Izrada računalne igre u tri dimenzije sa zagonetkama

Hrvoje, Hodak

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Hrvoje Hodak

IZRADA RAČUNALNE IGRE U TRI DIMENZIJE SA ZAGONETKAMA

ZAVRŠNI RAD

Varaždin, 2018.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Hrvoje Hodak

Matični broj: 43952/15–R

Studij: Informacijski sustavi

IZRADA RAČUNALNE IGRE U TRI DIMENZIJE SA ZAGONETKAMA

ZAVRŠNI RAD

Mentor:
Prof. dr. sc. Danijel Radošević

Varaždin, ožujak 2018.

Sadržaj

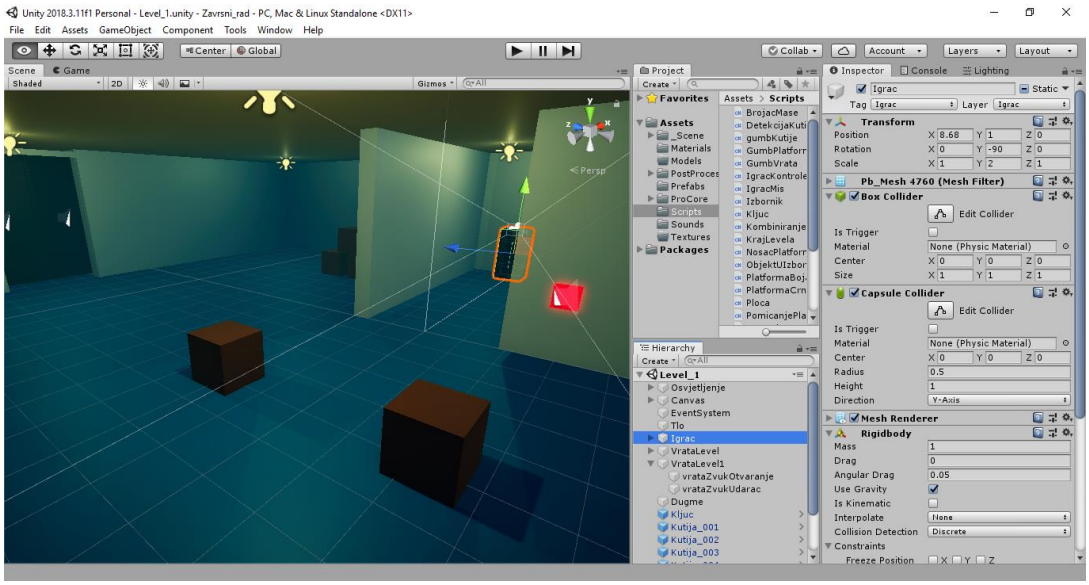
1. Uvod 1
2. Korišteni alati 2
2.1. Unity 2
2.2. Visual Studio 5
3. Opis mehanika 6
3.1. Inspektor 6
3.1.1. Preinaka 6
3.1.2. Kruto tijelo 7
3.1.3. Sudarač 7
3.1.4. Izvor zvuka 8
3.1.5. Programski kod (skripta) 9
3.2. Programski kodovi 10
3.2.1. IgracKontrole.cs 11
3.2.2. IgracMis.cs 13
3.2.3. UzmiKutiju.cs 14
3.2.4. VrataLevel.cs 15
4. Tijek igre 17
5. Zaključak 22
6. Popis izvora 23
7. Popis slika 24
8. Popis tablica 24

1. Uvod

Ubrzanim razvojem računala, igraćih konzola i mobilnih uređaja, povećao se i razvoj videoigara koje se koriste na tim platformama. Zbog velike potražnje, industrija videoigrara danas je jedna od najvećih industrija na svijetu. Videoigre postaju sve složenije i različitije pa tako imamo mnogo različitih vrsta igara kao što su akcijske, avanturističke, simulacije, strategije, igre uloga, igre zagonetki i slično, a osim toga, svaka vrsta ima i svoje podvrste i podjele, ovisno o tipu.

Ideja za ovaj projekt inspirirana je uspješno izdanim igrama zagonetke (eng. Puzzle Games) kao što su The Talos Principle hrvatskog tima programera videoigara Croteam (croteam.com, 2014.), Portal 2 američkog tima Valve (thinkwithportals.com, 2011.) te igra danskog razvojnog tima Playdead zvana Limbo (playdead.com, 2019).

