Szórakoztatóelektronikai eszközök programozása

Dr. Hideg Attila

Hideg.Attila@aut.bme.hu

2024. március 6.

BME AUT

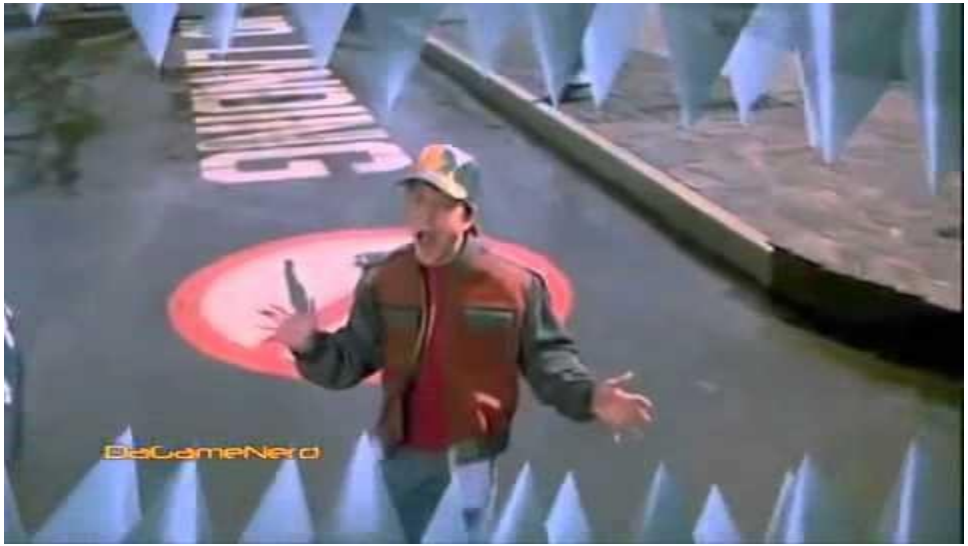
Automatizálási és
Alkalmazott
Informatikai Tanszék


unity

Unity

- Cross platform „game” engine
- C++ alapú, de C# wrapper-ben
- 2D-s és 3D-s interaktív tartalom
- Saját „all-in-one” fejlesztőkörnyezet
  - Kivéve Code Editor
    - Ajánlott: JetBrains Rider, Visual Studio, Visual Studio Code
- Komoly 3rd party támogatás
- Kezd túlnőni a játékfejlesztésen

„Miért jó ez nekünk?”



„Miért jó ez nekünk?”

- 3D tartalom interaktív megjelenítése
- AR/VR/XR támogatás
- Multiplatform -> Közös kód- és erőforrás-bázis
  - TV-re, konzolra, mobilra, PC-re
- Sokrétű fejlesztőkörnyezet
- Asset Store -> Számtalan „off-the-shelf” komponens

Története

- 2005
  - Apple World Wide Developer Conference (WWDC)
  - OS X exclusive game engine
- 2008
  - iOS támogatás
- Kezdetben nincs ingyenes csomag
- Később egyes funkciók csak a előfizetéssel érhetőek el
  - Pl. Inverz Kinematika

Jelenleg

- 18+ támogatott platform

A játékfejlesztésen túl

- Mára a Unity számos egyéb szolgáltatást is nyújt a játékfejlesztésen túl:
- Cinemachine
  - Filmkészítés és effektek https://unity.com/unity/features/editor/art-and-design/cinemachine
- PiXYZ
  - CAD fájlok real-time vizualizációja
  - https://unity3d.com/pixyz
- Weta Digital

Előfizetési mód

- Personal - ingyenes
  - Teljes funkcionalitás, de semmi más
- Student – ingyenes
  - Hasonló a Pro-hoz, de diáknak kell lenni
- Plus – kb. 40 $/hó*
  - Tananyag (Learn Premium), támogatás
- Pro – kb. 150 $/hó
  - Még több támogatás, analitycs
- Egyéb kiegészítő szolgáltatások

*Éves előfizetéssel olcsóbb, mint havonta fizetve

Asset Store

- There is an ~~App~~ Asset for that
- Saját „belső” bolt, amiben mindenfajta Asset-et meg lehet vásárolni.
- Tematikusan kategorizált tartalom
- Integrálva van az Editorba, de webes felülete is van hozzá
- A letöltött Asset-eken nincs védelem, így gyakorlatilag szabadon rendelkezünk* felettük

