Game Engines

Andrej Lúčny

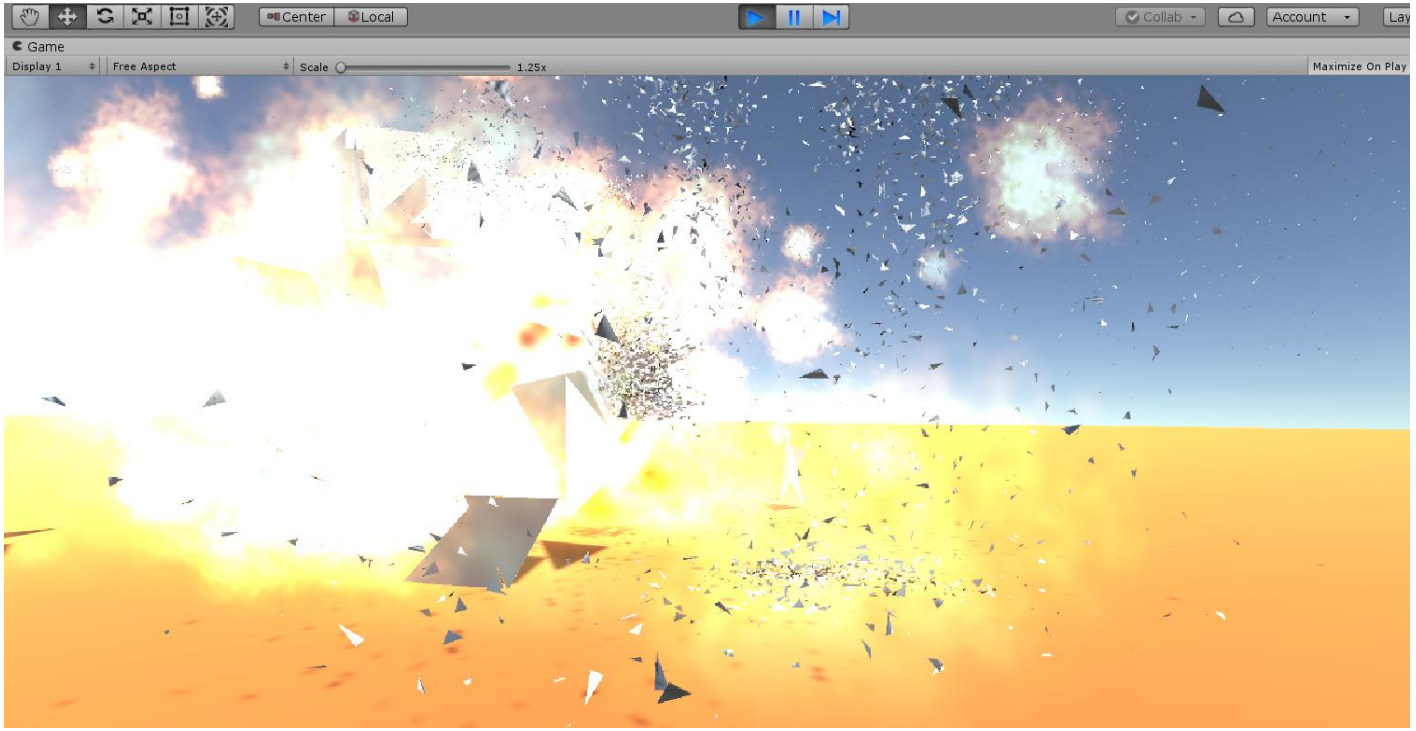
Katedra aplikovanej informatiky

lucny@fmph.uniba.sk

http://www.agentspace.org/ge

Explózia avatara

Keby sme ExplodeMesh použili na EthanBody (nie na Ethan), teoreticky by fungoval. Lenže pri počte trojuholníkov z ktorých sa Ethan skladá, by sme potrebovali enormný výkon počítača. (Pozor: Neskúšajte to !)



Ako s tým vybabrať: rozpad predpripravíme a bude zakaždým rovnaký

Explózia avatara

Zásah zložitejšieho objektu ako je avatar sa musí riešiť iným spôsobom ako zásah meshu zo zopár trojuholníkov.

- **Avatar sa môže rozletieť, ale na jednotlivé údy. Tu opäť urobíme kópiu údov a pôvodný objekt zrušíme. S údmi potom podnikneme kinematickú explóziu.**

- **Avatar môže spadnúť, pričom spravila padá do niekoľko málo polôh, trebárs len do jednej. Na to využívame animáciu**

Vložíme do projektu avatara Ethan, nastavíme správne jeho veľkosť.

Pohyb avatara

Upravíme skript avatara

Nech nejako pobehuje.

Postupnosť krokov v správaní by sme vedeli zariadiť aj stavovou premennou a meraním času, profesionálnejšie je však použiť Coroutine.

