Izrada 2D video igre u Unity-u

Malić, Zdravko

Sveučilište
Sjever

Završni rad br. 715/MM/2021

Izrada 2D video igre u Unity-u

Zdravko Malić, 2937/336

Varaždin, lipanj 2021.godine

Sveučilište
Sjever

Odjel za Multimediju, oblikovanje i primjenu

Završni rad br. 715/MM/2021

Izrada 2D video igre u Unity-u

Student

Zdravko Malić, 2937/336

Mentor

Doc.dr.sc. Andrija Bernik

Varaždin, lipanj 2021.godine

Sveučilište Sjever
Sveučilišni centar Varaždin
104. brigade 3, HR–42000 Varaždin

Prijava završnog rada

Definiranje teme završnog rada i povjerenstva

ODJEL Odjel za multimediju

STUDIJ preddiplomski stručni studij Multimedija, oblikovanje i primjena

PRISTUPNIK Zdravko Malić | MATIČNI BROJ 2937/336

DATUM 30.4.2021. | KOLEGIJ 3D modeliranje

NASLOV RADA Izrada 2D video igre u Unity-u

NASLOV RADA NA ENGL. JEZIKU Making a 2D video game in Unity

MENTOR doc.dr.sc. Andrija Bernik | ZVANJE Docent

ČLANOVI POVJERENSTVA

1. mr.sc. Dragan Matković, v. pred. - predsjednik
2. pred. Nikola Jozić - član
3. doc.dr.sc. Andrija Bernik - mentor
4. doc.art. Robert Geček - zamjenski član
5.

Zadatak završnog rada

BROJ 715/MM/2021

OPIS

Svaka računalna igra bilo 2D ili 3D predstavlja aplikaciju obzirom na skripte koje se izvode u njenoj pozadini. Izrada aplikacija za mobitel je proces stvaranja softvera za razne mobilne i slične uređaje. Taj biznis čvrsto raste i donio je mnogo poslova. Raznolik izbor alata današnjice nam omogućuje lagano, brzo i intuitivno stvaranje aplikacija. Za izradu aplikacije treba skicirati UI kako bi bio pregledan, originalan i zanimljiv. Asseti za 2D projekt mogu se napraviti unutar Illustratora te izvesti u Unity. Unity je besplatan program za izradu igara, ali može poslužiti i za izradu aplikacija. Asseti se slažu u prozoru Scena da odgovaraju skici te se slažu najboljim redoslijedom da budu pregledni i u korelaciji s programskim kodom. U Unity-u se sve može animirati na intuitivan način te se samim tipkama može dodati mogućnost kliktanja na njih bez ikakvog koda. Programski jezik korišten za ovaj projekt je C#. To je jednostavnija, sigurnija i brža verzija jezika C.

REPUBLIKA HRVATSKA SVEUČILIŠTE SJEVER KOPRIVNICA

ZADATAK URUČEN 16.06.2021. | POTPIS MENTORA

Predgovor

Tijekom izrade završnog rada primjenjeno je znanje stečeno na kolegijima Programski alati I, Programski alati II, Grafički dizajn, Grafički alati I i Web dizajn. Uz pomoć programa Illustrator koji je poslužio za skiciranje i grafički dizajn, Unity koji je game engine i Visual Studio koji se koristi zajedno sa Unity-em za dobro iskustvo u programiranju izrađena je aplikacija za Android mobilne uređaje.

Zahvaljujem se mentoru, doc.dr.sc.Andriji Berniku. Na pruženoj pomoći i savjetima prilikom izrade ovog završnog rada.

Sažetak

Izrada aplikacija za mobitel je proces stvaranja softvera za razne mobilne i slične uređaje. Ovo tržište čvrsto raste i donijelo je mnogo poslova. Raznolik izbor alata današnjice nam omogućuje lagano, brzo i intuitivno stvaranje aplikacija.

Aplikacija koja je napravljena za ovaj projekt zove se Twister. To je stvarna fizička igra no nema potrebe se kupuje kada je aplikacija dovoljna. Ipak, svi danas imaju pristup mobitelu pa je aplikacija dostupna svima. Ona nudi pristup raznim varijacijama igre, što ju čini još boljim izborom, nego kupovanje i trošenje novca na razne verzije igre. Također ima glasovne naredbe da ne treba gledati u mobitel tijekom igranja igre. Snimljene su te import-ane u Unity, pa je zvuk čišći nego da se koristi neki automatski glasovni čitač teksta. Imena igrača mogu se unijeti zvukom, te će ih aplikacija reproducirati po potrebi.

Za izradu aplikacije treba skicirati UI kako bi bio pregledan, originalan i zanimljiv. Asseti za 2D projekt mogu se napraviti unutar Illustrator-a te izvesti u Unity. Unity je besplatan program za izradu igara, ali može poslužiti i za izradu drugih aplikacija. Asseti se slažu u prozoru Scena da odgovaraju skici te se slažu najboljim redoslijedom da budu pregledni i u korelaciji s programskim kodom. U Unity-u se sve može animirati na intuitivan način te se samim tipkama može dodati mogućnost kliktanja na njih bez ikakvog koda. Programski jezik korišten za ovaj projekt je C# što je jednostavnija, sigurnija i brža verzija jezika C.

Ključne riječi: Unity, video-igre, C#, programiranje,

Popis korištenih kratica

UI – Useer Interface

UX – User Experience

IDE - Integrated Development Environment

GUI – Graphical User Interface

JIT – Just In Time Manuifactuirng

YT - Youtube

Sadržaj

1. Uvod

U današnjem svijetu je nemoguće zamisliti život bez aplikacija koje se koriste na dnevnoj bazi. Od razmjene poruka, učenja jezika i praćenja kalorija do bankarstva i ulaganja, čini se da za sve postoji aplikacija. Izrada aplikacija je upravo to što zvuči, iako nije tako jednostavno. Razvoj mobilnih aplikacija postupak je pisanja softvera za mobilne uređaje. To je jedinstven pothvat jer pruža programeru priliku da u relativno kratkom razdoblju izradi aplikaciju od nule. Programeri usavršavaju svoje vještine na temelju operativnog sustava na kojem rade. Trenutna dva najjača igrača su: Googleov Android OS i Appleov iOS. Iako postoje drugi pametni telefoni i mobilni uređaji sa svojim aplikacijama, oni čine mali dio ukupnog tržišta. Može se naučiti raditi s jednim operativnim sustavom i prelaziti na drugi dok napreduju u svojoj karijeri. Sposobnost rada s oba operativna sustava učini programera konkurentnijom na tržištu rada programera mobilnih aplikacija. Posao razvojnog programera mobilne aplikacije je da: dizajnira, izrađuje i ažurira mobilne aplikacije, bilo na Androidu ili iOS-u. On ili ona surađuje s dizajnerom korisničkog iskustva (UX) kako bi osigurao da je njihov dizajn usklađen s potrebama korisnika. Zajedno stvaraju makete za aplikaciju. Programer se također mora koordinirati s nadređenima u vezi sa strategijama objavljivanja. Kod izrade aplikacije developer nije ograničen jednim programom za izradu aplikacije. U današnje vrijeme postoji puno raznih programa od koji se dosta njih specijalizira za svoje područje i u njemu je bolje nego drugi programi. Na primjer Unity nije najbolje optimiziran za jednostavne aplikacije, ali u drugim programima se igre ne mogu napraviti lagano kao u njemu.

2. Najbolji programi za izradu Android aplikacija

Razvoj Androida raste svakim danom. Prema Statista-i (vodeći dobavljač podataka o tržištu i kupcima [2]) na Google Play-u dostupno je 2,9 milijardi aplikacija. Kako se upotreba Android uređaja povećava potreba za visokokvalitetnim Android aplikacijama raste. Programeri Androida pokušavaju ostati produktivni kako bi stvarali kvalitetnije aplikacije. U tu svrhu na raspolaganju su im mnogobrojni korisni alati i aplikacije.

Slika 1: Android Studio logo

Android Studio – stvoren je 2013. godine, nezaustavljiv je alat za razvoj Eclipse Androida je jedan i jedini IDE za nativne (stvorene za specifičnu platformu) Android aplikacije. Besplatan je i aktivno ga podržava Android razvojna zajednica. Android Studio je, bez sumnje, prvi među alatima Android programera. Omogućuje jednostavno uređivanje koda, otklanjanje pogrešaka i testiranje.

Slika 2: AIDE logo

AIDE - omogućuje izradu Android aplikacije na Android uređajima. Pruža način ne samo za pisanje koda na telefon ili tablet nego i pokretanje, testiranje i uklanjanje pogrešaka. Dobra je

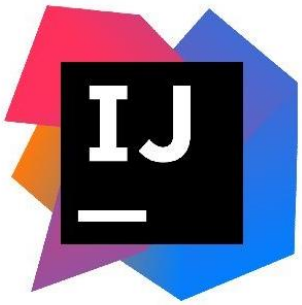
opcija za programere početnike, za razliku od Android Studija. Loša strana je što podržava samo Javu i C / C ++.

Slika 3: Stetho logo

Stetho – je biblioteka otvorenog koda koju je razvio Facebook i dizajnirano za brzo uklanjanje pogrešaka u aplikacijama. Stetho daje aplikaciji iskustvo kao na web mjestu jer omogućuje pristup Chrome Developer Tools značajci nativnoj desktop browser-u. Pomoću Chrome DevTools može se pregledati hijerarhija aplikacije, mogu se nadgledati mrežne aktivnosti, upravljati SQLite bazom podataka, nadgledati zajedničke postavke SharedPreferences, itd.

Slika 4: Gradle logo

Gradle – je ospen-source sustav automatizacije koji se pojavio 2013. Kombinirajući najbolje od Apache Maven i Apache Ant, ovaj je sustav idealan za velike projekte s više pod-projekata. Gradle olakšava dodavanje third-party library-ja s jednim linijom koda. Uglavnom koristi za razvoj softvera za Android pomoću Jave, ali mogu se koristiti i dodaci Groovy i Scala.



Slika 5: IntelliJ IDEA logo

IntelliJ IDEA - JetBrains je stvorio Java IDE s podrškom za Android. Dobra je alternativa za Android Studio, koja se uglavnom koristi za jednostavnije aplikacije. IntelliJIDEA je brz i odmah dolazi s raznim razvojnim alatima: pametno dovršavanje koda, trenutna analiza koda, refaktoriranje (rekonstrukcija postojećeg koga bez promijene njegove funkcionalnosti) i JetBrains dodatci.

Slika 6: Sourcetree logo

Source Tree - jednostavan i besplatan alat koji pruža jednostavan način upravljanja Git repozitorijima pomoću Git GUI-a. Sve vizualizirati promjene se mogu vizualizirati, commit-ovi, grananja, a da u naredbeni redak ne treba upisati niti jednu naredbu.

Slika 7: GameMaker: Studio logo

GameMaker: Studio - jedan od najpopularnijih game engine-a od YoYo games koji omogućuje razvoj 2D igara za Android i druge platforme. GameMaker: Studio moćan je, ali

jednostavan, lagan za korištenje s intuitivnim sučeljem za drag-n-drop. Također dobar alat za započinjanje karijere razvoja igara.

Slika 8: Unity logo

Unity 3D - stvarajući mobilne igre, može se pretvoriti cross-platform engine s bogatstvom značajki za grafički intenzivne i VR / AR igre. Unity pruža prilagođenije alate za razvoj igara, poput storytelling-a, promijene na sljedeći level, prikazivanja u stvarnom vremenu i mnogih drugih.[1][3]

3. Pojam Game Engine-a

Koncept game engine-a je jednostavan za shvatiti. To je platforma koja obavlja uobičajene zadatke vezane za igre kao što su: renderiranje, fizike, te input, tako da se developeri (umjetnici, dizajneri, programeri i ostali) mogu koncentrirati na detalje koji čine igru jedinstvenom. Engine-i su kolekcije komponenata koje imaju vrlinu ponovnog korištenja i mogu biti upotrijebljeni da se igra ostvari. Često game engine-i dizajnirani su namjerom da su u nekom određenom zadatku izrazito dobri npr. Havok za fizike, Miles Sound System za zvuk ili Blink za videozapise. Oni koji dobro rade s fizikama su jako popularni i značajni jer nam pružaju realističniju okolinu.[4]

4. Unity

Unity, koji se smatra jednim od najboljih game engine-a 2021. godine. Popularan je, moćan i svestran engine koji omogućuje stvaranje 2D, 3D i igara za više igrača na velikom broju platformi. Unity je alat za izradu prototipa svega, od igara do interaktivnih vizualizacija. Prvi

put je najavljen i objavljen u lipnju 2005. godine na Apple Inc.-ovoj svjetskoj konferenciji razvojnih programera kao Mac OS X ekskluzivni motor za igre. Prvu verziju Unity-a (1.0. 0) stvorili su: David Helgason, Joachim Ante i Nicholas Francis u Danskoj. Od tada se postupno proširivao kako bi podržavao razne desktop platforme, mobilne uređaje, konzole i platforme za virtualnu stvarnost. Posebno je popularan za razvoj iOS i Android mobilnih igara. Čak i s ograničenim iskustvom kodiranja ili bez njega, mogu se brzo steći vještine za izgradnju igara koristeći Unity Editor i C #. Ne može pružiti neke stvari kao što je najmodernija grafika današnjice, ali je zato vrlo fleksibilan, dobro je objašnjen u dokumentaciji i vrlo je otvoren za izgradnju gotovo bilo kojeg žanra igre. U Unity-u je nastalo mnoštvo vrlo uspješnih igara poput Escape from Tarkov (FPS), Monument Valley (Puzzler) i This War of Mine (Strategy / Survival). Unity ima punu podršku za VR i AR te bi stoga mogao biti izvrstan alat za istraživanje arhitekture, automatizacije i simulacija s klijentima.

u Unity-u se *gameplay* odvija u scenama. Scene su level-i u kojima se svi aspekti igre kao što su level-i igre, naslovnog zaslona, izbornika i video isječaka. Prema zadanim postavkama, nova Scena u Unity-u će imati kamera objekt u sceni koja se zove *Main Camera*. Moguće je dodati više kamera u scenu. Glavna kamera renderira sve što vidi ili zahvati u određenom regiji koja se zove *viewport*. Sve što dođe u ovu regiju postaje vidljivo za igrača.