Fejlesztés eszközei

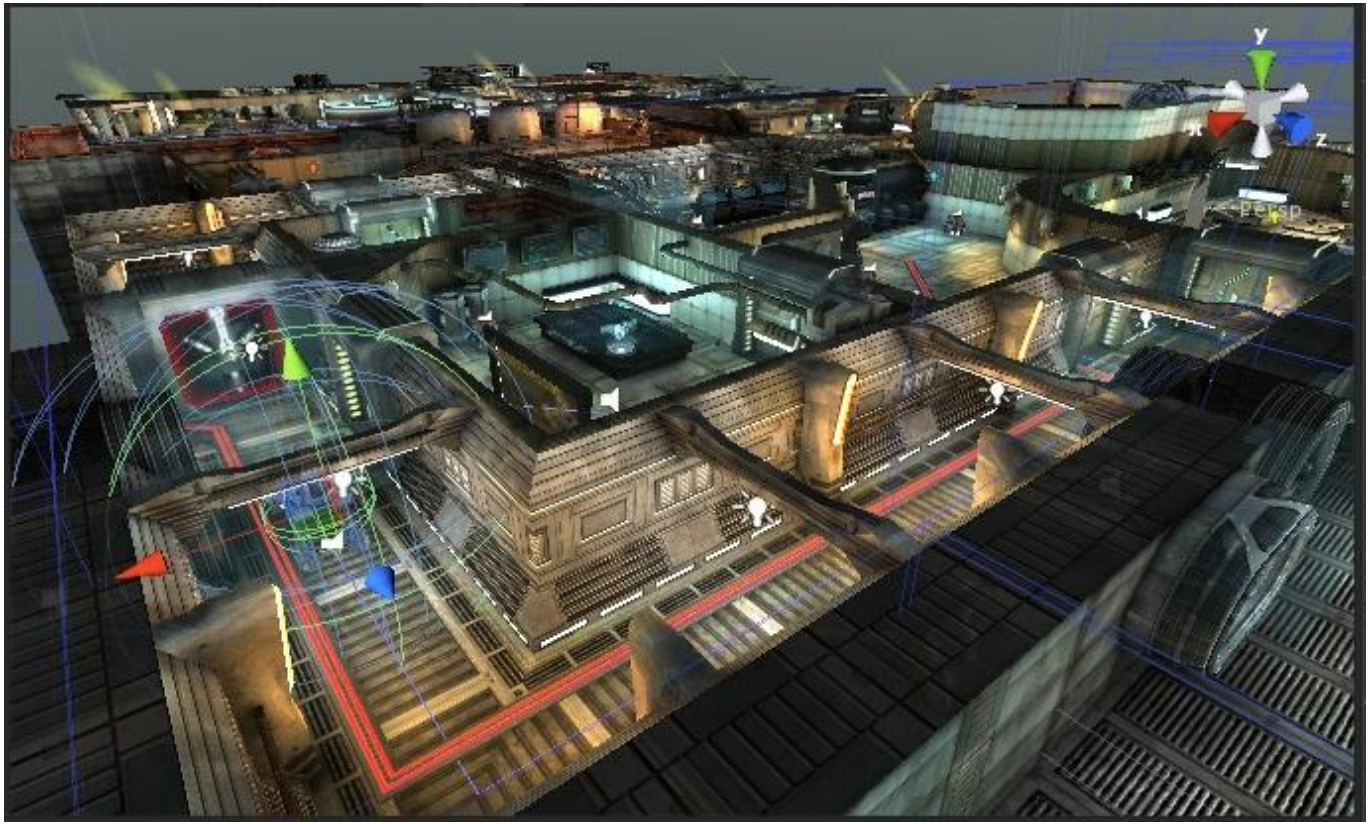
- Unity Editor
  - A fejlesztés nagy része – minden, ami nem kód írás – itt történik.
- 3rd party Text Editor
  - A kódoláshoz használt szövegszerkesztő
  - Bármi lehet, de a *Visual Studio/JetBrains Rider* az ajánlott
    - OS X-en is

A Unity alkalmazások felépítése

Scene

- Minden Unity alkalmazás Scene-ekből (jelenetekből) áll
- Ezeket a jeleneteket objetumokkal tudjuk „berendezni”
- Minden ilyen objektum, amit el tudunk helyezni egy jelenetben GameObject
  - Egy konténer és egyben interfész a rendszer callback-jeinek eléréséhez

Unity Editor - Scene



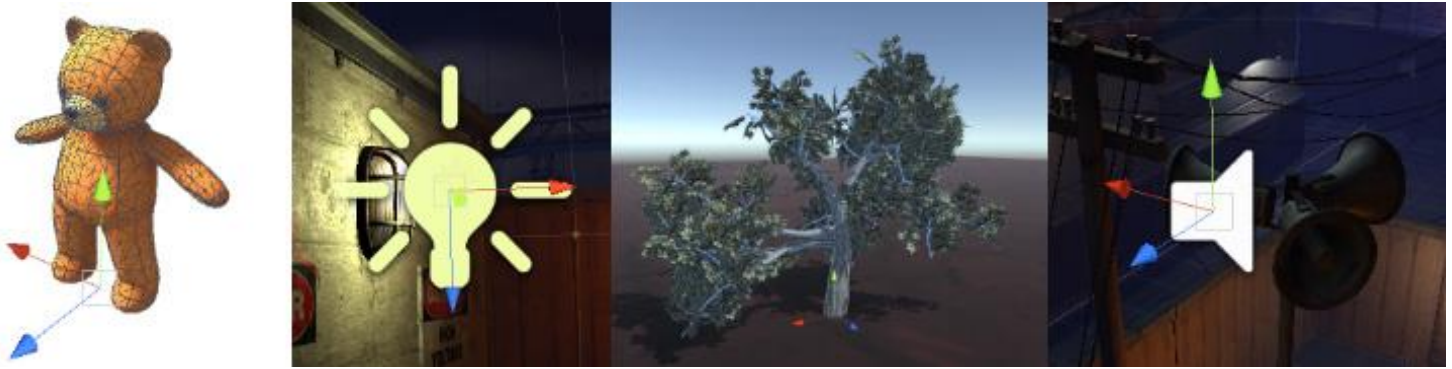
GameObject

- Elemi „objektuma” minden Unity alkalmazásnak
- Rendszer által kínált GameObject „sablonok”:
  - Fények
  - Kamerák
  - Statikus és dinamikus 2D-s és 3D-s objektumok
  - Canvas (lásd. később)
  - Részecskerendszerek
  - Stb.

„GameObject

- Minden GameObject komponensekből (*Component*) áll.
- Alapértelmezettként mindegyik tartalmaz egy *Transform* komponenst, melyet nem lehet eltávolítani.
  - Ez határozza meg az objektum
    - pozícióját
    - orientációját
    - méretét
- További komponenseket tetszőlegesen hozzáadhatunk, illetve elvehetünk bármilyen GameObject-től.
  - Pl.: Kamera mellé fényforrást vagy particle systemet is adhatunk

GameObject



Components

- Önállóan nem lehet egy jelenetbe elhelyezni
- Csak egy GameObject-tel együtt teljesértékűek
- A GameObject-hez tartozó script el tudja érni őket, illetve egymást is képesek elérni
  - ha tudnak egymásról
- A komponensek *property*-ket tartalmazhatnak.

Components

- További komponens típusok:
  - Material
  - Script
  - AudioSource
  - Collider
  - Mesh Renderer
  - Bilboard
  - Stb.

