Izdavanje višeplatformskih projekata korištenjem Unreal Enginea i standarda Vulkan

Sedlar, Hrvoje

Sveučilište u Rijeci – Fakultet informatike i digitalnih tehnologija

Sveučilišni prijediplomski studij Informatika

Hrvoje Sedlar

Izdavanje višeplatformskih projekata korištenjem Unreal Enginea i standarda Vulkan

Završni rad

Mentor: doc. dr. sc. Vedran Miletić

Rijeka, 05.06.2023.

FIDIT
Sveučilište u Rijeci
Fakultet informatike
i digitalnih tehnologija
www.inf.uniri.hr

Rijeka, 14. ožujka 2023.

Zadatak za završni rad

Pristupnik: Hrvoje Sedlar

Naziv završnog rada: Izdavanje višeplatformskih projekata korištenjem Unreal Enginea i standarda Vulkan

Naziv završnog rada na engleskom jeziku: Publishing cross-platform projects using Unreal Engine and Vulkan

Sadržaj zadatka:

Moderni pogoni za razvoj igara, kao što su Unreal Engine, Unity i Godot, značajno olakšavaju razvoj igara za različite platforme, a specijalno omogućuju i razvoj višeplatformske igre. Cilj rada je opisati način korištenja Unreal Enginea za razvoj i izdavanje višeplatformske igre oblikovane po želji studenta.

Mentor

Doc. dr. sc. Vedran Miletić

Voditelj za završne radove

Doc. dr. sc. Miran Pobar

Zadatak preuzet: 14. ožujka 2023.

(potpis pristupnika)

Adresa: Radmile Matejčić 2
51000 Rijeka,
Tel: +385(0)51 584 700
E-mail: ured@inf.uniri.hr
OIB: 64218323816
IBAN: HR1524020061400006966
uniri

Sadržaj

1. Uvod

U ovome završnom radu opisan je standard Vulkan koji je grafičko aplikacijsko sučelje (API) i služi za poboljšanje komunikacije između softvera (koda programa) i hardvera (grafičkih uređaja). Također, opisano je i Unreal Engine (UE) razvojno okruženje za izradu video igara, filmova, animacija i sličnih grafičkih projekata i njegova primjena uz standard Vulkan.

Kroz povijest računalstva bilo je potrebno stalno poboljšavati standarde kako je tehnologija napredovala i samim time je i nastao Vulkan koji je svojevrsna revolucija u tome kako razvojni programeri mogu u potpunosti kontrolirati funkcije i parametre računalne grafike tokom razvoja video igara ili drugih projekata. Druga grafička aplikacijska sučelja ovise o tome koliko su njihovi distributeri bili voljni razviti njihove funkcije i dostupnost istih, dok Vulkan omogućuje potpunu kontrolu nad hardverom grafičkog procesora.

Isto tako Unreal Engine je razvojno okruženje izvorno razvijano za first-person shootere, koje se svojevremeno nadograđivalo i postalo jedan od najpopularnijih razvojnih okruženja za razvoj višeplatformskih video igara, projekata i sl. Svojom jednostavnošću korištenja i detaljnom dokumentacijom privlačno je i ljudima sa manjak iskustva jer nudi dosta opcija kao vizualno programiranje sa blueprintovima ukoliko nam manjka znanja u pisanju programskog programskog koda, što je prednost za početnike.

U poglavlju 2. opisuje se standard Vulkan, njegove prednosti zajedno sa procesom standardizacije Grupe Khronos i njihovim utjecajem na samu industriju. Prikazujemo primjer grafike korištenjem Vulkana te opisujemo elemente njegovog izvođenja.

S poglavljem 3. prolazimo kroz povijest Unreal Enginea, te opisujemo trenutnu aktivnu generaciju i njezino sučelje objašnjavajući funkcije samog radnog okruženja za stvaranje video igara.

Poglavlje 4. je glavni dio ovog završnog rada i ono prikazuje razvoj višeplatformskih igara korištenjem Unreal Enginea i Vulkana, tj., neke od njegovih koraka kao kako se sam Vulkan testira u UE-u, koji koraci su potrebni za kompajliranje sadržaja i njegov izvoz u željeni direktorij te pokretanje na različitim platformama kao što su Windows OS i Android OS.

2. Standardi Vulkan

Vulkan je grafičko aplikacijsko sučelje (API) koju kojeg je razvila Grupa Khronos koja pruža bolju upotrebu modernih grafičkih procesora. Ovo sučelje omogućuje bolji opis namjene aplikacije, što vodi do boljih performansi i manje grešaka drivera (upravljačkih programa), kao posljedica boljeg poklapanja usklađenosti dizajna Vulkana i modernog grafičkog hardvera, u usporedbi s postojećim API-evima poput OpenGL-a i Direct3D-a. Vulkan ima prednost po tome što se potpuno može koristiti u višeplatformske svrhe i dozvoljava razvoj za Windows, Linux i Android OS-ove istovremeno [1].

Problem većine API-eva je taj što u vremenu u kojem su se izradili je bio dostupan grafički hardver koji je bio limitiran na konfiguraciju fiksne funkcionalnosti. Programeri su morali dostaviti podatke o vertex data (vršnim točkama) u standardnom formatu i sve opcije svjetline i sjenčanja su bile dostupne jedino od proizvođača grafičkih procesora.

S vremenom su arhitekture grafičkih procesora napredovale, pa su počele biti dostupne i funkcionalnosti koje se mogu više programirati. Sve se to moralo integrirati u postojeća API sučelja. To je rezultiralo tome da je puno programerovih namjera ovisilo o grafičkim driverima i njihovim funkcionalnostima u slijedu s modernim grafičkim arhitekturama. Zbog toga je bilo potrebno stalno ažurirati drivere za poboljšanje performansi u video igrama, ponekad sa značajnim poboljšanjima. Zbog kompleksnosti samih drivera, programeri samih aplikacija su se borili s nedosljednostima između distributera sučelja, kao što je sintaksa za shadere (sjenčanje). Također, velikom navalom mobilnih uređaja u prošlom desetljeću, njihovi mobilni grafički procesori su zahtijevale drugačiji pristup renderiranju (izvođenju grafike) zbog njihove efikasnosti. Jedan od primjera je tzv. „tiled rendering”, koji ima mogućnost većih performansi jer nudi programeru više kontrole oko te funkcije. Još jedno ograničenje iz doba ovih API-eva je limitirana podrška za multithreading (višenitnosti) koja može uzrokovati smanjenje performansi, ili tzv. „bottleneck” od strane procesora.

Vulkanom se ovi problemi rješavaju zbog toga što je iz početka razvijan za moderne grafičke arhitekture. Smanjuje ovisnost o driverima jer dozvoljava programerima da jasno specificiraju svoje namjene koristeći opširni API i dozvoljava višenitnosti za kreiranje i izvođenje naredbi paralelno. Smanjuje nedosljednosti u kompilaciji sjenčanja prebacivanjem na standardizirani format bajt koda s jednim kompajlerom. Naposljetku, priznaje mogućnosti obrade opće namjene modernih grafičkih procesora objedinjavanjem grafičke i računalne funkcionalnosti u jedan API [2].

Slika 2.1. - Vulkan logotip

2.1. Grupa Khronos i proces standardizacije

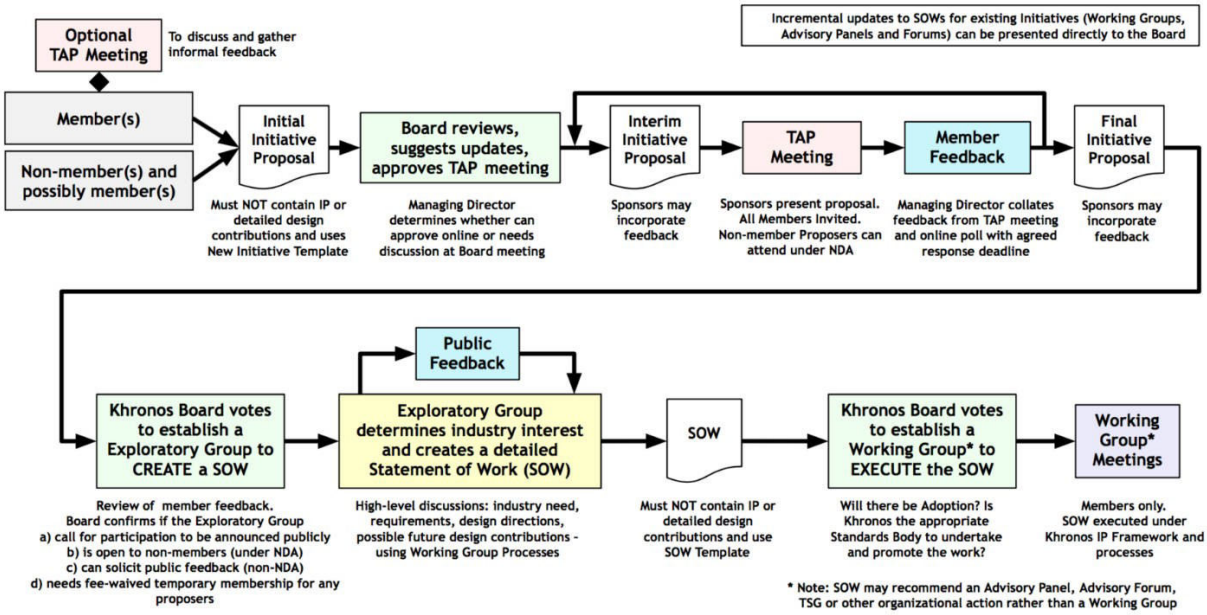
Grupa Khronos (The Khronos Group, Inc.) je otvoreni, neprofitni konzorcij koji se sastoji od preko 150 kompanija koje su vodeće u industriji i zaslužne su za stvaranje naprednih „royalty-free interoperability standards" (standarda interoperabilnosti bez naknade) za 3D grafiku, proširenu i virtualnu stvarnost, paralelno programiranje, „vision-acceleration" (vidno ubrzanje) i „machine learning" (strojno učenje).

Slika 2.1.1. - Logotip Grupe Khronos

Neki od njihovih standarda su Vulkan, Vulkan SC, OpenGL, OpenGL SC, WebGL, SPIR-V, OpenCL, SYCL, OpenVX, NNEF, OpenXR, 3D Commerce, ANARI i glTF. Članovi Grupe Khronos su dozvoljeni doprinositi razvoju Khronosovih specifikacija i omogućeni su glasati na raznim stadijima razvoja standarda prije predstavljanja javnosti i mogu ubrzati dostavu svojih vrhunskih platformi i aplikacija preko ranog pristupa nacrtima specifikacija i testovima usklađenosti. [3]

Grupa Khronos koristi metodični „New Initiative" (Nova Inicijativa) proces za procjenu interesa industrije u novim prijedlozima standardizacije. Proces je izravan za svaku kompaniju koja percipira potrebu industrije za prijedlog nove suradnje za Grupu Khronos,

bez obzira jesu li članovi organizacije. Pomno procjenjujući povratne informacije od industrije na identificirane probleme i predložene solucije, Grupa Khronos može odrediti prioritete standardizacijskih prijedloga koji imaju dobru šansu da pozitivno utječu na industriju u širokom spektru primjene. Khronosova Nova Inicijativa ima nekoliko segmenata procjene prije samog početka detaljnog dizajna, što omogućuje tvrtkama da procijene hoće li sudjelovati prije nego li preuzmu obveze licenciranja IP-a.



Slika 2.1.2. - Proces Khronosove Nove Inicijative

Svaka kompanija, ili skupina kompanija s istim ciljevima, može završiti ili sponzorirati „Prijedlog Khronosove Nove Inicijative" [4], koji ne smije sadržavati bilo kakve detaljne doprinose koji su opterećeni IP-om, već se usredotočuje na probleme industrije na visokoj razini i prijedloge kako ih obraditi, te kako Grupa Khronos može doprinijeti industriji rješavanjem tih problema. Ovaj Prijedlog je pregledan od strane Khronosovog odbora i diskutiran u „Khronos Tehnical Advisory Panel" (TAP) koji je dostupan bilo kojem članu Khronosa i omogućuje raspravu o novom temama ili mogućnostima koje obuhvaćaju više radnih skupina, bez razgranavanja IP licenciranja. Rasprave u TAP-u doprinose ažuriranja Prijedlogu Inicijative i podršku dodatnih sponzora. Ako su povratne informacije članova pozitivne, sponzori mogu proslijediti konačni Prijedlog Inicijative Khronosovom odboru.

Odbor će zatim razmotriti obrađeni Prijedlog Inicijative, koliko je on primjeren Khronosu da dalje istraži predloženu Inicijativu, kolika je razina podrške članova Khronosa i zatim će glasati za stvaranje „Exploratory Group" (Istraživačka Grupa). Ako je stvorena, Istraživačka Grupa dobiva resurse za raspravu i postizanje konsenzusa za stvaranje detaljnog „Statement of Work" (SOW). SOW onda dalje istražuje probleme i potrebe industrije

spomenute u Prijedlogu Inicijative i prijedloge kako proces standardizacije može doprinijeti industriji preko predloženih proizvoda visoke razine i općih smjernica dizajna.

Ovisno o prirodi prijedloga, Istraživačka Grupa će možda izdati javni poziv za participaciju, i omogućiti participaciju licima koji nisu Khronosovi članovi pod ugovorom o tajnosti (NDA), ili će zatražiti povratnu informaciju od strane javnosti na odabrane materijale koje su učinili dostupnima. Za izbjegavanje problema sa IP-em, Istraživačka Grupa se ne bavi detaljnim projektima, nego se usredotočuje na raspoznavanje potrebnih slučajeva, zahtjeva i ciljeva na visokoj razini i objekata koji se zapisuju u generiranom SOW-u.

Ako Istraživačka Grupa postigne koncenzus za nastavljanje inicijativom, poslat će zahtjev Khronosovom odboru za glasanje za uspostavljanje Working Group (Radna Grupa) koja će izvršiti SOW. Ako je Radna Grupa uspostavljena, Khronosovo članstvo je potrebno za participaciju. Grupa Khronos je otvorena bilo kojoj organizaciji koja se želi pridružiti za prijedloge i glasanje u stvaranju rezultirajućih standarda pod Khronosovim dokazanim IP Framework-om i procesa upravljanja više kompanija. [5]

2.2 Primjer prikaza grafike korištenjem Vulkana

Vulkan zahtjeva dosta koda za svoje izvršavanje jer u odnosu na ostale API-eve se može puno bolje optimizirati što i kako grafički procesori rade zbog performansi. Koristiti ćemo grafički cjevovod, zbog toga što osim grafike, Vulkan jako ozbiljno shvaća računalna polja. Generalno, računanje se sastoji od mnogo algoritama koje nisu 3D prirode (GPU ih ne obrađuje kao trokute), ali se mogu ubrzati distributivnom snagom samog GPU-a. Na ovaj način Vulkan specificira grafički i računalni cjevovod. Objekt grafičkog cjevovoda sadrži detalje o različitim razinama (vršne točke, geometrija, teselacija, itd.) koje izvršavaju shadere; stanja ulaznog sklopa koji stvara linije i trokute iz međuspremnika; okvir za prikaz i stanje rasterizacije; dubinsko stanje šablone, itd.

Prikazi slika i meta podatci su objekti koji se nalaze između shadera i stvarnih resursa iz kojih se čita ili u koje se piše. Dozvoljavaju nam ograničavanje pristupa resursima (npr. možemo stvoriti pogled slike koja predstavlja jednu sliku u nizu) i neku kontrolu nad izgledom formata resursa. Render pass (prolaz izvođenja grafike) održava listu svih resursa koji će se koristiti i njihovih ovisnosti (npr. kada se resurs promjeni iz izlaznog u ulazni tokom mapiranja sjena). Framebuffer (međuspremnik okvira) radi skupa sa prolazom izvođenja grafike tako da stvara vezu u dva koraka iz cjevovoda do resursa. Prolaz izvođenja grafike je vezan za međuspremnik naredbi i sadrži indekse u međuspremniku okvira. Međuspremnik okvira preslikava te indekse u prikaze slika (koji referenciraju stvarni resurs). [6]

Primjer koji ćemo koristiti se nalazi na web sjedištu Vulkan Tutorial i obuhvaća cijelo poglavlje o crtanju trokuta [7]. Kada ga pokrenemo, dobijemo skočni prozor na kojem se iscrta trokut koristeći Vulkan API i on izgleda ovako:

Slika 2.2.1. Trokut nacrtan Vulkanom

Primjetimo da je trokut obojan RGB bojama. Konfiguriran je na način da uz pomoć shadera obojamo okruženje vrhova bojama kojima želimo i one se lijepo prelijevaju jedna u drugu. Originalni vert.spv file izgleda ovako:

main.cpp vert.spv frag.spv

```
#version 450

layout(location = 0) out vec3 fragColor;

vec2 positions[3] = vec2[](
    vec2(0.0, -0.5),
    vec2(0.5, 0.5),
    vec2(-0.5, 0.5)
);

vec3 colors[3] = vec3[](
    vec3(1.0, 0.0, 0.0),
    vec3(0.0, 1.0, 0.0),
    vec3(0.0, 0.0, 1.0)
);

void main() {
    gl_Position = vec4(positions[gl_VertexIndex], 0.0, 1.0);
    fragColor = colors[gl_VertexIndex];
}
```

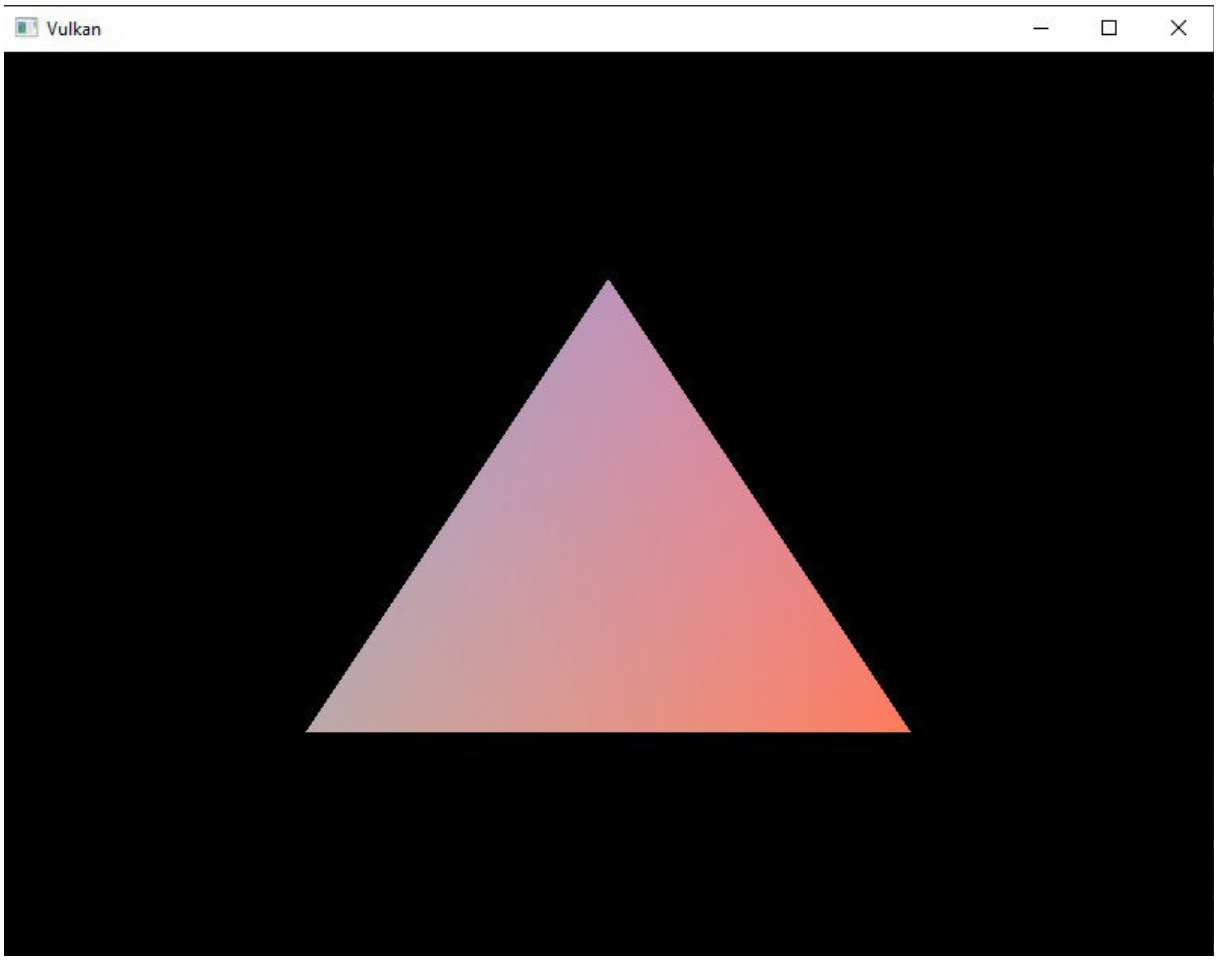
Slika 2.2.2. vert.spv

Uređivanjem parametara boja sve tri točke možemo dobiti različite rezultate. Svaka točka sadrži parametre crvenu, zelenu i plave boje (RGB) i tvori matricu. Npr. za ovaj unos:

```
vec3 colors[3] = vec3[](
    vec3(0.5, 0.3, 0.5),
    vec3(1.0, 0.2, 0.1),
    vec3(0.5, 0.4, 0.4)
);
```

