OYUN GELİŞTİRME ARAÇLARI

Yazılım Dilleri, Oyun Motorları

Kaan ESKİCİ

H.Mert DOĞARAY

OYUN PROGRAMLAMA

Dilleri

Programlama dili; oyun programcıları tarafından, bir oyunun ayarlarını ve mekaniğini şekillendirmek için kullanılan sistemin en önemli parçalarından biridir. Bir oyununun belirli bir şekilde çalışmasını sağlayan mekanizmayı temsil ederler.

Bu anlamda, oyunlarda kullanılan her programlama dilinin belirli amaçları vardır ve belirli bir nedenle kullanılırlar.

OYUN PROGRAMLAMA

Dilleri

Performans

Oyunlar genellikle, yüksek kare hızları ve düşük gecikme süreleriyle uygun şekilde çalışmak için yüksek performans gerektirirler. Bu nedenle, iyi bir oyun geliştirme dili verimli ve optimize edilmiş kod üretimine uygun olmalıdır. Ayrıca bazı durumlarda, GPU gibi donanım kaynaklarına Low-Level erişime sahip olmaları da gerekir.

Kullanım Kolaylığı

Oyun geliştirme karmaşık bir iştir, bu nedenle öğrenmesi ve kullanması kolay bir programlama dili, öğrenme eğrisini azaltmaya yardımcı olur. Geliştiricilerin oyun tasarımına ve oyunu geliştirmeye daha çok odaklanmasına imkan tanır.

OYUN PROGRAMLAMA

Dilleri

Kütüphaneler ve Framework

Oyun geliştirme, grafik işleme, fizik simülasyonu ve kullanıcı arayüzü tasarımı gibi işler için kütüphanelere ve framework'lere ihtiyaç vardır. İyi bir oyun geliştirme dili, bu görevleri kolaylaştırmak için zengin bir kütüphane ve Framework setine sahip olmalıdır.

Cross-Platform desteği

Oyunlar genellikle PC, konsollar ve mobil cihazlar gibi birden fazla platform için geliştirilir. İyi bir oyun geliştirme dili, geliştiricilerin birden fazla platformda çalışabilen oyunlar oluşturmasına olanak tanıyan cross-platform desteğine sahip olmalıdır.

OYUN PROGRAMLAMA

Dilleri

Uyumluluk

Bir programlama dilinin diğer diller ve teknolojilerle iyi çalışabilme yeteneği çok değerlidir. Oyun geliştirmede uyumluluk; çeşitli kütüphaneleri, oyun motorlarını ve diğer teknolojileri projeye entegre etmek için önemlidir. Oyun geliştirme için bir programlama dili seçerken bunu göz önünde bulundurmak süreci basitleştirebilir ve çeşitli teknolojilerin projeye entegrasyonuna imkan verebilir.

OYUN PROGRAMLAMA

Dilleri

Kaynaklar

Oyun geliştirme süreci işbirliğine dayalıdır. Proje için seçilen programlama dili, destek sağlayabilecek, bilgi paylaşabilecek, oyun motorlarının ve kütüphanelerinin geliştirilmesine katkıda bulunabilecek güçlü bir geliştirici topluluğuna sahip olmalıdır.

Oyun geliştirme için en iyi dil, nihayetinde oyunun özel gereksinimlerine ve geliştirme ekibinin tercihlerine bağlıdır.

OYUN PROGRAMLAMA
Dilleri

High-Level Programlama Dilleri

-High-Level(Scripting) programlama dilleri; programcıların bir algoritmayı program koduna aktarmasını kolaylaştırmak için tasarlanmış, kullanıcı odaklı dillerdir.

-Kolay yönetilebilir, anlaşılır, hata ayıklaması daha rahat ve günümüzde yaygın olarak kullanılan programcı dostu dillerdir.

-Bu dilleri, makine koduna çevirmek için bir compiler ya da interpreter kullanmak gerekir.

OYUN PROGRAMLAMA
Dilleri

High-Level Programlama Dilleri

-Bu diller, Bellek yönetimi konusunda çok verimli değillerdir. Bu durum genellikle, herhangi bir Low-Level dilden daha fazla bellek tükettikleri anlamına gelir.

-Makinelere(bileşenlere) bağlı değillerdir.

-Basit ve kapsamlı bir bakım(maintenance) olanağı sunarlar.

-Günümüzde programlama için yaygın olarak tercih edilmektedirler.

OYUN PROGRAMLAMA

Dilleri

High-Level Programlama Dilleri

-Low-Level dillere kıyasla uygulanması daha fazla zaman gerektirir çünkü ilave bir çeviri programına(compiler ya da interpreter) ihtiyaç duyulur.

-Genellikle donanım seviyesinde yapilacak isler icin uygun bir seçim olarak görülmezler.

OYUN PROGRAMLAMA

Dilleri

Low-Level Programlama Dilleri

-Low-Level programlama dilleri makine odaklıdır. Genel olarak donanım seviyesinde işlemler yapmak için kullanılırlar.

-Uygulama anlamında High-Level dillere göre daha zor bir yaklaşıma sahiptirler. Syntax(yazım kuralları) konusunda daha karmaşık bir düzenle ilerledikleri için kullanıcı tarafından anlaşılması ve uygulanması daha zordur.

OYUN PROGRAMLAMA
Dilleri

Low-Level Programlama Dilleri

-Low-Level diller çok yüksek bellek verimliliğine(memory management) sahiptir. Bu, herhangi bir yüksek seviyeli dile kıyasla daha az enerji tükettikleri anlamına gelir.

-Hata ayıklama(debug), basit ve anlaşılır bakım imkanı(maintenance) gibi konularda kullanıcı daha fazla zorlanır.

-Low-Level dillerin çeviri hızı(uygulama hızı) çok yüksektir.

OYUN PROGRAMLAMA

Dilleri

Oyun üretiminde tercih edilen
Programlama dilleri

-C++
-C#
-Lua
-Java
-JavaScript
-Python
-Objective-C

Programlama Dili	Artıları	Eksileri	Oyun Motoru
C++	-Yüksek performans ve donanım kaynaklarına düşük seviyeli erişim -Oyun motorları, kütüphaneler ve geliştirme topluluklarından güçlü destek -Cross-Platform desteği -Oyun endüstrisinde yaygın kullanım	-Öğrenmesi ve ustalaşması zor -Bellek yönetimi zorlayıcı olabiliyor. Memory Leak ve Buffer Overflow gibi potansiyel sorunlara yol açabilir -Kod ayrıntılı ve okunması zor olabilir	-Unreal Engine -Unity -CryEngine -Godot Engine -GameMaker Studio
C#	-Öğrenmesi ve kullanması kolay -İyi performans -NET Framework ile güçlü entegrasyon -Oyun endüstrisinde yaygın kullanım -.NET Core ve Unity aracılığıyla Cross-Platform desteği	-Donanım kaynaklarına sınırlı ve düşük seviye erişim -C++ ile karşılaştırıldığında oyun motorlarından ve kütüphanelerden sınırlı destek -Microsoft Visual Studio(IDE) gibi ek geliştirme araçları ve yazılımları gerektirebilir	-Unity -CryEngine -Godot Engine -MonoGame -Banshee 3D
Java	-Öğrenmesi ve kullanması kolay -İyi performans -Oyun motorları, kütüphaneler ve geliştirme topluluklarından güçlü destek -Android oyun geliştirme için ideal -Java Sanal Makinesi (JVM) aracılığıyla Cross-Platform desteği	-Donanım kaynaklarına sınırlı ve düşük seviye erişim -Garbage Collection işlemi öngörülemeyen performans sorunlarına yol açabilir -C++ ve C# ile karşılaştırıldığında oyun endüstrisinde yaygın olarak kullanılmaz	-jMonkeyEngine -LibGDX -LWJGL -Shark 3D -Solar2D

Programlama Dili	Artıları	Eksileri	Oyun Motoru
Python	-Öğrenmesi ve kullanması kolay -Daha kısa geliştirme süresi -Prototip oluşturma ve hızlı geliştirme için iyi -Oyun geliştirme için çok sayıda kütüphane mevcuttur -Cross-Platform desteği mevcut	-C++ ve C#'a kıyasla daha düşük performans -Bellek yönetimi zorlayıcı olabiliyor. Memory Leak gibi potansiyel sorunlara yol açabilir -Donanım kontrolü yok -C++ ve C# ile karşılaştırıldığında oyun endüstrisinde yaygın olarak kullanılmaz	-Pygame -Panda3D -PyOgre -Ren'Py (Görsel Roman türü için)
JavaScript	-Özellikle web tabanlı oyunlar için öğrenmesi ve kullanması kolay -Oyun geliştirme için Phaser ve Three.js gibi çok sayıda kütüphane ve Framework mevcut -Web tarayıcıları aracılığıyla Cross-Platform desteği -Prototip oluşturma ve hızlı geliştirme için iyi -Geniş bir geliştirici topluluğu ve buna bağlı kaynaklar mevcut	-C++ ve C# gibi Local dillere kıyasla daha düşük performans -Donanım kaynaklarına sınırlı ve düşük seviye erişim -Dilin dinamik ve açık yapısı nedeniyle potansiyel güvenlik açıkları -Oyun motorlarından sınırlı destek	-Phaser -Three.js -Babylon.js -Construct 3
Lua	-Lightweight bir yapısı vardır ve oyun motorlarına yerleştirilmesi kolaydır -Komut dosyası oluşturma ve Game Logic için iyi -Çok sayıda oyun motoru ve kütüphanesi Lua'yı desteklemektedir -Cross-Platform desteği	-Donanım kaynaklarına sınırlı ve düşük seviye erişim -Büyük ve karmaşık oyunlar geliştirmek için uygun değil -Diğer diller ve teknolojilerle entegrasyon için ek geliştirme araçları ve yazılımları gerektirebilir	-Corona SDK -Love2D -Gideros Mobile -Defold -Agen

OYUN
Motorları

Unreal Engine

Unity

Godot

Cry Engine

GDevelop

Game Maker

Construct 3

In-House Motorlar (Nintendo, Taleworlds vb. Şirket içi çözümler)

Your First Game in UE5 | Tech Talk | State of Unreal 2022

Unreal Engine
1,07 Mn abone
Abone ol

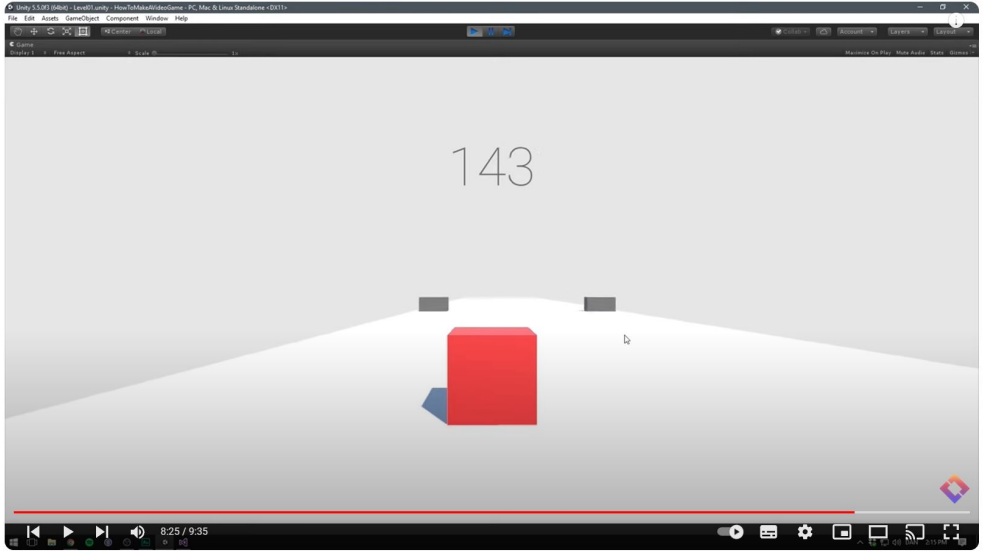
3,7 B | Paylaş | İndir | Klip | Kaydet

212 B görüntüleme 1 yıl önce #UnrealEngine5 #UE5 #VideoGames
Watch our tech talk 'Your First Game in UE5' from the State of Unreal 2022 livestream.

In this tech talk, we provide a practical in-editor demonstration of how to get started making a game from scratch, highlighting UE5 features along the way. We show how fast you can create ...daha fazla

UNREAL ENGINE 5

UNITY



FINISHING UP - How to make a Video Game in Unity (E10)

Brackeys
1,67 Mn abone
Abone ol

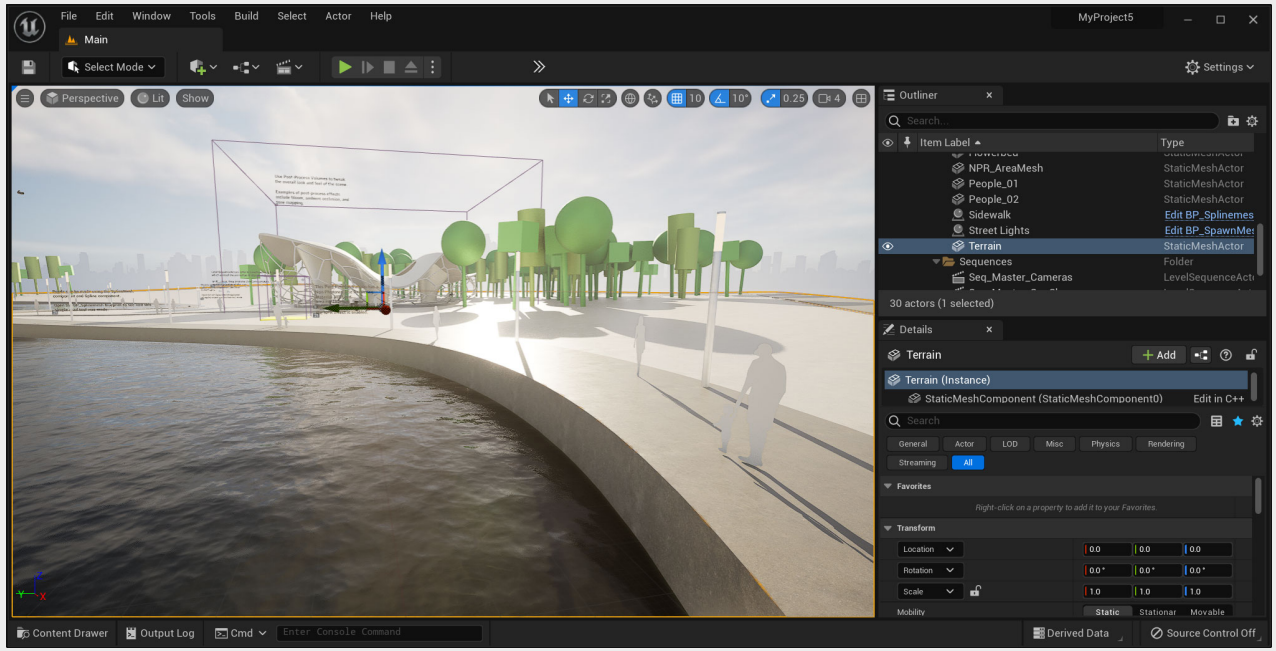
27 B | Paylaş | İndir | Klip | Kaydet

989 B görüntüleme 6 yıl önce How to make a Video Game
Let's wrap things up!

• Download the Project Files: http://devassets.com/assets/how-to-ma... ...daha fazla

UNREAL

Engine



Unreal Engine, Epic Games tarafından geliştirilen ve ilk olarak 1998 yılında yayınlanan Unreal isimli birinci şahıs nişancı oyununda kullanılan oyun motorudur. Esas olarak birinci şahıs nişancı oyunları için geliştirilmesine karşın, sonraları çeşitli türlerdeki oyunlarda kullanıldı.

