Game Engines

Andrej Lúčny

Katedra aplikovanej informatiky

lucny@fmph.uniba.sk

http://www.agentspace.org/ge

Prvý skript

(otvoríme projekt cviko2)
Chceme, aby gulička lietala zľava doprava

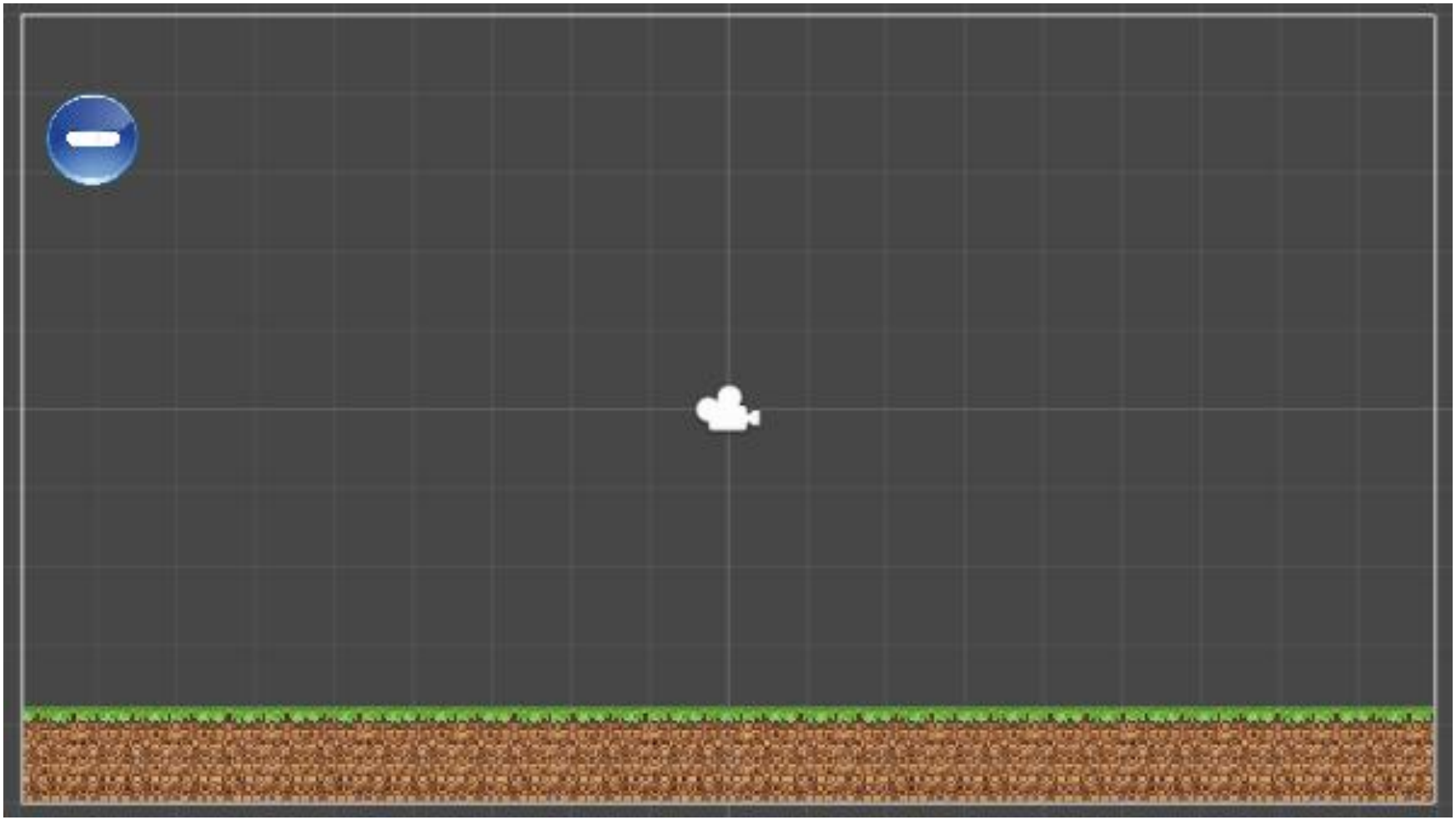
Pridáme jej skript
- **Add Component – New script**
- **Pomenujeme Ball**
- **Create and Add**
- **Kolečko – Edit script**
- **Otvorí sa nám Visual Studio s C#**

Skripty sú asociované s class-ou takže ich názov musí začínať veľkým písmenom.

Unity je previazané s C# ale nie je nevyhnutné tento jazyk ovládať na to, aby ste mohli skriptovať, nevyhnutný je help

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Ball : MonoBehaviour {

// Use this for initialization
void Start () {
}

// Update is called once per frame
void Update () {
    transform.position += new Vector3(0.1f, 0.0f);
    if (transform.position.x > 8)
    {
        transform.position = new Vector3(-8.0f, 3.42f);
    }
}