Slika 2.2.3. Prvi primjer modificiranog unosa

dobivamo trokut koji izgleda ovako:



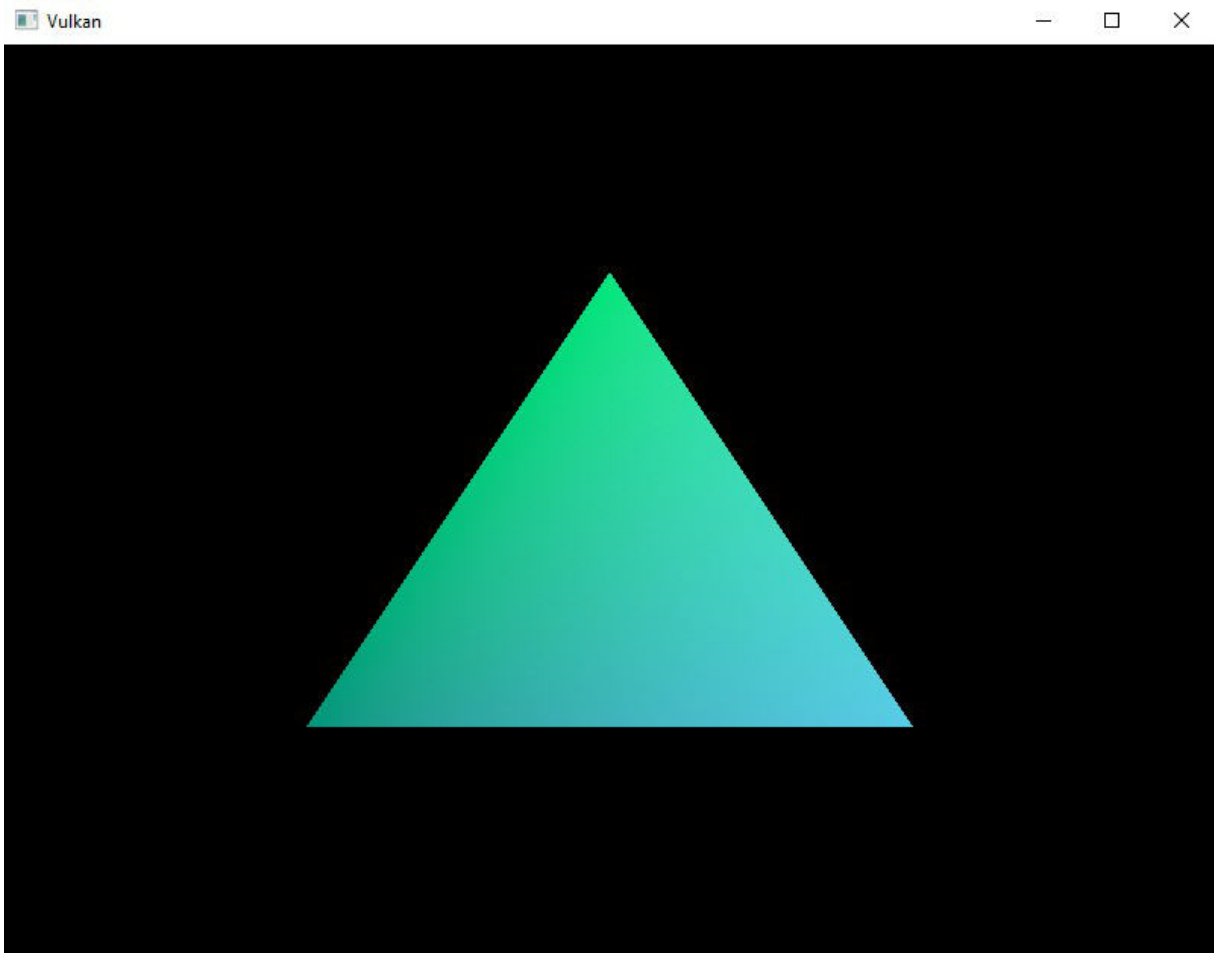
Slika 2.2.4. Prvi primjer modificiranog trokuta

Uredimo boje još jednom. Za sljedeći unos:

```
vec3 colors[3] = vec3[](
    vec3(0.0, 0.8, 0.2),
    vec3(0.1, 0.6, 0.8),
    vec3(0.0, 0.3, 0.2)
);
```

Slika 2.2.5. Drugi primjer modificiranog unosa

dobivamo idući trokut:

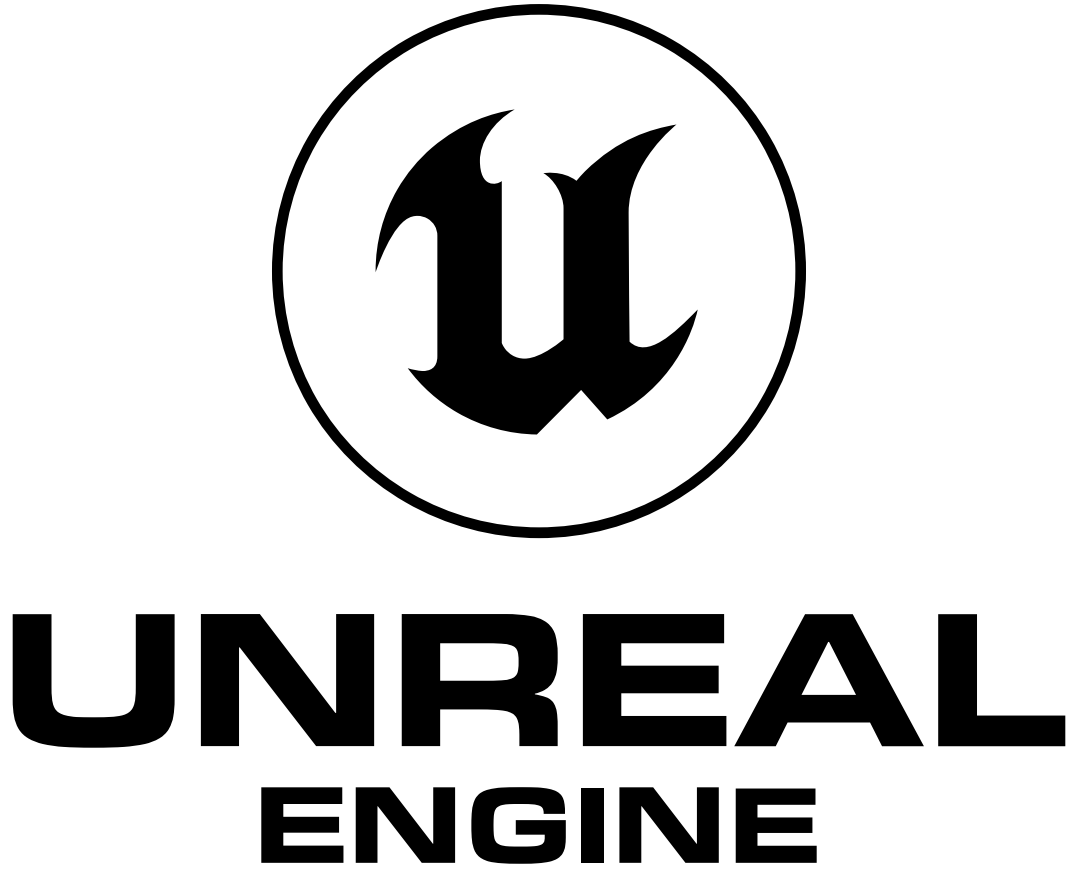


Slika 2.2.6. Drugi primjer modificiranog trokuta

3. Unreal Engine

Unreal Engine (UE) je razvojno okruženje za izradu 3D video igara razvijano od strane Epic Games-a, prvi put prikazano 1998. godine video igrom Unreal koja spada u kategoriju first-person shootera. Izvorno razvijan za PC upravo zbog first-person shootera, ali kroz vrijeme se koristio za svakakve kategorije video igara i pridobile su ga i druge industrije, poput kinematografije i televizijske industrije. UE je napisan u C++ programskom jeziku, sadrži veliku razinu portabilnosti i podržava mnoge platforme, kao što su računala, mobiteli, konzole i platforme virtualne stvarnosti. Važno je napomenuti da UE nije slobodni softver otvorenog koda, već je „source available".

Zadnja generacija, Unreal Engine 5, je izbačena u travnju 2022. godine. Source code je dostupan na GitHub-u, a komercijalna uporaba je dozvoljena na royalty modelu. Epic Games se odriče svoje marže zarade sve dok kompanije koje razvijaju video igru koristeći Unreal Engine ne zaradi preko milijun (1 000 000) američkih dolara prihoda, a naknada se ne plaća ako svoju igru objave na Epic Games Storeu [8]



Slika 3.1. - Unreal Engine logotip

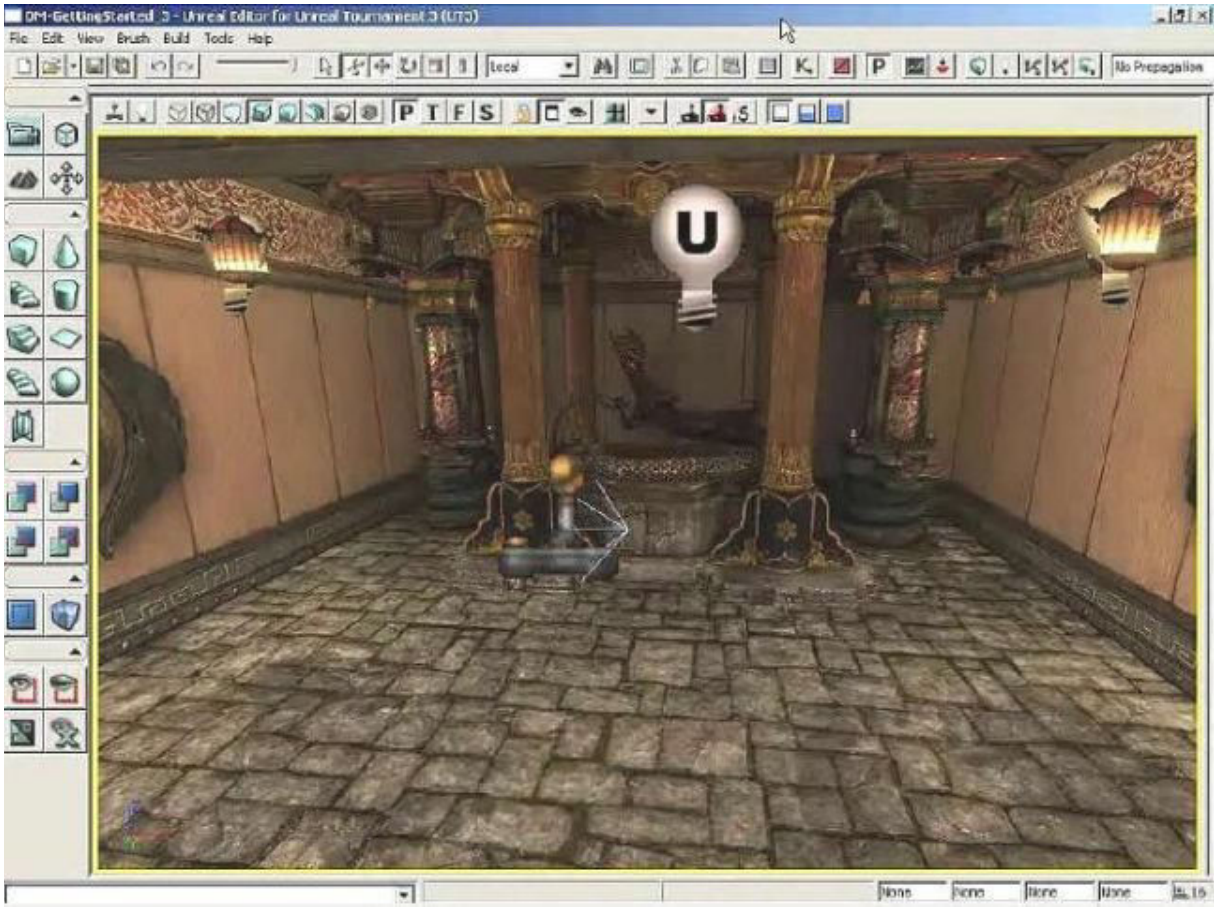
3.1. Povijest Unreal Enginea

Prvu generaciju UE-a je razvio Tim Sweeney, osnivač Epic Gamesa. Izradom uređivačkih alata za svoje shareware igre ZZT iz 1991. godine i Jill of the Jungle iz 1992. godine, Sweeney je počeo pisati razvojno okruženje 1995. godine za razvoj video igre koja će kasnije postati first-person shooter zvan Unreal. Nakon nekoliko godina razvoja, UE je debitirao zajedno sa video igrom 1998. godine, iako su kompanije MicroProse i Legend Entertainment imale pristup toj tehnologiji puno prije, licencirajući je 1996. godine. Na jednom od intervjua, Sweeney je izjavio da je sam napisao 90% koda UE-a, uključujući grafiku, alate i mrežni sustav.

UE se prvotno oslanjao na softversko izvođenje, sve grafičke izračune je vršio procesor (CPU). Nakon nekog vremena je počeo koristiti mogućnosti diskretnih grafičkih procesora, fokusirajući se na Glide API koji je specifično dizajniran za 3dfx akceleratore. Iako su OpenGL i Direct3D bili podržani, performanse su bile sporije u usporedbi sa Glide-om zbog njihovih nedostataka oko upravljanja teksturama u to vrijeme. Sweeney je kritizirao kvalitetu OpenGL drivera za korisnički hardver, opisujući ih kao ”ekstremno problematične, s greškama i neprovjerene”, a kod u implementaciji je označio kao „zastrašujući” naspram jednostavnije i preglednije podrške za Direct3D. Što se tiče zvuka, Epic Games je koristio Galaxy Sound System, softver koji je pisan u assembly-u koji je integrirao Enviromental Audio Extensions (EAX) [9] i Aureal tehnologije i dozvoljavao upotrebu modularne muzike, što je dalo dizajnerima razina fleksibilnost u tome kako se zvučni zapis video igre izvodi na različitim razinama u igri. Za umjetnu inteligenciju je bio zadužen Steve Polge, autor Reaper Bots dodatka za Quake, koji je svoje znanje razvio u IBM-u dizajnirajući protokole usmjerivača. Neke od značajki UE-a su detekcija kolizija, rasvjeta u boji, i limitirana izvedba filtiranja tekstura. Također ima integrirani uređivač razina, UnrealEd, koji je imao podršku za operacije konstruktivne solidne geometrije u stvarnom vremenu već 1996. godine, dopuštajući kartografima da mijenjaju dizajn razina u toku izrade.

Igra Unreal je bila hvaljena za svoje grafičke inovacije, ali Sweeney je priznao u intervju 1999. godine sa časopisom Eurogamer da puno aspekata igre su bili nedovršeni, navodeći pritužbe igrača o visokim sistemskim zahtjevima i problemima s igranjem na mreži. Epic Games se pozabavio ovim pritužbama tokom razvoja Unreal Tournamenta implementirajući nekoliko poboljšanja u razvojnom okruženju koji su bili usmjereni otpimizaciji performansi na slabijim računalima i poboljšanju mrežnog koda, istovremeno rafinirajući umjetnu inteligenciju botova za prikaz koordinacije u timskim modovima igre kao što je „Capture the Flag”. Originalno planirano kao dodatak za Unreal, igra je izašla sa poboljšanom kvalitetom slike sa podrškom za S3 Texture Compression (S3TC) [10] kompresijski algoritam, dozvoljavajući 24 bitne teksture visoke rezolucije bez kompromisa za performanse. Uz to što je UE bio dostupan za Windows, Linux, Mac i Unix OS-ove, razvojno

okruženje se implementiralo i na PlayStation 2 preko Unreal Tournamenta i na konzolu Dreamcast uz pomoc kompanije Secret Level, danas znane kao Sega Studios San Francisco.



Slika 3.1.1. - Prva verzija UE-a koju je razvio Tim Sweeney

Druga generacija UE-a (Unreal Engine 2), je izbačena 2002. godine sa besplatnom multiplayer shooter igrom „America's Army". Iako je UE2 bio baziran na prethodnoj generaciji, vidjeli su se veliki napredci u izvođenju i dostupnim alatima. Mogao je vrtjeti razine 100 puta detaljnije u onih u Unrealu, i razvojno okruženje je integriralo puno značajki, uključujući alat za kinematografsko uređivanje, particle systems (sustave čestica), dodatke za izvoz za 3D Studio Max i Mayu, te sustav skeletne animacije koji je prvi put prikazan u verziji Unreal Tournamenta za Playstation 2. Korisničko sučelje za UnrealEd je bilo ponovno napisano u C++ pomoću wxWidgets alata, za koji je Sweeney izjavio da je to bila „najbolja dostupna stvar" u to vrijeme. Uz korištenje Karma razvojnog okruženja za fiziku, softvera treće strane od kompanije Math Engine, Epic Games je pokretao fizičke simulacije kao što su ragdoll sudari igrača i proizvoljne dinamike krutog tijela. Sa Unreal Tournamentom 2004, igrivost temeljena na vozilima je uspješno implementirana, omogućujući borbe velikih razmjera.

Treća generacija (Unreal Engine 3) je predstavljena srpnja 2004. godine, u doba kada je razvojno okruženje bilo u razvoju preko 18 mjeseci. UE3 je i dalje bio temeljen na prvog generaciji, ali je sadržavao nove značajke. „Osnovne odluke arhitekture vidljive programerima od strane objektno-orijentiranog dizajna, skriptiranje temeljeno na podatcima i dosta modularni pristup podsistemima su dalje sadržani (od UE1). Ali dijelovi igre koje su vidljive igračima, kao što su izvoditelj grafike, sustavi fizike, zvuk i alati, su posve novi i dramatično jači" – izjavio je Sweeney. Za razliku od UE2, koji je i dalje podržavao cjevovod fiksne funkcije, UE3 je bio dizajniran da iskoristi potpuno programibilni hardver shadera. Sva svjetlost i kalkulacije sjena su se radile po pikselu, umjesto po vršnim točkama. Što se tiče izvođenja grafike, UE3 je podržavao gamma-correct izvoditelj grafike visokog dinamičkog raspona. Prve igre koje su izdane koristeći UE3 su bile Gears of War za Xbox360 i RoboBlitz za Windows OS, obje su izašle 7. studenog 2006. godine. Kasnije je dodana podrška za iOS i Adobe Flash Player 11.

Kolovoza 2005. godine, Mark Rein, podpredsjednik Epic Gamesa je otkrio da je Unreal Engine 4 u razvoju već dvije godine. Za C&VG je izjavio „Ljudi nisu svjesni, ali, već dvije godine razvijamo UE4. Još nemamo cijeli tim za razvoj, nego to radi jedan čovjek za kojeg već možete pogoditi tko je on". Sweeney je rane 2008. godine izjavio da je on sam radio na UE4, ali je potvrdio da će se njegov odjel za istraživanje i razvoj proširiti kasnije te godine, razvijajući UE4 paralelno sa UE3. U veljači 2012. godine, Rein je izjavio „ljudi će se šokirati kasnije ove godine kada vide Unreal Engine 4". Epic Games je otkrio UE4 maloj publici na 2012 Game Developers Conference, a video koji prikazuje razvojno okruženje demonstrirano od strane tehničkog umjetnika Alana Willarda je pušten u javnost 7. lipnja 2012 preko GameTrailers TV-a. Jedna od planiranih značajki za UE3 je bila globalna iluminacija u stvarnom vremenu koristeći voxel cone tracing, eliminirajući rasvjetu koja je izračunata unaprijed. Međutim, ova značajka, znana kao Sparse Voxel Octree Global Illumination (SVOGI) i prikazana preko Elemental dema, je zamijenjena sličnim i manje skupim algoritmom zbog brige o performansama. UE4 sadrži nove nacrte vizualno skriptiranog sistema (nasljednika Kismeta korištenog u UE3), koji je dozvoljavao brži razvoj logike video igre bez pisanja koda, rezultirajući manjim podjelama između umjetnika, dizajnera i programera.