Scena sama po sebi je napravljena od objekata koji se zovu GameObjects. GameObjects-i mogu biti bilo što od modela igrća do GUI-a na ekranu, od tipki i neprijatelja do nevidljivih „menadžera“ kao što su izvori zvuka. GameObject-i imaju skupinu komponenti koje su prikvačene na njih, te koje opisuju kako se oni ponašaju u sceni, također i kako reagiraju na druge stvari u sceni. Najvažnija komponenta bilo kojeg GameObject-a je Transform. Svaki objekt koji postoji u sceni će imati transform koji definira njegovu poziciju, rotaciju i veličinu uzeći u obzir Game World ili parent-a, ako ga ima.

Dodatne komponente mogu biti zakvačene na objekt klikom na Add Component i birajući željenu komponentu. Ako se objektima želi pružiti isprogramirano ponašanje na njih ćemo dodati Script.[5]

5. Nedostatci Unity-a kod razvoja Android aplikacija

Unity nije napravljen sa svrhom izrade jednostavnih aplikacija za Android, pa se lagano može zaključiti da postoji boljih programa za izradu aplikacija. Jedan od njih je Android Studio koji koristi novi programski jezik Kotlin. Neke od prednosti Android Studio-a su:

- **Mogućnosti hardvera / senzora** Unity podržava najčešće senzore, ali ako su potrebni senzori za posebne slučajeve poput NFC-a, moraju se upotrijebiti Android plug-ins ili library u asset-store-u. Dosta Unity-evih library-ja u asset store-u je zastarijela, a stvaranje vlastitih plug-in-ova često ionako zahtijeva upotrebu Kotlin-a.

- **Android-like UI** Unity nema integriranu podršku za uobičajene Android UI-elemente pa treba stvoriti novi korisnički UI. Znatno je sporije i tromije korisničko iskustvo nego da je korišten Android Studia (npr. stvaranje klizajućeg prikaza u Unityu je dosta zahtivijevan zadatak). Sporost također može biti povezana s činjenicom da će Unity aplikacija renderirari cijeli zaslon za svaki frame, dok izvorni Kotlin ima podršku za djelomično renderiranje. Takva sporost je vidljiva na starijim mobilnim uređajima i ima utjecaja na potrošnju energije.

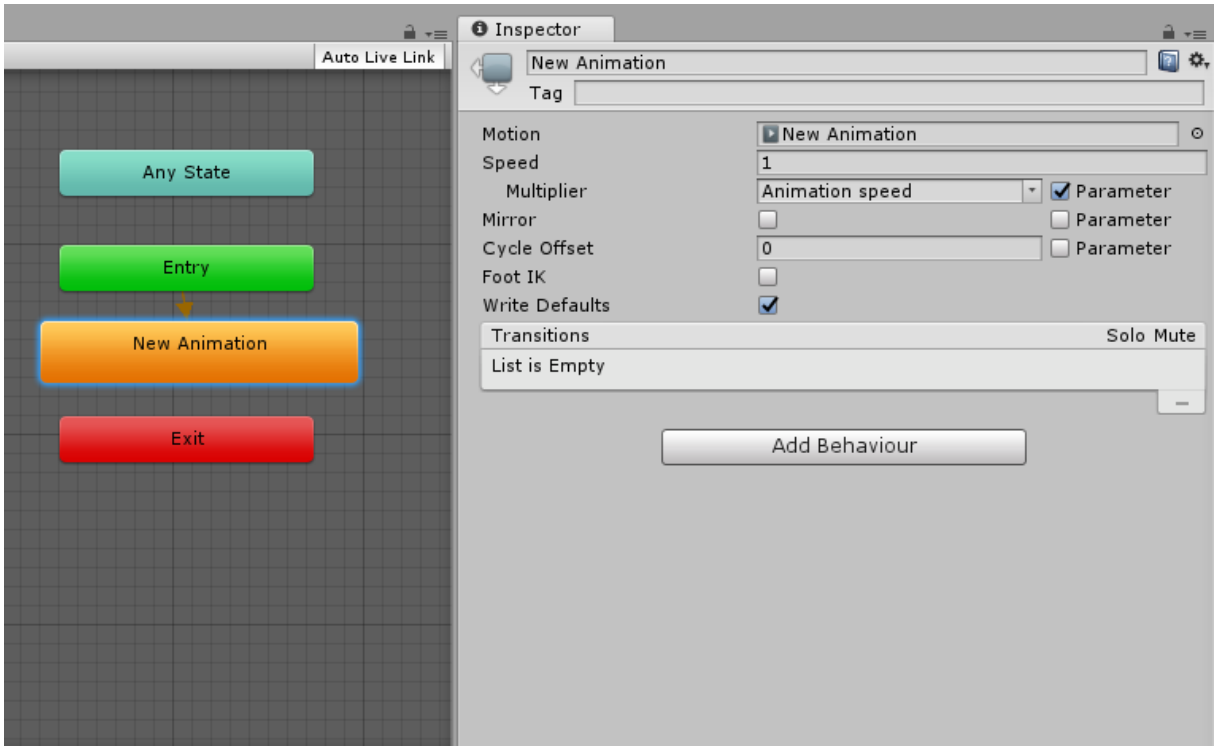
Pristupačnost Unity ne podržava čitače zaslona na pametnim telefonima zbog načina renderiranja UI-a. To se može riješiti na način da sami isprogramiramo funkcionalnost ili se kupi add-on. No korištenjem uobičajenog Android UI-a ovoga problema neće biti.[6]

6. Budućnost Unity-a

Dok Unity nastavlja poboljšavati svoje alate za stvaranje, jedan od dijelova u kojem se Clive Downie nada da će postići napredak u narednim godinama je obrazovanje. Clive Downie je glavni direktor marketinga Unity Technologies-a, te je odgovoran za globalne strategije i taktike u marketingu kupaca, marketingu proizvoda i e-trgoviniZadovoljava time što je Unity

samo jedan od elemenata u razvijanju programa. Umjesto toga, želi da engine postane temelj načina na koji se kreativne digitalne vještine predaju u učionicama. Downie je u intervjuu za gameindustry.biz rekao: "Obrazovanje nam je zanimljivo, kao i sposobnost da Unity bude nešto što će studenti shvatiti kao jednaku tranformaciju za učenje i razvoj kao što su nekada bili olovka i papir…Želimo da ovaj kreativni alat u današnjice bude nešto što će studenti uzeti sa sobom izlazeći iz obrazovanja te će ga znati koristiti jer će to biti važan dio mnogih različitih karijera tijekom sljedećeg desetljeća. Nije važno jeste li proizvođač igara, animator, filmaš, industrijski dizajner, arhitekt, dizajner proizvoda, Unity može biti dobar alat za sve to."[7]

7. Animiranje u Unity-u



Slika 9: Animator u Unity-u

Unity omogućuje animatorima rad u stvarnom vremenu ubrzavajući tradicionalne načine i pružajući umjetnicima, producentima i redateljima više mogućnosti za kreativnu slobodu, brze povratne informacije i umjetničke iteracije na fleksibilnoj platformi. Uz Unity, proizvodnja u stvarnom vremenu je stvarnost. Animatiom Component-i mogu se dodijeliti animacijski isječci, te se reprodukcija može kontrolirati iz skripte. Sustav animacije u Unity-u temelji se na weight-

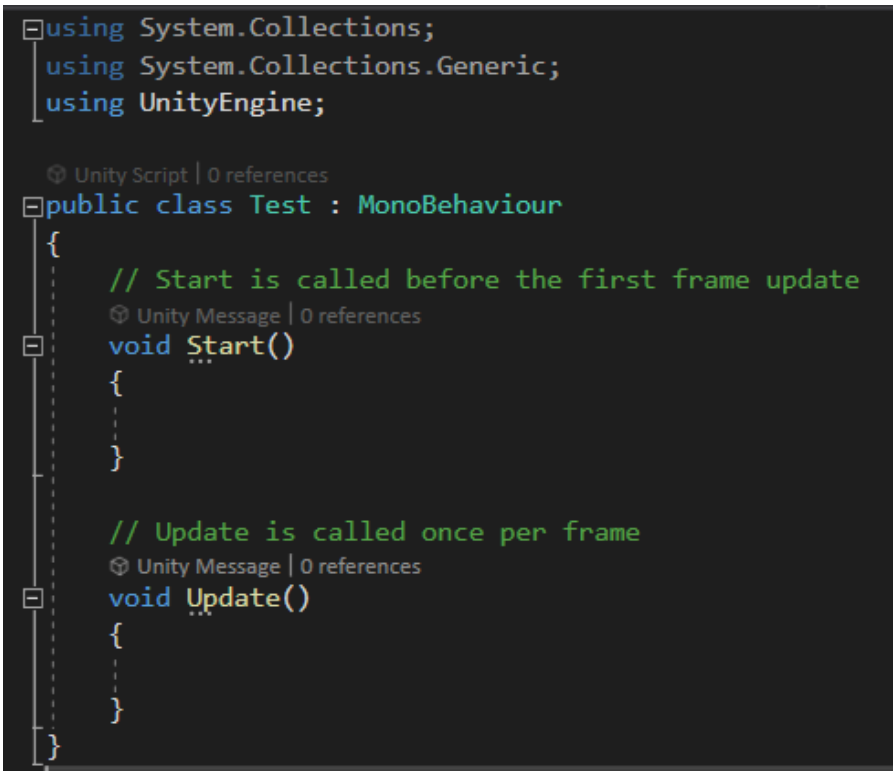
ovima te time podržava kombiniranje animacija, aditivne animacije, miješanje animacije, slojeve i potpunu kontrolu nad svim aspektima reprodukcije.[8]

8. Stvaranje i korištenje Unity skripti

Ponašanje GameObject-ata kontroliraju Component-e koje su stavljene na njih. Iako ugrađene Unity Component-e mogu biti korisne za mnogo raznih strvari, no ubrzo se pojavi potreba za novim stvarima, te trebata implementirati vlastite Component-e. Unity omogućava stvaranje novih jednistvenih Component-a pomoću skripti koje omogućuju aktivaciju događaja u igri, mijenjanje svojstava Component-a kroz vrijeme i odgovaranje na unos korisničkih input-a. Unity izvorno podržava programski jezik C #. To je industrijski standardni jezik sličan Javi ili C ++. Uz njega, mnogi drugi **.NET** jezici mogu se koristiti s Unity-em ako mogu kompilirati kompatibilni **DLL**.

Za razliku od većine ostalih Asset-a, skripte se uobičajeno izrađuju izravno u Unityju. Skripta može biti stvorena u izborniku Create u gornjem lijevom kutu Project panel-a ili odabirom Assets> Create> C # Script iz glavnog izbornika. Nova će se skripta stvoriti u bilo kojoj odabranoj mapi na Project panel-u. Odabrat će se naziv nove datoteke skripte, od koje će se zatražiti da se unese novo ime.

Kada se dvaput klikne na skript Asset u Unity-u, on se otvoriti u text editor-u Prema zadanim postavkama, Unity će koristiti Visual Studio, ali može se odabrati bilo koji uređivač (Unity> Preferences). Početni sadržaj skripte izgleda ovako:



Slika 10: Zadane Unity funkcije

Skripta povezuje internu funkcionalnost Unity-a implementirajući class koji proizlazi iz ugrađene klase MonoBehaviour. Klasa je kao svojevrsni nacrt za stvaranje nove vrste Component-e koja se može staviti na GameObject-e. Svaki put kada na GameObject stavimo Component skripte on stvara novu instancu objekta definiranog blueprint-om. Naziv class-a preuzet je iz imena koje je dodijeljeno datoteci kada je kreirana. Naziv klase i naziv datoteke moraju biti jednaki da bi Component skripte mogao biti spojen na GameObject.

Dvije funkcije su definirane unutar klase. Funkcija Update mjesto je za kod koji će se izvršiti jedanput u svakom frame-u za GameObject. To može uključivati kretanje, aktiviranje događaja i reagiranje na user input, u osnovi sve ono čime se treba rukovati tijekom igranja. Da bi se funkciji Update omogućili da obavlja svoj posao, često je korisno deklarirati varijable, pročitati preference i povezati se s drugim GameObject-ima prije nego što se dogodi bilo kakva radnja u igri. Unity će izvršiti funkciju Start prije početka gameplay-a (tj. Prije nego što se funkcija Update pozove prvi put) i idealno je mjesto za bilo kakvu inicijalizaciju. [9]

a. Funkcije

U C#-u, funkcija je način zapakiravanja koda koji radi nešto, te onda vraća vrijednost. U razliku na C, u C#-u i u drugim jezicima funkcije ne postoje same za sebe. Dio su objektno-orijentiranog pristupa programiranju. U C#-u funkcija može biti nazvana mebmber funkcija jer je member method-a. Postoje 2 vrste method-a: instance method i static method. Objastin će se instance method. Primjer ispod definira jednostavni class koji se zove Test. Ovaj program je jednostavan Consele program pa ne treba sve staviti u class.

var t = new Test( );

Ovaj kod je ispravan, ali neće učiniti ništa osim stvoriti instancu „t“ praznog class-a. Kod ispod će dodati funkciju, tj. metodu koja ispisuje riječ „Hello“.

```
Using System;
namespace function1
{

    class Test
    {

        public void SayHello()
        {
            Console.WriteLine(„Hello“);
        }
    }
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            var t = new Test();
            t.SayHello();
            Console.ReadyKey();
        }
    }
}
```

Slika 11: Primjer C# funkcije

Say Hello funkcija je public funkcija, što zanči da je vidljiva i izvan class-a. Ako maknemo riječ public i pokušamo compajlati kod dobit ćemo compilation error. Ako dodamo riječ private tamo gdje je bila riječ public dobit ćemo isti error. Riječ void znači da funkcija ne vraća nikakvu vrijednost.

