Izrada 3D simulacije Sunčevog sustava

Peša, Tomislav

VISOKA ŠKOLA ZA INFORMACIJSKE TEHNOLOGIJE

STRUČNI STUDIJ INFORMACIJSKIH TEHNOLOGIJA

ZAGREB

Tomislav Peša

ZAVRŠNI RAD

IZRADA 3D SIMULACIJE SUNČEVOG SUSTAVA

Zagreb, studenoga 2022.

Visoka škola za informacijske tehnologije
10000 Zagreb, Klaićeva 7

Studij:	Preddiplomski stručni studij informacijskih tehnologija smjer programiranje
Student:	**Tomislav Peša**
Matični broj:	2017097

Zadatak završnog rada

Predmet:	Programiranje u C#
Naslov:	**Izrada 3D simulacije Sunčevog sustava**
Zadatak:	U programskom jeziku C# napraviti aplikaciju koja će simulirati kretanje planeta u Sunčevom sustavu. U okviru rada opisati korištene biblioteke.
Mentor:	mr. sc. Julijan Šribar, v. pred.

Zadatak uručen kandidatu: 3.11.2021.

Rok za predaju rada: 30.11.2022.

Rad predan: _____

Povjerenstvo:

Dragana Čulina, pred.	član predsjednik	_____
mr. sc. Julijan Šribar, v. pred.	mentor	_____
Mariza Maini, pred.	član	_____

SADRŽAJ

POPIS SLIKA

POPIS KODOVA

1. UVOD

Industrija videoigara je ozbiljna grana informacijskih tehnologija koja zapošljava mnoge dizajnere, programere, audio i video stručnjake, te ostvaruje prihode od više stotina milijardi eura godišnje. Iz tog razloga razvijen je niz alata koji olakšavaju izradu videoigara, među kojima je danas jedan od najpopularnijih Unity. Svrha ovog rada je opisati korištenje Unity razvojnog okruženja kroz razvoj 3D aplikacije pod nazivom *Systellar*. S obzirom na veliki interes o Sunčevom sustavu i svemiru općenito, odlučeno je da praktični dio ovog završnog rada bude edukativna aplikacija koja prikazuje orbitu planeta te pruža osnovne informacije o suncu i pojedinim planetima Sunčevog sustava.

U početku je ideja bila napraviti 2D simulaciju koristeći programski jezik C# te jednostavnu i brzu multimedijsku biblioteku (engl. *Simple and Fast Multimedia Library, SFML*) za izradu 2D grafike, ali je naknadno odlučeno izdvojiti više vremena te napraviti 3D simulaciju. U početku je za izradu simulacije korišten *Unity 2021.2.1*, a za skriptiranje u programskom jeziku C# *Microsoft Visual Studio 2022*. Pred kraj izrade korišten je *Unity 2022.1.6*..

U drugom poglavlju opisani su alati korišteni u izradi praktičnog rada. Prikazana je povijest Unityja, neke verzije kroz povijest i značajke koje su implementirane u svakoj od njih, izbor i način kreiranja projekta, razlika između projekta tipa 2D/3D i 2D/3D URP/HDRP. U nastavku je prikazano korisničko sučelje koje sadrži tri najbitnije stavke Unityja, a to su prozori *igra* (engl. *Game)* i *scena* (engl. *Scene)*, te *hijerarhija* (engl. *Hierarchy)* u kojoj su prikazani svi objekti na sceni. Također, prikazan je *inspektor* (engl. *Inspector*) koji omogućuje promjenu nad objektima, npr. promjena pozicije ili rotacije, te *konzola* (engl. *Console*) koja ukazuje na pogreške u kôdu i na sceni. U *skladištu imovine* (engl. *Asset Store*) se nalaze razni paketi koje je moguće uvesti i koristiti u projektima, od tekstura, animacija, zvuka, 2D/3D modela do već gotovih projekata.

U trećem poglavlju fokus je na fizici, konkretno na Newtonovom zakonu gravitacije i brzini kruženja koje su korištene u skriptiranju za orbitu planeta. Četvrto poglavlje daje uvid u korake izrade aplikacije, detaljno je opisana izrada svaka scene. Prikazan je način i svrha korištenja određenih funkcija, objašnjene su animacije koje su kreirane za bolji i moderniji izgled aplikacije, te kreiranje postavki kako bi korisnici imali mogućnost izbora željene rezolucije, kvalitete slike i jačine zvuka. Samo težište četvrtog poglavlja je na izradi simulacije i informacija o pojedinim planetima.

2. KORIŠTENI ALATI I PROGRAMSKA RIJEŠENJA

Čak i danas velike tvrtke razvijaju svoje vlastite alate koje koriste u vlastitim projektima. To košta dosta novaca, ali i uvijek potrebnog programerskog znanja. U većini slučajeva, kada kreću na neki novi projekt, tvrtke ažuriraju postojeći alat te tako štede vrijeme sebi i krajnjim korisnicima koji koriste njihov finalni proizvod. Tako je npr. Poljska tvrtka *CD Projekt Red* korištenjem svog *REDengine 3* napravila jednu od najboljih video igara *The Witcher 3: Wild Hunt* koja je 2015. godine osvojila 281 nagradu za igru godine.

Na tržištu postoje mnogi programski alati za izradu videoigara i aplikacija od kojih su najpopularniji:

- Unreal Engine
- Unity
- Game Maker
- Godot
- Urho3D

Unreal Engine je najpopularniji alat za izradu videoigara i samim time najveća konkurencija Unityju. Prva verzija nastala je 1998. godine kada su inženjeri *Epic Games* studija razvili alat za potrebe kreiranja igre *Unreal*. Unreal Engine je stekao popularnost zbog vrhunske grafike, ali i zbog korištenja u vrhunskoj AAA industriji. AAA je neformalna klasifikacija koja se koristi za kategorizaciju igara koje proizvodi veći izdavač, a koji obično ima veći proračun za razvoj i marketing od ostalih izdavača.

Najveća razlika između Unreal Engine-a i Unityja je programski jezik koji koriste. Unity koristi programski jezik C#, dok Unreal Engine koristi programski jezik C++, te, za razliku od Unityja, ima mogućnost kreiranja igre bez pisanja programskog koda korištenjem vizualnog skriptiranja koje se naziva *nacrt* (engl. *Blueprint*). Pomoću nacrta programeri stvaraju čvorove i povezuju ih kako bi razvili logiku za video igru. Unreal Engine je potpuno besplatan ali on za razliku od Unityja ima poseban model naplate: kada igra ili aplikacija krene u prodaju, Unreal Engine je obavezan dobiti 5% naknade za svaki prodani primjerak.

2.1. Unity

Unity je računalni program za razvoj video igara i aplikacija tvrtke Unity Technologies. Koristi se za kreiranje 2D/3D videoigara i aplikacija za Windows, MacOS, Linux, PlayStation, Xbox,

iOS, Android. Unity koristi objektno orijentirani programski jezik C#. Cijelo razvojno okruženje napisao je u programskim jezicima C i C++. Alat nudi gotove klase koje olakšavaju izradu videoigara i aplikacija, ali ih je potrebno razumjeti radi što kvalitetnijeg iskustva izrade. Klase su opisane u nastavku rada prilikom opisa skriptiranja.

Unity su 2005. godine kreirali *David Helgason, Nicholas Francis i Joachim Ante*. U početku je Unity bio namijenjen samo za Mac OS. Unity verzija 2.0 je objavljena 2007. godine, a sadržavala je oko pedeset novih značajki. Neke od važnijih su optimizacija alata za izradu terena i osvjetljenja. Najbitnija značajka je *UDP* (engl. *User Datagram Protocol*) koji je omogućio programerima razvijanje igara i aplikacija za više igrača. Unity je tek 2009. godine dobio podršku za Windowse. Trenutna verzija Unity 2022 omogućuje poboljšanje produktivnosti pomoću naprednog pretraživanja, ubrzanog načina ulaska u Play mode, uvoza i izvoza datoteka, te bržeg tijeka generiranja materijala.

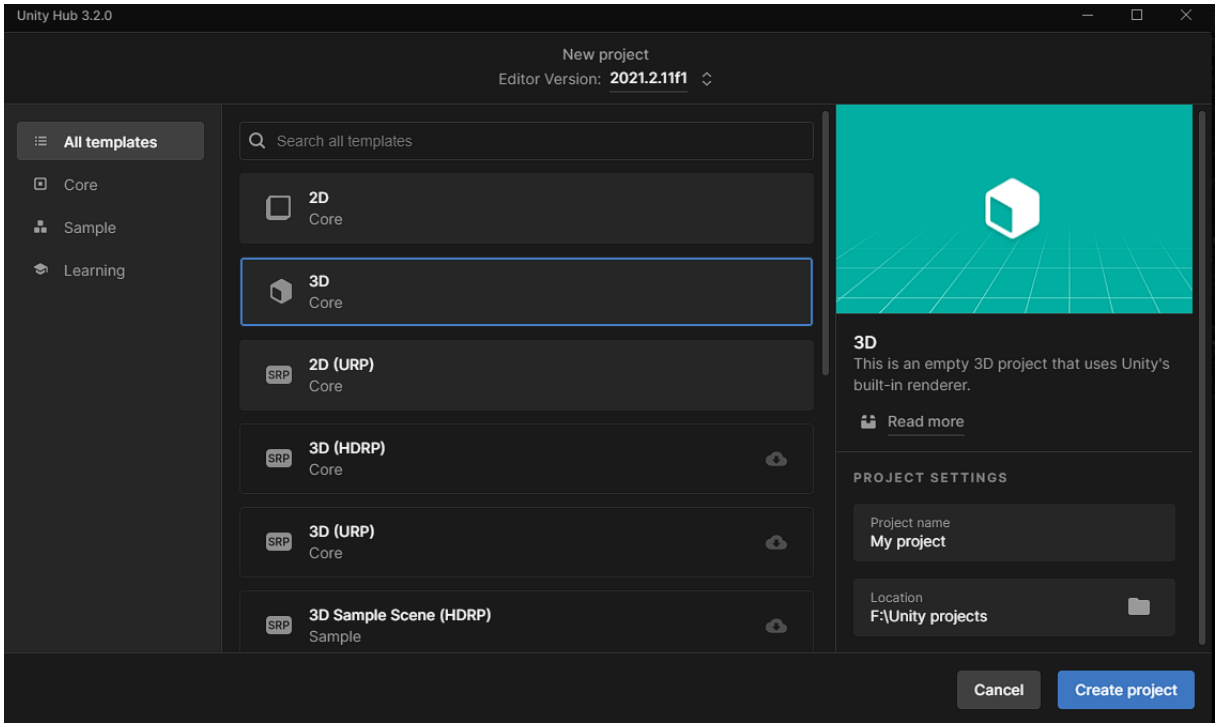
Zbog razumljivog i jednostavnog korisničkog sučelja, Unity je postao jedan od najpopularnijih programa za razvoj video igara i aplikacija na svijetu. Unity nudi besplatnu licencu, te omogućava studentima i samostalnim programerima da kreiraju vlastite projekte besplatno, sve dok ne prekorače određeni limit zarada. Postoje dvije vrste izdanja aplikacije Unity, jedna od njih je *verzija s dugoročnom potporom* (engl. *Long Term Support*, *LTS*) koja se preporučuje korisnicima koji cijene maksimalnu stabilnost i podršku za svoje projekte. Druga verzija aplikacije je *tehnološki tok* (engl. *Tech Stream*) koja je namijenjena korisnicima koji žele ranije korištenje novih značajki kako bi se pripremili za buduće projekte. Ta verzija se preporučuje za faze pripreme, otkrivanja i izrade prototipa.

