Laboratorul 3
Fizică

1 Fizică de bază

În Laboratorul 4 al materiei *Introducere in Programarea Jocurilor pe Calculator* s-a discutat despre fizicile din *Unity*. Pentru cine nu a parcurs acel laborator, recomand parcurgerea sa înainte de rezolvarea cerințelor acestui laborator.

2 Mișcare

Această serie de tutoriale prezintă foarte bine modul de lucrul cu fizică în *Unity* dar și metode pentru implementarea mișcării unui caracter într-un joc. Rezultatele obținute în aceste tutoriale sunt foarte finisate (polished).

3 Gravitație custom

În cerințele pentru laborator sunt implicate și forțe precum gravitație custom. Pentru implementarea acelor cerințe puteți urma acest tutorial sau acest tutorial.

4 Cerințe de laborator

4.1 Instanțierea obiectelor (0.025p bonus)

Creați un obiect care instanțiază constant diferite tipuri de obiecte în proximitatea acestuia. În scenă se va afișa și zona în care aceste obiecte sunt instanțiate folosind `Gizmos`.

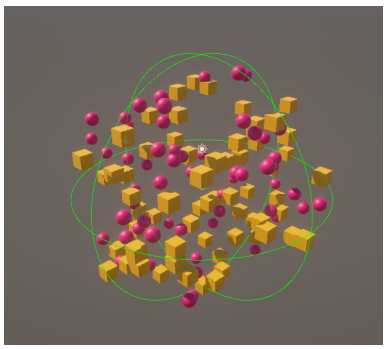

Figure 1: Spawner de cuburi și sfere.

4.2 Lifespan (0.025p bonus)

Pentru fiecare obiect instanțiat se va atribui o durată de viață. Pe parcursul duratei sale de viață, un obiect se va micșora constant, iar când dimensiunea acestuia va ajunge la 0, acesta va fi eliminat complet din scenă.

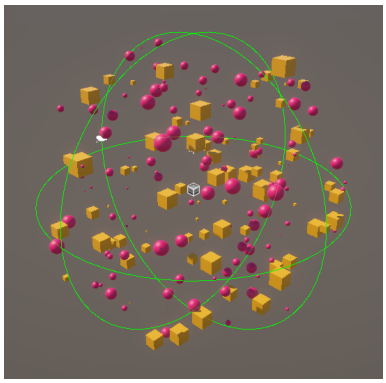

Figure 2: Obiecte cu dimensiuni diferite în funcție de durata lor de viață.

4.3 Comportament fizic (0.025p bonus)

Se vor adăuga componentele `Rigidbody` și `Trail Renderer` obiectelor pentru ca acestea să aibă comportament fizic, iar traiectorile lor să poată fi vizualizate. Se vor modifica proprietățile componentei `Trail Renderer` pentru ca aceasta să arate bine. În funcție de Lifetime se va actualiza și grosimea `Trail Renderer`-ului.

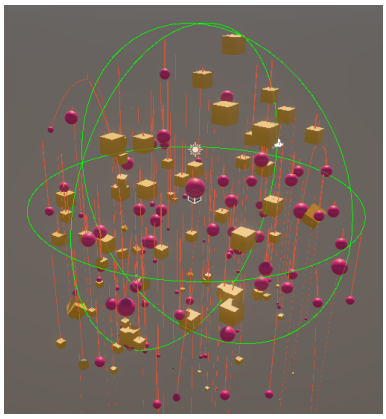

Figure 3: Obiecte cu comportament fizic și care lasă urme.

4.4 Impuls inițial (0.05p bonus)

Atunci când sunt instanțiate, obiectelor li se va atribui un impuls pe o direcție aleatoare și un moment al forței aleator. Se vor folosi metodele `rigidbody.AddForce`, respectiv `rigidbody.AddTorque`.

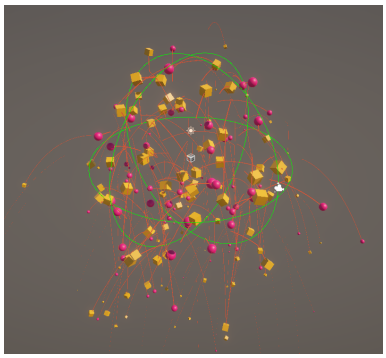

Figure 4: Obiecte instanțiate cu un impuls și un moment al forței.

4.5 Dezactivarea gravitației (0.025p bonus)

Dezactivați simularea gravitației pentru aceste obiecte.

4.6 Gravitație custom (0.1p bonus)

Adăugați o sfera în scenă care atrage obiectele instanțiate folosind o forță de gravitație custom. Această forță va fi implementată fie prin actualizarea constantă a proprietății `velocity` a obiectelor, fie prin aplicarea constantă a unor forțe de accelerație prin intermediul metodei `rigidbody.AddForce`.

Pentru această sferă se vor defini două raze de influență:

- Outer Radius - zonă în interiorul căreia forța gravitațională este la intensitate maximă.
- Outer Falloff Radius - zonă în interiorul căreia forța gravitațională este la intenstiate maximă.