Karakteristike funkcije su: razina pristupa (public, private itd.), return vrijednost (void ili neka druga vrsta kao int), ime method-a ili bilo koji parametri method-a (definirani su u zagradama poslije imena method-a).[10]

b. Singleton pattern

```
public class GameManager : MonoBehaviour
{
    public static GameManager Instance;

    void Awake()
    {
        if (Instance == null)
            Instance = this;
        else if (Instance != this)
            Destroy(this);
    }
```

Slika 12: Singleton Pattern

Singleton je klasa koja dopušta da samo jedna **instanca** sebe bude stvorena, te također često dopušta jednostavan pristup toj instanci. Najčešće, Singleton ne dopušta da bilo kakvi parametri (koje izradimo pomoću Constructor-a) budu specificirani kod stvaranja instance, jer bi u suprotnom drugi zahtjevi za instancu sa drugim parametrima mogli stvoriti problem. Tipično potreba za korištenje Singleton je u takozvanim lijenim situacijama. Drugim riječima instanca ne stvorimo sve dok nam ne zatreba.

Ima puno načina implementacije Singleton pattern-a u C#. Sad će svi biti napisani od najlošijeg, koji nije thread-safe, sve do najboljega koji potpuno laizy-loaded, thread-safe, jednostavan i ima visoke performase.

- Jedan Constructor koji je privatan i nema parametre. To spriječava druge klase da ga instanciraju. Također spriječava subclassing. The factory pattern može biti korišten ako je potrebna jedna instaca izvornog tipa, ali točan tip se ne zna sve do runtime-a.
- Klasa je zapečaćena. Ovo je zbog problema navedenog iznad, ali može pomoći JIT (just in time manufacturing – stvaranje tek dok treba) da bolje optimizira stvari.
- Statična varijabla koja sadrži referencu do jednom stvorene instance.
- Javna statična sredstva za dobivanje reference do jednom stvorene instance.[11]

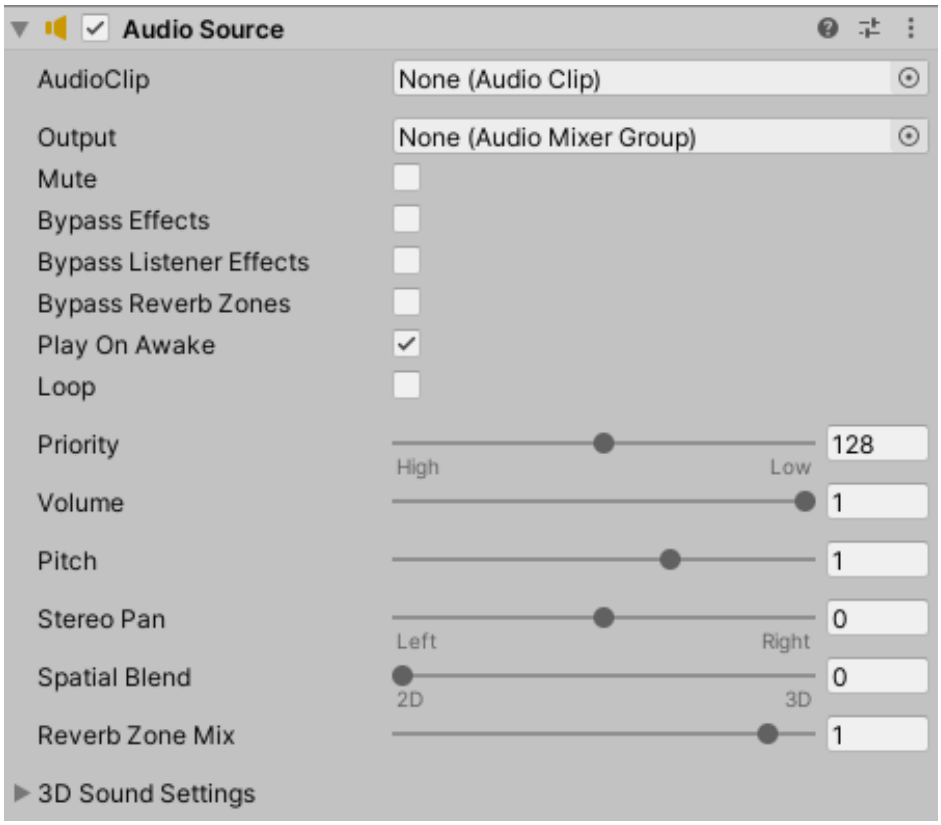
c. Input

Input je srce onoga što čini projekt interaktivnim u stvarnom vremenu. Pomoću dobrog sustava unosa brzo se mogu postaviti kontrole za više platformi, od mobilne do VR-a. Samo treba povezati radnje s logikom koda, a zatim se vizualno mogu omogućiti za različite uređaje i kontrole u prozoru Input Action.[12]

d. Audio Source

Postoji nekoliko načina za puštanje zvukova u Unity-u, neki od njih su:

- **audioSource.Play** pušta jedan klip iz skripte.
- **audioSource.PlayOneShot** pušta preklapajući, ponavljajući i non-looping zvuk.
- **AudioSource.PlayClipAtPoint** pušta zvuk na 3D poziciji bez Audio Source-a.
- **audioSource.PlayDelayed** ili **audioSource.Playscheduled** pušta zvuk u određeno vrijeme u budućnosti.
- **Play on Awake** na Audio Source-u pušta zvučni zapis kada je objekt stvoren.



Slika 13. Audio Source koopnenta

AudioSource.Play() je najčešća metoda reprodukcije zvuka i najprikladnija je za puštanje ponavljajućih zvukova i glazbe. To je zato što druge metode, poput Play One Shot ne mogu pokrenuti ponavljajuće zvukove i također zato što će se korištenjem Play-a omogućiti reprodukcija samo jednog isječka s tog izvora zvuka, drugim riječima, neće dopustiti istovremenu reprodukciju više zvukova u isto vrijeme. Dakle pozivanjem Play na Audio Source koji se već reproducira zaustavit će izvor zvuka i ponovno pokrenuti isječak.[13]

e. Corutines

```
void Update()
{
    if(Input.GetMouseButtonDown(0))
    {
        StartCoroutine(MoveTank());
    }
}

IEnumerator MoveTank()
    {
        while(facingWrongWay)
        {
            TurnTank();
            yield return null;
        }

        while (notInPosition)
        {
            MoveToPosition();
            yield return null;
        }

        yield return new WaitForSeconds(1);
        Fire();
    }
```

Slika 14: Primjer Corutine-a

Kod Courutine-a kod izgleda kao to-do list, no funcionira te se svaka radnja izvršava nakon završetka posljednje. Da se ovaj ista funkcionalnost pokušala napraviti samo u Update() funkciji kod bi bio duži i nepregledniji. Unity obrađuje svaku While petlji dok njeno stanje više nije istinito, te tek onda prijeđe na sljedeću. Zapravo se ne razlikuje od načina na koji Unity izvršava redovnu funkciju, osim što se sada logika provodi na više freme-ova umjesto u jednome frame-u. Funkcionira zbog ključne riječi yield koja Unity-u prekida rad i nastavlja ponovno na sljedećem frejmu. Prednost Courutine-e je slijed događaja koje je lakše napisati i kojima je lakše upravljati. Također je učinkovitiji, jer Unity više ne mora provjeravati uvjete koji još nisu relevantni. Nakon riječi yeld slijedni return koji kao i u običnoj funkciji zaustavlja izvršavanje koda do te točke i vraća kontrolu natrag na mjesto koje je pozvalo samu metodu. Nakod yield return se može

odrediti koliko dugo Unity treba čekati prije nego nastavi. Ključna riječ korištena u ovom projektu je WaitForSeconds. Dopušta specificiranje točne količine vremena prije daljnjeg nastavka izvršavanja koda. Može ga se koristiti samo unutar Corutine-a (ne radi u Update-u). Sama Corutine-a se poziva pomoću StartCoroutine(„CorutineName“);, a završava jednostavno kada je sav kod unutar nje izvršen. Ako se želi zaustiviti ručno, može se pomoću StopCorutine(„CorutineName“) ili StopAllCoroutines();.[14]

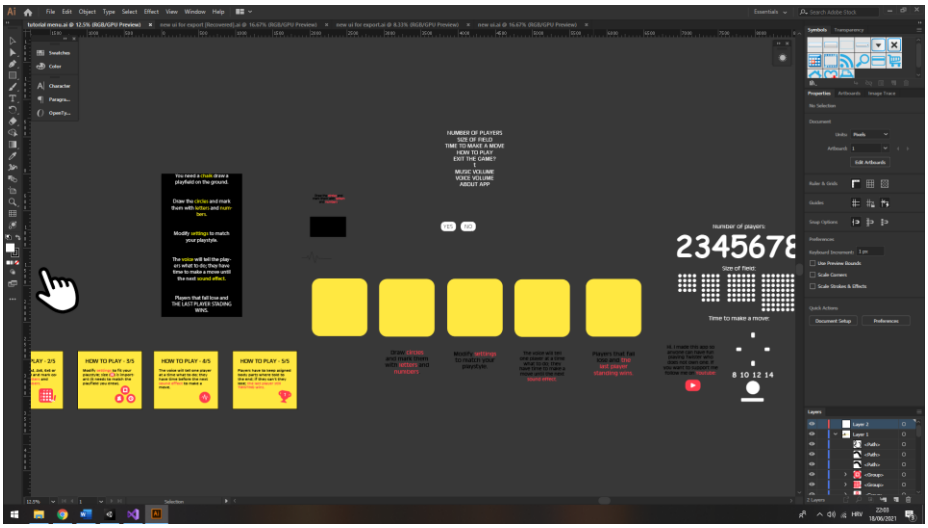
Slika 15: WaitForSeconds metoda

9. Praktični dio

a. Osmišljavanje izgleda i funkcionalnosti

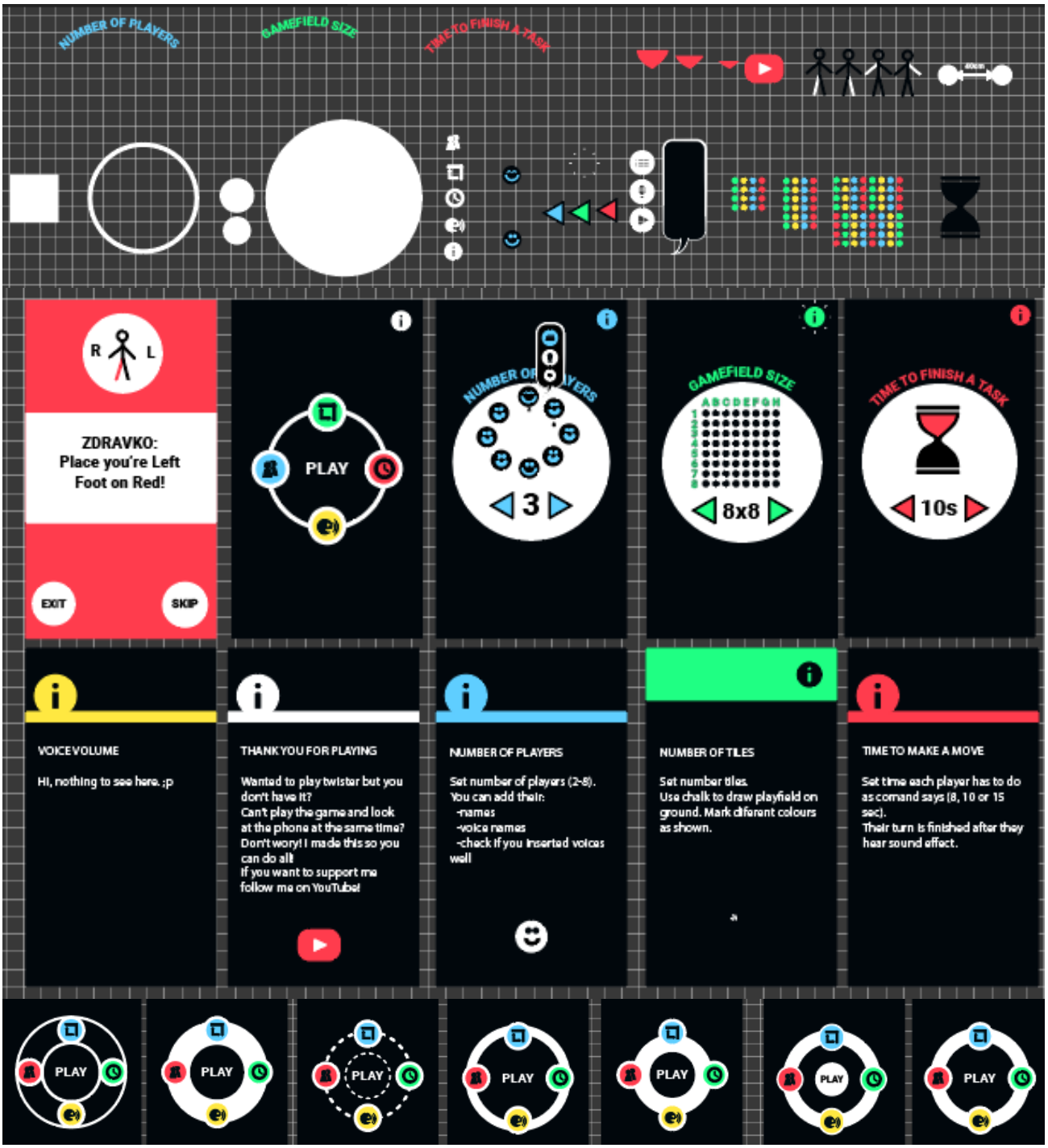
Slika 16: Meniji za Twister

U praktičnom dijelu rada bit će objašnjen proces izrade aplikacije „Twister“. Rad je započet kreacijom mnogobrojnih prototipa uzimajući u obzir što korisnici žele, što je jednostavno i što je zanimljivo. Ovaj proces se radi u Adobe Illustrator-u jer je izgled aplikacije jednostavan, a Illustrator može pružiti brzu kreaciju i jednostavnu organizaciju jednostavnih oblika. Također prednost Illustrator nad Photoshopom je ta da koristi vektorsku grafiku pa se kasnije sve može eksport-ati u bilo kojoj rezoluciji. Igra „Twister se igra na krugovima pa je odličan izbor UI-a bio krug gdje pojedinačne tipke sjedne na njegovoj liniji.