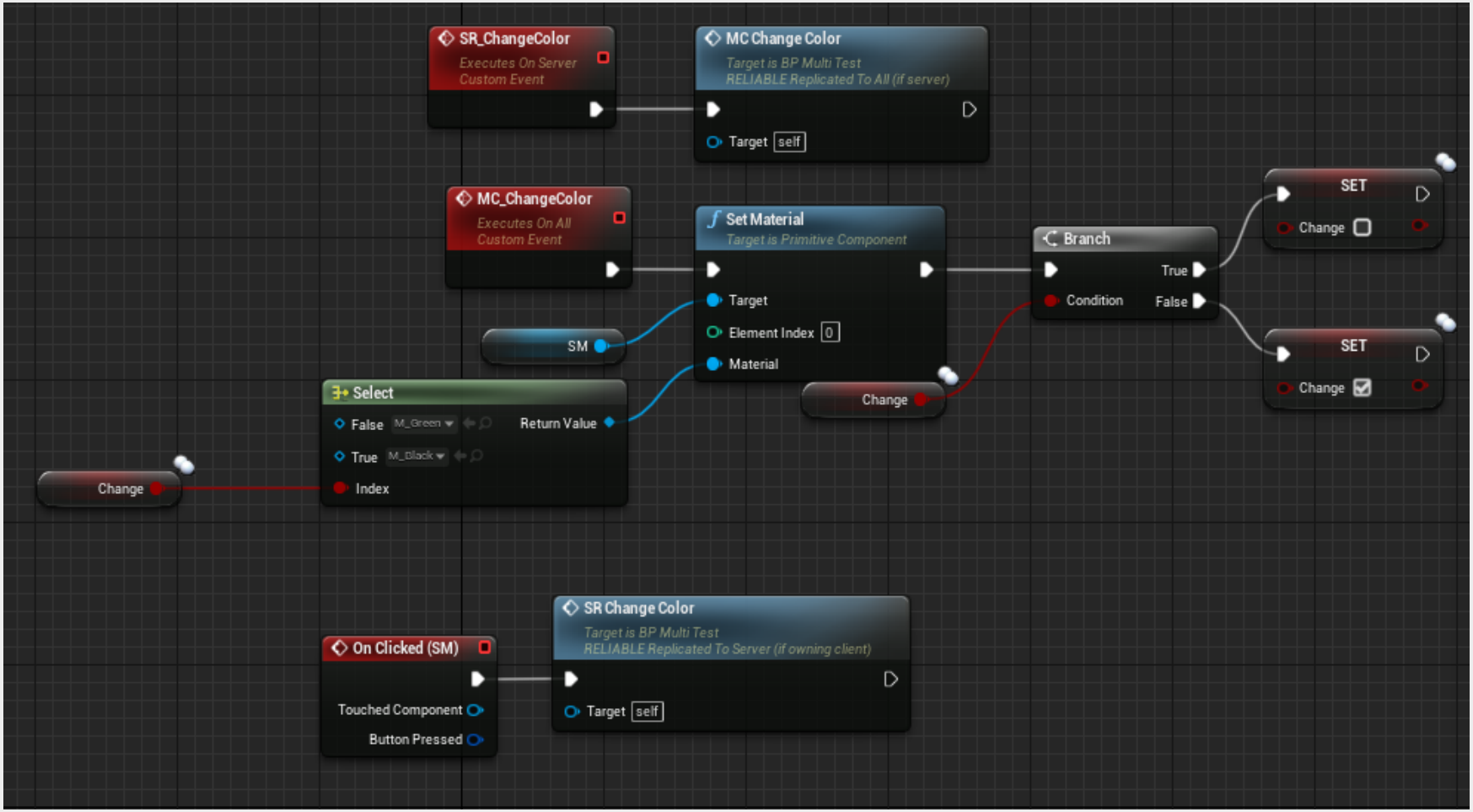
Unreal Engine 1, Quake oyun motorununun döneminde ana rakibiydi.

Unreal Engine 2, Çıkışını America’s Army ile yaptı.

Unreal Engine 3, ile geliştirlen BioShock....

Unreal Engine 3, ile geliştirilen Gears of War 2....

Unreal Engine 4’ün ses getiren oyunlarından Fortnite...

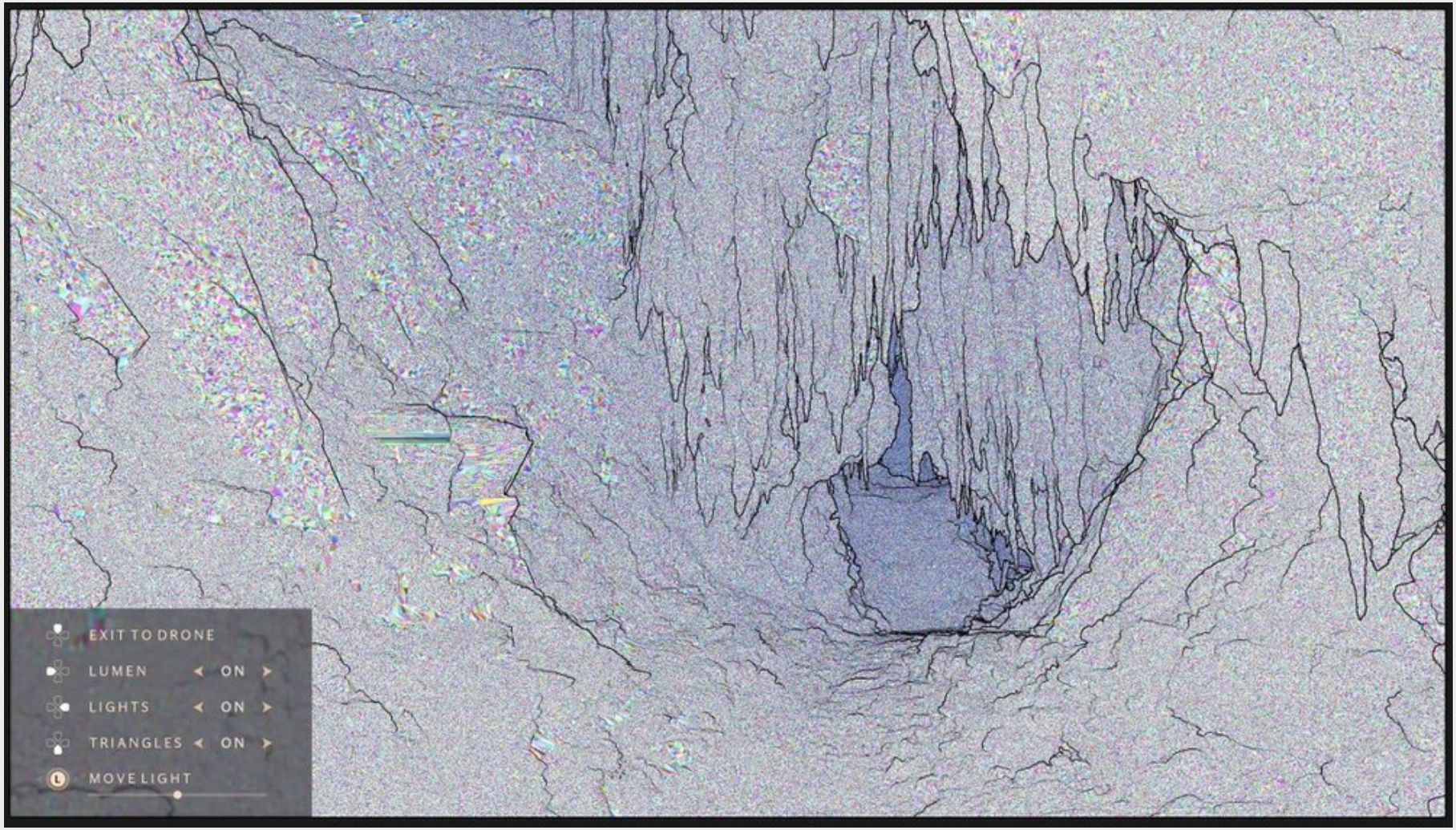


Unreal Engine 4 ile birlikte Blueprints sistemi tanıtıldı.

Unreal Engine 5 tech demo sahnesi.

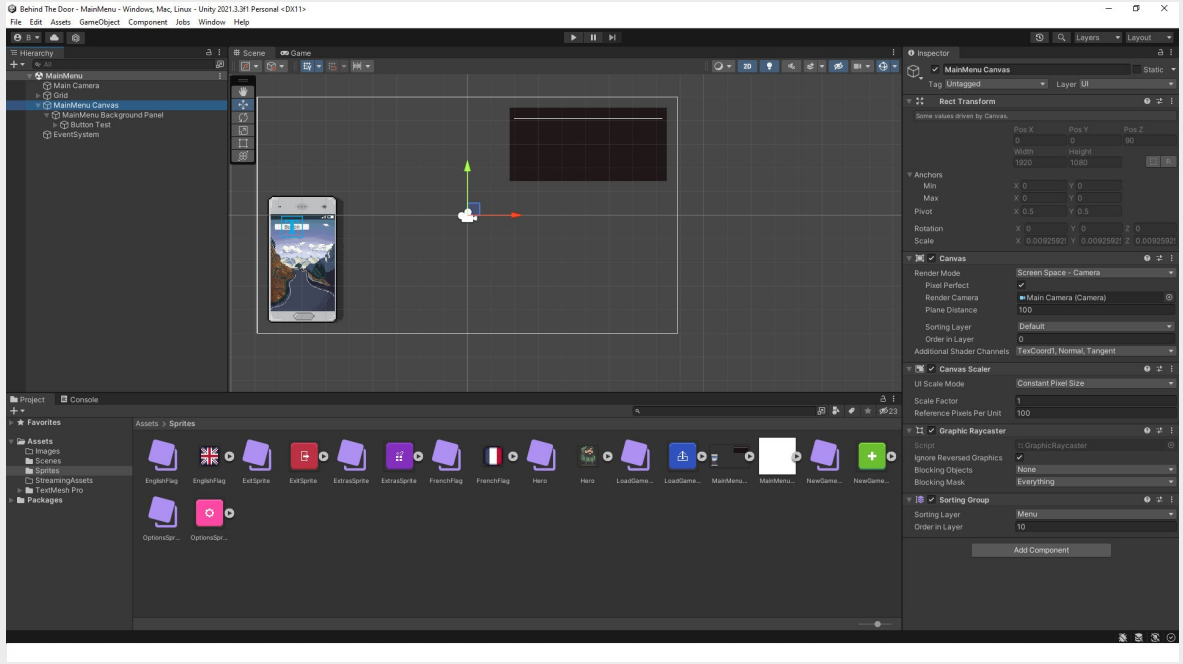
Unreal Engine 5 tech demo sahnesi.

Unreal Engine 5 lumen teknolojisi...



Unreal Engine 5 nanite teknolojisi...

UNITY



Unity, Unity Technologies tarafından geliştirilen cross-platform bir oyun motorudur. İlk kez Apple'ın 2005'teki Worldwide Developers Conference'da OS X için ilan edildi, bu tarihten sonra ise daha fazla platformu destekleyerek genişletildi. Unity oyun motoru; film sektörü, otomotiv sektörü, mimari, mühendislik ve inşaat gibi video oyunları dışındaki farklı endüstriler tarafından da benimsenmiştir ve kullanılmaktadır.

Unity, mobil pazarda sıklıkla Hyper-Casual geliştiriciler tarafından kullanılıyor.

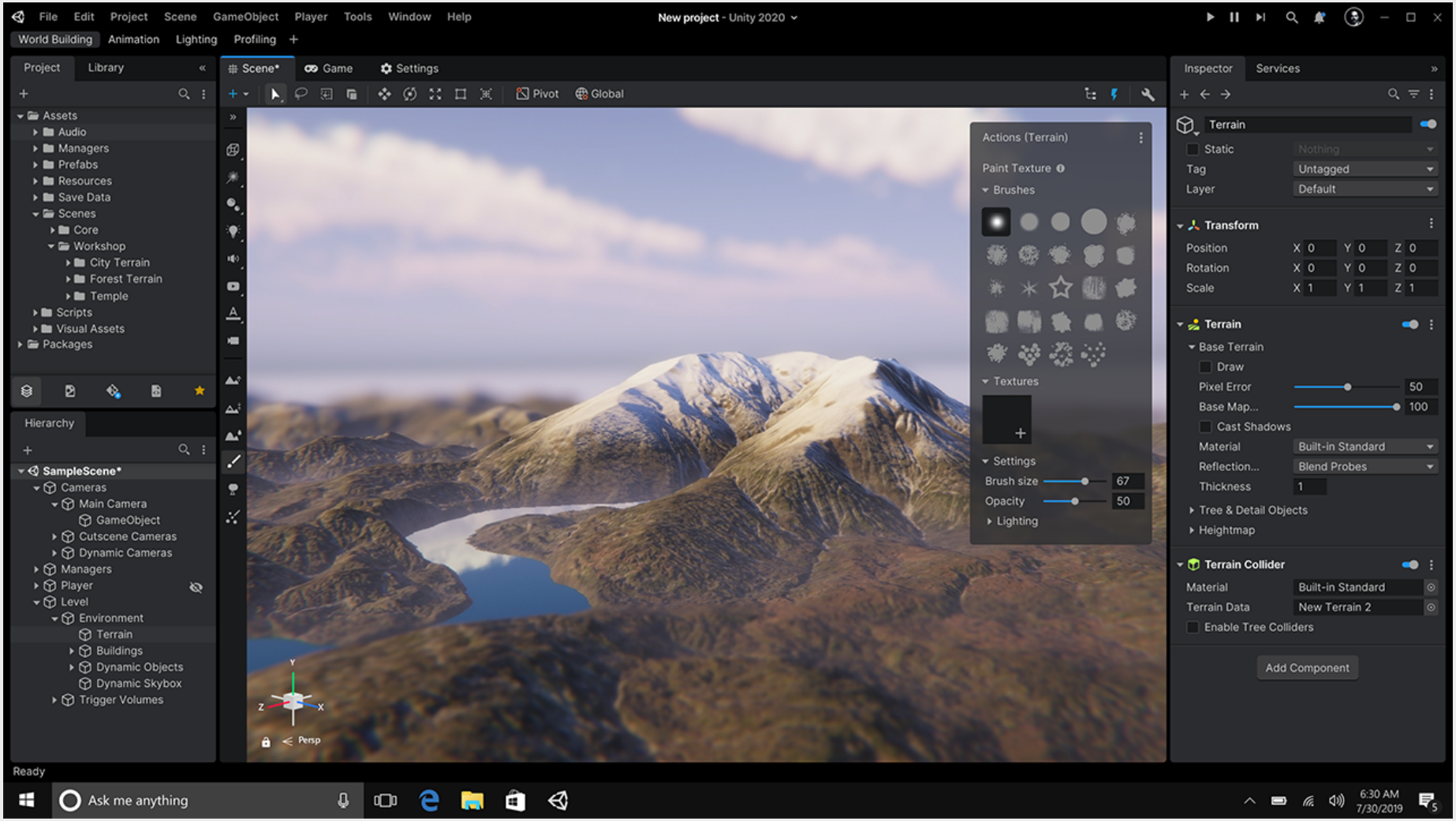
Unity, AA stüdyolar tarafından da tercih ediliyor : bir örneği FireWatch oyunu..