Četvrta generacija (Unreal Engine 4) je izbačena 19. ožujka 2014. godine na Game Developers Conference (GDC) preko novog modela licenciranja. Za mjesečnu naknadu od 19 američkih dolara, programeri su dobili cijelu verziju razvojnog okruženja, uključujući i C++ izvorni kod, koji se mogao preuzeti sa GitHub-a. Svaki izbačeni proizvod se naplaćivao sa iznosom od 5% bruto prihoda. Prva igra napravljena sa UE4 je bila Daylight, koja je izrađena sa ranim pristupom razvojnom okruženju i izbačena 29. travnja 2014. godine. Epic Games je 9. rujna 2014. godine izdao UE4 školama i sveučilištima za besplatno, uključujući i besplatne kopije za studente uključene u ovlaštene programa razvoja video igara, informatike,

umjetnosti, arhitekture, simulacije i programa vizualizacije. Također, otvoren je i Unreal Engine Marketplace za skidanje dodatnih sredstava za izradu video igre. Unreal Dev Grants projekt je izbačen 19. veljače 2015. godine, čime je Epic Games nudio 5 milijuna američkih dolara sredstva potpore za kreativne projekte koji su koristili UE4. U ožujku 2015. godine, UE4, zajedno sa svim ažuriranjima je postao potpuno besplatan za sve korisnike. Virtualna realnost koristeći Oculus VR i Oculus Rift, zajedno sa video igrom Fortnite su pridonijele dodatnim napredcima samog razvojnog okruženja zbog popularnosti istih. Sa izlaskom Epic Games Store-a u prosincu 2018. godine, Epic Games više ne naplaćuje 5% iznosa od bruto prihoda za izdane video igre, već pokriva sve troškove sa 12% ukupne zarade. Od 13. svibnja 2020. iznos izuzeća od prava povećan je na milijun američkih dolara doživotnog bruto prihoda po proizvodu. Za razliku od prije, UE4 je podržan na skoro svim platformama koje su bile dostupne u to vrijeme. [6]

3.2. Unreal Engine 5

Trenutna generacija (peta generacija) UE-a je Unreal Engine 5. UE5 je otkriven 13. svibnja 2020. godine i podržava sve postojeće platforme zajedno sa konzolama trenutne generacije (PlayStation 5 i Xbox Series X/S). Razvojno okruženje se počelo razvijati 2 godine prije njegove najave. UE5 je izbačen u early access verziji 26. svibnja 2021. godine, a formalno je puna verzija izdana za programere tek 5. travnja 2022. godine.

Jedna od najvažnijih značajki UE5 je Nanite, što je razvojno okruženje koje dozvoljava unos fotografskih materijala visoke razlučivosti u video igre. Tehnologija virtualizirane geometrije Nanite omogućuje Epic Gamesu da iskoristi jednu od svojih akvizicija, Quixel, što je najveća knjižnica fotogrametrije na svijetu, po podatcima iz 2019. godine. Cilj UE5 je bilo napraviti razvojno okruženje koje je programerima lagano za korištenje tako da mogu stvarati detaljne svjetove u video igrama bez trošenja puno vremena na razvoj novih detaljnih asseta (imovina). Nanite može uvesti skoro bilo koji trodimenzionalni prikaz objekta ili okoliša, uključujući Zbrush ili CAD modele, dozvoljavajući upotrebnu imovina filmske kvalitete. Nanite automatski obrađuje nivoe detalja (LOD) uveženih objekata relativno o ciljnoj platformi i nacrtima udaljenosti, zadatak koji bi inače umjetnik morao sam izvoditi.

Lumen je druga komponenta opisana kao „potpuno dinamička solucija globalne iluminacije koja odmah reagira na scenu i promjene svjetlosti". Lumen eliminira potrebu za umjetnicima i programerima u svrhu kreiranja svjetlosne mape za danu scenu, nego sam izračunava refleksiju svjetlosti i sjena u hipu, dozvoljavajući ponašanje svjetlosnih izvora u stvarnom vremenu.

Virtual Shadow Maps je komponenta dodana u UE5 i opisana kao „nova metoda kartografije sjena korištena za dostavu konzistentnih i visoko razlučivih sjena koje rade skupa sa izvorima filmske kvalitete i velikih, dinamično osvijetljenih otvorenih svjetova". Virtual Shadow Maps se razlikuju od običnih kartografija sjene u svojoj implementaciji , zbog

korištenja sjena ekstremno visoke razlučivosti sa više detalja i nedostataka „shadow poppinga" unutra i van koji se mogu naći u čestim tehnikama kartografije sjena zbog kaskade samih sjena. Dodatne komponente uključuju Niagaru za dinamike fluida i čestica, te Chaos za razvojno okruženje fizike.

Sa potencijalima od desetke milijardi poligona prisutnih na jednom ekranu 4K razlučivosti, Epic Games je razvio UE5 da iskoristi nadolazeća rješenja pohrane velikih brzina sa sljedećom generacijom konzola koje koriste mješavinu RAM-a i prilagođenih SSD-ova. Epic Games je blisko surađivao sa Sony-em u optimizaciji UE5 za PlayStation 5, te je Epic Games kolaborirao sa Sony-em na arhitekturi pohrane te konzole. Da bi demonstrirali kako je lagano kreirati detaljan svijet sa što manje napora, u svibnju 2020. godine je otkriven demo „Lumen in the Land of Nanite" koji se izvodio na Playstation 5 konzoli i većinom je stvoren sa izvlačenjem imovina iz Quixel biblioteke, i koristili su se Nanite, Lumen i druge UE5 komponente da kreiraju fotorealističnu postavku špilje koja se mogla istražiti. Epic games je potvrdio da UE5 će biti potpuno podržan na konzoli Xbox Series X, ali se fokusirao na PlayStation 5 tokom najave zbog njihovog prijašnjeg rada sa Sony-em.

Epic Games koristi igru Fortnite kao ispitnu ploču za UE5 da pokažu što razvojno okruženje može doprinijeti industriji, sa prelaskom te igre na UE5 u prosincu 2021. godine. Dodatne planirane značajke za UE5 dolaze od akvizicija i partnerstva od strane Epic Gamesa. The MetaHuman Creator je projekt baziran na tehnologiji razvijenoj od triju kompanija koje je Epic Games stekao, a to su 3Lateral, Cubic Motion i Quixel, što dozvoljava programerima da brzo kreirao realistične ljudske likove koji se onda mogu izvest za korištenje unutar Unreala. Zbog partnerstva sa Cesium-om, Epic Games planira ponuditi besplatan dodatak za pružanje geoprostornih podataka za korisnike Unreal-a, dozvoljavajući im da rekreiraju bilo koji dio mapirane površine Zemlje. Epic Games će uvesti RealityCapture, što je proizvod koji su stekli sa akvizicijom kompanije Capturing Reality, koji može generirati 3D modele bilo kojeg objekta iz kolekcije fotografija slikanih iz više kutova i raznih međuprogramskih alata dostupnih od Epic Game Tools.

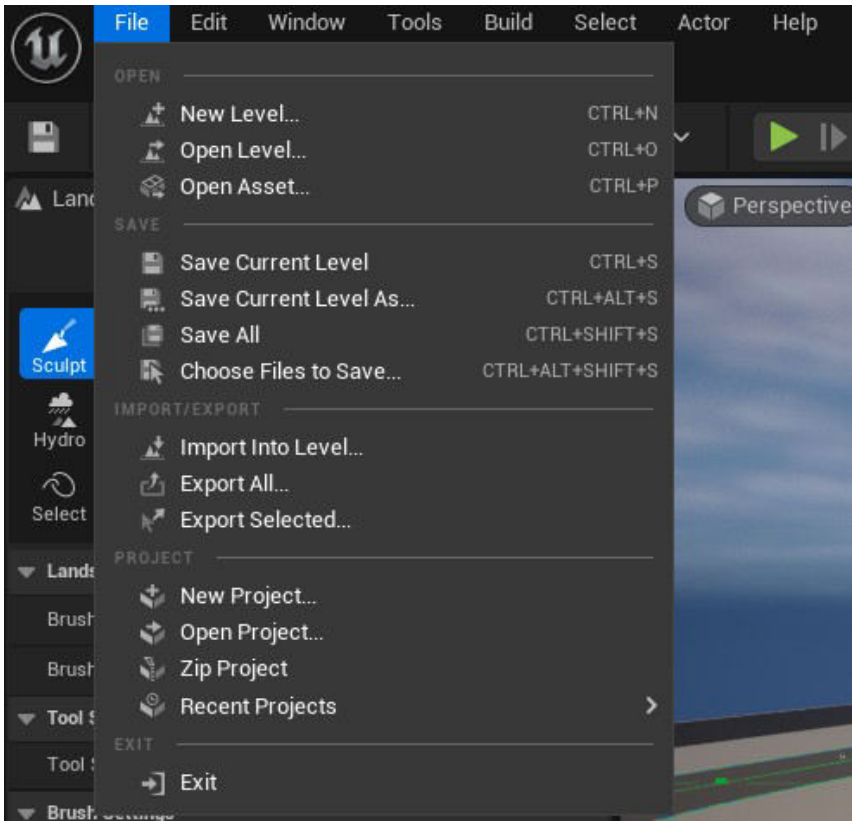
UE5 zadržava svoji trenutni model tantijema, programeri vraćaju 5% vrijednosti svih zarada Epic Games-u, ali se plaćanje ove naknade izbjegava ako se igra izda na Epic Games Store-u. Nadalje, Epic Games je najavio da uz UE5 neće naplaćivati naknadu od video igara koje koriste bilo koju verziju Unreal Engine-a za prvih milijun američkih dolara prihoda, retroaktivno od 1. siječnja 2020. godine. [6]

3.3. Sučelje Unreal Enginea

Opisati ćemo najosnovnije elemente sučelja UE5 (u ovom slučaju verzije 5.2). Pošto je to razvojno okruženje za stvaranje video igara, filmova, itd., sve je podređeno programeru da bude što jednostavnije za koristiti, čak i osobama koje nemaju prijašnjeg iskustva sa izradom istih. Samo okruženje izgleda ovako:

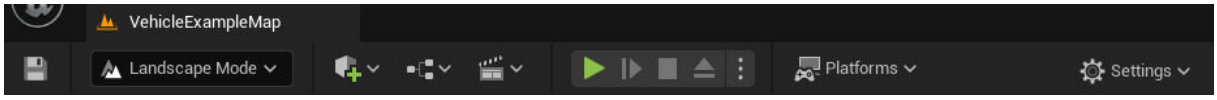
Slika 3.3.1. - Razvojno okruženje UE5

Na samom vrhu se nalazi alatna traka sa glavnim opcijama i padajućim izbornikom vezanih uz sami otvoreni projekt, alate, izdavanje projekata i nivoa, terena kojeg vidimo, glumaca, itd.



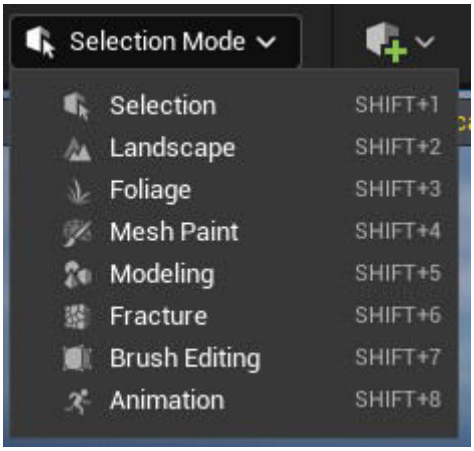
Slika 3.3.2. - Alatna traka sa padajućim izbornikom

Ispod alatne trake nam se nalazi traka za projekte koja nam pokazuje koji nivo je trenutno otvoren sa brzim pristupom glavnim opcijama, spremanjem promjena, pokretanjem samog nivoa, platformama na kojima ćemo emulirati nivo, i postavkama video igre, grafike, itd.



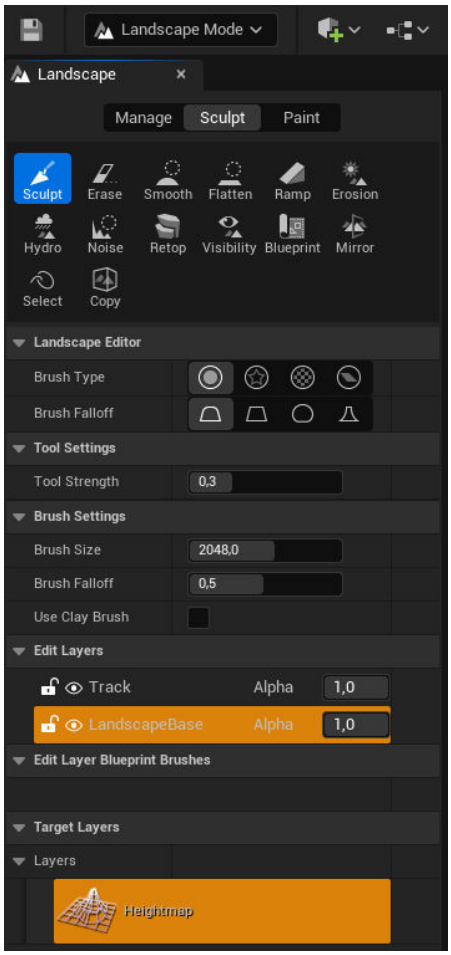
Slika 3.3.3. - Projektna traka

Prva stavka do ikone za spremanje na projektnoj traci nam pruža padajući izbornik na kome možemo birati alate za uređivanje samog svijeta na trenutnom nivou.



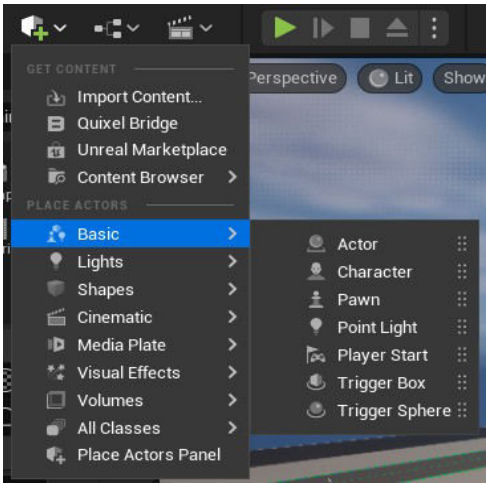
Slika 3.3.4. - Padajući izbornik sa alatima za uređivanje svijeta

Odabirom nekih od tih alata nam se na lijevoj strani ekrana pojavljuje cijeli meni za taj alat. Npr. meni za „Landscape" izgleda ovako:

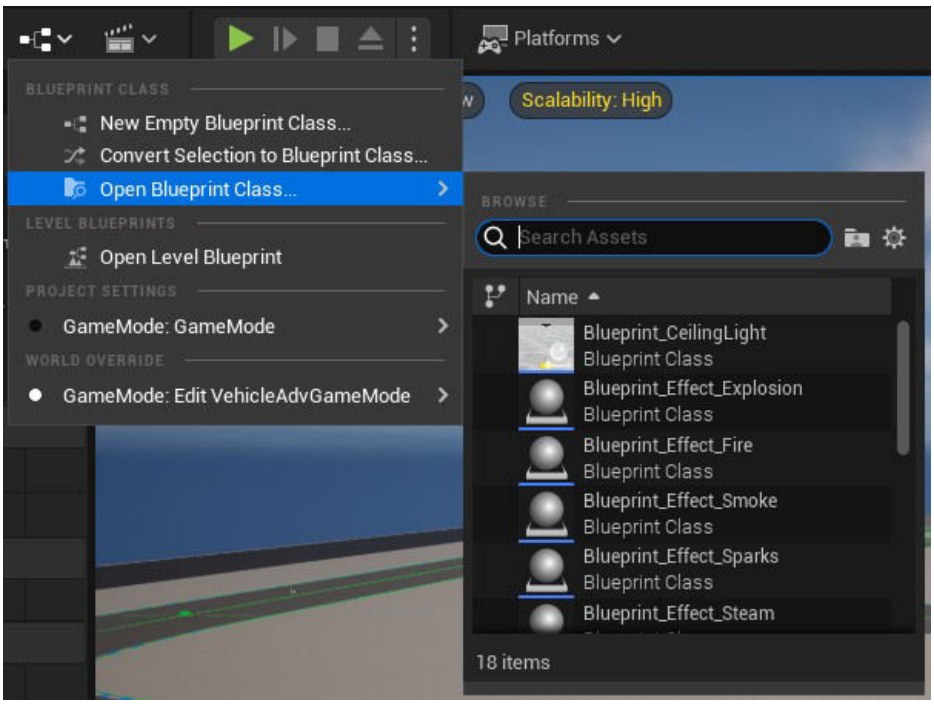


Slika 3.3.5. - Meni Landscape-a

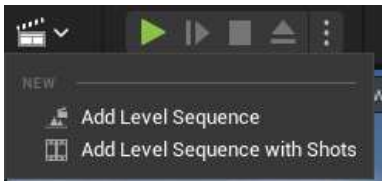
Iduća tri padajuća izbornika pored izbornika alata se odnose na objekte koje dodajemo u scenu, blueprintove za programiranje i animacije za taj teren, i sekvence samog nivoa.



Slika 3.3.6. - Padajući izbornik objekata

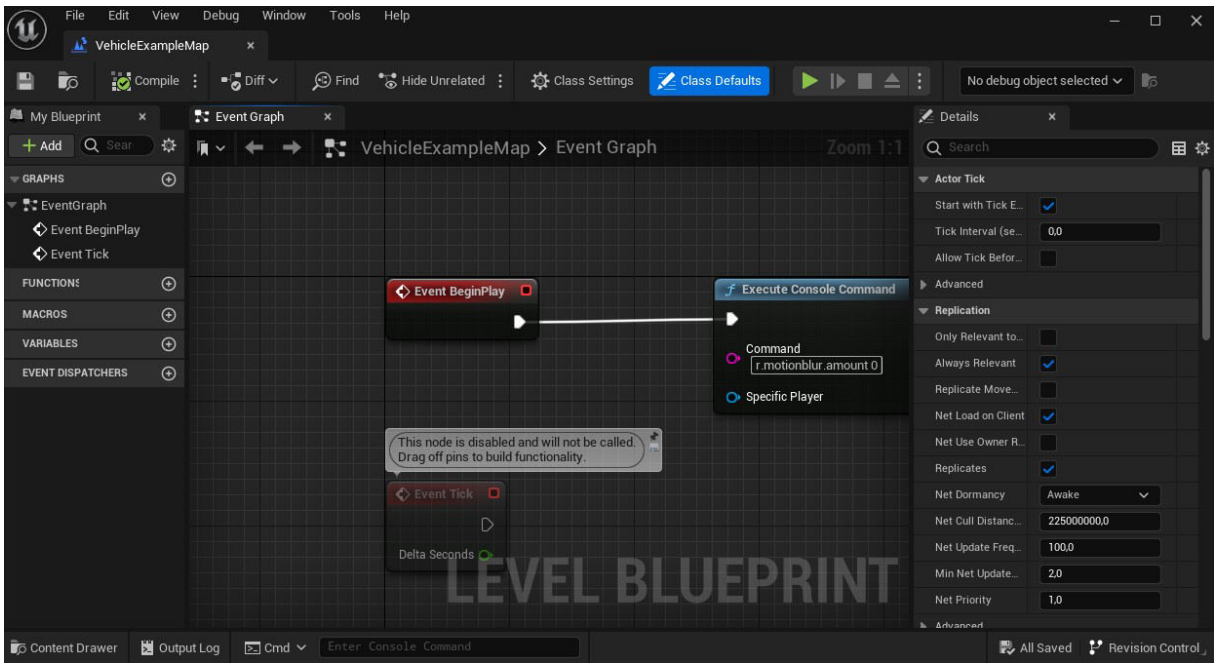


Slika 3.3.7. - Padajući izbornik Blueprintova



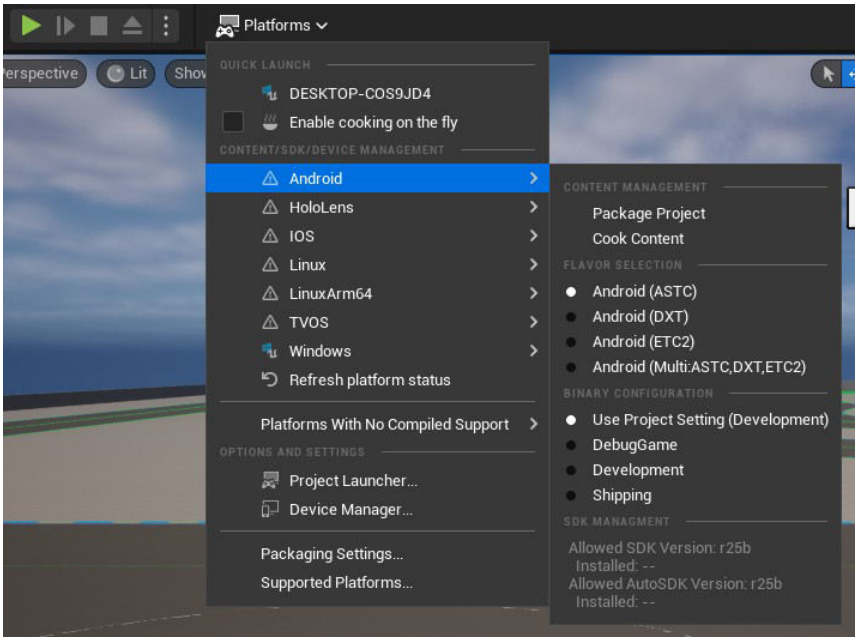
Slika 3.3.8. - Padajući izbornik sekvenca nivoa