Formula pentru determinarea forței aplicate asupra obiectului la o anumită poziție este următoarea:

$$acc(pos) = \begin{cases} \vec{0}, & dist > OuterFalloffRadius \\ g\frac{\overrightarrow{pos_{sphere} - pos}}{dist}(1 - \frac{dist - OuterRadius}{OuterFalloffRadius - OuterRadius}), & dist > OuterRadius \ \& \ dist \leq OuterFalloffRadius \\ g\frac{\overrightarrow{pos_{sphere} - pos}}{dist}, & dist \leq OuterRadius \end{cases}$$

Unde:

- *dist* - reprezintă distanța dintre poziția *pos* în care evaluăm acceleația și poziția sferei, adică pos_{sphere}.
- OuterFalloffRadius - Outer Falloff Radius.
- g - constantă gravitațională apromximativ egală cu 9.81. Putem folosi *Phisics.gravity.magnitude*.
- pos_{sphere} - poziția sferei.
- OuterRadius - Outer Radius.

Se vor afișa cele două raze de influență folosind `Gizmos`. De asemenea, accelearația gravitațională calculată anterior va fi aplicată constant asupra obiectelor în interiorul metodei *FixedUpdate*.

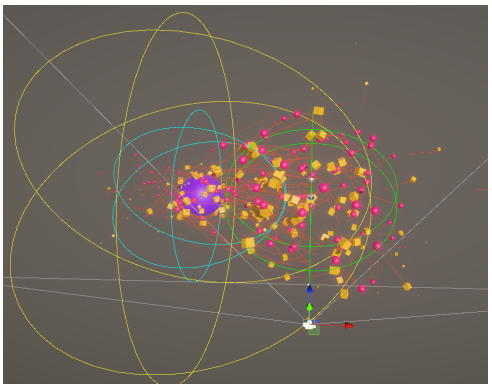

Figure 5: Obiecte atrase de sfera definită anterior.

4.7 Detectarea și răspunsul coliziunilor (0.05p bonus)

Detectați coliziunile dintre obiecte și sfera adăugată anterior folosind metoda `OnCollisionEnter(Collision collision)`. Atunci când un obiect atinge sfera, durata de viață a acestuia se va reseta, și va face un *bounce*. Pentru a implementa mecanica de *bounce* puteți reflecta velocity-ul acestuia folosind metoda $Vector3.Reflect$ din Unity, astfel:

```
rigidbody.velocity = Vector3.Reflect(rigidbody.velocity, (transform.position -
    collision.transform.position).normalized);
```

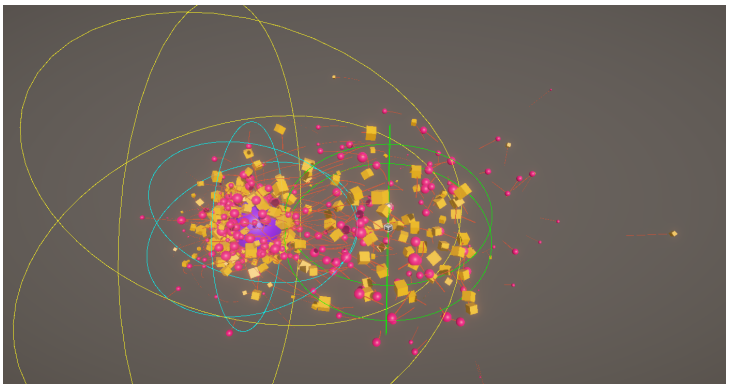

Figure 6: Acum că obiectele își extind durata de viață, sunt mai multe obiecte care înconjoară sfera.

4.8 Mișcarea sferei (0.05p bonus)

Deplasați sfera pe o traiectorie elipitcă astfel încât să se poată observa influența acesteia asupra obiectelor atunci când aceasta se află în mișcare. Pentru această mișcare se va folosi direct componenta $Transform$ fără a defini comportament fizic sferei.

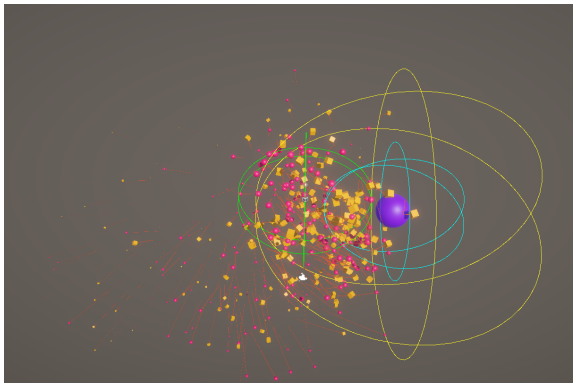

Figure 7: Sfera se deplasează iar obiectele încearcă să o urmărească.

Mai funcționează mecanica de *bounce*?

4.9 Mișcarea sferei v2 (0.05p bonus)

Modificați codul anterior astfel încât sfera să dispună de o componentă `Rigidbody` cu gravitația dezactivată și setată pe modul `Kinematic`. Deplasați sfera folosind $rigidbody.MovePosition$ în interiorul metodei $FixedUpdate$ în locul metodei definite pentru exercițiul anterior. Ce observați?

Aplicația finală