

Usporedba okruženja za izradu videoigara

Žužić, Alesandro

Undergraduate thesis / Završni rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Pula / Sveučilište Jurja Dobrile u Puli**

Permanent link / Trajna poveznica: <https://urn.nsk.hr/um:nbn:hr:137:120808>

Rights / Prava: [Attribution 4.0 International/Imenovanje 4.0 međunarodna](#)

Download date / Datum preuzimanja: **2023-12-27**



Repository / Repozitorij:

[Digital Repository Juraj Dobrila University of Pula](#)

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

Alesandro Žužić

USPOREDBA OKRUŽENJA ZA IZRADU VIDEOIGARA

Završni rad

Pula, 2022.

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

USPOREDBA OKRUŽENJA ZA IZRADU VIDEOIGARA

Završni rad

JMBAG: 03030881619, redovni student

Studijski smjer: Informatika

Kolegij: Multimedijalni sustavi

Znanstveno područje: Društvene znanosti

Znanstveno polje: Informacijske i komunikacijske znanosti

Znanstvena grana: Informacijski sustavi i informatologija

Mentor: doc. dr. sc. Željka Tomasović

Pula, rujan 2022.



IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Alesandro Žužić, kandidat za prvostupnika informatike ovime izjavljujem da je ovaj Završni rad isključivo rezultat mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljeni način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoći dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

Alesandro Žužić



IZJAVA O KORIŠTENJU AUTORSKOG DJELA

Ja, Alesandro Žužić, dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj Završni rad pod nazivom „Usporedba okruženja za izradu videoigara“ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

Student

Alesandro Žužić

U Puli, 26. rujna 2022. godine

Sadržaj

1.	UVOD	1
2.	OPĆENITO O OKRUŽENJIMA ZA IZRADU 3D VIDEO IGARA	2
2.1	Unity	2
2.2	Godot	3
2.3	Unreal Engine	4
3.	PROGRAMIRANJE	6
3.1	Unity - C#.....	6
3.2	Godot - GDScript	9
3.3	Unreal Engine – Blueprints	11
3.4	Usporedba koda	14
3.4.1	Unity	14
3.4.2	Godot	16
3.4.3	Unreal Engine	17
4.	SUČELJE PROGRAMA.....	20
4.1	Unity	20
4.1.1	Hijerarhija.....	21
4.1.2	Inspektor	21
4.1.3	Projekt	22
4.1.4	Scena/Igra	22
4.2	Godot	24
4.2.1	Stablo Scena	24
4.2.2	Inspektor/čvor	25
4.2.3	Datotečni sustav	26
4.2.4	Radni Prostor	26
4.3	Unreal Engine	27
4.3.1	Obrisnik	28

4.3.2 Detalji	28
4.3.3 Preglednik Sadržaja	29
4.3.4 Prozor za prikaz	30
5. FUNKCIONALNOSTI	31
5.1 Materijali	31
5.1.1 Unity	31
5.1.2 Godot	32
5.1.3 Unreal Engine	33
5.2 Teren	34
5.2.1 Unity	35
5.2.2 Godot	36
5.2.3 Unreal Engine	36
5.3 Sustav čestica	38
5.3.1 Unity	38
5.3.2 Godot	38
5.3.3 Unreal Engine	39
5.4 Animacije	40
5.4.1 Unity	41
5.4.2 Godot	41
5.4.3 Unreal Engine	42
5.5 Korisničko Sučelje	43
5.5.1 Unity	43
5.5.2 Godot	44
5.5.3 Unreal Engine	44
6. ISCRTAVANJE SLIKE	46
6.1 Unity	46
6.2 Godot	47

6.3 Unreal Engine	49
7. TABLICA USPOREDBE	51
8. ZAKLJUČAK	53
9. POPIS LITERATURE	55
10. POPIS SLIKA	60
11. POPIS TABLICA	63
12. POPIS GRAFIKONA	64

1. UVOD

Trenutno postoji velika količina okruženja za izradu videoigara, čak preko šezdeset i tri (63) [1]. Većina tih okruženja su programski proizvodi otvorenog koda (*engl. open-source software*) koje održavaju razne zajednice programera. Okruženja za izradu video igara su ujedno i programski alati koji omogućavaju jednostavniju i bržu izradu videoigara. Međutim, vrlo je teško odlučiti se za specifično okruženje za izradu videoigara, zato što nisu sva okruženja ista. Koriste različite programske jezike, funkcionalnosti te imaju različite mogućnosti. Neka okruženja su jednostavnija i mogu se brže naučiti, dok su druga kompleksnija i zahtijevaju više truda i vremena. Takve nedoumice otežavaju korisniku, a posebice početniku, odabir okruženja za izradu videoigara. Stoga, u ovom radu se nastoje riješiti nedoumice usporedbom triju popularnih okruženja za izradu videoigara – Unity, Godot i Unreal Engine. Ovim radom će se nastojati opisati osnovne komponente svakog od okruženja te navesti međusobne sličnosti i razlike. Da bi se postigla uspješna evaluacija, za potrebe ovog rada izradit će se tri jednostavne videoigre, tako da će se svaka napraviti u jednom od okruženja, pritom nastojeći da budu što sličnije. Videoigra će biti jednostavna *pucačina iz prvog lica* (*engl. first-person shooter*). Imat će jednostavne mehanike kretanja i pucanja. Objekti u koje igrač puca su: bačve, kutije, ograde, ljestve, daske i sl. Svi ti objekti će se moći razbiti i svaki će imati definirani broj života. Kroz igru, periodično se stvaraju zombiji u unaprijed određenim intervalima. Svrha zombija je da napadaju igrača, a igračev cilj je da ih izbjegava i eliminira. Eliminiranjem zombija, igrač dobiva bodove koji se pri završetku igre (kada igrač umre) zbroje i prikažu. Eliminiranjem zombija se povećava težina igre na načina da se sve više zombija stvara. Ovim demo projektom će se nastojati evaluirati sljedeći dijelovi: programiranje, sučelje programa, funkcionalnosti (materijali, teren, sustav čestica, animacije, i korisničko sučelje) te iscrtavanje slike (*engl. rendering*).

Na sljedećoj poveznici nalaze se videoigre izrađene u svima trima okruženjima:

<https://drive.google.com/drive/folders/1QDVqBIWYEfBUQLZ0yLJSFQMs1kOcyS96?usp=sharing>

2. OPĆENITO O OKRUŽENJIMA ZA IZRADU 3D VIDEO IGARA

Okruženja za izradu videoigara su programi koji korisnicima omogućavaju izradu dvo-dimenzionalnih (2D) i tro-dimenzionalnih (3D) igara te im pružaju mnogobrojne korisne alate za pomoć u konceptualizaciji te implementaciji same videoigre.

2.1 Unity

Unity je više-platformsko (*engl. cross-platform*) okruženje za izradu videoigara koji je razvila Unity Technologies tvrtka. Unity korisniku pruža temeljne funkcionalnosti poput: mehanizma iscrtavanja (*engl. rendering engine*), uvoza i korištenja zvuka, simulacije fizike (*engl. physics engine*), animacije i umrežavanja (*engl. networking*) [2]. Unity omogućava izgradnju (*engl. production build*) video igara za razne uređaje i operacijske sustave, poput:

- pametnih mobitela (iOS i Android),
- stolnih i prijenosnih računala (Windows, Mac OS, Linux),
- igračih konzola (PlayStation, Xbox, Nintendo)

te za tehnologije poput virtualne stvarnosti (Oculus Rift, Gear VR, Steam VR) [3]. Popularan je među programerima početnicima zbog velike količine dostupnih materijala za učenje te biblioteka s gotovim funkcijama za učestale radnje [4]. Osnovno aplikacijsko programsко sučelje koristi objektno-orientirani programski jezik C# [5]. Unity je vlasnički softver, to jest softver zatvorenog koda (*engl. closed-source software*). Ima besplatnu i plaćenu opciju licenciranja. Besplatna licenca je za osobnu upotrebu ili manje tvrtke koje generiraju manje od 100.000 dolara godišnje, kasnije podignuto na 200.000 dolara, a pretplate se temelje na prihodima ostvarenim igrama koje koriste Unity. Kreatori mogu razvijati i prodavati materijale koje su stvorili drugim proizvođačima videoigara putem Unity Asset Store-a. Neke od poznatijih igri napravljene koristeći Unity su: „Pokémon Go“, „Monument Valley“, „Call of Duty: Mobile“, „Beat Saber“ i „Cuphead“.

Postoje 4 različite verzije Unity-a:

- LTS (Long Term Support),
- Legacy LTS,
- Beta tehnički tokovi (*engl. beta tech stream*) i
- Alpha verzije.

LTS verzija programa pruža programerima dugoročnu podršku te maksimalnu stabilnost projektima koji su u izradi ili su spremni za tržište. Drugim riječima, pruža razvojnim timovima igara koje su već dulje vrijeme u izradi ili su gotove podršku verzije programa u kojoj su programeri inicijalno započeli razvijati igru, kako ne bi došlo do nekompatibilnosti [6]. Ažuriranja pokrivaju samo „popravke“ usmjerene na poboljšanje stabilnosti programa kako bi se korisnicima omogućio nesmetani rad na projektima [7]. Alpha i Beta tehnički tokovi su Unity verzije koje omogućavaju rani pristup značajkama u razvoju te rani pristup budućim planovima i alatima koji se mogu koristiti za izradu prototipa [8], [9].

Za izradu demo videoigre za ovaj rad korištena je verzija 2021.3.7f1 LTS. Svi modeli, objekti, efekti koji nisu samostalno izrađeni, preuzeti su besplatno s Unity Asset Store-a u svrhu izrade i testiranja Unity okruženja za izradu videoigre.

2.2 Godot

Godot je više-platformsko, besplatno, okruženje otvorenog koda (*engl. open source*) za izradu videoigara. Objavljen je pod MIT (Massachusetts Institute of Technology) licencom (permisivna licenca slobodnog softvera). Radi na više operacijskih sustava uključujući: Linux, BSD (Berkeley Software Distribution), macOS i Microsoft Windows-u [10]. Dizajniran je za izradu 2D i 3D videoigara za desktop, mobilne i web platforme. Također se može koristiti za izradu softvera koji nije za igre, uključujući uređivače (*engl. editors*).

Godot trenutno službeno ne podržava izradu videoigara za konzole (osim za Xbox One putem UWP-a (Universal Windows Platform)) iz razloga što je Godot, kako je već rečeno, program otvorenog koda pod MIT licencom, a za izradu videoigara za konzole razvojni softver mora biti licenciran u sklopu registrirane tvrtke. SDK-ovi (Software development kit) konzola su tajni i zaštićeni ugovorima o tajnosti podataka, što onemogućuje da se kod objavi kao otvoren kod. Tvrtke trećih strana nude usluge prebačaja i objavljivanja videoigara izrađenih u Godot-u na konzole. Jedna od takvih tvrtki, „Lone Wolf Technology“, nudi prebacivanje i objavljivanje Godot videoigara za Switch, PS4 i Xbox One [11].

Cilj Godot-a je ponuditi potpuno integrirano okruženje za razvoj videoigara. Omogućuje programerima stvaranje videoigre bez potrebe za drugim alatima (osim onih koji se koriste za stvaranje sadržaja poput grafičkih elemenata, glazbe i sl.). Godot je građen oko koncepta stabla čvorova. Čvorovi su organizirani unutar scena, koje se mogu upotrebljavati više puta te se mogu međusobno nasljeđivati (*engl. inheritance*). Svi resursi igre, uključujući programske skripte i grafički elementi spremaju se unutar datotečnog sustava računala [12].

Godot podržava niz programskih jezika za izradu videoigara, uključujući: integrirani jezik *GDScript*, *C++* i *C#*. Dodatno, omogućava stvaranje veza s drugim jezicima (*GDNative*). Službeno podržani *GDNative* jezici uključuju *C* i *C++* [13]. Također, Godot nudi mogućnosti vizualnog skriptiranja, međutim ono će biti uklonjeno u Godot verziji 4.0 [14].

Godot kao međunarodni projekt ima aktivnu zajednicu diljem svijeta te je korišten u srednjim školama zbog svoje jednostavne upotrebe za ne-programere ili programere početnike. Neke od poznatijih videoigara napravljenih u Godot-u su: „Deponia“, „Sonic Colors: Ultimate“, „Carol Reed Mysteries“, „Lumencraft“ i „The Zone: Stalker Stories“.

Godot nudi dvije verzije za preuzimanje - Standardnu verziju koja koristi *GDScript* kao programski jezik i verziju sa *C#* podrškom [15]. Za potrebe ovog rada, korištena je standardna verzija 3.5.0.0. Svi modeli, objekti, efekti koji nisu samostalno izrađeni, besplatno su preuzeti s Godot Asset Library-a u svrhu izrade i testiranja Godot okruženja za izradu videoigre.

2.3 Unreal Engine

Unreal Engine je više-platformsko okruženje za izradu videoigara koje je razvila tvrtka „Epic Games“. Koristi se za izradu raznih žanrova videoigara, pretežito pucačina u prvom licu. Također, pokazalo se uspješnim i u drugim industrijama, ponajviše u filmskoj i televizijskoj [16]. Obzirom što je napisan u *C++*, omogućuje visok stupanj prenosivosti, podržavajući širok raspon platformi.

Najnovija generacija okruženja, Unreal Engine 5, ima izvorni kod dostupan na GitHub-u, a komercijalna upotreba dopuštena je na temelju modela tantijema. Epic Games kroz Unreal Engine omogućava korisnicima korištenje mnogih značajki besplatno budući da su kupili tvrtke poput Quixel-a koja nudi sljedeće funkcionalnosti: alat za pregledavanje, pretraživanje, preuzimanje, uvoz i izvoz 3D visoke kvalitete [17].

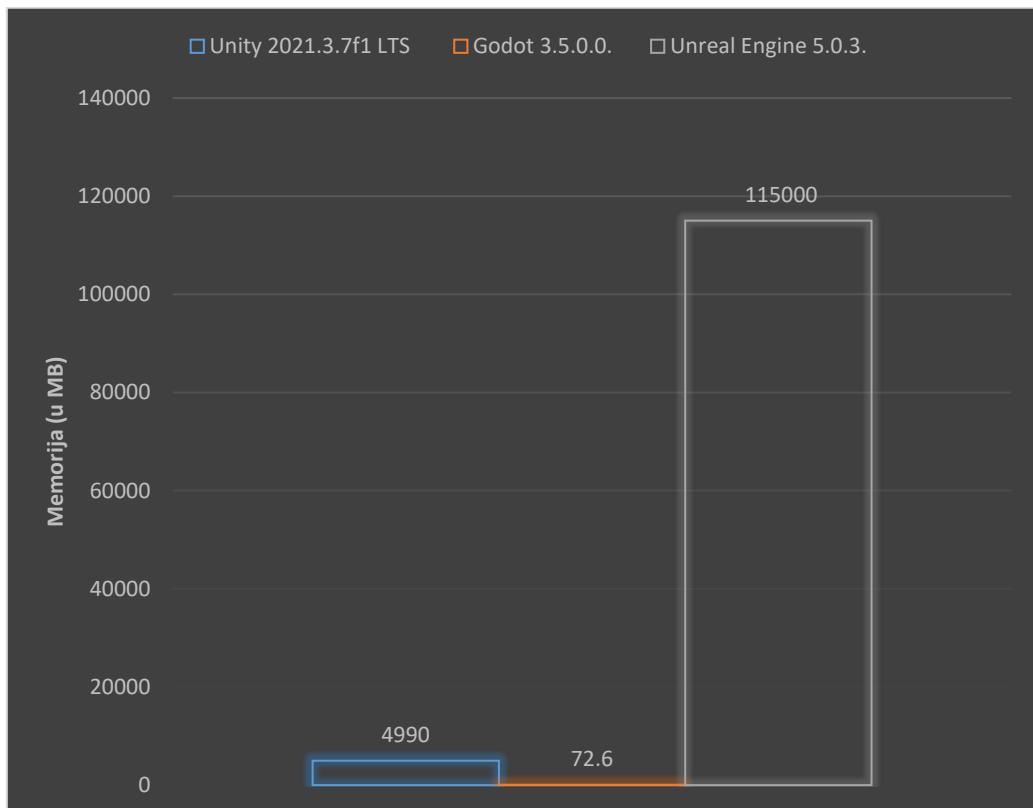
Unreal Engine 5 podržava sve postojeće sustave uključujući sljedeće generacije konzola PlayStation 5 i Xbox Series X/S [18]. Jedna od glavnih značajki je Nanite, tehnologija virtualizirane geometrije, koja omogućuje ubacivanje 3D modela iz Quixel-a. To omogućava programerima stvaranje detaljnih svjetova za videoigre bez potrebe sami izrađuju detaljne modele. Druga značajna komponenta je Lumen. Ona omogućava potpuno dinamično globalno osvjetljenje koje dinamički reagira na promjene scene i svjetla [19].

Unreal Engine koristi nacrt kao sustav vizualnog skriptiranja (*engl. blueprints visual scripting system*), odnosno sustav za skriptiranje koji se temelji na konceptu korištenja sučelja temeljenog

na čvorovima za stvaranje elemenata videoigre unutar Unreal uređivača (*engl. Unreal editor*). Koristi se za definiranje objektno orijentiranih klasa (*engl. object-oriented classes*) ili objekata (*engl. objects*). Sustav je iznimno fleksibilan i snažan jer omogućuje dizajnerima korištenje gotovo cijelog raspona koncepata i alata koji su općenito dostupni samo programerima [20]. Neke od poznatijih videoigara napravljene u Unreal Enginu su: „Gears od War“, „Stray“, „Rocket League“ i „Final Fantasy VII Remake“. Za potrebe ovog završnog rada, korišten je Unreal Engine 5.0.3. Svi modeli, objekti ili efekti koji nisu samostalno izrađeni, besplatno su preuzeti s Unreal Engine Marketplace-a u svrhu izrade i testiranja Unreal Engine okruženja za izradu videoigara.

Potrebno je napomenuti da navedena okruženja zahtijevaju značajno različite memorijske resurse. Unity Verzija 2021.3.7f1 LTS zauzima 4.99 GB, Godot verzija 3.5.0.0. zauzima 72.6 MB, dok Unreal Engine 5.0.3. zauzima 115 GB.

Grafikon 1. Prikaz potrebnih memorijskih resursa za svako od okruženja



Izvor: autor

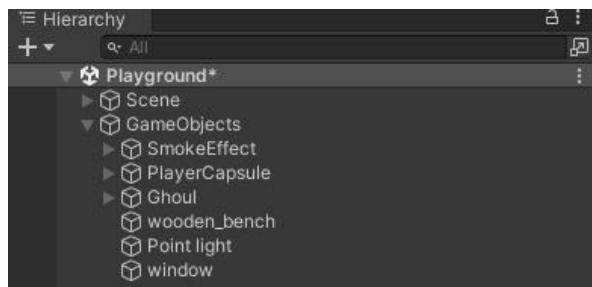
3. PROGRAMIRANJE

U ovom odjeljku prikazat će se: načini programiranja u sva tri okruženja, programski kod, razlike, sličnosti te primjeri koda.

3.1 Unity - C#

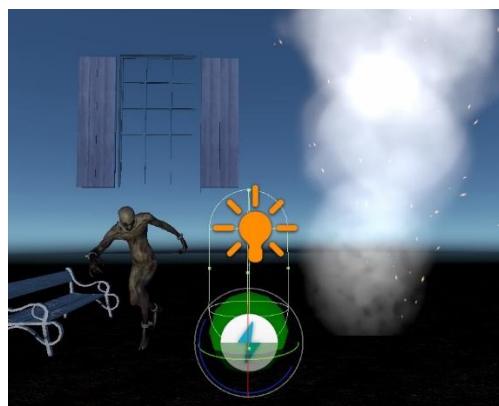
Unity koristi programski jezik *C#*. Svi jezici s kojima Unity radi su objektno orijentirani skriptni jezici. Objekt (*GameObject* u Unity-u) glavna je gradivna jedinica u Unity-u. On može biti lik, predmet, kamera, svjetlo, teren, slika, tekst itd. Sam po sebi, on ništa ne radi te mora imati određena svojstva, atributi (*engl. attributes*) ili skripte (*engl. scripts*). Na slici 1 i 2 su prikazani različiti objekti u hijerarhiji (*engl. hierarchy*) i sceni (*engl. scene*).

Slika 1: Prikaz objekata u hijerarhiji



Izvor: autor

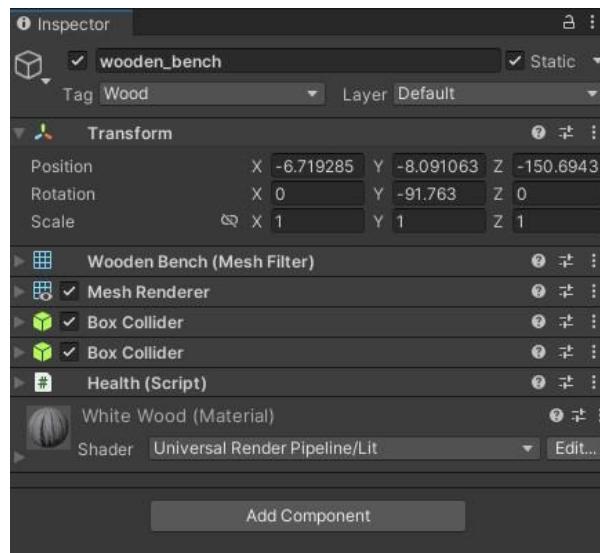
Slika 2: Prikaz raznih objekata u sceni



Izvor: autor

Da bi objekt poprimio sličnosti s objektima iz stvarnog svijeta, primjerice svjetlo, prozor ili kamera, potrebno mu je dodati komponente (*engl. components*). Ovisno o vrsti objekta koji se želi stvoriti, objektu se dodaju različite kombinacije komponenti kao što je prikazano na slici 3. Dodavanjem skripti objektu, moguće mu definirati ponašanje pisanjem programskog koda.

Slika 3: Prikaz komponenti objekta wooden_bench zajedno sa pridruženom skriptom Health u Inspektoru



Izvor: autor

Varijable sadrže vrijednosti i reference na objekte. One su poput kutije koje sadrže vrijednosti koje se mogu koristiti kasnije. Funkcije su zbirke koda koje uspoređuju te varijable i njima upravljaju. Kod je organiziran kroz funkcije tako da se može jednostavno ponovno iskoristiti više puta u različitim dijelovima programa. Klase su način strukturiranja koda za omotavanje kolekcija varijabli i funkcija (atributa i metoda) kako bi se stvorio predložak koji definira svojstva objekta [21]. Na slici 4 prikazan je primjer programske skripte u C#.