Neke od najvećih tvrtki koriste Unity za razvoj svojih videoigara i aplikacija. Nekoliko popularnih videoigara su *Subnautica* tvrtke *Unknown Worlds Entertainment*, *Fall Guys* tvrtke *Mediatonic*, *Ori and the Will of the Wisps* tvrtke *Moon Studios*, te aplikacije *Shapes* tvrtke *Learn Teach Explore* i *Nano Simbox* tvrtke *Interactive Scientific*.

2.1.1. Kreiranje projekta

Kao što je prikazano na Slika 1, Unity koristi *Unity hub* kao aplikaciju u kojoj je moguće kreirati projekte, instalirati određenu verziju Unityja, dodavati i uklanjati dodatke. Tako Unity hub omogućava zasebno generiranje izvršnih datoteka za Linux, Android, te iOS.

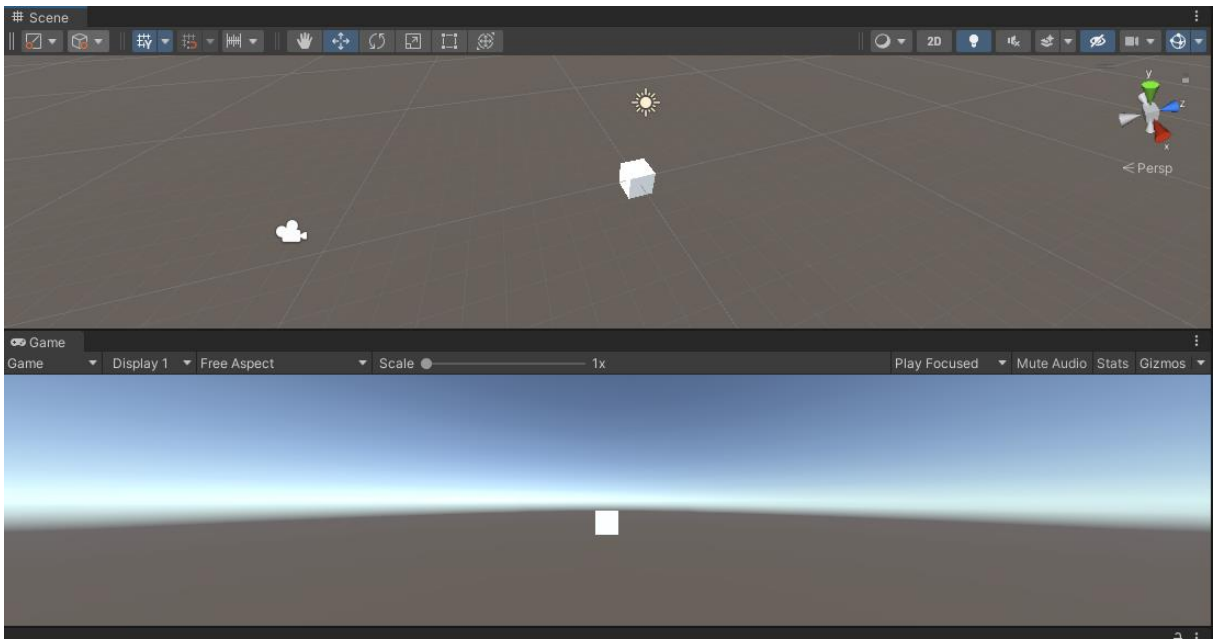
Unity nudi razne vrste projekata, kao što su klasični 2D i 3D. Također postoje 2D URP/HDRP i 3D URP/HDRP koji, za razliku od klasičnog 2D/3D, sadrže ugrađeni *Scriptable Render Pipeline (SRP)*, napravljen od strane Unityja. Scriptable Render Pipeline je aplikacijsko programsko sučelje odnosno API (engl. *Application Programming Interface)* sloj koji omogućuje raspoređivanje i konfiguriranje naredbi za iscrtavanje (engl. *Rendering*) koristeći C# skripte. *Universal Render Pipeline (URP)* omogućuje umjetnicima da brzo i jednostavno stvore optimiranu grafiku za platforme, od mobilnih uređaja pa sve do konzola i računala. *High Definition Render Pipeline (HDRP)* omogućuje reprodukciju širokog spektra realističnih površina, od složenih metalnih površina do specifičnih detalja poput tkanina, kosa i višeslojnih materijala. Sadrži postprocesne efekte kao što su depth of view, motion blur, te bloom. Unity podržava mogućnost izrade igara i aplikacija virtualne stvarnosti (engl. *Virtual Reality*) te proširene stvarnosti (engl. *Augmented Reality*).



Slika 1. Kreiranje projekta

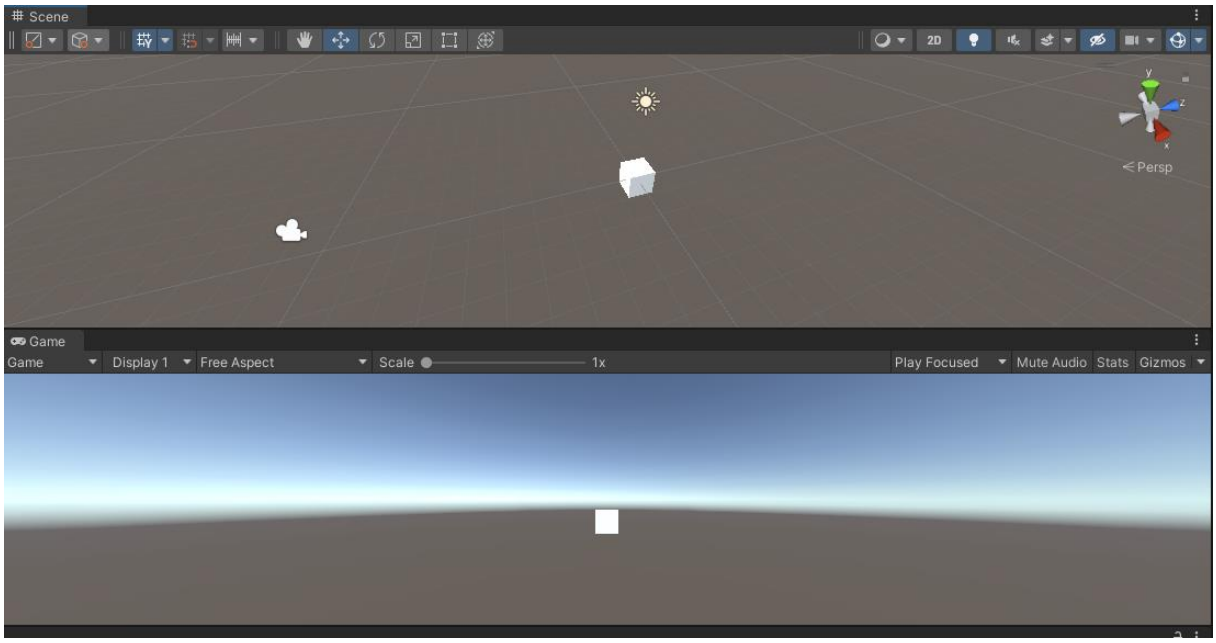
2.1.2. Korisničko sučelje

Otvaranjem projekta prikazuje se korisničko sučelje. Ono sadrži mnogo bitnih i korisnih informacija u izradi projekta. Kao što je prikazano na

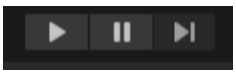


, najveći dio sadrže dva velika prozora: jedan je Scene, a drugi Game. U prozoru Scene se nalaze svi rekviziti i glumci, dok prozor Game daje prikaz finalnog proizvoda. Scena prikazuje interaktivni pogled na stvoreni svijet, u koji se postavljaju razni likove, kamere, svjetla i svi ostale objekti. Scena prikazuje 2D ili 3D perspektivu, ovisno o projektu na kojem se radi. Prozor Game prikazuje kako izgleda završna igra ili aplikacija kroz kameru. Na

Slika *3* prikazana je tipka za reprodukciju i zaustavljanje simulacije.

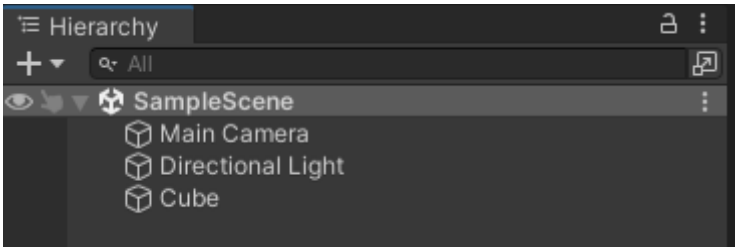


Slika 2. Prikaz prozora Scene i Game



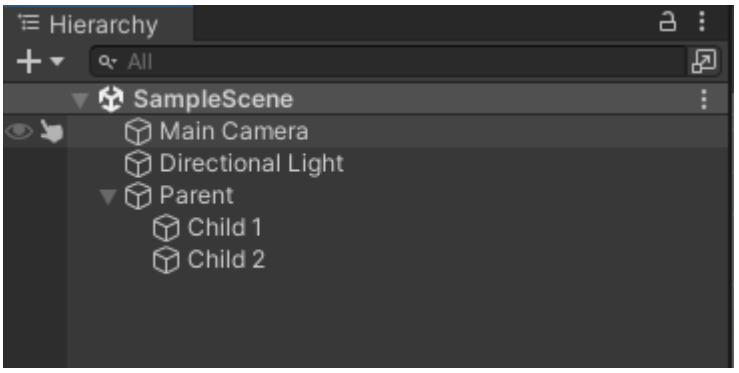
Slika 3. Tipka za reprodukciju i zaustavljanje simulacije

Sljedeći bitan element korisničkog sučelja je *prozor s hijerarhijom* (engl. *Hierarchy*). Na Slika 4 prikazana je hijerarhija, u kojoj se nalaze svi objekti na sceni, kamere itd. Hijerarhija se koristi za grupiranje i sortiranje objekata koji se koriste u sceni. Nakon dodavanja ili uklanjanja objekta sa scene, taj objekt se automatski uklanja iz hijerarhije.

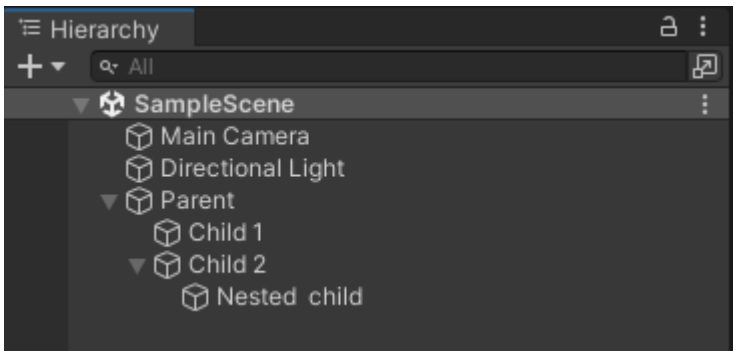


Slika 4. Prikaz prozora s hijerarhijom

Kao što je prikazano na Slika 5, *Unity* koristi koncept roditelj/dijete (engl. *Parent/Child*) za grupiranje objekata. *Parent* objekt sadrži druge *child* objekte koji nasljeđuju njegova svojstva, kao što su kretanje i rotacija. To znači da će se pomicanjem *parent* objekta, pomicati i svi njegovi *child* objekti. Kao što je prikazano na Slika 6, postoji mogućnost daljnjeg proširivanja hijerarhije ugnježđivanjem gdje *child* ima svoj *child* (engl. *Nested parent/child*).

