Edukativna igra za poučavanje učenika u razvrstavanju otpada

Bahnjik, Dorijan

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

STRUČNI PRIJEDIPLOMSKI STUDIJ RAČUNARSTVO

DORIJAN BAHNJIK

0313023921

EDUKATIVNA IGRA ZA POUČAVANJE UČENIKA U RAZVRSTAVANJU OTPADA

ZAVRŠNI RAD

Čakovec, rujan 2024.

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

STRUČNI PRIJEDIPLOMSKI STUDIJ RAČUNARSTVO

DORIJAN BAHNJIK

0313023921

EDUKATIVNA IGRA ZA POUČAVANJE UČENIKA U RAZVRSTAVANJU OTPADA

EDUCATIONAL GAME FOR TEACHING STUDENTS IN WASTE SORTING

ZAVRŠNI RAD

Mentor:

Nenad Breslauer, viši predavač

Čakovec, rujan 2024.

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU
ODBOR ZA ZAVRŠNI RAD

Čakovec, 16. veljače 2023.

Polje: **2.09 Računarstvo**

ZAVRŠNI ZADATAK br. 2022-RAČ-R-5

Pristupnik: **Dorijan Bahnjik (0313023921)**
Studij: Redoviti preddiplomski stručni studij Računarstvo
Smjer: Inženjerstvo računalnih sustava i mreža

Zadatak: **Edukativna igra za poučavanje učenika u razvrstavanju otpada**

Opis zadatka:

Zadatak je napraviti Edukativnu igru za poučavanje učenika u razvrstavanju otpada. Izraditi scenu te sve potrebne elemente, koristiti mogućnosti koje pruža podsustav za osvjetljenje scene, Implementirati metode kojima okolina igrača postaje interaktivna te reagira na akcije igrača.
Koristiti platformu Unity, programski jezik C# te dodatne programske alate. Završni rad mora sadržavati sažetak, sadržaj i uvod, nakon čega slijedi poglavlje u kojem je potrebno navesti i pojasniti osnovne ciljeve rada te očekivani rezultat. U narednom poglavlju potrebno je opisati primijenjene postupke, alate i metode. Poglavlje koje slijedi obrađivati će postignute rezultate nakon čega slijedi poglavlja u kojem se kritički raspravlja o primijenjenim metodama i postupcima te se u narednom poglavlju iznose glavni zaključci rada. Rad se završava poglavljima s popisom literature te prilozima.

Rok za predaju rada: 20. rujna 2023.

Mentor:

Breslauer

Nenad Breslauer, v. pred.

Predsjednik povjerenstva za završni ispit:

SAŽETAK

Završni rad opisuje proces izrade dvodimenzionalne edukativne računalne igre namijenjene poučavanju učenika o razvrstavanju otpada razvijene u Unity razvojnom okruženju. Igra je jednostavna, efektivna s glavnim ciljem da igrač razvrsta što više otpada u odgovarajuće spremnike unutar zadanog vremenskog ograničenja. Igra koristi kombinaciju tipkovnice i miša za kontrolu. Strelice na tipkovnici koriste se za kontroliranje otpada, a interakcija unutar izbornika kontrolira se mišem. Igrači imaju tri pokušaja pri čemu svaka greška umanjuje broj dostupnih pokušaja. Kada igrač iskoristi sve pokušaje, otvara se „Game Over“ izbornik koji nudi mogućnost ponovnog igranja, povratka na početni meni ili izlaska iz igre. Vizualni su elementi igre jednostavni i prilagođeni mlađim igračima, s naglaskom na jasnoću i prepoznatljivost grafičkih elemenata. Spremnici su vizualno različiti i tekstualno označeni, a otpad se prikazuje u obliku raspoznatljivih ilustracija. Zadatak je brzo i ispravno razvrstati otpad, što igri daje element izazova i potrebu veće koncentracije.

Razvojem igre u Unity razvojnom okruženju omogućen je modularan pristup dizajnu i implementaciji s korištenjem C# skripti za kreiranje funkcionalnosti igre. Implementacija programskog koda obuhvaća logiku razvrstavanja, kontrolu pokušaja igrača te sustav bodovanja koji prati uspješnost. Posebna pažnja posvećena je balansiranju težine igre kako bi bila dovoljno izazovna, ali i dalje pristupačna mlađim igračima. U igru su implementirani i zvučni efekti i glazba koji poboljšavaju doživljaj igranja označavajući uspješno razvrstavanje ili pogrešku.

Osnovni je cilj igre na zabavan način educirati igrače o pravilnom razvrstavanju otpada koristeći se jednostavnim i efektivnim dizajnom. Igra pruža edukativno iskustvo kroz intuitivne kontrole i jasne vizualne smjernice čime se postiže kombinacija edukativnog i zabavnog sadržaja.

Ključne riječi: *Videoigra, Spremnik, Otpad, 2D, Unity, Visual Studio, C#, Razvrstavanje*

ABSTRACT

The final thesis describes the process of creating a two-dimensional educational computer game aimed at teaching students about waste sorting, developed in the Unity development environment. The game is simple yet effective, with the primary goal of having the player sort as much waste as possible into the correct bins within a given time limit. The game uses a combination of keyboard and mouse controls – arrow keys on the keyboard are used to control the waste, while menu interaction is handled with the mouse. Players have a total of three attempts, with each mistake reducing the number of available tries. When the player uses all attempts, a “Game Over” menu opens, offering the options to replay, return to the main menu, or exit the game. The visual elements of the game are simple and tailored to younger players, with a focus on clarity and recognizability of graphic elements. The bins are visually distinct and labeled with text, while the waste is displayed in the form of easily recognizable illustrations. During the game, the player is under time pressure to quickly and correctly sort the waste, adding an element of challenge and concentration. Developing the game in the Unity development environment enabled a modular approach to design and implementation, using C# scripts to create the game's functionality. The code implementation covers the sorting logic, player attempt management, and a scoring system that tracks success. Special attention was given to balancing the game's difficulty, ensuring it is challenging enough yet still accessible to younger players. The game also includes sound effects and music that enhance the experience during gameplay, emphasizing successful sorting or mistakes.

The primary goal of the game is to educate players about proper waste sorting through fun, using a simple yet effective design. The game provides an educational experience through intuitive controls and clear visual guidelines, achieving a blend of educational and entertaining content.

Keywords: *Video Game, Bin, Waste, 2D, Unity, Visual Studio, C#, Sorting*

SADRŽAJ

1. UVOD

Tema je ovog završnog rada izrada 2D edukativne igre za poučavanje učenika u razvrstavanju otpada koristeći razvojnu okolinu *Unity* i programski jezik *C#*. Igra se sastoji od jedne vremenski ograničene razine prilikom koje igrač ima tri pokušaja za krivi odgovor. Potrebno je izraditi scene i sve potrebne elemente koristeći se mogućnostima koje pruža podsustav za osvjetljenje scene. Implementirajući razne metode, okolina igrača postaje interaktivna te reagira na njegove akcije. Cilj igre je igračevom interakcijom razvrstati otpad u odgovarajući koš, odnosno u vremenu od tri minute ostvariti što više bodova uz što manje krivih pokušaja. Kontrola unutar igre ostvaruje se strelicama na tipkovnici (prilikom gameplaya) i mišem (u izbornicima i glavnom meniju). Igrač na raspolaganju ima tri pokušaja. Ukoliko otpad odloži u krivi koš, otpad "izađe" van krajnjih točaka scene ili "padne" na dno igraće scene, igraču se broj pokušaja umanjuje za jedan. Ako igrač iskoristi sva tri pokušaja, pokreće se Game Over izbornik u kojemu igrač odabire restart igre ili povratak na početni meni. Igrač u bilo kojemu trenutku može pritiskom na Escape tipku otvoriti izbornik Pauza te ondje odabrati radnju koju želi izvršiti - nastaviti igru, vratiti se na početni meni ili izaći iz igre na radnu površinu.

Ovaj rad pokriva više elemenata koji su nužni za razvoj videoigre bilo koje vrste. Elementi uključuju razvoj skripta za mehaniku i kontroliranje objekata, fiziku same igre, razvoj tekstura ili *spriteova,* kreiranje korisničkog sučelja i menija te implementaciju zvučnih efekata i pozadinske glazbe.

Između ostaloga, u radu je opisana razvojna okolina Unity i detaljan proces igre u njoj te Microsoft Visual Studio pomoću kojega su napisane skripte i kodovi za funkcionalnost igre.

2. RAČUNALNE IGRE KROZ POVIJEST

Računalne igre kakve danas poznajemo započele su svoj razvoj sredinom prošloga stoljeća. Prva videoigra *Tennis for Two* Williama Hignbothama objavljena je 1958. godine. Iako nije prva objavljena, igra *Spacewar!* iz 1961. godine smatra se prvom važnijom igrom u povijesti. Bila je to igra dvojice studenata Massachusettskog instituta za tehnologiju (MIT) razvijena na mini računalu DEC PDP-1. Igra prikazuje dva svemirska broda, „iglu" i „klin", uključena u zračnu borbu koju kontroliraju igrači. Cilj igre je uništiti „neprijateljsku" letjelicu izbjegavajući pritom torpeda, zvijezde ili drugu letjelicu. [1]

Slika 1. Spacewar! na PDP-1 računalu

Izvor: https://shorturl.at/hlzuJ (pristup: 22.6.2024.)

Početkom sedamdesetih godina započinje era prvih kućnih konzola i igraćih automata. 1972. godine mali tim pod vodstvom Ralpha H. Baera razvio je *Magnavox Odyssey,* prvu kućnu konzolu za videoigre. Iste godine Atari lansira Pong, jednostavnu arkadnu igru tenisa koja postaje iznimno popularna i potiče širenje arkadnih videoigara.