Prefab

- Így lehet „specializálni” egy GameObject-et
- *Asset**, ami eltárol egy GameObject-et, az összes gyermek GameObject-jét, az összes komponensét és azok property-jait
- Drag&Drop-pal lehet létrehozni
  - Összeáálítjuk a GameObject-et, majd behúzzuk a projektünk asset-ei közé.

Asset

„An asset is representation of any item that can be used in your game or project. ”

- Gyakorlatilag minden állomány, ami a projektünkben van.
- Lehet „helyben”, editorral is létrehozni, de hozzá is adhatjuk a projekthez.
  - Textúrák
  - Modellek
  - Animációk
  - Hangok

Dinamikus viselkedés - Script

- A GameObject-ekhez komponensként *Script*-ek rendelhetők, melyek C# fájlokként (.cs) tárolódnak
  - Korábban léteztek alternatív nyelvek, de mára már a C# az egyedüli támogatott
  - A Script törzsét az Editor legenerálja, a többit nekünk kell megírni
  - Külső szerkesztővel kell szerkeszteni

Script

- Egy .NET-es osztály, mely a *MonoBehaviour* osztályból származik.
- A keretrendszer callback-ként hívja meg az egyes felüldefiniált függvényeit, ezek az *Event*-ek:
  - void Update() – Minden új *frame*-nél
  - void FixedUpdate() – Fix időközönként, a frameratehez igazodva
    - fizikai alrendszer frissítéséhez
  - void Start() – A GameObject létrehozásakor
  - void Awake() – A Start() előtt
    - Olyan dolgok inicializálására, melyekre a Start()-ban már szükségünk lesz
  - Ha új komponenseket adunk hozzá, akkor az azokhoz tartozó eseményeket is le tudjuk kezelni
    - Ütközés
    - Input
    - *AudioClip* lejátszása végetért.
  - Az egyes GameObject-ek között a hívási sorrend nem determinisztikus
    - Editorban a kard előbb inicializálódik, mint a pajzs, de játék közben meg pont fordítva

Script – property-k

- A script-ek tartalmazhatnak property-ket:
  - Referencia más GameObjectre
  - Referencia más Componentre
  - Vector3, int, double, float, stb. változók
- Láthatóságra ugyanolyanok, mint a hagyományos C# property-k
- Editorban nem látszanak, csak akkor ha
  - `public` a láthatóságuk
  - speciális annotációval látjuk el őket: `[SerializeField]`

Script - komponensek kezelése

- A tartalmazott komponensek elérése:
  - `T GetComponents<T>();`
  - `T GetComponentsInChildren<T>();`
  - `T GetComponentsInParent<T>();`
  - Ha nincs ilyen komponens, akkor *null*-t ad vissza.
- Komponens hozzáadása:
  - `T AddComponent<T>();`

GetComponent

```
using UnityEngine;

public class GetComponentGenericExample : MonoBehaviour
{
    void Start()
    {
        HingeJoint hinge =
                        gameObject.GetComponent<HingeJoint>();

        if (hinge != null)
            hinge.useSpring = false;
    }
}
```

AddComponent

```
using UnityEngine;
using System.Collections;

public class AddComponentExample : MonoBehaviour
{
    void Start()
    {
        SphereCollider sc =
            gameObject.AddComponent<SphereCollider>()
                                        as SphereCollider;
    }
}
```

Script – GameObject példányosítás

```
// Creates a game object named "Player" and
// adds a rigidbody and box collider to it.

using UnityEngine;

public class ExampleScript : MonoBehaviour
{
    void Start()
    {
        GameObject player;
        player = new GameObject("Player");
        player.AddComponent<Rigidbody>();
        player.AddComponent<BoxCollider>();
    }
}
```

Script – Prefab példányosítás

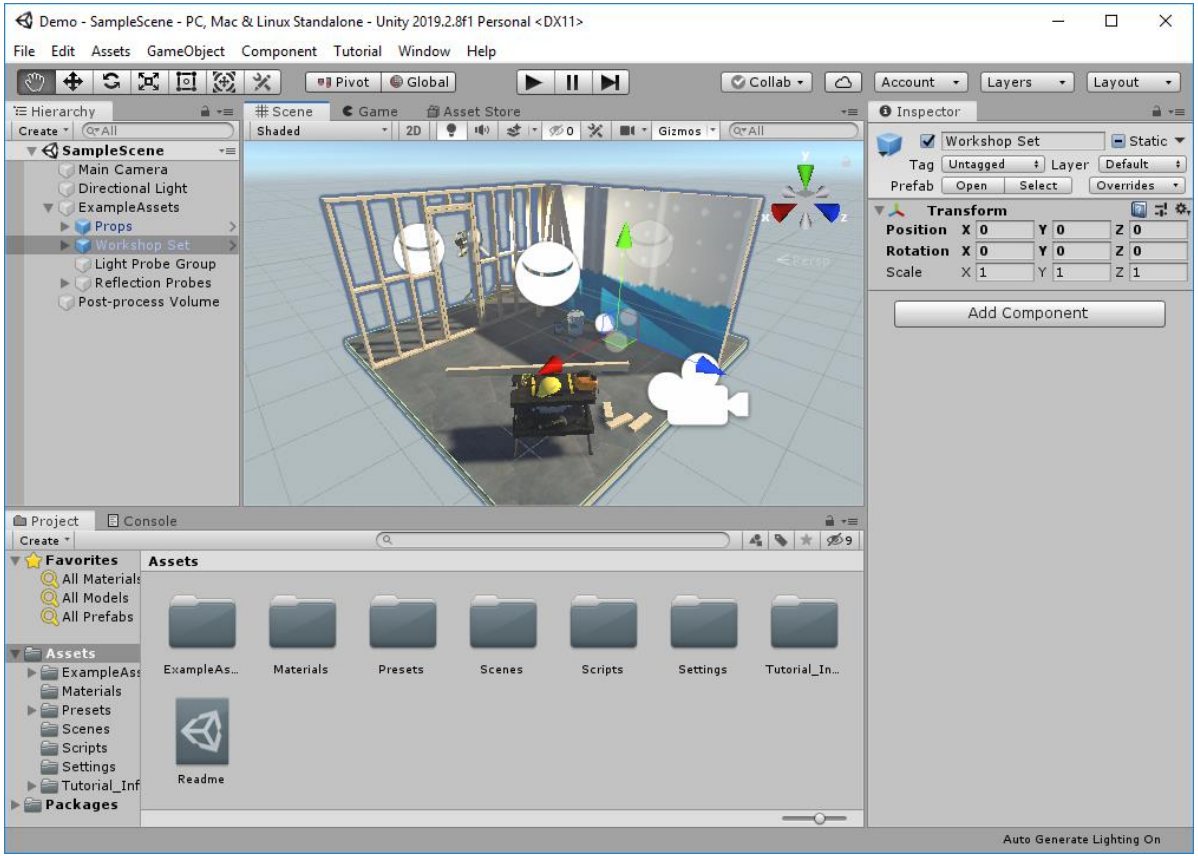
```
using UnityEngine;
using System.Collections;

public class ExampleClass : MonoBehaviour
{
    public Transform prefab;

    void Start()
    {
        for (int i = 0; i < 10; i++)
        {
            Instantiate(prefab, new Vector3(i * 2.0F, 0, 0), Quaternion.identity);
        }
    }
}
```

