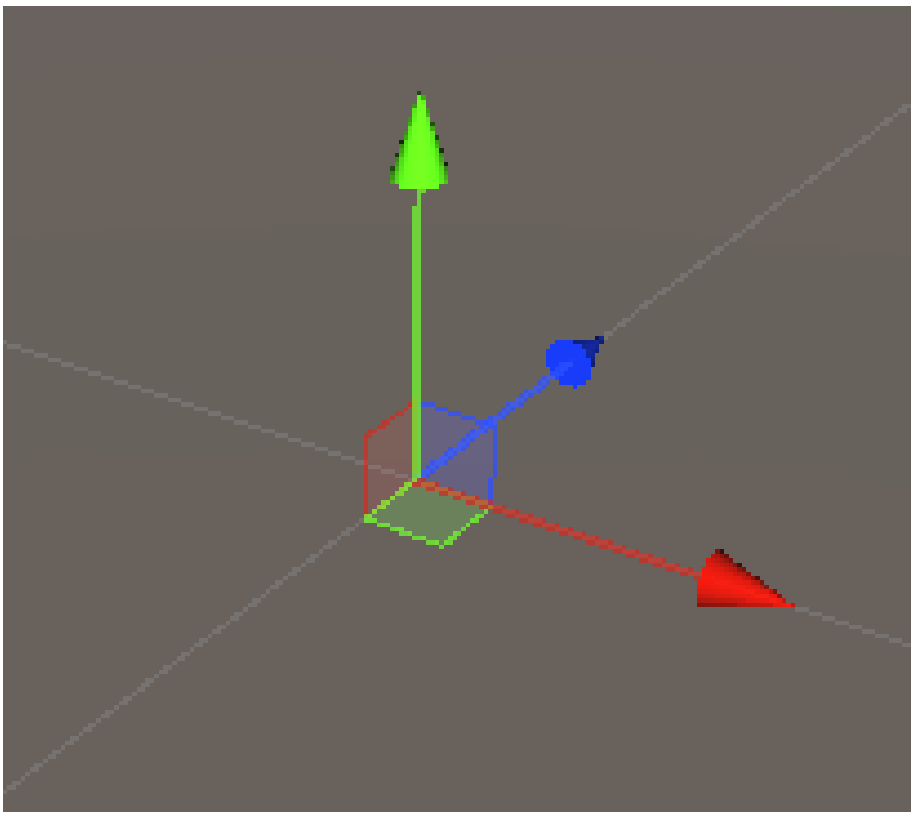
Pentru a adăuga obiecte în scenâ din fereastra *Hierarchy* în felul următor: *click dreapta/Create Empty*. De asemenea, se poate adăuga un obiect nou și prin *GameObject/Create Empty*. Această acțiuni vor crea un obiect nou.



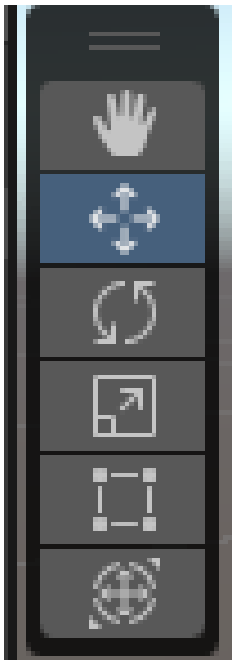
Dacă dați click pe acest obiect, el va fi selectat, iar în fereastra *Inspector* din partea dreaptă a ecranului vor apărea opțiuni de configurare ale acestui obiect. Vom redenumi obiectul nostru *Floor*.



Când este selectat, obiectul este vizibil și în scenă, unde poate fi repoziționat folosind săgețile atașate acestuia.



Obiectul poate fi și rotit/ scalat din fereastra *Scene*. Implicit, modul de manipulare este cel de manipulare a poziției. Acest mod poate fi schimbat din meniul din partea din stânga-sus a ferestrei *Scene*.



4 Componente

Comportamentul obiectelor este determinat de componente. Pentru a vedea componentele unui obiect, acesta trebuie selectat, iar apoi lista componentelor sale va apărea în fereastra *Inspector*.