Slika 4: Prikaz skripte s varijablama, funkcijama i klasama

```
5  @UnityScript | 0 references
6  public class Demo : MonoBehaviour
7  {
8      int i_am_a_variable;
9
10     0 references
11     public void I_am_a_function()
12     {
13     }
14
15     0 references
16     class I_am_a_class
17     {
18     }
19
20     // Start is called before the first frame update
21     @Unity Message | 0 references
22     void Start()
23     {
24         i_am_a_variable = 10;
25     }
26
27     // Update is called once per frame
28     @Unity Message | 0 references
29     void Update()
30     {
31         i_am_a_variable++;
32     }
33 }
```

Izvor: autor

Unity sadrži mnogo unaprijed definirane funkcije, najčešće korištene su:

- *Awake()*,
- *Start()*,
- *Update()*,
- *FixedUpdate()* i
- *LateUpdate()*

Awake() je prva funkcija koja se poziva kada se objekt instancira bez obzira no to je li komponenta aktivna ili ne. *Start()* se poziva nakon *Awake()* funkcije nakon što je objekt instanciran, a komponenta je aktivna. *Update()* i *FixedUpdate()* se aktiviraju nakon *Start()* funkcije i pozivaju se svaki korak (*engl. frame update*). *FixedUpdate()* ima jednake intervale između dva koraka, dok *Update()* ovisi o performansama od računala do računala. *LateUpdate()* jednak je *Update()* funkciji uz razliku što se poziva nakon nje, odnosno svaki korak [22], [23].

Unity nema ugrađeno sučelje za programiranje, već koristi vanjsko integrirano razvojno okruženje (*engl. Integrated Development Environment - IDE*) - Microsoft Visual Studio za pisanje koda. Omogućuje automatsko prepoznavanje i nadopunjavanje funkcija i varijabli, prepoznaje i označava greške u kodu nudeći moguća rješenja, kao što je pokazano na slici 5. Također, pokazuje referencirane funkcije u drugim dijelovima koda ili skriptama omogućavajući lakše otklanjanje pogrešaka (*engl. debugging*) te samim tim povećava produktivnost [24].

Slika 5: Prepoznavanje i nuđenje rješenja

The screenshot shows a code editor window with C# code. The code defines two methods: `Start()` and `Update()`. In the `Start()` method, there is a line of code: `i_am_a_variable = 10`. A tooltip appears over the identifier `i_am_a_variable`, providing information about it. The tooltip includes:

- readonly struct System.Int32
- CS1002: ; expected
- Show potential fixes (Alt+Enter or Ctrl+.)

```
18
19 // Start is called before the first frame update
20     Unity Message | 0 references
21 void Start()
22 {
23     i_am_a_variable = 10;
24 }
25 // Update is called once
26     Unity Message | 0 references
27 void Update()
28 {
29     i_am_a_variable+="apple";
30 }
31
```

Izvor: autor

3.2 Godot - GDScript

Godot koristi vlastiti dinamički programski jezik - GDScript, visoke razine (*engl. high-level language*). Koristi sintaksu sličnu Python-u (pr. uvlakama se stvaraju programski blokovi). Slika 6 prikazuje primjer programskog koda napisanog u GDScript-u.

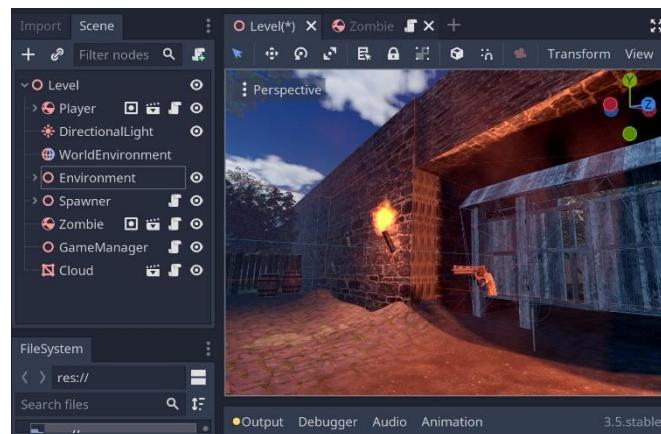
Slika 6: Primjer GDScript koda

```
1 extends Spatial
2
3 export var health = 20
4 onready var particleEffect0 = load("res://Prefabs/Particles/WoodParticleBreak.tscn")
5 onready var particleEffect1 = load("res://Prefabs/Particles/SmokeParticleBreak.tscn")
6
7 func _ready():
8     pass
9
10 func _process(delta):
11     if health <= 0:
12         _InstantiateParticle()
13         queue_free()
14
15 func _InstantiateParticle():
16     var effect_0 = particleEffect0.instance()
17     var effect_1 = particleEffect1.instance()
18     $"/root".add_child(effect_0)
19     $"/root".add_child(effect_1)
20     effect_0.global_transform = transform
21     effect_1.global_transform = transform
22
23     effect_0.set_emitting(true)
24     effect_1.set_emitting(true)
25 |
```

Izvor: autor

GDScript omogućava fleksibilnost za stvaranje sadržaja i integraciju unutar Godot-a [25]. U Godot-u videoigra je ustvari stablo (*engl. tree*) čvorova (*engl. nodes*) koji su grupirani u *scene* (*engl. scenes*) (Slika 7). Čvorovi se mogu međusobno povezivati tako da komuniciraju pomoću signala (*engl. signals*). Čvorovi, scene, stablo scena i signali su četiri gradivna elementa u Godot-u. Igra se dijeli na scene koje se mogu ponovno iskoristiti, gdje scena može biti primjerice: lik, oružje, kutija pa i cijeli nivo (*engl. level*). Scene su fleksibilne te imaju ulogu predefiniranih objekata (koji se mogu ponovno upotrebljavati).

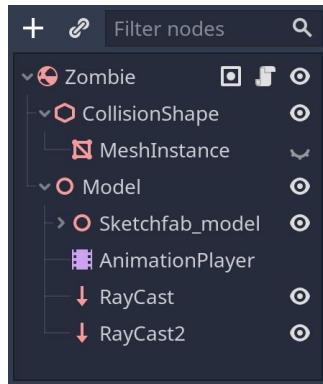
Slika 7: Prikaz scena u Godot-u



Izvor: autor

Čvorovi su najmanji gradivni elementi igre koji se svrstavaju u stabla (Slika 8).

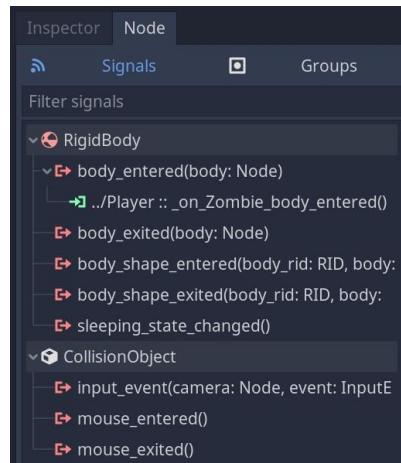
Slika 8: Prikaz čvorova u sceni „Zombie“



Izvor: autor

Stablo čvorova se može spremiti kao scena u projektu koja se kasnije može ponovo iskoristiti [26]. Posljednji koncept koji će se objasniti su signali. Čvorovi aktiviraju signale pri aktivaciji nekog događaja (*engl. event*). Slika 9 prikazuje primjer signala. Tako omogućuju jednostavniju komunikaciju između čvorova te time pružaju fleksibilnost prilikom strukturiranja scena.

Slika 9: Primjer signala u Godot-u



Izvor: autor

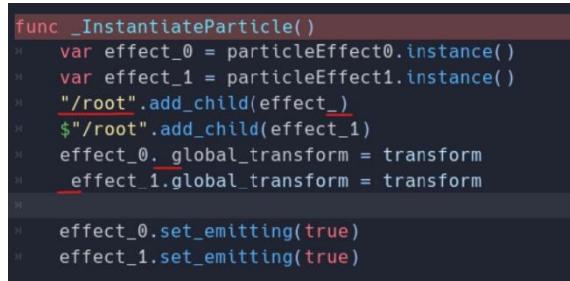
Osnovne funkcije u Godot-u su:

- `_init()`
- `_enter_tree()`
- `_exit_tree()`
- `_process()`
- `_physics_process()`

Funkcija `_init()` je konstruktor klase i poziva se prilikom realizacije klase (izrade čvora). Funkcije `_process()` i `_physics_process()` pozivaju se svaki korak.

Svakom čvoru se može pridružiti točno jedna skripta koja pritom nasljeđuje klasu čvora kome je pridružena. Stvaranjem skripte otvara se integrirano razvojno okruženje Godot-a (*engl. Integrated development environment*). On automatski uvlači linije, prepoznaje i nadopunjuje funkcije i varijable. Označava greške u kodu, međutim mana mu je što označava samo jednu grešku odjednom, kao što je prikazano na slici 10.

Slika 10: Prikaz programske greške u Godot-ovom IDE-u



```

func _InstantiateParticle()
    var effect_0 = particleEffect0.instance()
    var effect_1 = particleEffect1.instance()
    "/root".add_child(effect_0)
    $"/root".add_child(effect_1)
    effect_0.global_transform = transform
    effect_1.global_transform = transform

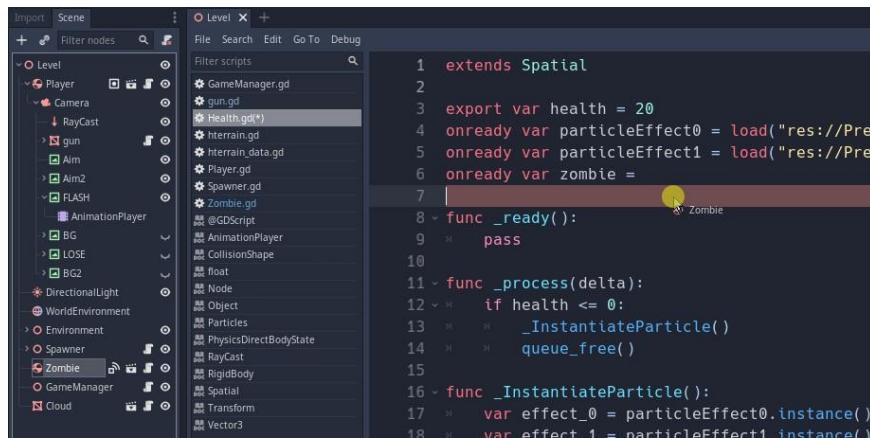
    effect_0.set_emitting(true)
    effect_1.set_emitting(true)

```

Izvor: autor

Godot uređivač koda omogućuje povlačenje i spuštanje (*engl. drag and drop*) čvorova, scena i komponenti direktno u kod [25], kao što je prikazano na slici 11.

Slika 11: Povlačenje i spuštanje scene u kod (Godot)



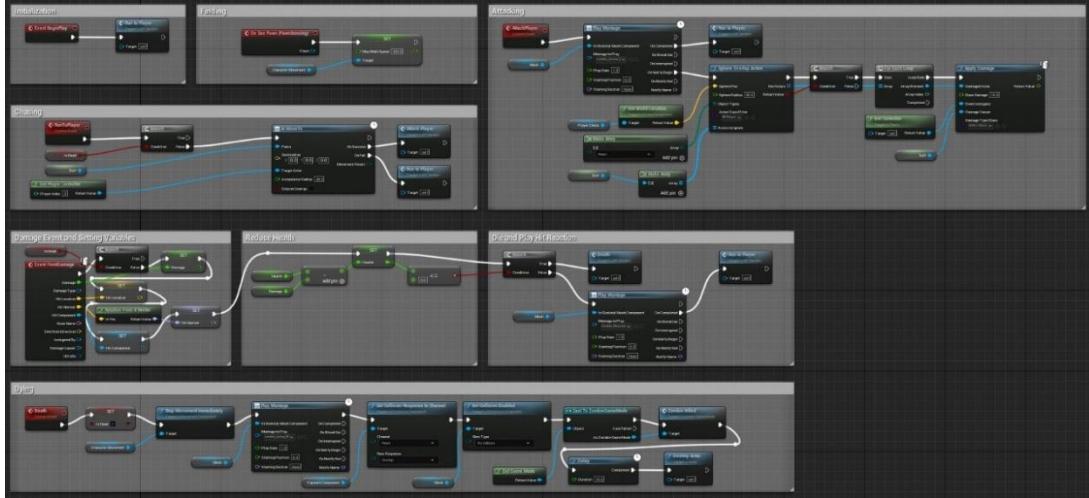
Izvor: autor

3.3 Unreal Engine – Blueprints

Vizualno skriptiranje nacrtima (*engl. Blueprints Visual Scripting*) je način kodiranja u Unreal Engine-u. Zasnovan je na konceptu korištenja sučelja temeljenog na čvorovima za stvaranje elemenata unutar Unreal uređivača (*engl. Unreal editor*) [27]. Nacrt (*engl. blueprint*) je vizualno-skriptirani dodatak videoigri (Slika 12). Povezivanjem čvorova, događaja, funkcija i

varijabli sa žicama (*engl. wires*), moguće je stvoriti složene elemente videoigre. Nacrti funkcionišu korištenjem grafikona čvorova u različite svrhe. Primjerice, za konstrukciju objekata, pojedinačnih funkcija i općih događaja videoigre koji su specifični za svaku instancu nacrtu.

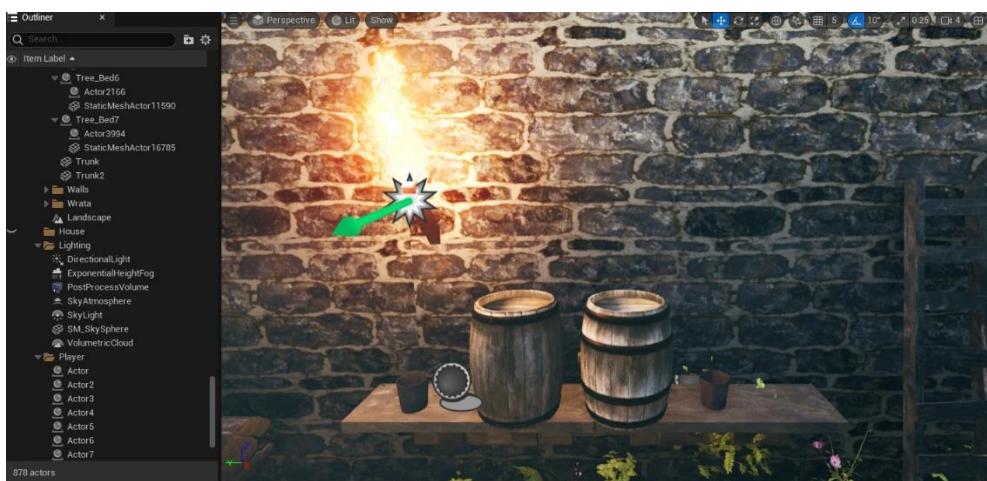
Slika 12: Primjer nacrtu u Unreal Engine-u



Izvor: autor

U Unreal Engine-u glavna gradivna komponenta je glumac (*engl. actor*), odnosno bilo koji objekt koji se može postaviti u nivo (*engl. level*). Primjerice, glumac može biti: kamera, statični objekt ili početna lokacija igrača (Slika 13). Glumci podržavaju 3D transformacije kao što su: micanje (*engl. move*), rotacija (*engl. rotate*) i skaliranje (*engl. scale*). Mogu se stvoriti (*engl. spawn*) i uništiti (*engl. destroy*) pomoću koda.

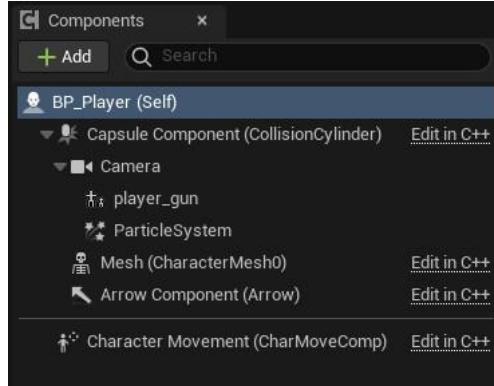
Slika 13: Prikaz Glumaca u Unreal Engine-u



Izvor: autor

Komponenta (*engl. component*) je dio funkcionalnosti koja se može dodati glumcu (Slika 14). Primjerice, kod dodavanja komponente glumcu, on može koristiti funkcionalnosti koju pruža komponenta. Primjerice, kako bi se implementirao glumac „automobil“ koji bi reprezentirao automobil iz stvarnog svijeta, potrebno mu je dodavati razne odgovarajuće komponente dok on dobro ne opisuje stvarni automobil.

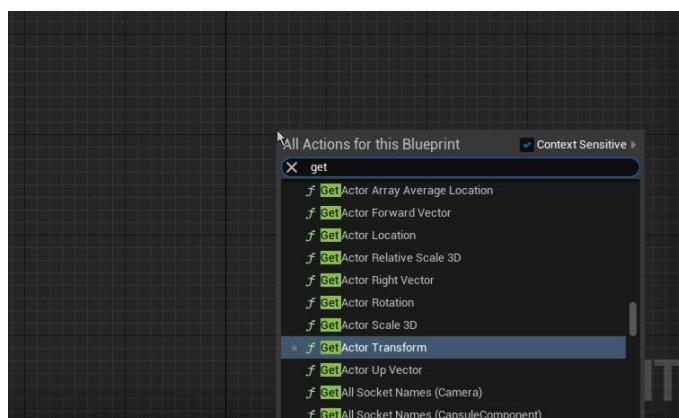
Slika 14: Primjer komponenti glumca „Igrača“ u Unreal Engine-u



Izvor: autor

Također, svakog glumca je moguće pretvoriti u nacrt, dodati u postojeći ili kombinirati više glumaca u jedan nacrt. Unutar nacrta, kod se ne piše, već se stvaraju čvorovi koji se međusobno povezuju i utječu jedan na drugog stvarajući tok koda (*engl. code flow*). Čvorovi se mogu stvoriti ili ubaciti povlačenjem i spuštanjem, kao što je prikazano na slici 15.

Slika 15: Prikaz stvaranja čvorova u Unreal Engine-u

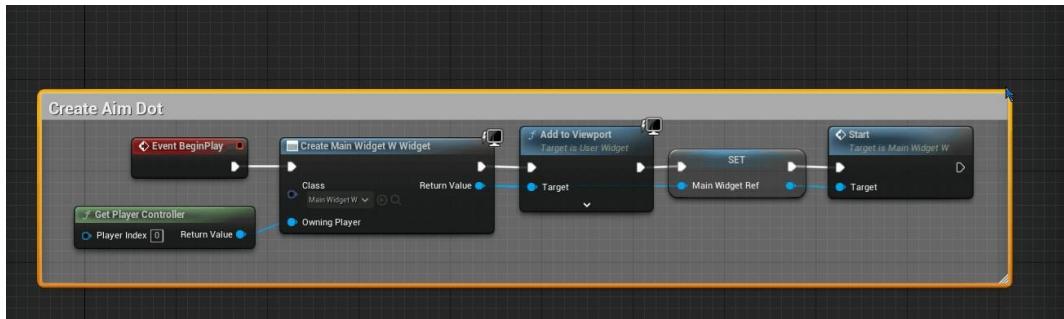


Izvor: autor

Dvije bitne funkcije tj. događaja (*engl. events*) u *nacrtu* su *Event BeginPlay* i *Event Tick*. *Event BeginPlay* događaj se pokreće jedanput, i to pri inicijalizaciji, dok se *Event Tick* poziva svaki korak. To nisu jedini događaji. Ovisno o glumcima i njihovim komponentama, može postojati puno različitih događaja za specifičnog glumca. Također se mogu specificirati i

vlastiti događaji, što je korisno za održavanje nacrt preglednim. Nacrt omogućava grupiranje i komentiranje više čvorova, kao što je prikazano na slici 16.

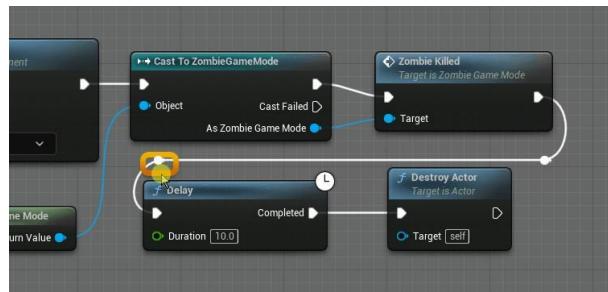
Slika 16: Komentiranje više čvorova i grupiranje u Unreal Engine-u



Izvor: autor

Također omogućava savijanje žica koje ih spajaju radi preglednosti i čitljivosti (Slika 17). U slučaju krivog spajanja ili ne-spajanja, nacrt daje upozorenje dok se greška ne ispravi [28].

Slika 17: Ispravljanje žica u Unreal Engine-u



Izvor: autor

3.4 Usporedba koda

U ovoj sekciji će se prikazati dio koda iz izrađenih videoigara, za sva tri okruženja. Prikazat će se dijelovi koda iz *Spawner* skripte koja služi za periodično stvaranje entiteta unutar scene.

3.4.1 Unity

Na slici 18 prikazan je Unity kod *Spawner* skripte.

Slika 18: Prikaz „Spawner“ skripte u Unity-u

```
4  Unity Script [1 asset reference] | 2 references
5  public class Spawner : MonoBehaviour
6  {
7      [SerializeField] GameObject[] mobs;
8      [SerializeField] private float spawnDelay;
9      [SerializeField] private int spawnAmount;
10     public int spawnerCapacity;
11     private Transform[] spawnLocations;
12     [HideInInspector] public int spawnedMobs;
13
14     Unity Message | 0 references
15     IEnumerator Start()
16     {
17         //SETUP
18         spawnLocations = new Transform[transform.childCount];
19         for (int i = 0; i < transform.childCount; i++) spawnLocations[i] = transform.GetChild(i);
20
21         //UPDATE
22         while (true)
23         {
24             if (spawnedMobs < spawnerCapacity)
25             {
26                 int s = spawnAmount;
27                 if (spawnedMobs + spawnAmount > spawnerCapacity)
28                     s = spawnerCapacity - spawnedMobs;
29                 for (int i = 0; i < s; i++)
30                 {
31                     Vector3 randomLocation = spawnLocations[Random.Range(0, spawnLocations.Length)].position;
32                     GameObject mob = Instantiate(mobs[Random.Range(0, mobs.Length)], randomLocation, new Quaternion(0, 0, 0, 0), transform);
33                     Spawned c = mob.AddComponent<Spawned>();
34                     c.addSpawner(this);
35                     spawnedMobs++;
36                 }
37             }
38             yield return new WaitForSeconds(spawnDelay);
39         }
40     }
}
```

Izvor: autor

Skripta sadrži 6 varijabli:

- *mobs*,
- *spawnDelay*,
- *spawnAmount*,
- *spawnerCapacity*,
- *spawnLocations* i
- *spawnedMobs*.