Pošto su blueprintovi važan dio razvojnog okruženja zbog njihove namjene, a to je vizualno programiranje spajanjem raznih objekata i ubacivanjem komandi i animacija za iste, prikazati ćemo ih. Iz padajućeg izbornika blueprintova, možemo otvoriti blueprint za cijeli nivo, a on izgleda ovako:



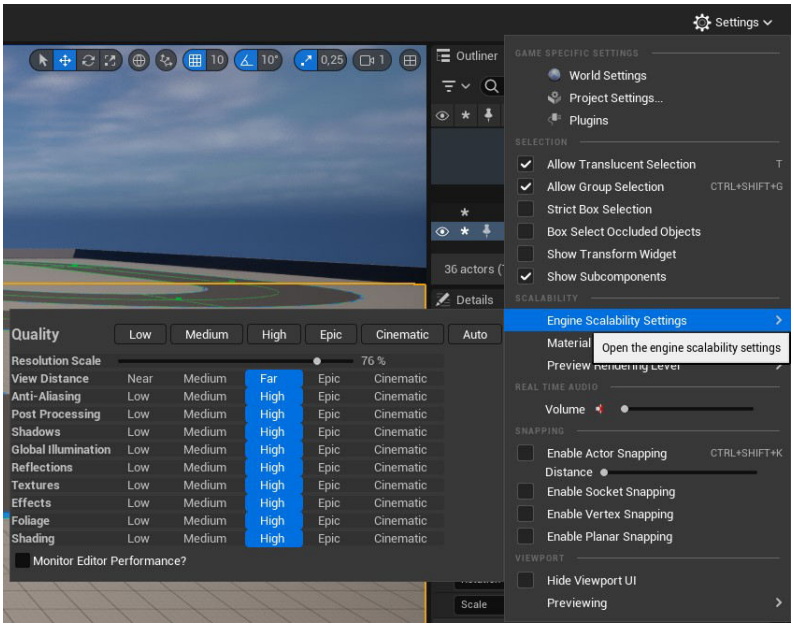
Slika 3.3.9. - Blueprint nivoa

Nakon ta tri padajuća izbornika, dolazimo do dijela do funkcija di pokrećemo, pauziramo ili zaustavljamo sami nivo preko emulacije da ga testiramo i pokraj tih funkcija imamo padajući izbornik na kojem odabiremo platformu na kojoj želimo testirati nivo. To nam također omogućuje da spojimo uređaje sa drugim operativnim sistemima da testiramo kako se sami nivo vrti.



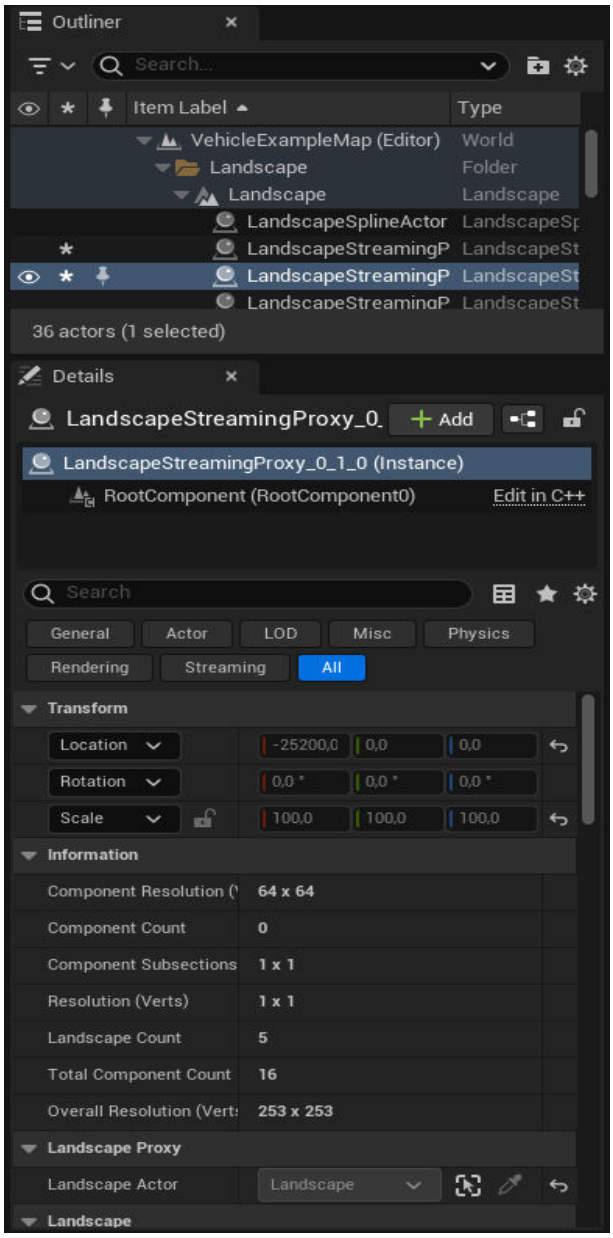
Slika 3.3.10. - Funkcije za pokretanje emulacije i padajući izbornik za platforme

Na samom kraju projektne trake sa desne strane se nalazi padajući izbornik za postavke sa kojima možemo promijeniti postavke projekta, svijeta, dodataka, podesiti samu skalabilnost grafike, kvalitetu materijala, odabrati pretpregled API-a na kojem želimo vidjeti sami nivo, itd.



Slika 3.3.11. - Padajući izbornik postavki

Sa desne strane ekrana razvojnog okruženja nam se nalazi obrisnik na kojemu vidimo popis svih objekata na nivou i ispod toga ovisno o alatu kojeg koristimo nam se pojavljuju dodatne opcije i informacije.



Slika 3.3.12. - Opisnik i dodatne opcije

Na dnu ekrana razvojnog okruženja nam se nalazi traka sa bibliotekom za sadržaj, dnevnikom izlaznih informacija, ulaznim okvirom za naredbe konzole kao i dodatnim kontrolama i informacijama samog razvojnog okruženja.

Slika 3.3.13. - Traka na dnu razvojnog okruženja sa otvorenom bibliotekom za sadržaj

Sama sredina cijelog razvojnog okruženja je prozor koji je uredno i uređivač samog nivoa gdje kreiramo sami nivo korištenjem i pozicioniranjem samih objekata, glumaca, svjetlosti, itd. Na vrhu tog prozora imamo brze opcije gdje možemo podesiti kako se sama scena izvodi i pratiti parametre za istu, odabrati perspektivu kamere, podesiti svjetlost ili tip modela koju želimo trenutno vidjeti na ekranu, odabrati elemente grafike koje želimo prikazati, promijeniti skalabilnost same grafike i još dodatne opcije za poziciju samih objekata, njihov kut nagiba, pozicioniranje sa mrežom ili bez, skaliranje veličine, brzinu kamere, itd.

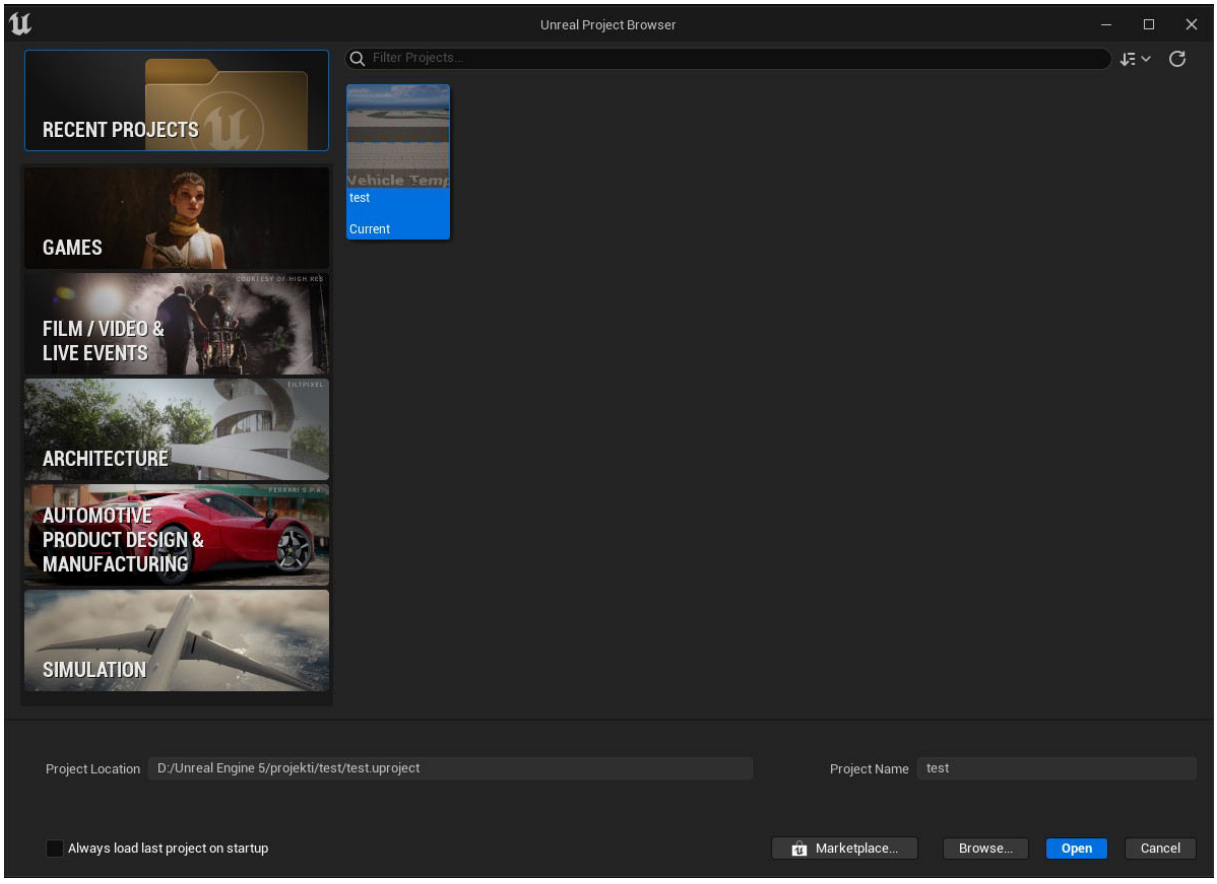
Slika 3.3.14. - Glavni prozor razvojnog okruženja

4. Razvoj višeplatformskih igara korištenjem Unreal Enginea i Vulkana

Kao primjer razvoja višeplatformskih igara korištenjem UE-a i Vulkana, koristiti ćemo „Vehicle Demo” koji je jedan od već dostupnih demo scenarija koji su integrirani u sami UE. Mjeriti ćemo performanse izvođenja same scene u Vulkanu, uspoređivati ga s drugim poznatim standardom DX12 i vidjeti kako možemo taj projekt izdati na različite platforme, tipa Windows i Android.

4.1. Kreiranje projekta

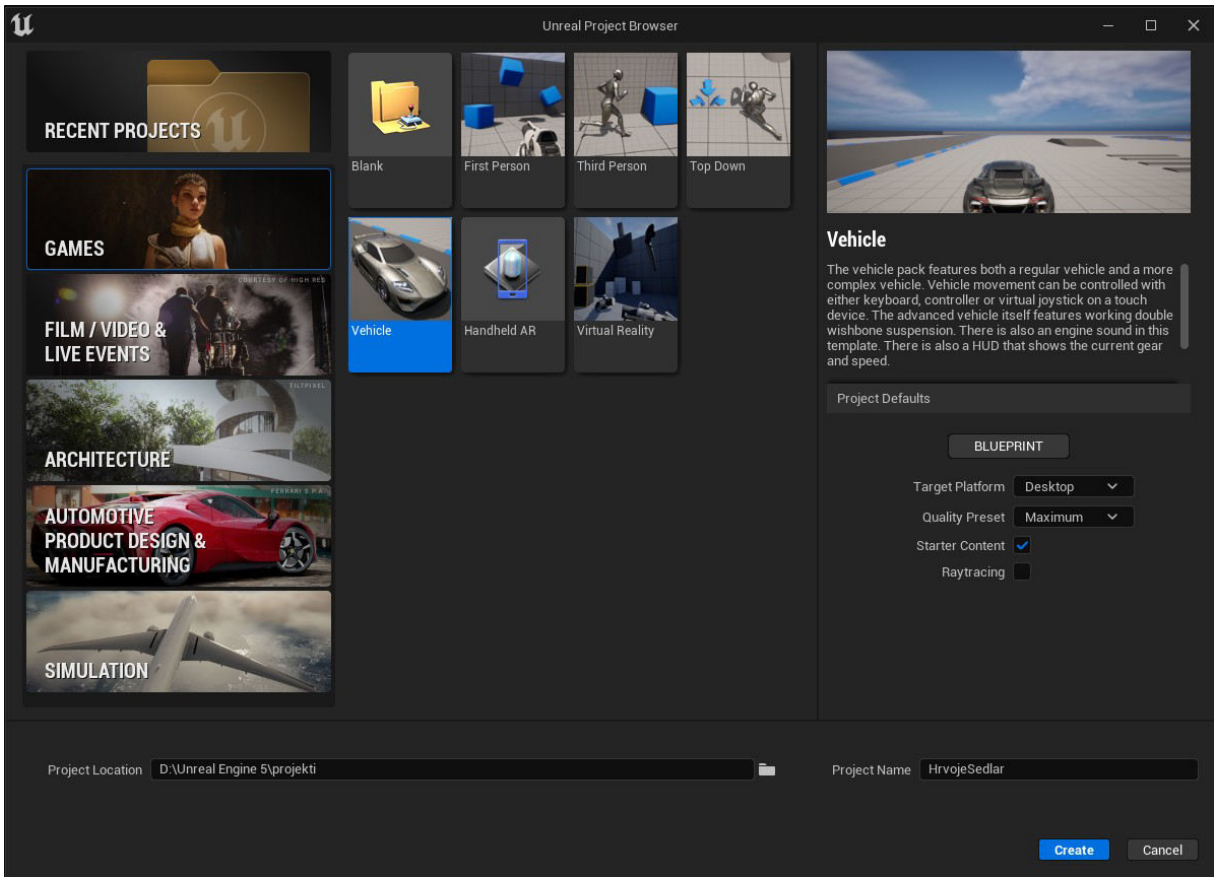
Prvo moramo kreirati projekt. Nakon što pokrenemo UE, pojavljuje nam se Unreal Project Browser prozor na kojem možemo odabrati tip projekta koji želimo napraviti (video igra, film/video i žive događaje, arhitektura, automobilistički dizajn i razvoj, simulaciju) ili možemo nastaviti sa projektom koji smo već kreirali.



Slika 4.1.1. - Unreal Project Browser

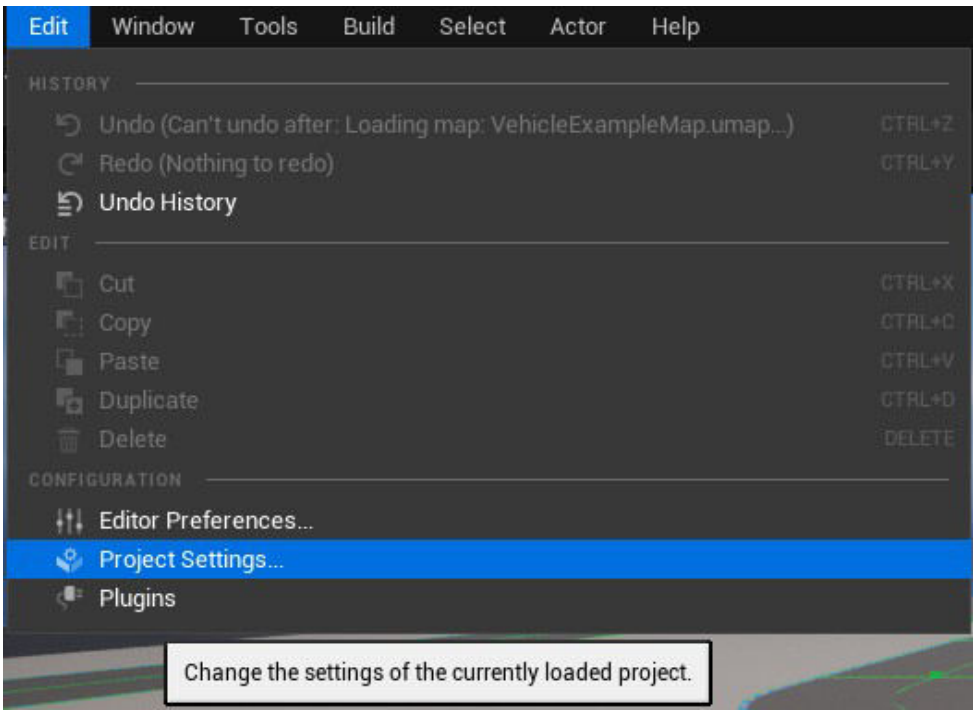
U našem slučaju biramo projekt za video igru, podstavku Vehicle Demo koji je scenarij koji će nam služiti za ovaj primjer, te pod Target Platform odabiremo Desktop platformu, jer

na njoj razvijamo video igru. Pošto sam lično limitiran hardverom grafičkog procesora (Nvidia GTX 1060 3GB iz 2016. godine), izostaviti ćemo opciju za ray tracing (tehnika modeliranja svjetlosnog prijenosa u širokom spektru izvođenja algoritama za generiranje digitalnih slika u 3D kompjuterskoj grafici) [11]. Ostavimo da projekt koristi blueprintove, kvalitetu postavimo na maksimalnu, imenujemo projekt i kreiramo ga.



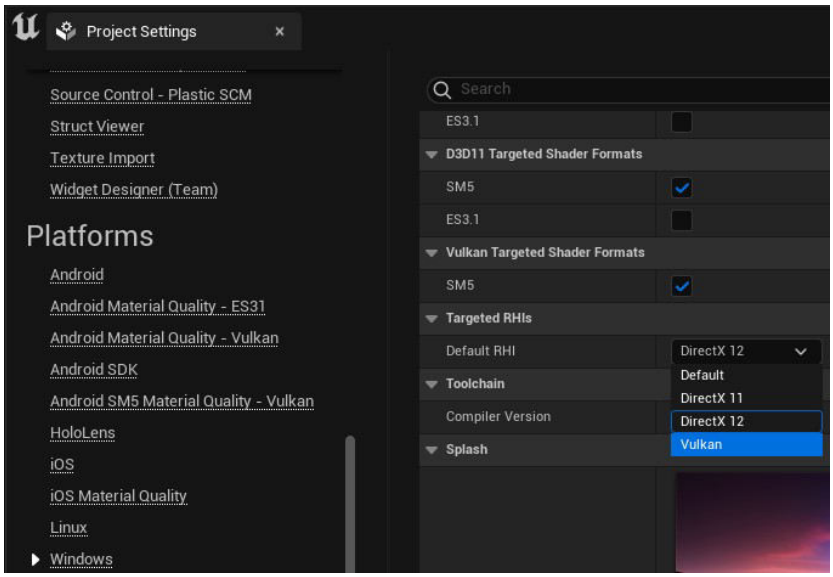
Slika 4.1.2. Kreiranje projekta

Nakon kreiranja projekta, UE će sastaviti shadere i sve ostale stvari koje su mu potrebne da se pokrene. To može potrajati ovisno o jačini hardvera koju koristite. Pred nama se zatim otvara radno okruženje koje smo u potpunosti opisali u prošlom poglavlju. Za postavljanje Vulkan API-a, potrebno je na alatnoj traci pod „Edit" otvoriti „Project Settings", tj. postavke projekta.