4.1 Componenta Transform

Toate obiectele din *Unity* au o componentă numită $Transform$. Această componentă nu poate fi eliminată. Prin intermediul acestei componente se modifcă poziția, rotația și scara obiectelor.

Vom selecta obiectul *Floor*, iar din valorile componentei $Transform$ îi vom modifica poziția. Se observă că obiectul se mișcă și în fereastra *Scene* atunci când modificăm valorile poziției din componenta $Transform$.

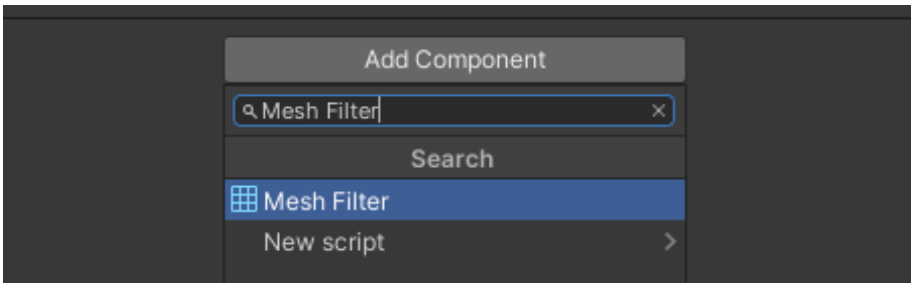
Pentru a aduce obiectul la valorile implicite ale componentei $Transform$, se poate apăsa pe cele 3 puncte din partea dreaptă a componentei, iar apoi pe butonul *Reset*.

4.2 Alte componente

Deși putem face multe cu un obiect prin intermediul componentei $Transform$, obiectul *Floor* nu este vizibil pentru că nu are componente care să specifice modul în care obiectul va fi afișat pe ecran.

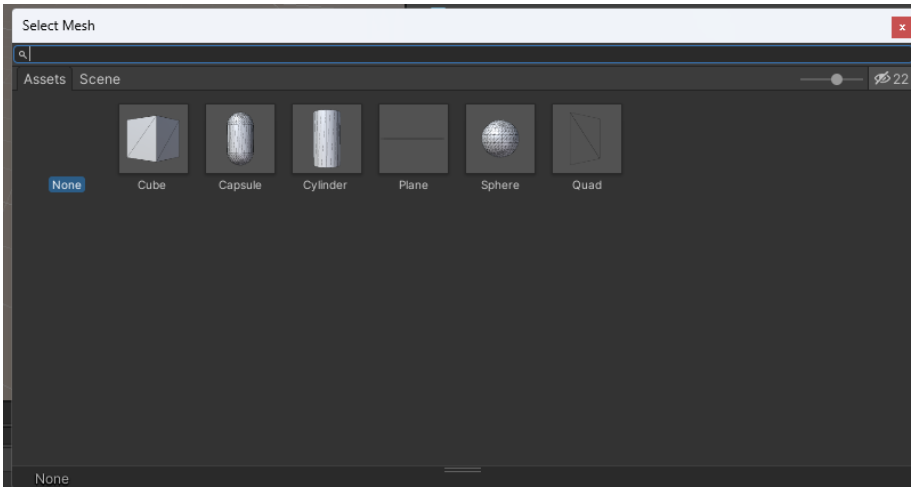
4.2.1 Mesh Filter

Vom adăuga o componentă de tipul $Mesh\ Filter$. Pentru a face asta trebuie dat click pe butonul *Add Component* din inspector. Se va deschide o listă cu componente în care putem căuta componenta $Mesh\ Filter$.



După ce adăugm această componentă putem observa că are o proprietate numită *Mesh* care are valoarea *None* (*Mesh*).