}
```

Základná scéna



dva assety (guličku zmenšujte so shiftom)

Prvý skript

Chceme, aby gulička lietala zľava doprava

Pridáme jej skript

- **Add Component – New script (úplne dole)**
- **Pomenujeme Ball**
- **Create and Add**
- **Kolečko – Edit script**
- **Otvorí sa nám Visual Studio s C#**

Skripty sú asociované s class-ou takže ich názov musí začínať veľkým písmenom, nesmie obsahovať medzery, mínus a podobne

Používame docs.unity3d.com/ScriptReference

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Ball : MonoBehaviour {

// Use this for initialization
void Start () {
}

// Update is called once per frame
void Update () {
    transform.position += new Vector3(0.1f, 0.0f);
    if (transform.position.x > 8.0f)
    {
        transform.position = new Vector3(-8.0f, 3.42f);
    }
}

}
```

Fyzika

Physics 2D

- guličke potrebuje komponent Rigidbody 2D

- oba assety potrebujú detektory zrážky - Box Collider 2D a Circle Collider 2D

Teraz chceme, aby gulička lietala zľava doprava a nepadala, ale stále mala Rigidbody 2D

- **Nastavíme do Rigidbody 2D ako bodyType nie Dynamics ale Kinematics, tj. nepodlieha fyzike ale hýbe sa sama**

Reálny čas

Keby sme púšťali projekt na výkonnejšom počítači, gulička by išla rýchlejšie. Dosiahnuť pohyb v reálnom čase sa dá pomocou statických atribútov triedy Time

`Time.deltaTime`, 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.

Násobíme ním želanú rýchlosť, čiže ak sa má gulička hýbať 2m/s, k polohe pripočítame `Time.deltaTime*2.0f`

Timery

`Time.deltaTime` je možné spočítavať do akumulačnej premennej, ktorá je privátnym atribútom class-y reprezentujúcej skript. Možno tak urobiť jednoduchý časovač

(zatiaľ nepotrebujeme)

Interakcia s užívateľom

Použite viacero kľúčov, napr. na skákanie

```
Input.GetKey(KeyCode.Space)
```

Myš

```
Input.GetMouseButton(0)
```

Pridávanie atribútov skriptu

A teraz chceme aby padla, keď stlačíme klávesu

```
- if (Input.anyKey) ...
```

a chceme použiť niečo ako

```
- body.bodyType = RigidbodyType2D.Dynamic;
```

kde ale to body vezmeme? (to nemáme ako transform) Zadefinujeme

```
public Rigidbody2D body;
```

A tým v Unity pribudne atribút body pre skript Ball (chvíľu to trvá a len ak nie sú kompilačné chyby)

Pridávanie atribútov skriptu

Do neho ako hodnotu vložíme smerník na Rigidbody 2D tým, že ho drag & drop potiahneme na dané miesto.

A podľa Input teraz nastavujeme do bodyType Dynamics

A po dopade (podľa transform.position.y – viď hodnoty v Transform) nastavíme zase Kinematics a pôvodnú polohu.

Pomáhajte si nápovedou C# a sledujte, či sa skript podarilo skompilovať. (Najľahšie sa o tom presvedčíte spustením scény, čo sa pri kompilačnej chybe skriptu nedá. Podrobnosti vidno v záložke Console v spodnom okne).

Skryté vlastnosti

Ale nerobí to to, čo chceme.

Ešte musíme pri zmene polohy vynulovať rýchlosť (viď Rigidbody2D Info počas spustenia scény)

```
body.velocity = new Vector3(0.0f,0.0f);
```

Teraz je to ono!

Môžeme dať do cesty guličke viacero prekážok a trochu sa zahrať

Získanie svojich atribútov

Získanie prístupu k atribútom komponentov herných objektov cez atribúty skriptu je najuniverzálnejší mechanizmus.

V prípade, že ide o atribút komponentu objektu, ktorého komponentom je daný skript to možno urobiť aj priamočiaro:

```
GetComponent<Rigidbody2D>().velocity = …
```
alebo `gameobject.GetComponent<Rigidbody2D>() …`

Môžeme taktiež nejako získať pointer na iný objekt a :
```
gameobj.GetComponent<Rigidbody2D>() …
```

Tagy

Ďalším spôsobom, ako sa k tomu dostať sú tagy.

Všimnite si, že každý game object má svoj tag, vidno ho hore v Inspectore

Default je Untagged a tagy možno pridávať a objekty tak pomenúvať i radiť do skupín.

Pridajte tag ball a dajte ho guličke.

Tagy

Tag sa dá použiť pri skriptovaní:

```
GameObject obj=GameObject.FindWithTag("tag");

if (obj.CompareTag("tag"))
```

A potom

```
obj.GetComponent<Rigidbody2D>().velocity = …
```

pridajte trávniku skript, ktorý na základe stlačenia A pohne guličkou a nebude mať vstupný atribút

Layer-y (vrstvy)

Okrem tagov možno každému objektu určiť layer, vidno ho v inspectore vedľa tagu

Layery ležia na sebe a číslujú sa od 0

Layery tiež možno pridávať.

Pridajte layer invisible (add layer v Inspectore alebo v okne tags and layers) a prekážkam priraďte tento layer

Keď potom kamere nastavíte do Culling mask tento layer, nebude pri spustení scény prekážky vidno, ale gulička sa bude od nich odrážať.

Spájanie objektov do celkov

Zduplikujte guličku

Add Component
Physics2D / Fixed Joint 2D

a vyberte meno „circle (1)“ ako hodnotu Connected Rigid Body

Hra

Navrhnite s dostupnými vedomosťami jednoduchú hru

Pokúste sa ju implementovať

Ktoré funkcie by sa Vám k implementácii hodili?