Slika 17: Prva verzija game Asseta

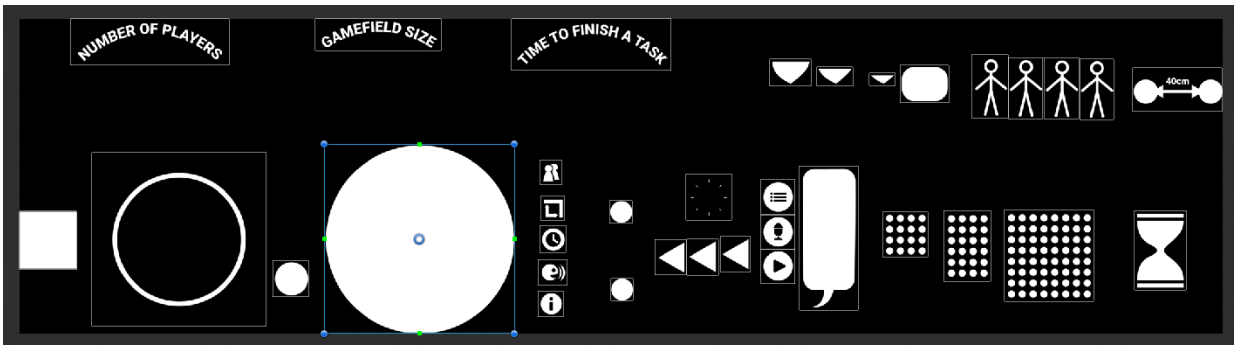
Nadalje treba napraviti detaljan dizajn svih ostalih elemenata koji će pojaviti pritiskom na rano razne tipke. Tu su i razni tutoriali koji objašnjavaju pravila, postavke za način igranja igre (broj igrača, trajanje poteza igrača, te broj/veličina polja na kojem se igra).



Slika 18: Druga verzija game Asseta

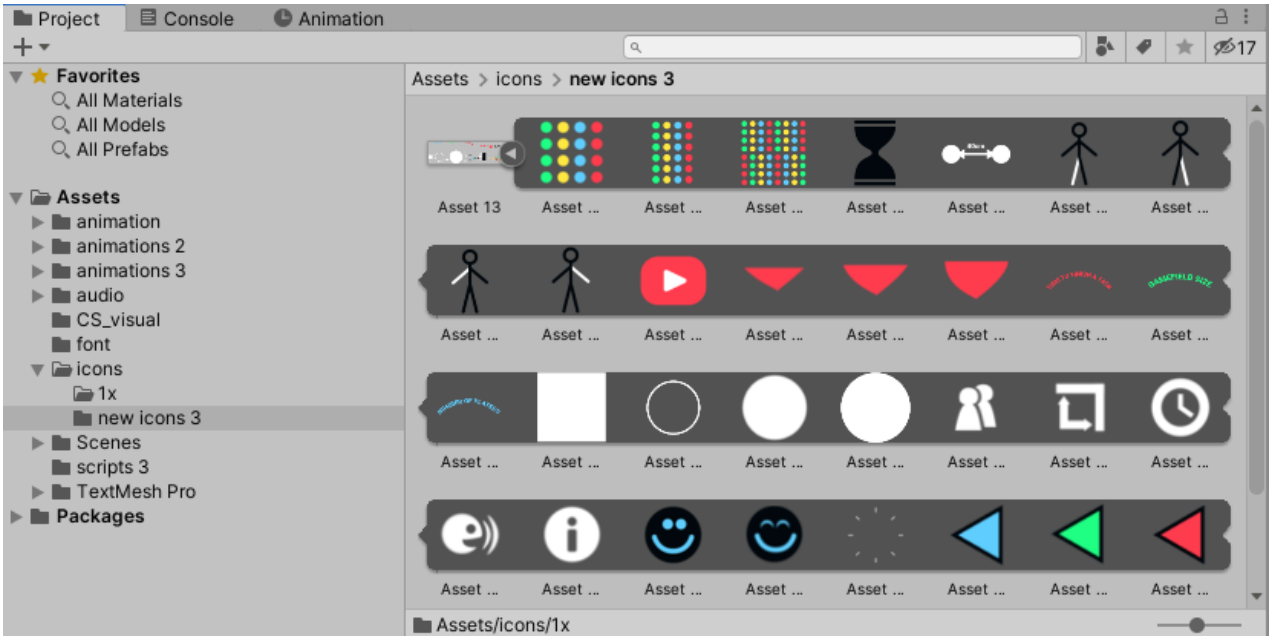
Ovako izgleda krajnji plan za dizajn aplikacije. Dobra ideja je napraviti sve u stilu krugova jer je već glavni izbornik izrađen u tom stilu. Također će biti isprogramirano da se glavni izbornik rotira za zanimljiviji UI. Treba pripaziti na ne pretjerivanje da se ne bi smanjio UX koji je još bitniji. Na posljetku sve dizajnirane elemente treba razdvojiti i rasporediti te eksport-ati u Unity.

b. Import grafičkog sadržaja u Unity



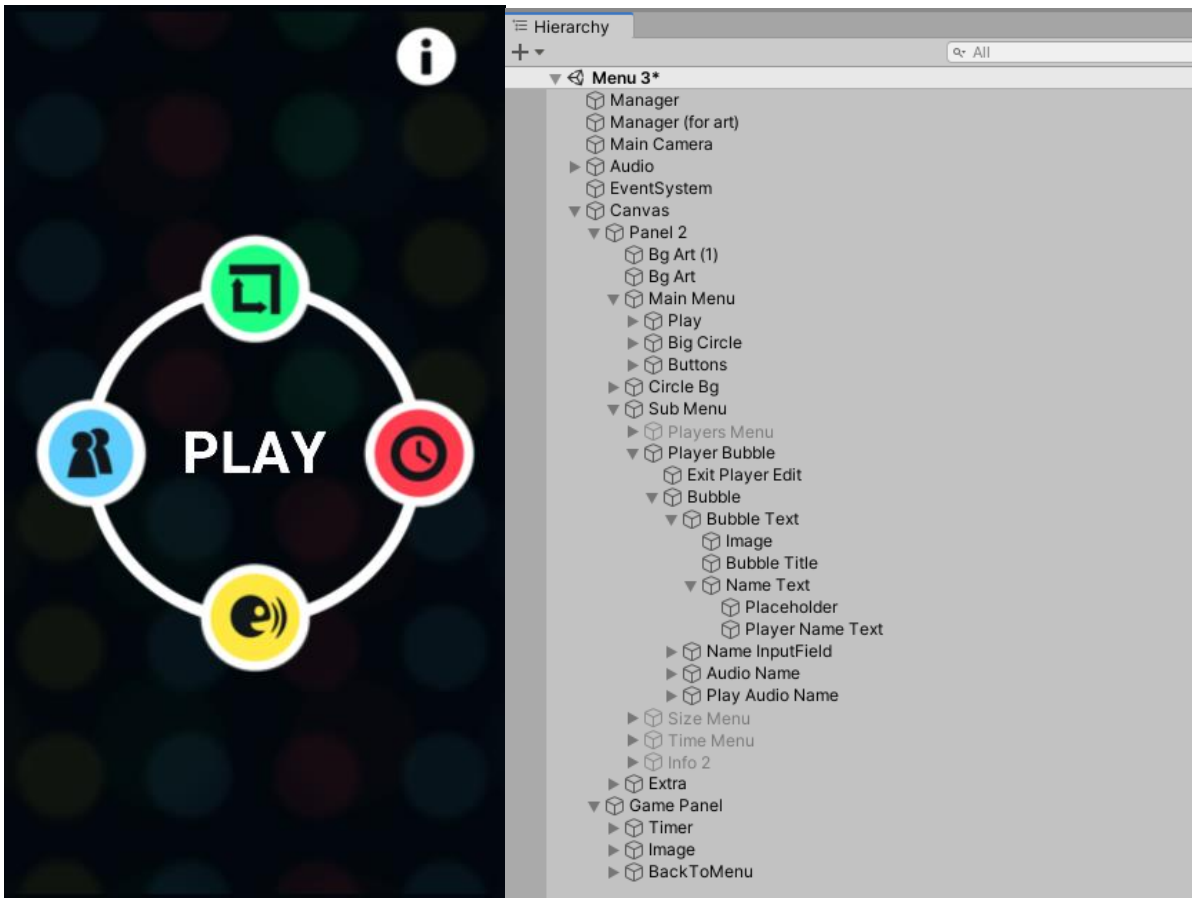
Slika 19: Odvajanje game asseta unutar Unity-a

U Unity-u se svi elementi moraju posebno označiti da bismo ih odvojili. To se radi u Sprite Editoru. Da bi se to moglo Sprite Mode u Inspector-u treba biti postavljen na Mulitple.



Slika 20: Import-ani asseti

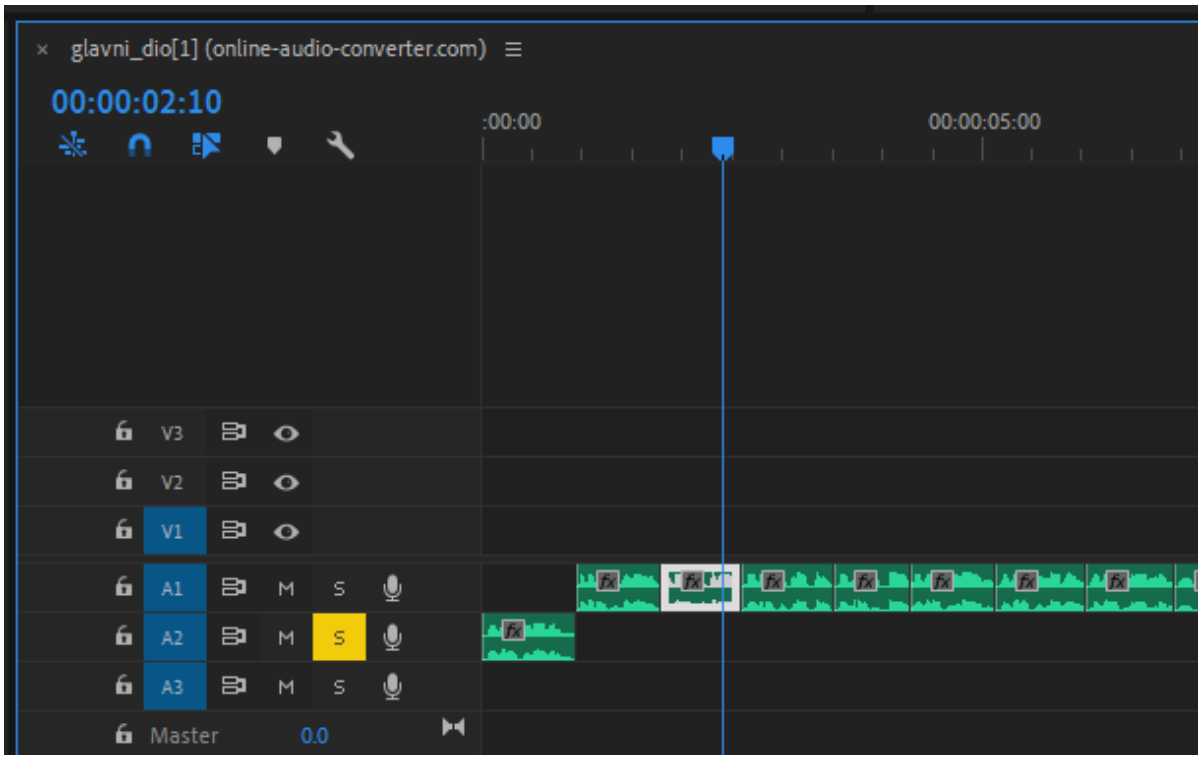
Kada se to napravi dobijemo pojedinačne slike koje možemo drag-n-drop-ati iz Project prozora direktno u Scene prozor ili u Hierarchy te ih tamo imenovati.