U ovom radu, opisat ćemo izradu jedne igre zagonetke koja će biti napravljena u alatu Unity. Cilj igre je rješavanje određenih zagonetki kako bi mogli prijeći na sljedeću razinu. Zagonetke koje igrač rješava razlikuju se na svakoj razini. Također, svaka razina razlikuje se od ostalih razina u nekom specifičnom elementu.



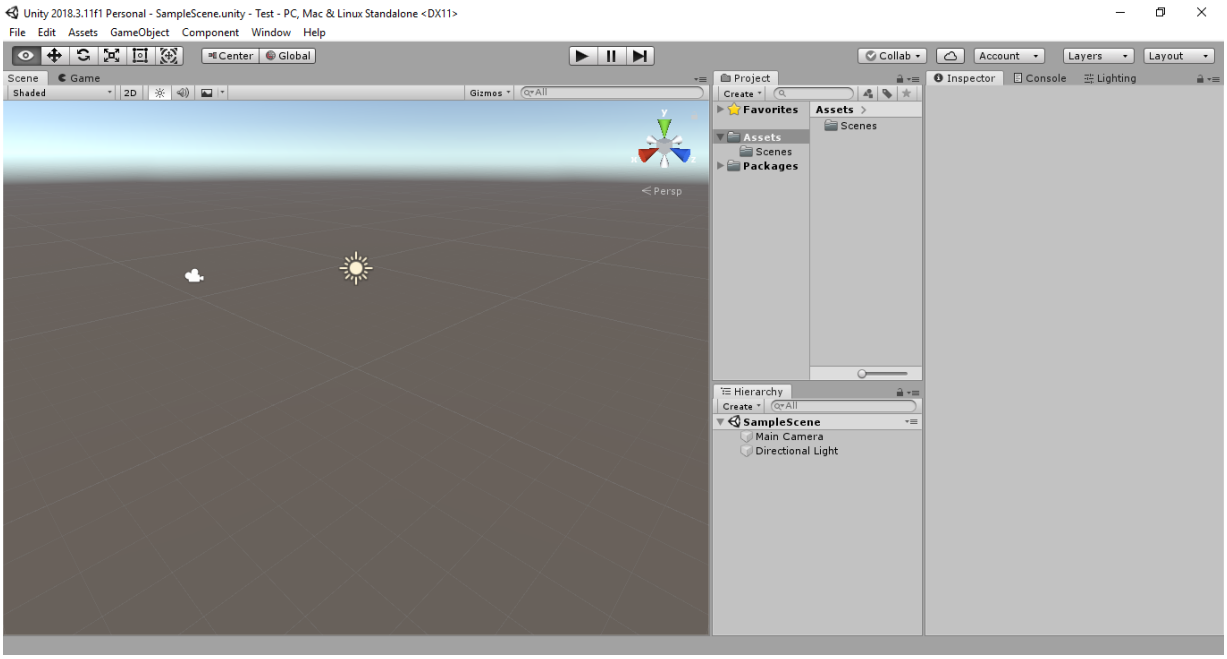
Slika 1: Prikaz izrade videoigre u tri dimenzije u alatu Unity (slika autora, 2019.)

2. Korišteni alati

Najčešće korišteni alati za izradu ove videoigre su Unity i Visual Studio. Unity je platforma koja se koristi za razvoj 2D i 3D igara kao i igara sa simulacijom virtualne stvarnosti, ali osim toga, često se koristi i u transportnim, automobilskim, filmskim i mnogim drugim insustrijama. Visual Studio je razvojno okruženje koje se koristi za razvoj računalnih, mobilnih i web aplikacija i servisa.

