Lesson 2

Game Objekti

Svi game objekti na sceni na sebi imaju komponente.
Komponente određuju kako će se game objekt ponašati u sceni, tj. u igri.

Ako želimo da se **Game Objekt** ponaša drukčije moramo mu promijeniti komponente.

Pogledajmo naše vozilo, trenutno na sebi ima par komponenata, čak nema ni fiziku. – Podignite ga i vidite da stoji na mjestu.

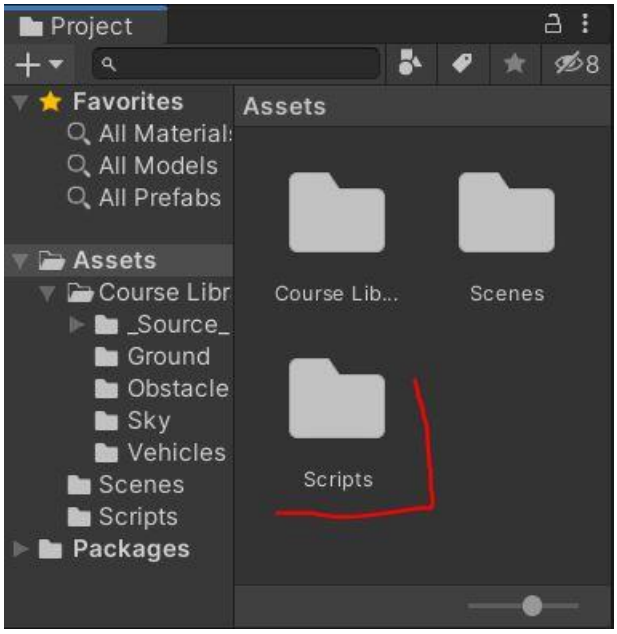
Moramo mu dodati komponentu koja mu daje fiziku.

Sada ćete vidjeti kako je programiranje moćno i što sve možemo samo s jednom linijom koda.

Skripte

Napravimo novi folder – Skripte.

Sve mora biti organizirano, to je jako bitno.



Napravimo novu skriptu – PlayerController.

Skriptu – Svako početno slovo veliko bez razmaka. Zašto?

Naming Convenction – to je standard koji developeri koriste.
Skripta bi radila i sa malim slovima, ali to je neko pravilo koje developeri koriste. I to je bitno zbog organizacije, sve mora biti lijepo posloženo i organizirano.

