Game Engines

Andrej Lúčny

Katedra aplikovanej informatiky

lucny@fmph.uniba.sk

http://www.agentspace.org/ge

Strielačky

Našim cieľom bude strieľať na rôzne objekty

Ako zistíme, všetko potrebné k tomu vieme okrem uskutočnenia procesu, ktorý sa deje po zásahu

Na to potrebujeme: kinematické explózie a efekty ako particle system.

Vyjdeme z projektu Ethan – ver 3.

v assetoch už je množstvo vecí, pribudli: simple particle pack, grenade sound, skript ExplodeMesh.cs a ďalšie

Kinematické explózie

Umiestnime do scény jednu kocku a dajme jej aj Rigidbody a nechajme BoxCollider, aby to bol reálny fyzikálny objekt (scale napr. 0.5, pri umiestnení pracujte s transform v Inspectore, aby bola na zemi)

Fyzika implementovaná v Unity umožňuje priamo týmto objektom zadefinovať explozívnu silu.

```
GetComponent<Rigidbody>().AddExplosionForce(
  force, transform.position, radius, 0.5f, ForceMode.Impulse);
```

Vhodná sila je 2.0f a polomer 5.0f

Napíšte skript Exploder, ktorý na klávesu nadhodí kocku

Aby neuletela pri podržaní klávesy, môžeme využiť

```
Input.GetKeyDown(KeyCode.Space)
```

Kinematické explózie

Pridajme do scény viacero kociek. Pomocou Colliderov v okolí objektu vieme výbochom jednej kocky odhodiť aj ostatné, napr. na kliknutie myšou

```
void OnMouseDown()
{
    Collider[] colliders = Physics.OverlapSphere(transform.position, radius);
    foreach (Collider c in colliders)
    {
        if (c.GetComponent<Rigidbody>() == null) continue;
        c.GetComponent<Rigidbody>().AddExplosionForce(force,
            transform.position, radius, 0.5f, ForceMode.Impulse);
    }
    //Destroy(gameObject);
}
```

Zvukový efekt

Prehráme ho v skripte za vhodných podmienok cez:

```
GetComponent<AudioSource>().Play();
```

AudioClip-u vypíname Play on awake

V projekte už máme 8-bit composer superpack v ktorom nájdeme pod 8-bit sound effect / Explosions vhodné zvuky

Efekty

Teraz by to chcelo aj nejaký oheň, dym a podobne.

Na to potrebujeme efekty. Efekty nie sú fyzikálnymi objektmi v scéne, sú do nej len doplnené pri rendering-u

Na implementáciu ohňa, dymu, letiacich črepín a podobne, sa používa efekt Particle System

Vložme ho do scény
GameObject / Effects / Particle System

Vynulujme jeho Transform a v hierarchy window ho podraďme pod výbušnú kocku

Particle System

Aj bez behu scény môžeme efekt prehrať pomocou simulátora, ktorý sa dá aj zastaviť a znovu spustiť.

Pomocou mnohých parametrov je možné particle System formovať:

- **Looping (zapnite len kým efekt formujete)**
- **Duration (dajte max 1 sekundu)**
- **Play on Awake (vypnite)**
- **Emission (dajte 50)**
- **Shape (vyformujte cone)**
- **SizeOverLifetime (nech klesne dosť rýchlo na nulu)**
- **ColorOverLifetime (kombinujte červenú a oranžovú)**

Pokúste sa vyformovať niečo podobné na oheň.

Particle System

Vypnite looping

Premenujeme Particle system v hierarchy window napr na Bum

Z hierarchy window ho stiahnite do Project window, čím z neho urobíte prefab. Odstráňte ho zo scény.

Vložte ho do vybuchovacieho skriptu ako parameter

```
public ParticleSystem explosion;
```

a vyvolajte ho:

```
Explosion.Play()
```

Particle System

Miesto svojho nedokonalého ohňa, použite hotovú explóziu z asset store

V projekte je už stiahnutý *simple particle pack*
Použite napr. Destruction01

(nezabudnite mu vypnúť Play on Awake)

Potom odstráňte kocku s vaším skriptom zo scény (môžete si ju uchovať ako prefab, keď ju pred ostránením stiahnete z hierarchy window do project window)