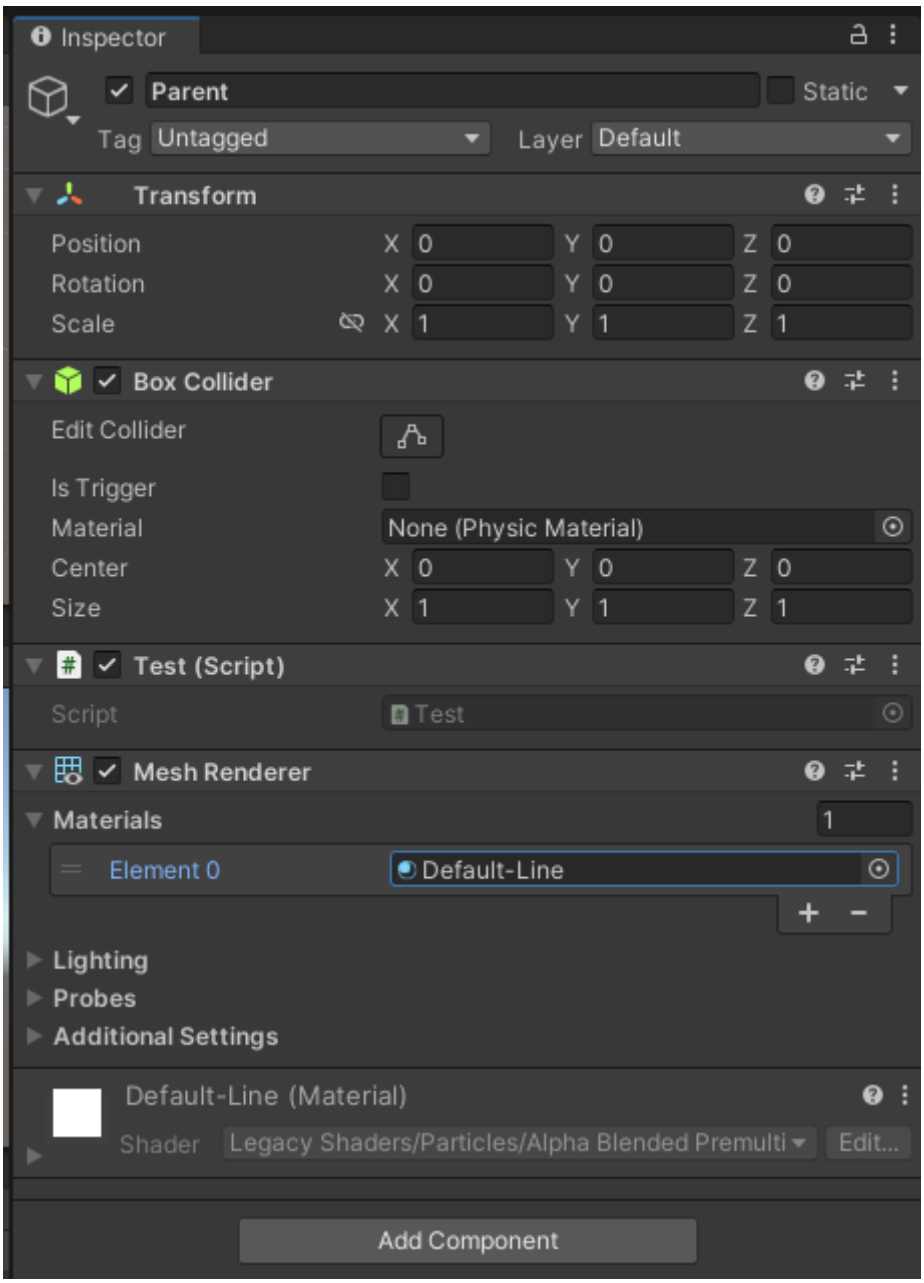


Slika 5. Prikaz odnosa roditelj/dijete u prozoru s hijerarhijom



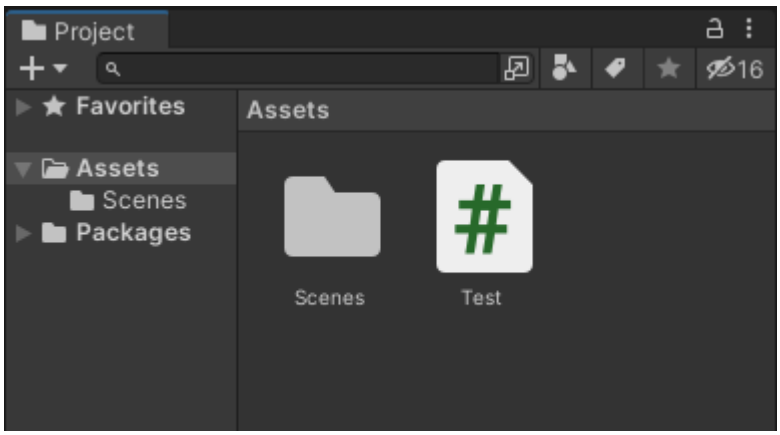
Slika 6. Prikaz odnosa ugniježđeni roditelj/dijete u prozoru s hijerarhijom

Na Slika 7 prikazan je prozor *Inspektor* (engl. *Inspector*). Inspektor je vrlo moćan i bitan prozor u Unityju koji omogućuje napredniju i detaljniju promjenu nad objektima. Koristi se za uređivanje specifičnih svojstava vezanih za objekt kao što su: materijali, sudarači (engl. *Colliders*), rendereri, skripte itd.



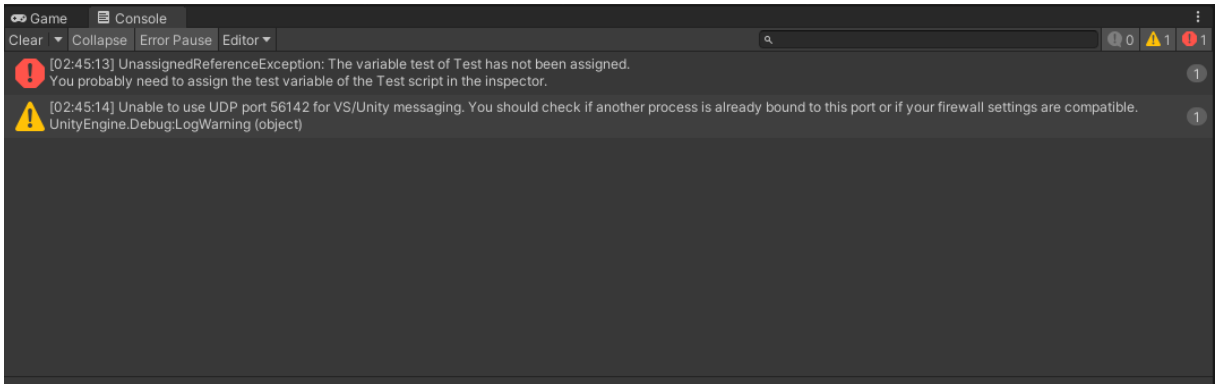
Slika 7. Prozor Inspektor

Kao što je prikazano na Slika 8, u *Projektnom prozoru* (engl. *Project window*) nalaze se sve datoteke vezane za projekt. Klikom na desnu tipku miša postoji mogućnost kreiranja mapa, skripti, materijala, scena itd.



Slika 8. Prikaz projektnog prozora

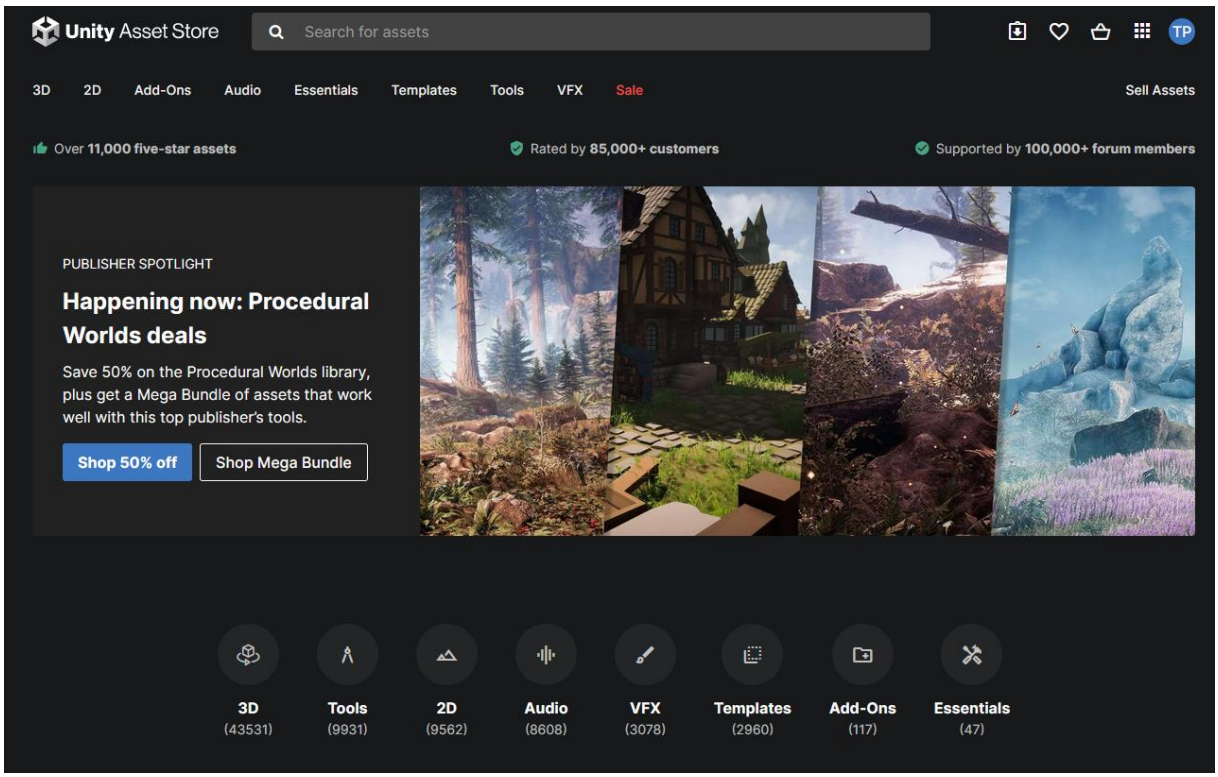
Na Slika 9 prikazana je *Konzola (*engl. *Console)*, u kojoj su prikazane razne pogreške i upozorenja koja ukazuju na probleme u kôdu ili na sceni. Crvenim uskličnikom označene su pogreške koje se moraju ispraviti da bi se simulacija uopće mogla pokrenuti. Žutim uskličnikom su označena upozorenja da nešto nije u potpunosti korektno napravljeno, ali ona neće spriječiti pokretanje simulacije.