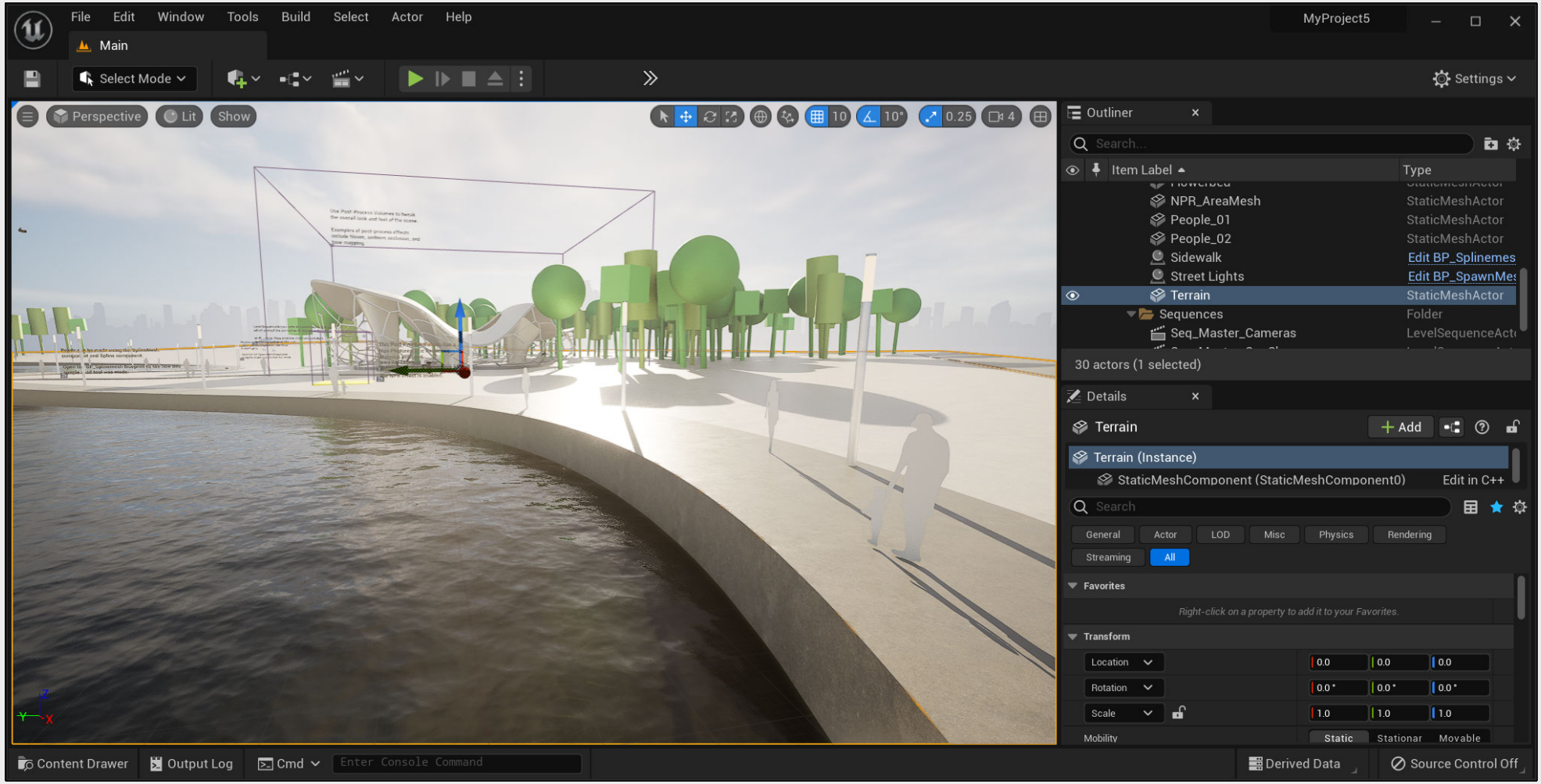
Unity ile geliştirilen The Long Dark...