Pentru a schimba valoarea acestei proprietăți putem apăsa pe cerculețul din partea dreaptă a valorii. Se va deschide această fereastră:



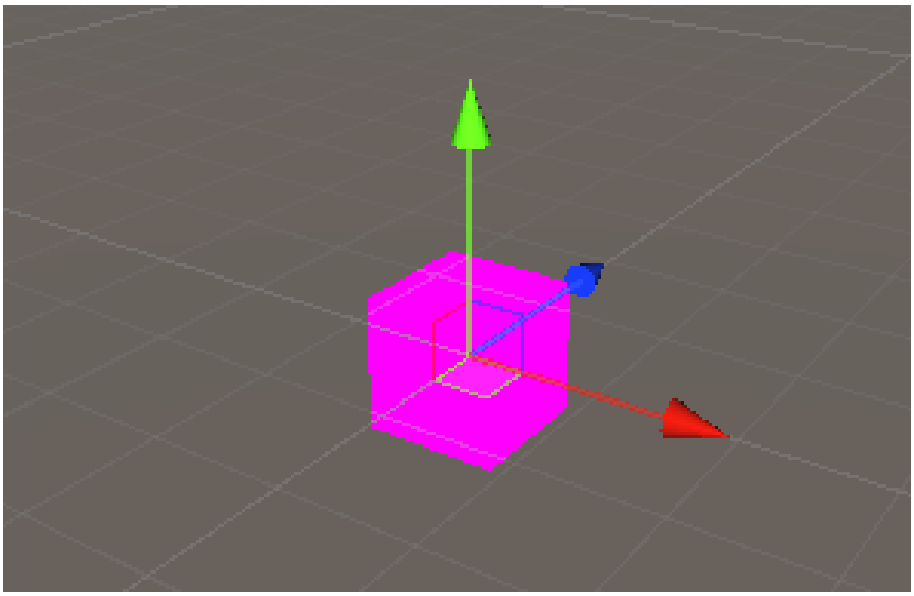
De aici vom selecta asset-ul *Cube* folosind dublu click.

Obiectul încă nu este vizibil în scenă, deoarece componenta *Mesh Filter* specifică doar geometria obiectului, nu și modul în care acesta trebuie afișat pe ecran.

4.2.2 Mesh Renderer

Vom adăuga o componentă de tipul *Mesh Renderer* asupra obiectului *Floor*. Această componentă se adaugă în același mod în care s-a adăugat componenta *Mesh Filter*.

Obiectul acum apare în scenă, dar este complet roz.



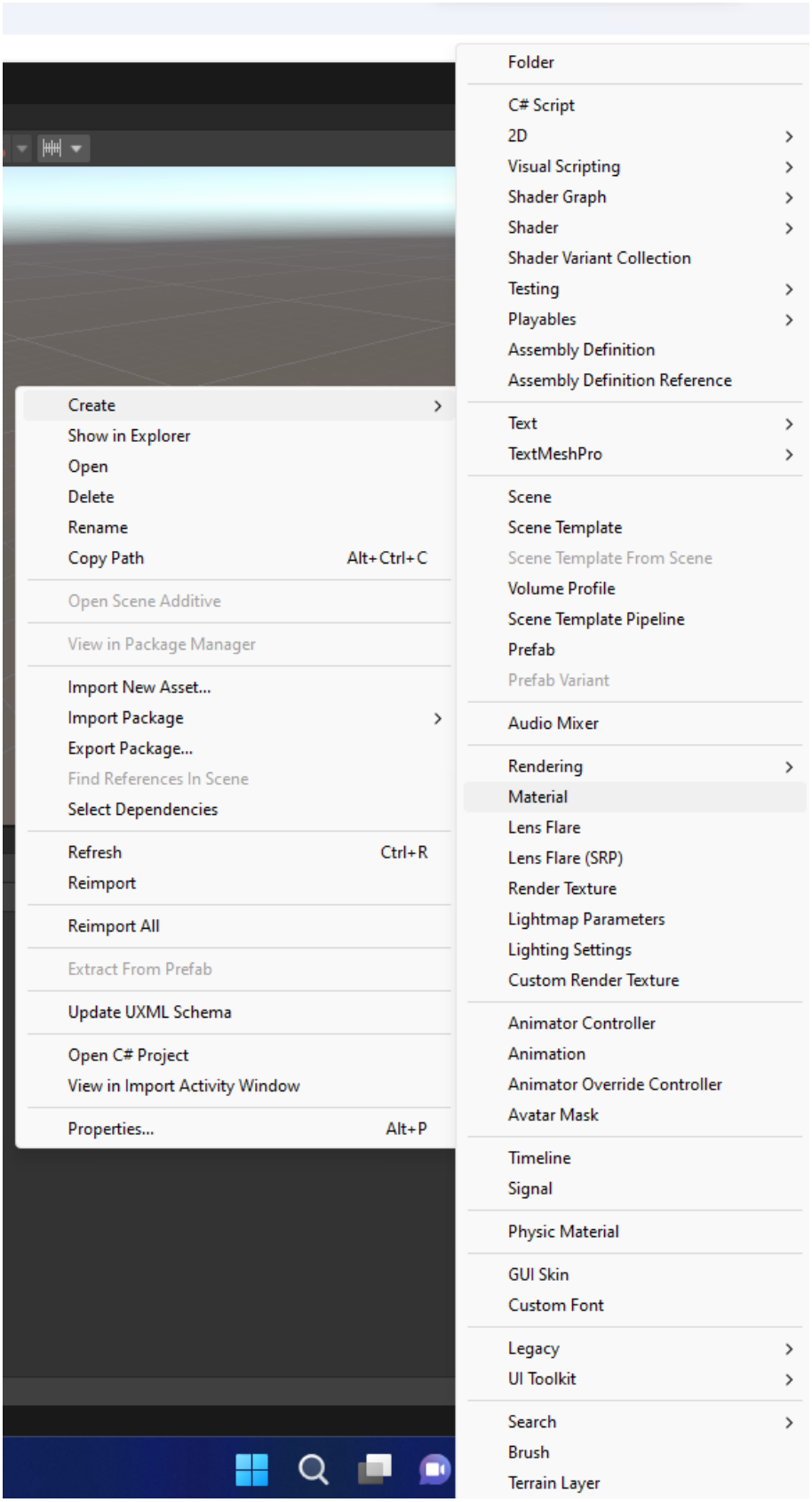
De regulă, când un obiect este desenat de către Unity complet roz, cauza este una din următoarele:

