Colegiul Național „Andrei Șaguna” Brașov

*LUCRARE PENTRU OBȚINEREA CERTIFICATULUI DE ATESTARE A COMPETENȚELOR PROFESIONALE*

## Tema: Joc în Unity – Flappy Bird

BABOIU RĂZVAN

Clasa a XII-a A

Profesori coordonatori:

Gabriela Tutunea și Valentina Gal

Cuprins

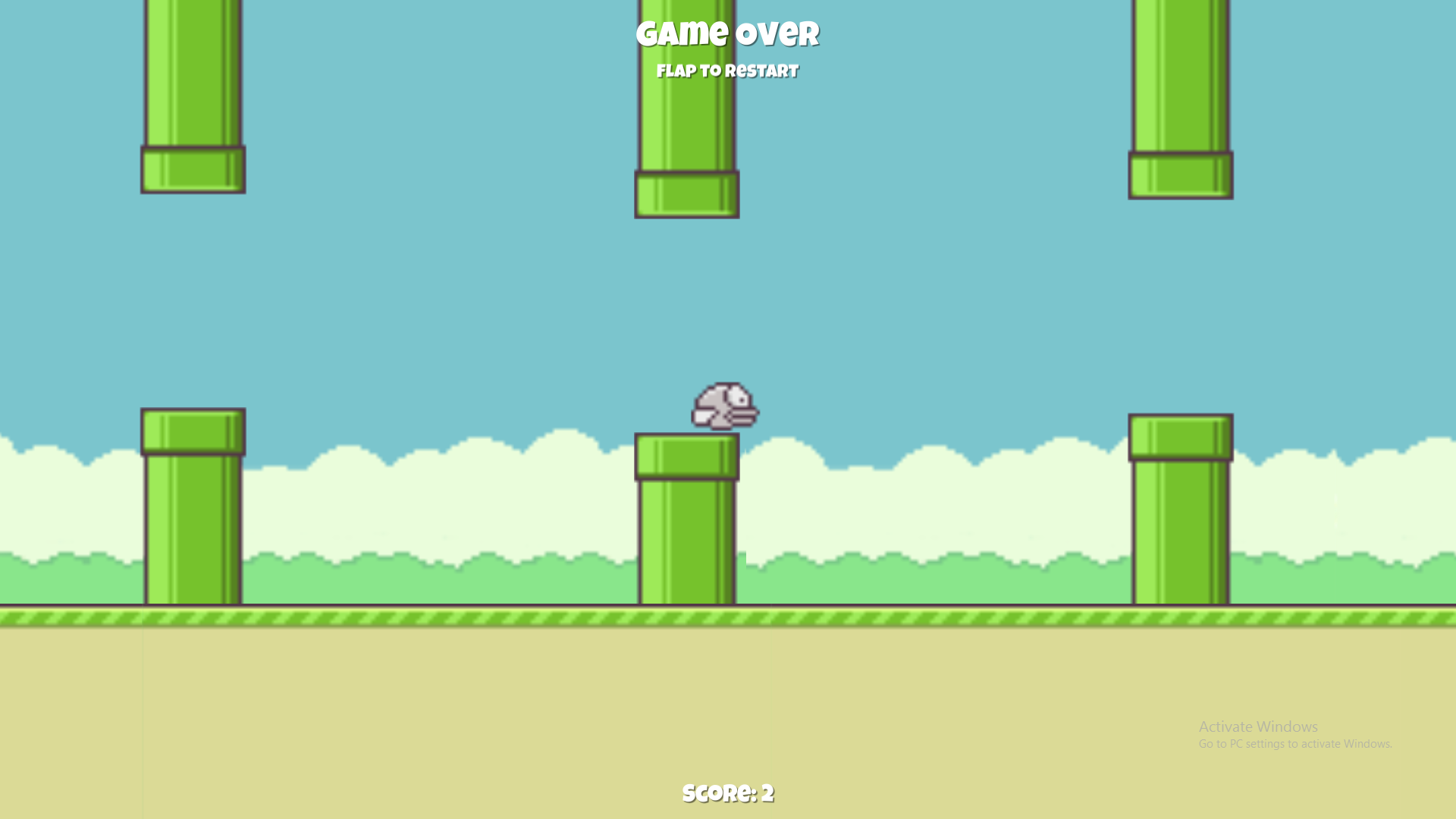
Motivarea temei alese și prezentarea pe scurt a jocului ………………………………… 1.