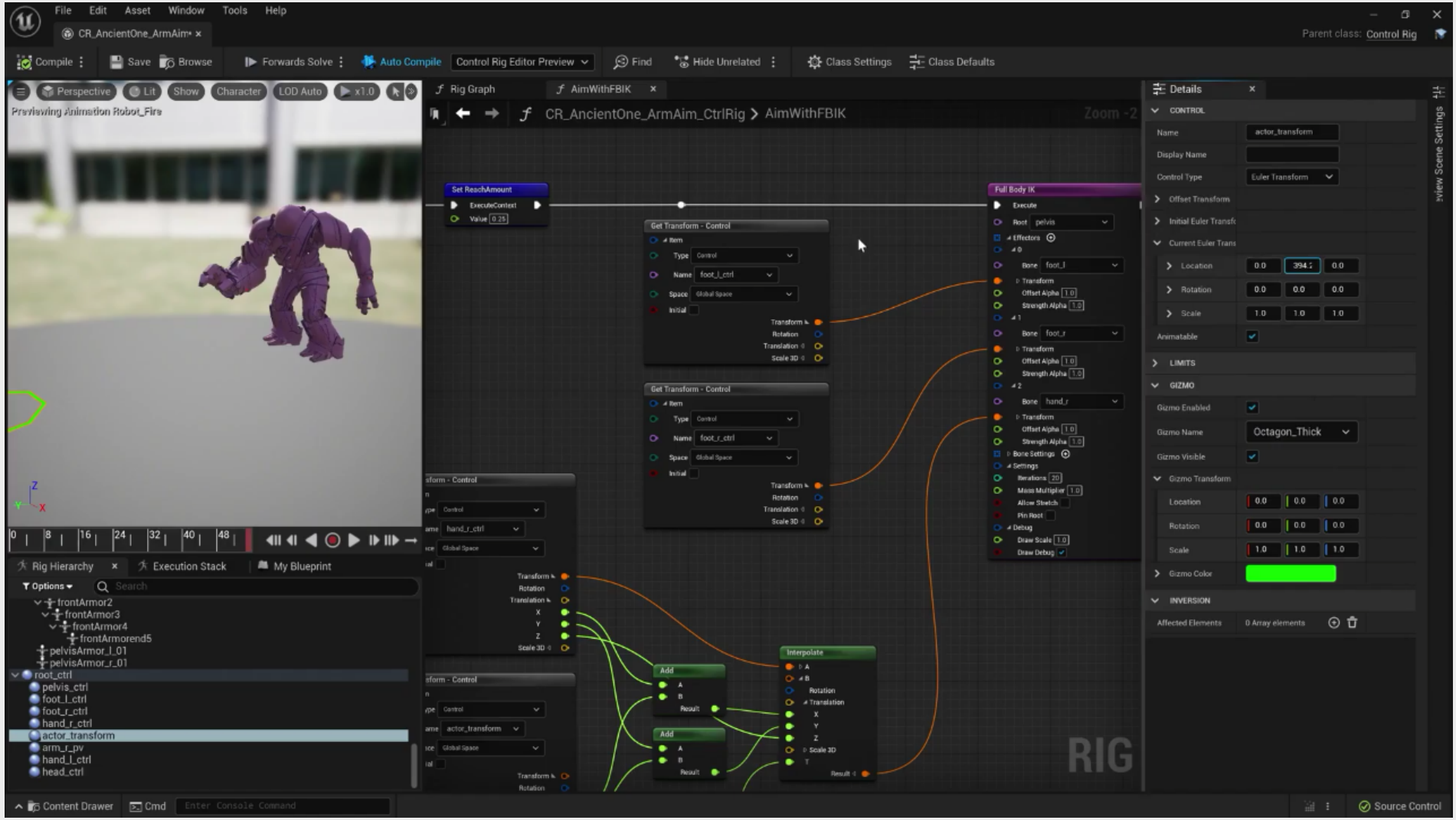

UNREAL ENGINE 5
TECH TALK
Your First Game in UE5

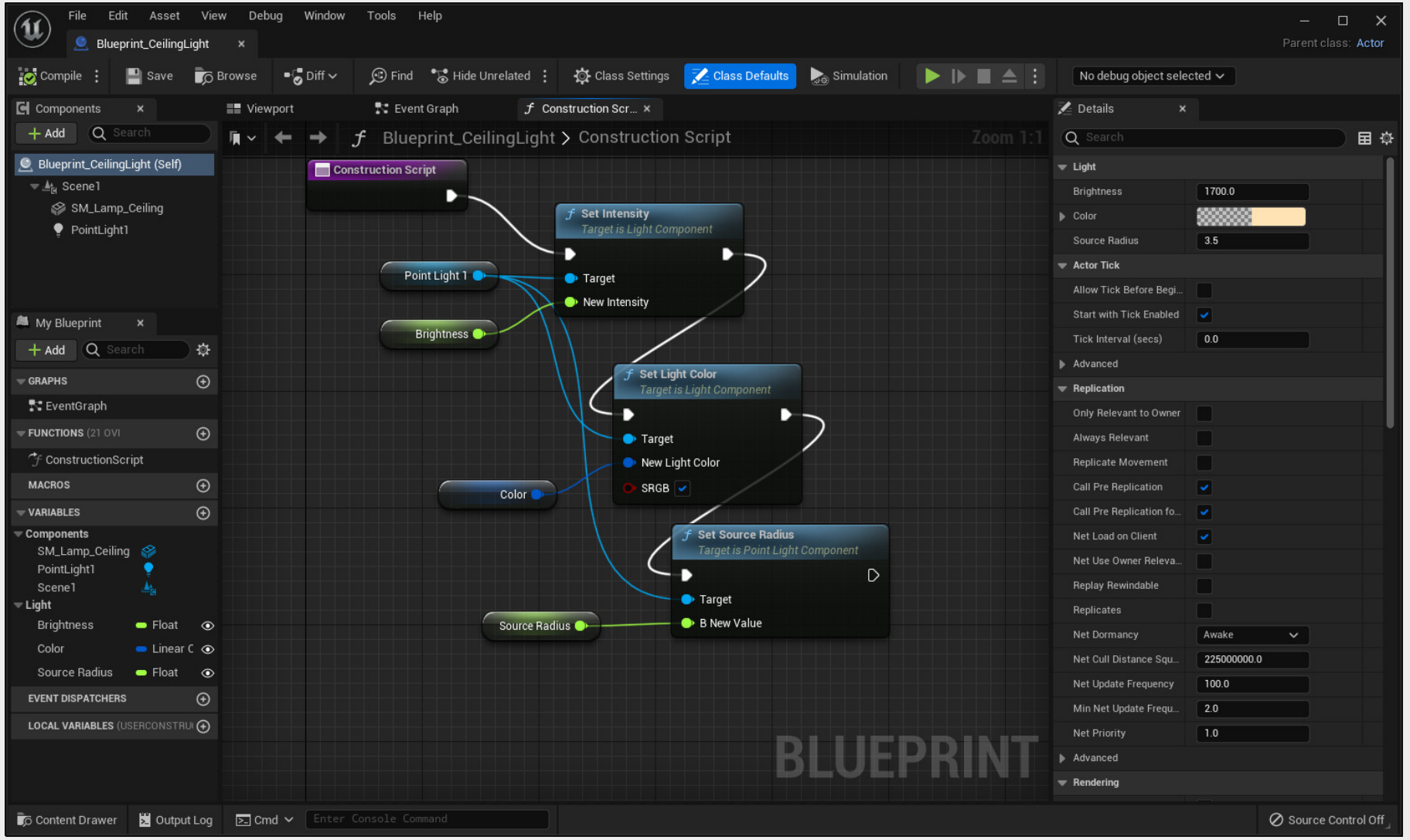
ARAYÜZ

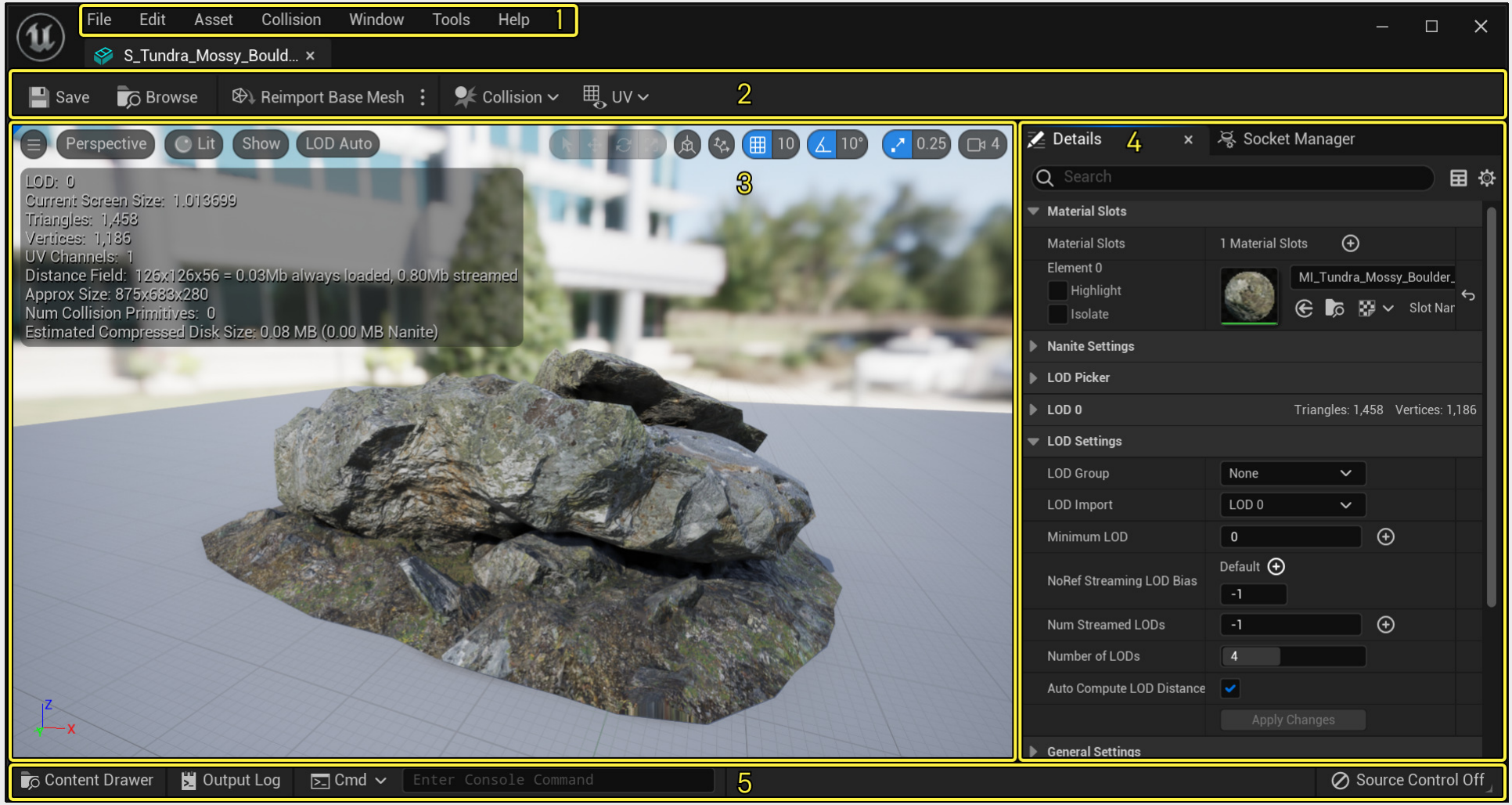
Karşılaştırması


File Edit Project Scene GameObject Player Tools Window Help
New project - Unity 2020
World Building Animation Lighting Profiling
Project Library
Assets
Audio
Managers
Prefabs
Resources
Save Data
Scenes
Core
Workshop
City Terrain
Forest Terrain
Temple
Scripts
Visual Assets
Packages
Hierarchy
SampleScene*
Cameras
Main Camera
GameObject
Cutscene Cameras
Dynamic Cameras
Managers
Player
Level
Environment
Terrain
Buildings
Dynamic Objects
Dynamic Skybox
Trigger Volumes
Scene* Game Settings
Pivot Global
Actions (Terrain)
Paint Texture
Brushes
Textures
Settings
Brush size 67
Opacity 50
Lighting
Persp
Inspector Services
Terrain
Static Nothing
Tag Untagged
Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Terrain
Base Terrain
Draw
Pixel Error 50
Base Map... 100
Cast Shadows
Material Built-in Standard
Reflection... Blend Probes
Thickness 1
Tree & Detail Objects
Heightmap
Terrain Collider
Material Built-in Standard
Terrain Data New Terrain 2
Enable Tree Colliders
Add Component
Ready
Ask me anything
6:30 AM
7/30/2019


File
Edit
Window
Tools
Build
Select
Actor
Help
MyProject5
Main
Select Mode
Settings
Perspective
Lit
Show
Outliner
Search...
Item Label
Type
NPR_AreaMesh
StaticMeshActor
People_01
StaticMeshActor
People_02
StaticMeshActor
Sidewalk
Edit BP_Splinemes
Street Lights
Edit BP_SpawnMes
Terrain
StaticMeshActor
Sequences
Folder
Seq_Master_Cameras
30 actors (1 selected)
Details
Terrain
Add
Terrain (Instance)
StaticMeshComponent (StaticMeshComponent0)
Edit in C++
Search
General
Actor
LOD
Misc
Physics
Rendering
Streaming
All
Favorites
Right-click on a property to add it to your Favorites.
Transform
Location
Rotation
Scale
Mobility
Static
Stationar
Movable
Content Drawer
Output Log
Cmd
Enter Console Command
Derived Data
Source Control Off

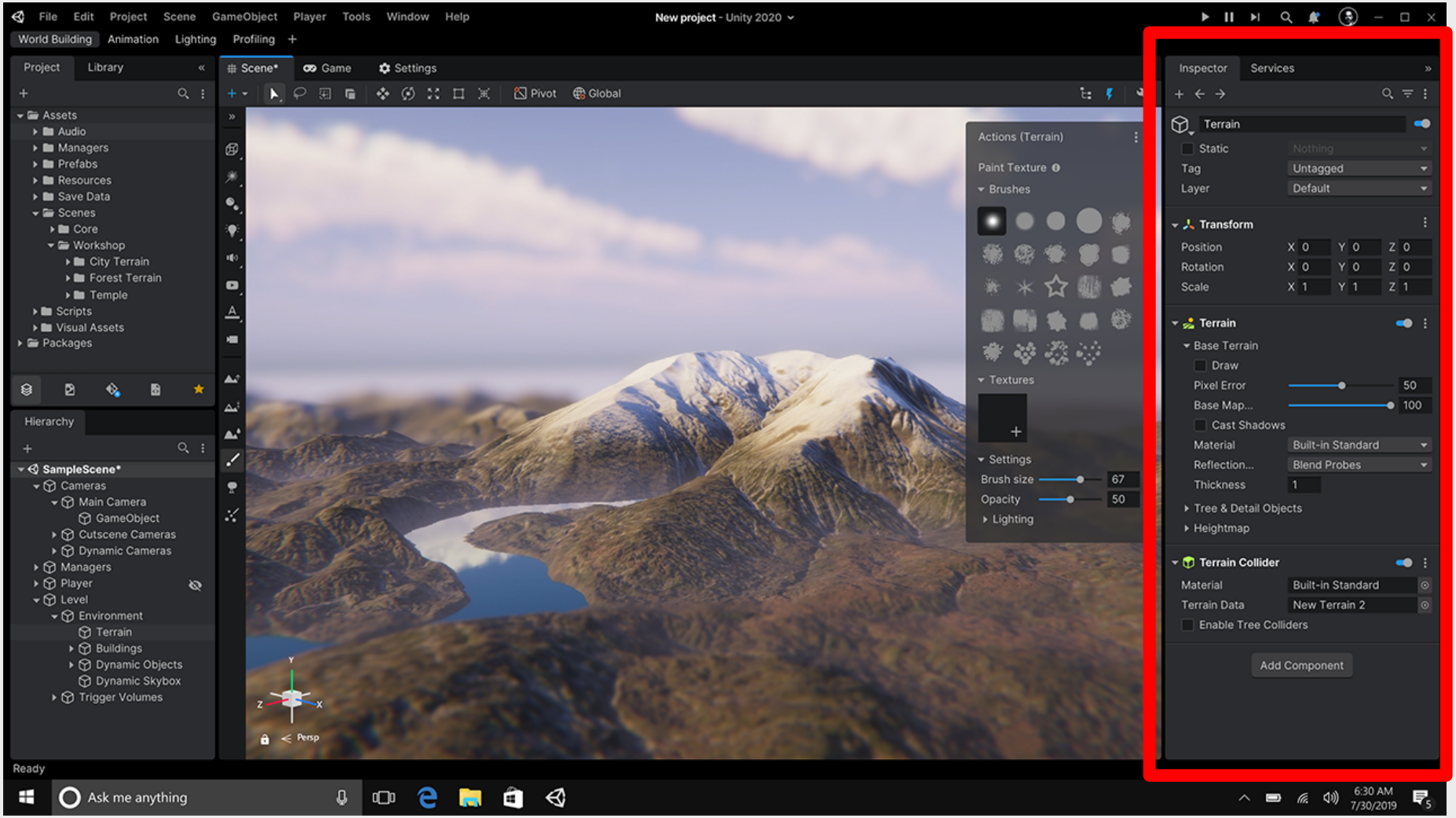

File Edit Asset Window Tools Help
CR_AncientOne_ArmAim*
Parent class: Control Rig
Compile
Save
Browse
Forwards Solve
Auto Compile
Control Rig Editor Preview
Find
Hide Unrelated
Class Settings
Class Defaults
Perspective
Lit
Show
Character
LOD Auto
x1.0
Previewing Animation Robot_Fire
Rig Graph
AimWithFBIK
CR_AncientOne_ArmAim_CtrlRig > AimWithFBIK
Set ReachAmount
ExecuteContext
Value 0.25
Get Transform - Control
Full Body IK
Execute
Root pelvis
Effectors
Bone foot_l
Bone foot_r
Bone hand_r
Add
Interpolate
Rig Hierarchy
Execution Stack
My Blueprint
Options
Search
frontArmor2
frontArmor3
frontArmor4
frontArmorend5
pelvisArmor_l_01
pelvisArmor_r_01
root_ctrl
pelvis_ctrl
foot_l_ctrl
foot_r_ctrl
hand_r_ctrl
actor_transform
arm_r_pv
hand_l_ctrl
head_ctrl
Details
CONTROL
Name
actor_transform
Display Name
Control Type
Euler Transform
Offset Transform
Initial Euler Transf
Current Euler Trans
Location
Rotation
Scale
Animatable
LIMITS
GIZMO
Gizmo Enabled
Gizmo Name
Octagon_Thick
Gizmo Visible
Gizmo Transform
Gizmo Color
INVERSION
Affected Elements
0 Array elements
RIG
Content Drawer
Cmd
Enter Console Command
Source Control

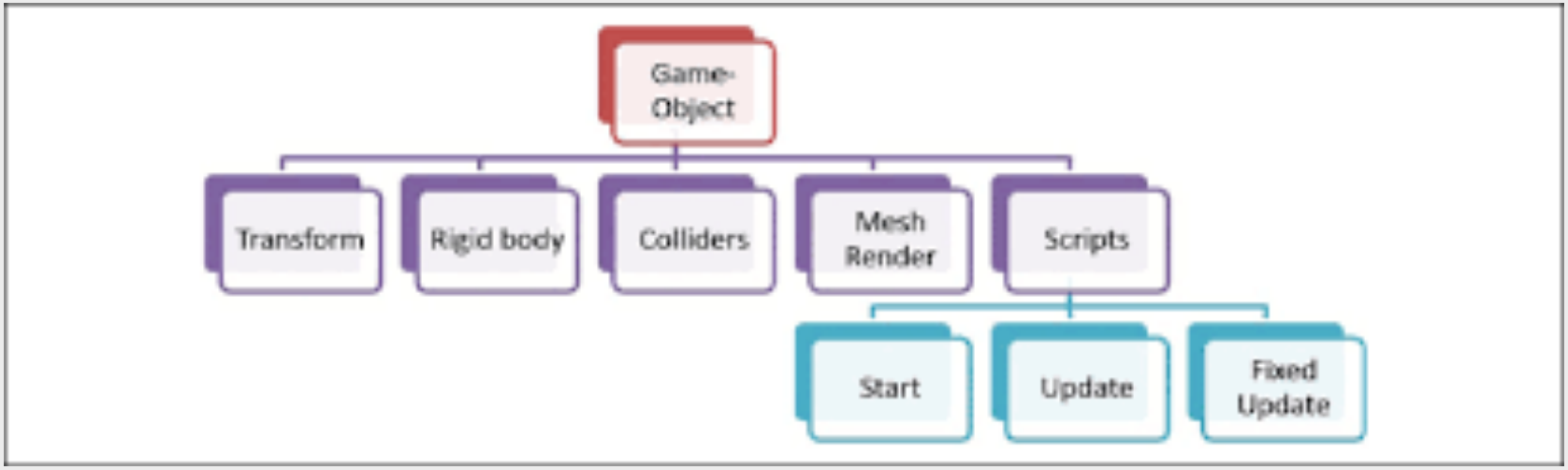

File Edit Asset View Debug Window Tools Help
Blueprint_CeilingLight
Parent class: Actor
Compile Save Browse Diff Find Hide Unrelated Class Settings Class Defaults Simulation
No debug object selected
Components
Add Search
Blueprint_CeilingLight (Self)
Scene1
SM_Lamp_Ceiling
PointLight1
Viewport
Event Graph
Construction Scr...
Blueprint_CeilingLight > Construction Script
Zoom 1:1
Construction Script
Set Intensity
Target is Light Component
Point Light 1
Target
New Intensity
Brightness
Set Light Color
Target is Light Component
Target
New Light Color
SRGB
Color
Set Source Radius
Target is Point Light Component
Target
B New Value
Source Radius
BLUEPRINT
Details
Search
Light
Brightness 1700.0
Color
Source Radius 3.5
Actor Tick
Allow Tick Before Begi...
Start with Tick Enabled
Tick Interval (secs) 0.0
Advanced
Replication
Only Relevant to Owner
Always Relevant
Replicate Movement
Call Pre Replication
Call Pre Replication fo...
Net Load on Client
Net Use Owner Releva...
Replay Rewindable
Replicates
Net Dormancy Awake
Net Cull Distance Squ... 225000000.0
Net Update Frequency 100.0
Min Net Update Frequ... 2.0
Net Priority 1.0
Advanced
Rendering
My Blueprint
Add Search
GRAPHS
EventGraph
FUNCTIONS (21 OVI
ConstructionScript
MACROS
VARIABLES
Components
SM_Lamp_Ceiling
PointLight1
Scene1
Light
Brightness Float
Color Linear C
Source Radius Float
EVENT DISPATCHERS
LOCAL VARIABLES (USERCONSTRU
Content Drawer
Output Log
Cmd
Enter Console Command
Source Control Off


File
Edit
Asset
Collision
Window
Tools
Help
1
S_Tundra_Mossy_Bould...
Save
Browse
Reimport Base Mesh
Collision
UV
2
Perspective
Lit
Show
LOD Auto
10
10°
0.25
4
3
LOD: 0
Current Screen Size: 1.013699
Triangles: 1,458
Vertices: 1,186
UV Channels: 1
Distance Field: 126x126x56 = 0.03Mb always loaded, 0.80Mb streamed
Approx Size: 875x683x280
Num Collision Primitives: 0
Estimated Compressed Disk Size: 0.08 MB (0.00 MB Nanite)
Z
X
Details
4
Socket Manager
Search
Material Slots
Material Slots
1 Material Slots
Element 0
Highlight
Isolate
MI_Tundra_Mossy_Boulder_
Slot Nar
Nanite Settings
LOD Picker
LOD 0
Triangles: 1,458
Vertices: 1,186
LOD Settings
LOD Group
None
LOD Import
LOD 0
Minimum LOD
0
Default
NoRef Streaming LOD Bias
-1
Num Streamed LODs
-1
Number of LODs
4
Auto Compute LOD Distance
Apply Changes
General Settings
Content Drawer
Output Log
Cmd
Enter Console Command
5
Source Control Off