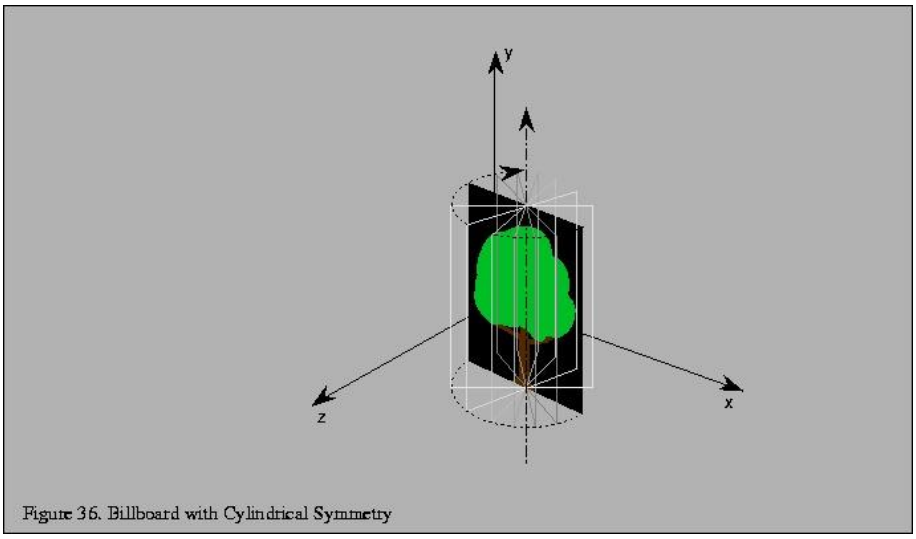
Razdoblje 80-ih zlatno je doba arkadnih igara i kućnih konzola. Dotada su videoigre bile jednostavne i uglavnom tekstualnog oblika. Imale su nekoliko kontrola pomoću kojih je igrač određivao kretanje igraćeg lika. Razvojem sklopovlja i naprednijih grafičkih procesora igre su postale zahtjevnije i grafički kompleksnije. To je dovelo do razvoja novih žanrova videoigara kao što su simulacija letenja i trkaće igre. *Microsoft Flight Simulator* i *Out Run* bile su igre koje su u ono vrijeme iskorištavale puni potencijal rendiranja 3D računalne grafike. Tada je na tržište lansirana jedna od najutjecajnijih igara svih vremena koja je definirala platformerski

žanr. Riječ je o Super Mario Bros-u iz 1985. godine za *Famicom* i *NES*. Osim Super Mario Bros-a 80-ih se godina proslavila i kompanija Nintendo te igre poput *Pac-Man*-a, *Donkey Kong*-a, *Tetris*-a i itd.

90-ih godina lansirane su dvije igre koje smatramo prekretnicom u svijetu videoigara. id Software je 1992. godine izdao igru Wolfenstein 3D. Wolfenstein nije bila prava 3D igra poput suvremenih igara. Bila je to 2D igra koja je pametnom tehnikom zvanom *billboarding* stvarala dojam da je igra ustvari 3D. *Billboarding* je tehnika u računalnoj grafici i videoigrama koja omogućuje stvaranje iluzije trodimenzionalnosti za objekte koji su ravne teksture štedeći pritom računalne resurse. [2]

Slika 2. Wolfenstein 3D

Izvor: https://shorturl.at/ColZB (pristup: 22.6.2024.)



Slika 3. Billboarding

Izvor: https://shorturl.at/mpxYf (pristup: 22.6.2024.)

Prvi pravi 3D engine bio je id Tech1, poznatiji kao Doom Engine. Debitirao je igrom DOOM 1993. godine. Engine je napisan u C programskom jeziku te koristi OpenGL API i Direct3D. Danas se utjecaj DOOM-a i njegove inovativnosti u svijetu First Person Shooter-a može vidjeti u gotovo svim igrama tog žanra poput videoigre Call of Duty-a izrađene na jednoj od iteracija id Tech Engine-a.

Slika 4. DOOM

Izvor: https://shorturl.at/WZwB1 (pristup: 22.6.2024.)

3. EDUKATIVNE IGRE

Zadnjih desetak godina edukativne računalne igre postale su ključni alat u modernom obrazovanju jer spajaju zabavu s učenjem na inovativan način. S pomoću interaktivnih elemenata i izazova igre omogućuju igračima da na zabavan način steknu nova znanja i vještine. Primjerice, igre kojima je u fokusu razvrstavanje otpada podučavaju o važnosti recikliranja i očuvanja okoliša i razvijaju motoričke vještine igrača te sposobnost donošenja brzih odluka. Igrači uče prepoznati različite vrste otpada i pravilno ih razvrstati, što im pomaže da usvoje navike koje će kasnije primjenjivati u stvarnom životu. Jedan je od glavnih benefita ovog tipa igara je njihova sposobnost da se prilagode individualnim potrebama igrača. Mnoge igre nude različite razine težine i zadatke prilagođene različitim tipovima igrača. Isto tako, igre često imaju elemente nagrađivanja kao što su bodovi, bedževi ili virtualna valuta. Ovi elementi potiču igrača da nastavi učiti i napredovati vlastitim tempom i načinom koji mu najviše odgovara. Budući da su edukativne igre kao žanr opširne, možemo ih podijeliti u nekoliko potkategorija. Stoga danas postoje: matematičke igre poput *2048* i *Sudoku*; geografske igre poput igre *GeoGuessr* koje pomažu igraču pronaći lokacije na Zemlji uz pomoć savjeta unutar igre, zatim *Wordscapes* igre u kojima igrač spaja slova kako bi formirao riječi koje se uklapaju u zadani mrežni obrazac, igre učenja jezika poput *Duolingo* (lekcije, kvizovi i izazovi s pomoću kojih igrač uči nove riječi, frazeme, gramatiku i izgovor riječi na raznim jezicima) i sl. [3] [4]

Slika 5. Wordle

Izvor: https://shorturl.at/C9THW (pristup: 23.6.2024)

4. ALATI

Pri izradi ovog završnog rada koristio sam se Unity razvojnom okolinom verzije 2022.3.30f1, a za pisanje kodova i skripti Microsoft Visual Studiom verzije 2019. te u manjoj mjeri alatima poput Adobe Photoshopa, Pixlra i Audacitya.

4.1. Unity

Unity je jedna od najpopularnijih razvojnih okolina za izradu dvodimenzionalnih i trodimenzionalnih sadržaja, uključujući videoigre, animacije i virtualnu stvarnost (VR), arhitektonske vizualizacije i sl. Razvio ga je u lipnju 2005. godine Unity Technologies pod vodstvom Davida Helgesona, Nicholasa Francisa i Joachima Antea. Unity je u početku bio osmišljen kao alat za razvoj igara na Appleovoj Mac platformi, no ubrzo je postao kompatibilan s ostalim platformama, uključujući Windows, Linux, iOS, Android te s igraćim konzolama poput Xboxa i Playstationa. Uspješnost Unitya kao razvojne okoline je vidljiva u industrijama izvan videoigara poput automobilske industrije, filmske industrije, arhitekture i građevine. Primarna je karakteristika razvojne okoline omogućiti razvoj igara dostupnih većem broju programera na svim platformama današnjega tržišta. Jezgra Unity razvojne okoline napisana je u C++ programskom jeziku, a korisnički sloj i skriptiranje napisano je u C# programskom jeziku. Kako bi se olakšao razvoj developerima igara, na jezgri sustava sagrađen je omotač (eng. *Wrapper*) koji postavlja sloj za pristup .NET jeziku. Tako se olakšava pisanje skripti i korištenje grafičkog sučelja koje Unity pruža. Kako bi se dodatno proširila raznovrsnost same platforme, osiguran je API za razvoj u Javascript-u, C# i Boo-u. [5]

Unity podržava integraciju i s alatima za 3D modeliranje, uređivanje fotografija i sl. Neki od ključnih alata su:

- Adobe Photoshop
- Blender
- Maya i 3ds Max
- Cinema 4D
- AutoCAD

Tijekom vremena izlazile su mnoge verzije Unitya, a prva značajna verzija nakon 1.0 bila je 2.0, službeno objavljenja 10. listopada 2007. godine. Glavne novosti koje je ova

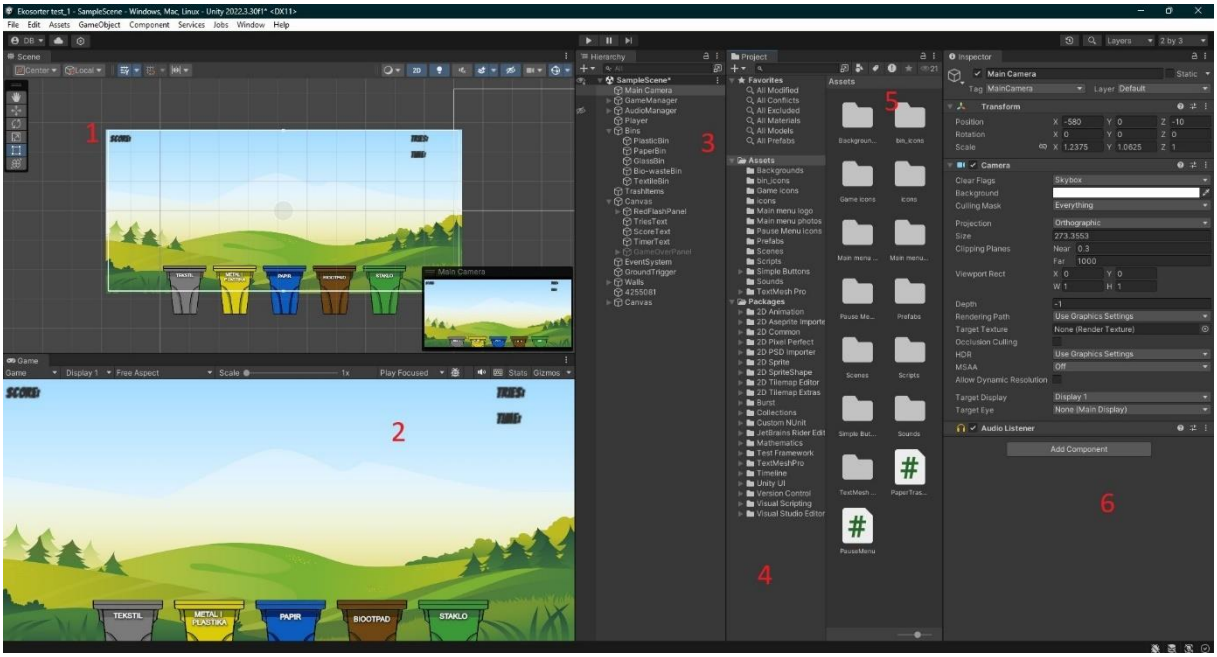
verzija donijela su: osjenčavanje u stvarnom vremenu, sustav za izradu korisničkog sučelja te alati za rad s terenom. Važno je spomenuti i verziju 3.0 čija je najznačajnija novost bila podrška za izradu igara za Android operacijske sustave. Podrška za DirectX dolazi s verzijom 4.0, a verzijom 5.0 podržana je izrada igara za gotovo sve današnje platforme. [6]

Slika 6. Unity logotip

Izvor: https://unity.com/legal/branding-trademarks (pristup: 26.7.2024.)