- Nu s-a specificat un material cu care să fie desenat obiectul.
- Materialul specificat este invalid în Render Pipeline-ul folosit.
- Shaderele folosite de materialul asignat obiectului conțin erori de compilare.

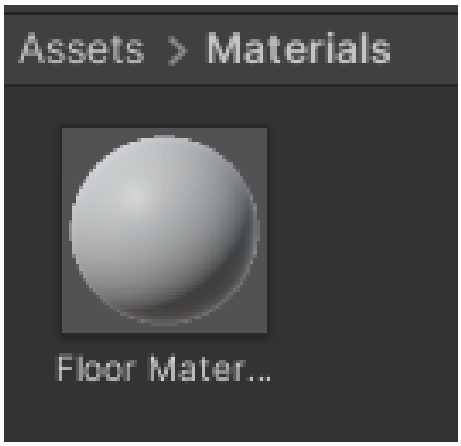
În cazul nostru, problema este că nu am atribuit un material cu care să fie desenat obiectul.

Pentru a face asta, trebuie să atribuim un material în field-ul *Element* 0 din lista *Materials* a componentei *Mesh Renderer*. De dată aceasta nu vom mai folosi cerculețul din dreptul field-ului pentru că acela ne va afișa o listă cu materiale care nu sunt compatibile cu *URP*, prin urmare, cubul va rămâne roz dacă încercăm să le atribuim.

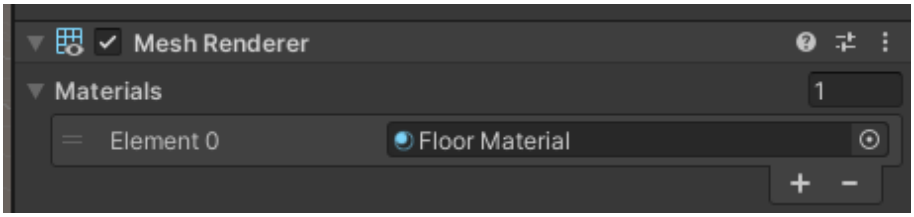
În interiorul directorului *Assets* vom crea un director nou numit *Materials*. În interiorul acestui folder vom crea un material nou: *click dreapta/Create/Material*.