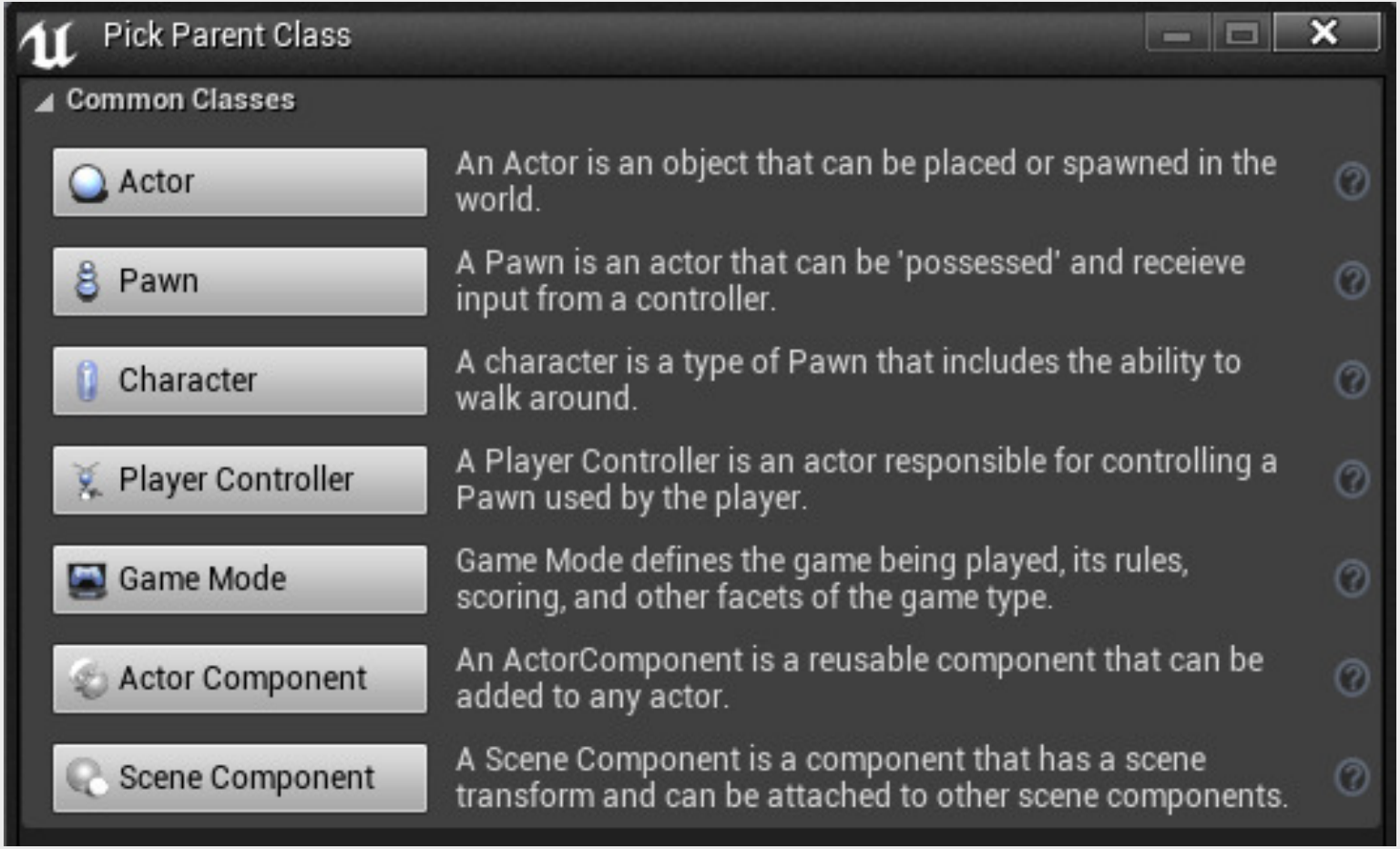
YAKLAŞIM

Farkları

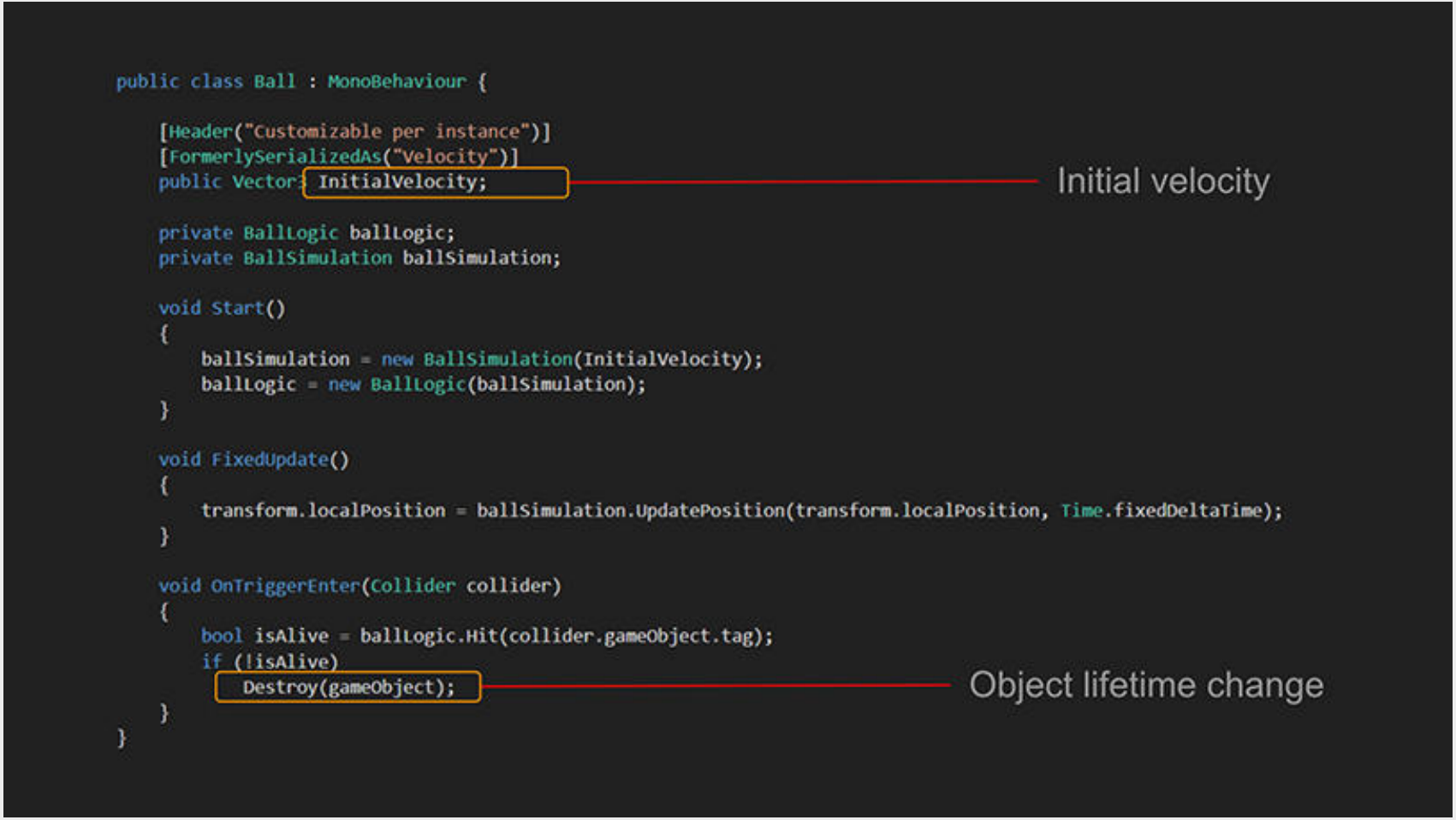

File Edit Project Scene GameObject Player Tools Window Help
New project - Unity 2020
World Building Animation Lighting Profiling
Project Library
Assets
Audio
Managers
Prefabs
Resources
Save Data
Scenes
Core
Workshop
City Terrain
Forest Terrain
Temple
Scripts
Visual Assets
Packages
Hierarchy
SampleScene*
Cameras
Main Camera
GameObject
Cutscene Cameras
Dynamic Cameras
Managers
Player
Level
Environment
Terrain
Buildings
Dynamic Objects
Dynamic Skybox
Trigger Volumes
Scene*
Game
Settings
Pivot
Global
Persp
Actions (Terrain)
Paint Texture
Brushes
Textures
Settings
Brush size 67
Opacity 50
Lighting
Inspector
Services
Terrain
Static
Nothing
Tag
Untagged
Layer
Default
Transform
Position
Rotation
Scale
Terrain
Base Terrain
Draw
Pixel Error 50
Base Map... 100
Cast Shadows
Material
Built-in Standard
Reflection...
Blend Probes
Thickness 1
Tree & Detail Objects
Heightmap
Terrain Collider
Material
Built-in Standard
Terrain Data
New Terrain 2
Enable Tree Colliders
Add Component
Ready
Ask me anything
6:30 AM
7/30/2019



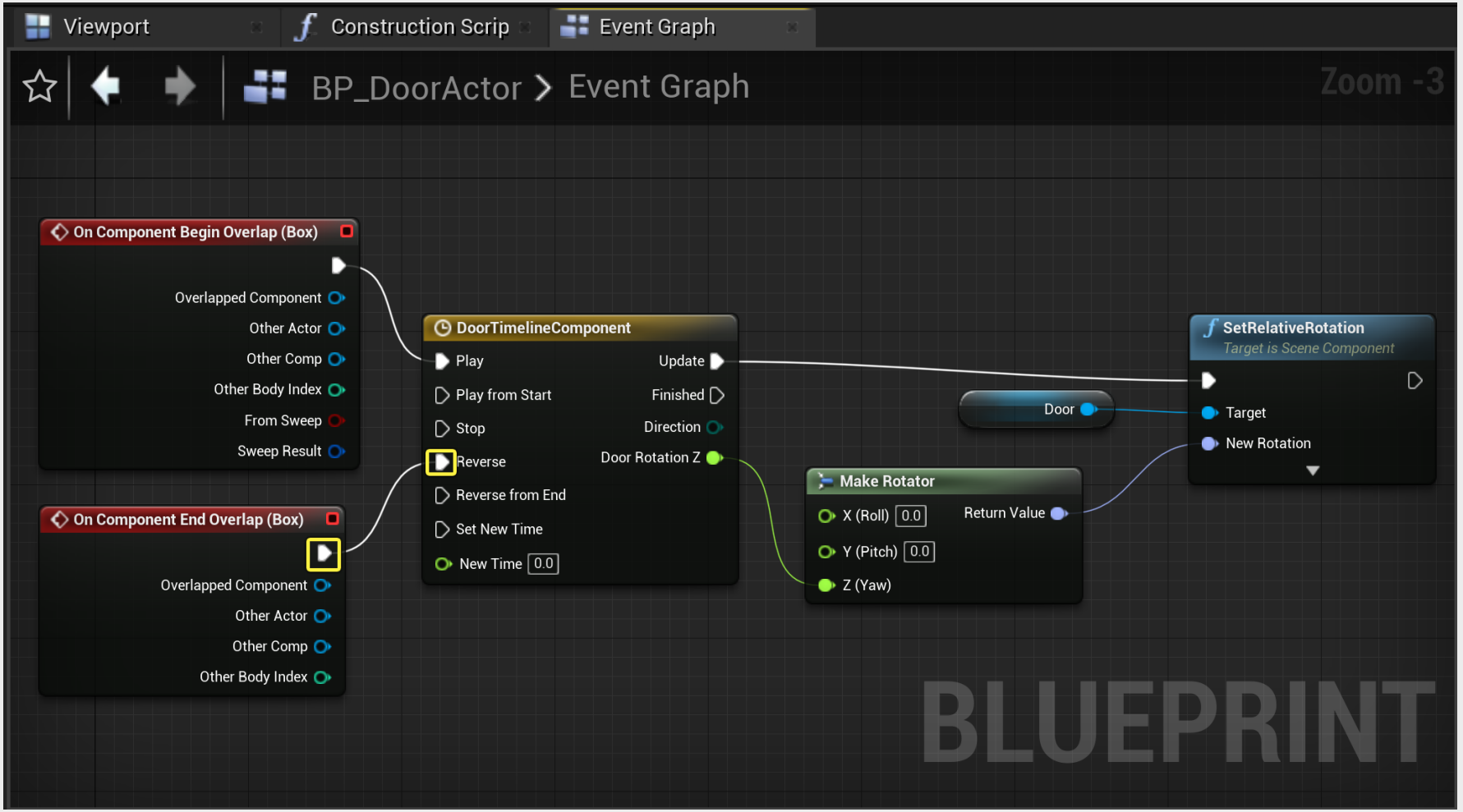
Unity’deki GameObject, Components ve MonoBehaviour mantığı



Unreal Engine’de blueprints class’ları



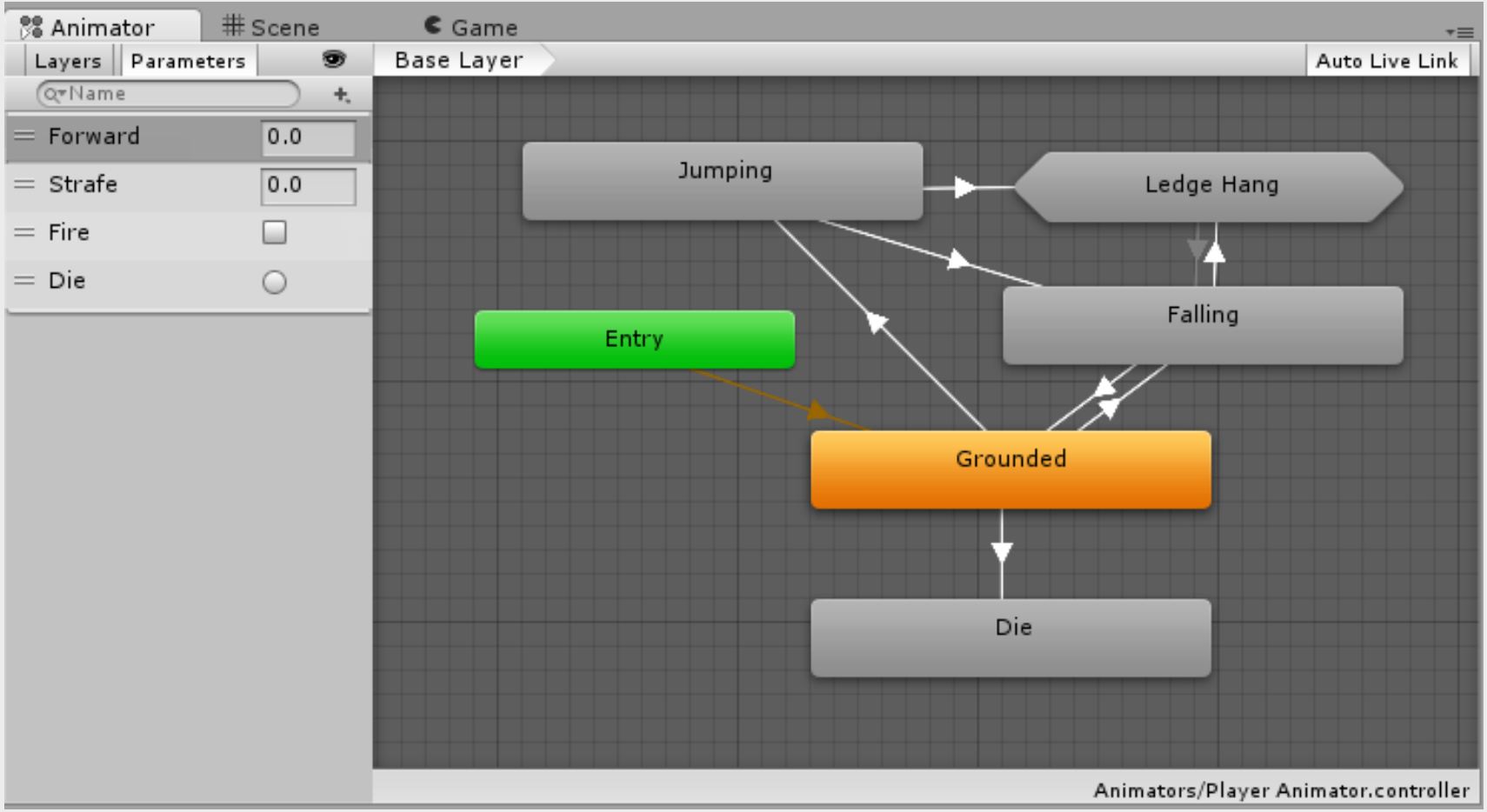
Unity MonoBehaviour lifecycle örneği



Unreal blueprint örneği (Actor Class)