Enumerátor

```
for (myItem : myList) { … }
```

```
for (myItem : myEnumerator) { … }
```

Pri enumerátore prvky cez ktoré sa prechádza naraz neexistujú, ale sú postupne počítané procedúrou, ktorá ich postupne vracia cez `yield return`

Renderer využíva túto možnosť pri coroutinách. Podľa toho čo sa mu enumerátora vráti, ho zavolá o nejaký čas. Typickou návratovou hodnotou je `WaitForSeconds()` **ktorý nestojí, ale povie rendereru, aby pýtal ďalšiu hodnotu o daný čas.**

Coroutine

```
IEnumerator LookAround()
{
    anim.SetFloat("Turn", 1);
    yield return new WaitForSeconds(3);
    anim.SetFloat("Turn", -1);
    yield return new WaitForSeconds(3);
    anim.SetFloat("Turn", 1);
    yield return new WaitForSeconds(3);
    anim.SetFloat("Turn", -1);
    yield return new WaitForSeconds(3);
    anim.SetFloat("Turn", 0);
}

StartCoroutine("LookAround");//StopCoroutine("LookAround");
```

Explózia avatara

Avatar sa skladá z celej sústavy objektov. Pri Ethan-ovi jeho videľnú zložku robí najmä EthanBody a trochu aj EthanGlasses, čo sú nepravidelné útvary, zložené z trojuholníkov (mesh) reprezentujúcich povrch objektu. Konštrukčnú zložku predstavuje EthanSkeleton a jeho podradený objekt EthanHips. Výsledný tvar vzniká navlečením mesh-u EthanBody na kostru EthanHips.

Na rozsekanie avatara teda stačí vhodným spôsobom okopírovať vhodné časti meshu v EthanBody.

Hierarchia objektov

Keďže collider a rigidBody má Ethan, hodí sa nám priradiť explodovací skript v hierarchii najvyššiemu Ethanu. Kopírovať ale budeme mesh component EthanBody, ktorý je Ethanu podradený. Z Ethanu sa teda nejako musíme dostať k EthanBody.

V Unity3D je hierarchia budovaná pomocou komponentu Transform, ktorý má nielen position, rotation a localScale ale aj children, čo sú Transform-y podradených objektov. Medzi nimi možno vyhľadávať podľa mena i tagu cez Find(). Transform taktiež vie, aký objekt umiestňuje, na to má atribút gameObject. Takže v skripte Ethanu:

```
GameObject body = transform.Find("Ethan Body").gameObject;
```

Mesh

```
Mesh mesh = body.GetComponent<SkinnedMeshRenderer>().sharedMesh;
```

Mesh je definovaný pomocou:

- **Súradníc vrcholov** `mesh.vertices`
- **Normálovými vektormi v týchto vrcholoch** `mesh.normals`
- **Súradnicami textúry v týchto vrcholoch** `mesh.uv`
- **Trojuholníkmi zadanými trojicami indexov súradníc vrcholov** `mesh.triangles`

Na okopírovanie časti meshu nám teda stačí redukovať indexy trojuholníkov na danú časť tela avatara

Explózia avatara

Takýto výbuch by bol ale pomalý. Zrýchliť sa dá za cenu, že bude zakaždým rovnaký. To spravíme tak, že si vybrané indexy zaznamenáme a zadefinujeme ich priamo v .cs

```
using System.IO;

//Write some text to the text file
StreamWriter writer = new StreamWriter("d:\\"+i+".txt", true);
for (int j = 0; j < mesh.triangles.Length; j++)
   writer.WriteLine(mesh.triangles[j] + ", ");
writer.Close();

int[] arr0 = new int[] { 0,1,2, 0,2,3, 4,3,2, ... };
```

Vložíme do projektu skript ExplodeAvatar.cs (je hotový v zipe).

Laserové lúče

Už vieme, že strelu obvykle implementujeme ako letiaci objekt nepodliehajúci gravitácii.

Iná možnosť je použiť laserovú zbraň. Unity podporuje túto možnosť. Je možné vyslať lúč určitým smerom a merať, kde sa dotkne najbližšej prekážky.

Najčastejšie využívame túto možnosť pre interpretáciu pohybu myši po obrazovke. Vysielame z kamery lúč spájajúci jej ohnisko a bod na obraze zodpovedajúci polohe myši. Zisťujeme čo ten lúč do určitej vzdialenosti trafí. Avatar preto exploduje nie na kliknutie myšou, ale v momente, keď sa myš dostane medzi neho a kameru

Laserové lúče

Laserový lúč je implementovaný nasledovne:

```
Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
RaycastHit hit;
if (Physics.Raycast(ray, out hit, 100))
{
     if (hit.collider.gameObject == gameObject)
     {
          Explode();
     }
}
```

Laserové lúče môžeme prepojiť s reálnou laserovou pišťolou

https://youtu.be/2HByBPgC6D8

Detekcia zrážky

```
void Fix()
{
  Rigidbody mBody = GetComponent<Rigidbody>();
  // Get the velocity
  Vector3 horizontalMove = mBody.velocity;
  // Don't use the vertical velocity
  horizontalMove.y = 0;
  // Calculate the approximate distance that will be traversed
  float distance = horizontalMove.magnitude * Time.fixedDeltaTime;
  // Normalize it since it should be used to indicate direction
  horizontalMove.Normalize();
  RaycastHit hit;
  // Check if the body's current velocity will result in a collision
  if (mBody.SweepTest(horizontalMove, out hit, distance))
  {
    // If so, stop the movement
    mBody.velocity = new Vector3(0, -3.5f, 0);
  }
}
```