4.1.1. Unity sučelje

Korisničko sučelje Unitya može se podijeliti na šest različitih prozora, kao što je prikazano Slikom 6. U ovom poglavlju ukratko će se opisati svaki pojedini prozor.



Slika 7. Objašnjenje Unity korisničkog sučelja

Izvor: Autor

Prozor označen brojem jedan je prozor Scena. Unutar ovog prozora kreiramo sve scene unutar igre, odnosno pozicioniramo objekte koji će se koristiti u igri. Ako se kreira dvodimenzionalna igra, ovaj je prozor ograničen na dvodimenzionalni prostor, odnosno ima samo x i y os. Prebacivanjem prozora u trodimenzionalni prostor dobivamo z os, odnosno dubinu. Z os može se koristiti i u 2D igrama prilikom kontroliranja slojeva (eng. *Layer*), primjerice poput slojeva u alatu *Photoshop*.

Prozor broj dva je prozor Igra. Unutar ovog prozora prikazan je izgled igre kada se ona pokrene, odnosno prozor prikazuje igru iz perspektive glavne kamere igre. Unutar prozora moguće je postaviti različite rezolucije i omjere ekrana kako bi se testirao izgled igre na različitim uređajima i zaslonima, primjerice na mobitelima i tabletima. Na alatnoj traci koja se nalazi iznad scene smještene su osnovne kontrole za pokretanje, zaustavljanje i ponovno pokretanje igre. Pomoću tih gumba možemo brzo testirati promjene i ispraviti *bugove,* odnosno pauzirati igru i pregledati je korak po korak radi detaljne analize njenog rada u određenim

trenutcima. Pored gumba nalazi se i klizač (eng. *Slider*) pomoću kojega je moguće odrediti širinu vidnog polja kamere (eng. *Field of View*).

Prozor pod brojem tri je prozor Hijerarhija (eng. *Hierarchy*). Ovaj prozor hijerarhijski prikazuje sve objekte koji se nalaze unutar trenutno odabrane scene. Unutar njega možemo kreirati, grupirati, pretraživati i filtrirati, odnosno upravljati objektima.

Četvrti prozor prikazuje sadržaj mape koja je trenutno odabrana. Ovaj prozor možemo povezati sa prozorom *Assets* (pod brojem pet) koji prikazuje sve mape unutar trenutnog Unity projekta. Uloga ovih prozora je organizacija i upravljanje svim resursima koji se koriste u projektu, uključujući objekte, slike, animacije, skripte i sve ostale datoteke potrebne za izradu igre ili aplikacije.

Prozor označen brojem šest je prozor *Inspector.* Ovim prozorom moguće je dodavati ili uklanjati komponente objekta, urediti teksture, animacije, zvukove i slično.

4.1.2. Unity Asset

Unity *Asset* je bilo koji resurs koji se koristi u projektu prilikom izrade igre ili aplikacije. To je osnovni i najvažniji objekt bez kojega je nemoguće kreirati bilo kakav Unity projekt. *Asset* može biti audiodatoteka, 3D model, animacija, tekstura, font ili bilo koji drugi oblik datoteke koja je kompatibilna s *Unity*em. Zbog toga je osnovan Unity Asset Store pomoću kojega je moguće direktno pristupiti *assetima* koji mogu poslužiti u izradi igre ili aplikacije. Više o samom Asset Storeu govori se u sljedećem poglavlju.

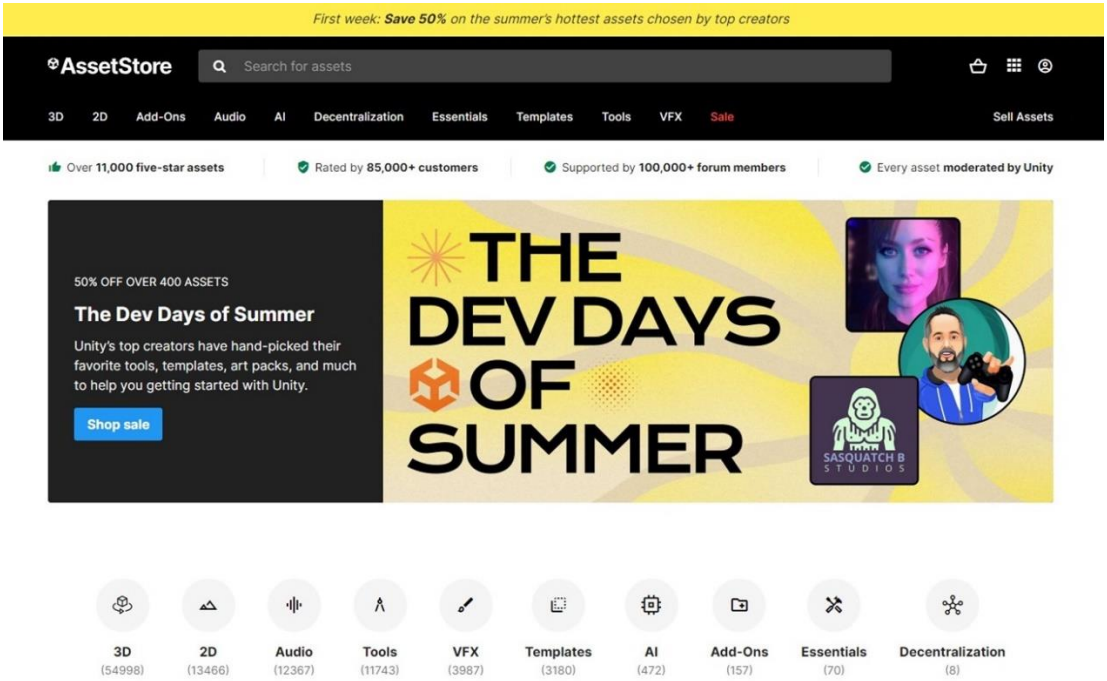
4.1.3. Unity Asset Store

Kao što i samo ime govori Unity Asset Store je virtualna trgovina gdje programeri i ostali korisnici mogu kupovati i prodavati razne *assete* za korištenje u projektima. Glavna karakteristika Asset Storea je raznovrsnost *asseta* kao što su teksture, 3D modeli, audio-datoteke, skripte i sl. Prednost korištenja Asset Storea je dostupnost kvalitetnih i besplatnih resursa koji uvelike mogu ubrzati razvoj projekta i tako smanjiti troškove. Ipak, u Asset Storeu postoje i resursi koji mogu biti loše dokumentirani ili neoptimizirani, odnosno mogu imati potencijalne probleme s licenciranjem i pravima korištenja.

Unity Asset Storeu možemo pristupiti izravno unutar samog Unity Enginea ili putem bilo kojeg podržanog web preglednika.

Neki od *asseta* koje je moguće pronaći u Storeu:

- C# skripte - omogućuju kontrolu nad igrom, reakcije na događaje i implementaciju logike
- 3D modeli i animacije - fizičke strukture igre, likovi, vozila, okolina te dodatak pokreta i života likovima
- audiozapisi - zvučne datoteke koje se koriste za dodavanje zvučnih efekata, dijaloga i glazbe u igri
- fontovi i UI elementi - omogućuju prikaz teksta i stvaranje korisničkog sučelja u igri
- Templates (Predlošci) - gotove skripte i alati namijenjeni prvenstveno početnicima.



Slika 8. Naslovna stranica Unity Asset Store-a

Izvor: Autor

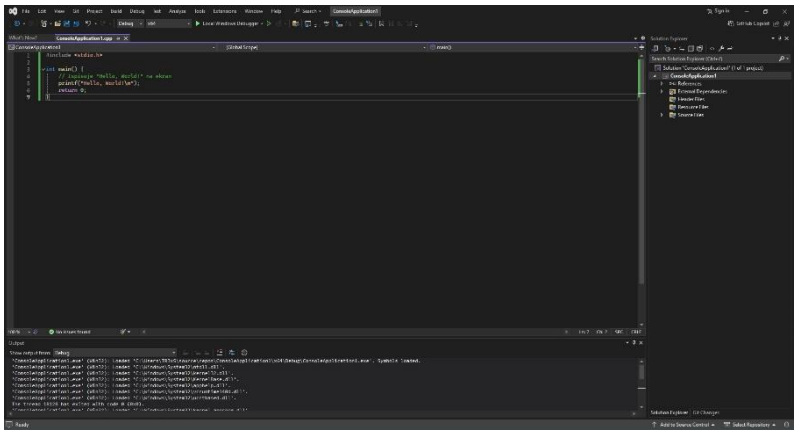
4.2. Microsoft Visual Studio

Microsoft Visual Studio integrirano je razvojno okruženje (eng. *Integrated Development Environment*) koje je razvio Microsoft. Prvi put lansiran 1997. godine, Visual Studio je prošao kroz mnoge verzije i poboljšanja, uključujući integraciju .NET Frameworka, uvođenje podrške za ASP.NET, integraciju s alatima kao što su Git, podršku za razvoj u oblaku, AI podršku kroz IntelliCode itd. Budući da službeno podržava čak 36 programskih jezika uključujući C, C#, C++, Javascript i ostale, Visual Studio je danas jedan od najpopularnijih alata za izradu web-aplikacija, web-stranica, računalnih igara i mobilnih aplikacija. Velika prednost Visual Studia njegova je potpuna integracija s Unity Editorom što rezultira učinkovitijim i bržim razvojem projekata, smanjenjem broja pogrešaka i poboljšanjem cjelokupne produktivnosti projekta.

Microsoft nudi tri različite verzije Visual Studia:

- Community – besplatna opcija koja nudi potpuno funkcionalno razvojno okruženje za studente, open-source programere i individualne programere
- Professional - opcija koja se temelji na pretplati za individualne programere ili male timove
- Enterprise - opcija koja se temelji na pretplati za male/velike tvrtke i korporativne organizacije.

Prilikom izrade ovog završnog rada, korištena je besplatna Community verzija koja sadrži sve potrebne funkcije i alate, bez dodatnih troškova.



Slika 9. Izgled Microsoft Visual Studio sučelja

Izvor: Autor

5. EKOSORTER

U ovom će se poglavlju opisati sama igra te predstaviti osnovne informacije i mehanike. Službeni naziv igre je *Ekosorter*. Kako je već navedeno, igra je 2D edukativne tematike s naglaskom na razvrstavanje otpada. Igra se sastoji od jedne vremenski ograničene razine (eng. *Level*) unutar koje se nasumično generira različita vrsta otpada. Igračev cilj je razvrstati otpad u pripadajući spremnik, odnosno sakupiti što više bodova unutar vremenskog okvira od tri minute. Igrač dobiva po jedan bod za svaku ispravno razvrstanu vrstu otpada. Sustav bodovanja može se povećati do maksimalno pet puta ovisno o količini uzastopno točnih odgovora. Primjerice, za kombinaciju od pet uzastopno točnih odgovora iznos bodova za odgovarajući otpad iznosi dva boda, za kombinaciju od 10 točnih odgovora iznos bodova za odgovarajući otpad iznosi tri boda itd. Ukoliko igrač odloži otpad u krivi koš, otpad „izađe“ izvan krajnjih točaka scene ili „padne“ na dno igraće scene, igraču se broj „života“ od ukupno tri umanjuje za jedan. Implementacija ovih mehanika donosi dašak izazova igračima, učenicima nižih razreda osnovne škole.

