Game Engines

Andrej Lúčny

Katedra aplikovanej informatiky

lucny@fmph.uniba.sk

http://www.agentspace.org/ge

3D

Dajme si vytvoriť projekt 3D s balíkom assetov „Roll and ball tutorial“ (predtým treba learn a download)

[ak Unity vyhlási pritom chybu, stačí ju clear-nuť]
[ak máte novšiu verziu Unity, kde tutotiál nie je, tak len sledujte prednášateľa, alebo si stiahnite cvičenie5 a z neho použite *Roll and Ball – ver2*]

Loadnime scénu (v project window je ako asset s ikonou Unity)
v Game window zvoľme Maximize on play
Play a zahrajme sa, aby sme hru spoznali

Transform tools. Prefabs

V Scene window vyskúšajme (v 3D) Transform tools:

ktorým zodpovedajú písmenká *Q W E R T*

Skúste si jeden žltý hranol (herný objekty Pick Up*) presunúť, jeden otočiť, jeden zväčšiť, sledujte pritom v Inspectore atribúty Transform

Vyskúšajte zmeniť jeden a aplikovať zmenu na všetky (Prefab - Apply), Prefab je nový typ assetu, ktorý vytvárate ako kompozíciu z existujúcich s niekoľko málo nastaviteľnými parametrami (create/prefab + drag&drop objekt z hierarchy window na prefab)

FPS

Editujte skript, ktorý rotuje so žltými hranolmi (skript Rotator v assete Pickup)

Rotácia je vyjadrená ako časť z *(rot$_x$, rot$_y$, rot$_z$)* čo sú otočenia okolo osí x, y, z v stupňoch.

Všimnite si faktor Time.deltaTime, ktorý udáva čas v sekundách od posledného zrenderovaného frame, napr pri 20 fps je to 0.05f. Pomocou tohto faktora sa zabezpečuje, aby rýchlosť otáčania bola rovnaká na každom stroji bez ohľadu na fps, ktoré sa na ňom dosiahne.

Využitie hierarchiu v Hierachy window a doplňte pod Canvas game objekt Text a dajte mu skript s atribútom `public Text winText; (import using UnityEngine.UI;)`
Zistite FPS ako prevrátenú hodnotu času, ktorý si zobrazíte do scény cez `winText.text = Time.deltaTime.ToString();`

Vytvorenie hry od začiatku

Urobite vlastný prázdny 3D projekt

a vytvorte od základu podobnú (alebo aj rovnakú) scénu

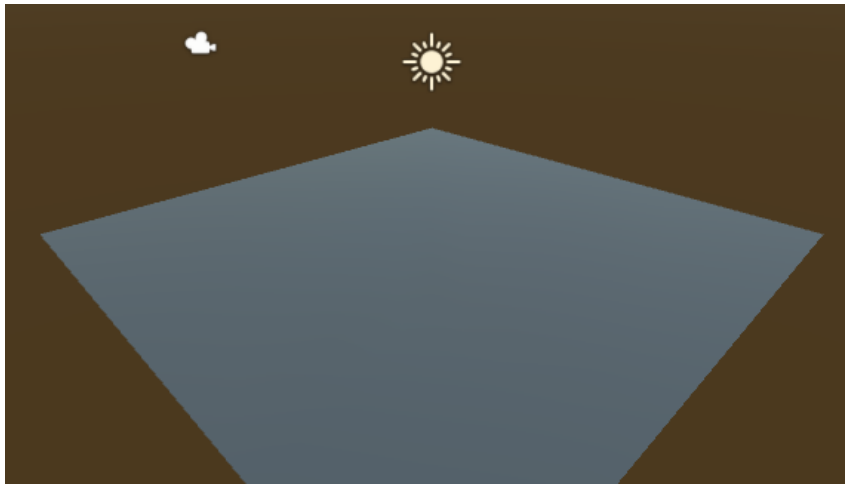
(Môžete si pomáhať kódmi a assetmi zo vzorového projektu)

1. hracia rovina

GameObject / 3D Object Plane

- **upravíme Transform na 0,0,0**
- **Cez *F* sa pozrieme na celú rovinu**
- **má meshCollider**