Slika 4.1.3. - Otvaranje postavki projekta

Odabirom „Project Settings" nam se pojavi skočni prozor postavki projekta na čijoj lijevoj strani se spuštamo do odjeljka za platforme na kojem odabiremo trenutno platformu na kojoj želimo aktivirati Vulkan API, a to je Windows. Pod „Targeted RHIs" za „Default RHI" umjesto trenutno postavljenog Direct X 12 odabiremo Vulkan. RHI je kratica za „Rendering Hardware Interface" (prikazanje hardverskog sučelja). Također, iznad označimo SM5 za shader pod „Vulkan Targeted Shader Formats".

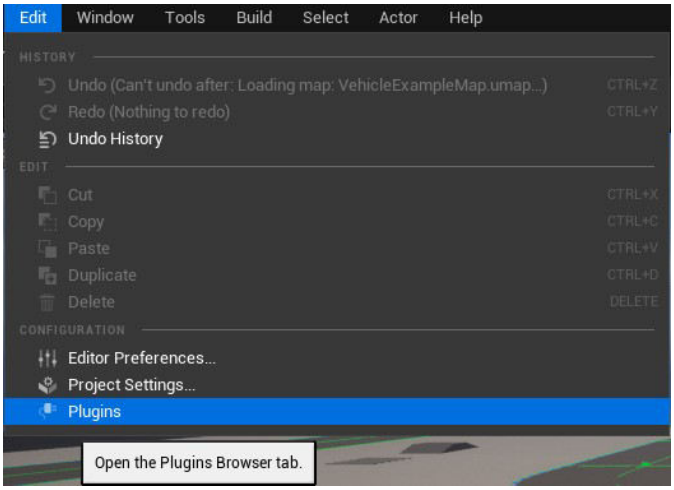


Slika 4.1.4 - Postavljanje Vulkan API-a

UE će nakon toga zahtjevati ponovno pokretanje radnog okruženja da opet sastavi sve shadere zbog promjene API-a.

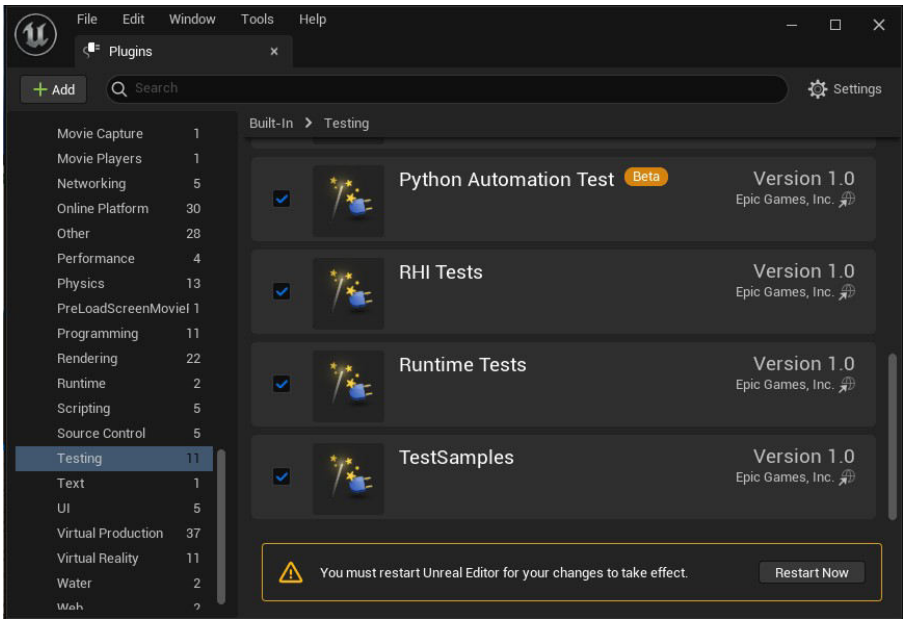
4.2. Testiranje

UE ima mnogo ugrađenih plug-inova (dodataka) i drivera koji nam pomažu pri razvoju video igre. Jedna od važnih stavki je testiranje da vidimo pravilnost rada svih elemenata, kako se koriste resursi i što treba optimizirati da krajnji rezultat bude što bolji. Za aktiviranje dodatka koji testira RHI koji smo postavili, u alatnoj traci pod Edit odaberemo Plugins.



Slika 4.2.1. - Otvaranje dodataka

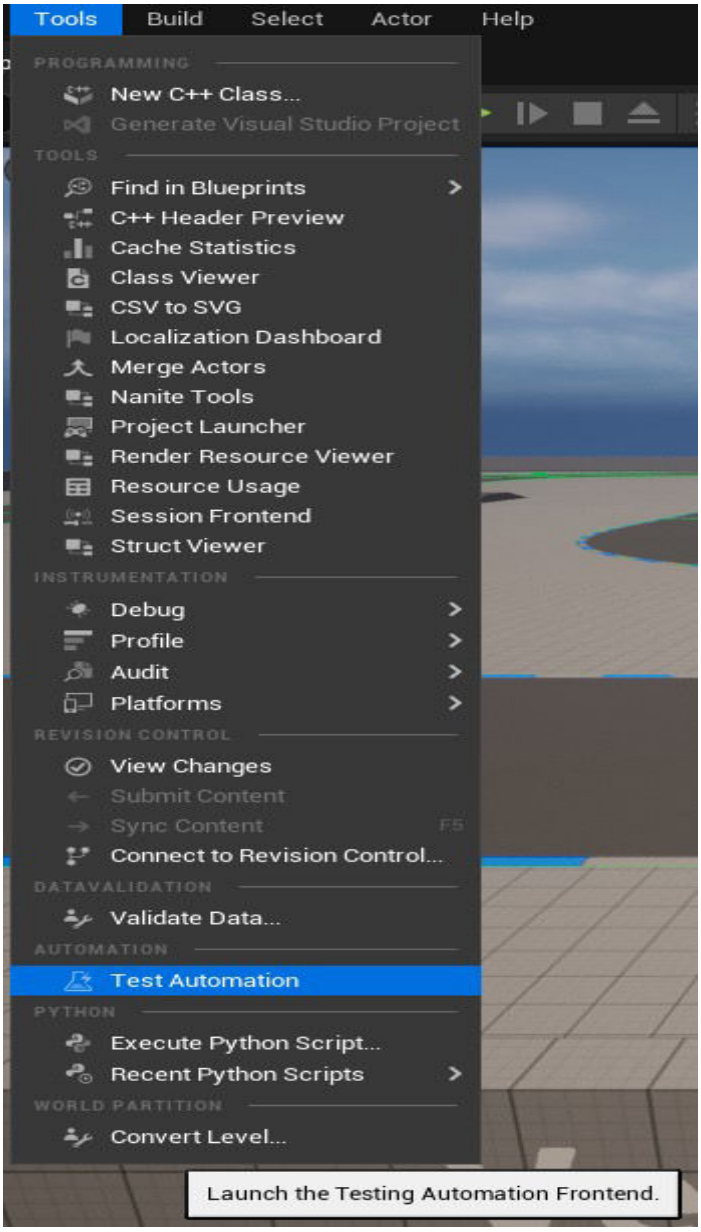
U skočnom prozoru dodataka pod „Built-in" navigiramo do „Testing" i sve dostupne testove aktiviramo kvačicom, te ponovno pokrenemo razvojno okruženje.



Slika 4.2.2. - Aktiviranje opcija za testiranje

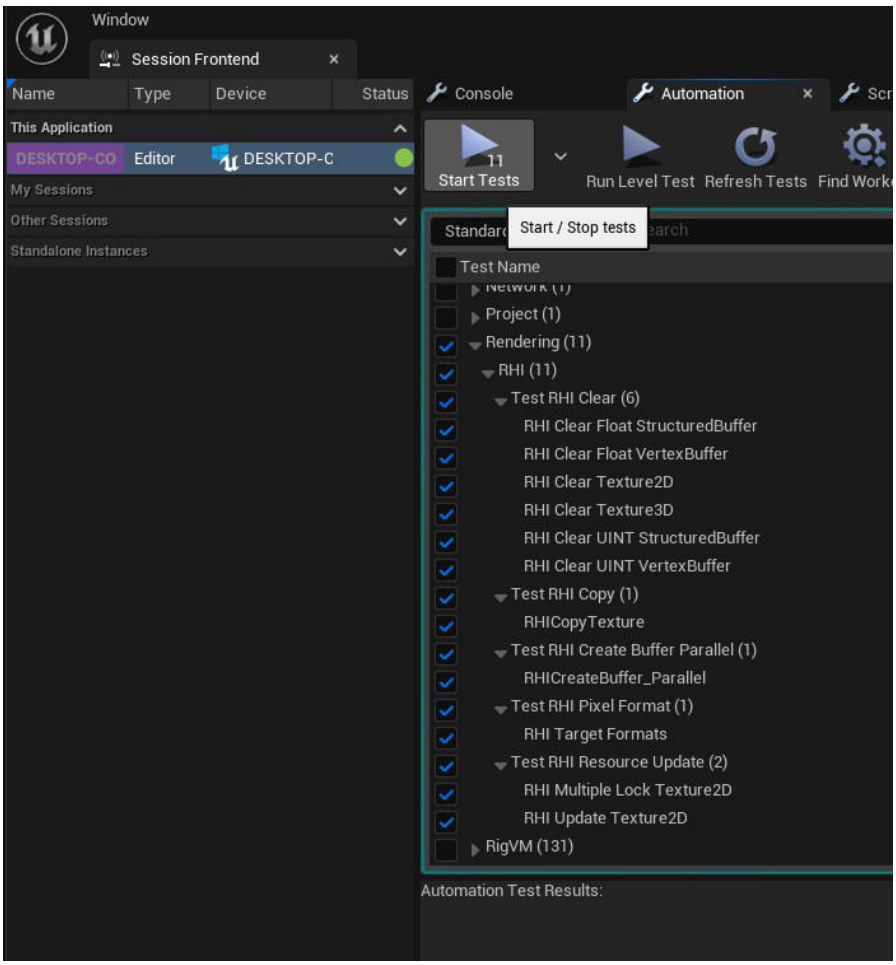
Nakon toga

možemo pokrenuti željne testove. Na alatnoj traci pod Tools odaberemo Test Automation. To će nam otvoriti skočni prozor Test Automation na kojem odabiremo testove koje želimo izvesti.



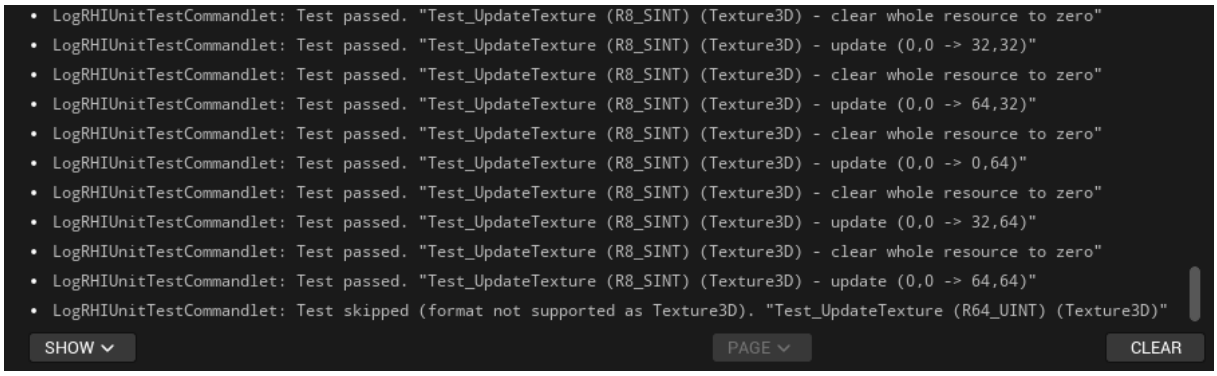
Slika 4.2.3. - Otvaranje Test Automation prozora za testiranje

U sredini prozora koji nam se otvorio spuštamo se do odjeljka Rendering koji označimo kvačicom što istovremeno označava sve funkcije pod njime a to su svi testovi za RHI. Iznad toga nam se nalazi gumb „Start Tests" na kojim zatim kliknemo za pokretanje odabranih testova.

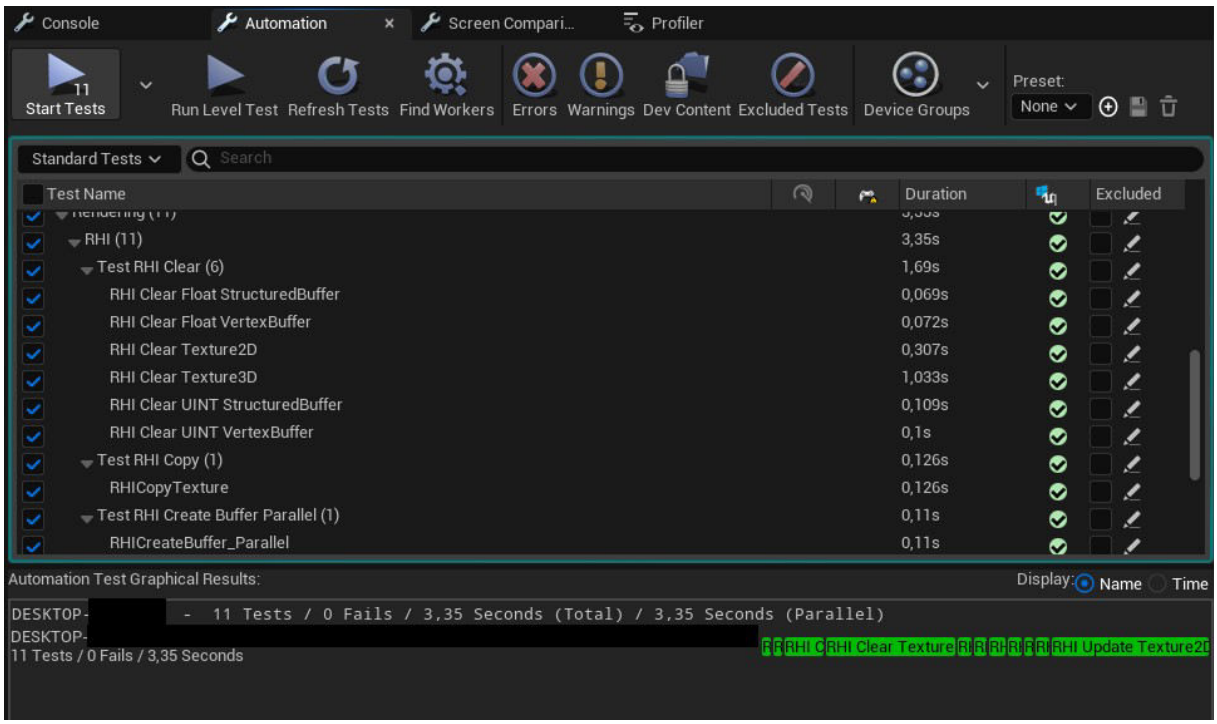


Slika 4.2.4. - Odabir i pokretanje RHI testova

Izvršavanjem testova nam se izbacuje prozor Message Log (Dnevnik Poruka) na kojem vidimo sve testove koji su se izveli i njihovu uspješnost. Povratkom na Test Automation prozor vidimo sažetak tih testova u grafičkom obliku te hardver na kojem su se izveli.

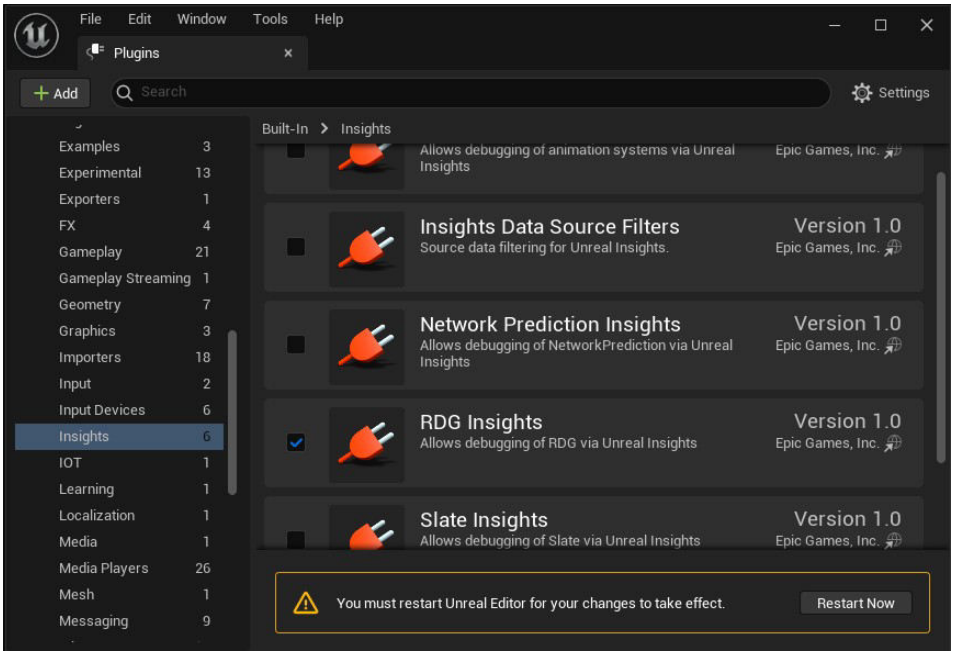


Slika 4.2.5. - Dio Message Log prozora nakon izvršenog testiranja



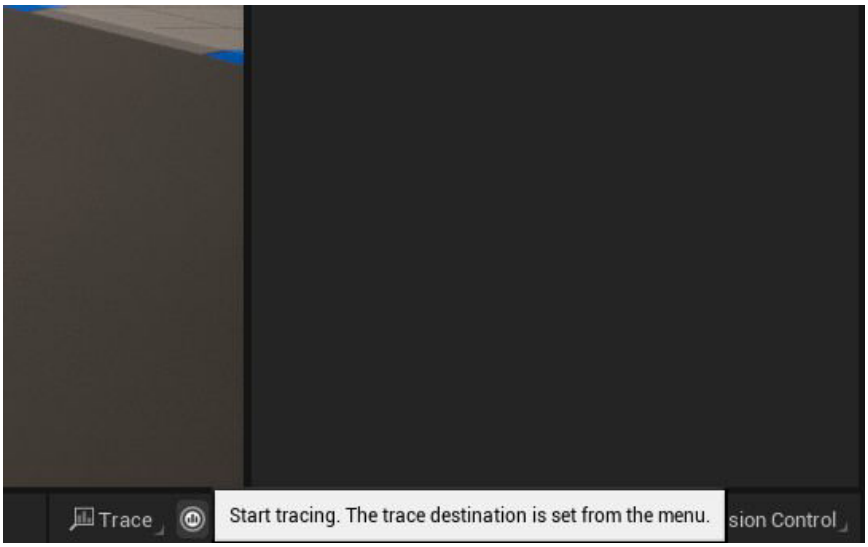
Slika 4.2.6. - Test Automation prozor nakon završenog testiranja

Ako želimo ljepšu i grafičku reprezentaciju toga što nam trenutni hardver radi, moramo aktivirati dodatak imena RDG Insights koji se nalazi u prozoru dodataka pod odjeljkom Insights (Uvidi). RDG je kratica za „Render Dependency Graph", tj. grafikon prikaza ovisnosti. Kao i prije, UE zahtjeva ponovno pokretanje razvojnog okruženja.