Slika 9. Prikaz pogreške i upozorenja unutar konzole

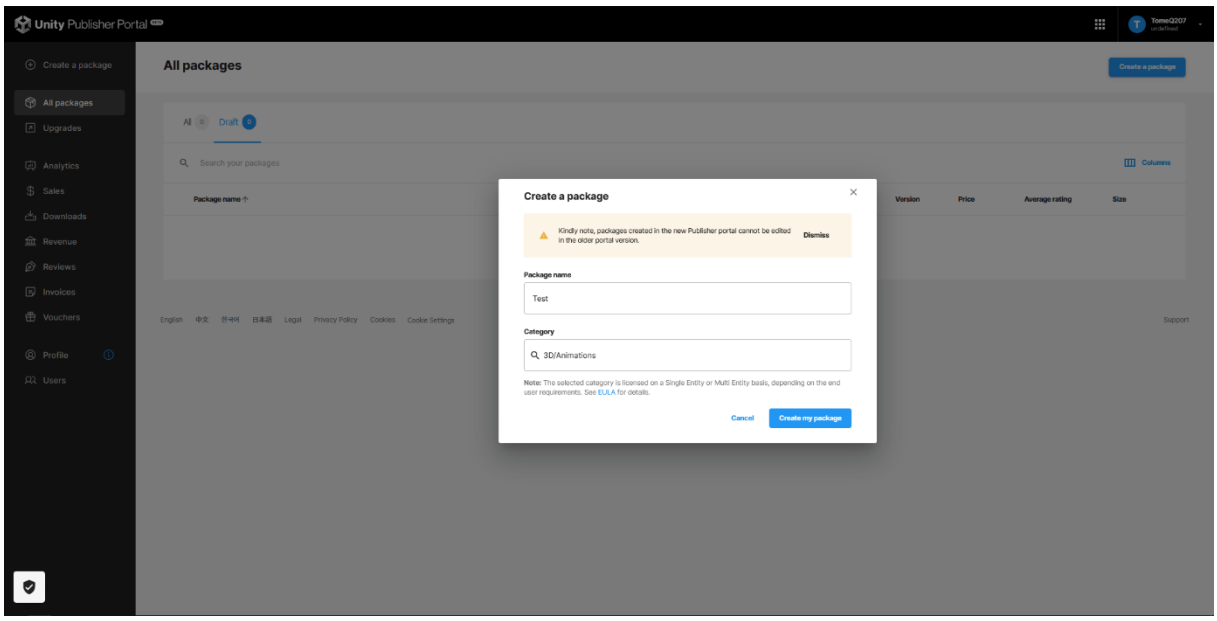
2.1.3. Skladište imovine

Na Slika 10 prikazano je *Skladište imovine* (engl. *Unity Asset Store*). To je trgovina i skladište svih imovina koje Unity i razni korisnici stvaraju te objavljuju, što uključuje razne vrste tekstura, animacija, 2D/3D modela, cijelih već gotovih projekata, uputstva i proširenja.



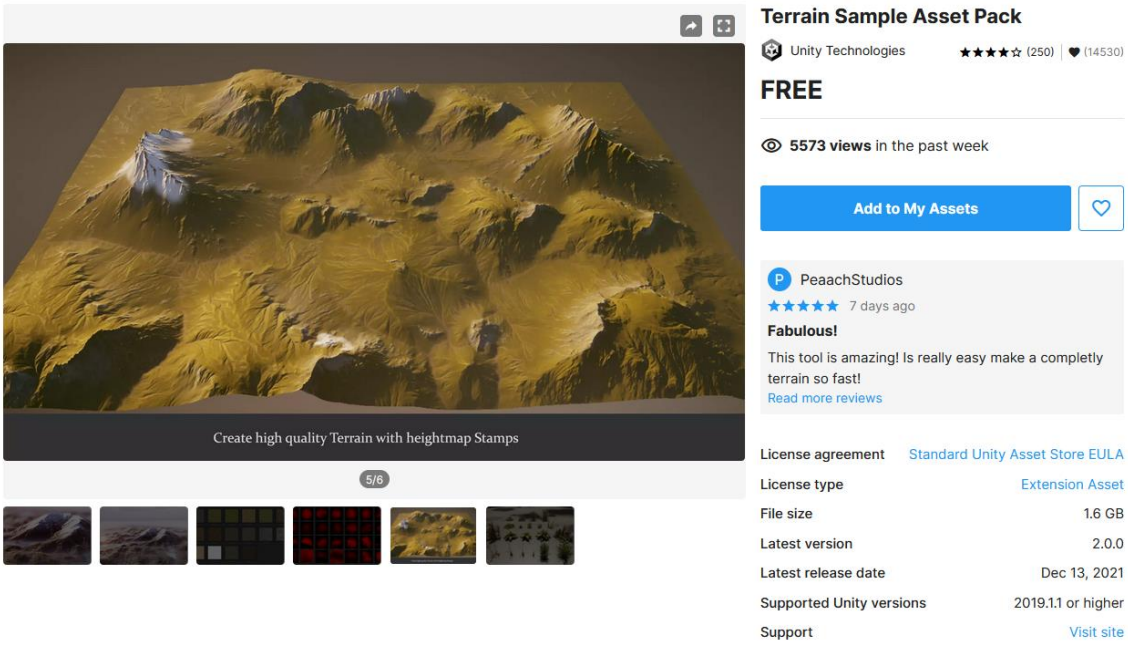
Slika 10. Početna stranica (Unity Asset Store, 2022)

Trgovina sadrži dostupne besplatne i komercijalne pakete koje je moguće preuzeti izravno u projekt. Postoje napredniji i zahtjevniji paketi koji su komercijalni. Neki paketi se mogu kreirati izvan Unity okoline, kao npr. 3D modeli, zvuk, fotografije ili bilo koji drugi tip podataka koji Unity podržava. Postoje paketi koje je moguće kreirati samo unutar Unity okoline, a to su *upravljač animacija* (engl. *Animator Controller*), *audio mixer* ili *renderer* tekstura. Svatko tko želi i ima znanja može postati izdavač i prodavati svoje pakete drugim tvrtkama ili korisnicima tako da se registrira na *Unity Publisher Portal* te klikom na kreiranje novog paketa kao što je prikazano na Slika 11 kreira i objavi željeni paket.



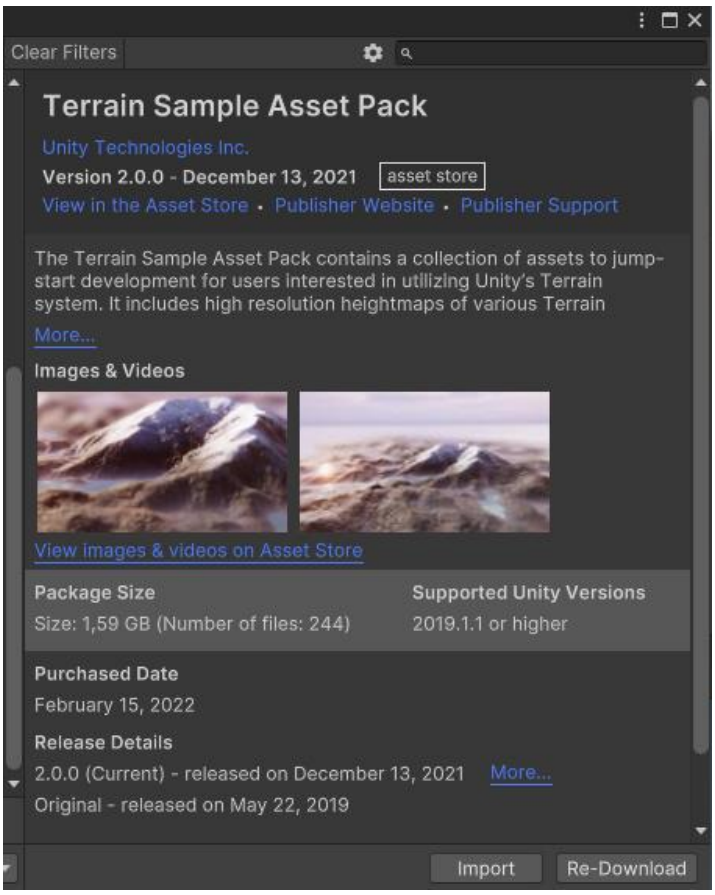
Slika 11. Kreiranje novog paketa (Unity Publisher Portal, 2022)

Preuzimanje paketa prikazano je na Slika 12. Klikom na tipku *Add to My Assets* paket se dodaje u grupu paketa koja je vezana za korisnički račun. Nakon toga paket je dostupan za uvoz unutar Unity razvojnog okruženja.



Slika 12. Dodavanje paketa (Unity Asset Store, 2022)

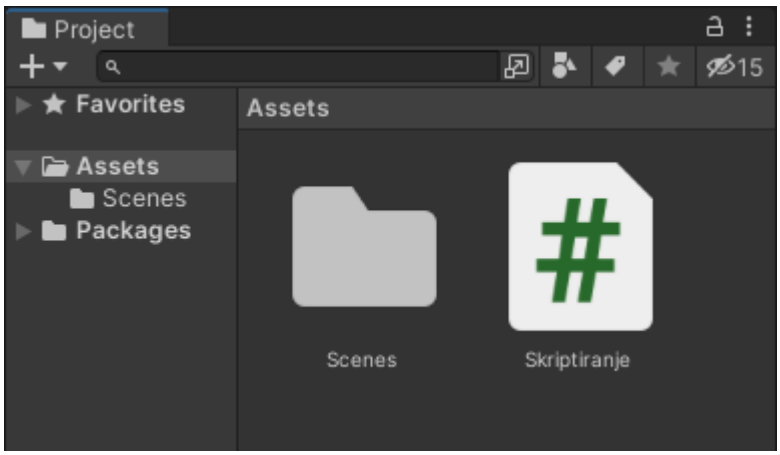
Na Slika 13 prikazano je ubacivanje paketa u projekt. U *Packet Manageru* se dobiva pregled svih paketa koji su vezani za korisnički račun. Nakon pregleda i odabira željeni paket se preuzima te, klikom na *uvezi* (engl. *Import*), on postaje dostupan u projektu.



Slika 13. Ubacivanje paketa u projekt

2.1.4. Skriptiranje

Skriptiranje (engl. *Scripting*) je bitna stavka svih aplikacija koje su napravljene u Unityju. Većina aplikacija treba skripte koje su odgovorne za unos od strane korisnika i kako bi se *događaj* (engl. *Event*) dogodio u određenom trenutku. Događaj je radnja koja klasama ili objektima daje obavijest da se nešto dogodilo. Osim navedenoga, skripte se mogu koristiti za stvaranje grafičkih efekata, kontrolu fizičkog ponašanja objekata ili čak implementaciju *umjetne inteligencije* (engl. *Artificial Intelligence AI*) za likove u video igri. Za razliku od većine paketa, skripte se kreiraju unutar Unityja desnim klikom u *Assets* mapi na *Create* te odabirom stavke *C# Script* iz kontekstnog izbornika, nakon čega se zadaje ime i skripta je spremna za korištenje, kao što je prikazano na Slika 14.



Slika 14. Kreiranje skripte

Prilikom stvaranja skripte, Unity kreira dvije specifične funkcije: `Start` i `Update`. Funkcija `Start` poziva se prije početka simulacije odnosno igranja, pa je ona idealno mjesto za inicijalizaciju objekata. Unutar funkcije `Update` nalazi se kôd koji se izvodi svakim brojem sličica u sekundi (engl. *Frame rate*), zbog različitog vremena renderiranja sličica u sekundi, vremenski period u kojem se `Update` poziva nije uvijek isti.