Nakon isteka vremenskog roka od tri minute ili gubitka „života“, igraču se prikazuje *Game Over* izbornik na kojemu može odabrati pokretanje *levela* ispočetka ili povratak na glavni meni. Igrač u bilo kojemu trenutku pritiskom na *Escape* tipku na tipkovnici može pauzirati igru i otvoriti izbornik *Pause.* Unutar ovog izbornika klikom na gumbe igrač može nastaviti igru, vratiti se na glavni izbornik ili potpuno izaći iz igre. Kao i svaka igra, tako i ova ima glavni izbornik iz kojega je moguće pokrenuti igru odnosno igraći *level*, ući u izbornik s postavkama ili potpuno izaći iz igre na radnu površinu.

Slika 10. Logotip igre

Izvor: Autor

6. IZRADA IGRE

Važno je pridržavati se nekoliko ključnih koraka u kreiranju uspješne igre:

1. izrada resursa (eng. *Assets*) koji će se koristiti u igri ili preuzimanje već gotovih resursa iz *Unity Asset Storea*
2. izrada i konfiguracija scena
3. animiranje i programiranje objekata i igrača (eng. *Player*)
4. implementiranje glazbe i zvučnih efekata
5. izrada grafičkog korisničkog sučelja (eng. *Graphical User Interface*).

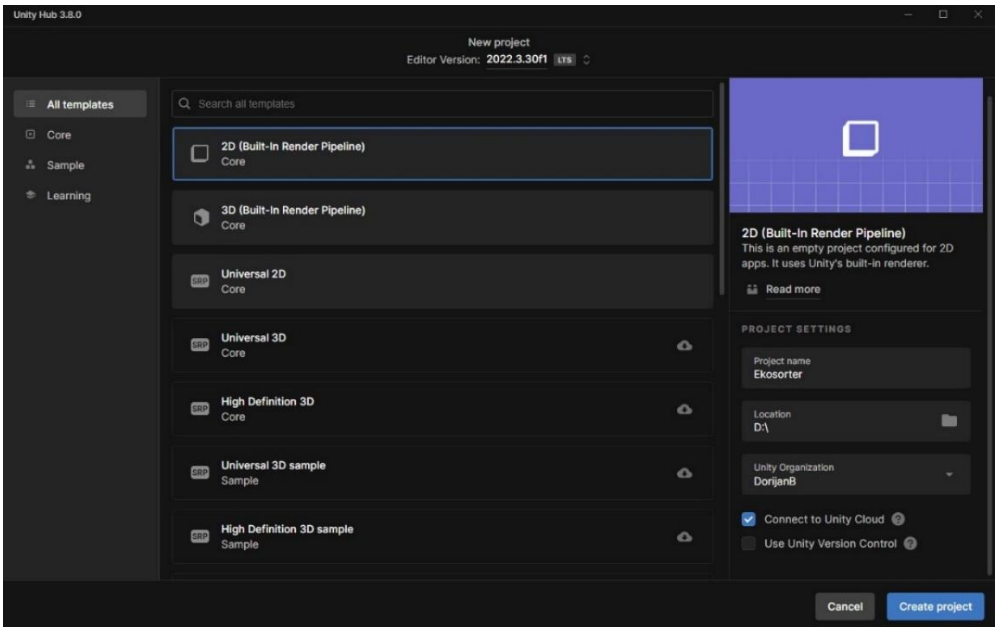
6.1. Izrada i prikupljanje resursa

Resursi (eng. *Assets*) su ključne sastavnice za kreiranje igre. Za potrebe ove igre većina je resursa preuzeta s Unity Asset Storea i ostalih besplatnih web-stranica za preuzimanje resursa. Pojedini resursi kao što su ikonice, glazba i zvučni efekti modificirani su kako bi se bolje uklopili u igru i njenu tematiku.

6.2. Izrada projekta u Unityu

Izrada bilo koje Unity igre ili aplikacije započinje kreiranjem novog projekta u Unity Hubu. Budući da je tema ovog završnog rada izrada dvodimenzionalne igre, odabran je 2D predložak. Nakon odabira predloška projekt je nazvan „Ekosorter“, što je i naziv same igre. Naposljetku je potrebno odabrati željenu lokaciju spremanja projekta i kliknuti na Create Project. Moguće je odabrati verziju editora u kojoj se projekt izrađuje te odabrati ime developera.

Prilikom kreiranja projekta moguće je odabrati Unity Version Control. Ta vrlo korisna opcija omogućuje pristup na daljinu do tri korisnika na istom projektu.



Slika 11. Kreiranje projekta

Izvor: Autor

Nakon kreiranja projekta, potrebno je uvesti resurse. Zbog lakše organizacije i veće preglednosti kreirane su posebne mape za objekte, glazbu, zvučne efekte i sl. unutar projektne ploče (eng. *Project Panel*).

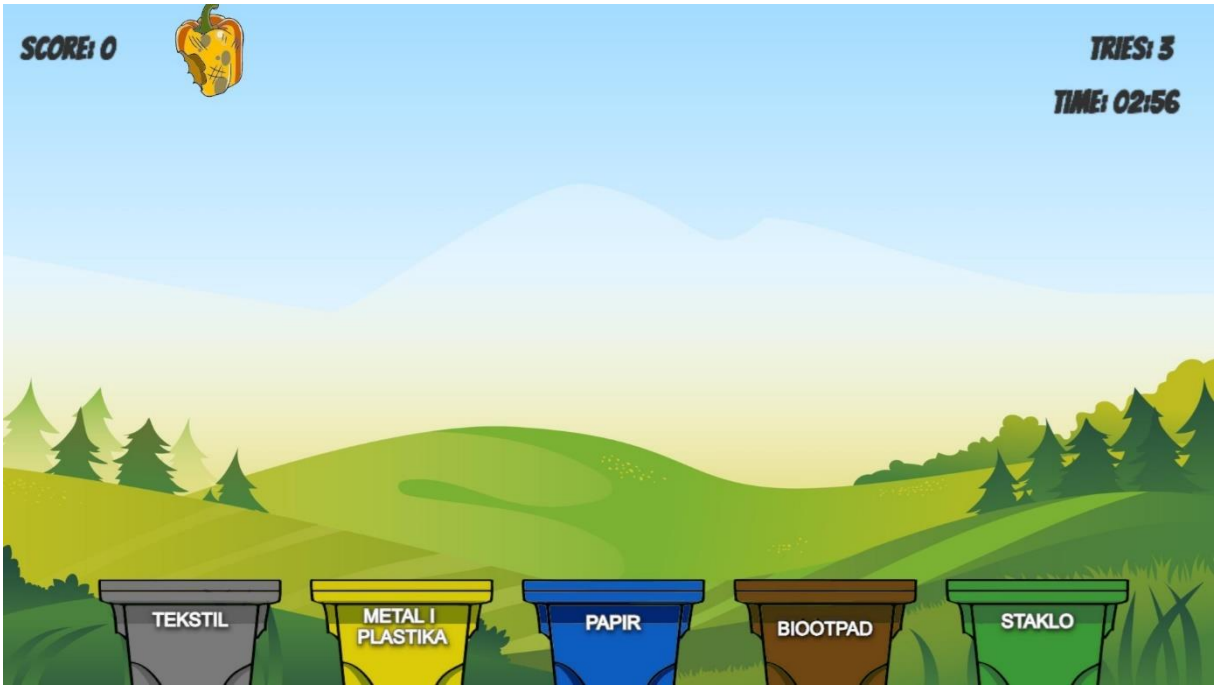
Slika 12. Dodavanje resursa

Izvor: Autor

6.3. Izrada i konfiguracija scena

U prethodnim koracima dodani su potrebni resursi za izradu 2D igre. Sljedeći korak je izrada scene. Kako se radi o jednostavnoj 2D igri s jednom vremenski ograničenom razinom, potrebno je izraditi nekoliko glavnih scena: Glavni meni, Igraća scena te scena Opcije. Osim ove tri scene, postoje još i Game Over i Pauza scene koje se aktiviraju igračevom interakcijom u određenom trenutku.

Proces izrade scene je jednostavan. Unutar Unity Editora potrebno je kliknuti na gumb „File“ te na gumb „New Scene“. Brži način kreiranja scene je pritiskom tipki Ctrl+N na tipkovnici.



Slika 13. Izgled igraće scene

Izvor: Autor

6.4. Razvrstavanje otpada

Glavni je smisao igre razvrstati otpad u za to predviđeni spremnik. Kako bi to uopće bilo moguće, na sceni je potrebno kreirati spremnike u koje će igrač razvrstavati otpad. Tako u igri postoji ukupno pet različitih spremnika: tekstil, metal i plastika, papir, biootpad te staklo. Otpad se nasumično generira unutar jedne od pet nevidljivih točaka raspoređenih pri vrhu igraće scene, u tzv. „Spawn Pointovima“. Ako igrač ispravno odloži otpad u odgovarajući koš, sustav bodovanja mu dodjeljuje jedan bod te se otpad iznova nasumično generira u jednom od pet „Spawn Pointova“. Ukoliko igrač odloži otpad u krivi spremnik ili ono „izađe“ izvan granica igraće scene, igrač gubi jedan od tri života te se otpad nasumično generira iznova. Otpad se generira sve dok igrač ne izgubi sva tri života ili dok brojač vremena ne dođe do nule.