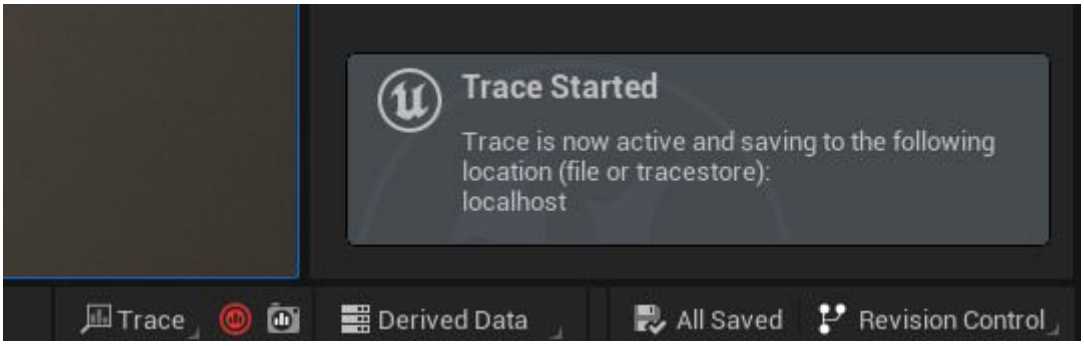


Slika 4.2.7. - Aktiviranje dodatka RDG Insights

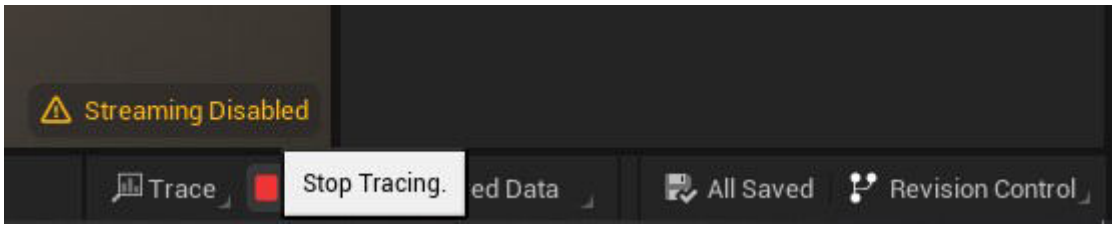
Poslije ponovnog pokretanja, pokrećemo nivo na projektnoj traci te aktiviramo „Start tracing” na traci na dnu razvojnog okruženja pored opcije „Trace” (Praćenje) da UE snimi sve što se događa u pozadini sa pokrenutim procesima dok se nivo odvija. Možemo se igrati sa nivoom dok se praćenje odvija u pozadini i kad odlučimo da je dovoljno snimljeno, prekinemo ga da nam se generira datoteka sa zapisom praćenja.



Slika 4.2.8. - Aktiviranje opcije Start tracing/praćenja

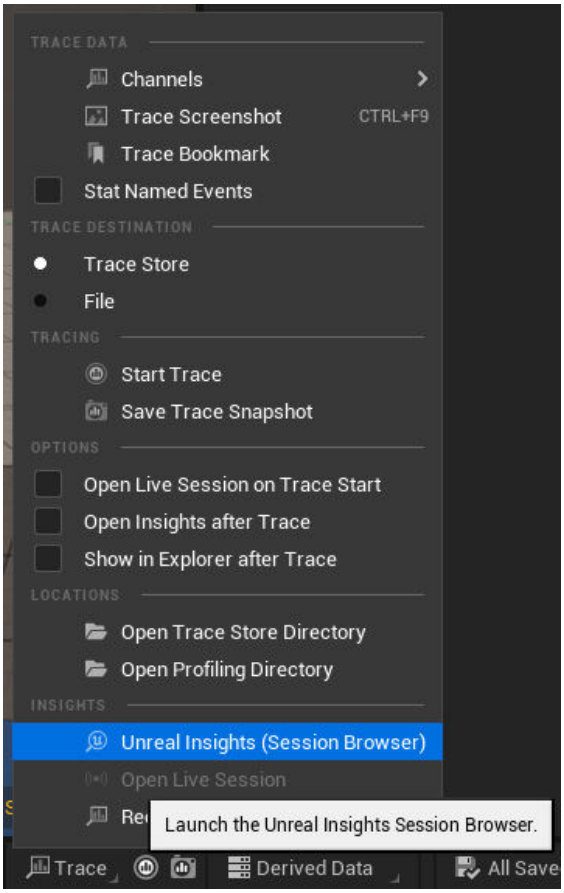


Slika 4.2.9. - Poruka da se praćenje izvodi u pozadini



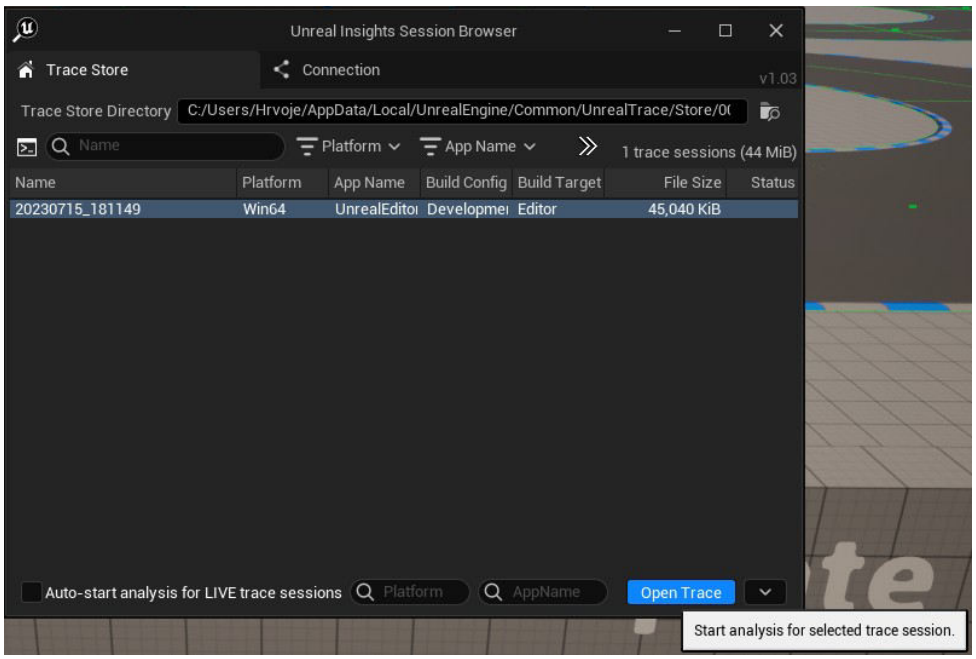
Slika 4.2.10. - Prekidanje praćenja

Nakon snimljenog praćenja, sada možemo pregledati koji parametri su zabilježeni. Za pristupanje grafikonu praćenja pod opcijom Trace na dnu u desnom kutu razvojnog okruženja otvaramo prozor Unreal Insights (Session Browser).



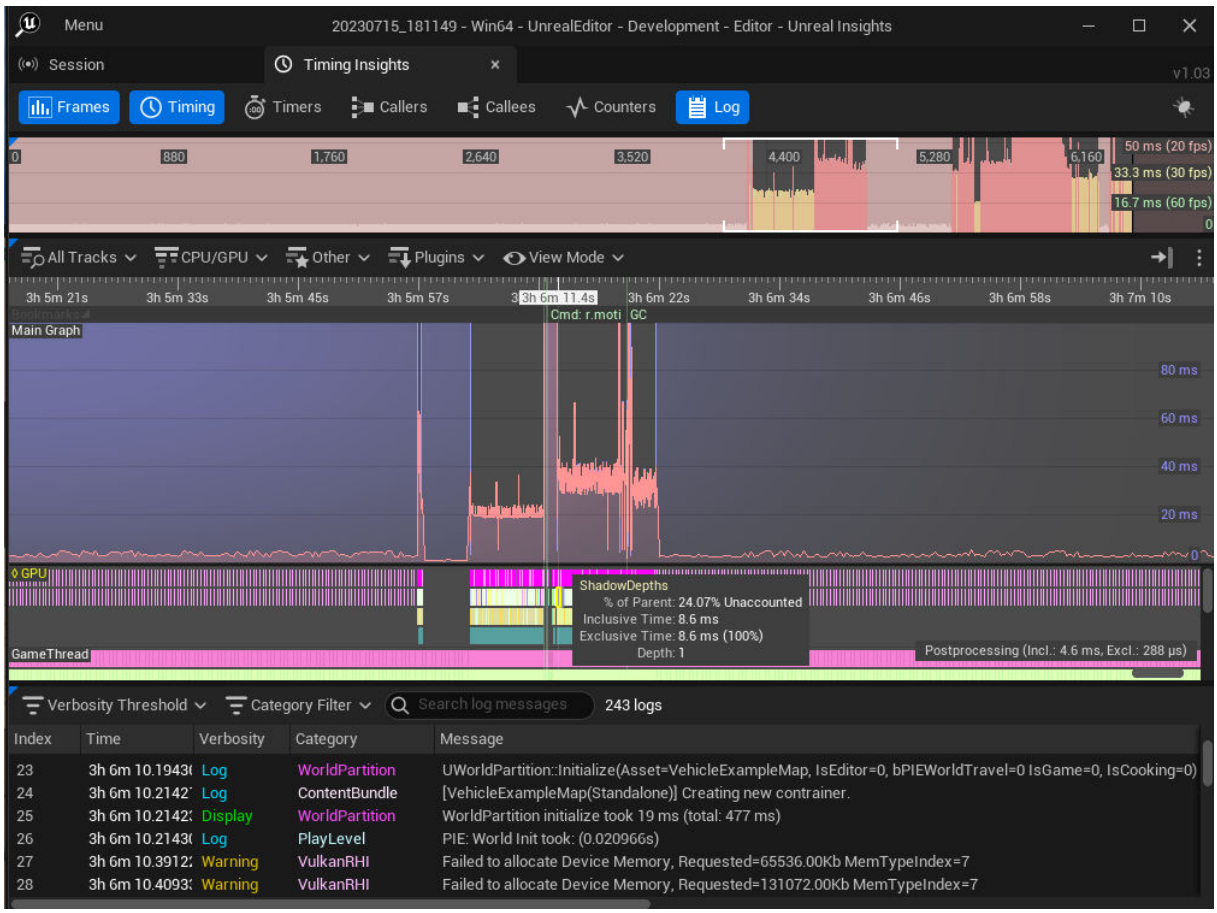
Slika 4.2.11. - Otvaranje Unreal Insights prozora

U Unreal Insights Session Browser-u odabiremo snimljenu datoteku praćenja i otvaramo je na gumb „Open Trace”.

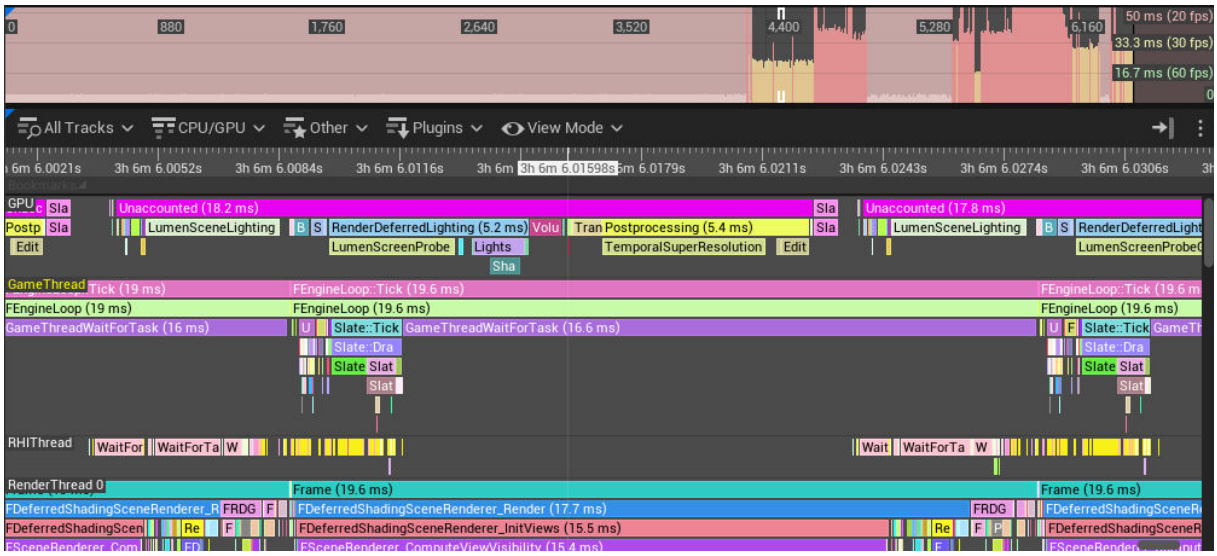


Slika 4.2.12. - Otvaranje datoteke praćenja

Prijašnjom aktivacijom dodatka RDG Insights, dobili smo mogućnost praćenja svakog parametra hardvera u obliku grafikona tokom praćenja procesa nivoa. To razvojnim programerima omogućava jasno čitanje mogućih problema koji se mogu pojaviti (npr. neuspješno lociranje memorije), što pridonosi lakšem rješavanju grešaka tokom razvoja. Grafikon u Unreal Insights-u nam točno govori koji dio se kada izvodio, a zumiranjem možemo točno razaznati u milisekunde koju funkciju je hardver tada izvršavao. Zapisnik poruka na dnu prozora nam također pokazuje svaku instancu u pismenom obliku.



Slika 4.2.13. - Unreal Insights prozor sa RDG Insights grafikonom

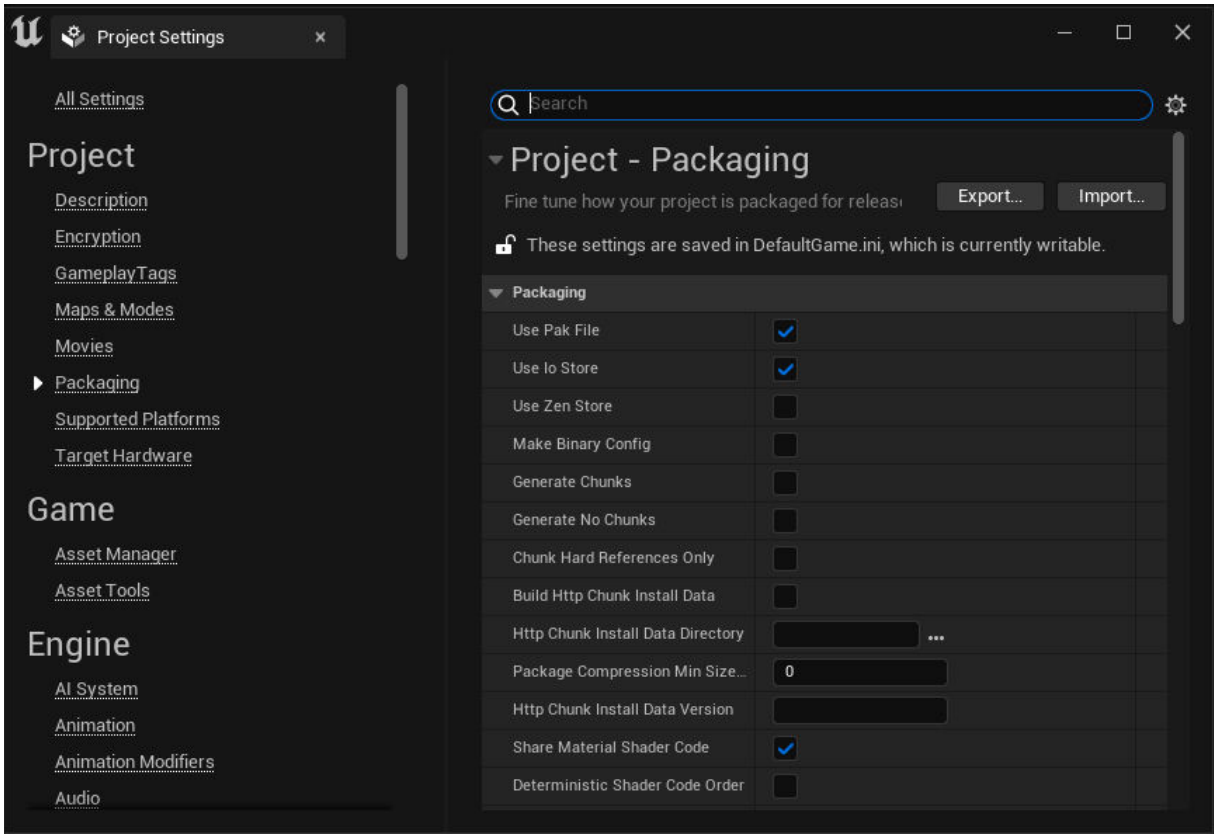


Slika 4.2.14. - Zumirani RDG Insights grafikon

Više o svemu što smo naveli o testiranju se može pročitati na web sjedištu dokumentacije Unreal Engine-a (u ovom slučaju verzije 5.2). [12] [13] [14]

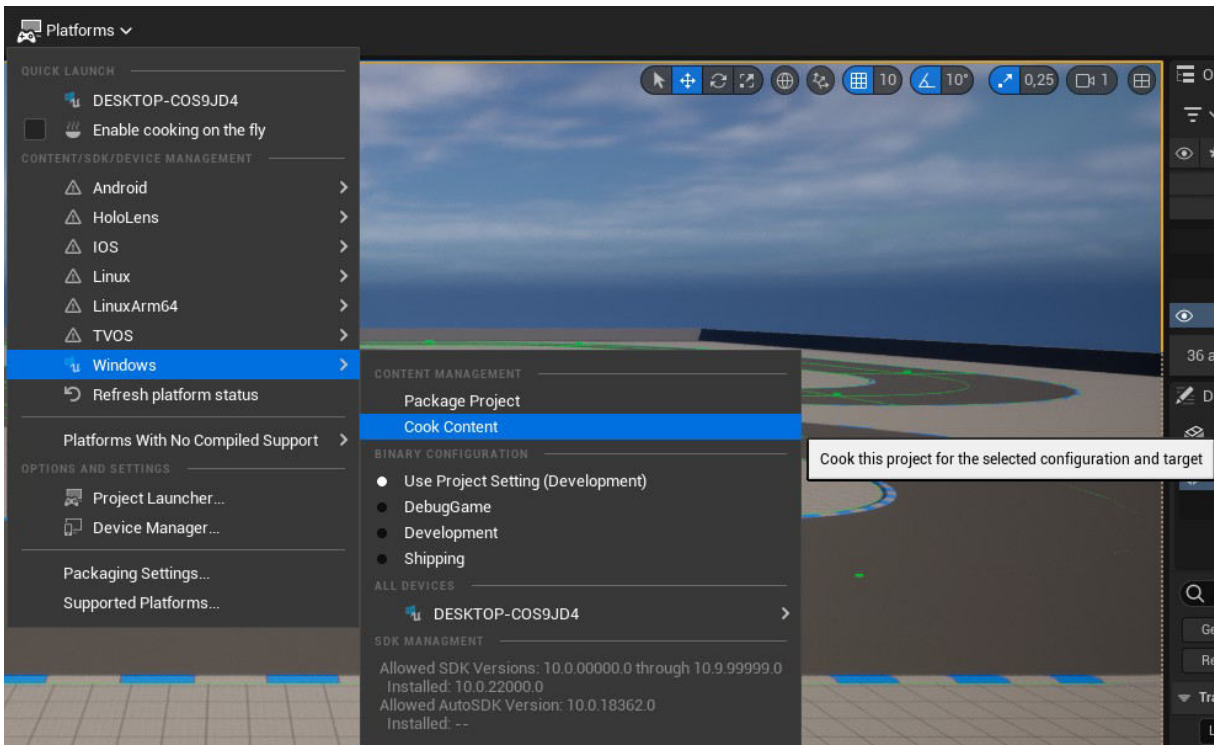
4.3. Kuhanje i pakiranje projekta za Windows OS

Za izdavanje video igre potrebno je prvo odraditi Content Cooking (Kuhanje sadržaja) koje je zapravo proces pretvaranja sadržaja iz internog formata koje razvojno okruženje UE koristi u specifičan format odabrane platforme. Postoji puno postavki koje se mogu odabrati ovisno o tome na koji način tim razvojnih programera želi izdati svoj proizvod, tj. video igru. UE u postavkama projekta već ima zadane postavke, no, za podešavanje istih u alatnoj traci pod Edit, moramo otvoriti Project Settings (već smo to prikazali u jednom od proteklih poglavlja). U skočnom prozoru postavki projekta, pod odjeljkom Project (Projekt) odaberemo pododjeljak Packaging (Pakiranje) i u njemu možemo podesiti što god nam treba ovisno o vrsti i veličini samog projekta kojeg smo izradili. Za potrebe ovog istraživačkog rada, držati ćemo se već zadanim postavkama. [15]



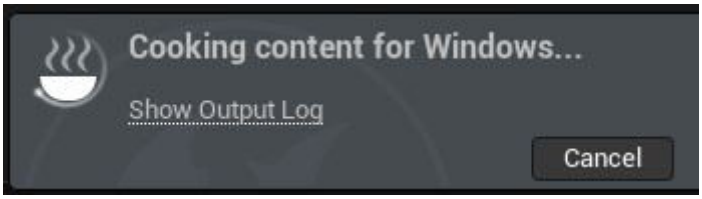
Slika 4.3.1. - Postavke pakiranja projekta

Nakon odabiranja željenih postavki možemo polagano preći na izvoz projekta. U projektnoj traci pod padajućem izborniku Platforms u odjeljku Content/SDK/Device Management postavimo miš na Windows i pod njegovim padajućim izbornikom odaberemo Cook Content.