Prezentarea game engine-ului și a limbajelor de programare folosite …………………...1.

Detalii de implementare ………………………………………………………………….2.

Bibliografie ………………………………………………………………………………8.

1. **Motivarea temei alese și prezentarea pe scurt a jocului**

„Flappy Bird” este unul dintre cele mai cunoscute jocuri din întreaga lume, în special în zona jocurilor mobile. Acesta a apărut în anul 2013 dar a crescut în popularitate în jurul anului 2014 datorită nivelului său de dificultate crescut, a nivelului de dependență pe care îl oferă și a simplității sale. „Flappy Bird” e un side-scroller în care jucătorul controlează o pasăre, în încercarea de a conduce pe aceasta printre o mulțime de tuburi și țevi verzi, fără să le atingă pe acestea. 

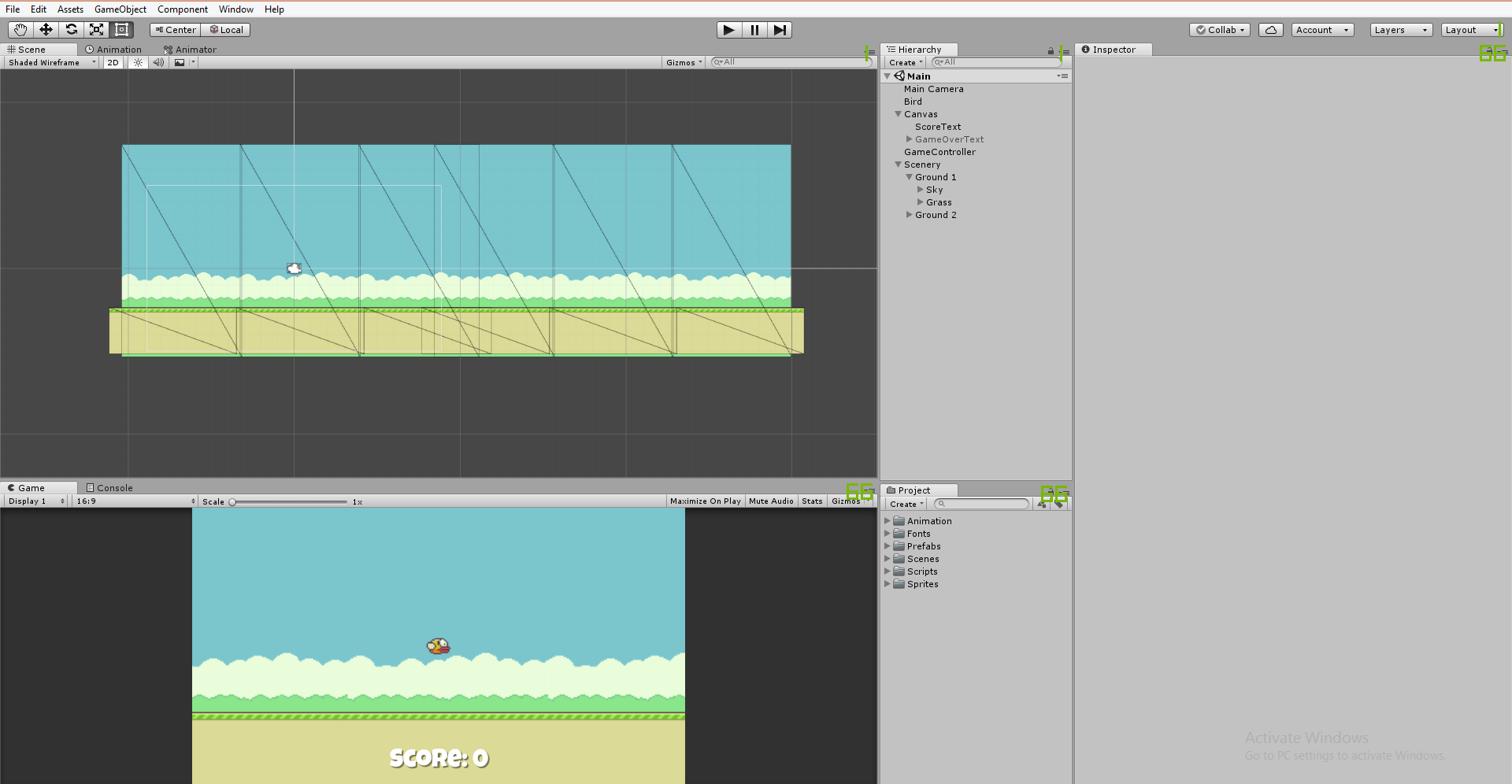
Jocul a devenit popular odată cu intrarea mea la liceu și, ca multor altora, mi-a consumat foarte mult din timpul meu liber (dar și din timpul orelor de școală), încercând să obțin un scor mai bun decât cel al prietenilor mei. De aceea mi-am dorit să creez o copie acestui joc care mi-a marcat mai mult sau mai puțin liceul.