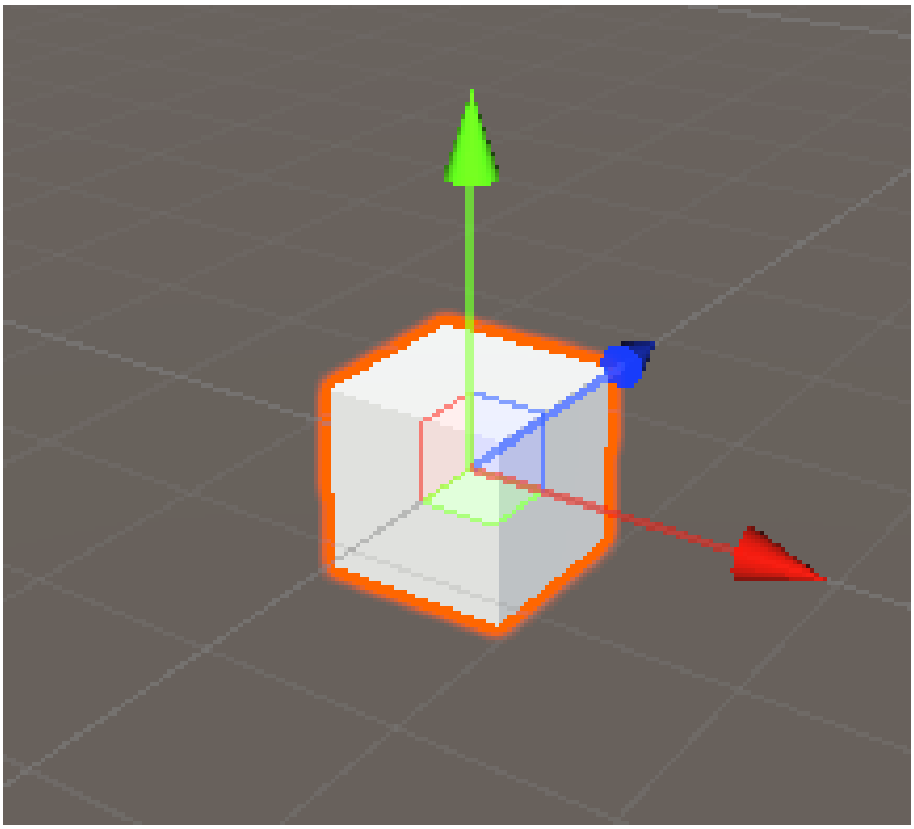
Vom numi acest material *Floor Material*.



Pentru a atribui acest material obiectului nostru, putem fie să dăm drag and drop materialului pe obiect, fie să îi dăm drag drop în field-ul specificat mai devreme.

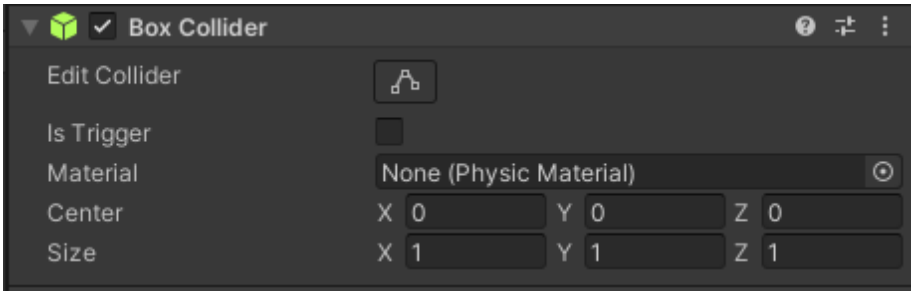


Acum cubul este afișat cum trebuie pe ecran.



4.2.3 Box Collider

Pentru ca un obiect să poată interacționa cu alte obiecte este necesar ca acesta să aibă o componentă de tipul *Collider*. Există mai multe tipuri *Collider*-e. Pentru obiectul nostru vom adăuga un obiect de tipul *Box Collider*.