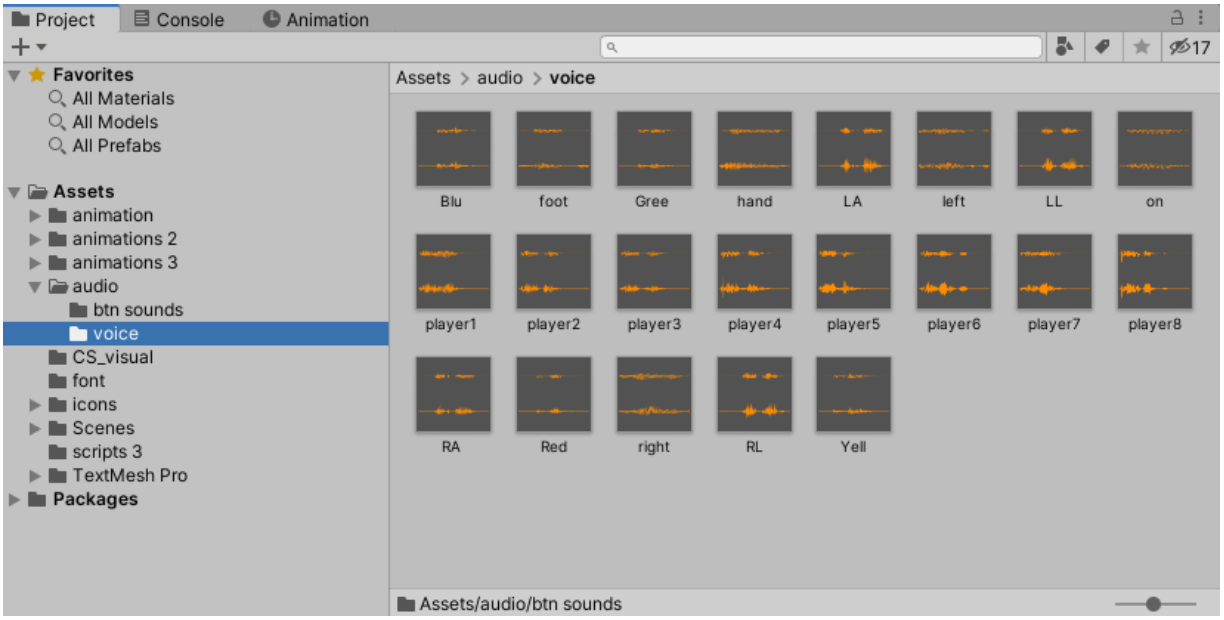
Slika 21: Usporedba jednostavnosti igre i kompleksnosti u Hierarchy-ji

Iako je aplikacija naizgled jednostavna, Hierarcy-ja je krcata raznim Asset-ima koji moraju postojati da bi individualne komponente aplikacije funkcionirale.

c. Stvaranje zvučnih datoteka



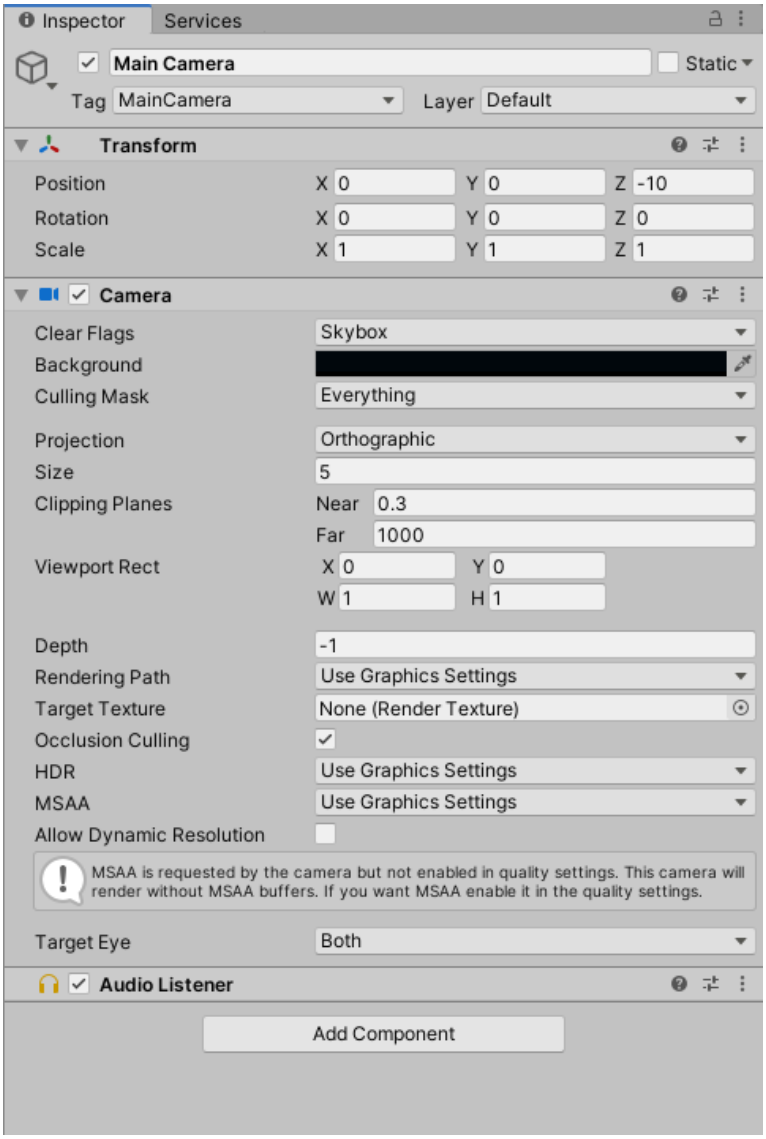
Slika 22: Izrezivanje zvučnih zapisa u Premiere-u Pro



Slika 23: Import-ani zvuk

Snimljeni su potrebni zvukovi pomoću snimača zvuka na mobilnom uređuju te eksport-ani u Premiere Pro. Tamo su pažljivo odvojeni i pojedinačno spremljeni u Unity u mp3 ili wav formatu.

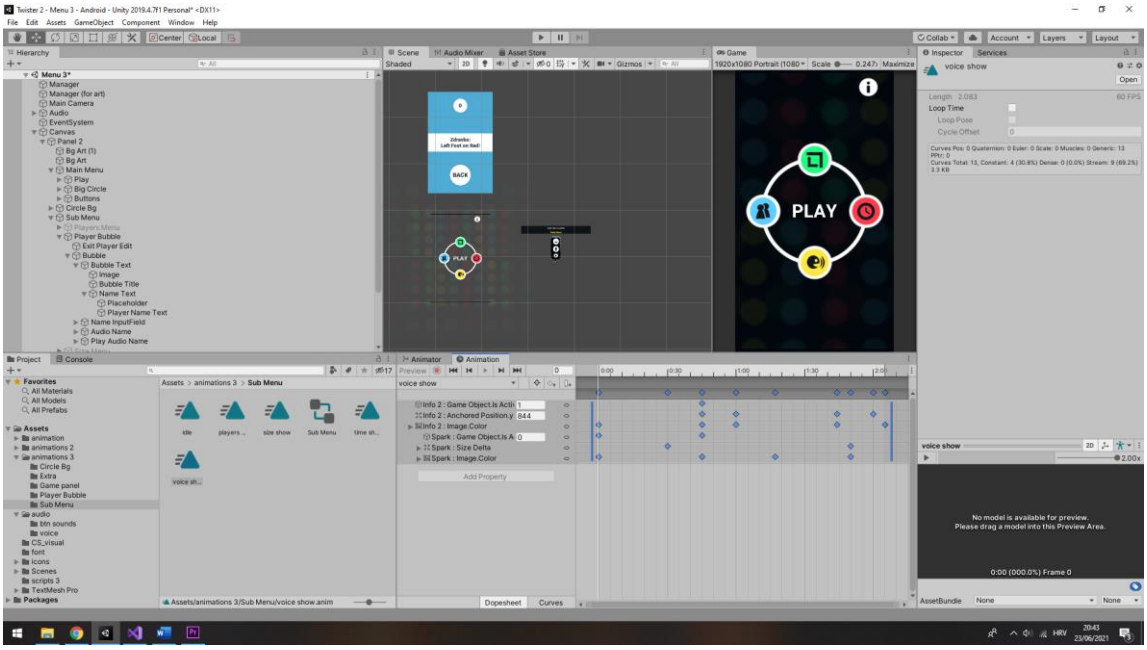
d. Kamera



Slika 24: Komponente kamere

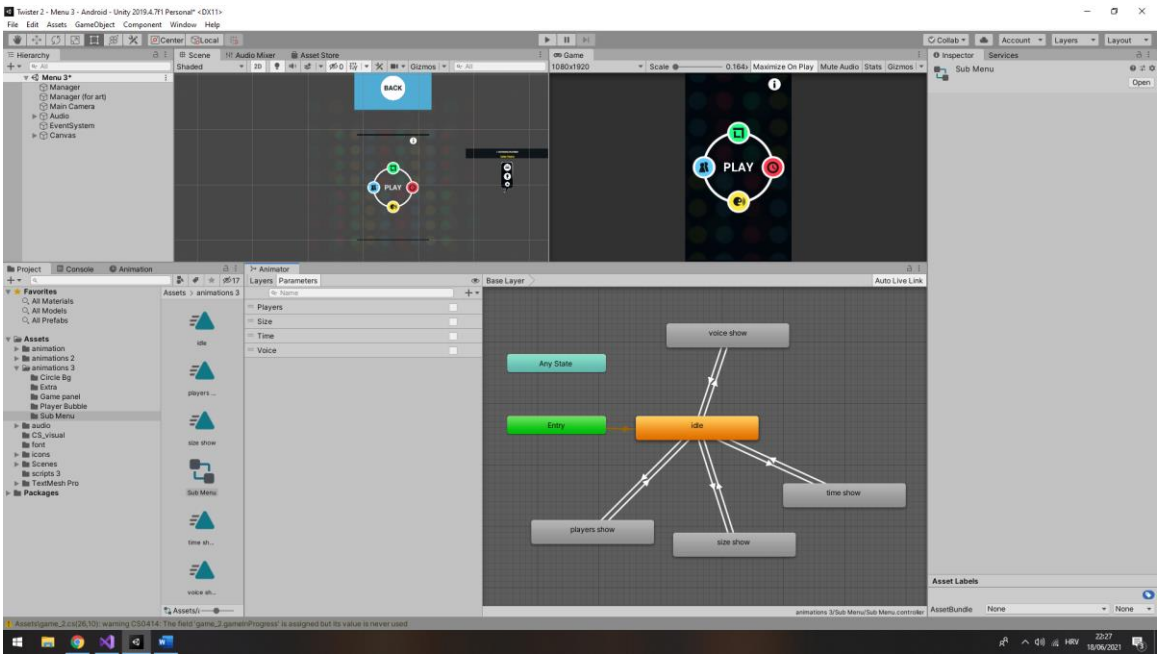
Dodana je Componnent-a kameri koja svaki frame renderira sve što je stavljeno u aplikaciju. Također sadrži Component-u Audio Listener koja sluša zvukove svih objekata te ih reproducira tijekom igranja igre.

e. Animator i Animacija



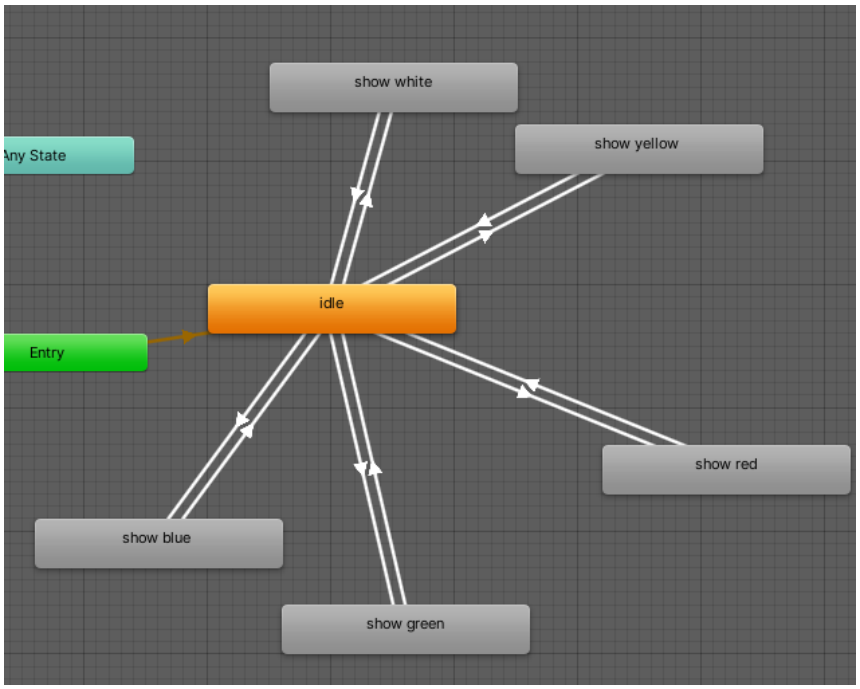
Slika 25: Primjer Animation prozora

Animiran je Menu koji se otvori pritiskom na jednu od 4 postavke. U svakom tom Menu-u animirano je mijenjanje stranica. Animiran je početak igre kada se pritisne tipka PLAY te mijenjanje naredbi unutar igre. Animirana je tipka „i“ koja sadrži informacije, te je također napravljena uočljivijom svaki put kada uđemo u svaki pod-izbornik.

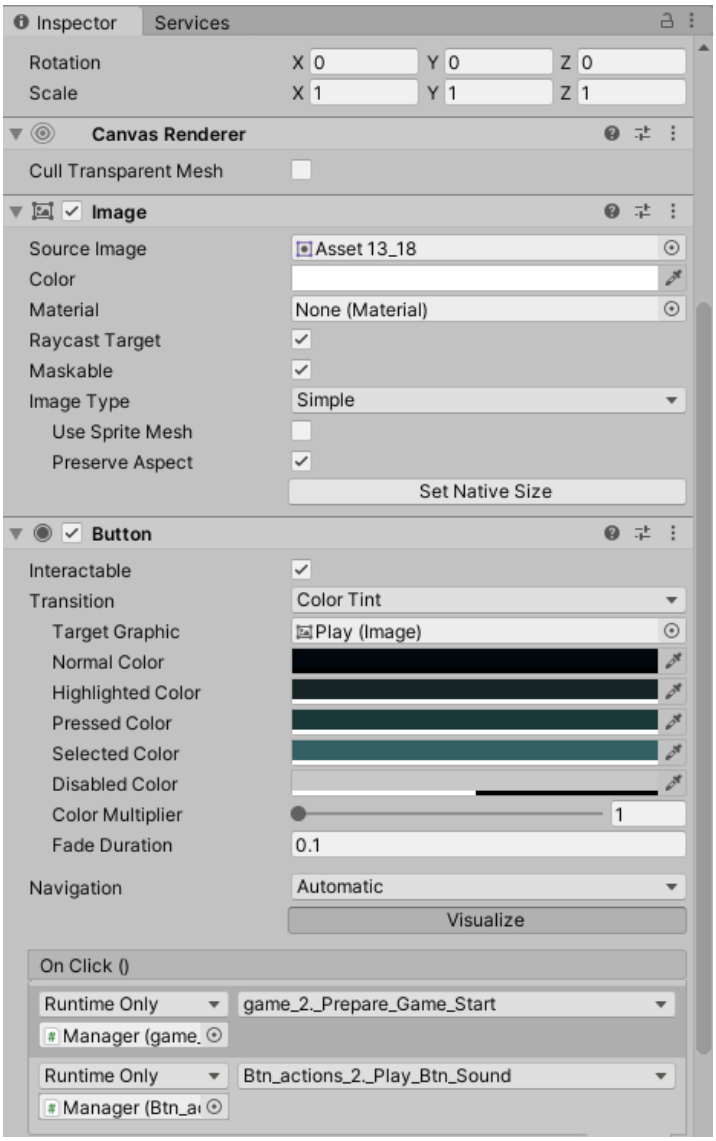


Slika 26: Animator prozor za sub-menu

Sve animacije spojene su Animator prozoru pomoću tranzicija.