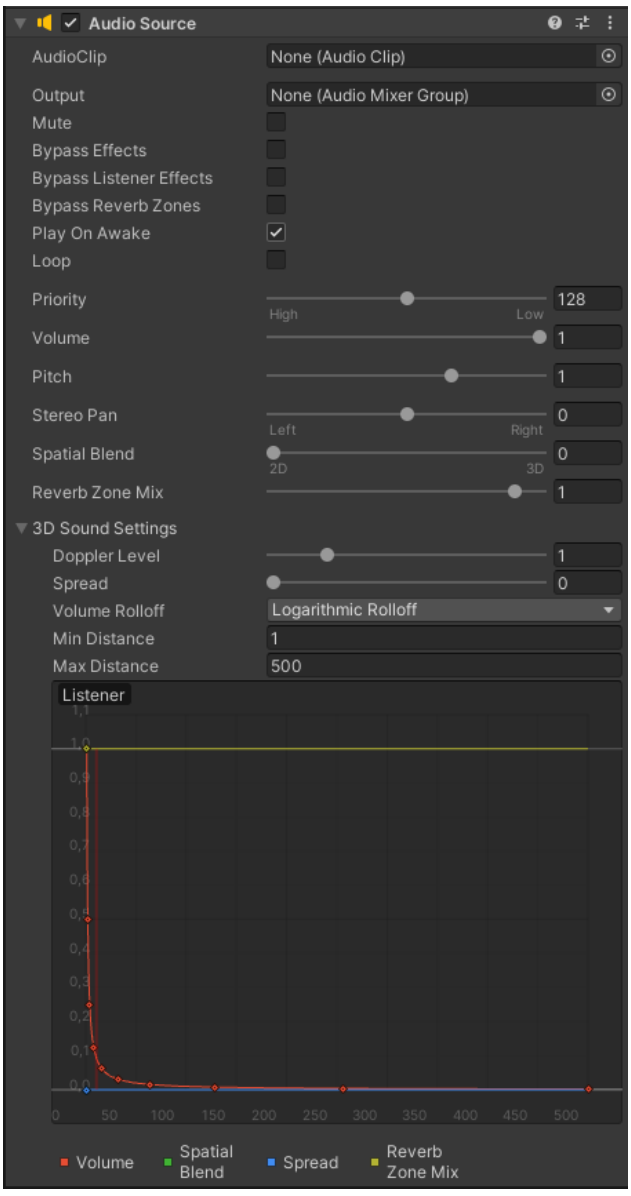
Neke od najvažnijih ugrađenih klasa specifičnih za Unity su:

- *GameObject:* Osnovna klasa za sve objekte koji mogu postojati u sceni.
- *MonoBehaviour:* Osnovna klasa iz koje se izvodi svaka skripta.
- *Object:* Osnovna klasa za sve objekte koje Unity može referencirati u editoru.
- *Transform:* Sadrži informacije i omogućuje rad s pozicijom, rotacijom i mjerilom objekta.
- *Vectors:* Klase kojima se prikazuje smjer i veličina, a služe za manipulaciju 2D i 3D točkama, linijama, pravcima.
- *Quaternion:* Predstavlja apsolutnu ili relativnu rotaciju i pruža metode za stvaranje i manipulaciju njima.
- *SciptableObject:* Spremnik podataka koji se koristi za spremanje velikih količina podataka.
- *Time:* Omogućuje mjerenje i kontrolu vremena te upravljanje brojem sličica u sekundi (engl. *Frame Rate*).
- *Mathf:* Zbirka matematičkih funkcija, uključujući trigonometrijske funkcije, logaritme i druge funkcije koje su potrebne za razvoj.
- *Random:* Pruža jednostavan način generiranja slučajnih vrijednosti.

- *Debug:* Omogućuje vizualizaciju informacija u editoru koje pomažu razumijevanju ili istraživanju što se događa u projektu dok je pokrenut.
- *Gizmos and Handles:* Omogućuju crtanje linija i raznih oblika u sceni i prikazu igre, kao i interaktivne kontrole.

2.1.5. Zvuk

Na Slika 15 prikazan je *Izvor zvuka* (engl. *Audio Source*) koji služi za reproduciranje audio zapisa u određenoj sceni. Izabrani zvuk može se reproducirati koristeći *Slušatelja zvuka* (engl. *Audio Listener*) ili *Audio mikser* (engl. *Audio Mixer*). Izvor zvuka reproducira bilo koju vrstu audio zapisa i može se konfigurirati za reprodukciju 2D, 3D ili kao mješovito. Pojedinačni filteri mogu se primijeniti na svaki audio izvor za još bolje audio iskustvo.



Slika 15. Prikaz Izvora zvuka

Svojstva izvora zvuka su:

- *Audio Clip:* Referenca na datoteku zvučnog isječka koji će se reproducirati.
- *Output:* Po podrazumijevanim postavkama , isječak se šalje direktno u Audio Listener, ali postoji mogućnost promjene tako da output bude vezan za Audio Mixer.
- *Mute:* Ako je uključen, zvuk se neće reproducirati.
- *Bypass Effects:* Služi za zaobilaženje efekata filtera primijenjenih na audio izvor.
- *Bypass Listener Effecsts:* Služi za zaobilaženje svih slušnih efekata.
- *Bypass Reverb Zones:* Koristi se za uključivanje i isključivanje zona s jekom.
- *Play On Awake:* Ako je opcija uključena, zvuk će se početi reproducirati čim se scena učita, a ako je isključena mora se pokrenuti unutar koda korištenjem funkcije Play().
- *Loop:* Omogućuje opetovanu reprodukciju.
- *Priority:* Određuje prioritet audio izvora među svima koji su u sceni, pri čemu 0 odgovara najvišem najvažnije, a 256 najnižem prioritetu. Podrazumijevani prioritet je 128.
- *Volume:* Glasnoća zvuka.
- *Pitch:* Promjena visine tona promjenom brzine reprodukcije. 1 je normalna brzina reprodukcije.
- *Stereo Pan:* Postavlja omjer jačine lijevog i desnog kanala u dvodimenzionalnoj reprodukciji.
- *Spatial Blend:* Postavlja koliko 3D mehanizam ima utjecaj na izvor.
- *Reverb Zone Mix:* Kontrolira jačinu jeke.
- *3D Sound Settings:* Postavke koje se primjenjuju proporcionalno na svojstvo Spatial Blend.

3. FIZIKALNI ZAKONI KORIŠTENI U PROJEKTU

U ovom poglavlju detaljno su opisani fizikalni zakoni korišteni u projektu. Jedan od njih je Newtonov zakon gravitacije koji se u radu koristi za izračunavanje gravitacije između dvaju međusobnih planeta, a drugi je brzina kruženja koja služi za izračunavanje putanje orbite planeta.

3.1. Newtonov zakon gravitacije

Newtonov zakon gravitacije je fizikalni zakon koji opisuje pojavu općeg privlačenja među svim tijelima u svemiru. Smatra se najveličanstvenijim poopćenjem koje je ikad ljudski um učinio. Kao što je ilustrirano na Slika 16, Newtonov zakon gravitacije iskazuje da svako tijelo mase m_1 privlači drugo tijelo mase m_2 silom koja je proporcionalna masama tih tijela, a obrnuto proporcionalna kvadratu njihove međusobne udaljenosti r:

$$F = G\frac{m_1 m_2}{r^2}$$

gdje su:

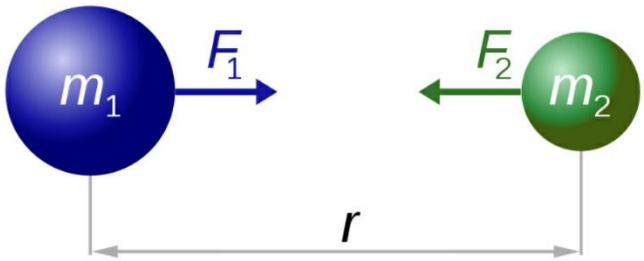
F – gravitacijska sila između dvaju tijela, u njutnima (N)

G - univerzalna gravitacijska konstanta, $G = 6.67 \times 10^{-11}$ N kg^{-2} m^2

m_1 - masa prvog tijela, u kilogramima (kg)

m_2 - masa drugog tijela, u kilogramima (kg)

r - međusobna udaljenost između središta dvaju tijela, u metrima (m)



Slika 16. Ilustracija Newtonovog zakona gravitacije (Wikipedia, 2022)

3.2. Brzina kruženja

Brzina kruženja ili prva kozmička brzina je orbitalna brzina nebeskog tijela u putanji. Ako je planetarna putanja tijela kružna, brzina kruženja je stalna. Za Zemlju (M = 6 ·10^{24} kg) brzina

kruženja na samoj površini (r = 6 378 km) iznosila bi 7.91 km/s. Ta se brzina naziva prvom kozmičkom brzinom i može se izračunati sljedećim izrazom:

$$v = \sqrt{\frac{G \cdot M}{r}}$$

gdje su:

G - univerzalna gravitacijska konstanta, $G = 6.67 \times 10^{-11}$

M - masa izvora gravitacijskog polja, u kilogramima (kg)

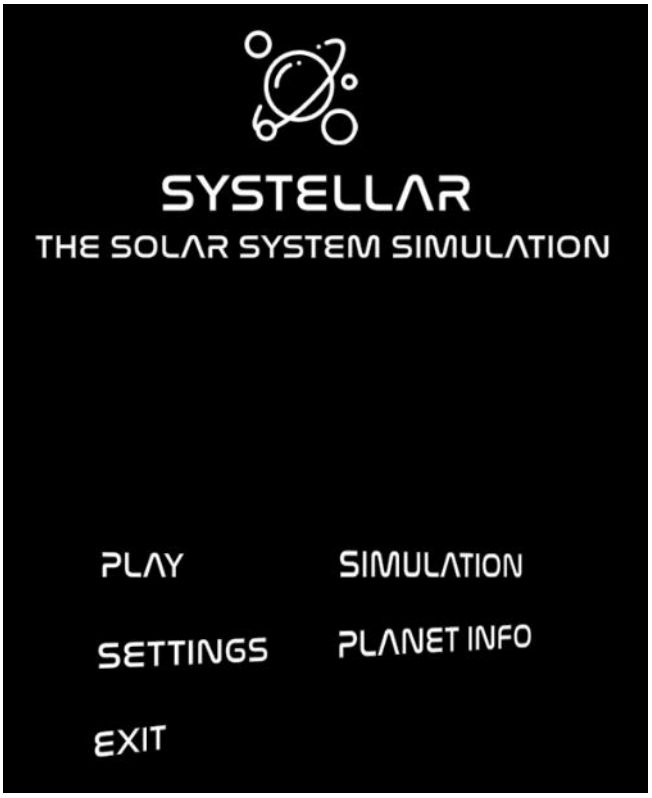
r - polumjer planetarne putanje, u metrima (m)

4. IZRADA APLIKACIJE

U ovom poglavlju detaljno su opisani koraci izrade aplikacije, počevši od glavnog izbornika u kojem se opisuje kako su kreirane tipke, efekti za pozadinu i animacije tipaka. Također je opisano postavljanje redoslijeda scena u aplikaciji. Potom se opisuje kreiranje postavki aplikacije koje korisniku pružaju mogućnost izbora željene rezolucije, kvalitete, jačine zvuka te mogućnost punog zaslona. Slijedi zaseban opis svake scene, odnosno scene simulacije koja je najbitnija stavka cijele aplikacije, te scena u kojoj se nalaze informacije o pojedinim planetima. Opisane su pauze u svakoj od tih scena, te za kraj, izrada izvršne datoteke koja omogućava korištenje aplikacije bez upotrebe Unity razvojnog okruženja.