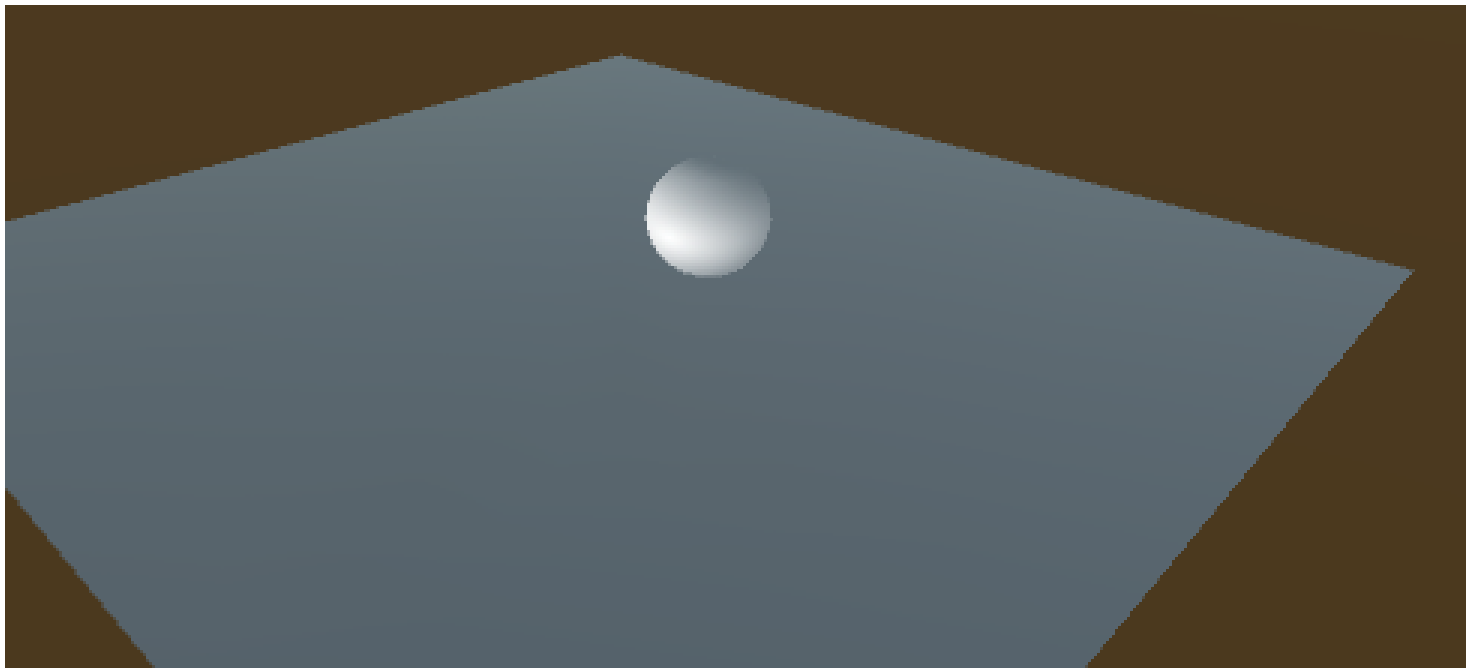
Gizmo – vypnime show grid



2. Guľa na kotúľanie

GameObject / 3D Object / Sphere

- **upravíme Transform tools polohu aby bola nad rovinou**
- **Mesh Renderer / Materials / Select Material / assets / ...**

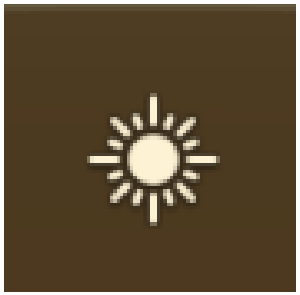


3. Kamera

Nastavíme taký transform kamery, aby sme rovinu dobre videli a celú aj pri spustení hry

4. Osvetlenie

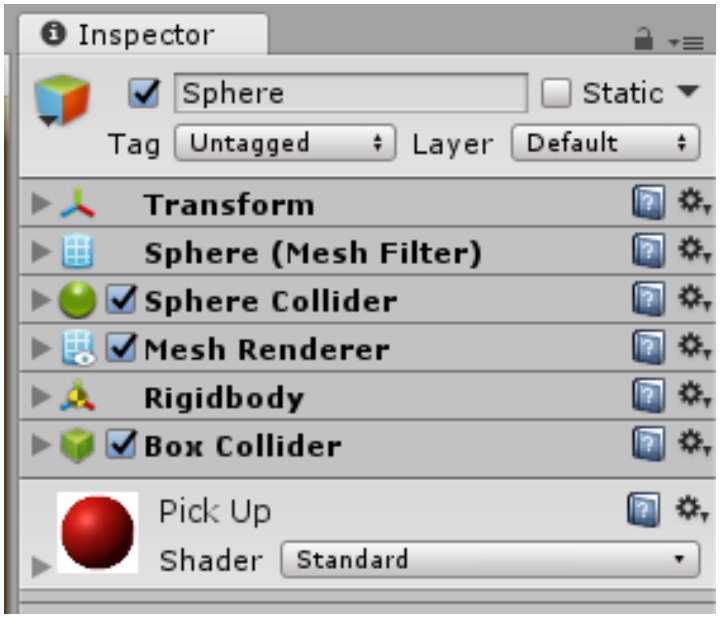
Upravíme uhol a výšku osvetlenia, napr. Y=60



5. Pohyb guličky

Pridáme jej Physics / Rigid Body a Sphere Collider

Pridáme jej skript v C#, ktorý v Rigidbody mení sily
Podľa `Input.GetKey()` alebo `Input.GetAxis("horizontal")`




```
public Rigidbody rb;
```

alebo (lebo je to naše r.b.)

```
Private Rigidbody rb;
rb = GetComponent<Rigidbody>();

rb.AddForce(
        new Vector3(fx,0.0f,fz)
);
```

6. Pohyb kamery

Kamere pridáme skript meniaci jej transform podľa polohy guličky, k nej sa dostaneme cez pridaný public atribút typu GameObject.

```
public class CameraMovement : MonoBehaviour {
    public GameObject player;
    private Vector3 offset;

void Start () {
    offset = transform.position - player.transform.position;
}

void Update () { // LateUpdate
    transform.position = player.transform.position + offset;
}

}
```

7. Prekážky

Narobíme do scény prekážky, niektoré, aby gulička neušla (rigidbody & collider), niektoré také, že majú len collider, tým zapneme trigger a otagujeme ich (Wall, Obstacle). Využijeme pritom prefabrikáty. Trigger ošetríme v skripte guličky a zasiahnutý objekt likvidujeme

```
void OnTriggerEnter(Collider other) {
  if (other.gameObject.CompareTag ("tag")) {
    //other.gameObject.SetActive (false); //invisible
    Destroy(other.gameObject); //removed
  }
}
```

Uloženie projektu

- **File – Save Scene**
- **File – Save Project**

- **Zavrieme Unity**