Koristi se *IEnumerator Start()* sučelje prilikom inicijalizacije što omogućava korištenje *WaitForSeconds()* funkcije koja obustavlja izvršavanje posebne „coroutine“ funkcije na zadani broj sekundi [29]. Sljedeći numerirani koraci prikazuju princip rada skripte:

1. Dohvaćanje svih mogućih lokacija koje su zapravo djeca objekta kojem je skripta pridružena.
2. Pokretanje beskonačne *while* petlje koja periodično izvršava kod stvaranja entiteta.

3. Stvaranje entiteta korištenjem *Instantiate()* funkcije koja klonira izvorni objekt i vraća klon (*engl. clone*). Funkcija prima kao argumente objekt koji će se klonirati, poziciju i roditelja.
4. Novonastalom klonu pridružuje se skripta *Spawned* te se dodaje referenca na *Spawner*.

Spawned skripta služi za smanjivanje *spawnedMobs* varijable kada se entitet uništi te poveća *spawnerCapacity* za 1, prikazano na slici 19.

Slika 19: Prikaz programskog koda „*Spawned*“ skripte u Unity-u

```
    Unity Message | 0 references
    private void OnDestroy() {
        spawner.spawnedMobs--;
        if (spawner.spawnerCapacity < 20)
            spawner.spawnerCapacity++;
    }
```

Izvor: autor

3.4.2 Godot

Slika 20 prikazuje programski kod *Spawner* skripte u Godot-u.

Slika 20: Prikaz „*Spawner*“ skripte u Godot-u

```
1 extends Spatial
2
3 onready var zombie = load("res://Prefabs/Zombie.tscn")
4 onready var spawnLocations = []
5 var spawnDelay = 3
6 var spawnAmount = 3
7 var spawnerCapacity = 4
8 var spawnedMobs = 0
9 var timer = Timer.new()
10
11 func _ready():
12     for spawn in get_children():
13         spawnLocations.push_front(spawn)
14     add_child(timer)
15     timer.set_autostart(true)
16     timer.set_wait_time(spawnDelay)
17     timer.connect("timeout", self, "spawning")
18     timer.start()
19
20 func spawning():
21     if spawnedMobs < spawnerCapacity:
22         var s = spawnAmount
23         if spawnedMobs + spawnAmount > spawnerCapacity:
24             s = spawnerCapacity - spawnedMobs
25         for i in s:
26             var mob = zombie.instance()
27             $"..".add_child(mob)
28             mob.global_transform.origin = spawnLocations[int(rand_range(0, spawnLocations.size()))].global_transform.origin
29             spawnedMobs+=1
30
```

Izvor: autor

Umjesto „coroutine“ funkcije, Godot koristi *Timer.new()* funkciju koja odbrojava do određenog intervala i emitira signal kada dosegne 0. *Timer* periodično poziva funkciju *spawning()* koja instancira entitete. Godot stvara entitete na sljedeći način:

1. Instancira se scena koja se dodaje (*var mob = zombie.instance()*).

2. Ubacivanje scene u stablo scena (`$"..add_child(mob)`), gdje se `$..` odnosi na dohvaćanje glavnog stabla
3. Ubačenom se entitetu pritom dodjeljuje nova pozicija u sceni.

Entitetu se ne dodjeljuje dodatna skripta „Spawned“, budući da on ima već u svojoj „Zombie“ skripti funkciju koja oponaša „Spawned“ skriptu.

Za uklanjanje scena ili čvorova iz stabla scena koristi se `queue_free()` funkcija koja stavlja čvor u red za brisanje na kraju trenutnog koraka (Slika 21). Kada se izbriše, bit će izbrisana i sva njegova djeca.

Slika 21: Prikaz koda „Zombie“ skripte zaslužan za ažuriranje „Spawner“ skripte u Godot-u

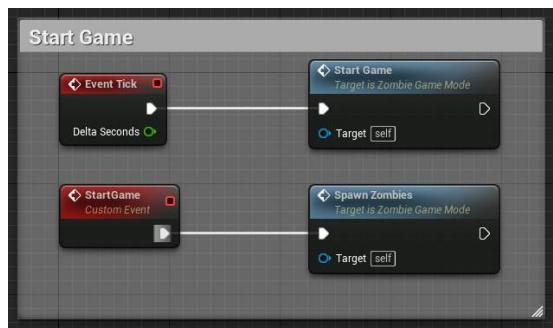
```
func remove():
    var spawner = $"/Spawner"
    spawner.spawnedMobs-=1
    if spawner.spawnerCapacity < 20:
        spawner.spawnerCapacity+=1
    queue_free()
```

Izvor: autor

3.4.3 Unreal Engine

U Unreal Engine-u nacrti su često složeniji od običnog programskog koda budući da se vizualno prikazuju programski elementi poput: varijabli, funkcija, selekcija, događaja, petlji i sl. Iz tog razloga, slike nacrtu su podijeljene u više dijelova. Na slici 22 prikazana je inicijalizacija skripte.

Slika 22. Unutar ZombieGameMode nacrtu svaki korak se pokreće stvaranje glumaca

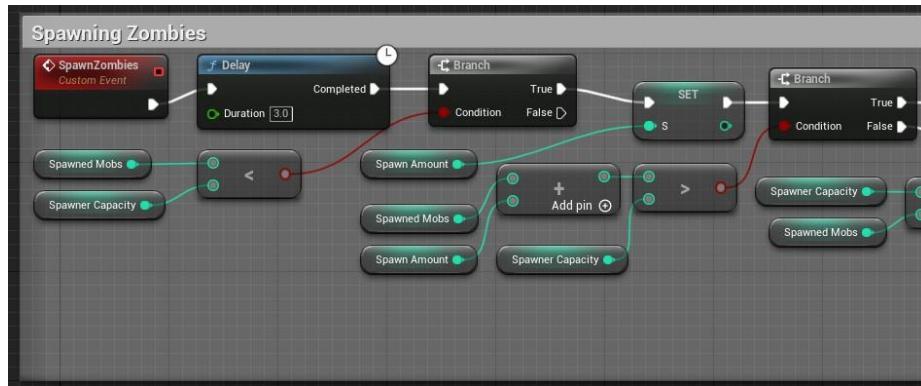


Izvor: autor

Slika 23 prikazuje `SpawnZombie` prilagođeni događaj gdje se kao „coroutine“ funkcija koristi `Delay` funkcija koja izvršava latentnu radnju s odgodom (određenim u sekundama). Ponovni poziv dok traje odbrojavanje `Delay` funkcije se zanemaruje [29]. `Branch` čvor je selekcijska izjava koja prima uvjet te na temelju njega određuje tok nacrtu. Slika 23 prikazuje čvorove povezane žicama u različitim bojama:

- Bijele žice predstavljaju tok nacrtu
- Crvene žice predstavljaju stanje
- Plave žice uglavnom predstavljaju glumce
- Ostale boje predstavljaju razno-razne vrste varijabli

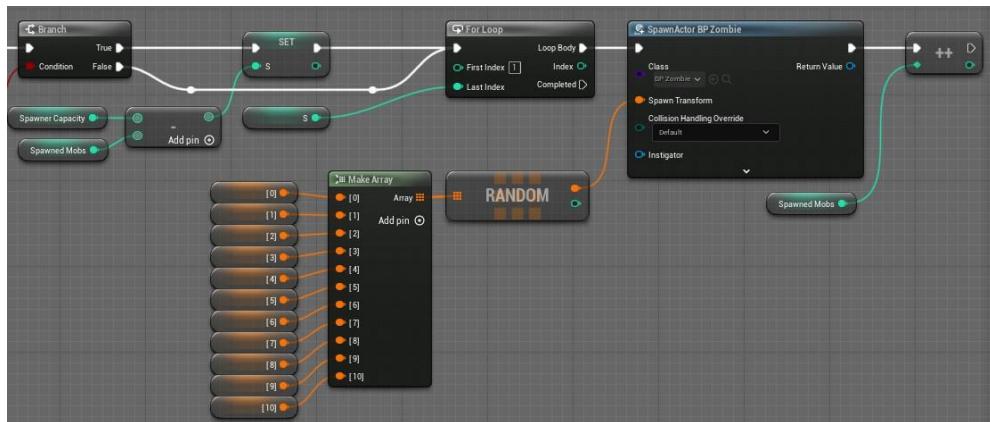
Slika 23: „Spawning Zombie“ prilagođeni događaj u Unreal Engine-u - prvi dio



Izvor: autor

Na slici 24 prikazano je stvaranje glumaca koristeći *SpawnActor* funkciju, gdje se odabire odgovarajući glumac koji će se stvoriti priključivanjem nasumične lokacije iz niza unaprijed dodijeljenih lokacija.

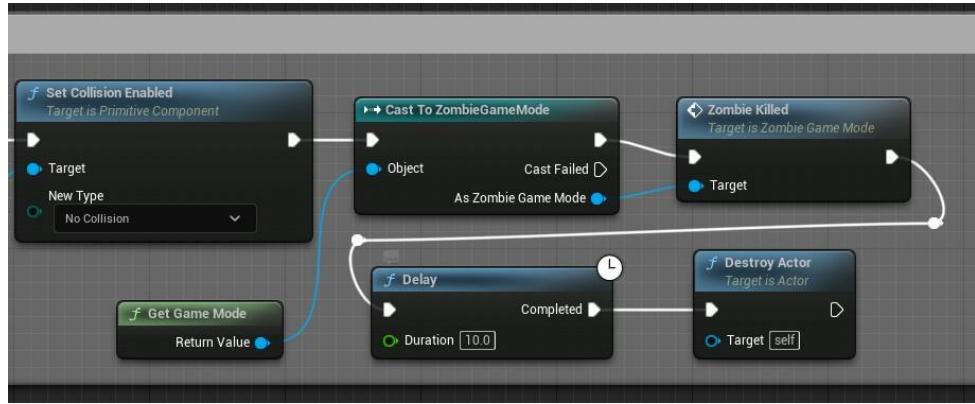
Slika 24: „Spawning Zombie“ prilagođeni događaj u Unreal Engine-u - drugi dio



Izvor: autor

Glumcu se ne dodjeljuje skripta pri stvaranju, već on sam prije uništenja poziva događaj unutar *ZombieGameMode* nacrtu (Slika 25).

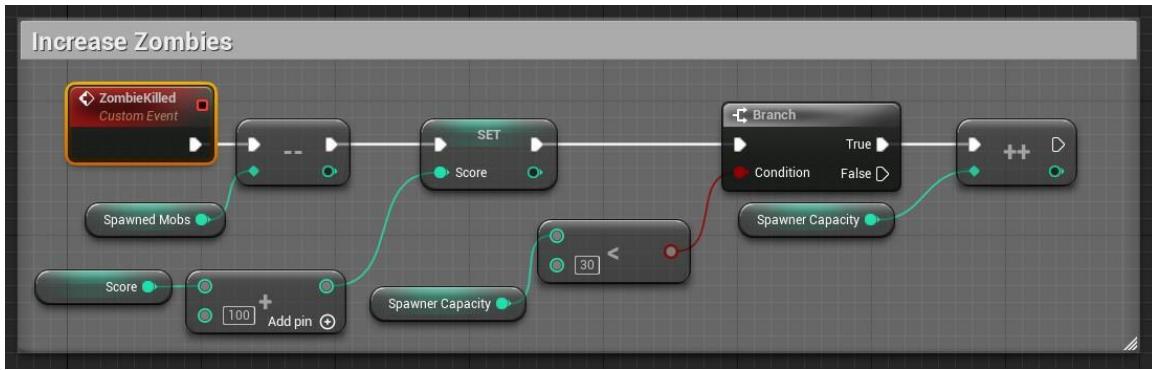
Slika 25: Pozivanje „Zombie Killed“ događaja iz „Zombie“ nacrta



Izvor: autor

Na slici 26 prikazan je *ZombieKilled* prilagođeni događaj koji se nalazi na „Zombie“ glumcu. Služi za smanjivanje *Spawned Mobs* varijable kada se glumac uništi te se time povećava *Spawner Capacity* varijabla za 1.

Slika 26: Prikaz „ZombieKilled“ prilagođenog događaja u Unreal Engine-u



Izvor: autor

4. SUČELJE PROGRAMA

Prvo što korisnik vidi pri pokretanju okruženja za izradu videoigara je programsko sučelje. Ono ima važnu ulogu za korištenje i upotrebu programa. Treba moći informirati korisnike o tome što se događa putem odgovarajućih povratnih informacija. Važno je da podržava „Undo“ i „Redo“ operacije te sadrži tipkovničke prečace. Personalizacija prilagođavanjem sadržaja i funkcionalnosti također je važna stavka kako bi si korisnici mogli osobno prilagoditi razvojno okruženje [30].

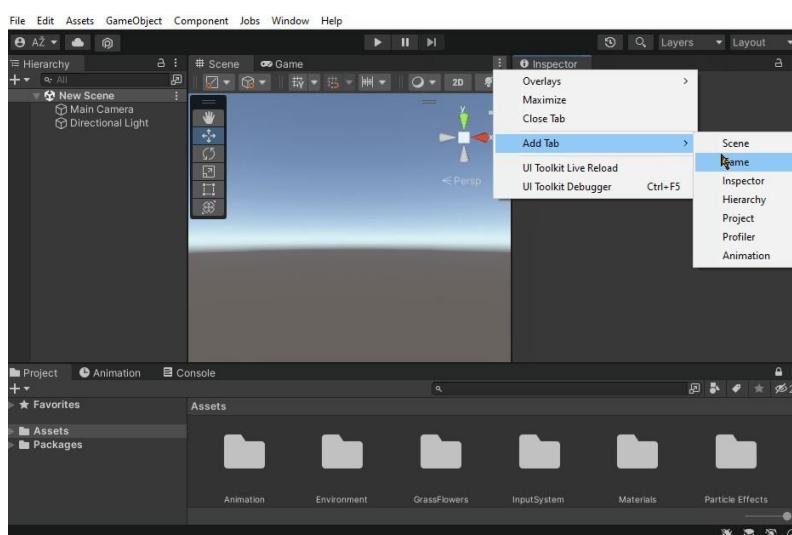
4.1 Unity

Sučelje Unity-a je veoma fleksibilno i pregledno. Podijeljeno je na 4 glavne kartice (*engl. cards*):

1. hijerarhija (*engl. hierarchy*),
2. inspektor (*engl. inspector*),
3. projekt (*engl. project*) i
4. scena/igra (*engl. scene/game*).

Cijelo sučelje se može uređivati po volji. Mogu se premještati kartice, povećavati, odvajati, dodavati nove kartice te skalirati. Također, svaka kartica se može zaključati za odabrani element (Slika 27). Zaključane kartice ostaju u istom stanju dok se ne otključaju (i dalje se mogu micati, odvajati i spajati), samo je njihov sadržaj statičan tj. ne mijenja se odabirom na druge elemente sučelja, ali dopušta mijenjanje varijabli i interakciju.

Slika 27: Prikaz sučelja u Unity-u

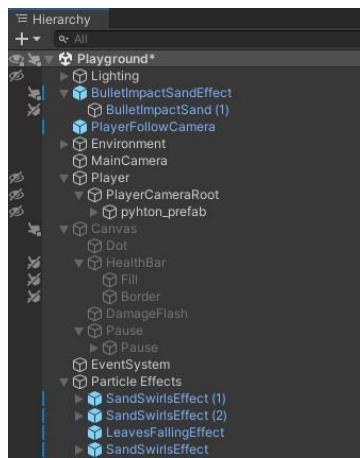


Izvor: autor

4.1.1 Hijerarhija

Hijerarhija je kartica gdje se nalaze svi objekti u sceni. Obično se nalazi s lijeve strane sučelja (Slika 28). Omogućuje pomicanje objekata na bilo koju poziciju u hijerarhiji. Poredak u hijerarhiji je bitan, pogotovo ako je objekt dijete (*engl. child*) drugog objekta. Kada neki objekt postane dijete, njegova pozicija i rotacija postaju lokalne pod roditeljem. Također, ako se uništi, onemogući ili sakrije roditelj isto tako će se sva njegova djeca uništiti, onemogućiti ili sakriti. Takav princip rada omogućava istovremeni rad s velikim brojem istovrsnih objekata. Pojedinačni objekt je moguće sakriti, no time se i dalje prikazuje pri pokretanju scene. Onemogućeni objekti su potamnjeni, dok sakriveni objekti imaju ikonu prekriženog oka. Također je moguće onemogućiti označavanje objekta u sceni (ikona prekrižene ruke). Ili-ili više objekata mogu imati isti naziv.

Slika 28: Primjer hijerarhije u Unity-u

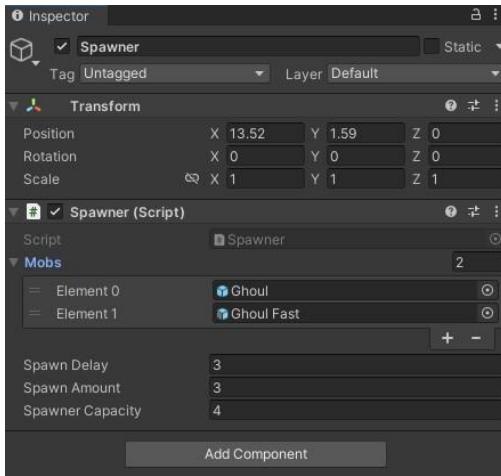


Izvor: autor

4.1.2 Inspektor

Inspektor je *kartica* u kojoj se nalaze sve komponente i skripte objekta (Slika 29). Obično se nalazi s desne strane sučelja. Svaki objekt ima komponentu *Transform* (pozicija, rotacija i proporcija). U inspektoru se mogu namjestiti ikone objekata koje će biti prikazane u sceni. Također se objekt može omogućiti i onemogućiti te mu se mogu namjestiti oznaka (*engl. tag*) i sloj (*engl. layer*) koji se korisni za implementaciju interakcije s drugim objektima. Inspektor se može urediti pomoću skripte, ili napraviti potpuno novi s različitim funkcijama i izgledom (Slika 30).

Slika 29: Primjer običnog inspektora u Unity-u



Izvor: autor

Slika 30: Primjer uređenog inspektora u Unity-u

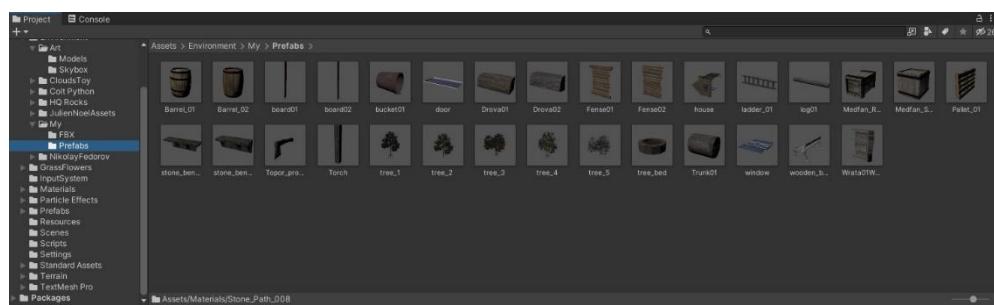


Izvor: autor

4.1.3 Projekt

Svi elementi, slike, zvuk, grafika, skripte, scene nalaze se u kartici projekt. Obično se nalazi na dnu sučelja (Slika 31). Tu se spremaju sve stvari koje se koriste u projektu pri izradi igre. Projekt ima dvije glavne mape *Assets* i *Packages*. Unutar *Assets* se nalaze svi elementi stvoreni ili ubaćeni od strane korisnika, poslagani unutar mapa, dok se unutar *Packages* nalaze svi paketi koje projekt koristi i ima ih ugrađene i spremne za korištenje. Ta mapa se uglavnom ne dira i ne mogu se unutar nje stvarati elementi. Projekt omogućuje pretraživanje upisivanjem ili odabirom vrste ili oznake. Mape i elementi unutar projekta se mogu podešavati kao „Windows“ mape.

Slika 31: Primjer „Project“ kartice u Unity-u



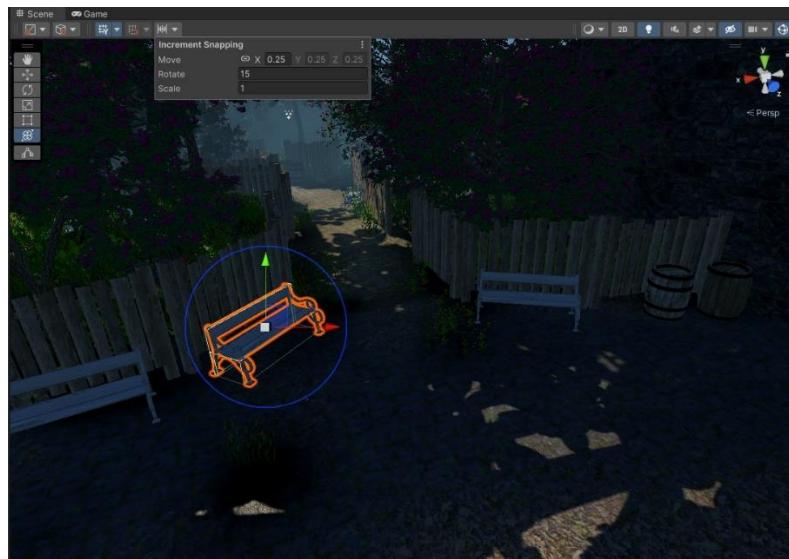
Izvor: autor

4.1.4 Scena/Igra

Scena je najveća i najbitnija kartica na sredini sučelja koja omogućava vizualni pregled videoigre i onog što se izrađuje. Unutar scene se mogu označavati jedan ili više objekata. Omogućava micanje, rotaciju, skaliranje objekta ili sve odjednom. Također, scena se dinamički

prilagođava ovisno o odabranom objektu te nudi dodatne opcije ovisno o njegovim komponentama (Slika 32).