Projekt könyvtár

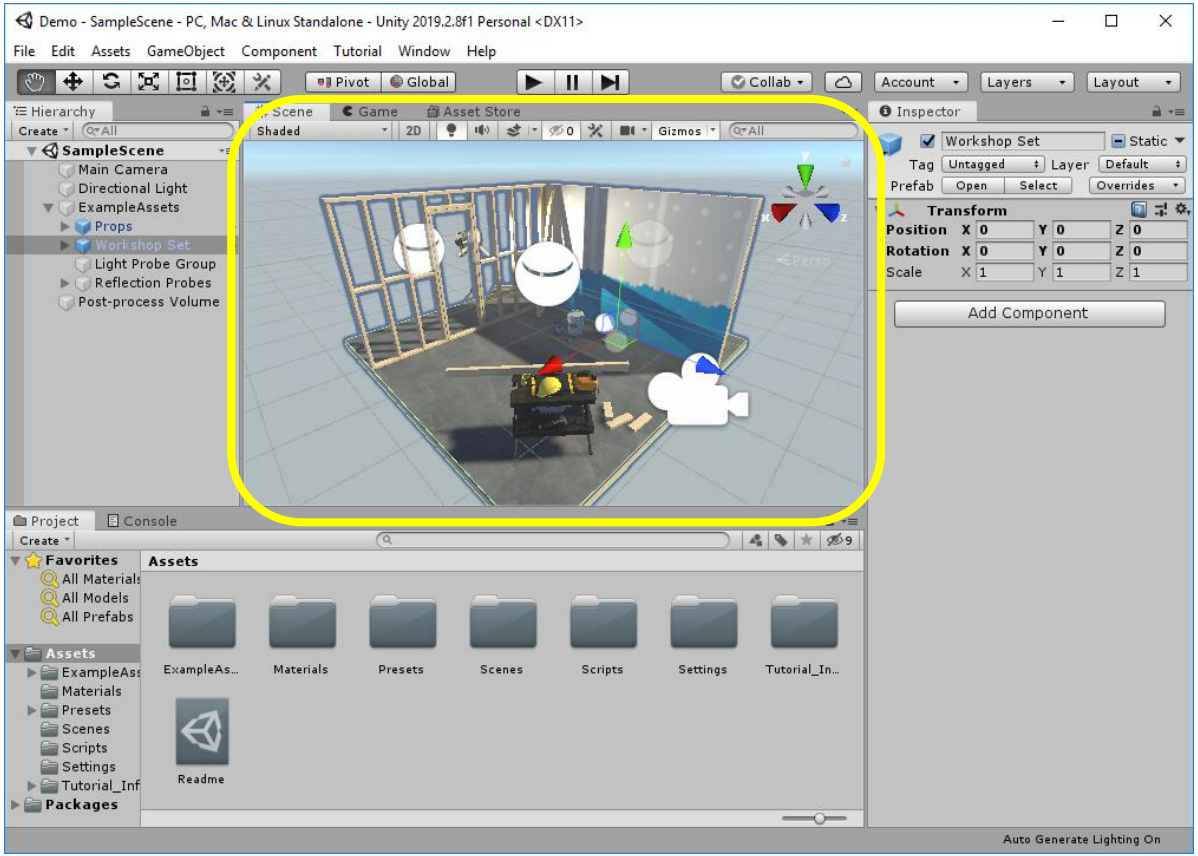
- A Unity-s projektünk *Assets* könyvtárának tartalma egy-az-egyben megjelenik az Editorban
- Azonban minden fájlhoz tartozik egy .meta fájl is, mely az Editor számára szükséges metainformációkat tartalmazza
  - Emiatt nem mindig célravezető a fájlokat közvetlenül másolni egyik projektből a másikba