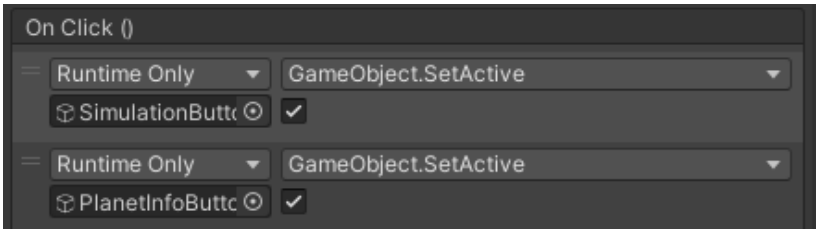
4.1. Glavni izbornik

Prvo što korisnik primijeti prilikom pokretanja aplikaciju je *glavni izbornik* (engl. *Main Menu*). Kao što je vidljivo na Slika *17*, glavni izbornik sadrži logo aplikacije, ime aplikacije te *tipke* (engl. *Button*). Klikom na *Play* tipku pojavljuju se još dvije tipke od kojih je jedna *Simulation* koji vodi na scenu simulacije sunčevog sustava, a druga *Planet Info* koja vodi u scenu koja prikazuje planete i informacije o njima. Tipka Settings prikazuje sve postavke aplikacije. Klikom na tipku *Exit* izlazi se iz aplikacije.



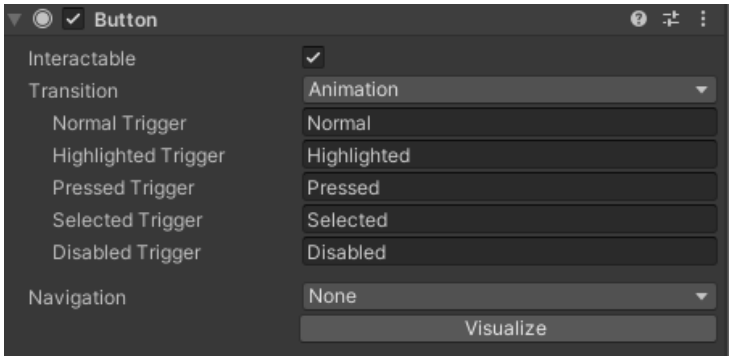
Slika 17. Prikaz glavnog izbornika

Tipka u Unityju ima ugrađen događaj `OnClick` koji se pokreće kada korisnik pritisne tipku. Tom događaju treba pridružiti metodu koja će biti pozvana kod pritiska na tipku. Kao što je vidljivo na Slika 18, u razvojnom okruženju dovoljno je izabrati koji objekt će biti referenca i što će se događati s tim objektom prilikom klika. Tako se mogu uključivati i isključivati skripte koje su vezane za određen objekt ili manipulirati objektima korištenjem opcija aktiviranja, brisanja, dodavanja, mijenjanje boja itd.



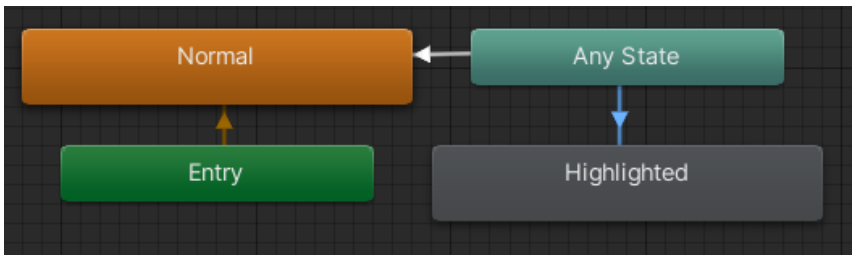
Slika 18. Obrada događaja `OnClick`

U ovom radu za animaciju tipki koristi se ugrađeni sustav zvan *animacije* (engl. *Animations*). Kao što je vidljivo na Slika 19, tipka sadrži tranziciju animacije i po podrazumijevanim postavkama Unity stvara animacije vezane za sve okidače tipki, odnosno *Normal*, *Highlighted*, *Pressed*, *Selected*, *Disabled*.



Slika 19. Prikaz tranzicija tipke

U ovom slučaju korišteni su samo okidači *Normal* i *Highlighted*, kao što je prikazano na Slika 20. Napravljena je jednostavna animacija koja povećava veličinu cijele tipke kada je okidač *Highlighted*, te se veličina vraća na početnu veličinu kada je okidač *Normal*, čime se dobiva željena animacija.



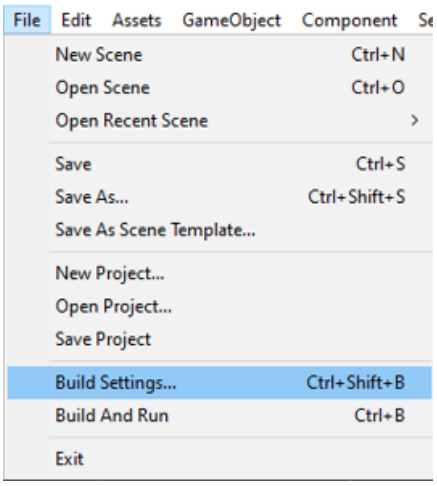
Slika 20. Prikaz animacija tipke

Za pozadinu glavnog izbornika kreirana je simulacija koja prikazuje iskrivljenost svjetla (engl. *Warp Effect*). Na Slika 21 prikazane su postavke *sustava čestica* (engl. *Particle System*) koji je korišten za izradu efekta iskrivljenosti.

Slika 21. Prikaz sustava čestica

Trajanje je postavljeno na pet sekundi i uključeno je opetovano emitiranje (engl. *Looping*) kako bi efekt trajao cijelo vrijeme dok korisnik ne pređe na drugu scenu. Početna veličina postavljena je na 0.12, dok je ukupan broj čestica koje će se emitirati 2000. Kako bi se postigao željeni efekt, u postavkama oblika cijeli sustav čestica postavljen je u obliku stošca, s kutem 5 i polumjerom 15.

Prijelaz sa scene na scenu klikom na određenu tipku napravljen je korištenjem skripte. Prije programiranja potrebno je složiti određene scene prema redoslijedu, odabirom stavke Build Settings u izborniku File, kao što je prikazano na Slika 22. Na Slika 23 prikazan je redoslijed scena ove aplikacije. Glavni izbornik je prvi, slijedi simulacija te na kraju informacije o planetima. Kôd *1* sadrži dvije funkcije zadužene za mijenjanje scena: `SimulationLoad` učitava scenu simulacije, dok `PlanetInfoLoad` učitava scenu na kojoj se nalaze informacije o planetima.



Slika 22. Prikaz build settings

Slika 23. Prikaz redoslijeda scena

```
public void SimulationLoad()
{
    SceneManager.LoadScene(1);
}

public void PlanetInfoLoad()
{
    SceneManager.LoadScene(2);
}
```

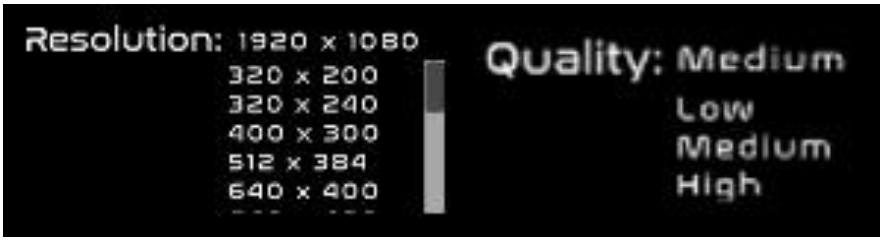
Kôd 1. Mijenjanje scena

4.2. Postavke

Pri pokretanju aplikacije, korisnik bi najprije trebao pregledati postavke kako bi bio siguran da će aplikacija raditi ispravno i po njegovim željama. Kao što je vidljivo na Slika 24, težište postavki je na osnovnim stvarima poput rezolucije, kvalitete, prikaza preko cijelog zaslona (engl. *Fullscreen*), te jačine zvuka.

Slika 24. Prikaz postavki

Na Slika 25 prikazan je prozor koji sadrži padajući izbornik s više mogućih rezolucija kako bi korisnik mogao izabrati onu koja mu najviše odgovara. Podrazumijevana vrijednost je najveća rezolucija korisnikovog monitora. Također, na Slika 25 vidljivo je da *Quality* sadrži padajući izbornik u kojem je moguće izabrati tri opcije kvalitete slike: Low, Medium i High. Podrazumijevana vrijednost je Medium. *Fullscreen* omogućava prikaz preko cijelog zaslona. Podrazumijevana vrijednost je puni zaslon (*engl. Fullscreen*). *Volume* sadrži traku po kojoj je moguće namještati jačinu zvuka od 0 do 1. Podrazumijevana vrijednost je 0.5.



Slika 25. Prikaz rezolucije i kvalitete

Prilikom prvog pokretanja aplikacije rezolucija se postavlja na maksimalnu rezoluciju monitora kojeg korisnik koristi. Ako korisnik promijeni postavke, one se pamte pri izlasku iz aplikacije, te će biti učitane pri sljedećem pokretanju. Unity sadrži ugrađenu funkciju `Screen.SetResolution` kojoj se zadaje širina i visina ekrana, te koristi li se puni zaslon kao što je prikazano **Error! Reference source not found.**.

```
private void ResolutionAwake()
{
    Screen.SetResolution(Screen.currentResolution.width,
                         Screen.currentResolution.height, true);
}
```

Kôd 2. Početna rezolucija

```
    resolutions = Screen.resolutions.Where(resolution =>
      resolution.refreshRate == 60).ToArray();
    resolutionDropdown.ClearOptions();

    List<string> options = new();

    for (int i = 0; i < resolutions.Length; i++)
    {
        string option = resolutions[i].width + " x " +
          resolutions[i].height;
        options.Add(option);

        if (resolutions[i].width == Screen.currentResolution.width &&
            resolutions[i].height == Screen.currentResolution.height)
        {
            currentResolutionIndex = i;
        }
    }
    resolutionDropdown.AddOptions(options);
    resolutionDropdown.value = currentResolutionIndex;
    resolutionDropdown.RefreshShownValue();
    qualityDropdown.value = 1;
```

Kôd 3. Kreiranje popisa rezolucija

Kako bi saznali koje su sve moguće rezolucije, Kôd 3 prikazuje funkciju `ResolutionStart`, koja se poziva prilikom svakog pokretanja aplikacije. Njezina zadaća je da prođe kroz sve moguće rezolucije koje sadrži statički član `resolutions` Unity klase `Screen` i doda ih u listu mogućeg odabira.

```
public void SetResoltion(int resolutionIndex)
{
    Resolution resolution = resolutions[resolutionIndex];
    Screen.SetResolution(resolution.width, resolution.height,
                         Screen.fullScreen);
}

public void SetQuality(int qualityIndex)
{
    QualitySettings.SetQualityLevel(qualityIndex);
}
```

Kôd 4. Postavljanje rezolucije i kvalitete

prikazuje funkcije `SetResolution` i `SetQuality`. One primaju cijeli broj tipa `int` koji predstavlja indeks na kojem se nalazi željena rezolucija ili kvaliteta. Kvaliteta se postavlja preko statičke metode `SetQualityLevel` Unity klase `QualitySettings`.

```
public void VolumeSet (float volume)
{
    AudioListener.volume = volume;
    volumeTextValue.text = volume.ToString("0.0");
}

public void VolumeApply()
{
    PlayerPrefs.SetFloat("mainVolume", AudioListener.volume);
}
```

Kôd 5. Postavljanje zvuka i spremanje vrijednosti

Kôd 5 prikazuje funkcije `VolumeSet` i `VolumeApply`. Funkcija `VolumeSet` postavlja jačinu zvuka na željenu vrijednost zadanu prilikom korištenja aplikacije, dok funkcija `VolumeApply`

sprema željenu vrijednost koristeći ugrađenu klasu `PlayerPrefs` kojom postavlja vrijednost na točno određenu lokaciju ovisno o operativnom sustavu.