1. **Prezentarea game engine-ului și a limbajelor de programare folosite**

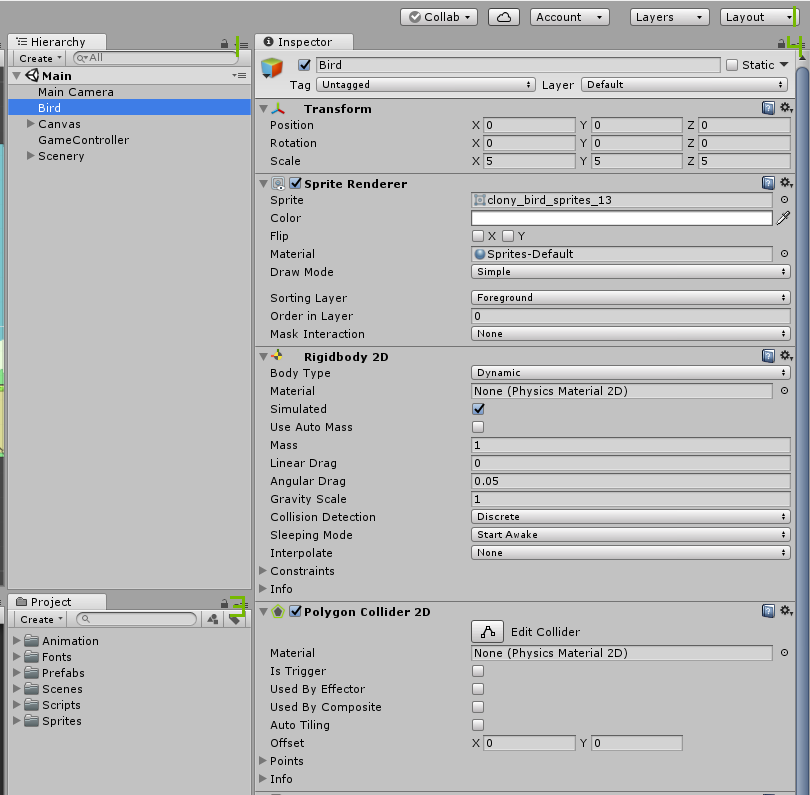
Unity este un game engine pentru multiple platforme (Windows, Mac, Android, IOS, etc.) creat de Unity Technologies, care e folosit de obicei pentru crearea de jocuri video 2D și 3D și simulări. Engine-ul suporta grafica 2D și 3D, funcționalitate de tipul drag-and-drop și realizare de scripturi în C#, Boo sau JavaScript. Pentru aplicațiile destinate rulării pe Windows, Unity se folosește de Direct3D, care este un API (application programming interface) grafic. Jocul meu este 2D, iar limbajul de programare folosit e C#.

Am ales să fac scripting-ul în C#. Acesta este un [limbaj de programare](https://ro.wikipedia.org/wiki/Limbaj_de_programare) orientat pe obiecte conceput de [Microsoft](https://ro.wikipedia.org/wiki/Microsoft) la sfârșitul anilor 90. A fost conceput ca un concurent pentru [limbajul Java](https://ro.wikipedia.org/wiki/Limbajul_Java). Ca și acesta, C# este un derivat al limbajului de programare [C++](https://ro.wikipedia.org/wiki/C%2B%2B).