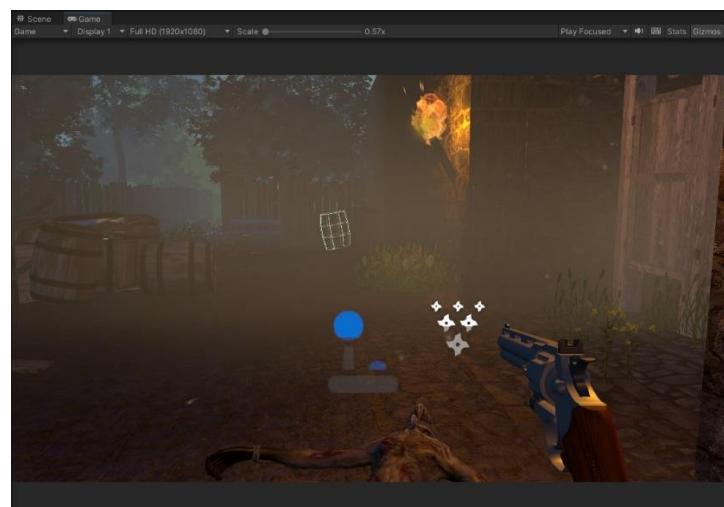
Slika 32: Primjer scene u Unity-u



Izvor: autor

U gornjem desnom kutu se nalazi „Gizmos“ alat koji se koristi za vizualno otklanjanje pogrešaka ili pomoć pri postavljanju *scene*. Moguće je mijenjati način prikaza scene, tako da se prikazuje sjena (*engl. shadow*), žičani okvir (*engl. wireframe*), njihova kombinacija te mnoge druge mogućnosti. Druga kartica slična sceni je kartica igre (*engl. game card*). U njoj je prikazano početno stanje igre (Slika 33). Pri pokretanju scene, moguće je mijenjati scenu dok je igra pokrenuta što omogućava lakše i jednostavnije otklanjanje grešaka. Također je moguće postavljanje različitih omjera ekrana i rezolucija za testiranje responzivnosti sučelja.

Slika 33: Prikaz pokrenute scene u Unity-u

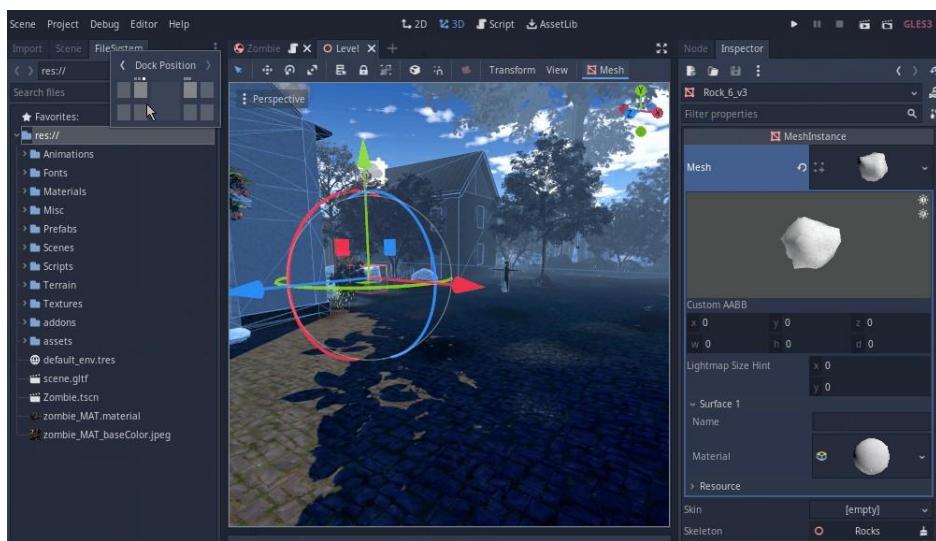


Izvor: autor

4.2 Godot

Sučelje Godot-a je relativno fleksibilno i jednostavno, podijeljeno je na kartice (*engl. tabs*). Četiri glavne kartice su: stablo scena (*engl. scene tree*), inspektor/čvor (*engl. inspector/node*), datotečni sustav (*engl. file system*) i radni prostor (*engl. workspace*). Slika 34 prikazuje primjer Godot sučelja. Za razliku od Unity-a, Godot je manje fleksibilan što se tiče uređivanja sučelja, kartice se mogu premještati, povećavati, smanjivati, skalirati ili dodavati nove. Scena se izričito nalazi na sredini i ne može se micati niti odvajati. Također, kartice se ne mogu odvajati van okvira sučelja.

Slika 34: Prikaz sučelja u Godot-u

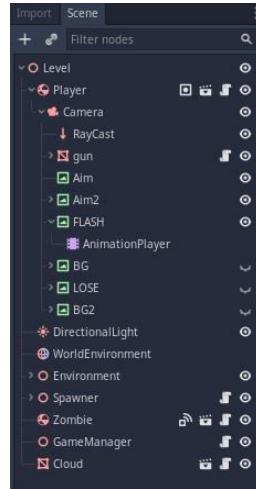


Izvor: autor

4.2.1 Stablo Scena

Kartica u kojoj se nalaze sve scene i čvorovi zove se stablo scena. Obično se nalazi s lijeve strane sučelja (Slika 35). Omogućuje pomicanje čvorova ili scena. Poredak u hijerarhiji je vrlo bitan pogotovo ako je čvor dijete drugog čvora. Kod predefiniranih objekata čvor se postavlja kao zadnje dijete budući da predefinirani objekti automatski sakrivaju svoje čvorove. Predefinirani objekt je ništa drugo nego spremljena scena u datotečnom sustavu. Poput Unity-a, kada se neki čvor postavi kao dijete, njegova pozicija i rotacija postaju lokalne s obzirom na roditelja. Također, ako se uništi ili sakrije roditelj, isto tako će se sva njegova djeca uništiti ili sakriti. Kad se pojedinačni čvor sakrije, ne prikazuje se pri pokretanju scene.

Slika 35: Primjer stabla scena u Godot-u

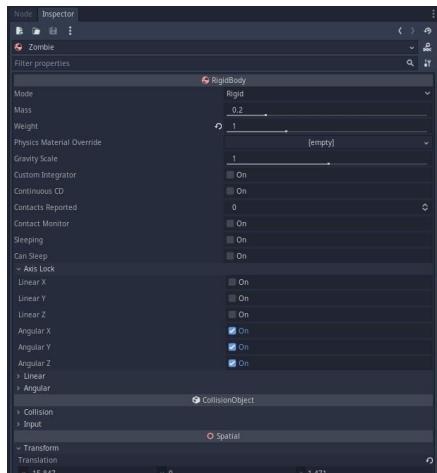


Izvor: autor

4.2.2 Inspektor/čvor

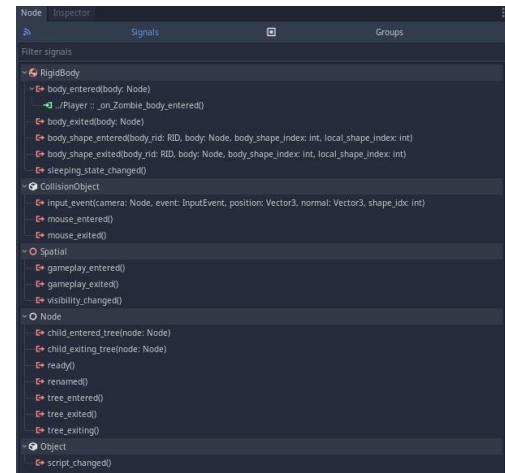
Kartica u kojoj su prikazani svi čvorovi i skripte scene. Obično se nalazi s desne strane sučelja. U inspektoru se mogu uređivati i namještati varijable čvorova, kao što je prikazano na slici 36. Osim inspektora, postoji i kartica čvor (*engl. node*) u kojem se nalaze signali (*engl. signals*) (Slika 37) [31]. U kartici čvor, pored signala se nalaze grupe koje se u Godot-u ponašaju kao oznake u Unity-u (Slika 38).

Slika 36: Primjer inspektora u Godot-u



Izvor: autor

Slika 37: Primjer signala u Godot-u



Izvor: autor

Slika 38: Primjer grupa u Godot-u

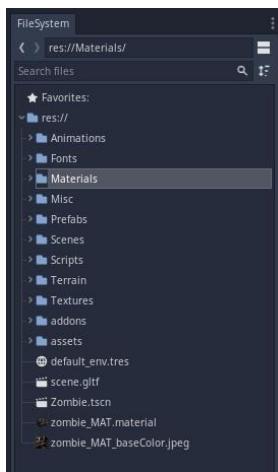


Izvor: autor

4.2.3 Datotečni sustav

Sustav datoteka upravlja na koji način se sredstva pohranjuju i kako im se pristupa. Svi elementi: slike, zvuk, grafika, skripte i scene nalaze se u kartici datotečnog sustava unutar *res://* mape (Slika 39). Obično se nalazi u donjem lijevom kutu sučelja. Svako sredstvo se može direktno referencirati iz mape povlačenjem i spuštanjem direktno u programski kod. Godot omogućuje pretraživanje upisivanjem ili sortiranjem. Mape i sredstva u datotečnom sustavu nije moguće prilagođavati kao „Windows“ mape.

Slika 39: Prikaz datotečnog sustava u Godot-u

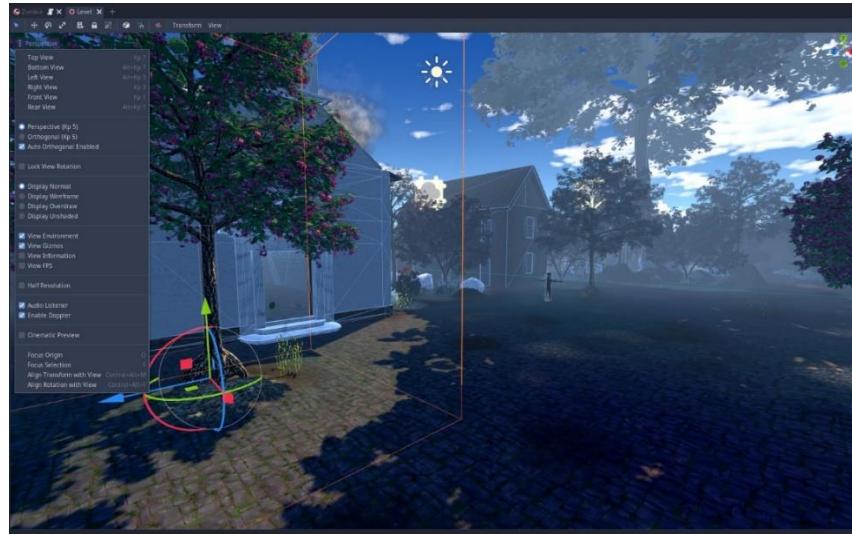


Izvor: autor

4.2.4 Radni Prostor

U radnom prostoru se nalazi prikaz glavne scene. Ova kartica se nalazi na sredini sučelja te pruža vizualni pregled videoigre i onog što se izrađuje (Slika 40). Unutar glavne scene se mogu označavati jedan ili više čvorova. Daje mogućnost micanja, rotiranja, skaliranja čvorova ili sve odjednom. Glavna scena se dinamički prilagođava ovisno o odabranom čvoru te nudi dodatne opcije ovisno o njegovim podčvorovima. U gornjem dijelu glavne scene se nalazi pogled (engl. view) koji sadrži „Gizmos“ alat za vizualno otklanjanje pogrešaka ili pomoći pri postavljanju scene. Moguće je mijenjati način prikaza scene u gornjem lijevom kutu glavne scene pritiskom na „perspective“ dugme, tako da se prikazuje sjena (engl. shadow), bez sjene (engl. no-shadow), žičani okvir (engl. wireframe) ili ostalo. Za pokretanje scene koristi se dugme „play scene“ koje se nalazi u gornjem desnom kutu sučelja. Dok je scena pokrenuta, moguće je dohvatiti „debug“ vrijednosti, međutim Godot ne omogućava interakciju sa scenom igre dok je scena pokrenuta, odnosno mijenjanje scene u uređivaču nema utjecaj na pokrenutu scenu.

Slika 40: Prikaz radnog prostora u Godot-u

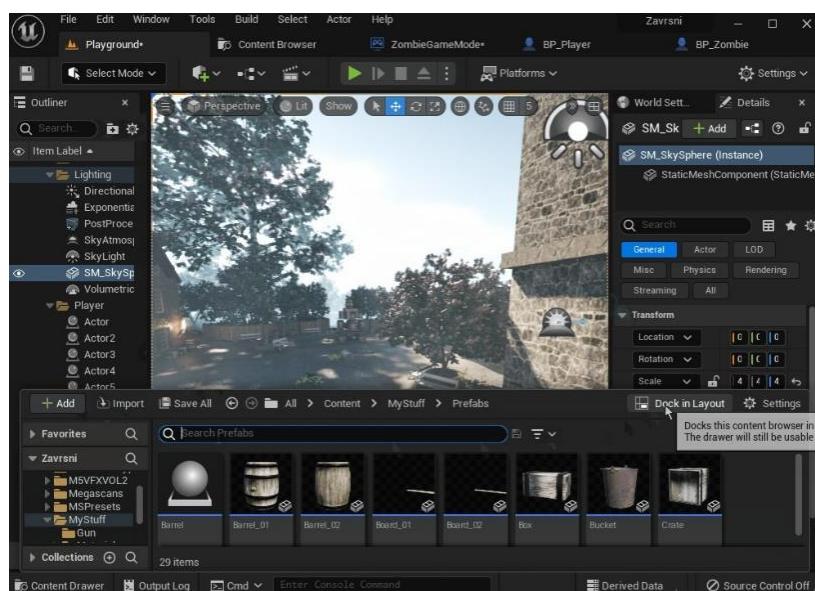


Izvor: autor

4.3 Unreal Engine

Sučelje Unreal Engine-a je fleksibilno i pregledno, podijeljeno na panele (*engl. panels*). Četiri glavna panela su obrisnik (*engl. outliner*), detalji (*engl. details*), preglednik *sadržaja* (*engl. content browser*) i prozor za prikaz (*engl. viewport*). Slika 41 prikazuje primjer Unreal Engine sučelja. Cijelo sučelje se može proizvoljno uređivati, omogućava premještanje, povećavanje, odvajanje te dodavanje dodatnih panela. Glavni prozor za prikaz je uvijek na sredini i ne može se isključiti niti pomaknuti.

Slika 41: Prikaz Unreal Engine sučelja

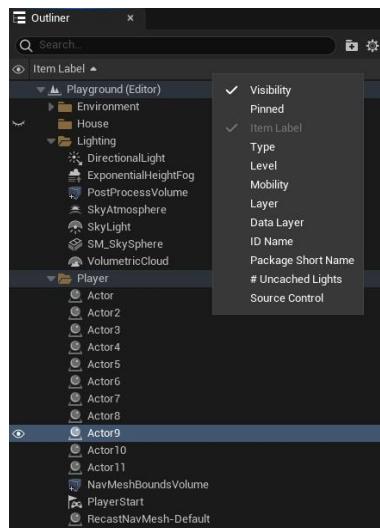


Izvor: autor

4.3.1 Obrisnik

Panel obrisnik prikazuje sve glumce unutar scene u hijerarhijskom prikazu stabla (Slika 42). Glumci se mogu birati i mijenjati izravno iz obrisnika. Također se može koristiti padajući izbornik za prikaz dodatnih stupaca koji prikazuju razine (*engl. levels*), slojeve (*engl. layers*) ili ID nazine (engl. *ID names*) [32]. U obrisniku nije bitan poredak glumaca, tj. glumci se sami poredaju abecedno. Koncept roditelja i djeteta nije jednak onima u Unity-u ili Godot-u. Samo određeni glumci mogu postati dijete drugih glumaca. Kada neki glumac postane dijete, njegova pozicija i rotacija postaju lokalne pod roditeljem. Međutim, djeca glumaca se ne unište kada im se roditelj uništi. Za lakšu organizaciju glumaca, mogu se raditi mape u obrisniku. Brisanjem mapa, djeca mape se ne brišu. Isto tako lokacija i rotacija djece mape ne ovise o mapi jer mape nemaju komponente. Dva *glumca* u obrisniku ne mogu imati isti naziv.

Slika 42: Prikaz obrisnik panele u Unreal Engine-u

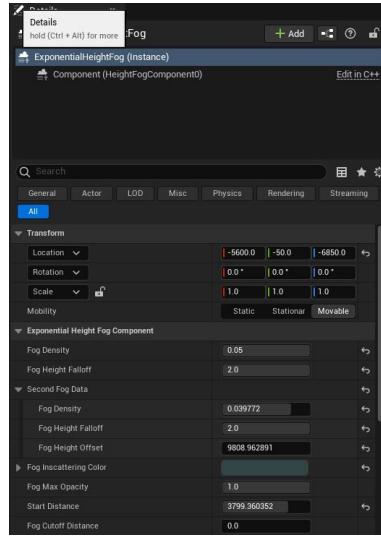


Izvor: autor

4.3.2 Detalji

Panel s detaljima omogućuje pristup uređivanju svojstava odabranih stavki unutar uređivača nacrta ili glumaca u obrisniku. Sastoji se od trake za pretraživanje za brzi pristup određenim svojstvima i općenito sadržava jednu ili više kategorija za organizaciju svojstava glumca. Također, panel s detaljima je mjesto gdje se obavljaju mnogi zadaci uređivanja nacrta, uključujući: uređivanje varijabli nacrta, promjena naziva ili tipa te dodavanje ulaza i izlaza za funkcije nacrta i događaja za odabranu komponentu (Slika 43) [33].

Slika 43: Prikaz panela detalja u Unreal Engine-u

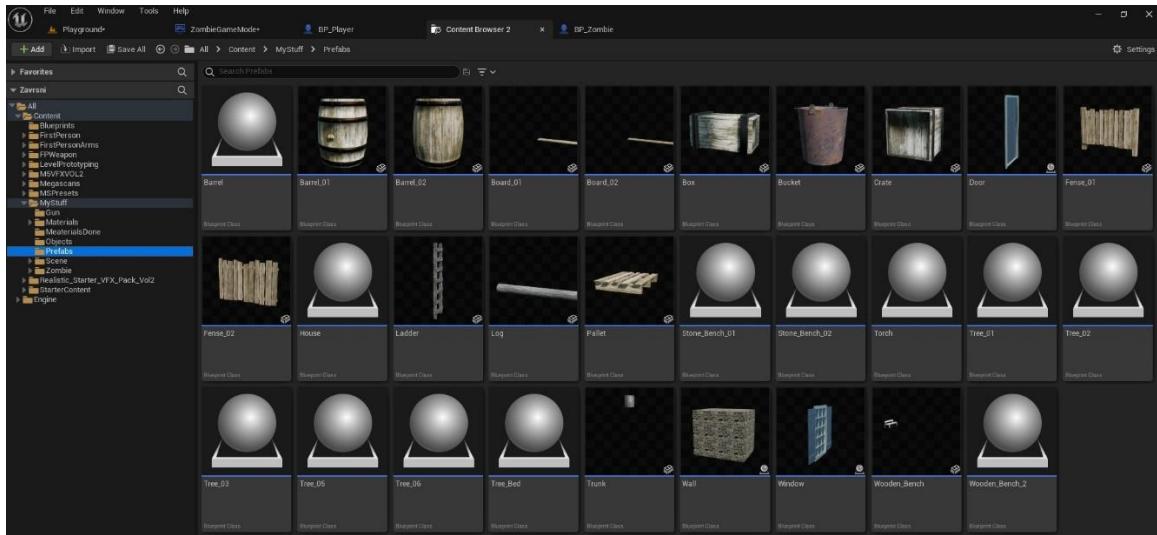


Izvor: autor

4.3.3 Preglednik Sadržaja

Preglednik sadržaja je panel koji se može prikazati pritiskom kombinacije tipki *Ctrl + Spacebar* s obzirom da se često koristi. Preglednik sadržaja primarno je područje Unreal uređivača (engl. Unreal editor) za stvaranje, uvoz, organiziranje, pregledavanje i modificiranje sadržaja (Slika 44). Štoviše, pruža mogućnost upravljanja mapama sa sadržajem, preimenovanje, premještanje, kopiranje i pregledavanje referenci. Preglednik sadržaja može pretraživati i komunicirati sa svim sredstvima u igri. Također, omogućuje pretraživanje upisivanjem ili odabirom vrste sadržaja. Mape i elementi unutar projekta se mogu prilagođavati kao „Windows“ mape.

Slika 44: Prikaz Unreal Engine panela preglednika sadržaja



Izvor: autor

4.3.4 Prozor za prikaz

Ovaj panel omogućuje vizualni pregled videoigre i onog što se izrađuje. Unutar prozora za prikaz se može označavati jedan ili više glumaca (Slika 45).