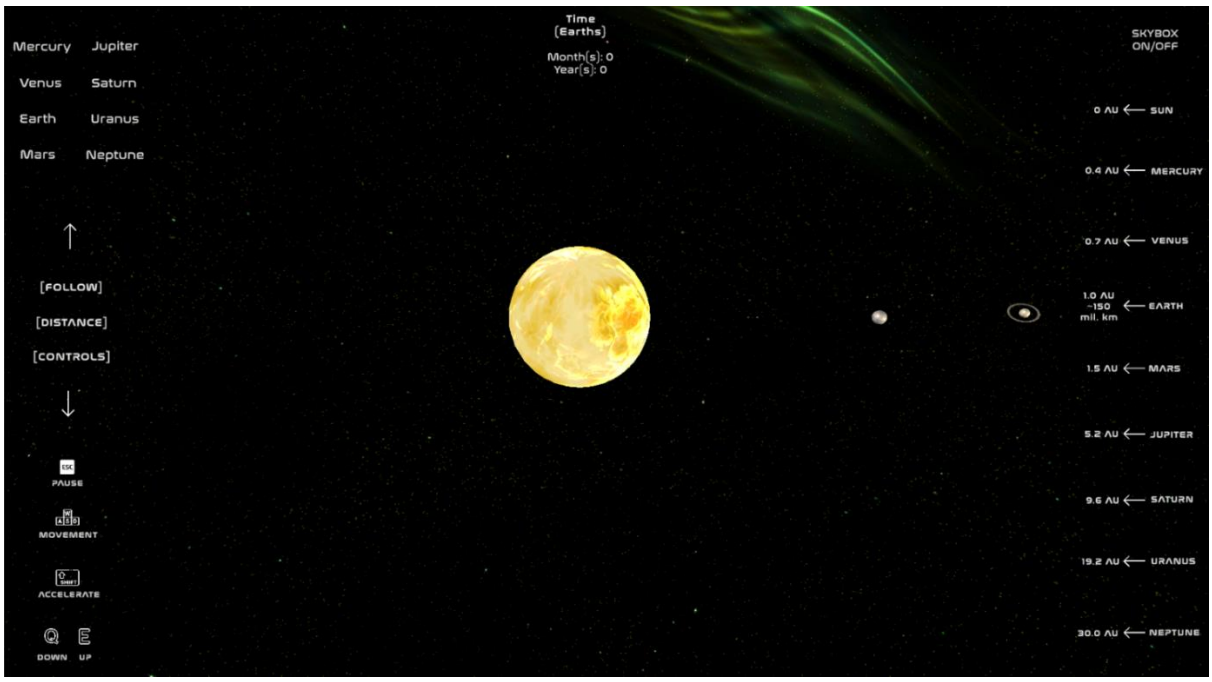
```
qualityDropdown.value = 1;
QualitySettings.SetQualityLevel(1);

Resolution currentResolution = Screen.currentResolution;
Screen.SetResolution(currentResolution.width,
                     currentResolution.height, Screen.fullScreen);
resolutionDropdown.value = resolutions.Length;

SetFullScreen(true);
fullScreenToggle.isOn = true;

AudioListener.volume = defaultVolume;
volumeSlider.value = defaultVolume;
volumeTextValue.text = defaultVolume.ToString("0.0");
VolumeApply();
```

Kôd 6. Vraćanje na podrazumijevane postavke

Klikom na tipku *Reset to default* sve postavke se vraćaju na podrazumijevanu vrijednost. Kôd 6 prikazuje funkciju `ResetButton` koja vraća željene varijable na podrazumijevane vrijednosti.

4.3. Simulacija

Na Slika 26 prikazana je scena simulacije na kojoj se nalaze planeti te određene upute vezane za simulaciju. Tipka *Follow* otvara listu s imena svih planeta, te se klikom na određeni planet prelazi u stanje praćenja željenog planeta. Tipkom *Controls* prikazuju se kontrole u simulaciji, a *Distance* udaljenost planeta od sunca izražena u astronomskim jedinicama (engl. *Astronomic Units* – AU, pri čemu je 1 AU ≈ 150 mil. km). U simulaciji se nalazi brojač mjeseci i godina kako bi cijela simulacija imala realniji osjećaj. Postoji mogućnost deaktivacije pozadinske fotografije (engl. *Skybox*), kako bi se planeti bolje vidjeli. Za pozadinu je u ovoj aplikaciji korišteno zvjezdano nebo.



Slika 26. Prikaz scene simulacije

Kao što je prije navedeno, za orbitu planeta korištene su dvije formule, jedna je vezana za Newtonov zakon gravitacije, a druga za brzinu kruženja. Svaki planet je označen (engl. *Tag*) kao planet te se ta oznaka koristi u skripti za simulaciju, što omogućuje manipulaciju podacima koji su vezani za objekt s određenom oznakom. Za početak deklarirana je gravitacijska konstanta, kreiran niz u koji se spremaju svi planeti kako bi mogli pristupiti svakome od njih te izračunata gravitacija i brzina kruženja za svaki planet. Kako bi simulacija ispravno funkcionirala, potrebno je pravilno postaviti udaljenosti planeta od sunca, polumjere, te mase sunca i planeta.

```
private void Velocity()
{
    foreach (GameObject x in planets)
    {
      foreach (GameObject y in planets)
      {
         if (!x.Equals(y))
         {
           float m = y.GetComponent<Rigidbody>().mass;
           float r = Vector3.Distance(x.transform.position,
                                      y.transform.position);

           x.transform.LookAt(y.transform);

           x.GetComponent<Rigidbody>().velocity +=  x.transform.right *
                                                    Mathf.Sqrt((G * m) /
                                                               r);
         }
      }
    }
}
```

Kôd 7. Izračun brzina kruženja

Kôd 7 prikazuje funkciju `Velocity` kojom se izračunavaju brzine kruženja tako što uspoređuje dva planeta koji se nalaze jedan pored drugog, uzima masu drugog planeta, izračunava udaljenost između njih, te korištenjem formule za brzinu i svojstva *velocity* izračunava brzinu.

```
private void Gravity()
{
    foreach (GameObject x in planets)
    {
      foreach (GameObject y in planets)
       {
         if (!x.Equals(y))
         {
           float m1 = x.GetComponent<Rigidbody>().mass;
           float m2 = y.GetComponent<Rigidbody>().mass;
           float r = Vector3.Distance(x.transform.position,
                                      y.transform.position);

           x.GetComponent<Rigidbody>().AddForce((y.transform.position -
           x.transform.position).normalized * (G * (m1 * m2) /
                                               Mathf.Pow(r, 2)));
         }
       }
     }
}
```

Kôd 8. Izračun gravitacija

Kôd 8 prikazuje funkciju `Gravity` koja je slična funkciji `Velocity`, samo što se u njoj učitavaju mase oba planeta te se korištenjem ugrađene Unity funkcije `AddForce` izračunava gravitacija.

4.4. Pauza

Pauza je bitna stavka svake videoigre, pa tako i ove aplikacije. Pritiskom na tipku ESC prikazuje se panel za pauzu, kao što je prikazano na Slika 27, na kojem se nalaze četiri tipke. Tipka *Resume* nastavlja simulaciju, *Planet Info* prebacuje scenu na informacije o planetima, *Main Menu* prebacuje scenu na glavni izbornik iz kojeg dalje postoji mogućnost biranja željene scene ili postavke. Klikom na *Exit* aplikacija se zatvara.

Slika 27. Prikaz pauza u sceni simulacije

```
public void Resume()
{
    mainCanvas.SetActive(true);
    pauseMenuUI.SetActive(false);
    Time.timeScale = 1f;
    gameIsPaused = false;
}

private void Pause()
{
    mainCanvas.SetActive(false);
    pauseMenuUI.SetActive(true);
    Time.timeScale = 0f;
    gameIsPaused = true;
}
```

Kôd 9. Nastavak i zaustavljanje aplikacije

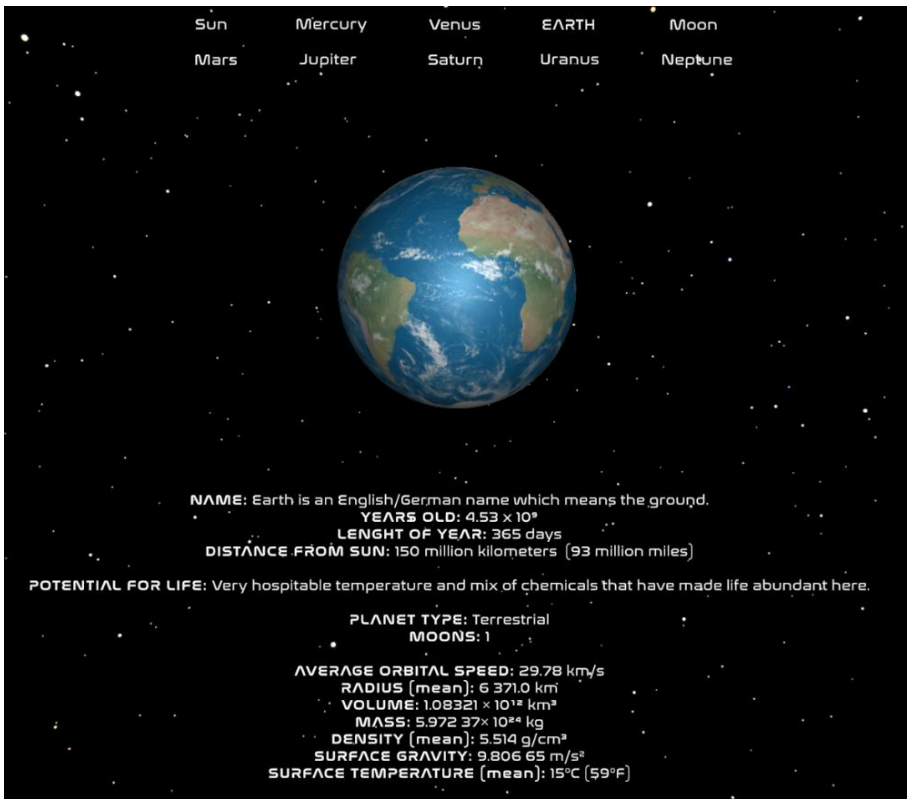
```
private void KeyCodeCheck()
{
    if (Input.GetKeyDown(KeyCode.Escape))
    {
       if (gameIsPaused)
       {
         Resume();
       }
       else
       {
         Pause();
       }
    }
}
```

Kôd 10. Provjera pritisnute tipke

Funkcije `Resume` i `Pause` prikazane su Kôd 9 i koriste se u Kôd 10 koji prikazuje funkciju `KeyCodeCheck` koja provjerava je li na tipkovnici pritisnuta tipka ESC i je li aplikacija zaustavljena. Ako je aplikacija zaustavljena, poziva se funkcija `Resume` koja nastavlja aplikaciju, u protivnom se poziva funkcija `Pause` koja zaustavlja aplikaciju.