Važno je napomenuti kako je u igri implementiran efekt bljeskanja ekrana crvenom bojom koji se aktivira prilikom netočnog odgovora ili kada otpad „izađe“ izvan granica igraće scene. Više o efektu bljeska, sustavu bodovanja, sustavu života i brojaču vremena bit će navedeno u zasebnim poglavljima.

```
public class Bin : MonoBehaviour
{
    public string type; // Type of trash this bin accepts
    void OnTriggerEnter2D(Collider2D other)
    {
        TrashItem item = other.GetComponent<TrashItem>();
        if (item != null) CheckItem(item);
    }

    void CheckItem(TrashItem item)
    {
        GameManager gameManager = FindObjectOfType<GameManager>();
        if (item.type == type)
        {
            gameManager.AddScore(1);
            Destroy(item.gameObject); // Destroy the trash item
```

```
            if (!gameManager.isGameOver)
gameManager.SpawnTrashItem(); // Spawn a new trash item
        }
        else
        {
            gameManager.LoseTry();
            Destroy(item.gameObject); // Destroy the trash item
            if (!gameManager.isGameOver)
gameManager.SpawnTrashItem(); // Spawn a new trash item
        }
    }
}
```

Kod 1. Spremnik za otpad

Izvor: Autor

6.5. Game manager

Skripta „Game Manager“ predstavlja centralni upravljački modul igre koji koordinira osnovne mehanike igre poput praćenja ostvarenih bodova, upravljanja preostalim pokušajima, upravljanja brojačem vremena te upravljanja stanjima igre, primjerice kada istekne vremensko ograničenje od tri minute ili kada igrač iskoristi sva tri pokušaja. Između ostaloga, skripta upravlja i s komponentama korisničkog sučelja te zvučnim efektima.

```
public class GameManager : MonoBehaviour {
    public int score = 0;
    public int tries = 3;
    public float gameDuration = 180f; // 3 minutes in seconds
    private float remainingTime;
    public UIManager uiManager;
    public GameObject[] trashPrefabs;
```

```
    public Transform[] spawnPoints; // Array of spawn points

    private GameObject currentTrashItem;

    public bool isGameOver = false;

    private AudioManager audioManager;

    private ScreenFlash screenFlash; // Reference to ScreenFlash

    private int consecutiveCorrect = 0; // Track consecutive correct items

    private int comboMultiplier = 1; // Track current combo multiplier

    public float initialGravityScale = 0.5f; // Initial gravity scale for natural falling


    void Start() {

        Time.timeScale = 1f; // Reset time scale at the start of the game

        audioManager = FindObjectOfType<AudioManager>();

        screenFlash = FindObjectOfType<ScreenFlash>(); // Find the ScreenFlash component

        StartGame();

    }

    void StartGame() {

        score = 0;

        tries = 3;

        remainingTime = gameDuration;

        isGameOver = false;

        consecutiveCorrect = 0;

        comboMultiplier = 1;

        uiManager.UpdateScore(score);
```

```
    uiManager.UpdateTries(tries);
    uiManager.UpdateTimer(remainingTime);
    audioManager.ResumeGameMusic();
    SpawnTrashItem();
}

void Update() {
    if (!isGameOver) {
        remainingTime -= Time.deltaTime;
        uiManager.UpdateTimer(remainingTime);
        if (remainingTime <= 0) EndGame();
    }
}

public void AddScore(int basePoints) {
    if (isGameOver) return;
    consecutiveCorrect++;
    UpdateComboMultiplier();
    score += basePoints * comboMultiplier;
    uiManager.UpdateScore(score);
    audioManager.PlayCorrectSound();
    SpawnTrashItem(); // Spawn new item after scoring
}

public void LoseTry() {
    if (isGameOver) return;
```

```
        tries--;
        uiManager.UpdateTries(tries);
        CheckGameOver();
        consecutiveCorrect = 0;
        comboMultiplier = 1;
        audioManager.PlayIncorrectSound();
        if (screenFlash != null) screenFlash.Flash(); // Trigger
the flash effect
        SpawnTrashItem(); // Spawn new item after losing a try
    }

    void CheckGameOver() {
        if (tries <= 0) EndGame();
    }

    void EndGame() {
        Debug.Log("Game Over");
        isGameOver = true;
        if (currentTrashItem != null) {
            Destroy(currentTrashItem);
            currentTrashItem = null;
        }
        uiManager.ShowGameOver();
    }

    void UpdateComboMultiplier() {
        if (consecutiveCorrect >= 20) comboMultiplier = 5;
```

```
        else if (consecutiveCorrect >= 15) comboMultiplier = 4;
        else if (consecutiveCorrect >= 10) comboMultiplier = 3;
        else if (consecutiveCorrect >= 5) comboMultiplier = 2;
        else comboMultiplier = 1;
    }
    public void SpawnTrashItem() {
        if (isGameOver) return;
        if (currentTrashItem != null) {
            Destroy(currentTrashItem);
            currentTrashItem = null;
        }
        Debug.Log("Spawning new trash item...");
        if (trashPrefabs.Length == 0) {
            Debug.LogError("No trash prefabs assigned!");
            return;
        }
        if (spawnPoints.Length == 0) {
            Debug.LogError("No spawn points assigned!");
            return;
        }
        int randomIndex = Random.Range(0, trashPrefabs.Length);
        int randomSpawnPointIndex = Random.Range(0,
spawnPoints.Length);
        Debug.Log("Selected spawn point index: " +
randomSpawnPointIndex);
        Transform selectedSpawnPoint =
spawnPoints[randomSpawnPointIndex];
```

```
        currentTrashItem = Instantiate(trashPrefabs[randomIndex],
selectedSpawnPoint.position, Quaternion.identity);

        currentTrashItem.tag = "TrashItem"; // Ensure the new item
has the correct tag

        Rigidbody2D                         rb                      =
currentTrashItem.GetComponent<Rigidbody2D>();

        if (rb != null) {

            rb.gravityScale = initialGravityScale;

            Debug.Log("New trash item spawned with initial gravity
scale: " + rb.gravityScale);

        } else {

            Debug.LogError("Rigidbody2D component not found on new
trash item.");

        }

    }

    public void RestartGame() {

        Time.timeScale = 1f; // Ensure time scale is reset

      SceneManager.LoadScene(SceneManager.GetActiveScene().name);

    }

    public void GoToMainMenu() {

        Time.timeScale = 1f; // Ensure time scale is reset

        SceneManager.LoadScene("MainMenu");

    }

}
```

Kod 2. Game Manager

Izvor: Autor

6.6. Collideri

Kako bi razvrstavanje otpada imalo smisla, potrebno je definirati fizičke granice (eng. *Collidere*) na spremnicima za otpad, otpadu te krajnjim točkama igraće scene. Drugim riječima, *collideri* su ključni za pravilno funkcioniranje igre jer omogućuju preciznu detekciju interakcija između objekata unutar igre. Postavljanjem granica na rubove igraće scene sprečava se izlazak otpada izvan igraćeg područja čime se održava kontinuitet same igre. Također, *collideri* na spremnicima za otpad prepoznaju ispravno razvrstan otpad, što omogućuje pravilno bodovanje i sprečava slučajne pogreške.

```
public class WallTrigger : MonoBehaviour
{
    void OnTriggerEnter2D(Collider2D other)
    {
        TrashItem item = other.GetComponent<TrashItem>();
        if (item != null)
        {
            HandleWallTrigger(item);
        }
    }
    void HandleWallTrigger(TrashItem item)
    {
        GameManager gameManager = FindObjectOfType<GameManager>();
        gameManager.LoseTry();
        Destroy(item.gameObject); // Destroy the trash item
        if (!gameManager.isGameOver) // Check if the game is not
over
        {
            gameManager.SpawnTrashItem(); // Spawn a new trash item
        }
}
}
```

Kod 3. Collider krajnjih točaka igraće scene

Izvor: Autor

6.7. Kontroliranje igre

Nakon što se otpad generira u jednom od pet *spawn pointova*, igrač ima kontrolu nad otpadom dok god je ono unutar granica igraće scene. Drugim riječima, igrač može kontrolirati vertikalni položaj otpada pritiskom tipki strelica na tipkovnici. Kako otpad horizontalno „pada“ uvijek istom brzinom, igrač to može ubrzati pritiskom tipke Space na tipkovnici.

```
public class PlayerController : MonoBehaviour
{
    private GameObject currentTrashItem;
    private Rigidbody2D currentRigidbody;
    public float moveSpeed = 250.0f; // Speed multiplier for horizontal movement
    public float fallSpeedMultiplier = 1.5f; // Multiplier for accelerated fall
    private float originalGravityScale;

    void Update()
    {
        GameManager gameManager = FindObjectOfType<GameManager>();
        if (gameManager == null || gameManager.isGameOver) return;

        if (currentTrashItem != null)
        {
            MoveItem();
            if (Input.GetKeyDown(KeyCode.Space))
            {
                DropItem();
            }
        }
        else
        {
            FindCurrentTrashItem();
        }
```

```
    }

    void MoveItem()
    {
        float moveHorizontal = Input.GetAxis("Horizontal");

        // Adjust the horizontal movement speed
        Vector2 movement = new Vector2(moveHorizontal * moveSpeed,
currentRigidbody.velocity.y);

        // Accelerate fall when down arrow key is pressed
        if (Input.GetKey(KeyCode.DownArrow))
        {
            currentRigidbody.gravityScale = originalGravityScale *
fallSpeedMultiplier;
        }
        else
        {
            currentRigidbody.gravityScale = originalGravityScale;
        }

        currentRigidbody.velocity = new Vector2(movement.x,
currentRigidbody.velocity.y);
    }

    void DropItem()
    {
        currentTrashItem = null;
        currentRigidbody = null;
    }
    void FindCurrentTrashItem()
    {
        currentTrashItem = GameObject.FindWithTag("TrashItem");
        if (currentTrashItem != null)
```