A Unity Editor

Unity Editor



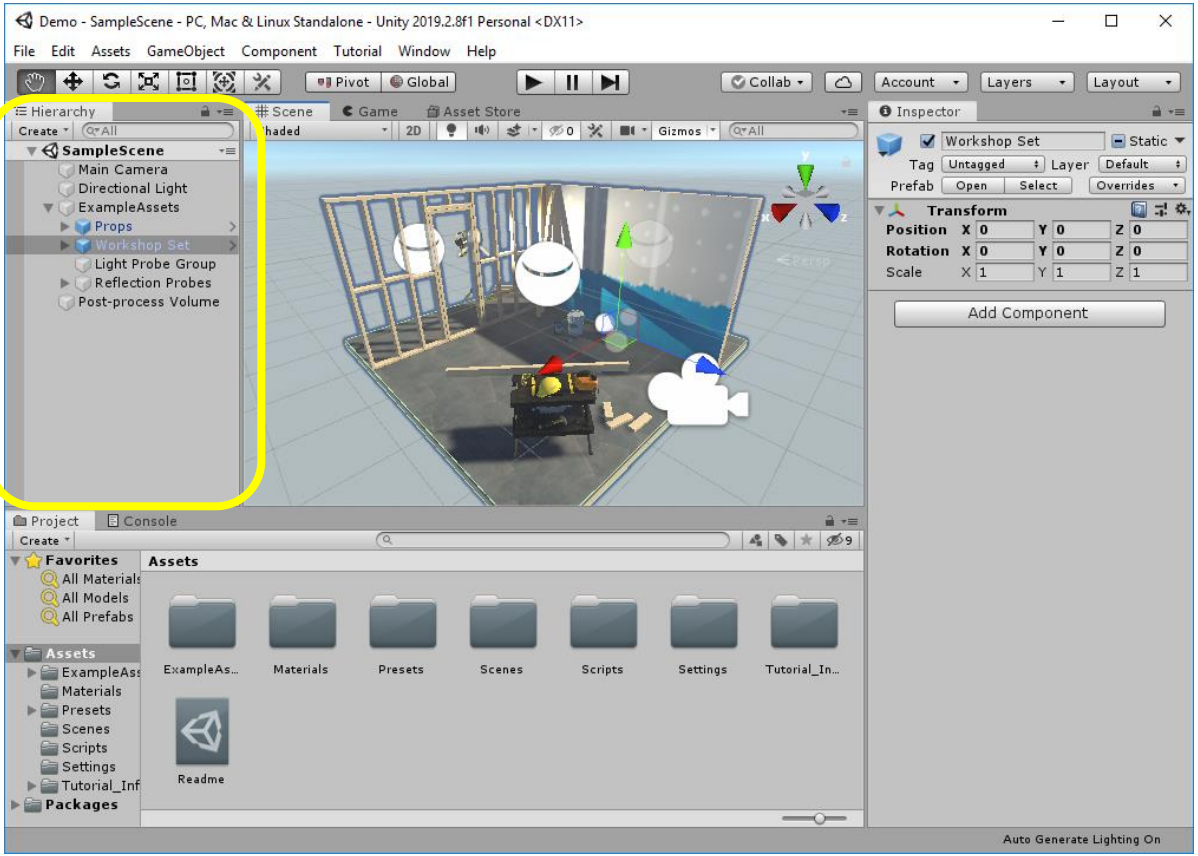
Unity Editor - Scene



Unity Editor – Scene

- Minden Unity alkalmazás Scene-ekből (jelenetekből) áll
- Ezeket a jeleneteket GameObject-ekkel tudjuk „berendezni”
- **A Unity Editorban úgy tudunk benne „dolgozni”, mint egy 3d modellező programban**

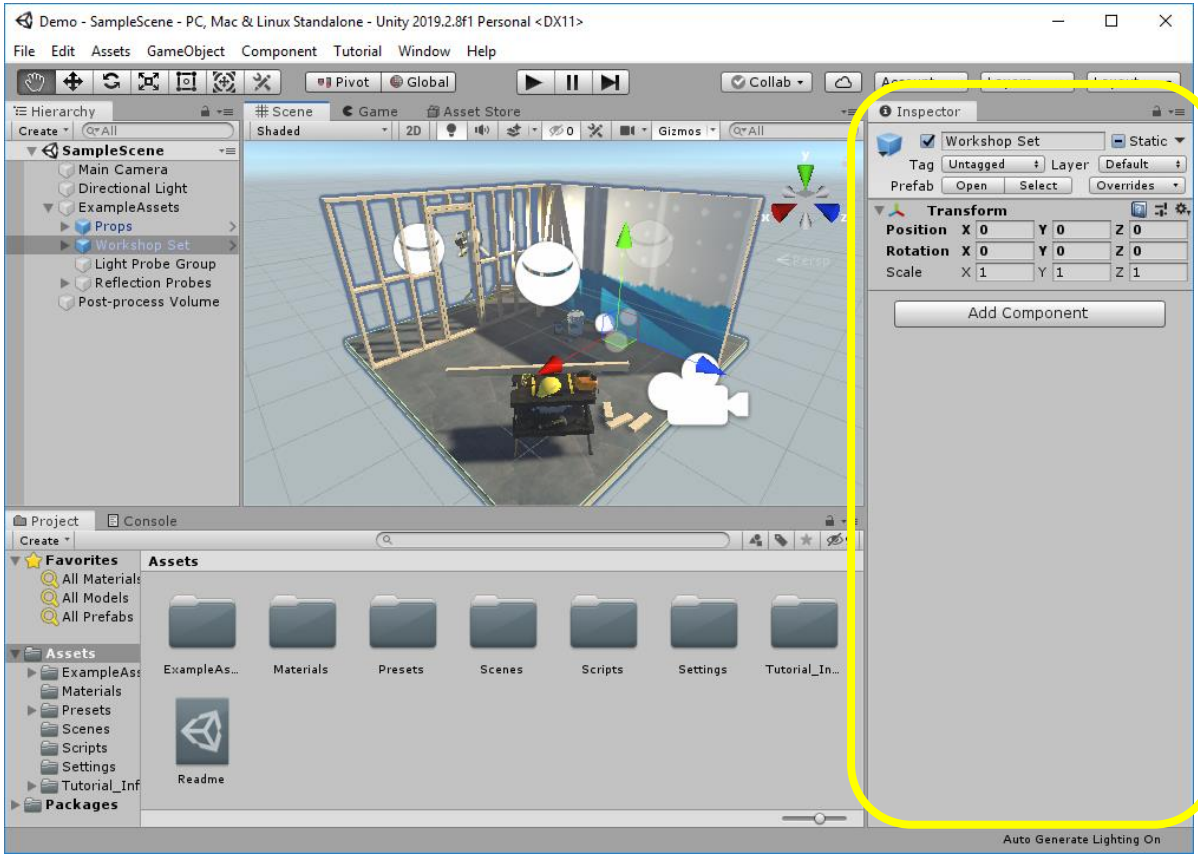
Unity Editor - Hierarchy



Unity Editor - Hierarchy

- Az egy jeleneten belül elhelyezett GameObjeketet jeleníti meg
- Fa struktúra alapú elrendezés
- Az egyes elemeket itt érdemes kijelölni/duplikálni/törölni
- Új GameObject elhelyezése:
  - Sablon GameObject a jobb egérgomb megnyomásával előjövő menüből
  - Saját GameObjectet Drag&Drop módszerrel
    - A Scene nézetre is működik