Rozpad pri explózii

Použite na kockách hotový skript ExplodeMesh na explóziu, ktorá objekt deštruuje na trojuholníky, z ktorých sa skladá, na ich likvidáciu využívame Destroy(object,delayTime)

```
Mesh M = GetComponent<MeshFilter>().mesh; …
Vector3[] verts = M.vertices; // vertices
Vector3[] normals = M.normals; // normal vectors
Vector2[] uvs = M.uv; // texture mapping
for (int submesh = 0; submesh < M.subMeshCount; submesh++)
{
   int[] indices = M.GetTriangles(submesh);
   for (int i = 0; i < indices.Length; i += 3)
   {
       Vector3[] newVerts = new Vector3[3];
       Vector3[] newNormals = new Vector3[3];
       Vector2[] newUvs = new Vector2[3];
       … new Mesh … new GameObject …  AddExplosionForce …
```

Zbraň

Vložte do scény guľomet. V assetoch už máte balík *Customizable Gun FreeEd*. Podraďte ho pod kameru a nastavte tak aby ho bolo vidno ale aby príliš nezakrýval scénu.

(zrušte mu pôvodný skript.)

Zariaďte rotáciu zbrane okolo osi y pomocou šípok. Prípadne aj okolo osi x

```
transform.rotation = Quaternion.Euler(
      transform.eulerAngles.x, y, transform.eulerAngles.z
);
```

Projektily

Strely sa v Unity3D realizujú ako skutočné malé fyzikálne objekty, ideálny tvar má Capsulle alebo Sphere, na ktorú nanášame vhodnú textúru. Nesmú byť akurát prirýchle, inak hrozí, že prekážkou preletia a nič nespôsobia, lebo kolízia sa nezdetekuje

Streľba potom spočíva vo vysielaní týchto objektov z hlavne zbrane, vždy po uplynutí timeoutu (pri stlačení spúšťe)

Musíme sa starať aj o to, aby strely nielen vznikali v hlavni, ale aj niekde alebo niekedy zanikali.

Empty object

Keď používame Capsulle ako projektil, narazíme na problém, že by ju potrebovali otočiť do vodorovnej polohy a až tak použiť. Šlo by to aj bez toho, ale potom by sme ju museli otáčať pri vytváraní inštancie a tiež jej vlastnom pohybe.

Na to sa dá do scény vložiť Empty object, pod ktorý sa Capsulle zavesí, či bude childom empty objektu. Jej nastavíme rotáciu Y a Z na 90 stupňov a vďaka tomu empty objekt nazvaný Projectile bude správne smerovaný.

Z vyrobeného projektilu spravíme Prefab *Projectile* a odstránime ho scény.

Projektily

Aby strely vyletovali z hlavne, zariadi sa ľahko: projektil pri inštancovaní preberie pozíciu a rotáciu hlavne zbrane (zbraň je tak orientovaná, aby to sedelo). A vypne sa mu gravity, aby letel rovno. Dá sa pohrať s jeho textúrou, napr. mobile shader - particles – additive.

Pridajte projektilu vlastný skript `Fly` s parametrom `cadence` udávajúcim rýchlosť projektilu vypáleného z hlavne.
Tento skript je veľmi jednoduchý a obsahuje jediný príkaz:

```
GetComponent<Rigidbody>().velocity =
    cadence * transform.forward;
```

pričom vhodná kadencia je cca 20.0f

Streľba

pri stlačenej spúšti (napríklad klávesa medzera alebo enter) potom strieľame volaním `Instantiate:`

```
Instantiate(projectile,transform.position,transform.rotation);
```

Pričom `projectile` je prefab, tj. `GameObject` odovzdávaný do skriptu ako parameter.

Časový limit určujúci frekvenciu streľby, počítame akumuláciou `Time.deltaTime` alebo rozdielmi `Time.time`

Zásah

Zásah zdetekujeme pomocou Collidera typu trigger, zrušíme strelu a vyvoláme explóziu (nahradenie objektu jeho časťami, ich kinematickou explóziou a particle efektom). Na to musíme prefab-u Projectile dať CapsuleCollider a zapnúť ho ako triggerovací. Udalosť potom chytáme cez OnTrigger() a pointer na zasiahnutý objekt získame:

```
public GameObject explosion_prefab;
void OnTriggerEnter (Collider other)
{
  GameObject explosion = Instantiate(
        explosion_prefab,transform.position,transform.rotation
  );
  Destroy(explosion,3); // destroy after 3s
  Destroy(other.gameObject); // destroy immediatelly
  Destroy(gameObject);
}
```

Pozor na podlahu, tú nechceme odstreliť!

Úloha

Dorobte ovládanie zbrane a strieľajte z nej na cieľ

V prípade zásahu, cieľ rozbite a urobte nový.