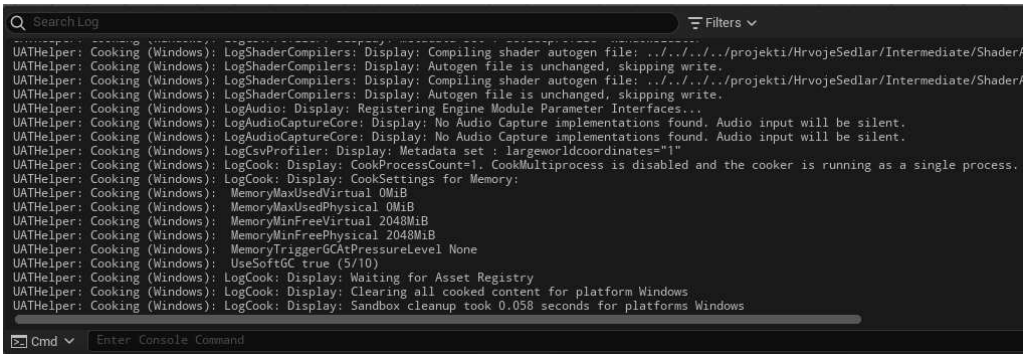


Slika 4.3.2. - Odabir postavke kuhanja sadržaja za Windows OS

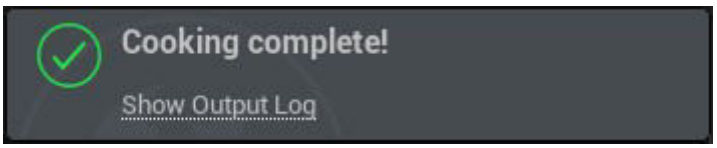
Odabirom Cook Content funkcije nam se pojavi obavijest o tome da UE upravo to izvodi u pozadini. Na obavijesti možemo odabrati „Show Output Log" što nam prikazuje dnevnik aktivnosti svega što se izvodi u pozadini. Ova akcija može potrajati neko vrijeme.



Slika 4.3.3. - Obavijest kuhanja sadržaja

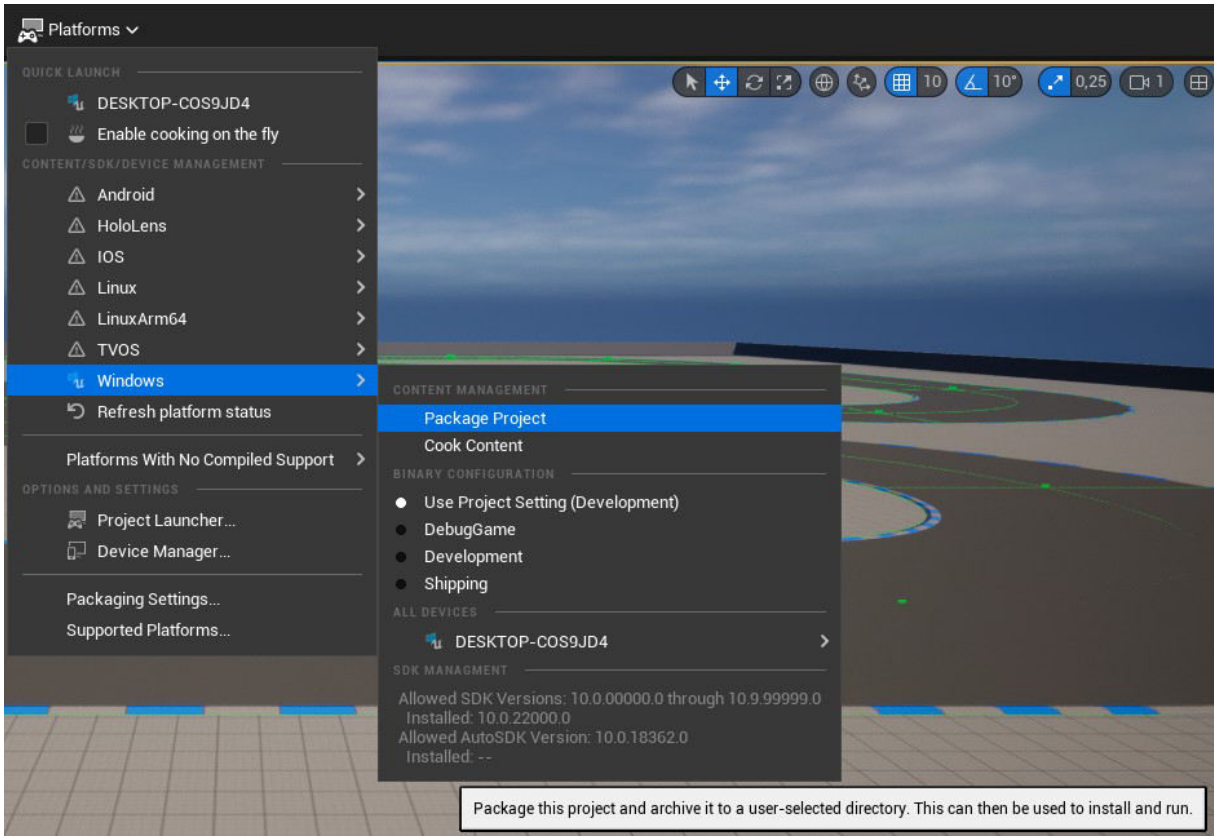


Slika 4.3.4. - Dnevnik aktivnosti kuhanja sadržaja



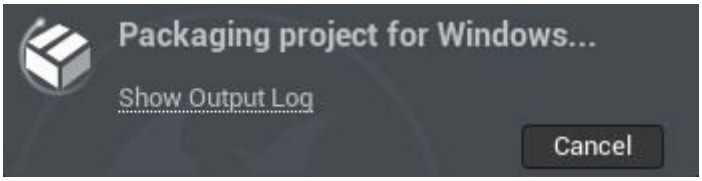
Slika 4.3.5. - Poruka za uspješno odrađeno kuhanje sadržaja

Nakon uspješnog odrađenog kuhanja sadržaja, sada možemo pakirati cijeli projekt jer je spreman za izvoz na platformu za koju smo ga „skuhali". U istom padajućem izborniku za platforme, postavimo se opet na Windows i odaberemo Package Project. Važno je napomenuti da ispod u odjeljku Binary Configuration, možemo odabrati za koju svrhu želimo pakirati projekt. Pošto mi ne izbacujemo video igru na tržište, koristiti ćemo već zadane postavke projekta. UE će nam dopustiti da odaberemo željeni direktorij gdje hoćemo arhivirati projekt.

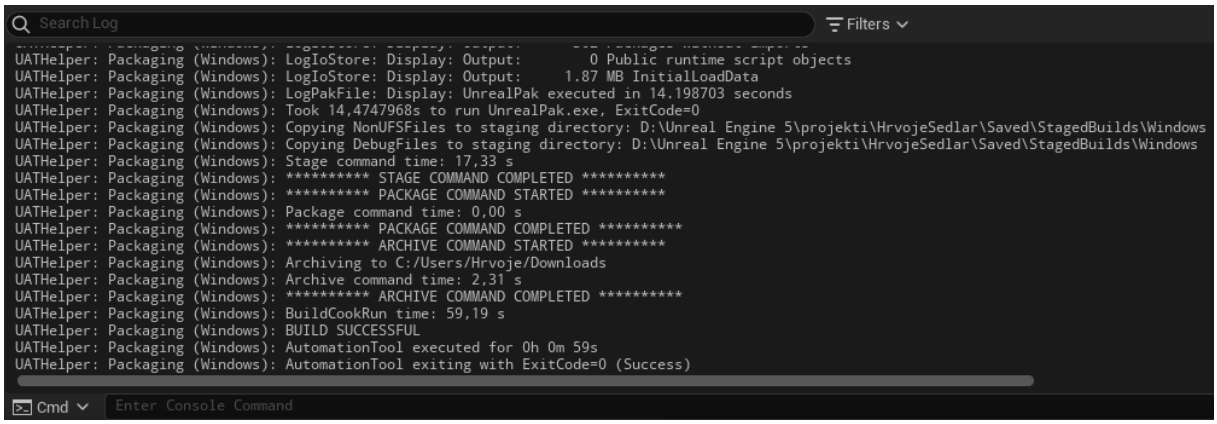


Slika 4.3.6. - Odabir pakiranja projekta za Windows OS

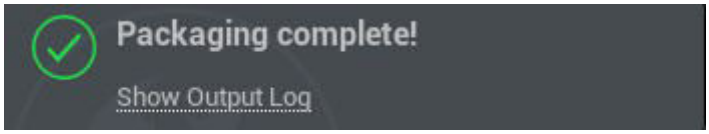
Odabirom direktorija gdje želimo arhivirati naš projekt, UE nam opet izbacuje obavijesti sa dnevnikom aktivnosti o izvođenju tog procesa u pozadini. Pošto smo već odradili kuhanje sadržaja, pakiranje projekta će se brzo izvršiti.



Slika 4.3.7. - Obavijest pakiranja projekta

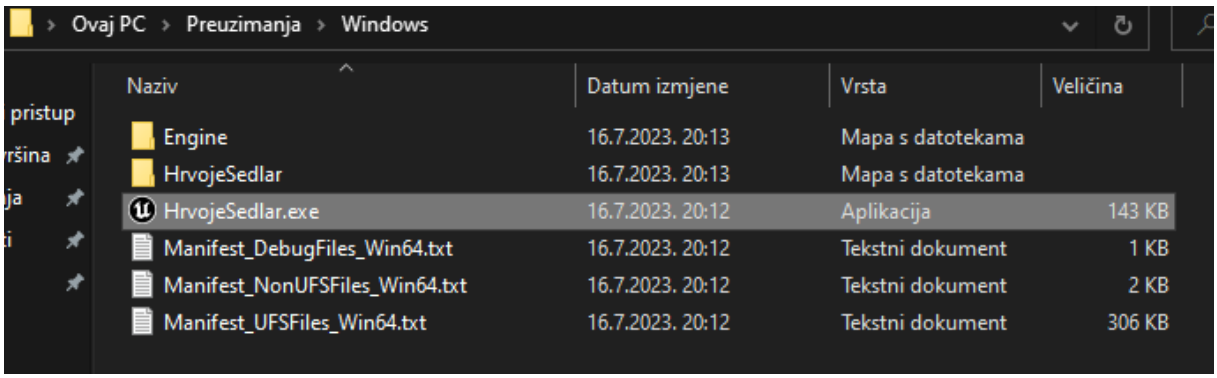


Slika 4.3.8. - Dnevnik aktivnosti pakiranju projekta



Slika 4.3.9 – Poruka za uspješno odrađeno pakiranje projekta

Poslije svega ovoga se lociramo u direktorij koji smo odabrali za pakiranje projekta gdje uz sav sadržaj i datoteke koji je UE pretvorio iz internog formata u format za Windows, vidimo executable (.exe) datoteku sa imenom našeg projekta čijim pokretanjem nam se otvara „video igra” koju smo izveli.



Slika 4.3.10. - Direktorij nakon kuhanja sadržaja i pakiranja projekta za Windows OS

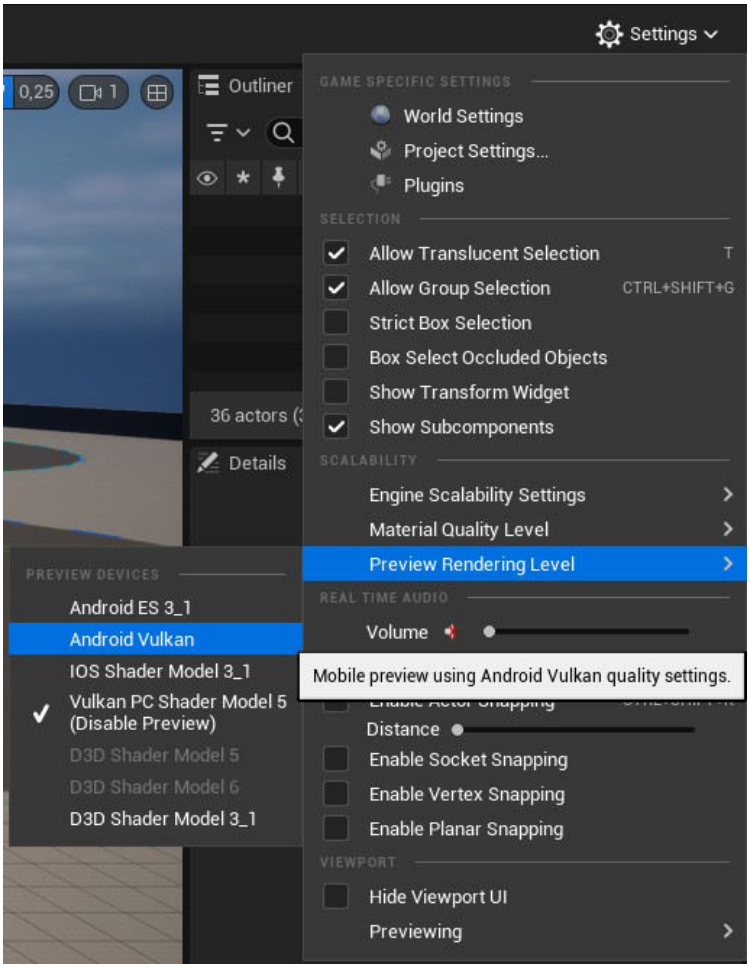
Slika 4.3.11. - Pokrenuta exe datoteka izvezene "video igre"

4.4. Kuhanje i pakiranje projekta za Android OS

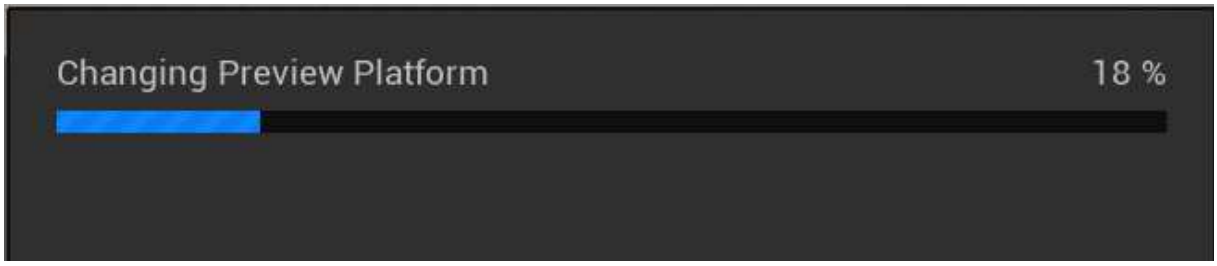
Kao što smo to odradili za Windows OS, kuhanje sadržaja i pakiranje vrijedi i za ostale platforme, međutim, za njih je potrebno imati odgovarajući ili odgovarajući uređaj (npr. u slučaju Android OS-a) ili emulator zajedno sa Software Development Kit (SDK) [16] ukoliko želimo tokom razvoja testirati sami projekt lokalno na računalu.

U razvojnom okruženju možemo vidjeti Preview Rendering Level (pretpregled prikazanog nivoa) za bilo koji dostupni API platforme zbog lakšeg razumijevanja kako se ista video igra izvodi na različitim platformama za koju je razvijamo. Za promjenu na Android

Vulkan, u projektnoj traci skroz desno na padajućem izborniku postavki, spuštamo se na opciju Preview Rendering Level i odabiremo Android Vulkan. UE će zatim kompajlirati potrebne datoteke i shadere prije prikazivanja drugog API-a u glavnom prozoru.



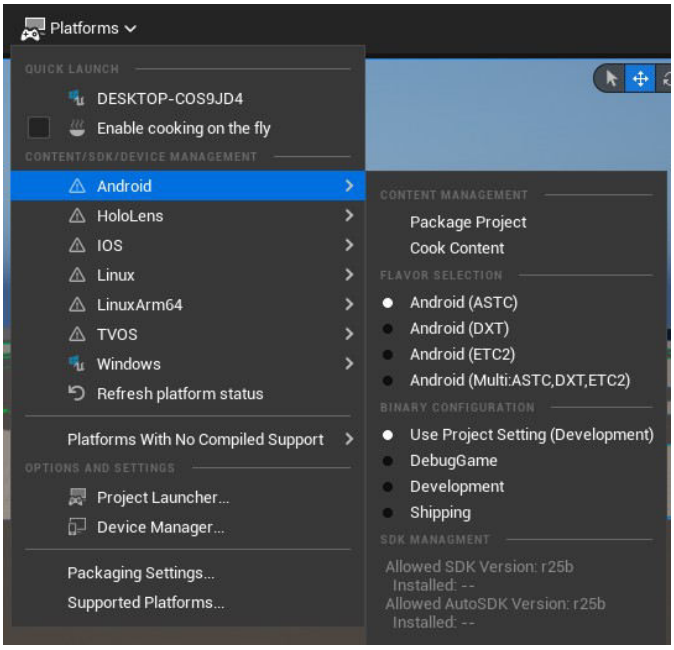
Slika 4.4.1. - Promjena pretpregleda razvojnog okruženja na Android Vulkan API



Slika 4.4.2. - Obavijest mijenjanja API-a

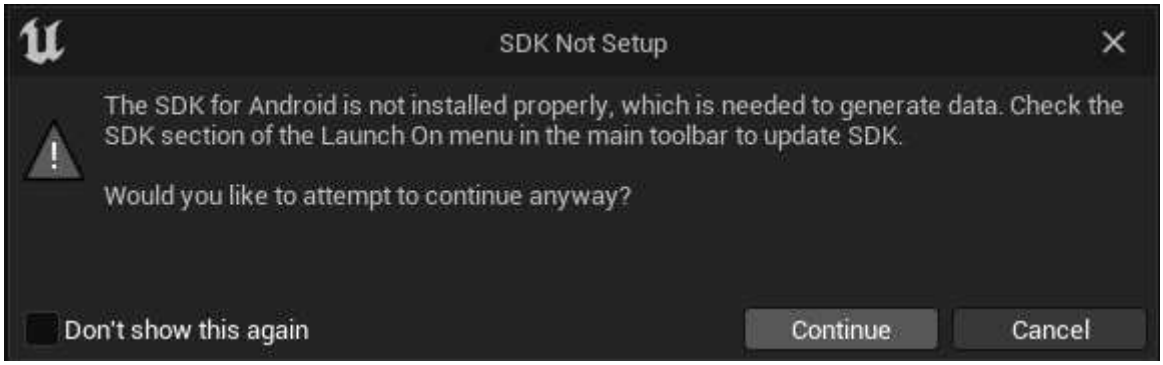
Slika 4.4.3. - Glavni prozor razvojnog okruženja prikazuje nivo koristeći Android Vulkan API

Ostali koraci su isti kao što smo ih opisali u proteklom poglavlju za Windows OS, jedino što se razlikuje je platforma za koju ih izvodimo. Sve se nalazi u padajućem meniju Platforms na projektnoj traci. U ovom slučaju kuhanje sadržaja i pakiranje projekta izvodimo za Android OS.



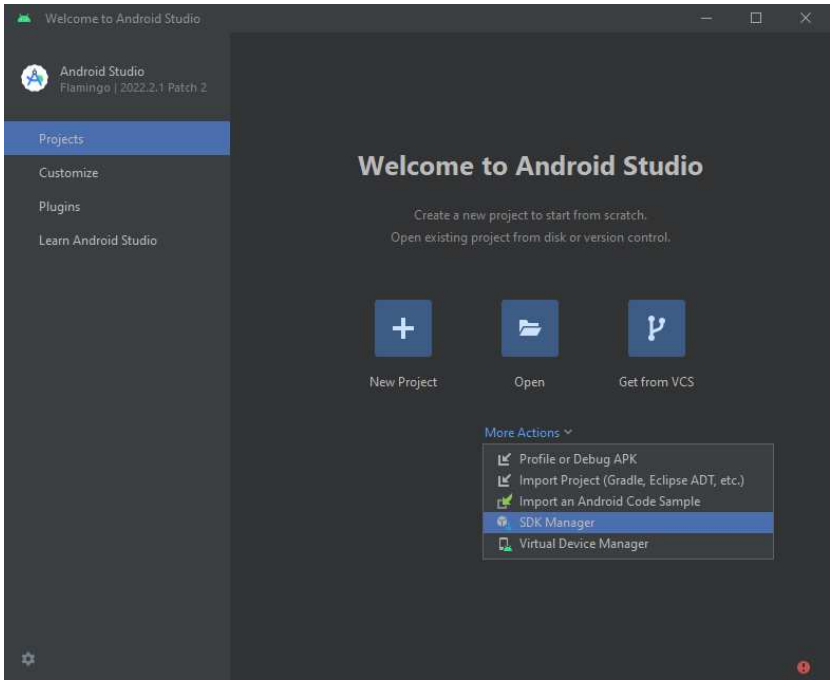
Slika 4.4.4. - Meni za kuhanje sadržaja i pakiranje projekta za Android OS

Pokretanjem jedne od te dve opcije, postoji mogućnost pojave greške da nam SDK nije dobro postavljen. Pošto ćemo mi nakon svega projekt otvoriti u Android Studiu [17] , moramo pravilno postaviti njegov SDK.