Unity Editor - Inspector



Unity Editor - Inspector

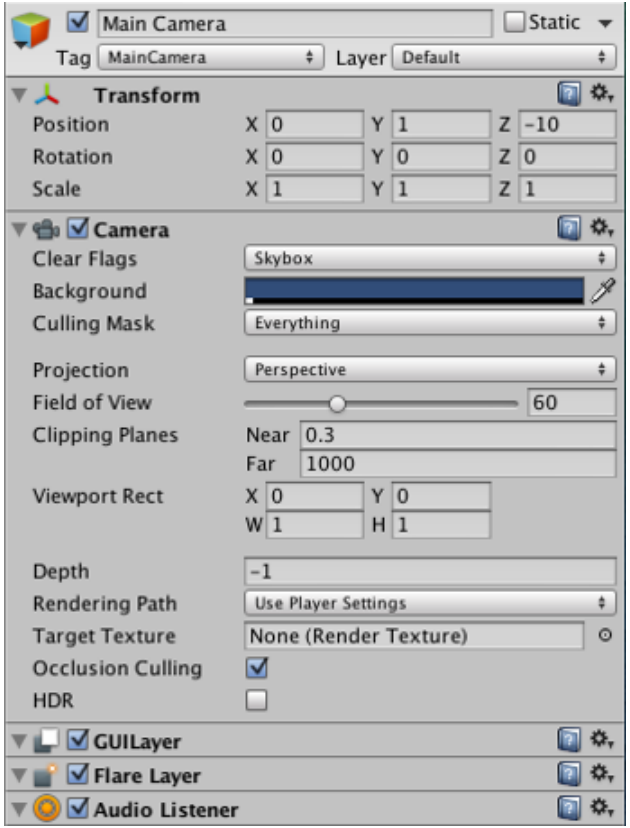
- Lehetőséget biztosít:
  - Az aktuálisan kijelölt GameObject alaptulajdonságainak és komponenseinek konfigurálására.
  - Új komponens hozzáadására
  - Meglévő komponens eltávolítására
  - Az egyes komponensek sorrendjének átrendezésére
  - Zárolni a panelt a kijelölt GameObject-hez
    - Amikor más GameObjectet jelölünk ki, akkor is a zárolt GameObject paramétereit tudjuk módosítani.