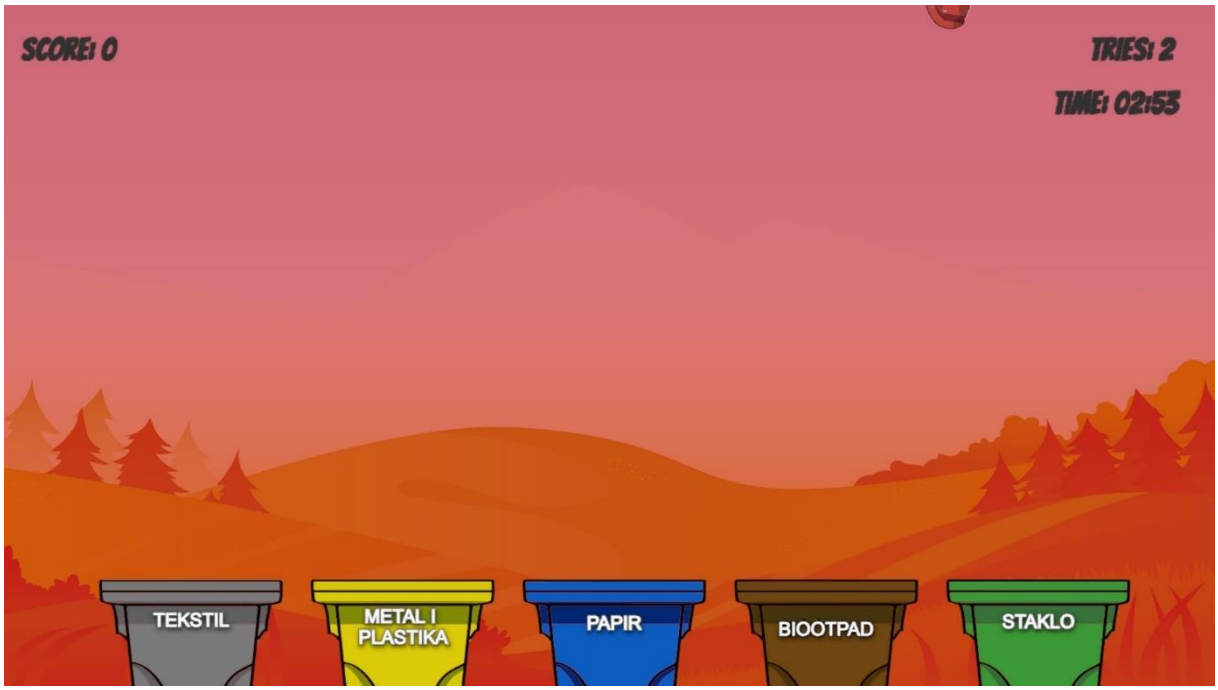
```
        {
            currentRigidbody =
currentTrashItem.GetComponent<Rigidbody2D>();
            if (currentRigidbody != null)
            {
                originalGravityScale =
currentRigidbody.gravityScale;
            }
            else
            {
                Debug.LogError("No Rigidbody2D found on the current
trash item!");
            }
        }
    }
}
```

Kod 4. Kontroliranje igre

Izvor: Autor

6.8. Efekt bljeskanja ekrana

Bljeskanje ekrana crvenom bojom prilikom krivog sortiranja otpada koristi se kao vizualni efekt koji igraču pravovremeno signalizira grešku kao u akcijskim igrama gdje bljesak označava primljenu štetu. Efekt pruža jasnu povratnu informaciju da je otpad razvrstan pogrešno čime se igrača podučava pravilima recikliranja. Implementacija ove metode poboljšava učenje putem pogrešaka i čini igru dinamičnijom te motivira igrača da usvoji ispravne navike razvrstavanja otpada.



Slika 14. Efekt bljeskanja ekrana

Izvor: Autor

```
public class ScreenFlash : MonoBehaviour
{
    public Image flashImage;
    public float flashDuration = 0.5f;
    public Color flashColor = new Color(1f, 0f, 0f, 0.5f);

    private void Start()
    {
        if (flashImage != null)
        {
            flashImage.color = new Color(flashColor.r, flashColor.g,
flashColor.b, 0); // Start fully transparent
        }
    }

    public void Flash()
    {
```

```
        if (flashImage != null)
        {
            StartCoroutine(FlashRoutine());
        }
    }

    private IEnumerator FlashRoutine()
    {
        float elapsedTime = 0f;
        while (elapsedTime < flashDuration)
        {
            elapsedTime += Time.deltaTime;
            float alpha = Mathf.PingPong(elapsedTime, flashDuration
/ 2) / (flashDuration / 2);
            flashImage.color = new Color(flashColor.r, flashColor.g,
flashColor.b, alpha);
            yield return null;
        }
        flashImage.color = new Color(flashColor.r, flashColor.g,
flashColor.b, 0); // Ensure fully transparent at the end
    }
}
```

Kod 5. Efekt bljeskanja ekrana

Izvor: Autor

6.9. Brojač vremena, sustav bodovanja i sustav života

Brojač vremena ima ključnu ulogu u postavljanju ritma i izazova unutar igre. Vremensko ograničenje od tri minute ograničava igrača i stvara mu pritisak kako bi postigao što bolji rezultat prije isteka vremenskog ograničenja. U nastavku je prikazan izdvojeni dio koda iz skripte Game Manager koji se odnosi na brojač vremena.

```
     public float gameDuration = 180f; // 3 minutes in seconds
private float remainingTime;

void StartGame()
{
    remainingTime = gameDuration;
    uiManager.UpdateTimer(remainingTime);
}

void Update()
{
    if (!isGameOver)
    {
        remainingTime -= Time.deltaTime;
        uiManager.UpdateTimer(remainingTime);

        if (remainingTime <= 0)
        {
            EndGame();
        }
    }
}
```

Kod 6. Brojač vremena

Izvor: Autor

Sustav bodovanja u igri ključan je za praćenje igračeva uspjeha i za motivaciju za daljnje igranje. Kada igrač pravilno razvrsta otpad, sustav bodovanja dodjeljuje mu jedan bod. Važno je napomenuti da sustav bodovanja sadrži i takozvani „*Combo Multiplier*“, sustav koji nagrađuje igrača za dosljedne uspjehe. Početni „*Combo Multiplier*“ iznosi jedan, ali se povećava kada igrač napravi niz uzastopnih ispravnih radnji. Na primjer: ako igrač napravi pet uzastopnih ispravnih odgovora, „*Combo Multiplier*“ raste na 2, ako ih napravi 10, raste na 3 i tako dalje do maksimuma od 5. Kod u nastavku prikazuje izdvojeni dio koda iz skripte Game Manager koji se odnosi na sustav bodovanja.

```
public class GameManager : MonoBehaviour {
    public int score = 0;
    private int consecutiveCorrect = 0; // Track consecutive correct
items
    private int comboMultiplier = 1; // Track current combo
multiplier

    void StartGame() {
        score = 0;
        consecutiveCorrect = 0;
        comboMultiplier = 1;
        uiManager.UpdateScore(score);
    }

    public void AddScore(int basePoints) {
        if (isGameOver) return;
        consecutiveCorrect++;
        UpdateComboMultiplier();
        score += basePoints * comboMultiplier;
        uiManager.UpdateScore(score);
        audioManager.PlayCorrectSound();
        SpawnTrashItem(); // Spawn new item after scoring
    }

    public void LoseTry() {
```

```
        if (isGameOver) return;
        tries--;
        uiManager.UpdateTries(tries);
        CheckGameOver();
        consecutiveCorrect = 0;
        comboMultiplier = 1;
        audioManager.PlayIncorrectSound();
        if (screenFlash != null) screenFlash.Flash(); // Trigger the
flash effect
        SpawnTrashItem(); // Spawn new item after losing a try
    }

    void UpdateComboMultiplier() {
        if (consecutiveCorrect >= 20) comboMultiplier = 5;
        else if (consecutiveCorrect >= 15) comboMultiplier = 4;
        else if (consecutiveCorrect >= 10) comboMultiplier = 3;
        else if (consecutiveCorrect >= 5) comboMultiplier = 2;
        else comboMultiplier = 1;
    }
}
```

Kod 7. Sustav bodovanja

Izvor: Autor

Sustav pokušaja u igri kontrolira broj pokušaja koje igrač ima prilikom ispravno sortiranog otpada. Kako je već navedeno ranije, igrač ima tri pokušaja, a svakom pogrešnom reakcijom broj pokušaja se smanjuje za jedan. Ako broj pokušaja padne na nulu, igra se automatski prekida.

Između ostaloga, kada igrač izgubi pokušaj, sustav resetira praćenje uzastopnih ispravnih odgovora i poništava trenutni „*Combo Multiplier*“. Ovim se sustavom dobija dinamičnost i izazovnost igre jer igrač mora biti precizan u svojoj reakciji kako bi izbjegao gubitak pokušaja i na kraju završio igru. Kod u nastavku prikazuje izdvojeni dio koda iz skripte Game Manager koji se odnosi na sustav pokušaja u igri.

```
public int tries = 3;


public void LoseTry() {
    if (isGameOver) return;
    tries--;
    uiManager.UpdateTries(tries);
    CheckGameOver();
    consecutiveCorrect = 0;
    comboMultiplier = 1;
    audioManager.PlayIncorrectSound();
    if (screenFlash != null) screenFlash.Flash(); // Trigger the
flash effect
    SpawnTrashItem(); // Spawn new item after losing a try
}


void CheckGameOver() {
    if (tries <= 0) EndGame();
}
```

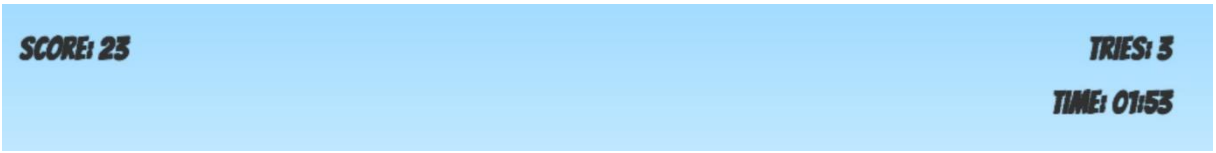
Kod 8. Sustav pokušaja

Izvor: Autor

7. HEADS UP DISPLAY

HUD (eng. *Heads Up Display*) je korisničko sučelje koje u svakom trenutku prikazuje ključne informacije igraču tijekom igranja: najčešće je to prikaz mini-mape, razine zdravlja, status misije ili ciljeva i sl. Time se povećava igračeva brzina reagiranja i njegova efikasnost jer su sve bitne informacije odmah vidljive. Primjerice, u igrama brzog tempa poput „Call of Dutya“ trenutna informacija o količini metaka ili razini zdravlja može predstavljati razliku između pobjede i poraza. Sučelje je obično smješteno u uglovima ili na rubovima ekrana. Pravilno implementiran HUD sadrži dovoljno velike i vidljive, djelomično transparentne elemente kako igraču ne bi odvlačili pozornost i narušavali vizualni dojam igre. [9]

HUD unutar ove igre prikazuje broj ostvarenih bodova, odnosno rezultat u gornjem lijevom uglu. Prikaz broja preostalih pokušaja i brojač vremena nalaze se u gornjem desnom uglu ekrana. Važno je napomenuti da se pod HUD ne ubrajaju izbornici u igri kao što su glavni meni, izbornik pauza ili izbornik opcije.



Slika 15. Izgled HUD-a

Izvor: Autor

8. IZBORNICI

Prva stvar koju igrač vidi u bilo kojoj igri je glavni izbornik. To je „lice“ preko kojega igrač već u prvim sekundama doživljava atmosferu igre. Zlatno pravilo kojega se treba pridržavati prilikom izrade izbornika glasi „manje je više“. Drugim riječima, brojne opcije u izborniku nisu nužno povezane s „dobrim dizajnom igre“. Dobar je izbornik jednostavan za korištenje, vizualno privlačan i jedinstveno povezan s igrom.