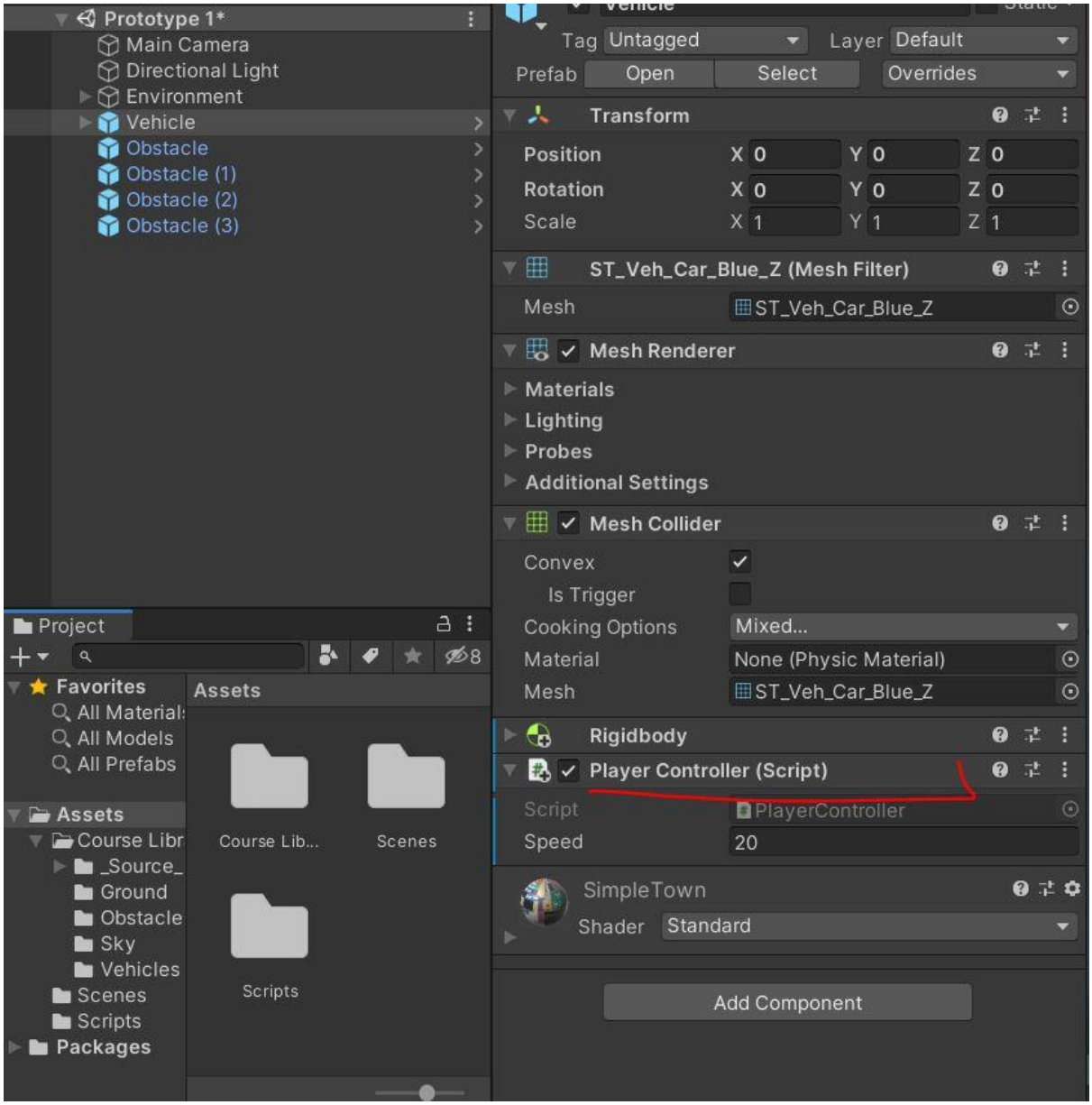
Ako netko nastavi raditi na projektu nakon vas sve mora biti čitljivo i organizirano.

Skripte nazivamo imenima – što one rade.

Dodaj skriptu na vozilo

Mi programiramo ponašanje objekata i ako želim da se vozilo ponaša onako kako smo isprogramirali. Moram skriptu dodati na vozilo.

Drag and Drop



Desno vidimo da smo dodali još jednu komponentu.

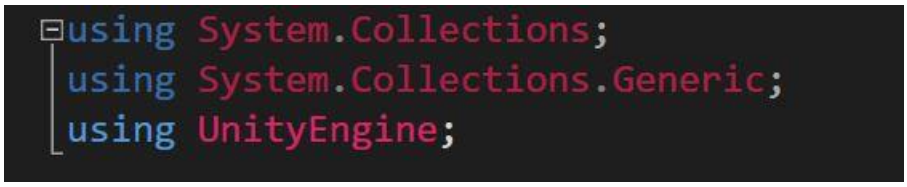
Komponente su svojstva nekog objekta.

Skripte – koristimo da Unity-u kažemo što želimo da objekt radi.

Otvorimo Skriptu

Skripta nam se otvorila u VSC-u, a to je naše okruženje za programiranje.

Na početku skripte vidimo **library** – s time pristupamo knjižnicama Unity-a, u kojima imamo gotove stavke koje ćemo koristiti. Neke knjižnice su ovdje, a neke ćemo dodavati mi.



Ispod toga imamo ime naše klase koja je odvojena sa dvotočkom od **MonoBehavior-a.** To znači da naša class-a/skripta zapravo potječe iz druge klase. Što znači da je **MonoBehaviour** glavna klasa, a klase koje mi kreiramo su njezine podklase.

Monobehaviour je glavna klasa iz koje ćemo povlačiti metode koje su već kreirane.

Za sada imamo kreirane dvije metode koje smo uzeli iz **MonoBehaviour** klase.

Start i Update.

Te metode su potjekle iz klase **MonoBehaviour.**

Sintaksa metode je:

void, ime metode, oble zagrade i vitičaste zagrade.

```
// Syntax
void Start()
{
    // Your code
}
```

Naš kod pišemo između vitičastih zagrada.