5 Fizici de bază

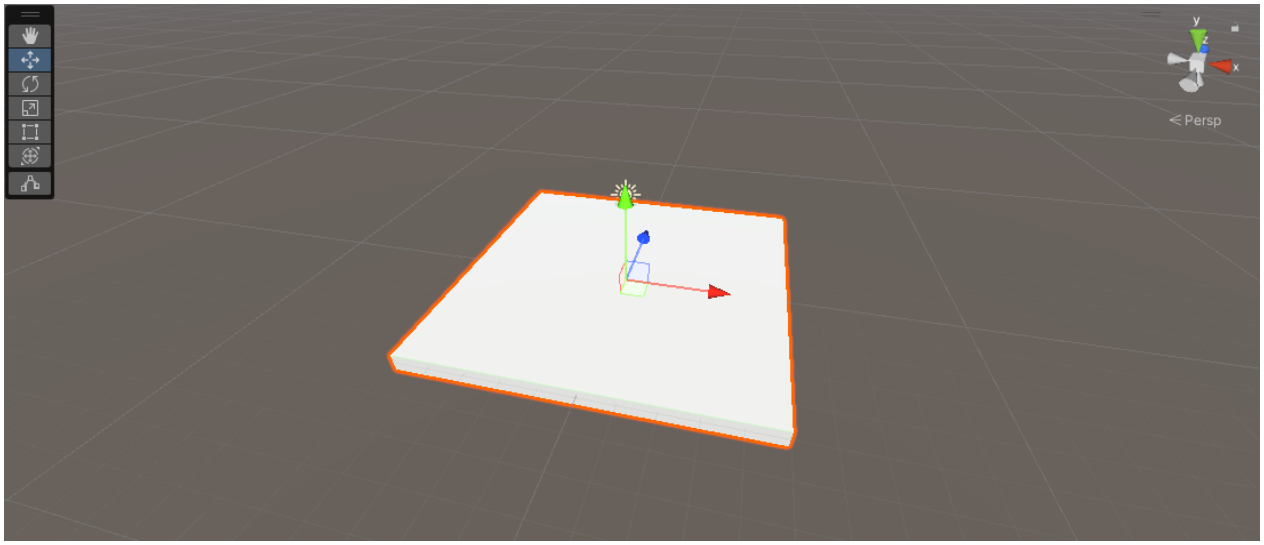
Vom simula căderea unei bile peste o podea. În primul rând trebuie să transformăm cubul creat anterior într-o podea.

5.1 Transformări

Acum că obiectul este vizibil, se poate observa efectul rotației și al scalării obiectului din componenta *Transform*. Pentru exemplul din acest laborator, vom atribui componentei Transform următoarele valori:

- Position: X 0 Y 0 Z 0
- Rotation: X 0 Y 0 Z 0
- Scale: X 10 Y 0.5 Z 10

Rezultatul obținut va fi o suprafață care seamănă cu o podea.



5.1.1 De unde am obținut valorile pentru *Scale*?

Valoarea componentei X determină cât de mare este obiectul de-alungul axei determinate de săgeata roșie din editor. Y reprezintă acelasiși lucru, doar că pentru axa determinată de săgeata verde. Analog, Z.

Din moment ce cubul nostru nu a fost rotit, axa Y reprezintă înălțimea. Cum noi vrem să obținem o podea, aceasta trebuie să aibă o înălțime mică. Vrem ca podeaua să fie lungă și lată, deci am atribuit valori mari componentelor X și Z.

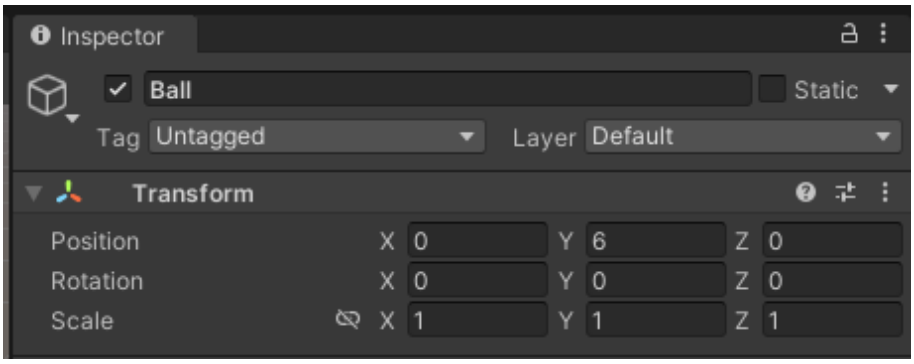
Vom continua să discutăm despre sistme de coordonate în laboratoarele viitoare.