Slika 27: Animator prozor za objašnjenja igre



Slika 28/: Button komponente

U Inspectoru se na tipke može dodati Button Component. Ona omogućuje da lagano implementiramo funkcionalnost tipkama. Ne treba pisati novi kod koji će prepoznati pritisak klikom miša ili pritisak prostom na ekran na pametnom mobitelu. Ovo pruža neka ograničenja, ali u puno slučajeva je sasvim dovoljno. Može se promijeniti boja i visibility na hover, pressed, disabled, … button. Također postoje funkcije koje možemo dodati na OnClick(). Tako će djelovati većina ove aplikacije. Jedna funkcija će odraditi animaciju otvaranja potrebnih komponenti dok će druga pustiti zvuk pritiska tipke bez koje aplikacija bila prazna.

f. Programiranje Glavnog Izbornika

```
/// <summary> "public" tags mean the var is for the "Game" script!
//main
public Animator animator_Circle_Bg;
public Animator animator_Sub_Menu;
public Animator player_Bubble;
public Animator animator_Extra;


//players
[HideInInspector]public int players_Index = 2;
public GameObject[] player_Icons;
public TextMeshProUGUI players_Text;
//playerdata
[HideInInspector] public int selectedPlayer;//public
[HideInInspector]public string[] stringNames = {"Player 1", "Player 2", "Player 3", "Player 4", "Player 5", "Player 6", "Player 7", "Player 8"};
public AudioClip audioName0, audioName1, audioName2, audioName3, audioName4, audioName5, audioName6, audioName7;
//player bubble
public TextMeshProUGUI playerBubbleTitle;
public AudioSource audioSource;
public AudioClip recordSound;

//size
[HideInInspector]public int selected_Size = 1;
public GameObject[] size_Icons;
public TextMeshProUGUI size_Text;

//time
[HideInInspector] public int selected_Time = 1;
public GameObject[] time_Icons;
public TextMeshProUGUI time_Text;

bool allowQuit;

public static Btn_actions_2 instance;

public AudioClip btnSound;
```

Slika 29: Deklaracija potrebnih varijabli

```
Unity Message | 0 references
private void Awake()
{
    if (instance == null)
        instance = this;
    else
        Destroy(gameObject);

    DontDestroyOnLoad(gameObject);
    return;
}
Unity Message | 0 references
```

Slika 30: Singleton pattern

Ovdje je napisan Singleton Pattern koji dopušta da objekt bude aktvian kroz sve scene. Ako bi se kasnije dodalo još scena objekt koji sadrži ovu skriptu bi ostao aktivan kroz sve level-e. DontDestroyOnLoad zaustavlja Unity da uništava objekt. Na vrhu funkcije provjeravamo ako

postoji još istih objekata jer ne želimo stvoriti 2 ista. Ako objekt nije null onda to znači da postoje 2 ista objekta te uništavamo duplikat.

```
private void Update()
{
    if (Input.GetKeyDown(KeyCode.Escape))...
}
0 references
IEnumerator Quit()
{
    allowQuit = true;
    yield return new WaitForSeconds(1);
    allowQuit = false;
}
```

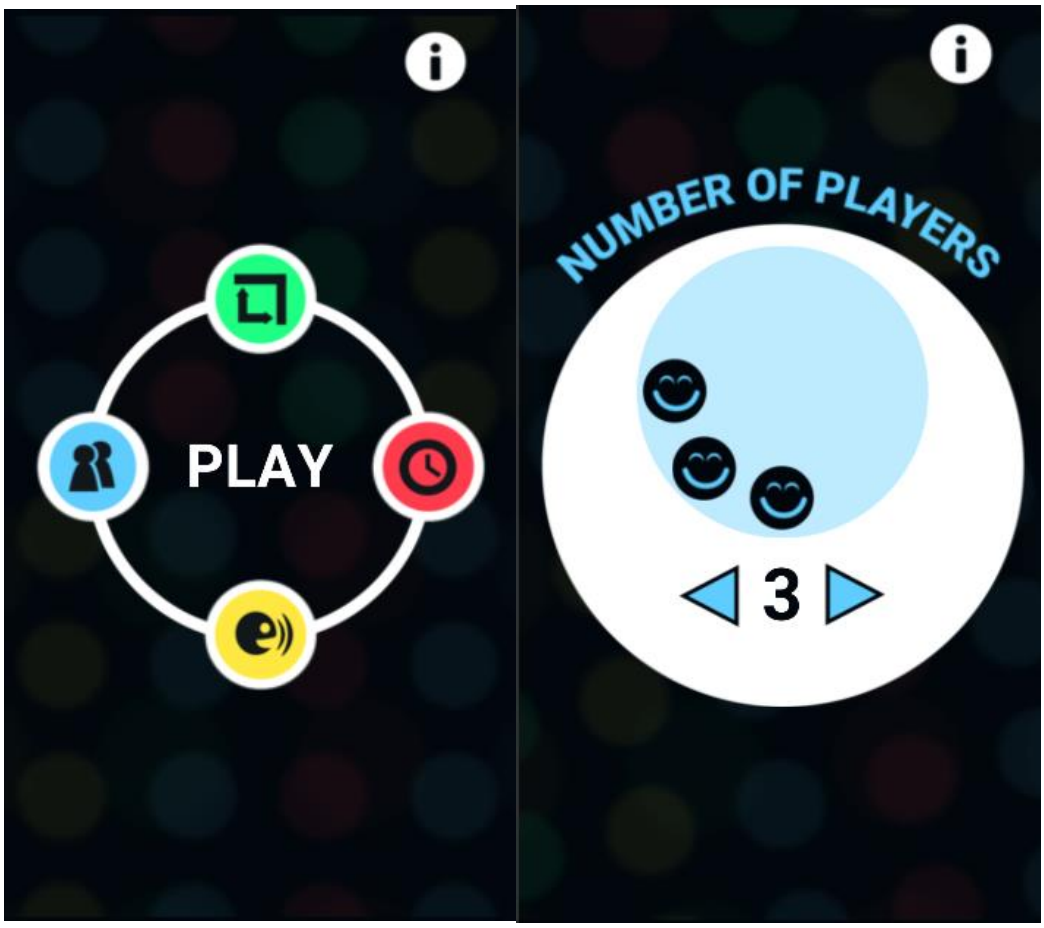
Slika 31: Izlaz iz igre

Napravljen je način za izlalzak iz aplikacije. Koristi se UnityEgine class Input koji sadrži funkciju GetKeyDown koja vraća true u frejmu u kojem je korisnik pritisnuo tipku koja je navedena u parametru. U ovom slučaju to je Esc tipka na tipkovnici, ali Esc također djeluje kada pritisnime „Natrag“ na mobitelu. Tipka je implementirana tako da ju treba pritisnuti 2 puta za redom unutar intervala od jedne sekunde. U suprotnom se neće aktivirati funkcionalnost.

```
0 references
void TrueArrayElementsRange(GameObject[] array)
{
    foreach (GameObject element in array)
        element.SetActive(false);
}
7 references
void TrueArrayElementsRange(GameObject[] array, int max)
{
    foreach (GameObject element in array)
        element.SetActive(false);
    for (int i = 0; i <= max; i++)
        array[i].SetActive(true);
}
6 references
void TrueArrayElementsRange(GameObject[] array, int min, int max)
{
    foreach (GameObject element in array)
        element.SetActive(false);
    for (int i = min; i <= max; i++)
        array[i].SetActive(true);
}
```

Slika 32: Funkcije za pretvaranje u odgovarajuću bool

Napisane su funkcije koje vraćaju bool vrijednost (true/false) elemenata niza prema potrebi. Kod unosa parametara koji se stavljaju nakon imena funkcije u zagradama se unosi varijabla niza, te 2 int-a koji predstavljaju prvi i posljednji bool u nizu koji će imati vrijednost true. Funkcija koristi foreach petlju koja ne postoji u C++-u. Ova petlja sama vidi koliko je elemenata u nizu te iterira onoliko puta koliko je elemenata. Svaki bool u array-u je pretvoren u false, te je kasnije pomoću klasične for petlje interval elemenata prema unosu pretvoren u true.



Slika 33: Sub-menu igrača

```
0 references
public void _Players()
{
    animator_Circle_Bg.SetBool("show", true);
    animator_Sub_Menu.SetBool("Players", true);
}
0 references
public void _Players_Change(string direction)
{
    if (direction == "right")
    {
        if (players_Index >= 2 && players_Index <= 7)
            players_Index++;
    }
    else if (direction == "left")
    {
        if (players_Index >= 3 && players_Index <= 8)
            players_Index--;
    }
    ////////////////////////
    if (players_Index == 2)
    {
        players_Text.text = "2".ToString();
        TrueArrayElementsRange(player_Icons,  1);
    }
    else if (players_Index == 3)
    {
        players_Text.text = "3".ToString();
        TrueArrayElementsRange(player_Icons, 2);
    }
    else if (players_Index == 4)...
    else if (players_Index == 5)...
    else if (players_Index == 6)...
    else if (players_Index == 7)...
    else if (players_Index == 8)...
}
```

Slika 34: Funkcija za kontrolu izbornika igrača

Pritiskom na tipku “Players” otvara se novi Menu. “animator_Circle_Bg” je ime skupine animacija koje su animirane u Animate prozoru te spojene u Animator prozoru. SetBool aktivira mijenja parametar u Animator-u. Kada je bool parametar promijenjen može se događa se tranzicija koja može biti puštena.

"_Players_Change" funkcija mijenja broj igrača ovisno o strelici koja je pritisnuta. Parametar "direction" provjerava koja je strelica pritisnuta. If uvjeti kontroliraju da tipka ne prekorači sadržaj koji je stvoren. "TrueArrayElementRange" je funkcija koja je već navedena te čini da se pojave ikone koje su potrebne.

Slika 35: Dodatna kontrola izbornika za igrače

```
// WHEN YOU PRESS EVRY PLAYER ICON
0 references
public void _Player_Select(int plIndex)
{
    selectedPlayer = plIndex;
    GameObject playerBubble = GameObject.Find("Player Bubble");
    RectTransform bubbleText = GameObject.Find("Bubble Text").GetComponent<RectTransform>();

    // Bubble pos
    playerBubble.transform.position = player_Icons[plIndex].transform.position;
    player_Bubble.SetBool("open", true);

    // Extreme text positions
    if (selectedPlayer == 3)
    {
        bubbleText.anchoredPosition = new Vector2(350, 135);
    }
    else if (selectedPlayer >= 4 && selectedPlayer <= 6)
    {
        bubbleText.anchoredPosition = new Vector2(-350, 212);
    }
    else...
    _Set_Name_Once();
}
```

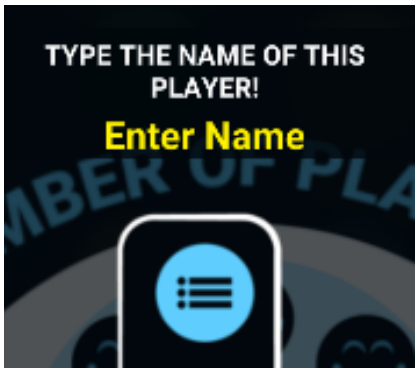
Slika 36: Funkcija za kontrolu dodatnog izbornika za igrače

„_Player_Select“ kako ime nalaže dopušta da odaberemo igrača. Pritiskom na ikonu ovara se još jedan sub-menu te možemo mu dodijeliti ime pismeno i zvučno. Pozicija novog menija nije uvijek idealna pa ju treba ručno podesti kada izlazi iz ekrana. To se može postići pomoću RectTransform.anchouredPostion kojem se dodijeli Vector 2 što su zapravo samo 2 broja. Imena po deafult-u će biti Player1, Player2,…tako da neće biti nužno unijeti sve podatke.

```
/// PLAYER BUBBLE
0 references
public void _Player_Settings(string setting)
{
    if (setting == "textName_type")
        _Save_Name_OnType();
    else if (setting == "voiceName")
        StartCoroutine("RecordNameSound");
    else if (setting == "playName")
        StartCoroutine("PlayNameSound");
    else print("Button is asigned wrong name!");
}
/// TYPE NAME
1 reference
public void _Set_Name_Once()
{
    playerBubbleTitle.text = "TYPE THE NAME OF THIS PLAYER!".ToString();

    TMP_InputField plName = GameObject.Find("Name InputField").GetComponent<TMP_InputField>();
    plName.text= stringNames[selectedPlayer].ToString();
}
1 reference
public void _Save_Name_OnType()
{
    TMP_InputField plName = GameObject.Find("Name InputField").GetComponent<TMP_InputField>();
    stringNames[selectedPlayer] = plName.text;
}
```

Slika 37: Funkcije za tipke na dodatno izborniku

„_Player_Settings“ funkcija provjerava koja tipka od 3 je pritisnuta (upisivanje imena, govor imena ili provjera točnog unosa imena) te poziva drugu odgovarajuću funkciju.