4.5. Planet Info

Scena *Planet info* sadrži Sunce i sve planete, te osnovne informacije o njima. Kao što je vidljivo na Slika 28, na sredini ekrana nalazi se izabrani planet koji se rotira, ispod njega nalazi se tekst s određenim korisnim informacijama. Na vrhu se nalazi izbor planeta. Klikom na određeni planet pojavljuje se prikaz planeta i informacije o njemu.



Slika 28. Prikaz scene planet info

```
private void Caps()
{
    if (planetPanel.activeInHierarchy)
    {
      planetText.fontStyle = FontStyles.UpperCase;
    }
    else
    {
      planetText.fontStyle = FontStyles.Normal;
    }
}
```

Kôd 11. Mijenjanje scena

```
private void LateUpdate()
{
    transform.Rotate(speed * Time.deltaTime * rotation);
}
```

Kôd 12. Rotacija planeta

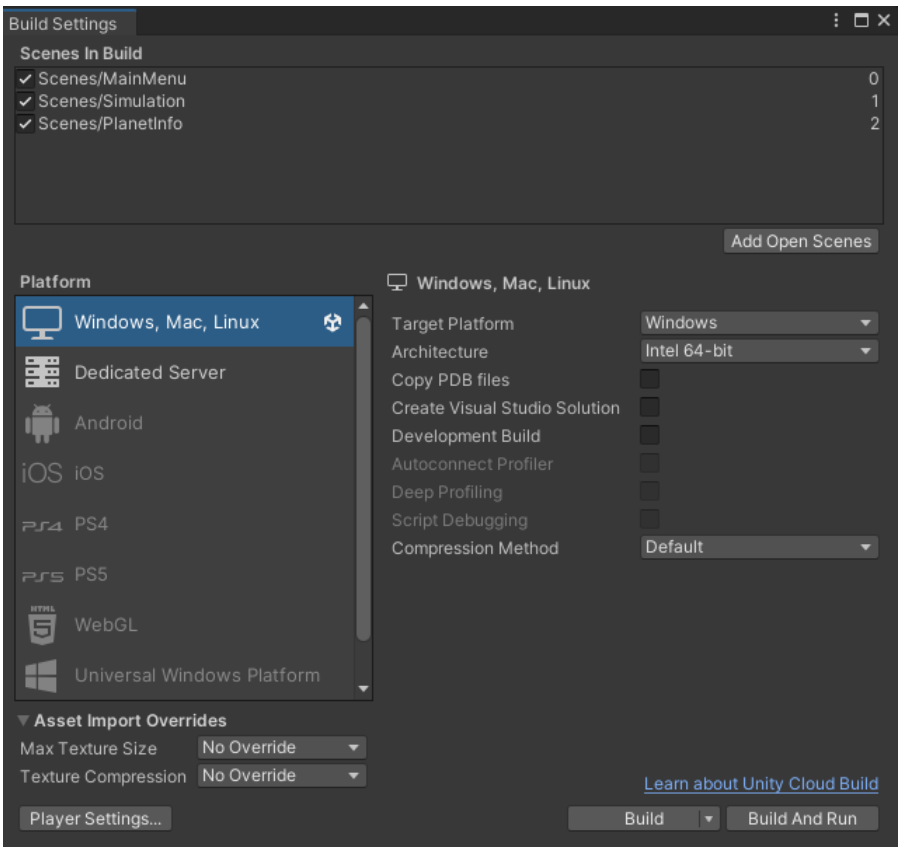
Kako bi se dobio uvid koji je planet trenutno izabran, napravljena je funkcija `Caps`, prikazana Kôd 11. Ako je željeni planet odabran, funkcija postavlja naziv planeta u sva velika slova. prikazuje postavljanje rotacije pozivom metode `Rotate`, koja u ovom slučaju koristi lokalnu varijablu `speed` tipa `int` koja predstavlja brzinu rotacije i lokalnu varijablu `rotation` tipa `Vector3` koja predstavlja os po kojoj se planet rotira.

Slika 29. Prikaz pauze u sceni planet info

Kao i scena simulacije, tako i scena za informacije o planetima sadrži mogućnost zaustavljanja. Pritiskom na tipku ESC prikazuje se panel za pauzu, a jedina razlika je to što je promjena scene sada iz planet info u simulaciju što je vidljivo na Slika 29.

4.6. Izrada izvršne datoteke

Na Slika 30 prikazana je izrada izvršne datoteke (engl. *Build*), koja je jednostavna i u većini slučajeva nije potrebno puno vremena da bi se izrada izvršila, ali sve ovisi o veličini projekta. Unity omogućuje izradu izvršne datoteke za Windows, Mac, Linux, PS4, PS5, Android, iOS, WebGL, tvOS. Ovaj rad napravljen je za operativni sustav Windows. Kao što je navedeno, prije izrade izvršne datoteke potrebno je pristupiti postavkama u kojima se podešava redoslijed scena. Nakon slaganja scena potrebno je izabrati platformu i željenu arhitekturu, te pritisnuti tipku *Build*. Potrebno je izabrati mapu i lokaciju za spremanje izvršne datoteke. Nakon uspješne izrade, aplikacija je spremna za korištenje i pokreće se kao svaka druga aplikacija na računalu.



Slika 30. Prikaz build postavki aplikacije

5. ZAKLJUČAK

Sukus rada je Unity razvojno okruženje i izrada 3D aplikacije koja prikazuje simulaciju Sunčevog sustava. Aplikacija prikazuje kretanje planeta u Sunčevom sustavu, udaljenost planeta od Sunca, te pruža mogućnost praćenja željenog planeta. Prikazane su korisne informacije o planetima kao što su ime, starost, tip, mogućnost života, volumen, masa i slično. Za skriptiranje u praktičnom dijelu projekta korišten je programski jezik C#. Iako je Unity okruženje pogodno za početnike, zanimljivo i zabavno, prethodno iskustvo u programiranju je neophodno kako bi se samo okruženje lakše koristilo.

Izrada ovakve vrste aplikacije je korisna, ali može biti vrlo komplicirana i frustrirajuća osobama koje nisu spremne uložiti dovoljno vremena i truda za savladavanje svih gotovih funkcija koje nudi Unity okruženje. Unity je stekao popularnost jer ima donekle jednostavno korisničko sučelje te gotove funkcije koje olakšavaju posao samom programeru koji je zadužen za funkcionalnost aplikacije, kao i ostatku tima koji se bavi zvukom, animacijama i ostalim segmentima aplikacije ili igre. Kako bi završna aplikacija bila što kvalitetnija, najbitnije je posvetiti dosta vremena i pažnje.

Prilikom izrade nije bilo susreta s nečim previše kompliciranim zbog prijašnjeg iskustva na dva manja projekta. Bez obzira na to, projekt je iziskivao dosta vremena i fokusa na određene stvari kako bi sve izgledalo što kvalitetnije i profesionalnije. Vizualni dizajn same aplikacije mijenjan je u nekoliko navrata, pa je time utrošeno dosta vremena i pažnje na detalje. Također, dosta vremena je uloženo u istraživanje službene Unity dokumentacije, te fizike u samom Unity okruženju koje s godinama sve više i više napreduje, prati potrebe razvojnih inženjera i tvrtki, a može se reći i samih korisnika koji zahtijevaju što kvalitetnije aplikacije i igre.

Kroz ovaj završni rad razvijene su vještine vezane za izradu videoigara, te usavršeno određeno programersko znanje koje je stečeno studirajući i radeći na raznim projektima. S obzirom na budući plan nastavka rada na razvoju videoigara i aplikacija sličnih ovoj, spomenuto stečeno znanje će biti od velikog značenja. Nadogradnja ove aplikacije bi se temeljila na pojednostavljenje samih scena, spajanje scene simulacije i planet info u kojoj bi bilo moguće saznati informacije o planetima tako da bi se prilikom interakcije s određenim planetom pojavljivao prozor s informacijama i detaljima vezanim za izabrani planet, poboljšala bi se kvaliteta tekstura planeta i kreirale kvalitetnije i modernije animacije. Plan je prebaciti aplikaciju u oblik proširene stvarnosti (engl. *Augmented Reality AR*), a da bi to bilo izvedivo

potreban je kompletan redizajn aplikacije. U početku takva aplikacija bila bi namijenjena za Android mobilne uređaje.

LITERATURA

1. Wikipedia.org, Unity (game engine), https://en.wikipedia.org/wiki/Unity_(game_engine) (pristupljeno 29.6.2022.)
2. Unity.com, Unity User Manual, https://docs.unity3d.com/Packages/com.unity.renderpipelines.universal@11.0/manual/ (pristupljeno 1.7.2022.)
3. Unity.com, Unity User Manual, https://docs.unity3d.com/Manual/UsingTheEditor.html (pristupljeno 1.7.2022.)
4. Unity.com, Unity User Manual, https://docs.unity3d.com/Manual/UsingTheSceneView.html (pristupljeno 1.7.2022.)
5. Unity.com, Unity User Manual, https://docs.unity3d.com/Manual/GameView.html (pristupljeno 2.7.2022.)
6. Unity.com, Unity User Manual, https://docs.unity3d.com/Manual/Hierarchy.html (pristupljeno 2.7.2022.)
7. Unity.com, Unity User Manual, https://docs.unity3d.com/Manual/UsingTheInspector.html (pristupljeno 5.7.2022)
8. Unity.com, Unity User Manual, https://docs.unity3d.com/Manual/ProjectView.html (pristupjeno 5.7.2022.)
9. Unity.com, Unity User Manual, https://docs.unity3d.com/Manual/Console.html (pristupljeno 5.7.2022.)
10. Unity.com, Unity User Manual, https://docs.unity3d.com/Manual/AssetStore.html (pristupljeno 5.7.2022.)
11. Unity.com, Unity User Manual, https://docs.unity3d.com/Manual/ScriptingSection.html (pristupljeno 7.7.2022.)
12. Unity.com, Unity User Manual, https://docs.unity3d.com/Manual/class-AudioSource.html (pristupljeno 7.7.2022.)
13. E-skole.hr, Opći zakon gravitacije, https://edutorij.e-skole.hr/share/proxy/alfresco-noauth/edutorij/api/proxy-guest/8b109d99-b37e-4aa4-821c-ab1d3c48e3d6/html/24168_Opci_zakon_gravitacije.html (pristupljeno 12.7.2022.)
14. Wikipedia.org, Brzina Kruženja https://hr.wikipedia.org/wiki/Brzina_kru%C5%BEenja (pristupljeno 12.7.2022.)

SAŽETAK

Današnje vrijeme je takvo da je nemoguće zamisliti život bez mnogih korisnih aplikacija koje ljudi koriste na dnevnoj bazi. Ovom radu pristupilo se s željom da se u edukativne svrhe kreira aplikacija i prikaže simulacija našeg Sunčevog sustava. Opisano je Unity razvojno okruženje, te njegove bitne funkcije bez kojih izrada ne bi bila moguća. Opisan je razvojni proces aplikacije, kreiranje glavnog izbornika, postavk, i na kraju simulacije. U zaključku je izneseno vlastito iskustvo s radom u Unity razvojnom okruženju.

Ključne riječi: Unity, programiranje, C#, sunčev sustav, razvoj aplikacije

SUMMARY

Today's time is such that it is impossible to imagine life without many of the useful applications that people use daily. This work was approached with the desire to create an application and show a simulation of our solar system for educational purposes. The Unity development environment, as well as its essential functions without which development would not be possible are described, following the application development process, the creation of the main menu, the setting, and, finally, the simulation. My own experience regarding working in the Unity development environment was presented in the conclusion.

Keywords: Unity, programming, C#, solar system, application development