ANİMASYON

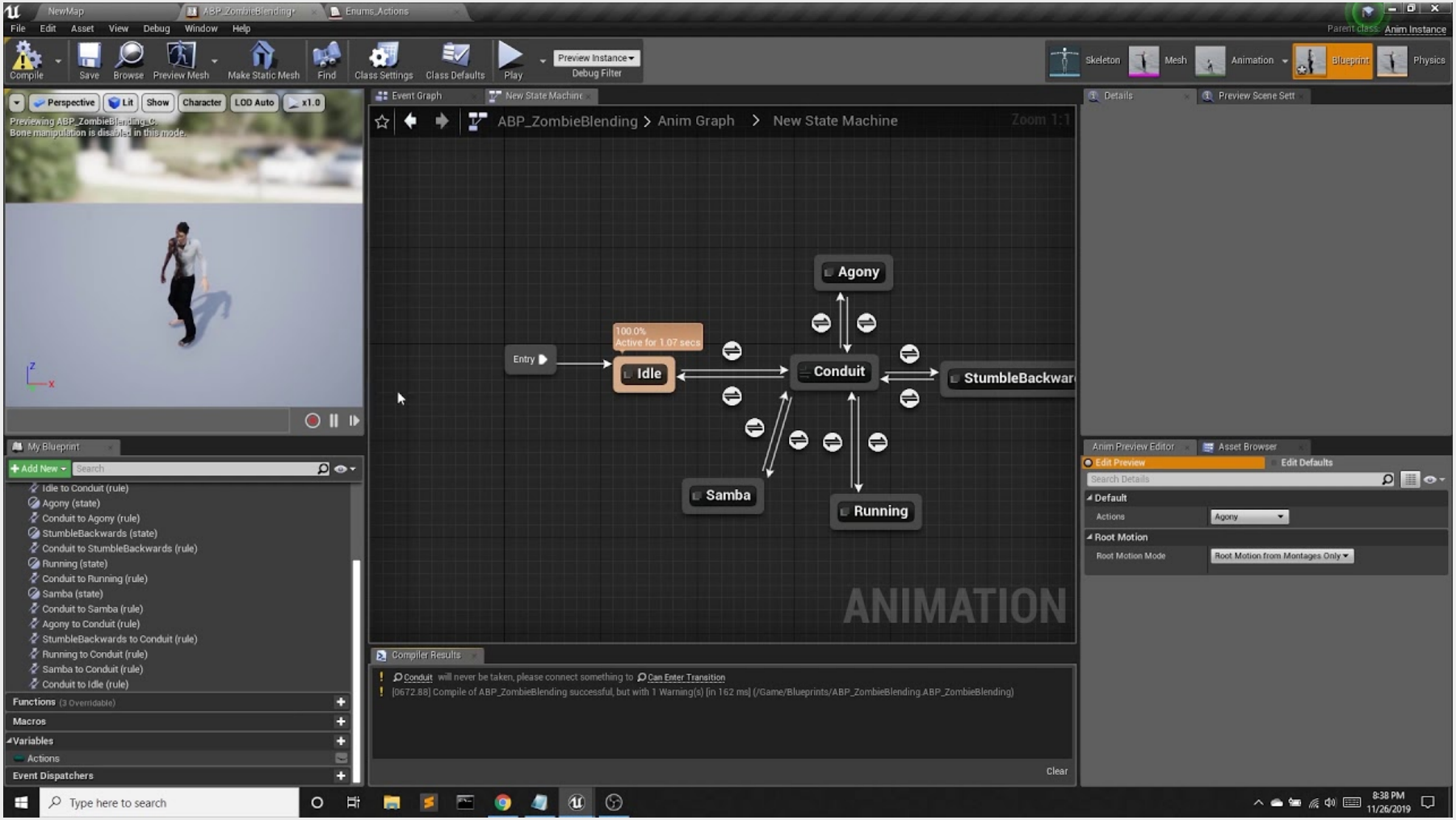
Araçları



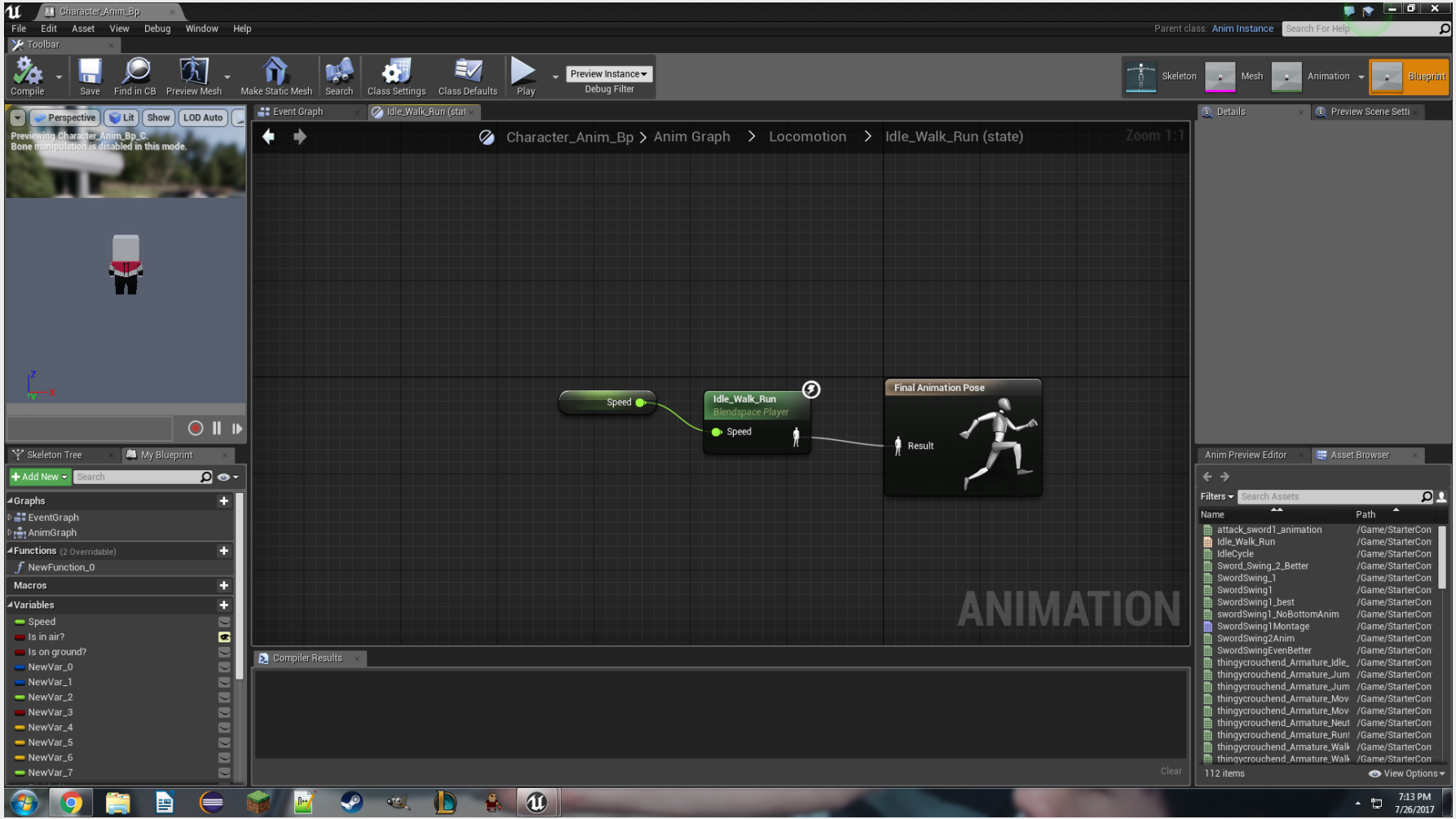
Unity Animator Component

```
using UnityEngine;
using System.Collections;

public class RedScript : MonoBehaviour {

    Animator anim;

    void Start ()
    {
        anim = GetComponent<Animator>();
    }

    void Update ()
    {
        if (Input.GetKeyDown(KeyCode.Space)) anim.SetTrigger("MakeRed");
    }
}
```

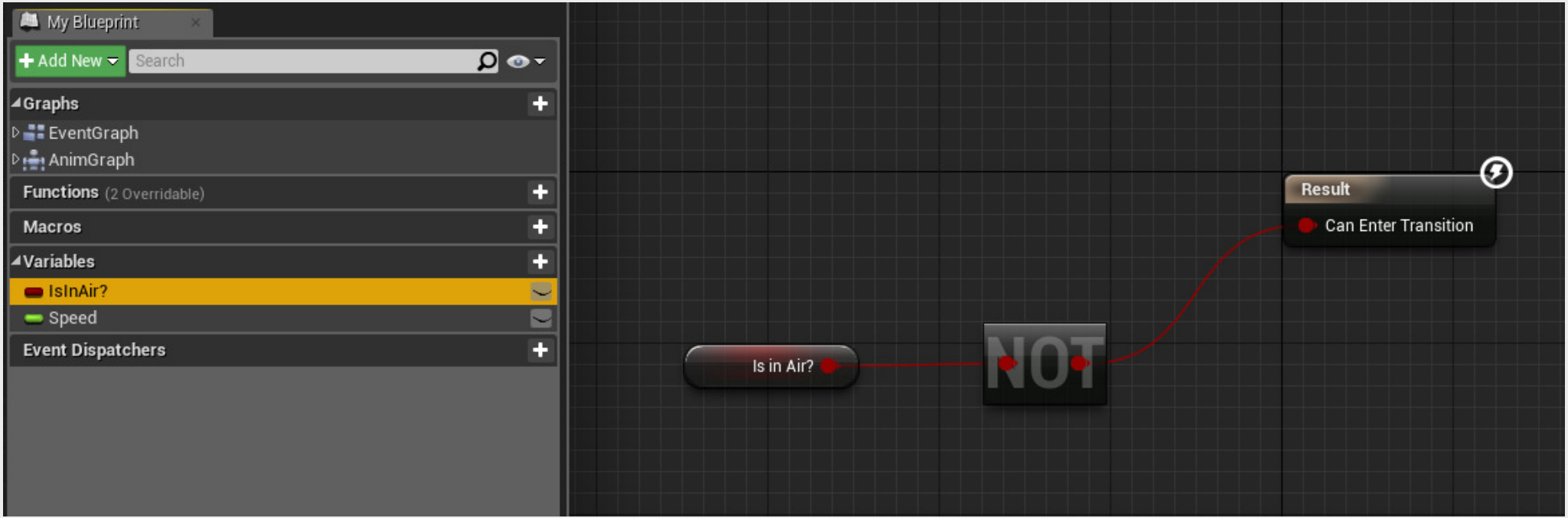
Unity Script ile Animator Component ile haberleşme



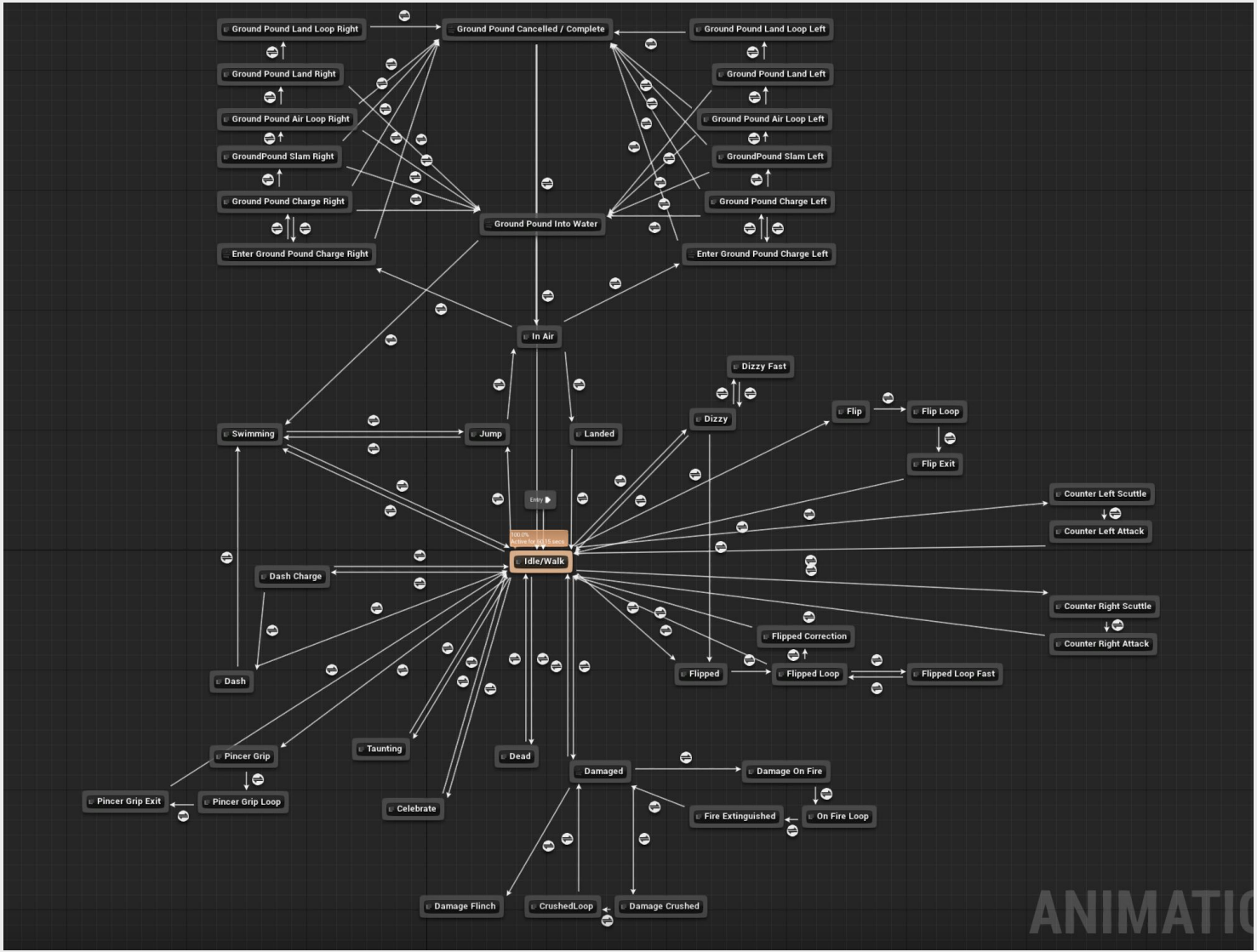
Unreal’daki Çok Derinlikli Animasyon Yapısı



Unreal Engine Animasyon Aracı



Unreal Engine Animation Transition Blueprint ekranı


Ground Pound Land Loop Right
Ground Pound Cancelled / Complete
Ground Pound Land Loop Left
Ground Pound Land Right
Ground Pound Land Left
Ground Pound Air Loop Right
Ground Pound Air Loop Left
GroundPound Slam Right
GroundPound Slam Left
Ground Pound Charge Right
Ground Pound Charge Left
Ground Pound Into Water
Enter Ground Pound Charge Right
Enter Ground Pound Charge Left
In Air
Dizzy Fast
Dizzy
Flip
Flip Loop
Flip Exit
Swimming
Jump
Landed
Entry
Counter Left Scuttle
Counter Left Attack
Idle/Walk
Dash Charge
Counter Right Scuttle
Counter Right Attack
Flipped Correction
Dash
Flipped
Flipped Loop
Flipped Loop Fast
Pincer Grip
Taunting
Dead
Damaged
Damage On Fire
Pincer Grip Exit
Pincer Grip Loop
Celebrate
Fire Extinguished
On Fire Loop
Damage Flinch
CrushedLoop
Damage Crushed

UNREAL ENGİNE

Daha kompleks bir editör ve geliştirme mimarisi ama büyük projeler için daha düzenli bir geliştirme ortamı

Daha iyi grafik altyapısı ve daha büyük poligonlarla başa çıkabilme

Nanite, lumen gibi otomatik optimizasyon araçları

Daha artist-friendly kullanım yapısı

Blueprints yapısı

UNITY

Anlaşılması kolay, sezgisel bir editör ve daha basit bir geliştirme mimarisi

Küçük çaplı projeler ve az sistem gereksinimi isteyen işler için daha uygun.