Unity Editor - Inspector

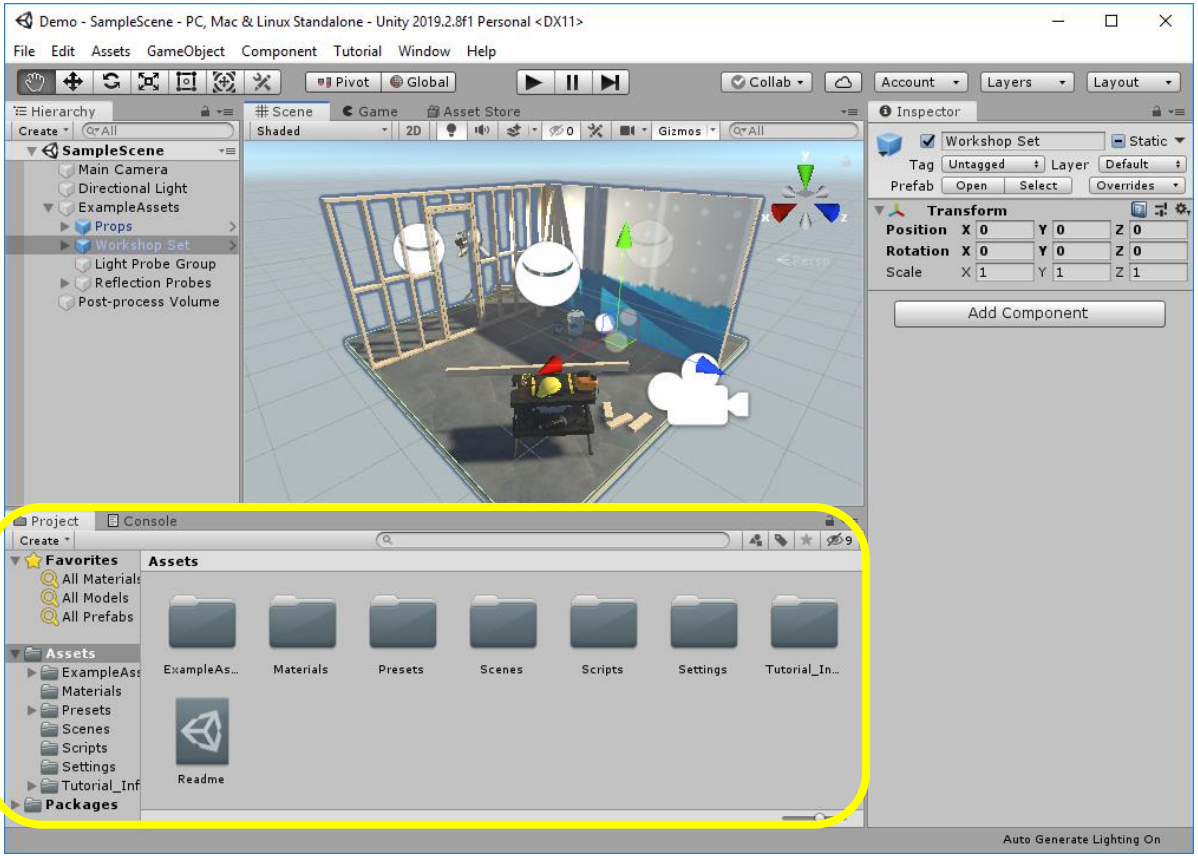
Transform
komponens

Camera
komponens

További
komponensek



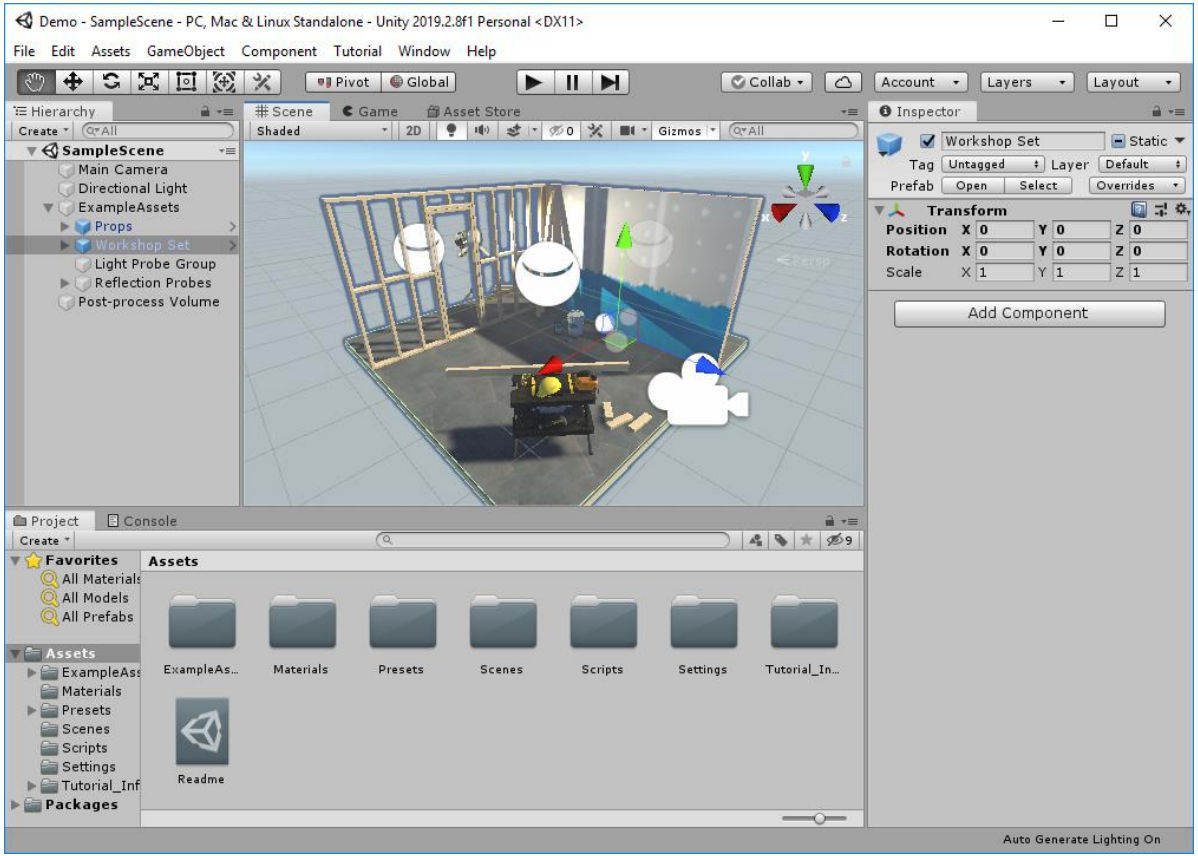
Unity Editor - Project



Unity Editor - Project

- A projekt erőforrásai között teszi lehetővé a tallózást, módosítást
- A Unity projektek könyvtárstruktúrája megegyezik a fájlrendszerbeli megfelelőjével.
  - Nincsenek virtuális csoportok/könyvtárak