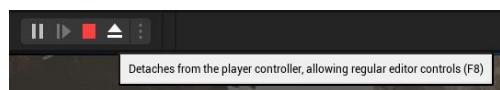
Slika 45: Prozor za prikaz u Unreal Engine-u



Izvor: autor

Daje mogućnost micanja, rotiranja te skaliranja glumaca. Međutim, nema mogućnost korištenja svih triju funkcionalnosti odjednom. Isto tako, mogućnost „letenja“ (engl. *flying*) u prozoru za prikaza ne podržava „Shift“ dugme za ubrzavanje. Micanje i rotiranje glumaca se može raditi po inkrementima koji se mogu odrediti unutar prozora za prikaz. Dodatno, glumci se mogu poravnavati na površinama drugih glumaca (engl. *surface snapping*). U gornjem lijevom kutu se nalazi izbornik (engl. *menu*) koji se koristi za vizualno otklanjanje pogrešaka ili kao pomoć pri postavljanju scene. Omogućava promjenu načina prikaza scene. Pri pokretanju nivoa, prozor za prikaz sam postaje nivo. Njime se tada može upravljati, testirati varijable na promjene te sve glumce u obrisniku (Slika 46)

Slika 46: Izlaz iz igrača u pokrenutom nivou



Izvor: autor

5. FUNKCIONALNOSTI

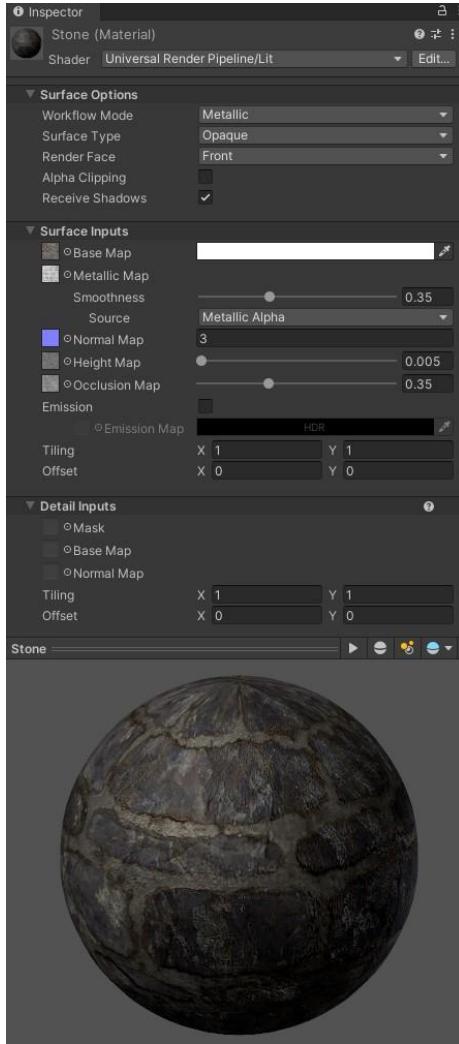
5.1 Materijali

Materijali definiraju površinska svojstva objekata u sceni. U najširem smislu, materijal se može zamisliti kao "boja" koja se nanosi na mrežu objekta kako bi se kontrolirao njegov vizualni izgled. Materijali govore mehanizmu za prikazivanje (*engl. rendering*) kako točno površina treba komunicirati sa svjetлом u sceni. Materijali definiraju svaki aspekt površine uključujući: boju, refleksiju, neravninu, prozirnost i ostale [34].

5.1.1 Unity

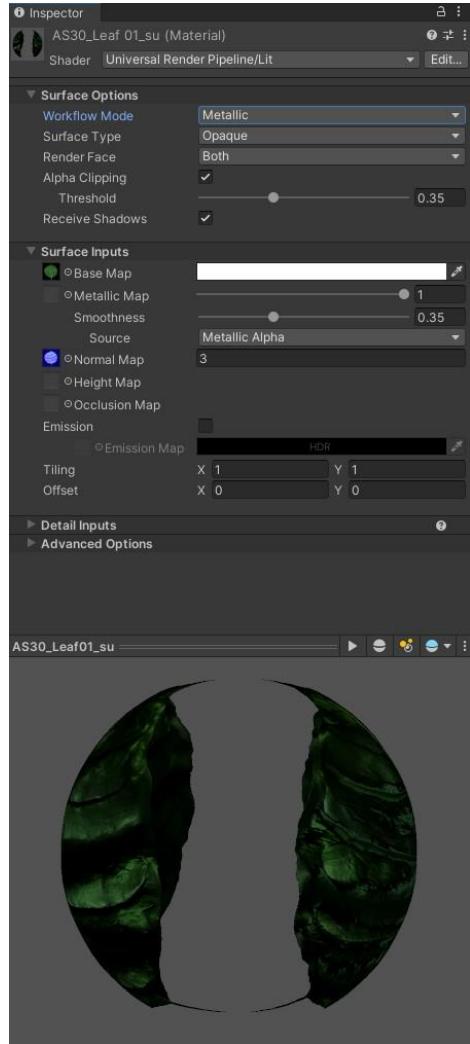
U Unity-u se koristite materijali i sjenčanje (*engl. shaders*) zajedno kako bi se definirao izgled scene. Za prikaz jednostavnog materijala poput kamenog zida, dovoljno je koristiti zadane postavke, ubaciti razne teksture zida u na to predviđeno mjesto te se poigrati s vrijednostima da bi se dobio željeni rezultat (Slika 47) [35]. Međutim, za prikaz lišća što ima prozirnu teksturu treba podesiti određene varijable. Potrebno je uključiti alfa isječak (*engl. clipping*) i podesiti prag (*engl. threshold*), što omogućava da okolina lista bude prozirna (uzimajući crnu boju kao varijablu koju treba ukloniti). Zatim, potrebno je namjestiti „render face“ na obostrano (*engl. both*), tako da lišće s obje strane modela pokazuje teksturu, kao što je prikazano na slici 48.

Slika 47: Primjer neprozirnog materijala u Unity-u



Izvor: autor

Slika 48: Primjer prozirnog materijala u Unity-u

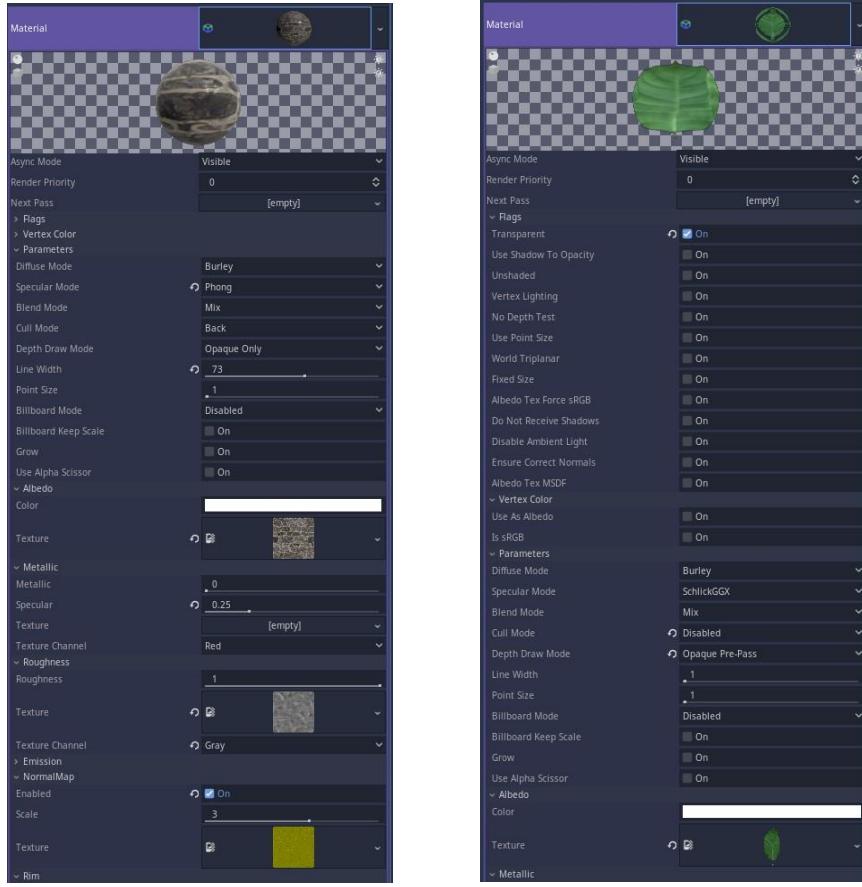


Izvor: autor

5.1.2 Godot

Kod Godot-a, često je korišten *SpatialMaterial* - zadani 3D materijal koji pruža većinu značajki koje umjetnici traže u materijalu, bez potrebe za pisanjem koda za sjenčanje (Slika 49). Alternativa je pretvorba u kod za sjenčanje ako su potrebne dodatne funkcije [36]. Za prozirni materijal – lišće, potrebno je uključiti zastavu (*engl. flag*) prozirnost (*engl. transparency*) koja omogućava prozirnost oko lišća. Također, treba namjestiti modalitet dubine crtanja (*engl. depth draw mode*) na *Opaque Pre – Pass* koji omogućuje pravilno sjenčanje lišća te onemogućiti *Cull Mode*, tako da se lišće vidi obostrano, kao što je prikazano na slici 50.

Slika 49: Primjer neprozirnog materijala u Godot-u Slika 50: Primjer prozirnog materijala u Godot-u



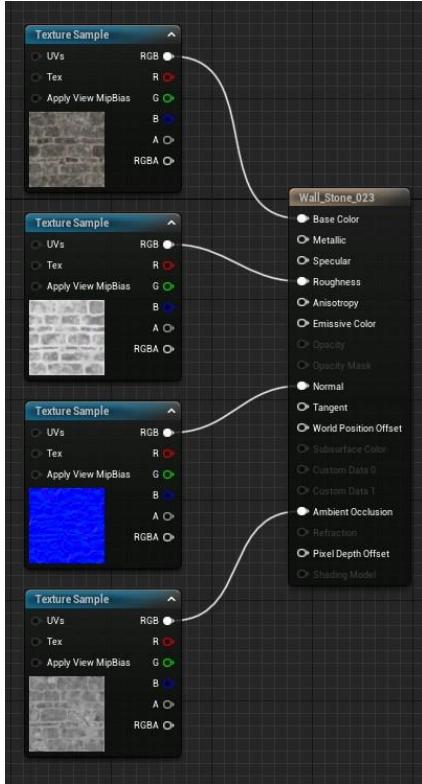
Izvor: autor

Izvor: autor

5.1.3 Unreal Engine

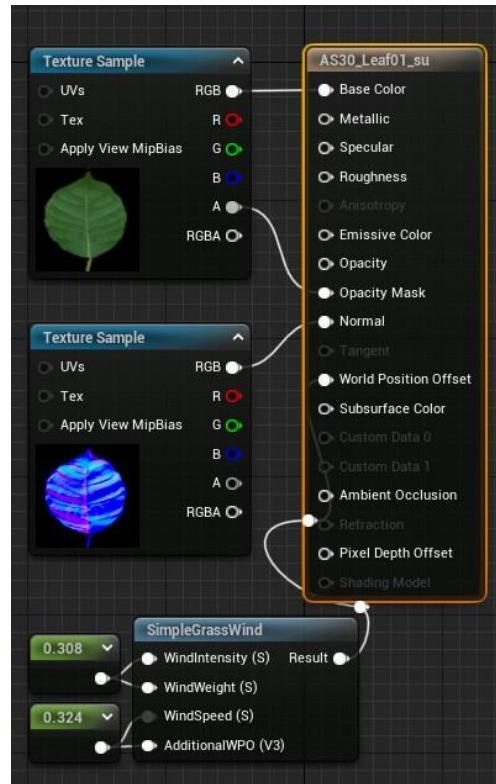
Materijal je sredstvo koje se može primijeniti na mrežu glumca za kontrolu vizualnog izgleda scene. Unreal Engine za materijale koristi *vizualno skriptiranje* tj. *material graph*, gdje se sve spaja pomoću čvorova i žica [34]. Primjerice, za teksturu kamena dovoljno je koristiti osnovne postavke (Slika 51). Međutim, prozirni materijali zahtijevaju više posla. Treba u pojedinostima materijala podesiti *Blend Mode* na *Masked* što će omogućiti prozirnost oko lišća. Zatim, model sjenčanja (*engl. shading model*) treba postaviti na dvostrano lišće (*engl. two sided foliage*) i omogućiti varijablu obostranost (*engl. two sided*), čime se tekstura lišća vidi s obje strane (Slika 53). Konačno, treba povezati alfa kanal (*engl. alpha channel*) obojene teksture lišća na *Opacity Mask*, kao što je prikazano na slici 52.

Slika 51: Primjer neprozirnog materija Unreal Engine



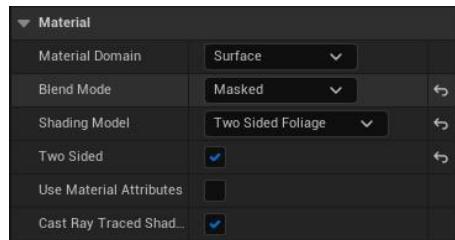
Izvor: autor

Slika 52: Primjer prozirnog materija Unreal Engine



Izvor: autor

Slika 53: Primjer pojedinosti prozirnog materijala Unreal Engine



Izvor: autor

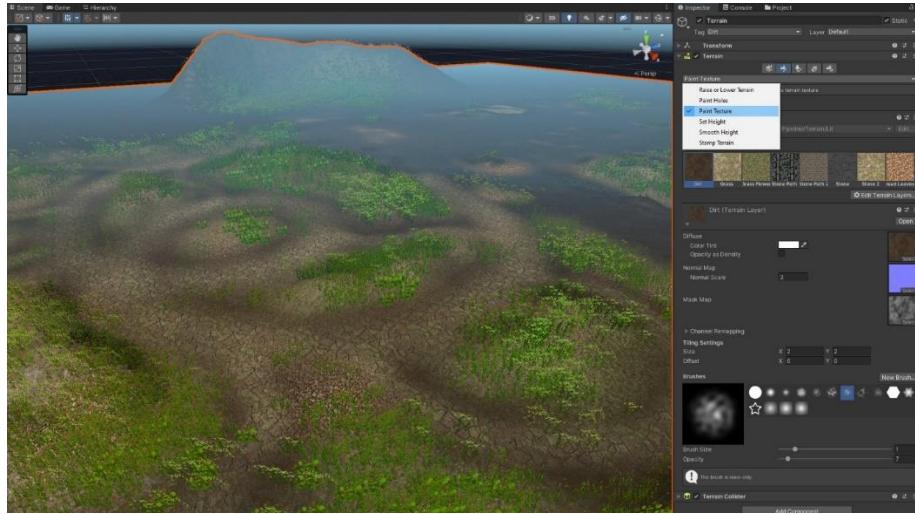
5.2 Teren

Teren (*engl. terrain*) omogućava dodavanje pejzaža u videoigru. Moguće je stvaranje masivnih svjetova koristeći alate za uređivanje terena. Teren bi trebao omogućiti bojanje/dodavanje tekstura/materija na podlogu, oblikovati podlogu kao glinu (pr. povećavanje i snižavanje visine) te dodavanje primjerice razno-raznih biljaka, kamenja, cvijeća ili bilo kojih drugih objekata.

5.2.1 Unity

Unity Editor uključuje ugrađeni skup *Terrain* značajki koje omogućuju uređivanje svijeta videoigre. *Terrain* je već ugrađen u Unity-u te ga je moguće jednostavno dodati kao zasebni objekt. U inspektoru se kod bojanja tekstura dodaju slojevi materijala (*engl. material layers*) za bojanje. Za svaki sloj materijala se može lako odrediti veličina tekstuure (Slika 54) [37].

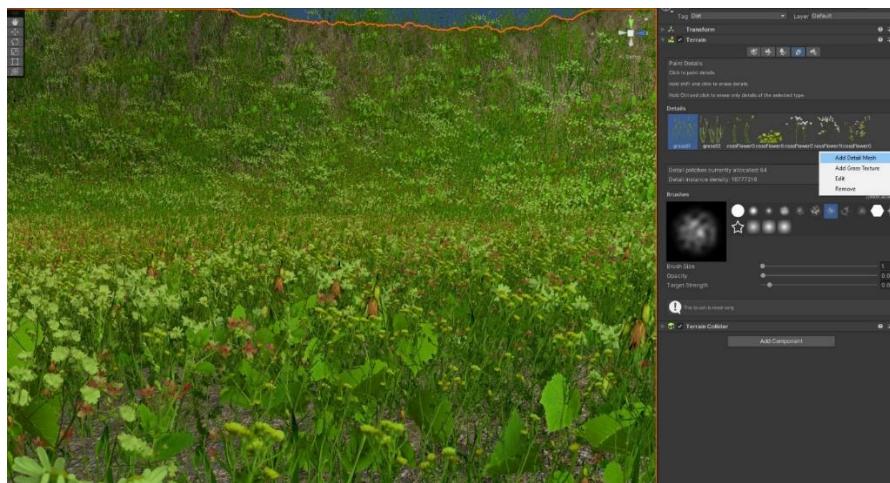
Slika 54: Prikaz *Terrain* objekta u Unity-u



Izvor: autor

Moguće je mijenjati visinu terena te ukrašavati krajolik vegetacijom dodavanjem tekstura ili *mesh objekata*, kao što je prikazano na slici 55. Broj tekstura prilikom bojanja terena nema ograničenja. Također, *Terrain* omogućava stvaranje sudarača (*engl. collider*) s kojima objekti mogu vršiti interakciju.

Slika 55: Prikaz ukrašavanja krajolika u Unity-u

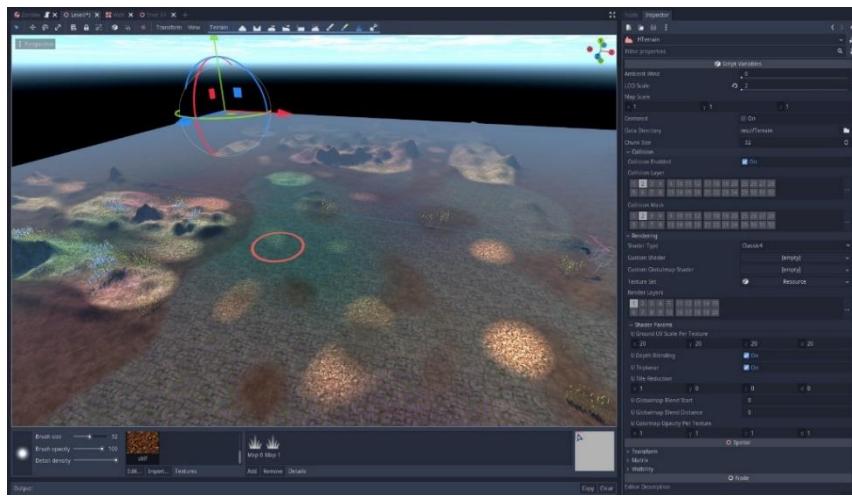


Izvor: autor

5.2.2 Godot

Godot nema ugrađeni sustav za obradu terena, već se koristi „Heightmap terrain 3D“ alat iz AssetLibrary-a (Slika 56). U usporedbi s Unity terenom, teren sustav Godot-a ima više ograničenja te stvara probleme u interakciji s čvorovima. Jedno od ograničenja je što je sveukupno moguće koristiti samo četiri teksture za bojanje terena. Dalje, interakcija terena s čvorovima može uzrokovati pad performansi (*engl. framerate drop*). Također, izgled, oblik te samo postavljanje objekata na teren je nepraktično.

Slika 56: Primjer Godot terena

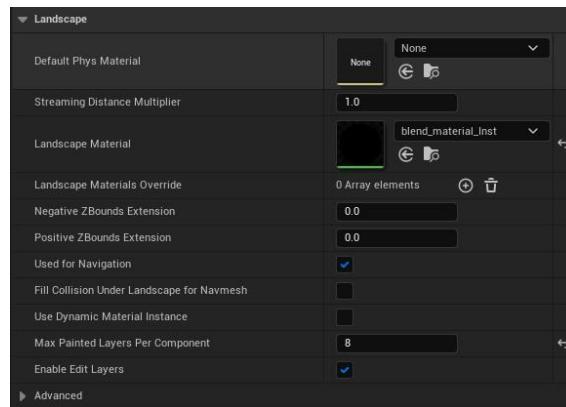


Izvor: autor

5.2.3 Unreal Engine

Pejzažni sustav unutar Unreal Engine-a zbirka je alata koja omogućuje stvaranje ekspanzivnih vanjskih okruženja [38]. Nakon inicijalizacije i postavljanja terena u obrisniku, treba napraviti pejzažni materijal (*engl. landscape material*), kao što se može vidjeti na slici 57.

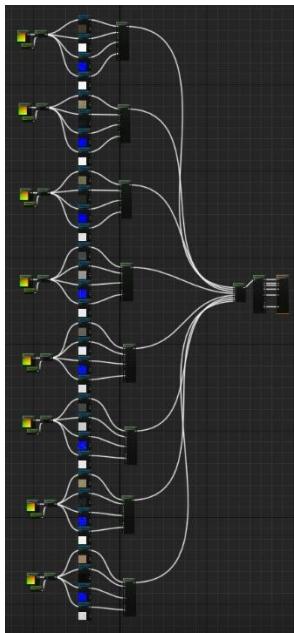
Slika 57: Postavljanje pejzažnog materijala u Unreal Engine-u



Izvor: autor

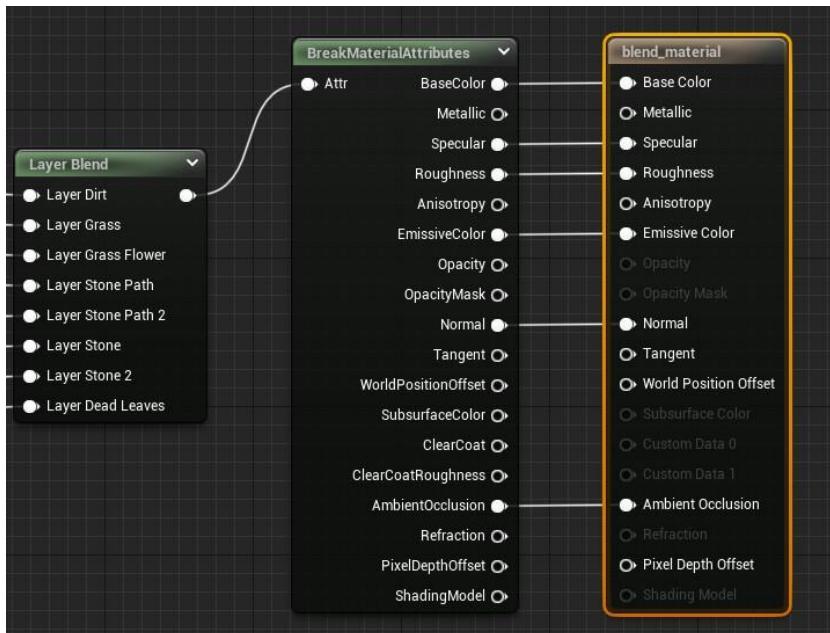
U pejzažnom materijalu se zapravo nalaze obični materijali koji se zajedno spajaju kako bi se mogli koristiti za bojanje terena (Slike 58 i 59).