Ova igra sadrži ukupno četiri izbornika: glavni izbornik (eng. *Main Menu*), postavke (eng. *Settings*), pauza (eng. *Pause*) i igra je gotova (eng. *Game Over*). Prilikom izrade glavnog izbornika velika je pažnja posvećena jednostavnosti. Na izborniku se nalazi logotip igre te svega tri gumba od kojih prvi pokreće igru, drugi otvara zasebni izbornik s postavkama i treći koji služi za izlaz iz igre. Kako bi ovi gumbi mogli funkcionirati, potrebno je za svaki od njih napisati skriptu.

Slika 16. Glavni izbornik

Izvor: Autor

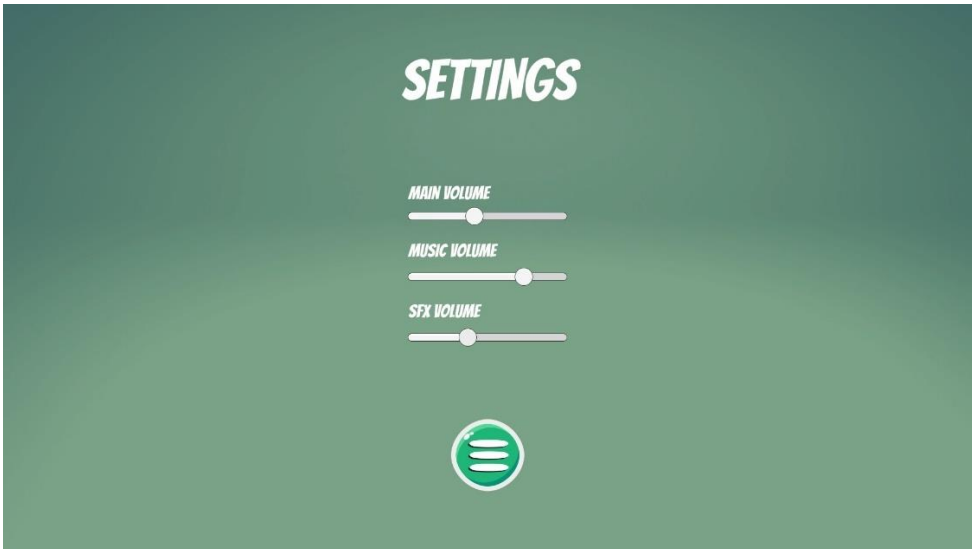
```
public class MainMenuManager : MonoBehaviour
{
    public void StartGame()
    {
        SceneManager.LoadScene("SampleScene");
    }
```

```
    public void OpenSettings()
    {
        SceneManager.LoadScene("SettingsScene");
    }

    public void QuitGame()
    {
        Application.Quit();
        Debug.Log("Game is exiting");
    }
}
```

Kod 9. Main Menu Manager

Izvor: Autor

Klikom na drugi po redu gumb na glavnom izborniku otvara se izbornik opcije (eng. *Settings*). Unutar opcija nalaze se tri klizača (eng. *Slider*) pomoću kojih je moguće podesiti glasnoću glavnog zvuka, glazbe ili zvučnih efekata. Klikom na gumb ispod vraćamo se na glavni izbornik.



Slika 17. Izbornik opcija

Izvor: Autor

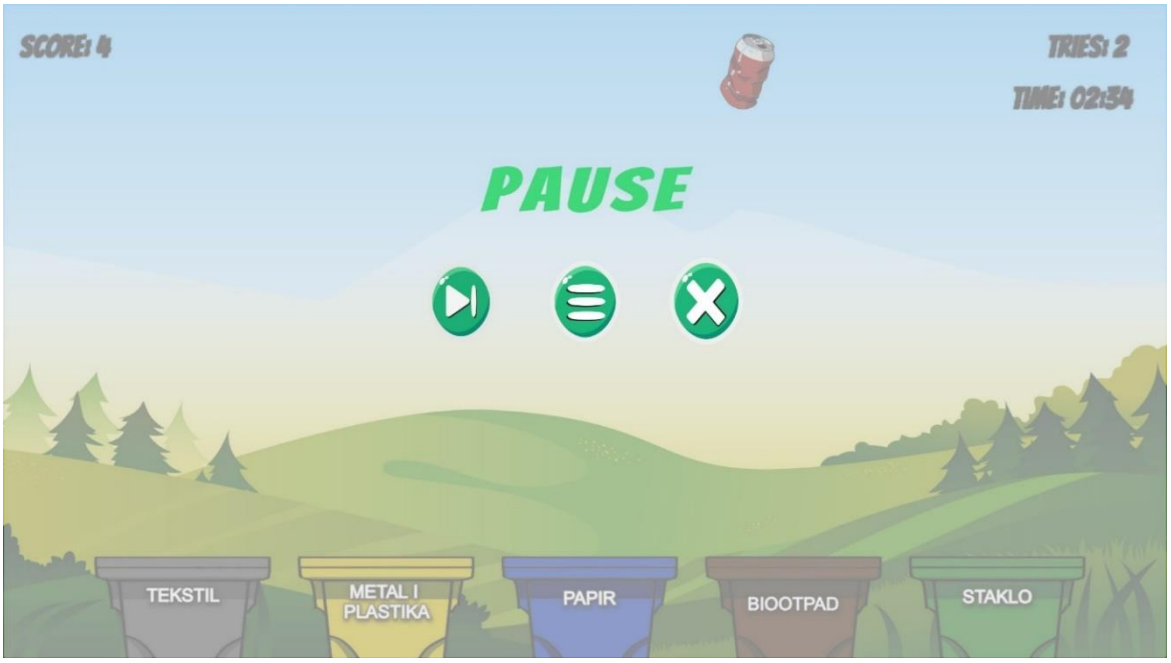
```
public class SettingsManager : MonoBehaviour {
    public Slider mainVolumeSlider;
    public Slider musicVolumeSlider;
    public Slider sfxVolumeSlider;
    private AudioManager audioManager;
    void Start() {
        audioManager = FindObjectOfType<AudioManager>();
        // Load saved volume levels
        mainVolumeSlider.value = PlayerPrefs.GetFloat("MainVolume",
1.0f);
        musicVolumeSlider.value =
PlayerPrefs.GetFloat("MusicVolume", 1.0f);
        sfxVolumeSlider.value = PlayerPrefs.GetFloat("SFXVolume",
1.0f);
        // Add listeners to update audio levels
        mainVolumeSlider.onValueChanged.AddListener(SetMainVolume);

musicVolumeSlider.onValueChanged.AddListener(SetMusicVolume);
        sfxVolumeSlider.onValueChanged.AddListener(SetSFXVolume);
    }
    void SetMainVolume(float volume) {
        AudioListener.volume = volume;
        PlayerPrefs.SetFloat("MainVolume", volume);
    }
    void SetMusicVolume(float volume) {
        if (audioManager != null) {
            audioManager.SetMusicVolume(volume);
        }
        PlayerPrefs.SetFloat("MusicVolume", volume);
    }
    void SetSFXVolume(float volume) {
        if (audioManager != null) {
            audioManager.SetSFXVolume(volume);
        }
```

```
        PlayerPrefs.SetFloat("SFXVolume", volume);
    }
    public void BackToMainMenu() {
        // Save settings before going back to the main menu
        PlayerPrefs.Save();
        // Load the main menu scene
UnityEngine.SceneManagement.SceneManager.LoadScene("MainMenu");
    }
}
```

Kod 10. Settings Manager

Izvor: Autor

Igrač može u svakom trenutku pauzirati igru pritiskom na tipku Escape na tipkovnici prilikom čega se otvara izbornik koji nudi mogućnost nastavka igranja, povratka na glavni izbornik ili izlaza iz igre na radnu površinu. Otvaranjem izbornika pauze brojač vremena se u potpunosti zaustavlja te se na trenutnoj sceni stvara efekt zamagljenosti.



Slika 18. Izbornik pauza

Izvor: Autor

```
public class PauseMenu : MonoBehaviour {
    public static bool GameIsPaused = false;
    public GameObject pauseMenuUI;
    private GameManager gameManager;

    void Start() {
        SceneManager.sceneLoaded += OnSceneLoaded;
        gameManager = FindObjectOfType<GameManager>();
        GameIsPaused = false; // Ensure the game is not paused at
the start
    }

    void OnDestroy() {
        SceneManager.sceneLoaded -= OnSceneLoaded;
    }

    void OnSceneLoaded(Scene scene, LoadSceneMode mode) {
        if (scene.name == "SampleScene") {
            Resume();
            GameIsPaused = false; // Ensure the game is not paused
when the scene loads
        }
    }

    void Update() {
        if (Input.GetKeyDown(KeyCode.Escape)) {
            if (gameManager != null && gameManager.isGameOver) {
                LoadMenu();
            } else {
                if (GameIsPaused) {
                    Resume();
                } else {
```

```
                    Pause();
                }
            }
        }
    }

    public void Resume() {
        pauseMenuUI.SetActive(false);
        Time.timeScale = 1f;
        GameIsPaused = false;
    }

    void Pause() {
        pauseMenuUI.SetActive(true);
        Time.timeScale = 0f;
        GameIsPaused = true;
    }

    public void LoadMenu() {
        Time.timeScale = 1f;
        SceneManager.LoadScene("MainMenu");
    }
    public void QuitGame() {
        Debug.Log("Quitting game...");
        Application.Quit();
    }
}
```