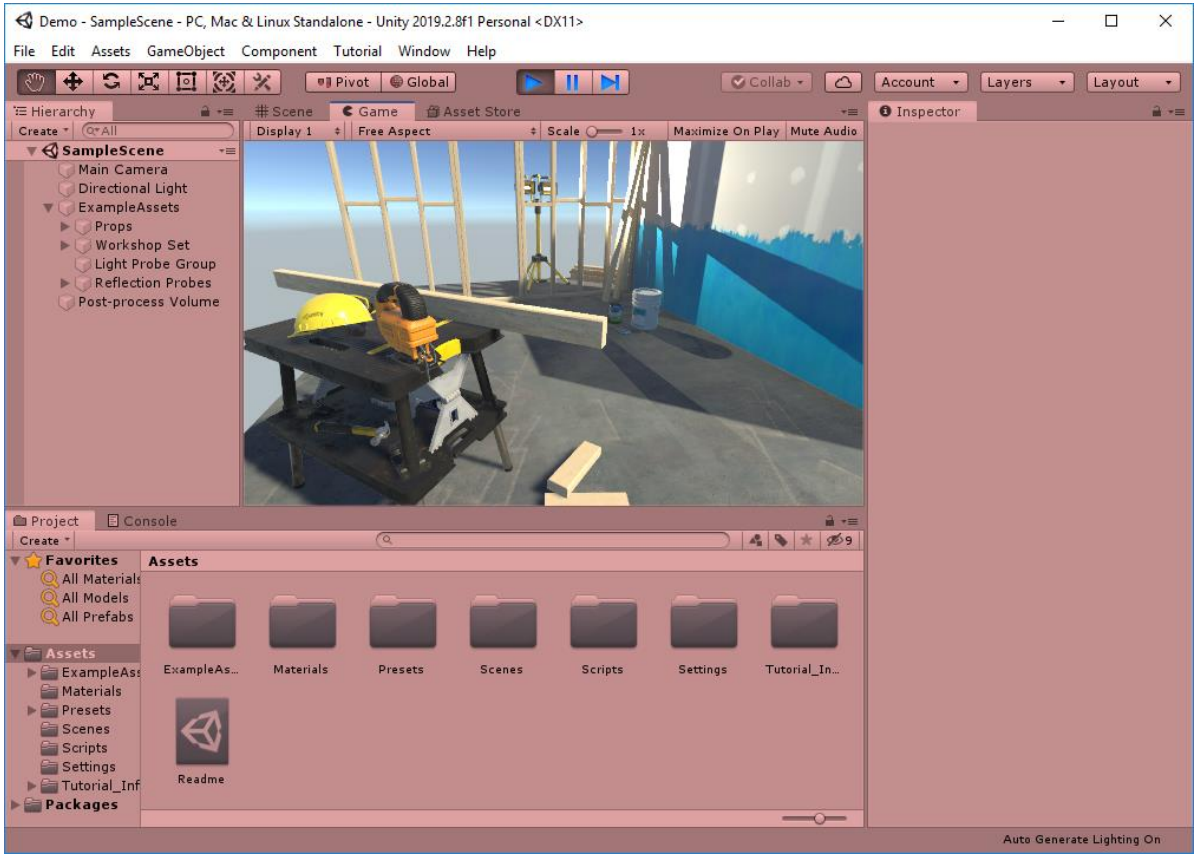
Unity Editor – Edit Mode



Unity Editor – Edit Mode

- Alapértelmezett mód
- A jelenetek összeállítása során ebben a módban kell lennünk
- Ilyenkor lehet hozzáadni/elvenni asset-eket
- A script-ek ilyenkor nem kapnak event-eket
  - Kivéve az Editor Script-ek.
    - Speciális Script, mely Edit módban is fut.

Unity Editor – Play Mode



Unity Editor – Play Mode

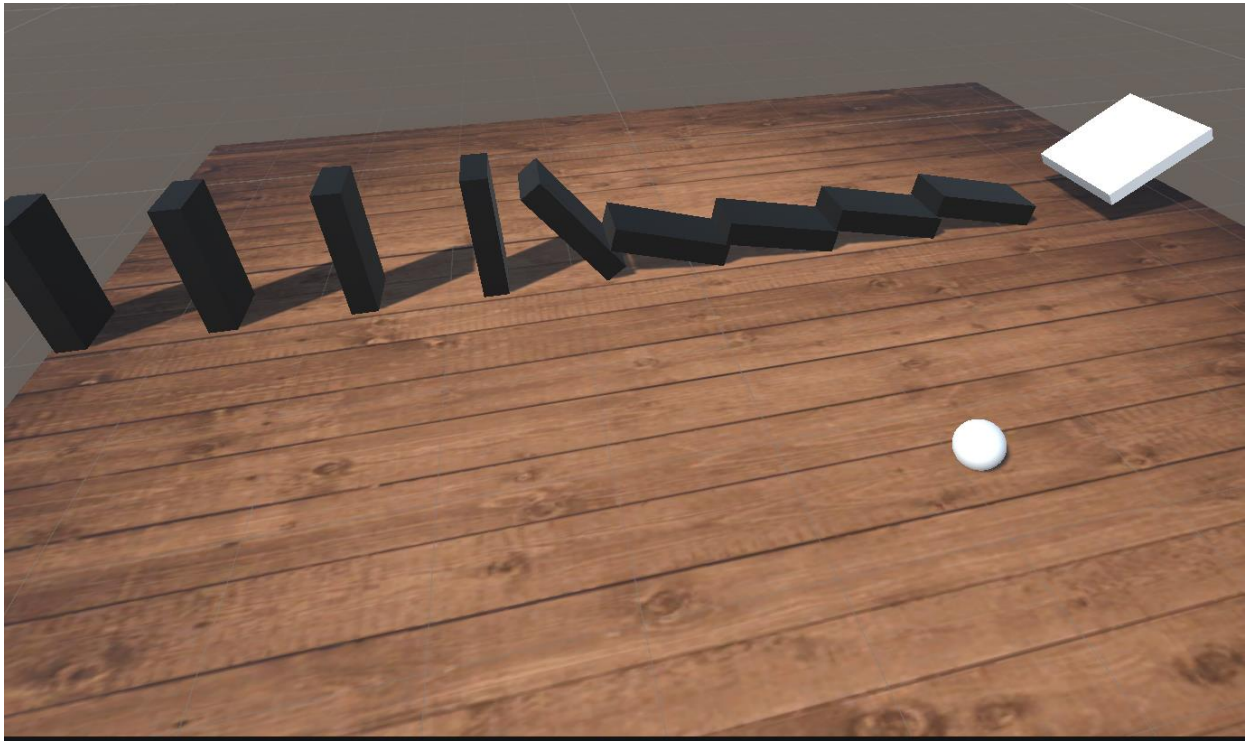
- Debug mód, melyben a script-ek már kapnak Event-eket
- Külön *Game* nézetben renderelődik ki a tartalom
  - Ha ez van fókuszban, itt fogadhatunk inputot
- Menet közben felfüggeszthetjük a futtatást (Pause)
- A Scene nézeben az Edit Mode-hoz hasonlóan lehet navigálni
- Az Inspector és a Hierarchy nézetben az éppen aktuálisan betöltött GameObjectek és paramétereik láthatók

Unity Editor – Play Mode

- Menet közben módosíthatóak az Editorból elérhető property-k
  - **Vigyázat!** Amint, visszatérünk Edit Mode-ba, a változtatásaink elvesznek!
    - Érdemes „élénk” háttérszínt beállítani hozzá, hogy minden esetben látszódjon, hogy mit állítunk.
- Visual Studio-val képesek vagyunk a kódot Play mode-ban debuggolni is
  - A breakpoint-ok ugyanúgy működnek

Unity Editor

- Demo



Köszönöm a figyelmet!