1. **Detalii de implementare**

Unity folosește, pentru construcția jocurilor, elemente numite scene. Aplicația mea are o singură astfel de scenă, denumită de mine Main Scene. Aceasta are la rândul ei mai multe componente: camera principală prin care este redat jocul (Main Camera), modelul jucătorului pe care acesta îl controlează (Bird), textul care afișează scorul (ScoreText) și textul care anunță finalul jocului (GameOverText), un obiect de tipul GameObject denumit GameController cu prin intermediul căruia controlăm, setăm și schimbăm starea jocului, două fundaluri și două porțiuni de pământ suprapuse pentru a crea un fundal de două ori mai mare decât ceea ce poate surprinde camera (Scenery). 

Câmpul de joc este format din două background-uri și doua forme de soluri lipite unele de altele și un anumit număr de obstacole care poate să fie setat din interiorul game engine-ului. Pentru a da senzația de mișcare a jucătorului, background-ul și obiectele se mișcă cu o viteză constantă negativă, în timp ce modelul jucătorului sta pe loc și i se aplică o forță constantă orientată în sus, atunci când jucătorul apasă butonul din stânga al mouse-ului. Pasărea, solul și obstacolele au asociate cate un GameObject de tip RigidBody2D (obiect rigid în 2 dimensiuni), pentru a putea avea loc ciocniri între modelul jucătorului și restul corpurilor, dar și un GameObject de tip BoxCollider2D, pentru a mă putea folosi de momentul de atingere între aceste corpuri.

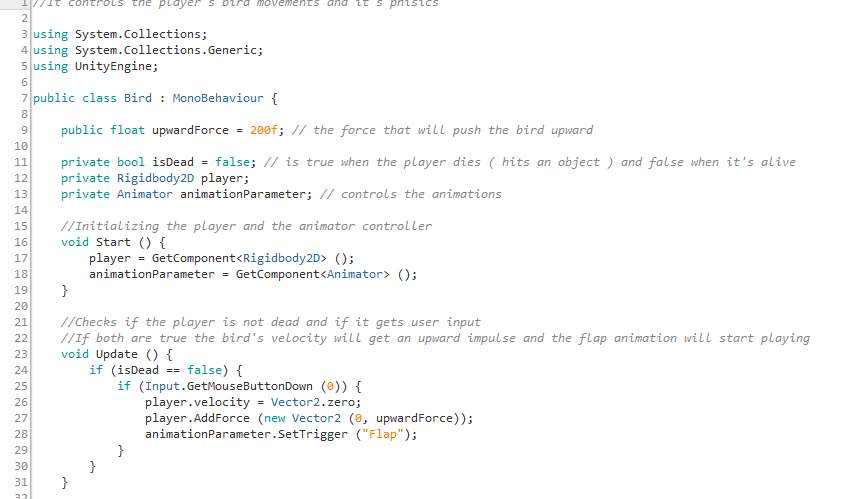


Grafica jocului este bazată pe o singura imagine de tip Sprite, în care se găsesc toate elementele grafice de care are nevoie jocul, acestea fiind selectate din imagine automat. Astfel, jocul este optimizat din punct de vedere al vitezei nude render-uire a graficii, primind o singură imagine pe frame, nu câte o imagine pentru fiecare element grafic.

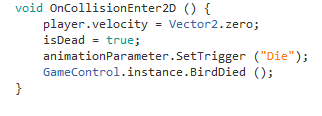
Jocul conține 6 script-uri care controlează tot jocul: pentru fizica și mișcările păsării, pentru mișcarea backgroundului și a obstacolelor, pentru adăugarea punctelor în momentul în care jucătorul trece de niște obstacole și scriptul care controlează tot jocul.