Slika 58: Prikaz pejzažnog materijala (smanjeni prikaz)



Izvor: autor

Slika 59: Prikaz pejzažnog materijala (uvećani prikaz)



Izvor: autor

Važno je da sve teksture koje se koriste u pejzažnom materijalu imaju postavljenu varijablu „izvor uzorka“ (engl. *sampler source*) na *Shared: Wrap* što će omogućiti korištenje više od 3 materijala po terenu bez da se teksture izobliče. Koristeći „Quixel bridge“, mogu se dohvatiti realistični modeli biljaka te koristiti kao dekoracija koje sustav terena automatski prepozna (Slika 60).

Slika 60: Primjer ukrašavanja terena u Unreal Engine-u



Izvor: autor

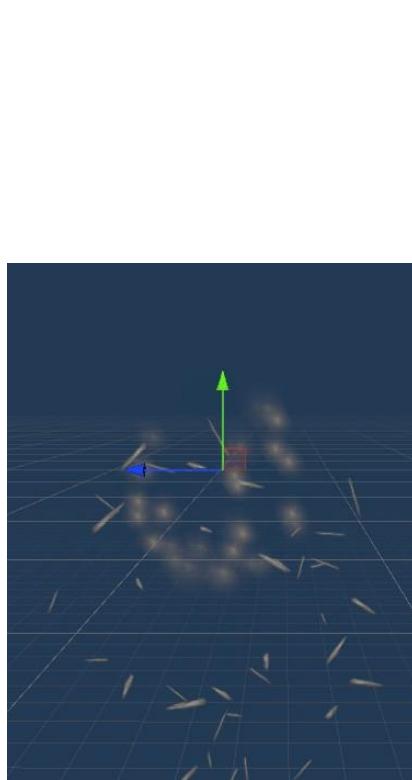
5.3 Sustav čestica

Sustav čestica (*engl. particle system*) je tehnika u fizici igara (*engl. game physics*), grafici pokreta (*engl. motion graphics*) i računalnoj grafici (*engl. computer graphics*) koja koristi mnoge sitne teksture (*engl. sprites*), 3D modele ili druge grafičke objekte za simulaciju određenih vrsta "nejasnih" fenomena [39].

5.3.1 Unity

Sustav čestica u Unity-u je sustav u kojem se mogu simulirati razni efekti poput: tekućine, dima, oblaka, plamena, magičnih čarolija i cijeli niz drugih efekata. Slika 61 prikazuje vizualni efekt sustava čestica u sceni, dok je na slici 62 pokazan inspektor sustava čestica.

Slika 61: Primjer sustava čestica u Unity sceni



Izvor: autor

Slika 62: Primjer sustava čestica u Unity inspektoru



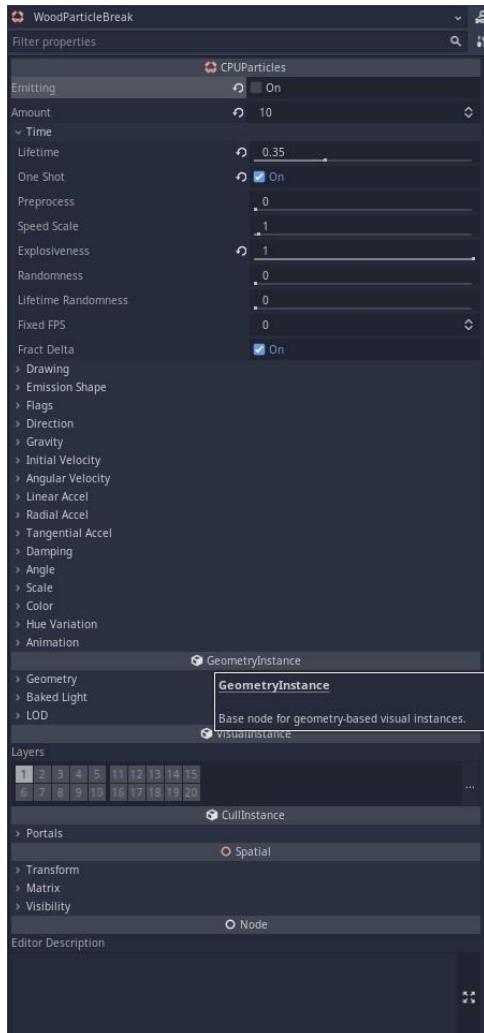
Izvor: autor

5.3.2 Godot

Godot sadrži 3D čvor čestica koji se koristi za stvaranje raznih sustava čestica i efekata. Taj čvor ima *emitter* koji generira proizvoljni broj čestica odabranom brzinom. Godot sustav čestica je sličan Unity sustavu čestica, međutim manjka u određenim stavkama. Godot sustav čestica ne može stvoriti i rotirati čestice u sve tri dimenzije. Također, nema funkcionalnosti poput: nasumičnog šuma (*engl. noise*), tragova (*engl. trails*), okidača (*engl. triggers*), sudara (*engl. collision*), svjetla i dr. Bez obzira na sve to, i dalje je moguće napraviti kvalitetne i

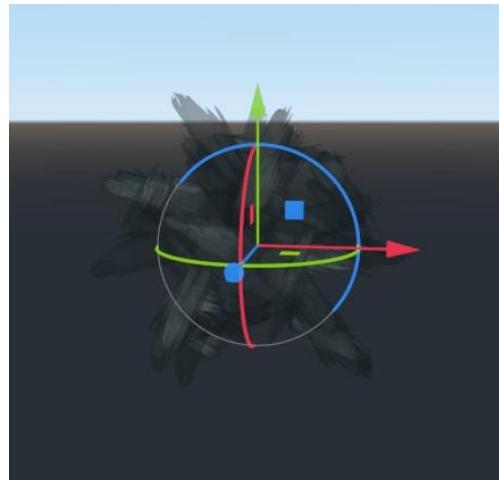
učinkovite sustave čestica koji oponašaju određene efekte. Na slici 64 prikazan je vizualni efekt sustava čestica u sceni, dok je na slici 63 pokazan inspektor sustava čestica.

Slika 63: Primjer sustava čestica u Godot inspektoru



Izvor: autor

Slika 64: Primjer sustava čestica u Godot sceni

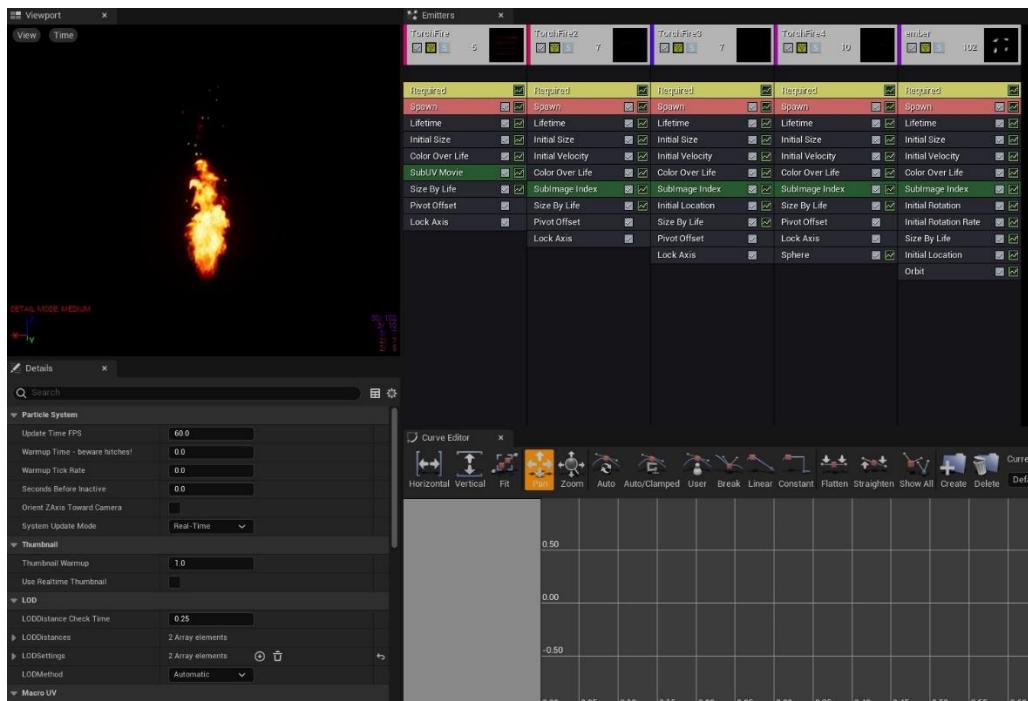


Izvor: autor

5.3.3 Unreal Engine

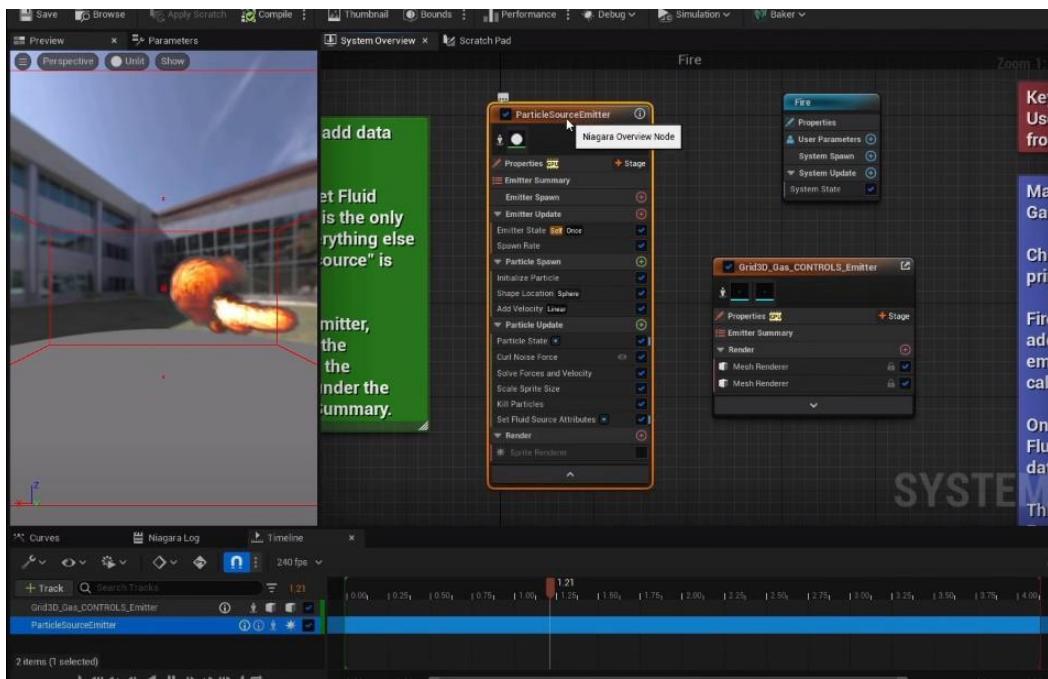
Unreal Engine-ov sustav za stvaranje efekata čestica zove se *Cascade*, te je prikazan na slici 65. Ovaj sustav omogućuje stvaranje modularnih efekata i jednostavnu kontrolu ponašanja čestica. Također sadrži i *Niagara*, VFX (*visual effects*) sustav čestica sljedeće generacije (Slika 66). Pomoću *Niagare*, tehnički umjetnik može samostalno izraditi dodatne funkcionalnosti, bez pomoći programera.

Slika 65: Primjer Unreal Engine Cascade sustava čestica



Izvor: autor

Slika 66: Primjer Unreal Engine Niagara sustava čestica



Izvor: autor

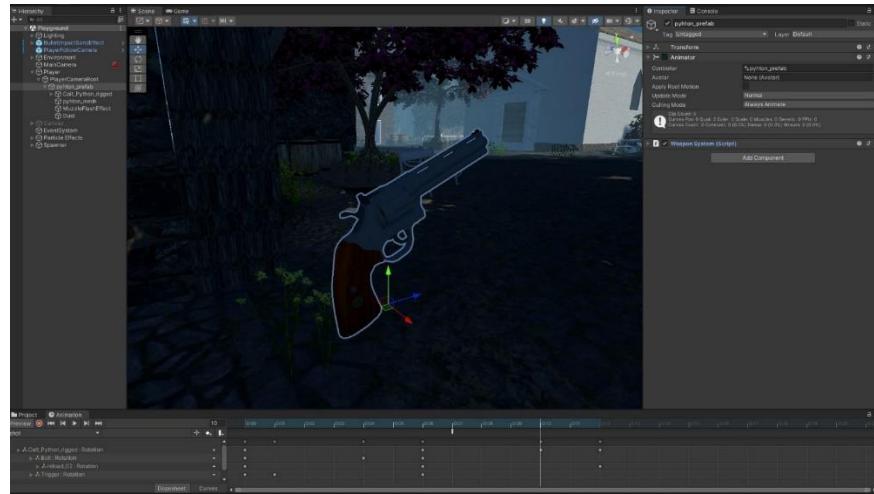
5.4 Animacije

Jedno od važnijih funkcionalnosti razvojnih okruženja su sustavi za animiranje (*engl. animation systems*). Oni omogućavaju izradu animacija koje obogaćuju igračeve iskustvo te čine igricu realističnjom i ugodnijom.

5.4.1 Unity

Animacije u Unity-u omogućavaju potpunu kontrolu nad objektima. Svaki objekt koji se nalazi u sceni moguće je animirati, kao i sve njegove komponente i varijable (Slika 67). Također, animacije mogu pozivati događaje. Primjerice, tako se može implementirati dodavanje zvuka pri svakom koraku (kod animacije hodanja lika) [40].

Slika 67: Primjer animacije revolvera s događajem u Unity-u

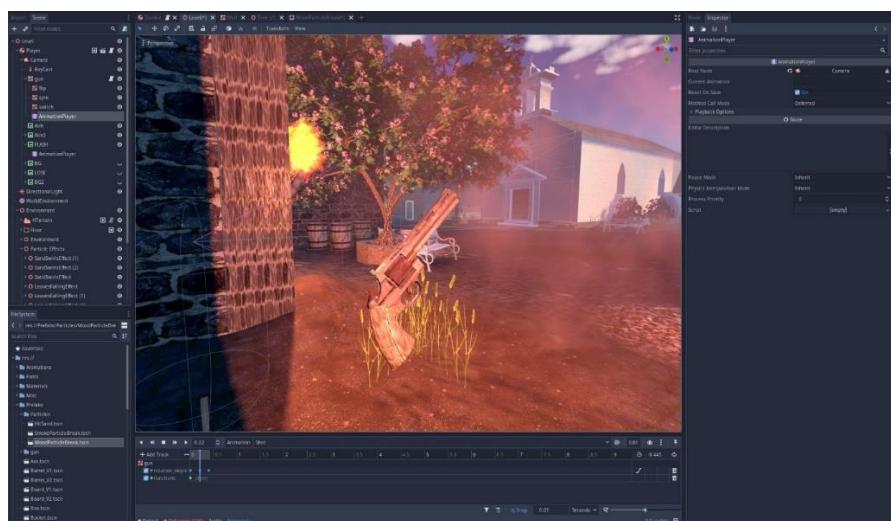


Izvor: autor

5.4.2 Godot

U Godotu se može animirati sve što je dostupno u inspektoru, poput: transformacija čvorova, tekstura, elemenata korisničkog sučelja, čestica, materijali itd. Također se mogu mijenjati vrijednosti varijabli skripti na čvorovima. Isto tako, moguće je pozivati vanjske funkcije pomoću animacija kao što je slučaj kod Unity-a (Slika 68) [41].

Slika 68: Godot primjer animacije



Izvor: autor

Razlika između Unity-a i Godot-a je u tome što Unity ima mogućnost snimanja animacija. Kada se pokrene snimanje, snimaju se sve promjene nad objektima u sceni. Nasuprot tomu, u Godot-u se animira pomoću ključeva (*engl. keys*) bez snimanja, već je potrebno u vremenskoj liniji (*engl. timeline*) postaviti ključeve koji mijenjaju vrijednosti čvorova.

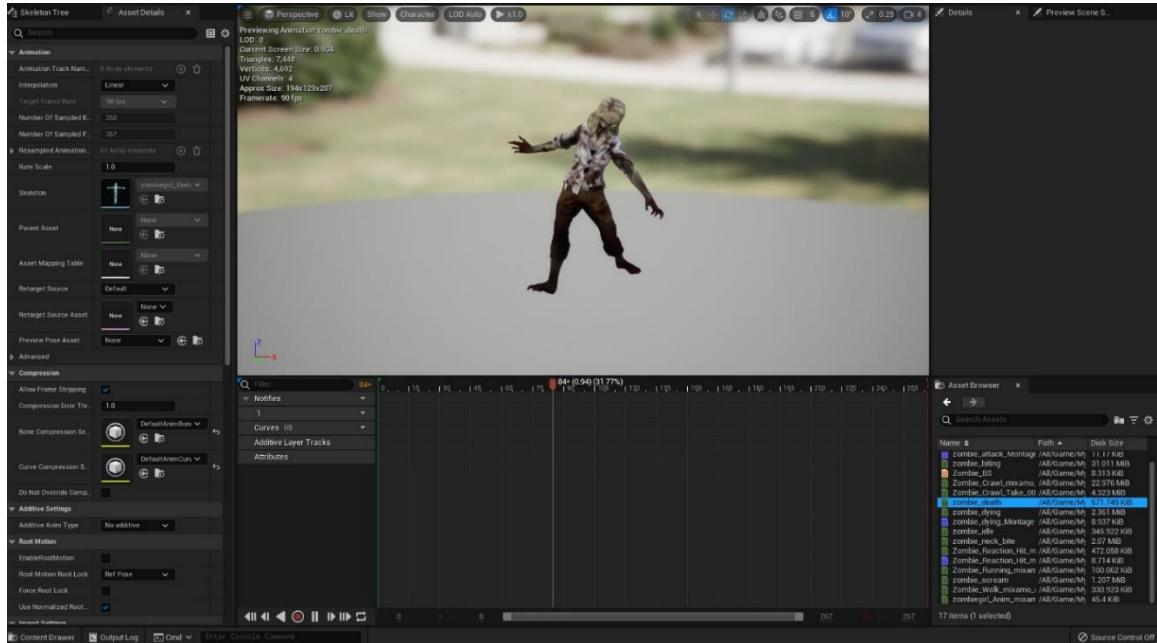
5.4.3 Unreal Engine

Unreal Engine nema spomenute mogućnosti mijenjanja varijabli i glumaca na jednostavan način kao što to pružaju Godot i Unity. Međutim, Unreal Engine pruža široki izbor alata za upravljanje likovima, stvaranje kinematografskog sadržaja i animiranje izravno u programu.

Sa sustavom animacije *Skeletal Mesh* može se upravljati uvezenim likovima, kosturima i animacijama. On se može proširiti kako bi se stvorila interaktivna animacija igranja pomoću različitih značajki kao što su: *Blend Spaces*, *Animation Blueprints* i *State Machines*. Slika 69 prikazuje primjer animacije, dok slika 70 prikazuje kod koji poziva tu animaciju.

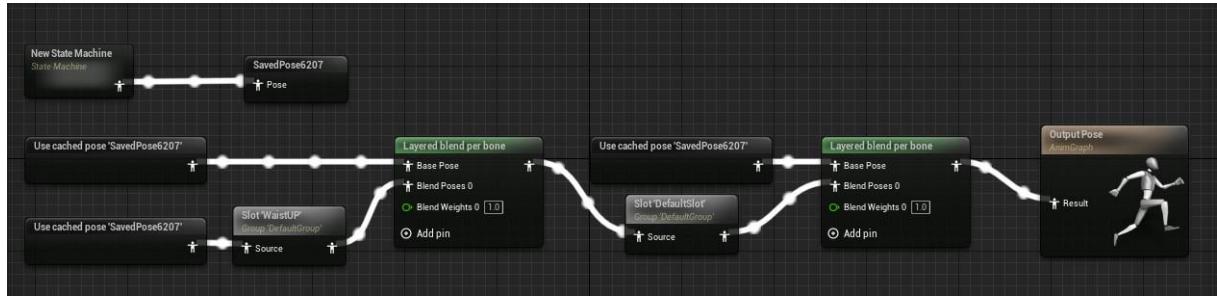
Filmske sekvene mogu se stvoriti pomoću alata *Sequencer*, kojim se mogu animirati kamere, likovi i efekti. Animiranje likova može se izvršiti pomoću *Control Rig-a*, koji je ujedno i Unreal Engine-ov ugrađeni alat.

Slika 69: Primjer animacije u Unreal Engine-u



Izvor: autor

Slika 70: Prikaz koda koji poziva animaciju u Unreal Engine-u



Izvor: autor

5.5 Korisničko Sučelje

U većini videoigara žele se prenijeti neke informacije igračima putem korisničkog sučelja (*engl. user interface – UI*) videoigre. Korisničko sučelje može se sastojati od elemenata kao što su: glavni izbornik, izbornik za zaustavljanje igre, heads-up display (HUD) elementi (pr. zdravlje, energija ili pomoćne upute igračima što učiniti u određenoj situaciji) [42].

5.5.1 Unity

Unity nudi tri sučelja koji se mogu koristiti za stvaranje korisničkih sučelja:

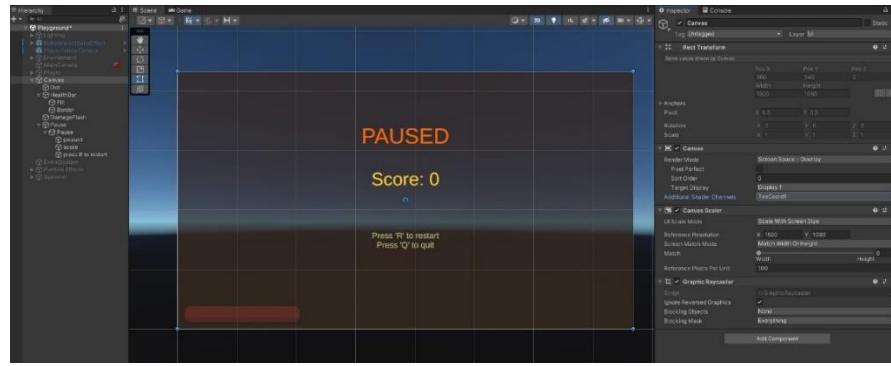
- *UI Toolkit*,
- *Unity Graphical User Interface (uGUI)* i
- *Immediate Mode Graphical User Interface (IMGUI)*.

UI Toolkit je najnoviji UI sustav u Unity-u. Osmišljen je za optimizaciju performansi na svim platformama i temelji se na standardnim web tehnologijama.

uGUI je stariji sustav korisničkog sučelja temeljen na objektima koji se mogu koristiti za razvoj korisničkog sučelja za igre i aplikacije (Slika 71). Koristite se komponente objekata za pozicioniranje i oblikovanje korisničkog sučelja unutar scene igre.