Start – poziva se samo prvi frame

Update – poziva se svaki frame

print(“hello world“);

Komentari

Računalo ne čita komentare.

```
// Update is called once per frame
void Update()
{
    // We'll move the vehicele forward
}
```

Sada napravimo komentar Update metodi.

Vozilo forward

Za sve što radimo u Unity-u koristimo C# programski jezik.

Z - axis se mijenja kada mičemo vozilo naprijed.

Transform komponenta - gledamo koju **OS** pomičemo kada idemo naprijed.

Znači moramo pristupiti **Transform** komponenti.

Transform sa velikim T znači da pristupamo Transform klasi koja u sebi ima kreirane metode.

transform sa malim t znači da pristupamo komponenti objekta.

transform. nakon točke pristupamo drugim metodama

transform.Translate – Translate je metoda jer ima veliko početno slovo.

Točka nam dopušta pristup drugim metodama u različitim klasama.

Ovdje kažemo da želimo **Translate** poziciju vozila koristeći **transform** komponentu.

Sada želimo reći računalu da izvrši zadatak koristeći oble zagrade.

Sada u zagrade moramo upisati koju **OS** želimo mijenjati.

X Y Z.

Točka zarez

Govori našoj skripti da je ovo kraj naše linije koda.

Update se poziva 60 puta u sekundi.

Znači da će naše vozilo prijeći 60 metara u sekundi.

CTRL + S - obavezno

```
// Update is called once per frame
void Update()
{
    transform.Translate(0, 0, 1);

}
```

Više rješenja

Skraćenice

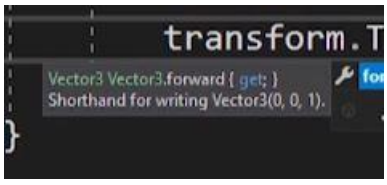
U programiranju do istog rješenja možemo doći na više načina.

0,0,1 – tri dimenzije. Pozicije naše **transform** komponente.

To možemo skratiti da dodamo Vector3 – Vector3 su pozicije.

X,Y,Z.

Dodajemo **Vecrtor3.forward** je skraćenica za 0,0,1.



Isto tako imamo left, right, up, down, back – skraćenice.

```
// Update is called once per frame
void Update()
{
    //transform.Translate(0, 0, 1);
    transform.Translate(Vector3.forward);

}
```

Usporavanje vozila

Update frame ovisi o brzini vašeg računala – to nije pravilan način jer su računala brža i sporija.

Zato Update umjesto svakog frame-a moramo staviti da se Update-a, po sekundama. Jer vrijeme je uvijek isto, bez obzira koje računalo koristimo.

Zato naše kretanje moramo pomnožiti s vremenom.

Za to postoji kreirana klasa koja prati vrijeme.

Koristimo **Time** koje prati naše vrijeme.

Delta time – kontrolira koliko je vremena prošlo.

Sada se kreće jedan metar u sekundi.

Prije se kretalo jedan metar po frame-u.

```
// Update is called once per frame
void Update()
{
    //transform.Translate(0, 0, 1);
    transform.Translate(Vector3.forward *  Time.deltaTime);

}
```

Sada pomnožimo s nekim brojem i možemo točno definirati koliko metara u sekundi želimo da se vozilo kreće.

```
// Update is called once per frame
void Update()
{
    //transform.Translate(0, 0, 1);
    transform.Translate(Vector3.forward *  Time.deltaTime * 5);

}
```

Vector3.forward sprema 1, ako pomnožimo sa 5, krećemo se 5 metara u sekundi.

RigidBody

Sada na naš objekt moramo dodati komponentu koja nam simulira fiziku.

Dodajte na vozilo – istražite masu i use gravity.

Dodajte na obstacle.

Dodajte kilažu na opstacle.

Collider

Još jedna bitna komponenta koja nam služi za okvir oko objekta je **Collider.** Isključite collider na obstacle.

Dupliciranje

Dupliciraj objekte.

Pomakni po Z osi.

Lesson Recap

Što smo naučili?

C# Scripts

Start Vs Update

Comments

Metods

Parametars

Time.deltaTime

Components

Collider and RigidBody