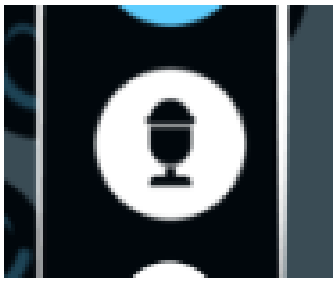


Slika 38: Tipka za tekstni unos imena

„_Set_Name_Once“ postavlja zadana imena, dok „_Save_Name_OnType“ sprema uneseno ime (string) u niz svaki put kada se dogodi promjena u prostoru za unos.

```
/// RECORD AUDIO
0 references
IEnumerator RecordNameSound()
{
    playerBubbleTitle.text = "SAY NAME AFTER FIRST 'BEEP' SOUND!".ToString();

    audioSource.PlayOneShot(recordSound);
    yield return new WaitForSeconds(.5f);
    AsignAudio();
    yield return new WaitForSeconds(1f);
    Microphone.End(null);
    audioSource.PlayOneShot(recordSound);
}
1 reference
void AsignAudio()
{
    if (selectedPlayer == 0)
        audioName0 = Microphone.Start(null, false, 1, 44100);
    if (selectedPlayer == 1)
        audioName1 = Microphone.Start(null, false, 1, 44100);
    if (selectedPlayer == 2)
        audioName2 = Microphone.Start(null, false, 1, 44100);
    if (selectedPlayer == 3)
        audioName3 = Microphone.Start(null, false, 1, 44100);
    if (selectedPlayer == 4)
        audioName4 = Microphone.Start(null, false, 1, 44100);
    if (selectedPlayer == 5)
        audioName5 = Microphone.Start(null, false, 1, 44100);
    if (selectedPlayer == 6)
        audioName6 = Microphone.Start(null, false, 1, 44100);
    if (selectedPlayer == 7)
        audioName7 = Microphone.Start(null, false, 1, 44100);
}
```

Slika 39: Funkcije za tipke na dodatno izborniku



Slika 40: Tipka za zvučni unos imena

„RecordNameSound“ se brine da korisnik glasovno snimi svoje ime. Tekst prvo ispiše što treba napraviti. Nakon toga pritiskom na tipku PlayOneShot (Unity AudioSource funkcija) pušta se zvuk tako da korisnik zna kada može početi govoriti. Tada pomoću Microphone.Start funkcije i deafult parametara program snima zvuk i sprema ga u AudioClip varijablu. Za svakog igrača treba zasebna varijabla, pa jer je maksimalan broj igrača 8, toliko je i varijabli. Na posljetku se još jedan put pusti zvuk nakon pola sekunde za kraj glasovnog unosa. Možda pola sekunde zvuči malo ali je najoptimalnije vrijeme za reći ime. I ako se ne izgovori u točnom vremenskom intervalu, odmah se može provjeriti i ispraviti.

```
/// PLAY AUDIO
0 references
IEnumerator PlayNameSound()
{
    playerBubbleTitle.text = "IF YOU RECORDED WELL, YOU SHOULD HEAR YOU'RE VOICE!".ToString();

    //audioSource.PlayOneShot(recordSound);
    yield return new WaitForSeconds(.3f);
    audioSource.PlayOneShot(GetAudio(selectedPlayer));
}
1 reference
AudioClip GetAudio(int sp)
{
    if (sp == 0)
        return audioName0;
    else if (sp == 1)
        return audioName1;
    else if (sp == 2)
        return audioName2;
    else if (sp == 3)
        return audioName3;
    else if (sp == 4)
        return audioName4;
    else if (sp == 5)
        return audioName5;
    else if (sp == 6)
        return audioName6;
    else if (sp == 7)
        return audioName7;
    else
    {
        print("no sounds found");
        return audioName0;
    }
}
```

Slika 41: Funkcija za puštanje snimljenog zvuka

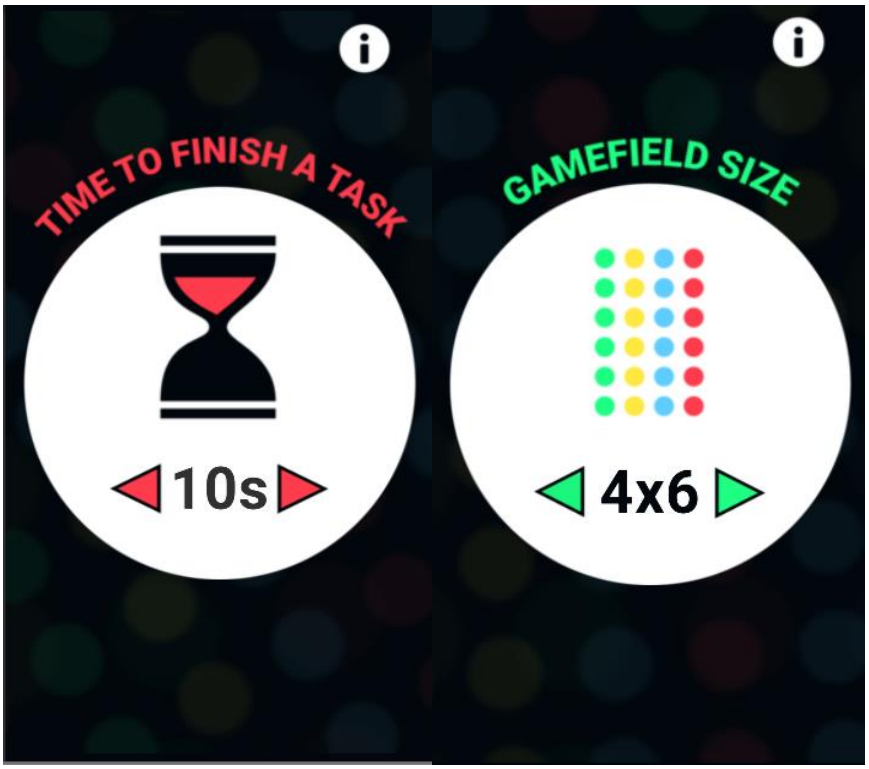
Slika 42: Tipka za puštanje snimljenog zvuka

„PlayNameSound“ pušta zvuk koji snimljen iz AudioClip varijable. Napisana je dodatna funkcija „GetAudio“ koja vraća AudioClip varijablu temeljenu na int vrijednosti.

```
//////////////////////////////////////////
0 references
public void _Size()
{
    animator_Circle_Bg.SetBool("show", true);
    animator_Sub_Menu.SetBool("Size", true);
}
0 references
public void _Size_Change(string direction)
{
    if (direction == "right")
    {
        if (selected_Size == 0 || selected_Size == 1)
            selected_Size++;
    }
    else if (direction == "left")
    {
        if (selected_Size == 1 || selected_Size == 2)
            selected_Size--;
    }
    if (selected_Size == 0)
    {
        size_Text.text = "4x4".ToString();
        TrueArrayElementsRange(size_Icons, 0, 0);
    }
    else if (selected_Size == 1)
    {
        size_Text.text = "4x6".ToString();
        TrueArrayElementsRange(size_Icons, 1, 1);
    }
    else if (selected_Size == 2)
    {
        size_Text.text = "8x8".ToString();
        TrueArrayElementsRange(size_Icons, 2, 2);
    }
}
```

```
//////////////////////////////////////////
0 references
public void _Time()
{
    animator_Circle_Bg.SetBool("show", true);
    animator_Sub_Menu.SetBool("Time", true);
}
0 references
public void _Time_Change(string direction)
{
    if (direction == "right")
    {
        if (selected_Time == 0 || selected_Time == 1)
            selected_Time++;
    }
    else if (direction == "left")
    {
        if (selected_Time == 1 || selected_Time == 2)
            selected_Time--;
    }
    if (selected_Time == 0)
    {
        time_Text.text = "8s".ToString();
        TrueArrayElementsRange(time_Icons, 0, 0);
    }
    else if (selected_Time == 1)
    {
        time_Text.text = "10s".ToString();
        TrueArrayElementsRange(time_Icons, 1, 1);
    }
    else if (selected_Time == 2)
    {
        time_Text.text = "15s".ToString();
        TrueArrayElementsRange(time_Icons, 2, 2);
    }
}
```

Slika 43: Funkcije za veličinu i vrijeme



Slika 44: Izbornici za vrijeme i veličinu igre

„_Size“ i „_Time“ funkcije rade na isti princip kao i prvi dio „Players“ funkcije. Pritiskom na strelice, događa se OnClick() event i pokreće se animacija prelaska na sljedeću stranicu. Također kod prati što smo odabrali pa se ti podatci mogu primijeniti na gameplay.

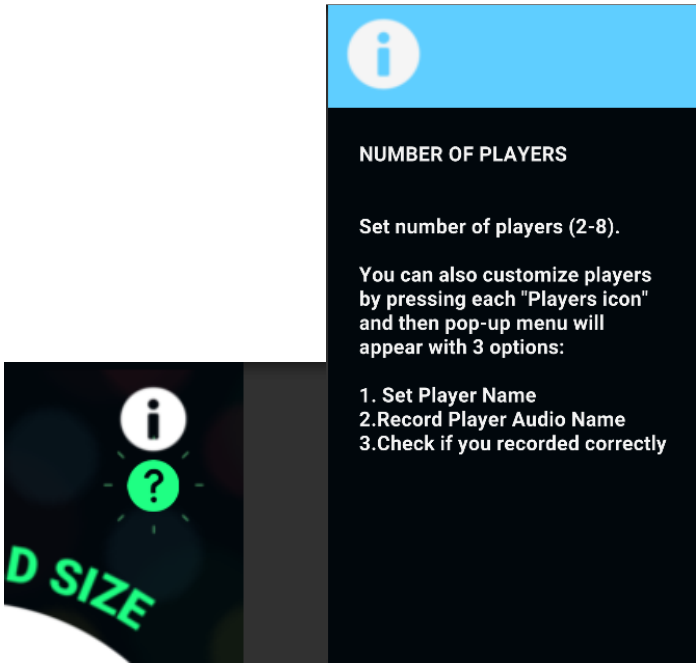
```
0 references
public void _Exit_Sub_Menu()
{
    animator_Circle_Bg.SetBool("show", false);

    animator_Sub_Menu.SetBool("Players", false);
    animator_Sub_Menu.SetBool("Size", false);
    animator_Sub_Menu.SetBool("Time", false);
    animator_Sub_Menu.SetBool("Voice", false);
}
0 references
public void _Exit_Player_Bubble()
{
    player_Bubble.SetBool("open", false);
}
```

Slika 45: Funkcionalnost za izlazak iz izbornika

„_Exit_Sub_Menu“ se izvršava pritiskom bilo gdje na ekran kada je otvoren sub-menu. On osigurava da se aplikacija vrati u glavni izbornik zatvarajući sve izbornike.

```
0 references
public void _Info()
{
    if (animator_Sub_Menu.GetBool("Players"))
        animator_Extra.SetBool("show blue", true);
    else if (animator_Sub_Menu.GetBool("Size"))
        animator_Extra.SetBool("show green", true);
    else if (animator_Sub_Menu.GetBool("Time"))
        animator_Extra.SetBool("show red", true);
    else if (animator_Sub_Menu.GetBool("Voice"))
        animator_Extra.SetBool("show yellow", true);
    else
        animator_Extra.SetBool("show white", true);
}
0 references
public void _Close_Info()
{
    animator_Extra.SetBool("show blue", false);
    animator_Extra.SetBool("show green", false);
    animator_Extra.SetBool("show red", false);
    animator_Extra.SetBool("show yellow", false);
    animator_Extra.SetBool("show white", false);
}
```

Slika 46: Funkcija za otvaranje informacija



Slika 47: Tipka s informacijama i info izbornik

Za one koji ne znaju igrati Twister ili im aplikacija nije jasna, napravljena su uputstva o načinu igranja igre. Ovisno gdje se korisnik nalazi ova tipka ima drugačiju funkcionalnost. Ako je u glavnom izborniku, pritiskom na „i“ tipku otvorit će se opće informacije o aplikaciji, a ako je u nekom sub-izborniku otvorit će se informacije o tom izborniku (npr. kako se koristiti „Players“ odjeljenjem.

```
0 references
public void _Play_Btn_Sound()
{
    audioSource.PlayOneShot(btnSound);
}
```

Slika 48: Funkcija za puštanje zvuka

„_Play_Btn_Sound“ se izvršava kada pritisnuta bilo koja tipka za bolje iskustvo uz zvukove.

```
0 references
public void _YouTube()
{
    Application.OpenURL("https://www.youtube.com/channel/UC9PXmtKPW2oDVPZ5Ve7uVlw/featured?view_as=subscriber");
}
```

Slika 49: Funkcija za link na YT

„_YouTube“ funkcija koristi Aplication.OpenURL Unity funkciju da bi korisnika odvela na goole link.

g. Programiranje igre

```
public Btn_actions_2 menuScript;
public TextMeshProUGUI commands;
public TextMeshProUGUI timeText;
public Animator playGame;
public GameObject bg;
public AudioSource audioPlayer;
private AudioClip nameClip0, nameClip1, nameClip2, nameClip3, nameClip4, nameClip5, nameClip6, nameClip7;
public AudioClip leftLeg, leftArm, rightLeg, rightArm;
public AudioClip blue, green, yellow, red;

int players;
string[] strPlNames;
int size;
int time;

int playerIndex;
bool gameInProgress = false;
int gameTimer;
```

Slika 50: Deklaracija varijabli

Napravljena je nova skripta koja će se kontrolirati igru da se ponaša prema postavkama koje su zadane. Menu skripti se može pristupiti upisivanjem njenoga imena kao tipa varijable. Ako se napravi public varijabla i skripta se spremi, u Unity Inspectoru se može drag-n-drop-ati skripta u varijablu. Deklarirane su potrebne varijable.

```
//prepare game
0 references
public void _Prepare_Game_Start()
{
    //player number
    players = menuScript.players_Index; //1-8 > 2 deafult
    //string player names
    strPlNames = menuScript.stringNames;
    //audio player names
    nameClip0 = menuScript.audioName0;
    nameClip1 = menuScript.audioName1;
    nameClip2 = menuScript.audioName2;
    nameClip3 = menuScript.audioName3;
    nameClip4 = menuScript.audioName4;
    nameClip5 = menuScript.audioName5;
    nameClip6 = menuScript.audioName6;
    nameClip7 = menuScript.audioName7;
    //size
    size = menuScript.selected_Size; // 0-2 > 1 default
    //time
    time = menuScript.selected_Time; // 0-2 > 1 default
    if (time == 0)
        time = 8;
    else if (time == 1)
        time = 10;
    else if (time == 2)
        time = 15;


    playGame.SetBool("open", true);
    gameInProgress = true;
    gameTimer = 0;
    InvokeRepeating("TimerText", 0, 1f);
    InvokeRepeating("GivePlayerComand", 0, time);
    audioPlayer.volume = 1;
}
```