Kod 11. Pause Manager

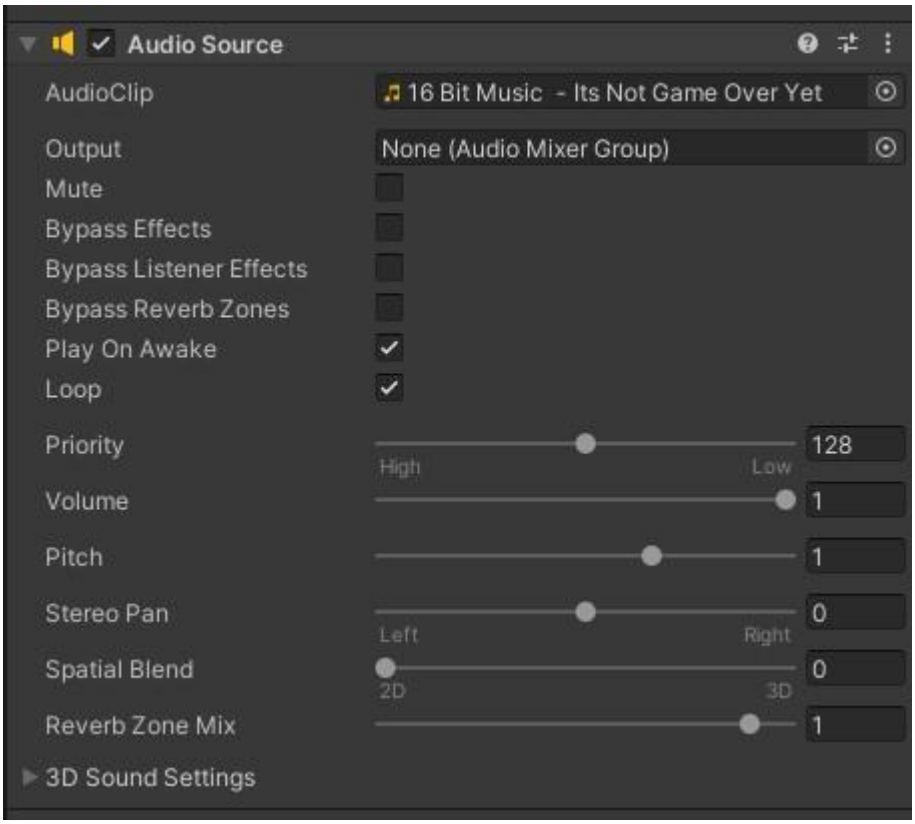
Izvor: Autor

9. ZVUČNI EFEKTI I POZADINSKA GLAZBA

Svaki oblik medija koristi zvuk kako bi poboljšao narativno iskustvo. Upravo se zbog toga glazba i zvučni efekti u videoigrama često uspoređuju s onima koje možemo čuti u filmovima ili na televiziji. Koliko god zvučalo nerealistično u usporedbi s filmskom industrijom gdje je glazba unaprijed određena i prati fiksni slijed događaja, glazba i zvučni efekti u videoigrama ipak idu korak dalje jer moraju besprijekorno reagirati na promjene u igri. Zvučni efekti također imaju drugačiju ulogu u videoigrama u usporedbi s filmskom industrijom. U videoigrama zvučni efekti služe kao ključni alat povratnih informacija obavještavajući igrača o njegovim radnjama i događajima u igri, dok u filmovima pretežno služe kako bi pojačali dojam realističnosti i atmosfere pojedine scene. Primjerice, u igrama zvuk koraka neprijatelja može upozoriti igrača na neposrednu opasnost i omogućiti mu pravovremenu reakciju. U filmovima, takav zvučni efekt najčešće služi za stvaranje napetosti i pojačavanje dramskog efekta, ali nema interaktivnu komponentu jer gledatelj ne može utjecati na ishod situacije. [7]

S obzirom da je izrada vlastitih zvučnih efekata i pozadinske glazbe u igrama danas veoma složen proces za koji su zasluženi specijalizirani stručnjaci, u ovom projektu koriste se zvučni efekti i pozadinska glazba preuzeti s besplatnih stranica s resursima za nekomercijalnu upotrebu.

Da bi reprodukcija glazbe i zvučnih efekata uopće bila moguća, Unity sadrži komponentu zvanu Audio Source unutar koje se nalazi polje Audio Clip u koje je potrebno dodati zvučne datoteke. Audio Source komponenta omogućuje mnogo opcija i mogućnosti podešavanja audiodatoteka. [8]



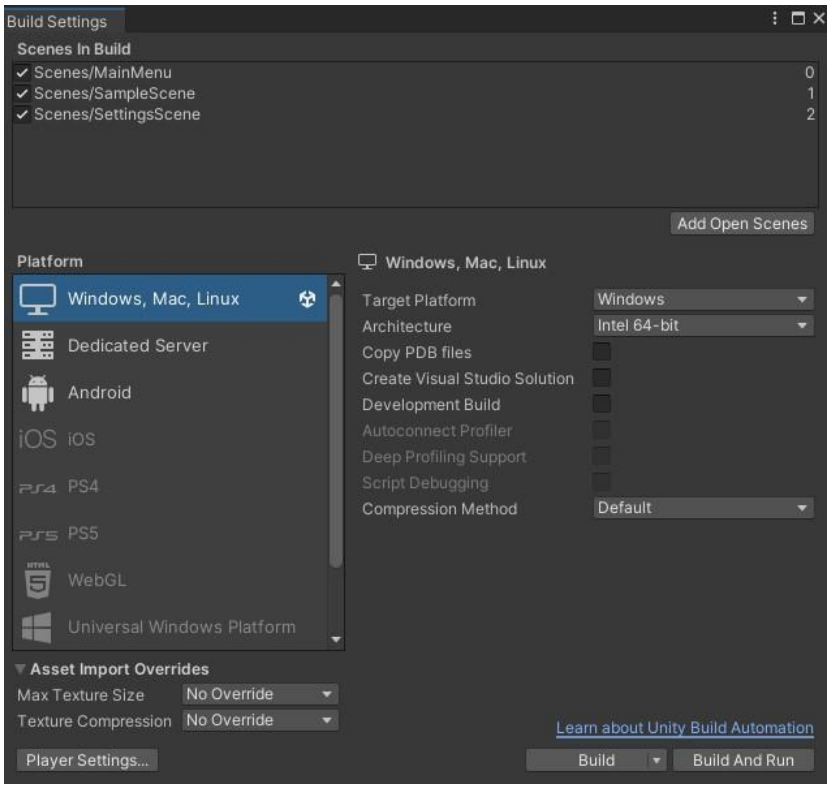
Slika 19. Audio Source

Izvor: Autor

10. DISTRIBUCIJA

Posljednji korak prilikom izrade bilo kakvog projekta unutar Unity editora je njegova distribucija. Distribucija u Unity editoru odnosi se na proces pripreme i objavljivanja igre ili aplikacije kako bi ona bila dostupna krajnjim korisnicima. Unity editor već u sebi ima opciju distribucije koja omogućuje pretvaranje Unity projekta u već gotov proizvod za čije pokretanje nisu potrebni programi treće strane. Drugim riječima, distribucija kreira jednostavnu izvršnu programsku datoteku koja se pokreće dvostrukim klikom na ikonicu.

Proces Unity distribucije započinje konfiguracijom *Build Settings*a gdje se odabire ciljna platforma na kojoj će se izvršavati programska datoteka npr. Windows, Linux, PS4, Android itd. te se postavljaju opcije poput rezolucije i kvalitete grafike. Potrebno je još odabrati nekoliko osnovnih podataka poput imena igre i autora, broja verzije aplikacije kao i njezinu ikonicu. Nakon što su postavljene sve željene postavke, potrebno je pritisnuti tipku *Build* koja stvara izvršnu programsku datoteku igre.



Slika 20. Build postavke

Izvor: Autor

11. ZAKLJUČAK

Pitamo li starije naraštaje, većina njih bi nam rekla da su videoigre gubitak vremena. Podatak da je industrija videoigara od svojih samih početaka pa do danas narasla na preko 184 milijarde dolara pokazuje upravo suprotno. Svakodnevno možemo vidjeti utjecaj videoigara, od adaptacija videoigara u filmove pa do odjeće i obuće s likovima iz igara.

Tijekom godina razvoj računalnih igara postao je vrlo kompleksan i grafički napredan proces u kojemu sudjeluje čitav tim developera, a sam razvoj često zna trajati i do nekoliko godina prije nego što je igra u potpunosti spremna za izlazak. Zahvaljujući brojnim alatima i besplatnim resursima, danas se gotovo svatko može iskušati u procesu izrade videoigara.

Gledano u cjelini, ova igra predstavlja spoj edukacije i zabave, pružajući igračima praktično znanje kroz interaktivno iskustvo. Njezina jednostavna struktura omogućuje laku adaptaciju i potencijalno unaprjeđenje u budućnosti dodavanjem novih vrsta otpada ili složenijih zadataka. Krajnji rezultat je igra koja podučava igrače pravilnom razvrstavanju otpada i potiče ih na kritičko razmišljanje o važnosti ekološke svijesti i održivog razvoja u svakodnevnom životu.

12. POPIS LITERATURE

[1] Graetz, J.M. (Kolovoz 1981) The origin of Spacewar! Creative Computing. Vol. 7 (22.6.2024.)

[2] Billboarding

https://nintendo.fandom.com/wiki/Billboarding (22.6.2024.)

[3] 13 (Secretly) Educational Video Games That Kids Will Actually Like

https://www.cnet.com/tech/gaming/13-secretly-educational-video-games-that-kids-will-actually-like/ (23.6.2024.)

[4] Educational Games: Creating the Learning Enviorment of the Future

https://blog.searchmyexpert.com/game-dev-education/ (23.6.2024.)

[5] Kudeljnjak , I., & Lozić, S. (2012). Unity game engine (26.7.2024.)

https://rg.c-hip.net/2012/seminari/kudeljnjak-lozic/works.html (26.7.2024.)

[6] Unity Technologies. (2023). Unity download archive

https://unity.com/releases/editor/archive (26.7.2024.)

[7] Neven Jović (Svibanj 2018.) Koliko nam je zvuk u videoigrama stvarno važan?

https://www.hcl.hr/game-special/koliko-nam-je-zvuk-u-videoigrama-stvarno-vazan-121904/ (6.8.2024.)

[8] Unity Technologies. Audio Source

https://docs.unity3d.com/Manual/class-AudioSource.html (6.8.2024.)

[9] Head-up-display (video gaming)

https://handwiki.org/wiki/Head-up_display_(video_gaming) (7.8.2024.)

13. POPIS SLIKA

14. IZJAVA O AUTORSTVU

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

Bana Josipa Jelačića 22/a, Čakovec

IZJAVA O AUTORSTVU

Završni/diplomski rad isključivo je autorsko djelo studenta te student odgovara za istinitost, izvornost i ispravnost teksta rada. U radu se ne smiju koristiti dijelovi tuđih radova (knjiga, članaka, doktorskih disertacija, magistarskih radova, internetskih i drugih izvora) bez pravilnog citiranja. Dijelovi tuđih radova koji nisu pravilno citirani, smatraju se plagijatom i nezakonitim prisvajanjem tuđeg znanstvenog ili stručnoga rada. Sukladno navedenom studenti su dužni potpisati izjavu o autorstvu rada.

Ja, DORIJAN BAHNJIK (ime i prezime studenta) pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor/ica završnog/diplomskog rada pod naslovom EDUKATIVNA IGRA ZA POUČAVANJE UČENIKA U RAZVRSTAVANJU OTPADA

te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

Student/ica:

DBahnjik

(vlastoručni potpis)