Daha özgür bir geliştirme ortamı

Daha iyi cross-platform desteği

İki farklı render Pipeline ile her cihaza göre sistem gereksinimlerini ayarlayabilme

Daha programmer-friendly kullanım yapısı

UNREAL ENGİNE

Basit projeler için fazla kompleks bir editör, ve genel basit işler için motorla boğuşma durumu

Blueprints'in kompleks projelerde sınırlandırıcı etkisi ve C++ desteği gerektirmesi

Fazla sistem gerektirmesi bu yüzden verdiği çıktıların alt düzey cihazlar için çok da uygun olmaması

UNITY

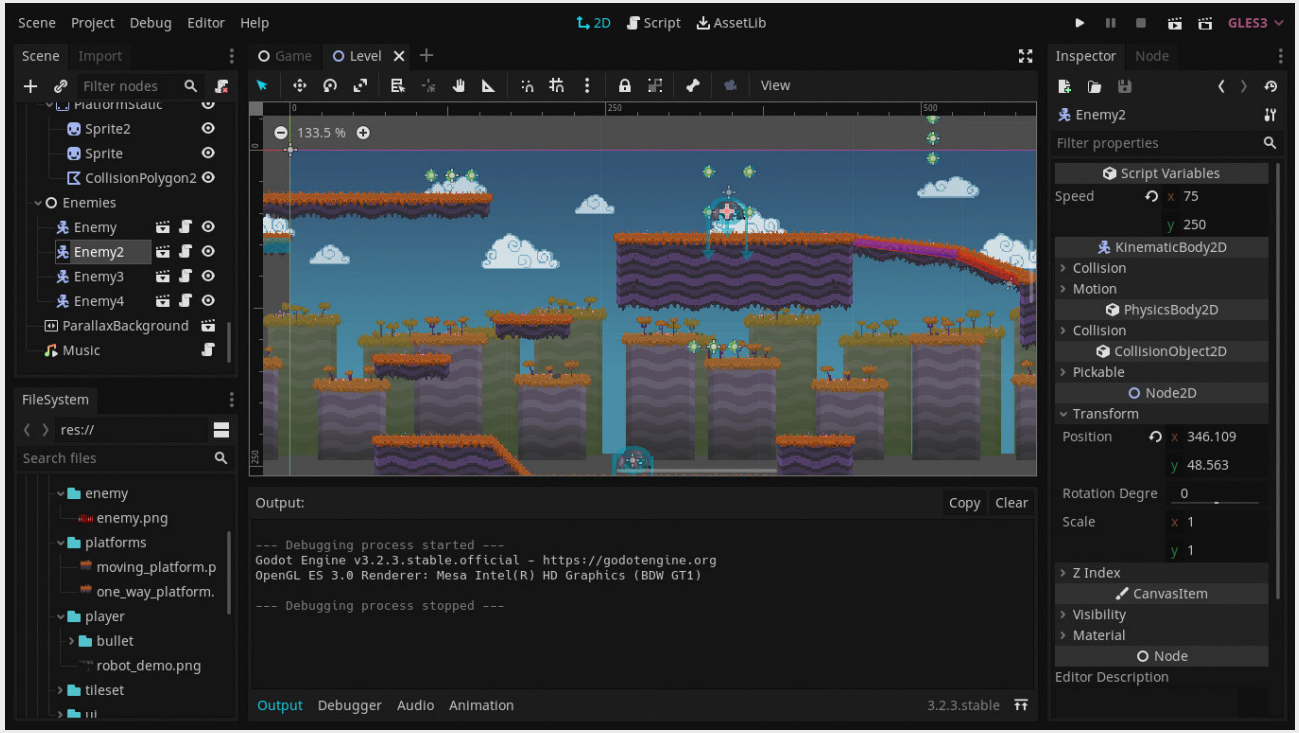
Kompleks projeler için kontrolü giderek zorlaşan yönetilmesi zor geliştirme mimarisi

İki farklı render pipeline olduğu için assetlerin uyumluluk problemi

Built-in özellikler yerine add-on lara fazlaca ihtiyaç duyması

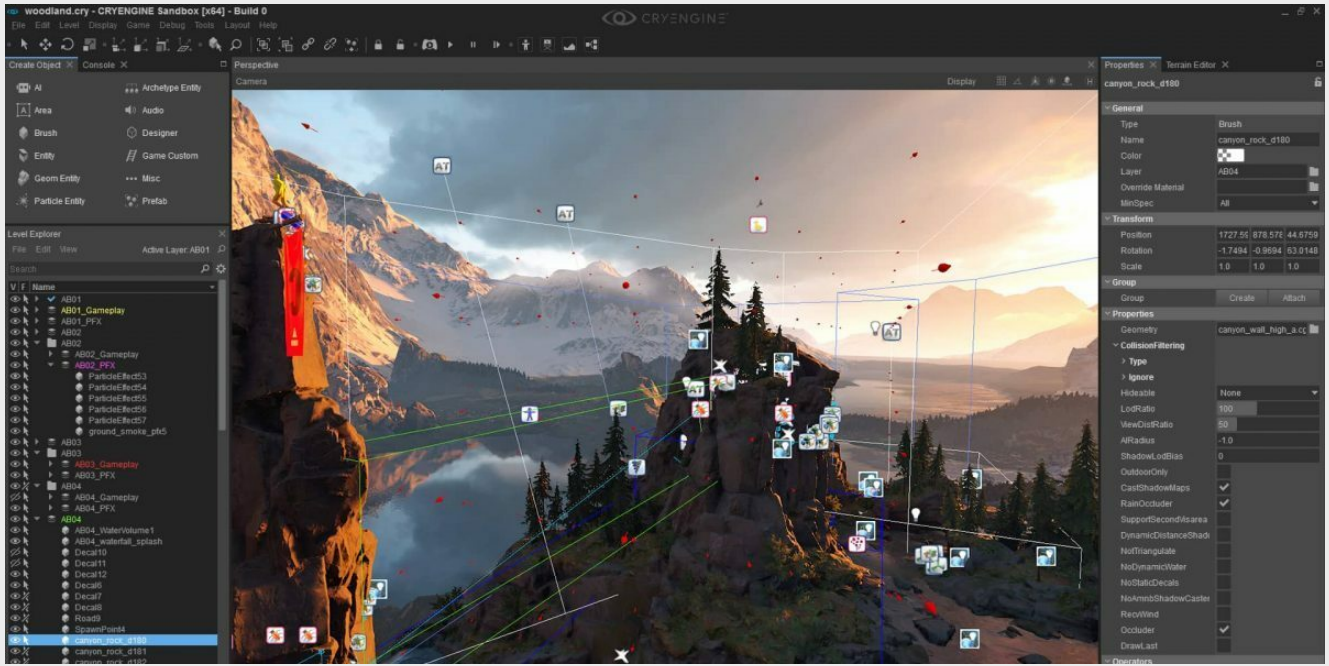
Kodların hızlıca spagetti kod olma ihtimaline karşı dikkatli çalışma gerektirmesi

GODOT



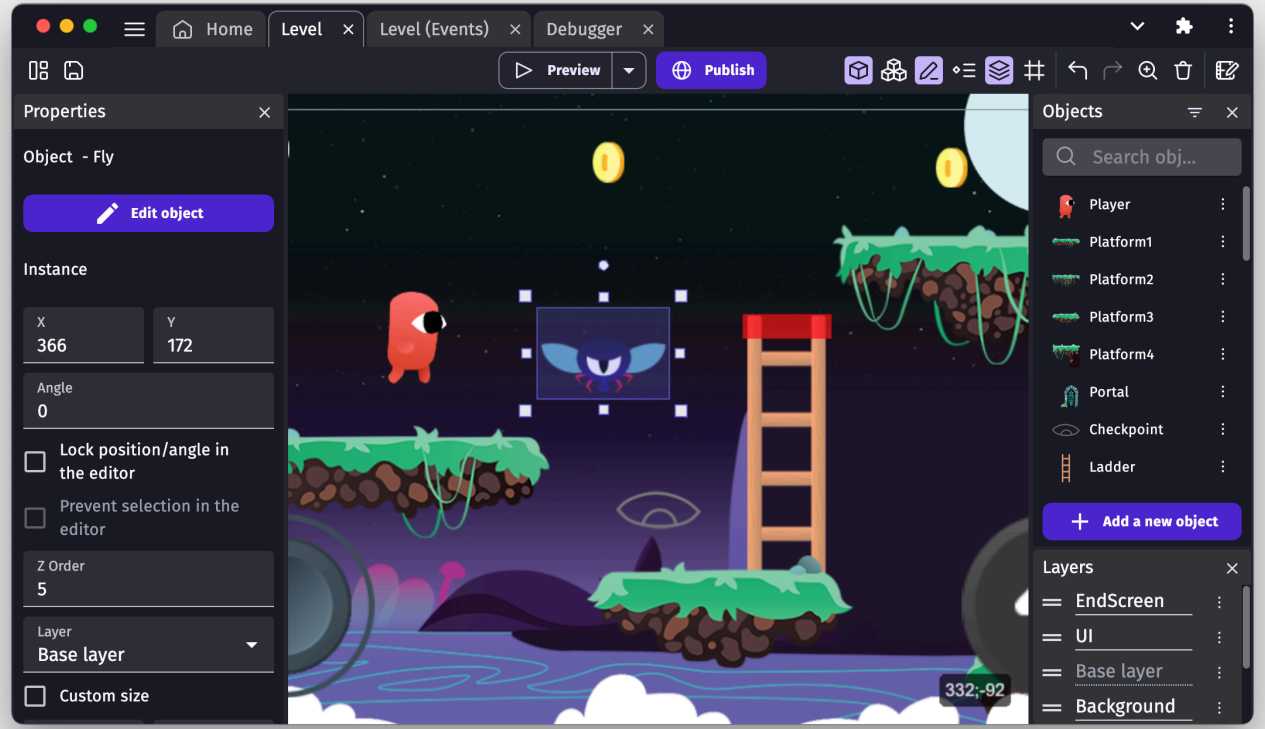
Godot, ilk olarak açık kaynak kodlu ve ücretsiz olarak halka açılmıştır. Platformlar arası bir geliştirme ortamı sağlayan Godot, MacOs, Windows ve Linux'da çalışır. Çıktılarını bilgisayar, mobil ve web için verebilmektedir. İlk olarak 2014'te Juan Linietsky, Ariel Manzur tarafından yayınlanmıştır. Hem 2d, hem 3d geliştirmeye olanak sağlayan Godot'un 3d konusunda ana rakipleri kadar başırılı olduğu söylenemez ancak 2d için güçlü bir seçimdir; bu sebeple şu an henüz daha çok indie ekipler tarafından kullanılmaktadır.

CryENGINE

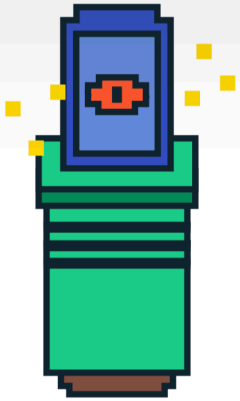


CryEngine, Crytek şirketinin geliştirmiş olduğu bir oyun motorudur. Şirketin kendi oyunları için geliştirilmiş olan motor daha sonra halka yayınlanmıştır. Daha çok birinci şahıs nişancı oyunları için tasarlanmıştır. (Örneğin Far Cry bu motorla geliştirildi.) Güncel sürümü 5.7 olan motorun geliştirilme süreci oldukça yavaşlamıştır ve pek çok kişi motorun ölmeye başladığını savunmaktadır, bu yüzden popülaritesini gün geçtikçe kaybeden motor, yeni bir ekipler için riskli bir yatırım olduğu söylenebilir.

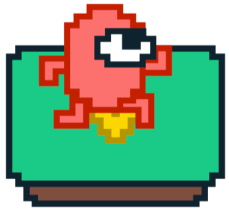
GDevelop



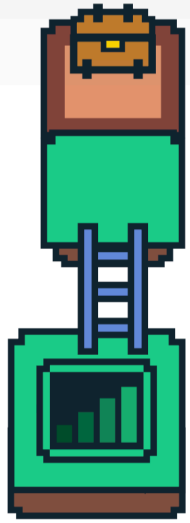
GDevelop, açık kaynak kodlu, ücretsiz, multi-platform geliştirmeye imkan sağlayan(pc, mobil, web), no-code olarak tanımlanan bir oyun geliştirme motorudur. 2d ve 3d geliştirmeye ortam sağlar. Sezgisel ve hızlı oyun üretme amacı taşır. Oyuna interaktivite kazandırabilmek için yazılım dilleri yerine sürükle-bırak bir event sistemi kullanır.