*Scriptul Bird*

Controlează mișcările pasării (a modelului jucătorului) și fizica din spatele ei. In funcția Start se inițializează componentele jucătorului: player (de tip RigidBody2D) și animationParameter (de tip Animator). In funcția Update se verifică o data pe fiecare frame dacă a fost primit un Input de la butonul din stânga al mouse-lui și dacă se primește i se aplică o forță verticală upwardForce (variabilă de tip public – permite schimbarea din interiorul game engine-ului) și se activează animația de zbor.

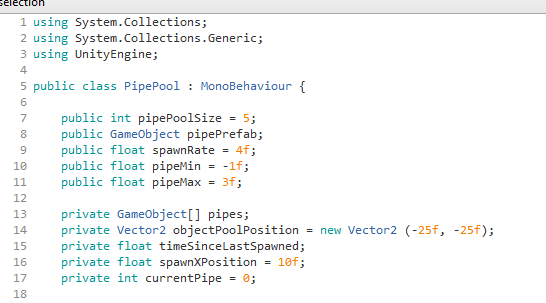


In plus mai am funcția standard din Unity OnCollisionEnter2D în care se intră atunci când corpul rigid al păsării intră în contact cu alt corp rigid (pământul sau obstacolul). Aici se resetează mișcarea păsării la 0, starea păsării se schimbă la moartă atât în script cât și în instanța jocului și animația de moarte se activează.

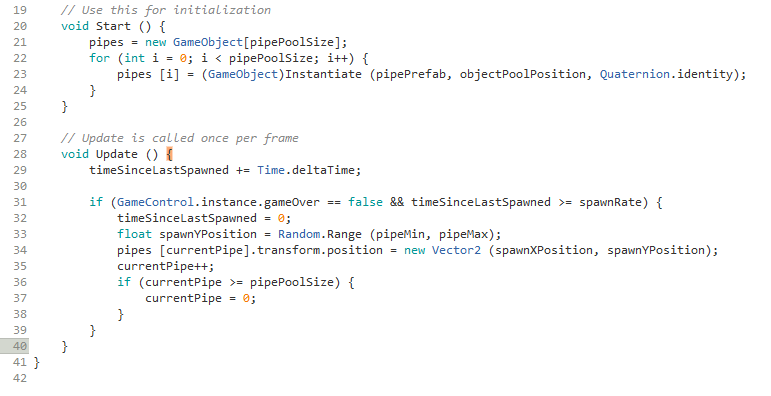


*Scriptul PipePool*

Scriptul conține variabile de tip public: pipePoolSize (numărul de obstacole din pool), pipePrefab (un GameObject pentru a lua atribuțiile obstacolelor), spawnRate (viteza cu care se resetează obstacolele), pipeMin și pipeMax (intervalul pe axa Y în care o să fie puse obstacolele). Scriptul conține si variabile private: pipes (un vector de GameObejct-uri), objectPoolPosition (poziția inițială la care vor fi instanțiate obstacolelor pentru a nu fi în interiorul camerei jocului), timeSinceLastSpawned (timpul la care ultimul obstacol a fost repoziționat), spawnedXPosition (poziția pe axa X la care se repoziționează obstacolele) și currentPipe (numărul curent al obstacolului în vectorul pipes).



In funcția Start obstacolele sunt instanțiate și poziționate în afara camerei. In funcția Update se adaugă, la fiecare frame, timpul trecut de la frame-ul trecut în timeSinceLastSpwned și dacă se depășește timpul setat pentru resetarea obstacolelor (spawnRate): timpul se resetează, spawnYPosition se setează ca fiind un număr real ales aleatoriu din intervalul (pipeMin, pipeMax), se schimbă poziția obstacolului și se trece la următorul în vector.



Această metodă de a avea control asupra obstacolelor – numită object pooling – este una optimă deoarece, în loc de a aloca și distruge obiecte încontinuu, metoda creează un set de obiecte ce se refolosesc.

1. **Bibliografie**

<https://unity3d.com/learn/tutorials/s/2d-game-creation>

<https://unity3d.com/learn/tutorials/topics/2d-game-creation/rigidbody-2d?playlist=17093>

<https://unity3d.com/learn/tutorials/topics/2d-game-creation/collider-2d?playlist=17093>

<http://quill18.com/unity_tutorials/>

<https://docs.unity3d.com/Manual/AnimationSection.html>