IMGUI je *UI Toolkit* vođen kodom koji koristi *OnGUI* funkciju. Često je korištena biblioteka za crtanje i uređivanje korisničkim sučeljima. *IMGUI* se koristiti za izradu prilagođenih inspektora i proširenja za Unity Editor [43].

Slika 71: Primjer izrade korisničkog sučelja koristeći uGUI u Unity-u

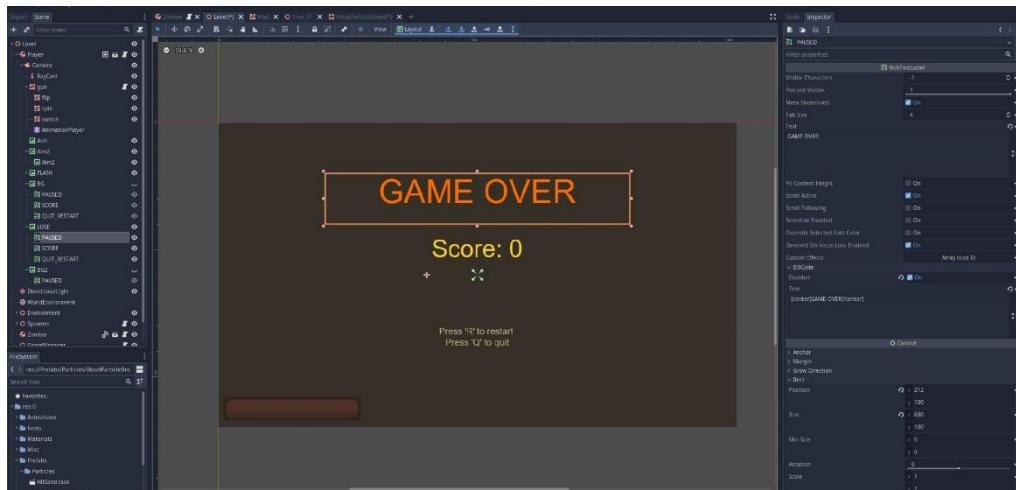


Izvor: autor

5.5.2 Godot

Kod Godot-a se na glavnom radnom prostoru u 2D sceni se izrađuje korisničko sučelje slaganjem čvorova u stablu scena, kao što je prikazano na slici 72.

Slika 72: Primjer izrade korisničkog sučelja u Godot-u

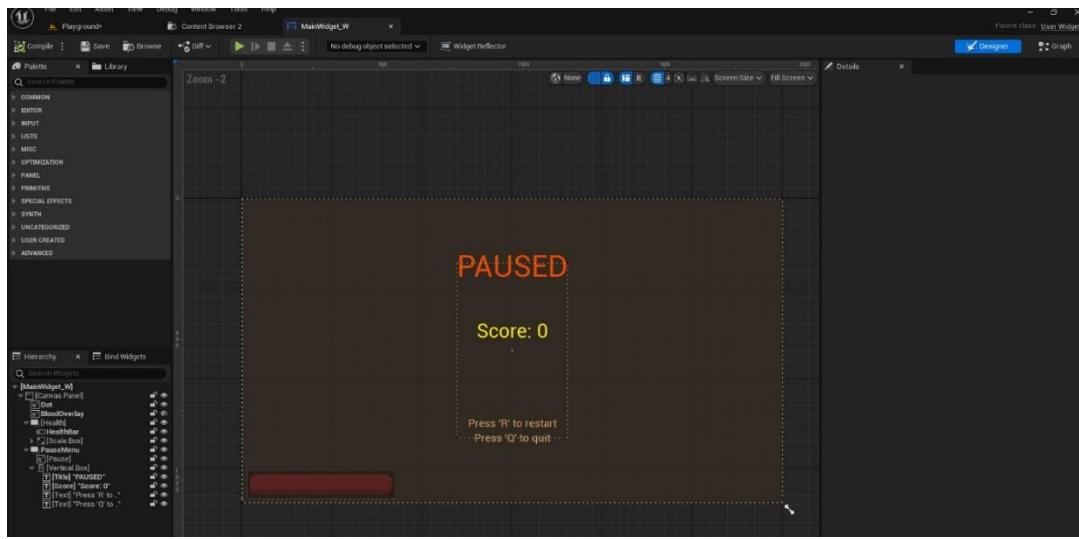


Izvor: autor

5.5.3 Unreal Engine

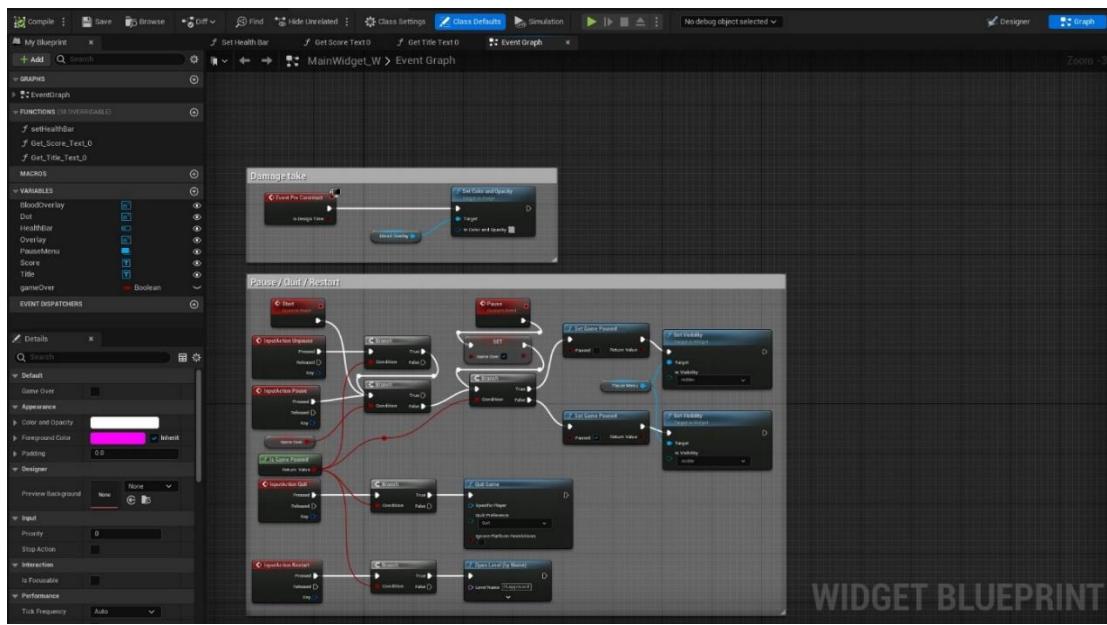
Unreal Motion Graphics (UMG) omogućava izradu *Widget Blueprint*-a za upravljanje prikazom UI elemenata u projektu. *Widget Blueprint* koristi *Blueprint Visual Scripting* za dizajn, kao i za implementaciju funkcionalnosti za elemente korisničkog sučelja [42]. Na slici 73 su prikazani elementi sučelja u hijerarhiji, dok je na slici 74 prikazan *Widget Blueprint*.

Slika 73: Primjer izrade korisničkog sučelja u Unreal Engine-u



Izvor: autor

Slika 74: Primjer funkcionalnosti korisničkog sučelja u Unreal Engine-u



Izvor: autor

6. ISCRTAVANJE SLIKE

Iscrtavanje slike (*engl. rendering*) je proces generiranja fotorealistične slike iz 2D ili 3D modela pomoću računalnog programa. Rezultirajuća slika naziva se prikaz (*engl. render*). Više modela može se definirati u datoteci scene koja sadrži objekte u strogo definiranom jeziku ili strukturi podataka. Datoteka scene sadrži sljedeće informacije koje opisuju virtualnu scenu [44]:

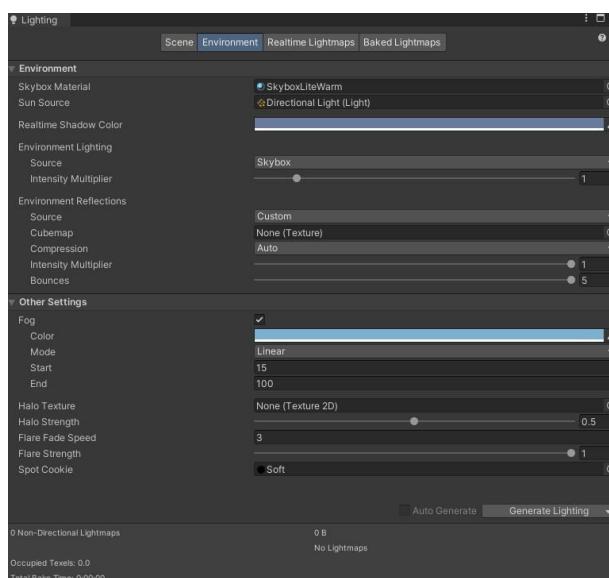
- Geometrija,
- Točka gledišta,
- Tekstura i
- Osvjetljenje.

3D iscrtavanje slike (*engl. 3d rendering*) je proces stvaranja slike na temelju trodimenzionalnih podataka pohranjenih na računalu. Također se smatra kreativnim procesom, slično fotografiji ili kinematografiji [45].

6.1 Unity

Universal Render Pipeline (URP) ugrađeni je sustav koji se može koristiti za kodiranje i izradu sjenčanja [46]. U izrađenoj videoigri izrađeno je okruženje (*engl. environment*) tako da paše uz postavljene objekte na sceni (Slike 75 i 76).

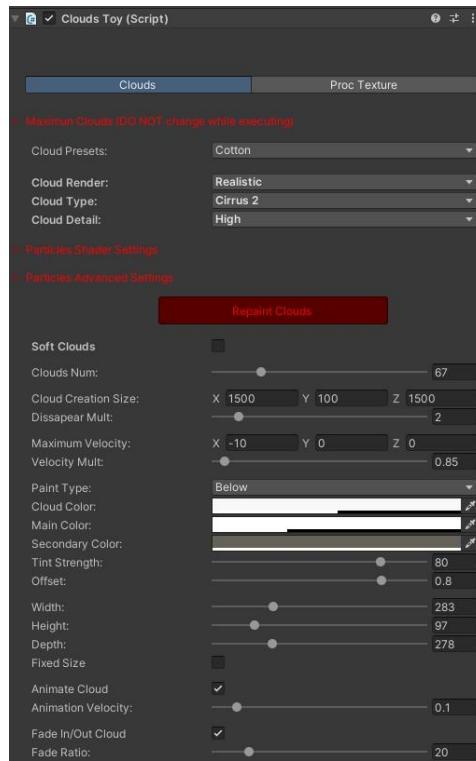
Slika 75: Postavke okruženja u Unity-u



Izvor: autor

Za izradu oblaka koristi se paket *CloudsToy* koji omogućava generiranje razno-raznih oblika oblaka.

Slika 76: Unity paket - *Clouds Toy*

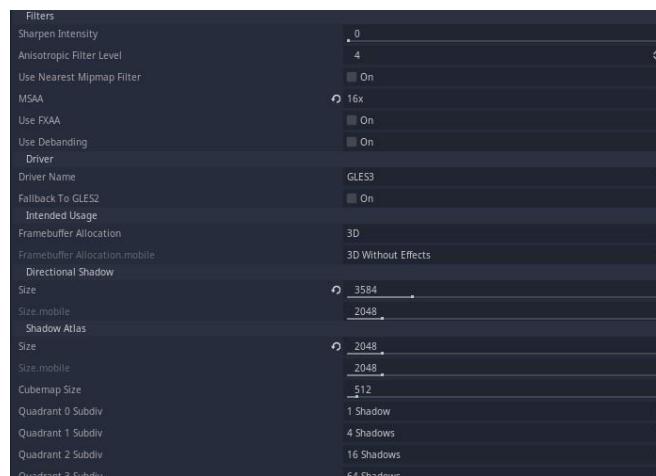


Izvor: autor

6.2 Godot

Godot nudi *GLES2* i *GLES3* koji se koriste za iscrtavanje grafike temeljene na OpenGL-u. Unutar *WorldEnvironment* čvora su za potrebe izrađene videoigre podešene varijable tako da pašu uz postavljene objekte na sceni (Slike 77 i 78).

Slika 77: Dodatne postavke za namještanje kvalitete okruženja u Godot-u



Izvor: autor

Slika 78: Godot WorldEnvironment



Izvor: autor

Za generiranje oblaka koristi se *Cloud* čvor sličan Unity *Clouds Toy*-u (Slika 79).

Slika 79: Podešavanje oblaka u Godot-u

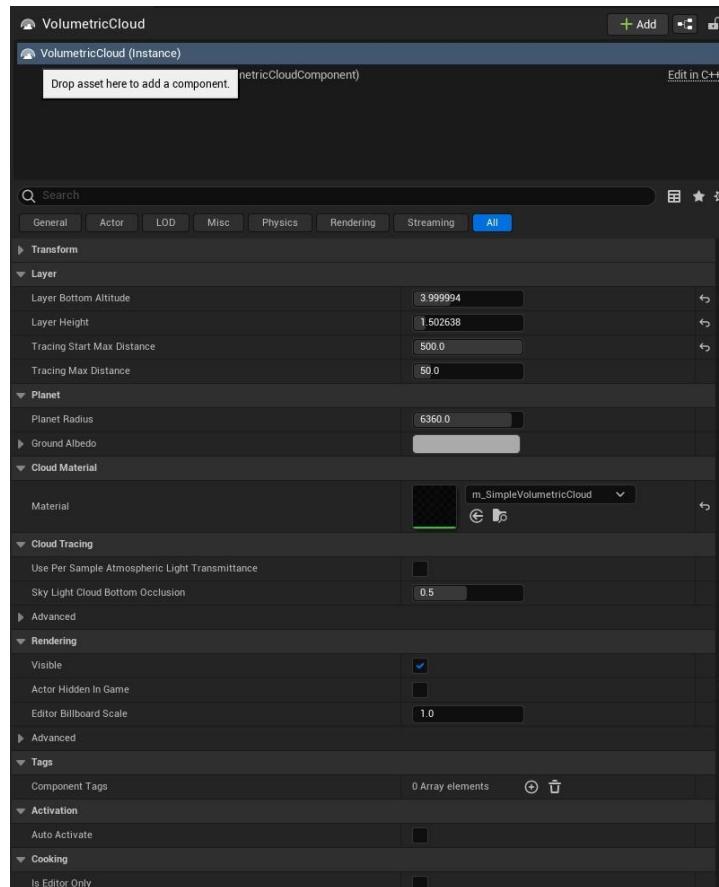


Izvor: autor

6.3 Unreal Engine

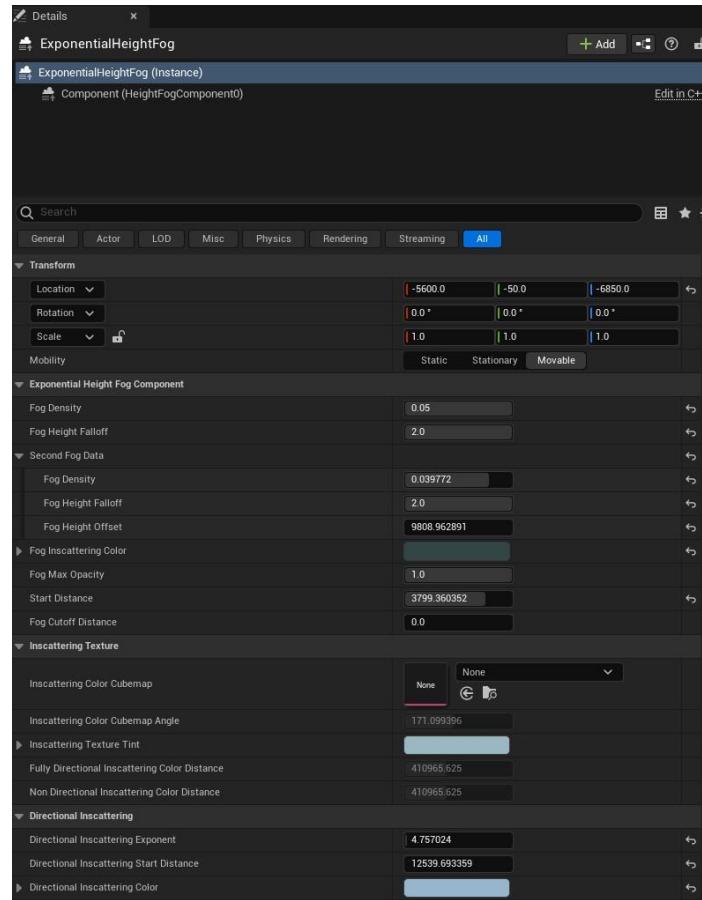
Unreal Engine koristi *DirectX 11* i *DirectX 12* tehnologije koje omogućavaju: odgođeno sjenčanje, globalno osvjetljenje, naknadnu obradu (*engl. post processing*), kao i graphics processing unit (GPU) simulaciju čestica koja koristi vektorska polja (Slike 80 i 81) [47].

Slika 80: Podešavanje oblaka u Unreal Engine-u



Izvor: autor

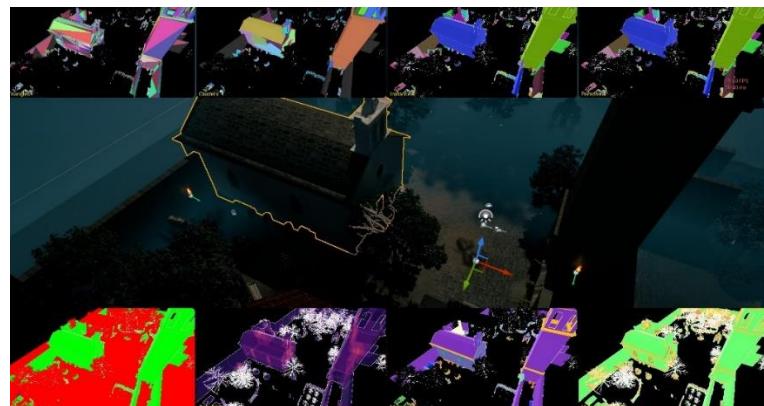
Slika 81: Podešavanje magle u Unreal Engine-u



Izvor: autor

Unreal Engine 5 pruža tehnologiju *Lumen* - potpuno dinamički sustav globalnog osvjetljenja i odraza, kao i *Nanite* - virtualizirani geometrijski sustav koji koristi novu tehnologiju prikazivanja (*engl. rendering technology*) za prikazivanje (*engl. rendering*) detalja skale piksela (*engl. pixels*) i velikog broja objekata (Slika 82) [47].

Slika 82: Unreal Engine Nanite Vizualizacija



Izvor: autor

7. TABLICA USPOREDBE

Tablica 1 prikazuje usporedbu triju okruženja (Unity, Godot, Unreal Engine) po predmetima vrednovanja tako da sadrži sažetke najbitnijih razlika i sličnosti.

Tablica 1: Usporedba okruženja po predmetima vrednovanja

Predmet vrednovanja	Unity	Godot	Unreal Engine
Programiranje	<ul style="list-style-type: none"> • C# objektno orijentiran programski jezik, • vanjsko integrirano razvojno okruženje (IDE), • prikaz referenciranih funkcija 	<ul style="list-style-type: none"> • GDScript programski jezik • „drag and drop“ čvorova unutar koda • unutarnje integrirano razvojno okruženje 	<ul style="list-style-type: none"> • Vizualno skriptiranje nacrtima • „drag and drop“ varijabli unutar koda • unutarnje integrirano razvojno okruženje
Sučelje programa	Hijerarhija	<ul style="list-style-type: none"> • Higerarhija – sadrži objekte • Poredak objekata je bitan • Veze roditelj-dijete • Dva objekta mogu imati isti naziv 	<ul style="list-style-type: none"> • Stablo scena – sadrži scene i čvorove • Poredak čvorova i scena je bitan • Veze roditelj-dijete • Dva čvora ili dvije scene ne mogu imati isti naziv
	Inspektor	<ul style="list-style-type: none"> • Inspektor – prikaz komponenti i skripti objekata • Dodjeljivanje oznaka i slojeva • Mogućnost uređivanja inspektora 	<ul style="list-style-type: none"> • Inspektor – prikaz varijabli čvorova • Korištenje signala • Dodjeljivanje grupe
	Projekt	<ul style="list-style-type: none"> • Projekt – prikaz mapa i datoteka koje se koriste pri izradi projekta • Dvije glavne mape: „Assets“ i „Packages“ • Mogućnost pretraživanja i skaliranja veličine ikona datoteka/mapa 	<ul style="list-style-type: none"> • Preglednik sadržaja – stvaranje, uvoz, organiziranje, pregledavanje i izmjenjivanje sadržaja • Mogućnost pretraživanja i skaliranja veličine ikona datoteka/mapa • Prikaz kombinacijom tipki „Ctrl + Spacebar“

Funkcionalnosti	Scena/Igra	<ul style="list-style-type: none"> • Scena – vizualni pregled igre dok se uređuje • Brzo „letenje“ pomoću tipke „Shift“ • Mogućnost micanja, rotiranja, skaliranja objekata odjednom • Igra – prikaz početnog stanja igre • Mogućnost mijenjanja scene dok je igra pokrenuta 	<ul style="list-style-type: none"> • Radni prostor – prikaz glavne scene. Omogućava vizualni pregled videoigre i onog što se izrađuje • Brzo „letenje“ pomoću tipke „Shift“ • Mogućnost micanja, rotiranja, skaliranja čvorova odjednom • Nije moguće mijenjati scenu dok je igra pokrenuta 	<ul style="list-style-type: none"> • Prozor za prikaz – vizualni pregled videoigre i onog što se izrađuje • Nema mogućnost brzog letenja pomoću tipke „Shift“ • Nema mogućnost micanja, rotiranja, skaliranja glumaca odjednom • Mogućnost mijenjanja scene dok je igra pokrenuta
	Materijali	<ul style="list-style-type: none"> • Pojednostavljen prikaz uredivanja materijala 	<ul style="list-style-type: none"> • Pojednostavljen prikaz uredivanja materijala 	<ul style="list-style-type: none"> • Vizualno skriptiranje materijala
	Teren	<ul style="list-style-type: none"> • Ugrađen unutar uređivača • Neograničen broj slojeva materijala za bojanje terena • Određivanje veličine teksture • Modeliranje terena • Ukrašavanje vegetacijom pomoću tekstura • Generiranje sudarača 	<ul style="list-style-type: none"> • „Heightmap terrain 3D“ alat iz AssetLibrary-a • 4 sloja za bojanje terena • Ograničeno modeliranje terena • Ograničeno ukrašavanje vegetacijom pomoću tekstura 	<ul style="list-style-type: none"> • Ugrađen unutar uređivača • Neograničen broj slojeva materijala za bojanje terena • Napredno modeliranje terena • „Quixel bridge“ automatizacija dodavanje visokokvalitetnih 3D modela • Generiranje sudarača
	Sustav čestica	<ul style="list-style-type: none"> • Intuitivan i lako razumljiv sustav čestica • Velika mogućnost personalizacije efekata 	<ul style="list-style-type: none"> • Intuitivan i lako razumljiv sustav čestica • Manja mogućnost personalizacije efekata 	<ul style="list-style-type: none"> • Cascade – sustav za stvaranje efekata čestica • Niagara – napredan sustav za stvaranje efekata čestica
	Animacije	<ul style="list-style-type: none"> • Mogućnost animacije svih objekata u hijerarhiji • Lako i jednostavno za korištenje 	<ul style="list-style-type: none"> • Mogućnost animacije svih varijabli u inspektoru • Lako i jednostavno za korištenje 	<ul style="list-style-type: none"> • Napredno animiranje pomoću: <ul style="list-style-type: none"> - „Skeletal Mesh“ - „Blend Spaces“ - „Animation Blueprints“ - „State Machines“ - „Sequencer“ • Teže animirati jednostavne animacije
	Korisničko sučelje	<ul style="list-style-type: none"> • Tri vrste sučelja: <ul style="list-style-type: none"> - UI Toolkit - Unity Graphical User Interface (uGUI) - Immediate Mode Graphical User Interface (IMGUI) 	<ul style="list-style-type: none"> • Na glavnom prostoru na 2D sceni se izrađuje korisničko sučelje videoigre 	<ul style="list-style-type: none"> • Unreal Motion Graphics (UMG) – izrada „Widget Blueprint“ pomoću „Blueprint Visual Scripting“
Rendering		<ul style="list-style-type: none"> • Universal Render Pipeline (URP) – iscrtavanje grafike • „CloudsToy“ – generiranje razno-raznih oblaka 	<ul style="list-style-type: none"> • GLES2 i GLES3 – iscrtavanje grafike temeljene na OpenGL-u • „Cloud“ – alat iz AssetLibrary-a za generiranje oblaka 	<ul style="list-style-type: none"> • DirectX 11 i DirectX 12 • „Nanite“ – tehnologija prikazivanja. Omogućava velik broj objekata s velikim brojem poligona • „Lumen“ – dinamički sustav globalnog osvjetljenja

Izvor: autor

8. ZAKLJUČAK

Za potrebe ovog završnog rada izrađene su tri slične videoigre u tri različita okruženja za izradu videoigara – Unity-u, Godot-u i Unreal Engine-u. Općenito, okruženja za izradu video igara su računalni programi koji pružaju korisnicima mnogobrojne korisne alate za jednostavniju izradu videoigara. Izradom videoigara evaluirana su sva tri okruženja te se zaključuje da svako ima svoje prednosti i mene, te se međusobno razlikuju u mnogobrojnim stavkama.