5.2 Adăugarea unei bile

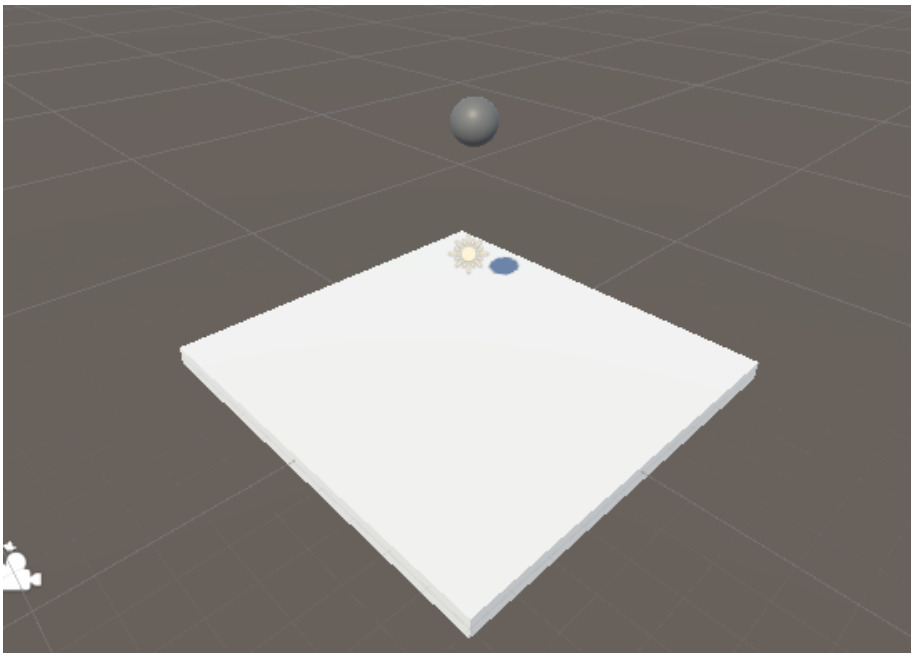
Vom adăuga o sferă în scenă. În loc să reluăm tot procesul anterior de a adăuga componente manual, vom adăuga o bilă direct din Unity. În loc de *Create Empty* vom selecta *3D Object/Sphere* din meniul care apare când dăm click dreapta în fereastra *Hierarchy* sau click pe tab-ul *GameObject*.

În scenă va apărea o bilă gri care conține toate componentele pe care le conține podeaua creată de noi anterior. Deși are aceleași componente, ea arată diferit, deoarece valorile proprietăților componentelor sunt diferite. Se observă că sfera dispune de un *Sphere Collider* în locul unui *Box Collider*

Vom muta această bilă deasupra podelei create anterior, astfel, poziția acesteia va fi (0, 6, 0) (De acum așa vor fi notate atât pozițiile cât și vectorii în cadrul acestor laboratoare, prima componentă reprezintă valoarea componentei X, a doua reprezintă valoarea componentei Y, iar ultima reprezintă valoarea componentei Z).



În fereastra *Scene* bila va fi desenată deasupra podelei.