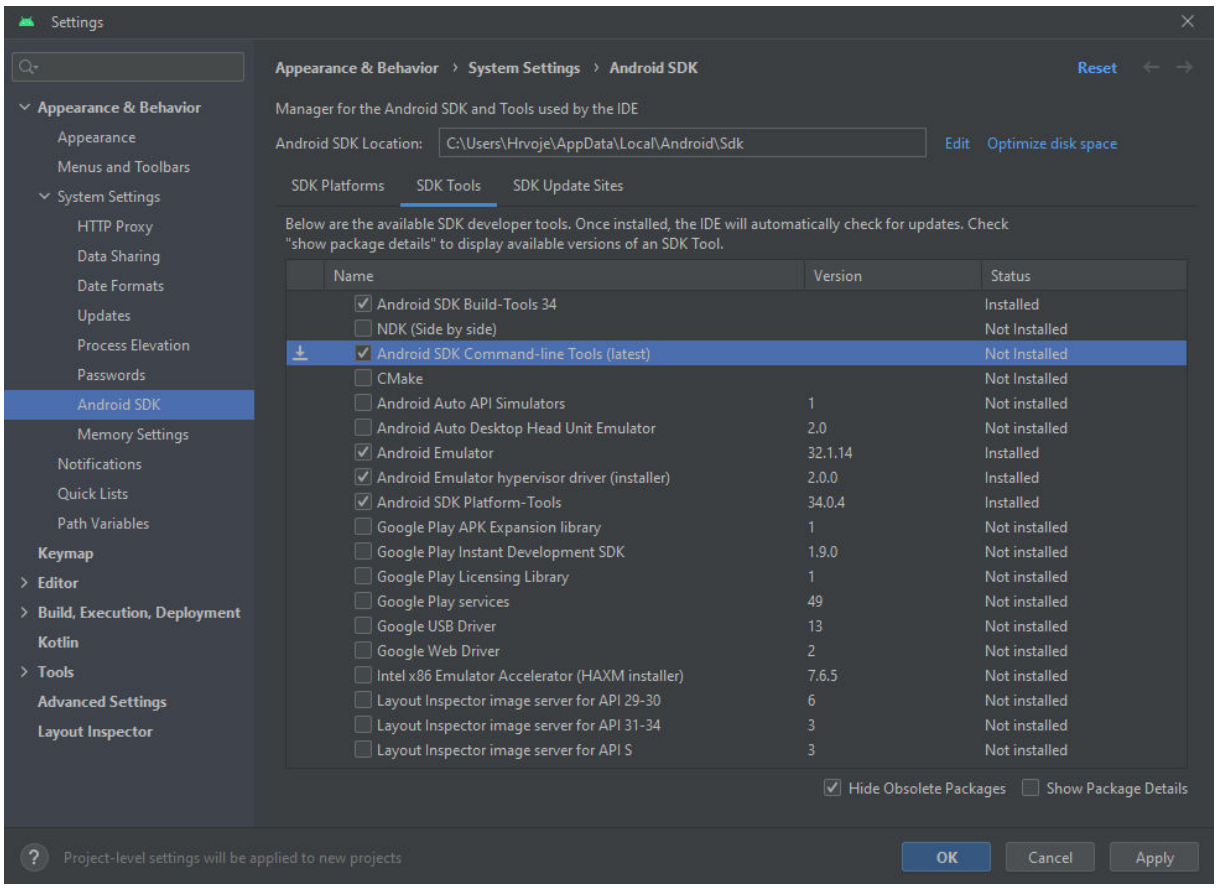
Slika 4.4.5. - Greška zbog nepravilnog postavljanja SDK

Nakon instalacije Android Studia, na početnom prozoru pod More Actions (Više Akcija) biramo opciju SDK Manager (SDK Upravitelj).



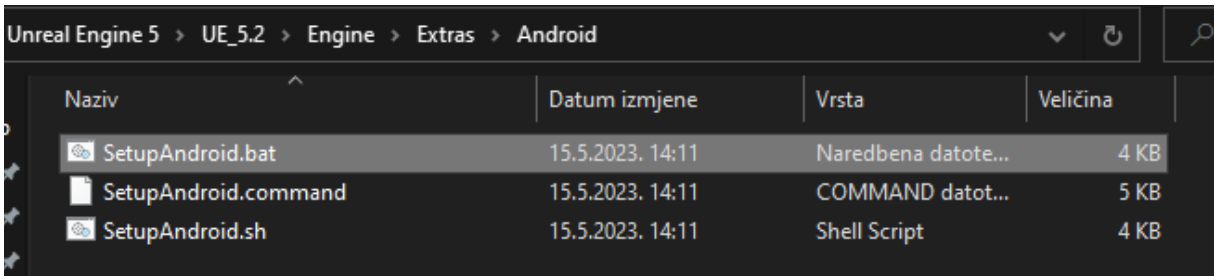
Slika 4.4.6. - Otvaranje SDK Manager na Android Studiu

Na prozoru koji se otvorio se postavimo u karticu SDK Tools (SDK Alati) i kvačicom označimo opciju „Android SDK Command-line Tools" što nam s lijeve strane stvara ikonu za preuzimanje sadržaja nakon čeka kliknemo dugme Apply (Potvrdi) za potvrđivanje i zatim OK za zatvaranje prozora.

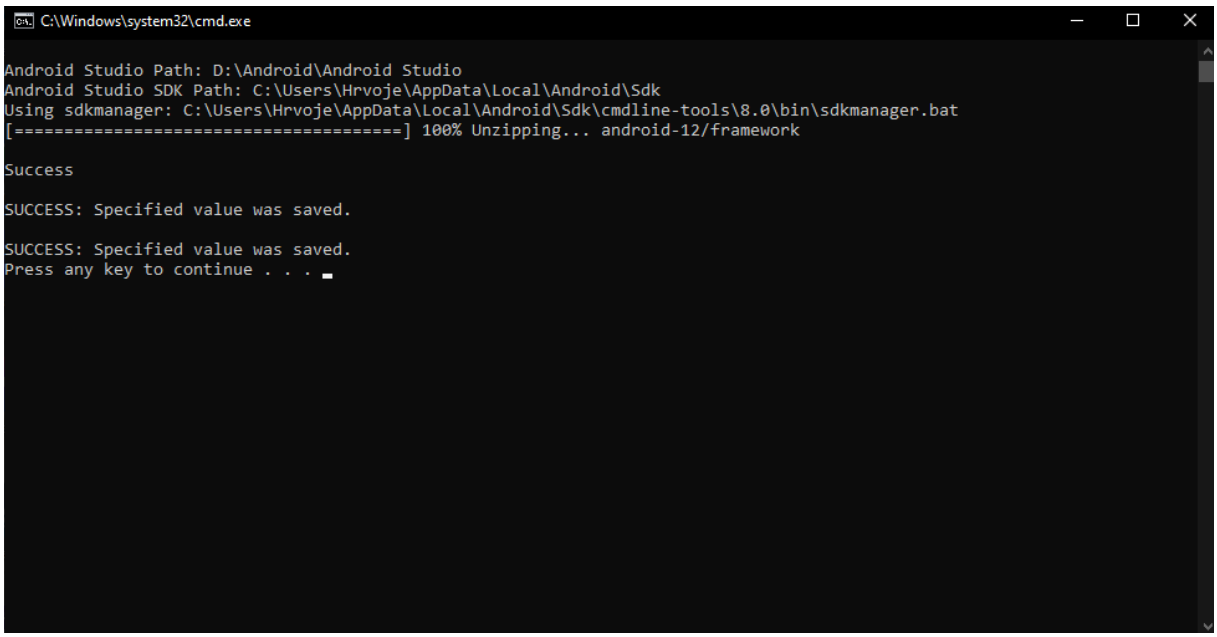


Slika 4.4.7. - Postavljanje Android SDK

Android Studio će zatim sam instalirati odabrani SDK nakon čega je potrebno ponovno pokrenuti računalo. Još je potrebno postaviti i Android NDK [18] što ćemo učiniti pokretanje odgovarajuće skripte „SetupAndroid" (.bat za Windows OS, .command za MacOS, .sh za Linux OS) koja se nalazi u direktoriju UE_5.2 pod Engine/Extras/Android.



Slika 4.4.8. - Direktorij SetupAndroid skripti

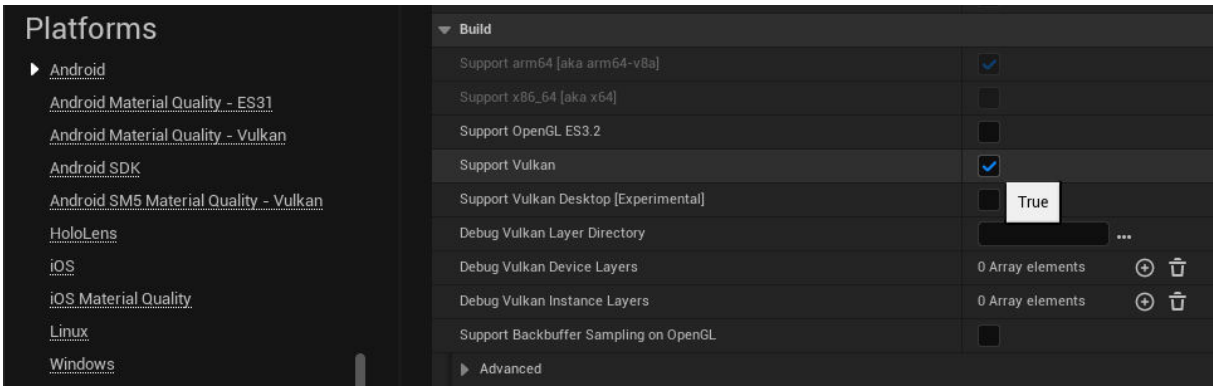


Slika 4.4.9. - Izvršavanje SetupAndroid skripte

Primjećujemo da se u meniju platformi pored opcija za Android umjesto ikone upozorenja pojavio logo Androida što potvrđuje da je SDK pravilno postavljen. [19]

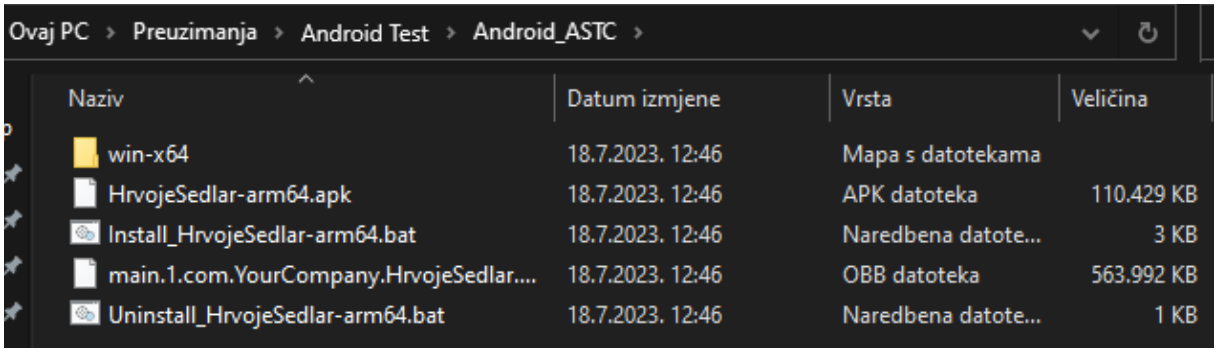
Slika 4.4.10. - Pravilno postavljeni Android SDK

Prije nego što nastavimo sa kuhanjem sadržaja i pakiranjem projekta, moramo podesiti da Android koristi Vulkan API. Tvorničke postavke za Android platforme su obično na postavljene na „Support OpenGL ES3.2”. U postavkama projekta se opet spuštamo do odjeljka Platforms i u Android pod Build mičemo kvačicu sa spomenute OpenGL postavke i postavljamo je na „Support Vulkan”.



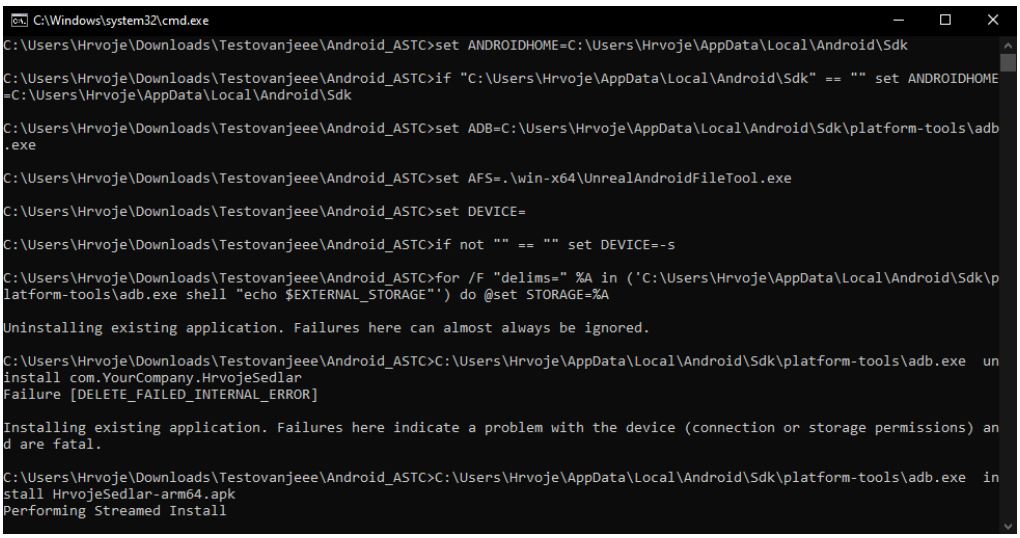
Slika 4.4.11. - Postavljanje Vulkan API-a za Android

Zatim odradimo spomenuto kuhanje sadržaja i pakiranje projekta. Nakon što se to izvede, direktorij koji smo odabrali za pakiranje bi trebao izgledati kao na slici ispod.



Slika 4.4.12. - Direktorij nakon kuhanja sadržaja i pakiranja projekta za Android OS

Taj projekt zatim otvorimo u Android Studiu i pokrenemo AVD emulator po želji na kojem ga želimo testirati. Nakon što je Android OS pokrenut u emulatoru, u direktoriju gdje smo smjestili naš projekt pokrenemo datoteku „Install_imeprojekta-arm64.bat" da se aplikacija uspješno instalira na emulirani mobilni uređaj. Zatim kliknemo na aplikaciju da pokrenemo našu „video igru".



Slika 4.4.13. - Instaliranje naše aplikacije na Android OS

Slika 4.4.14. - Android OS sa instaliranom aplikacijom (ikona u sredini)

Slika 4.4.15. - Pokrenuta aplikacija izvezene „video igre”.

5. Zaključak

Standard Vulkan i Unreal Engine se dosta koriste u današnje vrijeme od strane raznih razvojnih timova, bez obzira koja je namjena njihovih projekata. Za to su zaslužne godine i godine rada i promatranja tržišta da bi se na njega plasirao što bolji proizvod. Danas je važna brzina rada da se dostignu svi nametnuti rokovi projekta, zajedno sa kvalitetom i mogućnosti da se projekt plasira na što više platformi jer se tako dopire do više korisnika.

Iako sam koristio demo već dostupan u datotekama UE razvojnog okruženja kao primjer na koji se implementira Vulkan i zatim se izvozi na više platformi (Windows i Android), samo sučelje UE-a je intuitivno za korištenje i bez problema se može snaći u njemu i napraviti što god treba, pogotovo programiranje koje je olakšano za svakog korisnika u obliku blueprintova. Promjena postavki kuhanja sadržaja i pakiranja projekta ovisno o platformi za koju želimo izdati projekt se izvršava brzo jer velika većina stvari je već dostupna u postavkama projekta. Izvoz projekta u obliku kuhanja sadržaja i pakiranja projekta se izvršavalo dosta brzo iako sam ja radio na hardveru koji je već zastario koristeći komponente koje nisu ni približno blizu najbolje dostupnima. Naravno, uvijek se sve može poboljšati sa više optimizacije, istraživanjem koji parametri Vulkana najbolje rade u razvojnom okruženju UE-a, zajedno sa implementacijom različitih razina grafičkih postavki jer svi hardveri nisu isti pa se demo neće vrtjeti niti izgledati dovoljno dobro na slabijem hardveru kao što izgleda na jačemu. Kao što to vrijedi za osobna računala, tako to vrijedi i za mobilne uređaje.

Obje tehnologije imaju svijetlu budućnost jer se na njima radi iz dana u dan prateći najnovije trendove u svijetu hardvera i grafike, video igara, itd. Sve je lakše i lakše razvijati velike projekte i od kuće, zato vidimo veliki porast neovisnih razvojnih programera u zadnjih nekoliko godina, a dosta njih koristi Vulkan i UE. Isto tako i ogromne kompanije prelaze na standardizirane tehnologije umjesto razvijanja svojih zbog toga jer to manje košta, brže je vrijeme razvijanja i na kraju krajeva olakšava se općeniti razvoj projekta.

6. Popis slika

7. Popis literature

[1] “Vulkan-Tutorial - About,” accessed June 7, 2023, https://vulkan-tutorial.com/Introduction.

[2] “Vulkan-Tutorial - Origin of Vulkan,” accessed June 7, 2023, https://vulkan-tutorial.com/Overview.

[3] “Khronos.Org - About,” accessed June 8, 2023, https://www.khronos.org/about/.

[4] “Khronos.Org - Khronos New Initiative Proposal Document,” accessed July 7, 2023, https://www.khronos.org/files/khronos_new_initiative_proposal.pdf.

[5] “Khronos.Org - New Initative Process Overview,” accessed June 8, 2023, https://www.khronos.org/api/exploratory.

[6] “Ogldev.Org - Vulkan First Triangle,” accessed June 20, 2023, https://ogldev.org/www/tutorial52/tutorial52.html.

[7] “Vulkan-Tutorial - Drawing a Triangle,” accessed July 7, 2023, https://vulkan-tutorial.com/Drawing_a_triangle/Setup/Base_code.

[8] “Unreal Engine - Wikipedia,” accessed June 13, 2023, https://en.wikipedia.org/wiki/Unreal_Engine.

[9] “Enviromental Audio Extensions - Wikipedia,” accessed July 7, 2023, https://en.wikipedia.org/wiki/Environmental_Audio_Extensions.

[10] “S3 Texture Compression - Wikipedia,” accessed July 7, 2023, https://en.wikipedia.org/wiki/S3_Texture_Compression.

[11] “Ray Tracing (Graphics) - Wikipedia,” accessed July 8, 2023, https://en.wikipedia.org/wiki/Ray_tracing_(graphics).

[12] “Unreal Engine Docs - Automation System Overview,” accessed July 15, 2023, https://docs.unrealengine.com/5.2/en-US/automation-system-in-unreal-engine/.

[13] “Unreal Engine Docs - Task Graph Insights,” accessed July 15, 2023, https://docs.unrealengine.com/5.2/en-US/task-graph-insights-in-unreal-engine-5/.

[14] “Unreal Engine Docs - Automation System Overview.”

[15] “Unreal Engine Docs - Cook, Package, Deploy and Run,” accessed July 16, 2023, https://docs.unrealengine.com/5.2/en-US/build-operations-cooking-packaging-deploying-and-running-projects-in-unreal-engine/.

[16] “Software Development Kit - Wikipedia,” accessed July 16, 2023, https://en.wikipedia.org/wiki/Software_development_kit.

[17] “Android Studio,” accessed July 16, 2023, https://developer.android.com/studio.

[18] “Android NDK,” accessed July 16, 2023, https://developer.android.com/ndk.

[19] “Unreal Engine Docs - Setting up Android SDK,” accessed July 16, 2023, https://docs.unrealengine.com/5.2/en-US/how-to-set-up-android-sdk-and-ndk-for-your-unreal-engine-development-environment/.