Unity koristi *C#* kao primarni programski jezik te se glavni značaj pridaje objektima. U Godotu je primarni programski jezik *GDS*cript, dok Unreal Engine koristi *Visual Scripting* i nacrte. Unreal Engine i Unity imaju detaljan prikaz greški, dok Godot-ov IDE prikazuje jednu grešku odjednom. Što se tiče sučelja programa, Unity je u tom pogledu najjači. Omogućuje najveću prilagodbu programskog sučelja te dinamično micanje svih kartica, dok su Godot i Unreal ograničeni (pr. scena se uvijek nalazi na sredini sučelja i ne može se pomaknuti). Godot također manjka u pogledu uklanjanja greški tijekom pokretanja nivoa. Unity i Godot oboje nude funkcionalnosti istovremenog micanja, rotiranja te proporcioniranja objekta što uvelike ubrzava i pojednostavljuje rad na sceni, dok Unreal Engine tu funkcionalnost nema. Prozirne materijale je najlakše implementirati u Godot okruženju dok je u Unity-u i Unreal Engine-u sama implementacija nešto složenija. Što se tiče animacija, Unreal Engine sadrži veliki broj naprednih alata za animiranje, međutim izradu jednostavnijih animacija je lakše realizirati u Godot-u ili Unity-u. Unreal Engine ima napredan sustav čestica *Niagara* koja nadmašuje one u Godot-u i Unity-u. Dakako, sustave čestica je mnogo jednostavnije implementirati u Godot-u i Unity-u i pritom dobiti zadovoljavajuće rezultate. Izrada terena je uvelike različita u sva tri okruženja. Godot sadrži najmanje razvijen sustav za izradu terena, dok je Unity-jev sustav poprilično napredniji, no ne može se usporediti sa sustavom terena u Unreal Engine-u. U Unreal Engine-u je integriran *Quixel* alat za pregledavanje, pretraživanje, preuzimanje, uvoz i izvoz 3D modela visoke kvalitete. Naposlijetku, što se tiče iscrtavanja slike, Unreal Engine ima odlične *Lumen* (dinamički sustav globalnog osvjetljenja i odraza) i *Nanite* (korištenje objekata s velikim brojem poligona) tehnologije.

Evaluacijom okruženja za izradu videoigara temeljem izrade jednostavne videoigre, zaključuje se da Unreal Engine okruženje nudi najviše mogućnosti, međutim najzahtjevniji je za savladati, posebice za početnike. Unity okruženje idealno je za početnike i naprednije korisnike zbog velikog broja alata i materijala, intuitivnosti i mogućnosti personalizacije radnog prostora.

Godot, kao relativno novo okruženje u stalnom razvoju, nudi veliku količinu alata kao i druga dva okruženja. Međutim, kako se radi o besplatnom okruženju otvorenog koda s manjim brojem razvojnih programera, Godot nudi nešto jednostavnije alate sa svojim ograničenjima i manjim brojem funkcionalnosti.

9. POPIS LITERATURE

- [1] Stefanidis, M. (18. travanj 2022). *Exploring the Growth of Game Engines*. Preuzeto 20. rujan 2022 iz Round Hill Investments: <https://www.roundhillinvestments.com/research/metaverse/exploring-the-growth-of-game-engines#:~:text=On%20Steam%20alone,%20there%20are,:%20open-source%20and%20proprietary>.
- [2] Unity (game engine). (n.d.). Preuzeto 05. rujan 2022 iz Wikipedia: [https://en.wikipedia.org/wiki/Unity_\(game_engine\)](https://en.wikipedia.org/wiki/Unity_(game_engine))
- [3] List of game engines. (n.d.). Preuzeto 15. rujan 2022 iz Wikipedia: https://en.wikipedia.org/wiki/List_of_game_engines
- [4] Why Unity is the Best Game Engine for Beginners. (6. srpanj 2021). Preuzeto 26. rujan 2022 iz Mastery Coding: <https://www.masterycoding.com/blog/unity-best-beginner-engine>
- [5] Whitepaper - Feature List. (22. veljača 2006). Preuzeto 26. rujan 2022 iz Unity 3D: <https://web.archive.org/web/20060222000905/http://unity3d.com/whitepaper.html>
- [6] Unity download archive. (n.d.). Preuzeto 5. rujan 2022 iz Unity: <https://unity3d.com/get-unity/download/archive>
- [7] Long Term Support. (n.d.). Preuzeto 5. rujan 2022 iz Unity: <https://unity3d.com/unity/qa/lts-releases>
- [8] Unity 2022.1. (n.d.). Preuzeto 6. rujan 2022 iz Unity: <https://unity.com/releases/2022-1>
- [9] Unity 2023.1.0 Alpha 4. (n.d.). Preuzeto 5. rujan 2022 iz Unity: <https://unity3d.com/unity/alpha/2023.1.0a4>
- [10] Godot Docs Introduction. (n.d.). Preuzeto 12. rujan 2022 iz Godot Engine: <https://docs.godotengine.org/en/stable/about/introduction.html>
- [11] Console Support in Godot. (n.d.). Preuzeto 6. rujan 2022 iz Godot Engine: <https://docs.godotengine.org/en/3.0/tutorials/platform/consoles.html>
- [12] Godot Docs File system. (n.d.). Preuzeto 7. rujan 2022 iz Godot Engine: <https://docs.godotengine.org/en/stable/tutorials/scripting/filesystem.html>

- [13] *Godot Docs GDNative C++ example.* (n.d.). Preuzeto 7. rujan 2022 iz Godot Engine: https://docs.godotengine.org/en/stable/tutorials/scripting/gdnative/gdnative_cpp_example.html
- [14] *Godot 4.0 will discontinue VisualScript.* (n.d.). Preuzeto 5. rujan 2022 iz Godot Engine: <https://godotengine.org/article/godot-4-will-discontinue-visual-scripting>
- [15] *Godot Download.* (n.d.). Preuzeto 2. rujan 2022 iz Godot Engine: <https://godotengine.org/article/godot-4-will-discontinue-visual-scripting>
- [16] *The world's most open and advanced real-time 3D creation tool.* (n.d.). Preuzeto 3. rujan 2022 iz Unreal Engine: <https://www.unrealengine.com/en-US>
- [17] *MetaHumans in Quixel Bridge.* (n.d.). Preuzeto 26. rujan 2022 iz Unreal Engine: <https://docs.metahuman.unrealengine.com/en-US/metahumans-in-quixel-bridge#:~:text=In%20Unreal%20Engine%205,%20Quixel,with%20your%20Epic%20Games%20account>.
- [18] Statt, N. (13. lipanj 2020). *Epic Games announces Unreal Engine 5 with stunning PlayStation 5 demo.* Preuzeto 4. rujan 2022 iz The Verge: <https://www.theverge.com/2020/5/13/21256079/epic-unreal-engine-5-playstation-5-demo-next-gen-graphics-release-date>
- [19] Orland, K. (14. svibanj 2020). *How Epic got such amazing Unreal Engine 5 results on next-gen consoles.* Preuzeto 6. rujan 2022 iz Ars Technica: <https://arstechnica.com/gaming/2020/05/behind-the-scenes-of-that-incredible-unreal-engine-5-tech-demo/>
- [20] *Blueprint Visual Scripting.* (n.d.). Preuzeto 6. rujan 2022 iz Unreal Engine: <https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/>
- [21] *Coding in C# in Unity for beginners.* (n.d.). Preuzeto 1. rujan 2022 iz Unity: <https://unity.com/how-to/learning-c-sharp-unity-beginners#:~:text=The%20language%20that's%20used%20in,variables,%20functions,%20and%20classes>
- [22] *Update and FixedUpdate.* (4. sječanj 2022). Preuzeto 2. rujan 2022 iz Unity Learn: <https://learn.unity.com/tutorial/update-and-fixedupdate>

- [23] *Awake and Start*. (4. sječanj 2022). Preuzeto 2. rujan 2022 iz Unity Learn: <https://learn.unity.com/tutorial/awake-and-start>
- [24] *Build Unity Games with Visual Studio*. (n.d.). Preuzeto 4. rujan 2022 iz Microsoft: <https://visualstudio.microsoft.com/vs/unity-tools/>
- [25] *Godot Docs GDScript basics*. (n.d.). Preuzeto 9. rujan 2022 iz Godot Engine: https://docs.godotengine.org/en/stable/tutorials/scripting/gdscript/gdscript_basics.html
- [26] *Godot Docs Overview of Godot's key concepts*. (n.d.). Preuzeto 13. rujan 2022 iz Godot Engine: https://docs.godotengine.org/en/stable/getting_started/introduction/key_concepts_overview.html
- [27] *Introduction to Blueprints*. (n.d.). Preuzeto 12. rujan 2022 iz Unreal Engine: <https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/GettingStarted/#:~:text=The%20Blueprint%20Visual%20Scripting%20system,or%20objects%20in%20the%20engine>
- [28] *Blueprint Overview*. (n.d.). Preuzeto 13. rujan 2022 iz Unreal Engine: <https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/Overview/>
- [29] *WaitForSeconds*. (n.d.). Preuzeto 14. rujan 2022 iz Unity Documentation: <https://docs.unity3d.com/ScriptReference/WaitForSeconds.html>
- [30] Nielsen, J. (15. studeni 2020). *10 Usability Heuristics for User Interface Design*. Preuzeto 7. rujan 2022 iz Nielsen Norman Group: <https://www.nngroup.com/articles/ten-usability-heuristics/>
- [31] *Godot Docs Using signals*. (n.d.). Preuzeto 12. rujan 2022 iz Godot Engine: https://docs.godotengine.org/en/stable/getting_started/step_by_step/signals.html
- [32] *World Outliner*. (n.d.). Preuzeto 16. rujan 2022 iz Unreal Engine: <https://docs.unrealengine.com/4.27/en-US/BuildingWorlds/LevelEditor/SceneOutliner/>
- [33] *Details Panel*. (n.d.). Preuzeto 12. rujan 2022 iz Unreal Engine: <https://docs.unrealengine.com/4.27/en->

[US/ProgrammingAndScripting/Blueprints/Editor/UIComponents/Details/#:~:text=The%20Details%20panel%20is%20a,to%20organize%20the%20included%20properties](#)

- [34] *Materials.* (n.d.). Preuzeto 14. rujan 2022 iz Unreal Engine: <https://docs.unrealengine.com/5.0/en-US/unreal-engine-materials/>
- [35] *Materials.* (n.d.). Preuzeto 18. rujan 2022 iz Unity Documentation: <https://docs.unity3d.com/Manual/Materials.html>
- [36] *Godot Docs Spatial Material.* (n.d.). Preuzeto 16. rujan 2022 iz Godot Engine: https://docs.godotengine.org/en/stable/tutorials/3d/spatial_material.html
- [37] *Creating and editing Terrains.* (n.d.). Preuzeto 18. rujan 2022 iz Unity Documentation: <https://docs.unity3d.com/Manual/terrain-UsingTerrains.html>
- [38] *Landscape Outdoor Terrain.* (n.d.). Preuzeto 12. rujan 2022 iz Unreal Engine: <https://docs.unrealengine.com/5.0/en-US/landscape-outdoor-terrain-in-unreal-engine/>
- [39] *Particle system.* (n.d.). Preuzeto 19. rujan 2022 iz Wikipedija: https://en.wikipedia.org/wiki/Particle_system
- [40] *Animation.* (n.d.). Preuzeto 15. rujan 2022 iz Unity Documentation: <https://docs.unity3d.com/Manual/AnimationSection.html>
- [41] *Godot Docs Introduction to the animation features.* (n.d.). Preuzeto 17. rujan 2022 iz Godot Engine: <https://docs.godotengine.org/en/stable/tutorials/animation/introduction.html>
- [42] *Creating and Displaying UI.* (n.d.). Preuzeto 18. rujan 2022 iz Unreal Engine: <https://docs.unrealengine.com/4.27/en-US/InteractiveExperiences/UMG/HowTo/CreatingWidgets/>
- [43] *Create user interfaces (UI).* (n.d.). Preuzeto 11. rujan 2022 iz Unity Documentation: <https://docs.unity3d.com/Manual/UIToolkits.html>
- [44] *Rendering (computer graphics).* (n.d.). Preuzeto 10. rujan 2022 iz Wikipedija: [https://en.wikipedia.org/wiki/Rendering_\(computer_graphics\)](https://en.wikipedia.org/wiki/Rendering_(computer_graphics))
- [45] *Real-time rendering in 3D.* (n.d.). Preuzeto 17. rujan 2022 iz Unity: <https://unity.com/how-to/real-time-rendering-3d>

- [46] *Universal Render Pipeline overview*. (n.d.). Preuzeto 19. rujan 2022 iz Unity: <https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@12.1/manual/index.html>
- [47] *Lumen Global Illumination and Reflections*. (n.d.). Preuzeto 19. rujan 2022 iz Unreal Engine: <https://docs.unrealengine.com/5.0/en-US/lumen-global-illumination-and-reflections-in-unreal-engine/>

10. POPIS SLIKA

Slika 1: Prikaz objekata u hijerarhiji	6
Slika 2: Prikaz raznih objekata u sceni	6
Slika 3: Prikaz komponenti objekta wooden_bench zajedno sa pridruženom skriptom Health u Inspektoru.....	7
Slika 4: Prikaz skripte s varijablama, funkcijama i klasama	7
Slika 5: Prepoznavanje i nuđenje rješenja.....	8
Slika 6: Primjer GDScript koda	9
Slika 7: Prikaz scena u Godot-u	9
Slika 8: Prikaz čvorova u sceni „Zombie“	10
Slika 9: Primjer signala u Godot-u.....	10
Slika 10: Prikaz programskih greški u Godot-ovom IDE-u	11
Slika 11: Povlačenje i spuštanje scene u kod (Godot)	11
Slika 12: Primjer nacrtu u Unreal Engine-u	12
Slika 13: Prikaz Glumaca u Unreal Engine-u	12
Slika 14: Primjer komponenti glumca „Igrača“ u Unreal Engine-u.....	13
Slika 15: Prikaz stvaranja čvorova u Unreal Engine-u.....	13
Slika 16: Komentiranje više čvorova i grupiranje u Unreal Engine-u	14
Slika 17: Ispravljanje žica u Unreal Engine-u.....	14
Slika 18: Prikaz „Spawner“ skripte u Unity-u.....	15
Slika 19: Prikaz programskog koda „Spawned“ skripte u Unity-u.....	16
Slika 20: Prikaz „Spawner“ skripte u Godot-u.....	16
Slika 21: Prikaz koda „Zombie“ skripte zaslužan za ažuriranje „Spawner“ skripte u Godot-u	17
Slika 22. Unutar ZombieGameMode nacrtu svaki korak se pokreće stvaranje glumaca	17
Slika 23: „Spawning Zombie“ prilagođeni događaj u Unreal Engine-u - prvi dio	18
Slika 24: „Spawning Zombie“ prilagođeni događaj u Unreal Engine-u - drugi dio	18
Slika 25: Pozivanje „Zombie Killed“ događaja iz „Zombie“ nacrtu.....	19
Slika 26: Prikaz „ZombieKilled“ prilagođenog događaja u Unreal Engine-u	19
Slika 27: Prikaz sučelja u Unity-u.....	20
Slika 28: Primjer hijerarhije u Unity-u.....	21
Slika 29: Primjer običnog inspektora u Unity-u.....	23
Slika 30: Primjer uređenog inspektora u Unity-u.....	22

Slika 31: Primjer „Project“ kartice u Unity-u	22
Slika 32: Primjer scene u Unity-u	23
Slika 33: Prikaz pokrenute scene u Unity-u	23
Slika 34: Prikaz sučelja u Godot-u.....	24
Slika 35: Primjer stabla scena u Godot-u	25
Slika 36: Primjer inspektora u Godot-u.....	26
Slika 37: Primjer signala u Godot-u.....	25
Slika 38: Primjer grupa u Godot-u	25
Slika 39: Prikaz datotečnog sustava u Godot-u.....	26
Slika 40: Prikaz radnog prostora u Godot-u.....	27
Slika 41: Prikaz Unreal Engine sučelja	27
Slika 42: Prikaz obrisnik panele u Unreal Engine-u	28
Slika 43: Prikaz panela detalja u Unreal Engine-u.....	29
Slika 44: Prikaz Unreal Engine panela preglednika sadržaja.....	29
Slika 45: Prozor za prikaz u Unreal Engine-u.....	30
Slika 46: Izlaz iz igrača u pokrenutom nivou.....	30
Slika 47: Primjer neprozirnog materija u Unity-u.....	33
Slika 48: Primjer prozirnog materijala u Unity-u.....	32
Slika 49: Primjer neprozirnog materijala u Godot-u.....	34
Slika 50: Primjer prozirnog materijala u Godot-u.....	33
Slika 51: Primjer neprozirnog materija Unreal Engine.....	35
Slika 52:Primjer prozirnog materija Unreal Engine.....	34
Slika 53: Primjer pojedinosti prozirnog materijala Unreal Engine	34
Slika 54: Prikaz Terrain objekta u Unity-u	35
Slika 55: Prikaz ukrašavanja krajolika u Unity-u.....	35
Slika 56: Primjer Godot terena	36
Slika 57: Postavljanje pejzažnog materijala u Unreal Engine-u	36
Slika 58: Prikaz pejzažnog materijala (smanjeni prikaz).....	38
Slika 59: Prikaz pejzažnog materijala (uvećani prikaz)	37
Slika 60: Primjer ukrašavanja terena u Unreal Engine-u	37
Slika 61: Primjer sustava čestica u Unity sceni.....	39
Slika 62: Primjer sustava čestica u Unity inspektoru.....	38
Slika 63: Primjer sustava čestica u Godot inspektoru.....	40
Slika 64: Primjer sustava čestica u Godot sceni.....	39

Slika 65: Primjer Unreal Engine Cascade sustava čestica	40
Slika 66: Primjer Unreal Engine Niagara sustava čestica	40
Slika 67: Primjer animacije revolvera s događajem u Unity-u	41
Slika 68: Godot primjer animacije	41
Slika 69: Primjer animacije u Unreal Engine-u	42
Slika 70: Prikaz koda koji poziva animaciju u Unreal Engine-u	43
Slika 71: Primjer izrade korisničkog sučelja koristeći uGUI u Unity-u	44
Slika 72: Primjer izrade korisničkog sučelja u Godot-u	44
Slika 73: Primjer izrade korisničkog sučelja u Unreal Engine-u	45
Slika 74: Primjer funkcionalnosti korisničkog sučelja u Unreal Engine-u	45
Slika 75: Postavke okruženja u Unity-u	46
Slika 76: Unity paket - Clouds Toy	47
Slika 77: Dodatne postavke za namještanje kvalitete okruženja u Godot-u	47
Slika 78: Godot WorldEnvironment	48
Slika 79: Podešavanje oblaka u Godot-u	48

11. POPIS TABLICA

Tablica 1: Usporedba okruženja po predmetima vrednovanja 51

12. POPIS GRAFIKONA

Grafikon 1. Prikaz potrebnih memorijskih resursa za svako od okruženja..... 5