Slika 51: Funkcija za početak igre

„_Preapare_Game_Start“ radi sve što treba biti učinjeno prije nego igra počne. Zadaje broj igrača (2 po deafult-u), sprema imena igrača u pismenom i zvučnom zapisu, sprema veličinu igre te vrijeme koje je potrebno da se napravi potez, započinje globalni timer, otvara Game UI i sve što je na njemu, te postavlja glasnoću.

```
0 references
public void _Back_To_Menu()
{
    playGame.SetBool("open", false);

    gameInProgress = false;
    playerIndex = 0;
    CancelInvoke("TimerText");
    CancelInvoke("GivePlayerComand");
    //this not work lol
    CancelInvoke("PlayVoice");
    CancelInvoke("ChangePlayerIndex");

    StartCoroutine(MuteAudioShortly());
}
1 reference
IEnumerator MuteAudioShortly()
{
    audioPlayer.volume = 0;
    yield return new WaitForSeconds(3f);
    audioPlayer.volume = 1;
}
```

Slika 52: Funkcija za povratak na glavni izbornik

„_Back_To_Menu“ vraća u glavni izbornik te resetira sve postavke koje trebaju biti resetirane kada igra počinje, ali ostavlja unesene postavke korisnika.

```
0 references
void TimerText()
{
    gameTimer++;
    timeText.text = gameTimer.ToString();
}
```

Slika 53: Funkcija za globalni tajmer

„TimerText“ se brine da trenutačno vrijeme od početka igre bude ispisano na ekran.

```
0 references
void GivePlayerComand()
{
    _bodyPart = RandomBodyPartText();
    _color = RandomColorText_and_ChangeColorBg();
    commands.text =   strPlNames[playerIndex].ToString() + ":\n" + _bodyPart + " on " + _color + "!";

    StartCoroutine (PlayVoice());
    StartCoroutine (ChangePlayerIndex());
}
```

Slika 54: Funkcija za ispis teksta naredbe

„GivePlayerComand“ izvrašava razne funkcije. Prvo varijablama daje podatke kao što su koji dio tijela i nasumična boja, te onda provjerava koji igrač je na redu te ispisuje cijelu komandu. Također su započete dvije Courutine-e koje dopuštaju da se lakše kontrolira vrijeme/delay, „PlayVoice“ i „ChangePlayerIndex“.

```
1 reference
IEnumerator PlayVoice()
{
    AudioClip[] nameClips = { nameClip0, nameClip1, nameClip2, nameClip3, nameClip4, nameClip5, nameClip6, nameClip7 };
    yield return null;

    for (int i=0; i<3; i++)
    {
        //names
        if (i == 0)
        {
            audioPlayer.clip = nameClips[playerIndex];
        }
        //body parts
        else if (i == 1)
        {
            if (_bodyPart == "Left Leg")
                audioPlayer.clip = leftLeg;
            else if (_bodyPart == "Left Arm")
                audioPlayer.clip = leftArm;
            if (_bodyPart == "Right Leg")
                audioPlayer.clip = rightLeg;
            else if (_bodyPart == "Right Arm")
                audioPlayer.clip = rightArm;
        }
        //color
        else if (i == 2)
        {
            if (_color == "Blue")
                audioPlayer.clip = blue;
            else if (_color == "Green")
                audioPlayer.clip = green;
            else if (_color == "Yellow")
                audioPlayer.clip = yellow;
            else if (_color == "Red")
                audioPlayer.clip = red;
        }

        //play
        audioPlayer.Play();

        //wait to stop playing
        while (audioPlayer.isPlaying)
            yield return null;
    }
}
```

Slika 55: Funkcija koja slaže glasovnu rečenicu i poziva funkciju Play

„PlayVoice“ funkcija slaže AudioClip temeljen na imenu igrača koji je na redu, te nasumičnom odabiru dijela tijela i boje koji će biti napravljen. Pomoću AuidoSource.Play( ) Unity funkcije pušta clip pazeći da se glasovi ne poklapaju.

```
1 reference
IEnumerator ChangePlayerIndex()
{
    yield return new WaitForSeconds(3f);
    if (playerIndex < players - 1)
        playerIndex++;
    else
        playerIndex = 0;
}
```

Slika 56: Funkcija za promjenu Indexa igrača

„Change Player Indeks“ mijenja igrača koji je na redu.

```
1 reference
string RandomBodyPartText()
{
    int rand = Random.Range(1, 5);
    if (rand == 1)
        return "Left Leg";
    else if (rand == 2)
        return "Left Arm";
    else if (rand == 3)
        return "Right Leg";
    else if (rand == 4)
        return "Right Arm";
    return "error";
}
```

Slika 57: Funkcija za nasumični ud

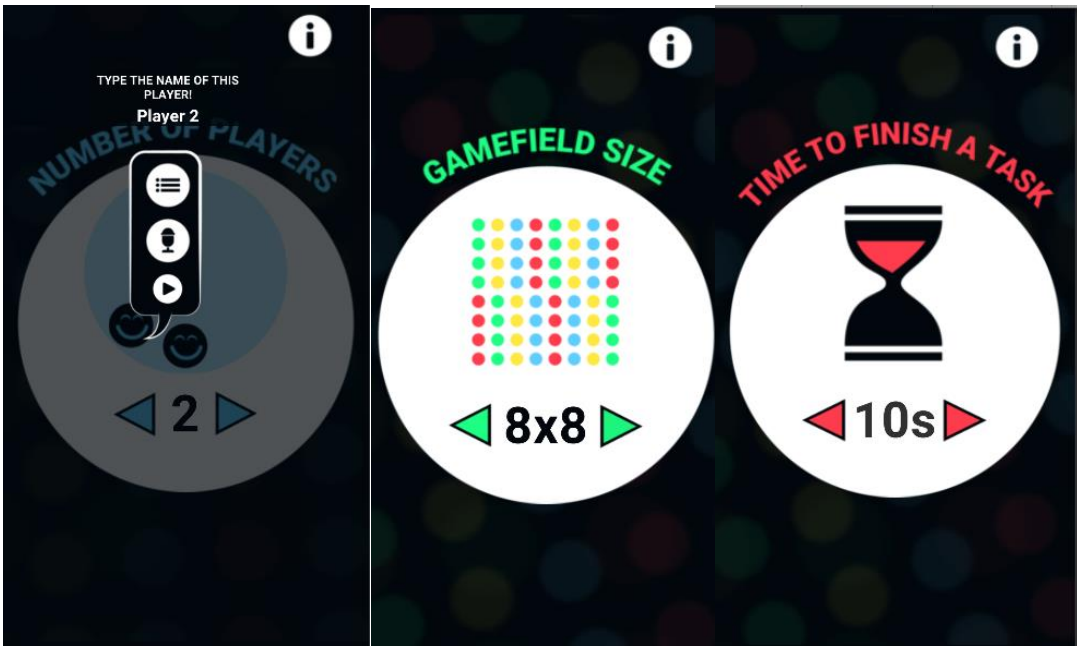
Funkcija „RandomBodyPartText“ vraća nasumični string dijela tijela.

```
1 reference
string RandomColorText_and_ChangeColorBg()
{
    int rand = Random.Range(1, 5);
    if (rand == 1)
    {
        bg.GetComponent<Image>().color = new Color(0.372549f, 0.8078432f, 1);
        return "Blue";
    }
    else if (rand == 2)
    {
        bg.GetComponent<Image>().color = new Color(0.1215686f, 1, 0.5137255f);
        return "Green";
    }
    else if (rand == 3)
    {
        bg.GetComponent<Image>().color = new Color(1, 0.9098039f, 0.254902f);
        return "Yellow";
    }
    else if (rand == 4)
    {
        bg.GetComponent<Image>().color = new Color(1, 0.2352941f, 0.3019608f);
        return "Red";
    }
    return "color error";
}
```

Slika 58: Funkcija za nasumičnu boju

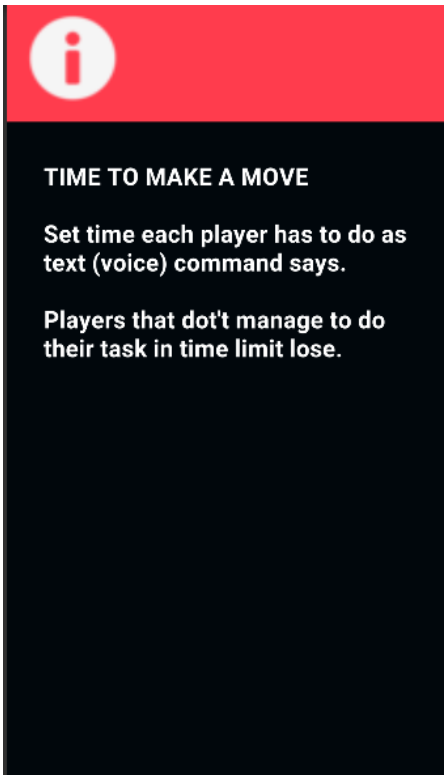
Funkcija „RandomColorText_and_ChangeColorBg“ vraća nasumični string boje te postavlja pozadinu pomoću Image.color u boju koja je nasumično odabrana tako da se pozadina poklapa s izabranom bojom.

h. Slike završene aplikacije



Slika 59: Postavke igre

Slika 60: Izgled igre



Slika 61: Tutorial za igru

10. Zaključak

U današnjem svijetu je nemoguće zamisliti život bez aplikacija koje se koriste na dnevnoj bazi. Izrada aplikacija ima puno potencijala, pa time i postoji puno izbora alata kojima želimo izraditi te aplikacije. Unity game engine je jedan od tih alata. Iako nije najbolji izbor za izradu uobičajenih aplikacija jer njegova funkcionalnost teži izradi igara, Unity je bez sumnje go-to razvojno okruženje za indie i mobilne igre. Ima intuitivan interface, programski kod i razvijen priručnik, te razne druge korisne alate poput Animator-a. Unity-eva Personal licenca dopušta stvaranje komercijalne igre dok god zarada ne prijeđe određenu količinu novca. Pruža mogućnost kreacije 2D. 3D, VR i AR sadržaja te svo znanje stečeno radeći ovaj projekt može biti preneseno na druga mjesta koja su u skladu s stvarima u kojima je engine najbolji.

U Varaždinu, _____ _____

Potpis studenta

11. Literatura

[1] The Best Tools for Android Software Development https://ncube.com/blog/the-best-tools-for-android-software-development

[2] Statista https://www.statista.com/aboutus/

[3] Uvod https://careerkarma.com/careers/mobile-development/

[4] History and comparative study of modern game engines https://www.researchgate.net/publication/259496289_History_and_comparative_study_of_modern_game_engines

[5] Tutorialspoint, unity_tutorial https://www.tutorialspoint.com/unity/unity_tutorial.pdf

[6] Forum zaš unity ne valja za android aps https://stackoverflow.com/questions/63687616/is-it-better-to-develop-app-with-unity-or-android-studio/63691263

[7] The Ultimate Beginners Guide To Game Development In Unity https://www.freecodecamp.org/news/the-ultimate-beginners-guide-to-game-development-in-unity-f9bfe972c2b5/

[8] Animacija https://docs.unity3d.com/Manual/AnimationOverview.html

[9] Creating and Using Scripts https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html

[10] Introduction to Functions in C# https://www.thoughtco.com/introduction-to-functions-in-c-958367

[11] Implementing the Singleton Pattern in C# https://csharpindepth.com/articles/singleton

[12] Input https://unity.com/features/input-system

[13] Audio source https://gamedevbeginner.com/how-to-play-audio-in-unity-with-examples/

[14] Corutines https://gamedevbeginner.com/coroutines-in-unity-when-and-how-to-use-them/

[15] https://www.gamesindustry.biz/articles/2020-01-16-what-is-the-best-game-engine-is-unity-the-right-game-engine-for-you

12. Popis slika

Sveučilište Sjever

SVEUČILIŠTE SJEVER

IZJAVA O AUTORSTVU
I
SUGLASNOST ZA JAVNU OBJAVU

Završni/diplomski rad isključivo je autorsko djelo studenta koji je isti izradio 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, izvora s interneta, i drugih izvora) bez navođenja izvora i autora navedenih radova. Svi dijelovi tuđih radova moraju biti pravilno navedeni i citirani. Dijelovi tuđih radova koji nisu pravilno citirani, smatraju se plagijatom, odnosno nezakonitim prisvajanjem tuđeg znanstvenog ili stručnoga rada. Sukladno navedenom studenti su dužni potpisati izjavu o autorstvu rada.

Ja, ZDRAVKO MALIĆ *(ime i prezime)* pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor/ica završnog/diplomskog (*obrisati nepotrebno*) rada pod naslovom Izrada 2D video u Unity-u *(upisati naslov)* te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

Student/ica:
(upisati ime i prezime)

Malić
(vlastoručni potpis)

Sukladno Zakonu o znanstvenoj djelatnost i visokom obrazovanju završne/diplomske radove sveučilišta su dužna trajno objaviti na javnoj internetskoj bazi sveučilišne knjižnice u sastavu sveučilišta te kopirati u javnu internetsku bazu završnih/diplomskih radova Nacionalne i sveučilišne knjižnice. Završni radovi istovrsnih umjetničkih studija koji se realiziraju kroz umjetnička ostvarenja objavljuju se na odgovarajući način.

Ja, ZDRAVKO MALIĆ *(ime i prezime)* neopozivo izjavljujem da sam suglasan/na s javnom objavom završnog/diplomskog (*obrisati nepotrebno*) rada pod naslovom Izrada 2D video igre u Unityu *(upisati naslov)* čiji sam autor/ica.

Student/ica:
(upisati ime i prezime)

Malić
(vlastoručni potpis)