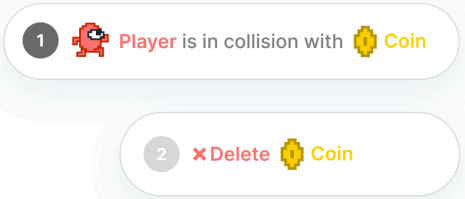


Oyun Yapmanın Kolay, Sezgisel Yolu

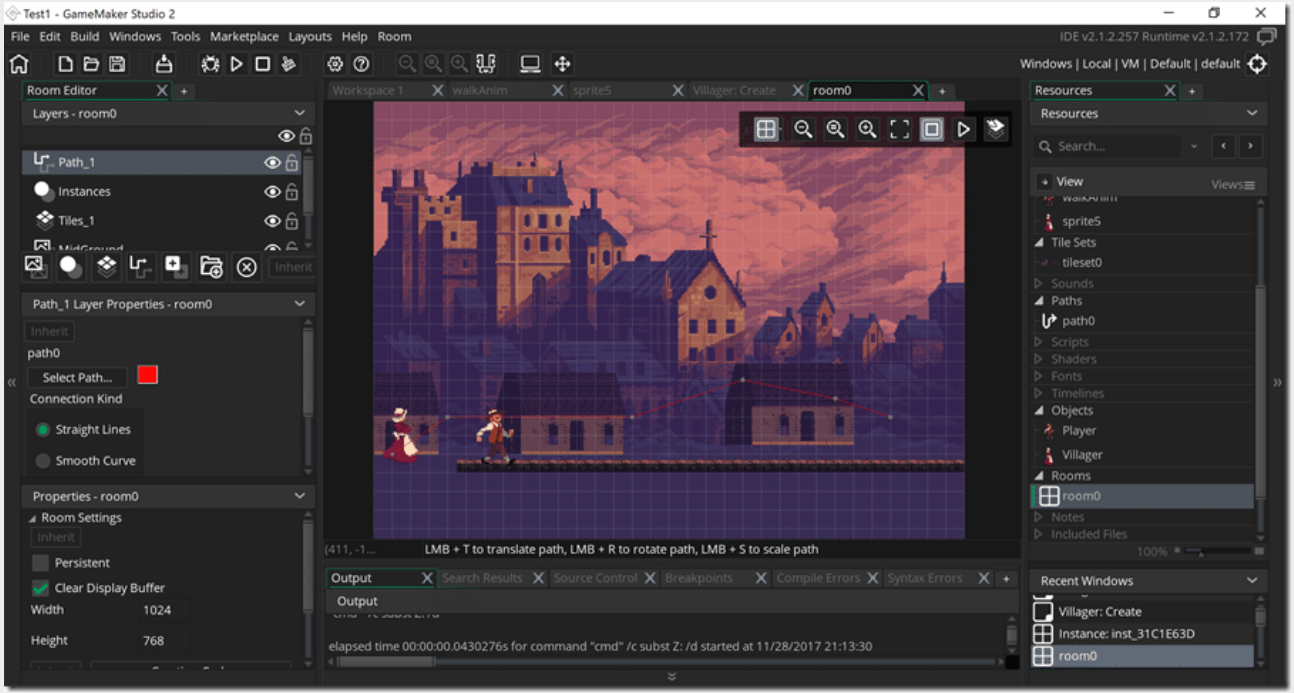


What makes GDevelop **unique and so easy to use** is the event system. Events are a powerful way to express the logic of your game: it's as efficient as coding, but without the complexities of a programming language.



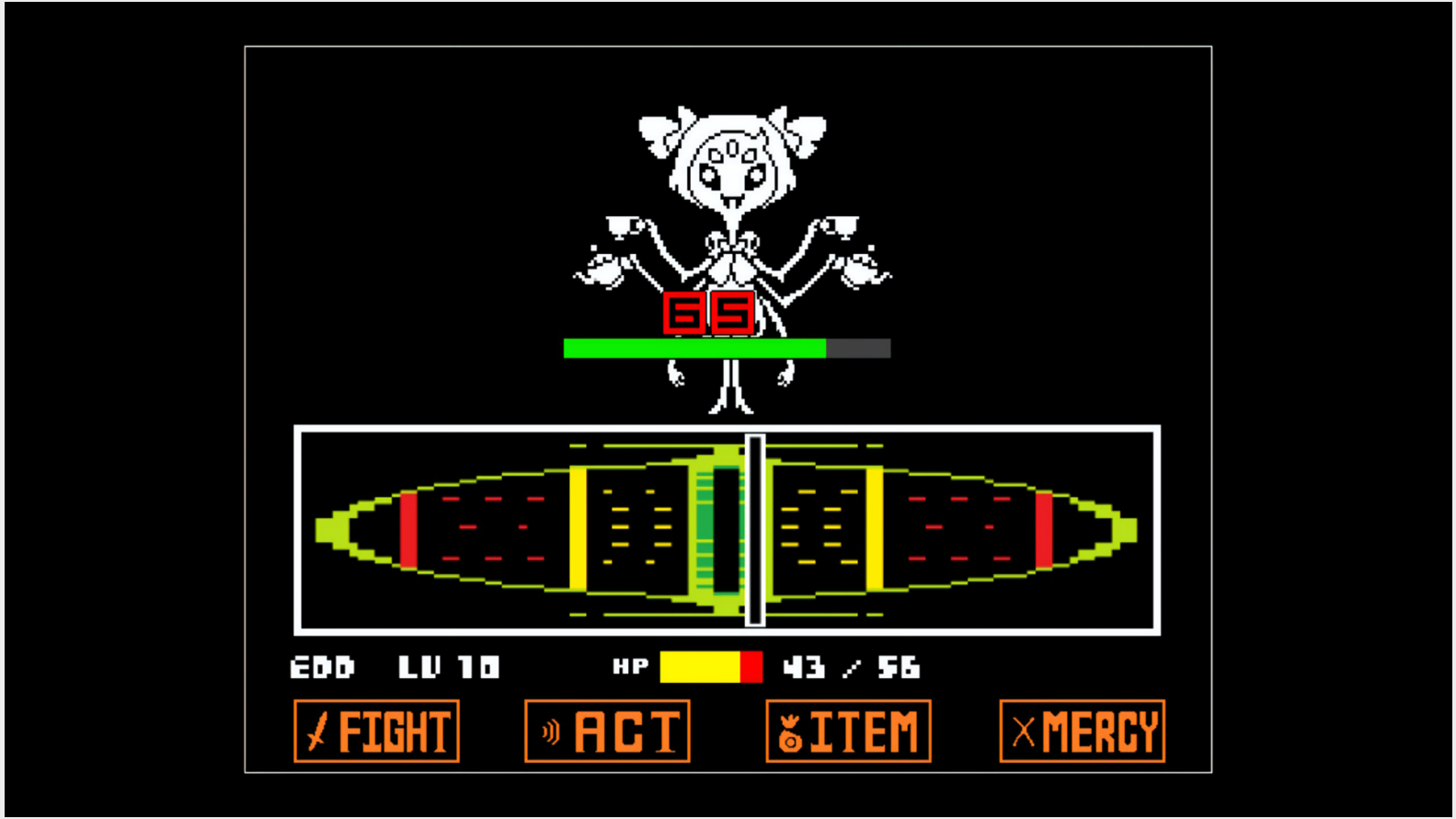


GameMaker



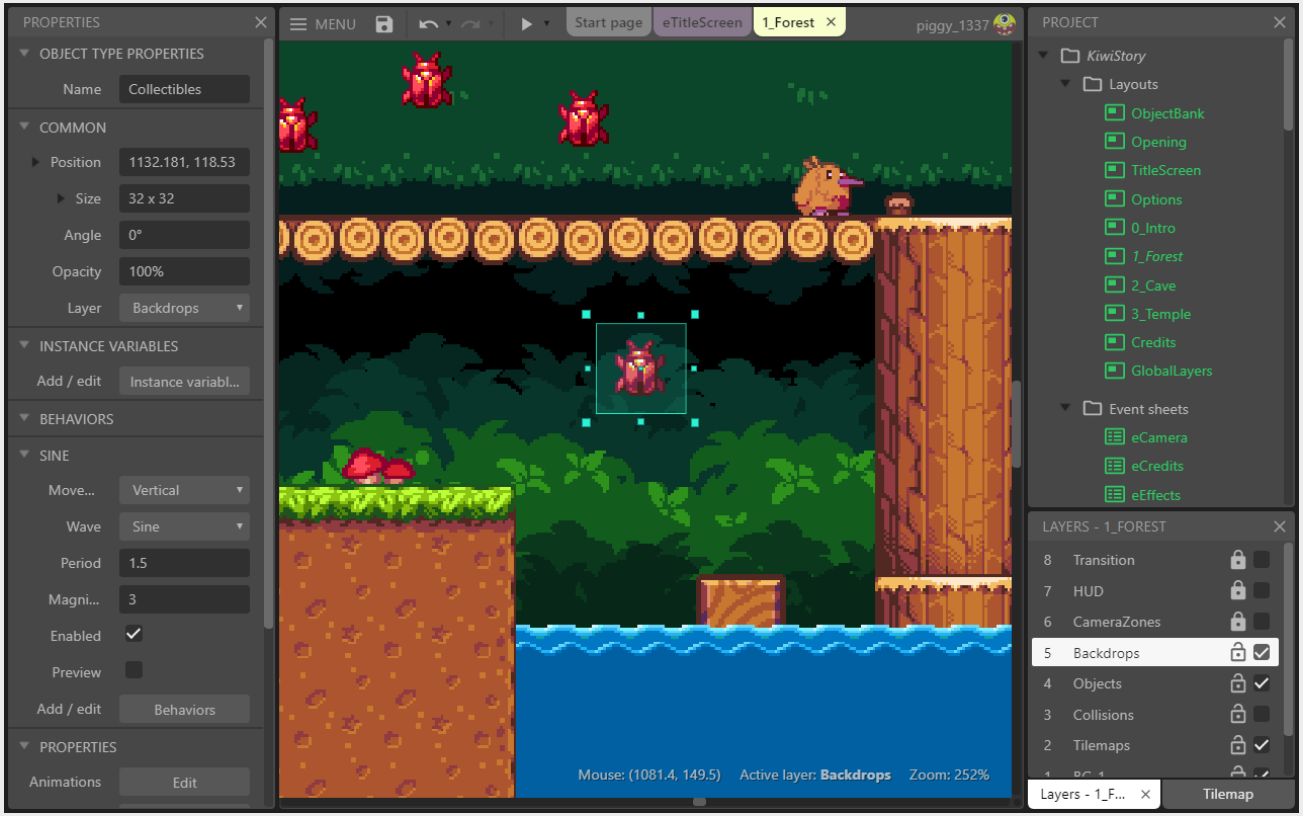
Ücretli ve ücretsiz sürümleri bulunan Game Maker, 2d oyunlar üzerine yoğunlaşmış bir motordur. “Action” denen davranışları sürükle bırak mantığıyla oyun yapımını kolaylaştırdığı gibi betik dillerle esnek bir çalışma ortamı da sunar. Eski bir teknoloji olduğu söylenebilir. Bilinen bazı pixelart oyunlar bu motorla geliştirilmiştir.

Toby Fox, tarafından tek başına geliştirilen oyun, GameMaker motorunu kullanarak geliştirildi.



İyi bir rol yapma örneği olan oyunda karşımıza çıkan bossların çok farklı davranış şekilleri vardı ve sadece savaşarak değil konuşarak da bossları geçebiliyorduk.

Construct 3



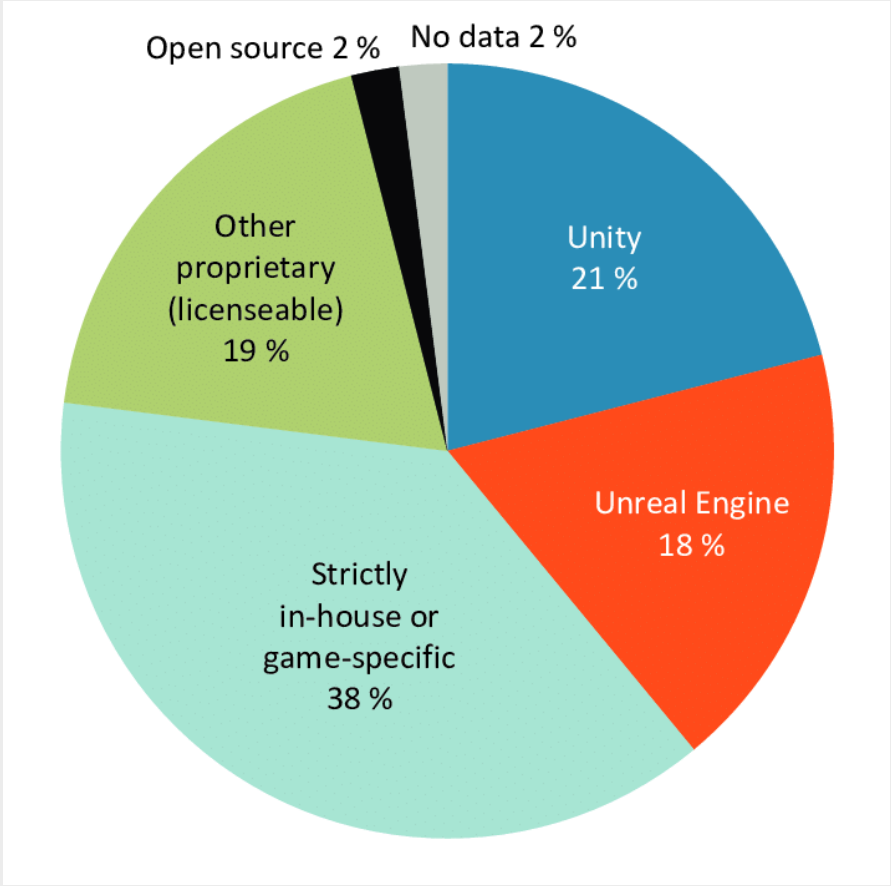
Construct, html tabanlı 2d bir oyun geliştirme platformudur. Önceliği programcı olmayanlara yöneliktir ancak 2019'da Javascript desteği getirerek ileri ihtiyaçlarıda karşılamaya yönelik bir hamlede bulunmuştur. Construct, nesneler ile event-based bir yaklaşımla etkileşime geçerek oyun geliştirmeyi ön plana çıkarır.

In-House Çözümler

Şirket içi motorlar

Nintendo, Switch platformuna çıkardığı “Legends of Zelda”, “Mario Odyssey” gibi exclusive oyunlarını kendi geliştirdiği motorlar üzerinden geliştirmektedir.

Yerli oyun firması Taleworlds, Mount and Bade oyununda kendi geliştirdiği motoru kullanmakta..



Steam'de yayınlanan en popüler 250 oyunun geliştirildiği engineler (** The Little Engines That Could – Game Industry Platforms and the New Drivers of Digitalization, Juri Mattilla, Timo Seppala, ResearchGate)

FRAMEWORKLER

OpenGL, DirectX

https://history-computer.com/opengl-vs-directx-key-differences-and-full-comparison/

Teşekkürler..

Kaan ESKİCİ

H.Mert DOĞARAY