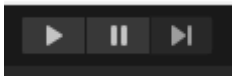


5.2.1 De ce bila este gri iar podeaua este albă?

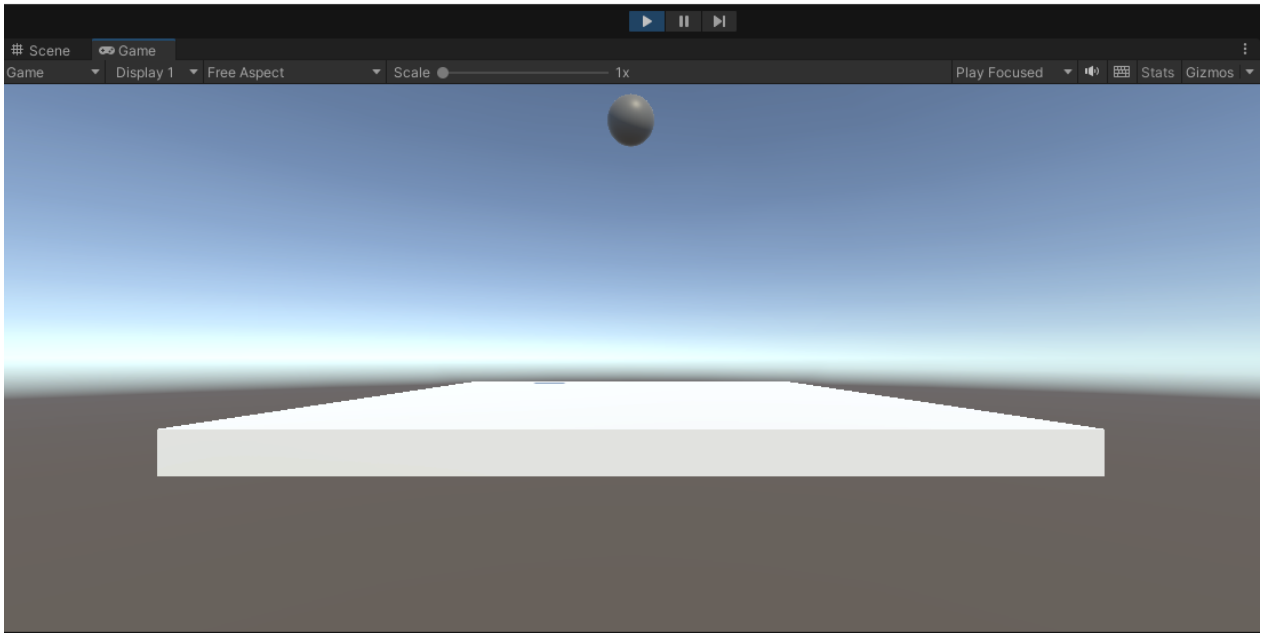
Podelei i-am atribuit un material nou. Materialele noi au valoarea implicită pentru culoare egală cu alb. Sferei nu i-am atribuit manual materialul, a fost atribuit de către *Unity* în momentul în care am creat-o. Materialul folosit este unul din pachetul *URP* care are culoarea gri.

5.3 Pornirea jocului

Pentru a porni jocul realizat trebuie să apăsăm pe butonul *Play* aflat în partea de sus a editorului.



După ce jocul este pornit, fereastra *Game* va fi activă. în interiorul ei rulează jocul. Momentan, în această fereastră se vede doar ce se vede din punctul de vedere al camerei din scenă.



După cum se poate observa, ambele obiecte create anterior sunt statice. Comportamentul pe care ni-l dorim este ca bila să cadă peste podea.

5.4 Rigidbody

Vom adăuga o componentă de tip *Rigidbody* asupra sferei. Această componentă comunică cu sistemul de fizici din *Unity* și include obiectul asupra căreia este aplicată în calculele pentru fizică. Se pot specifica proprietăți ale acestei componente, dar aceasta este o temă pentru laboratoarele ulterioare. După ce este adăugată această componentă, bila va cădea peste podea în momentul în care se intră în modul de *Play*.

6 Exerciții

6.1 Exercițiul 1

Instalați pachetul *Unity.Mathematics* în proiectul creat anterior.

6.2 Exercițiul 2

Poziționați camera asfel încât să se vadă cât mai bine căderea bilei peste podea.

6.3 Exercițiul 3

Schimbați culoarea materialului pentru podea și adăugați un material nou pentru bilă.

6.4 Exercițiul 4

Adăugați mai multe bile cu proprietăți diferite (poziție, dimensiune, material) care să cadă deasupra podelei.

6.5 Exercițiul 5

În timp ce rulează jocul, intrați în fereastra *Scene*, iar de acolo modificați pozițiile unor obiecte. Ce se observă în fereastra *Scene*? Dar în fereastra *Game*?

6.6 Exercițiul 6

Creați o scenă nouă în care și creați un robot din cuburi în acea scenă.