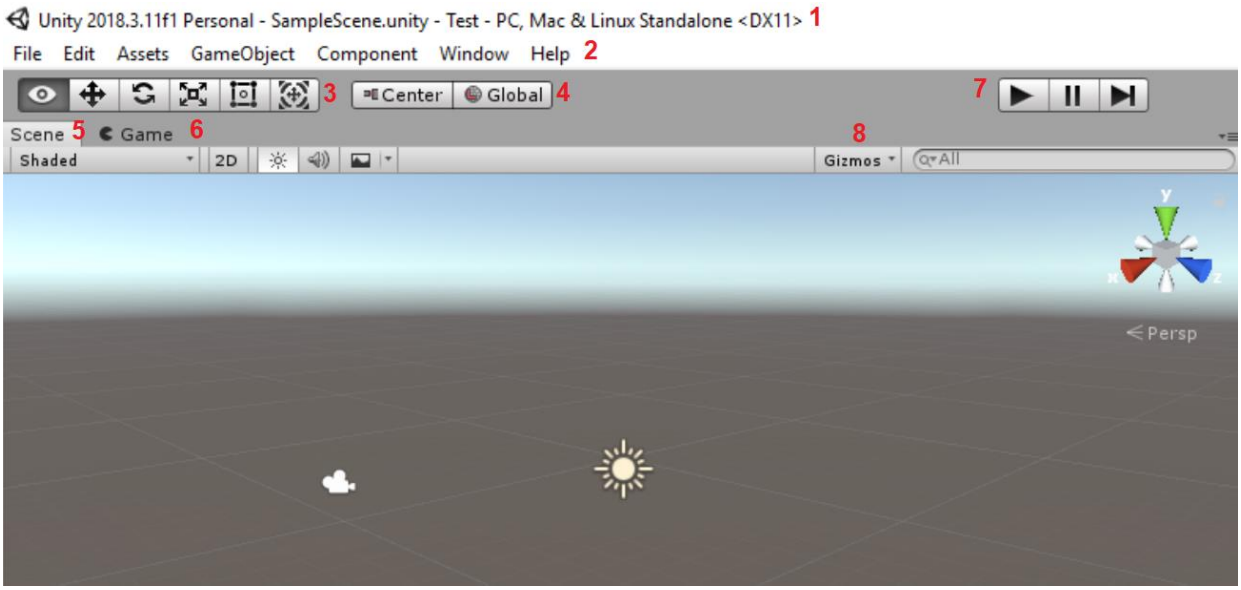
2.1. Unity

Unity je platforma u starnom vremenu (svaka promjena koju korisnik napravi je odmah vidljiva) koja se koristi za razvoj videoigara, ali i u mnogim drugim insustrijama (unity3d.com, 2019.). Za izradu ove igre korištena je besplatna licenca koju je moguće dobiti izradom računa na internetskoj stranici Unity-a. Verzija platforme koja je korištena prilikom razvoja je 2018.3.11f1 .



Slika 2: Početni prikaz u alatu Unity prilikom otvaranja novog praznog projekta (slika autora, 2019.)

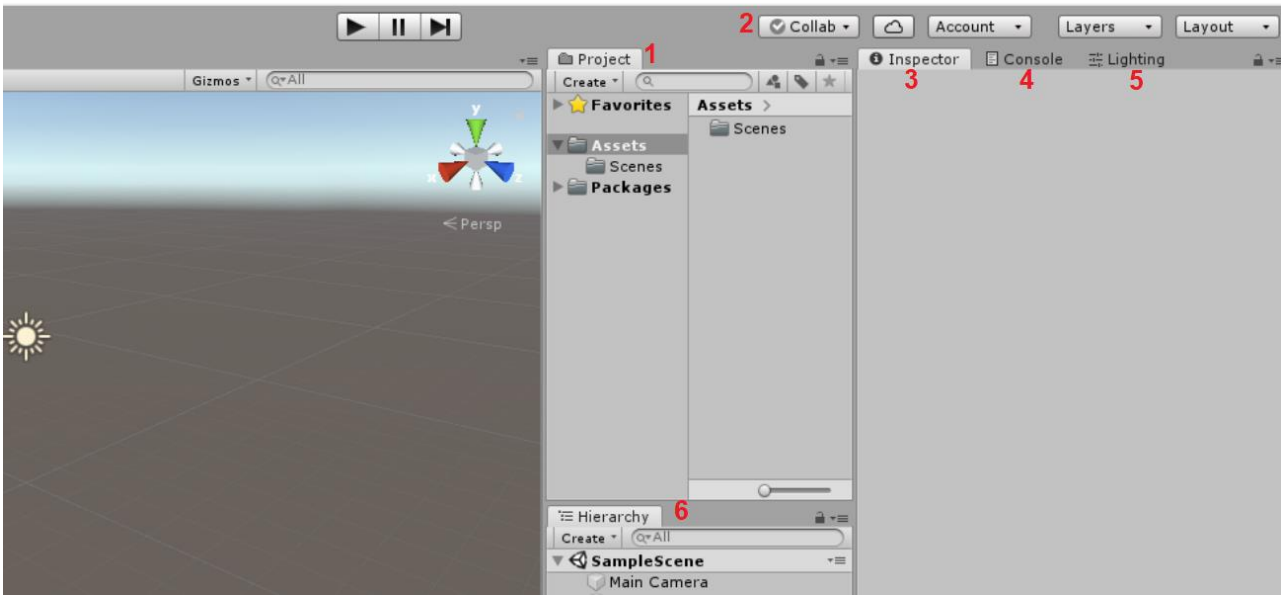
Budući da Unity ima veliki broj različitih opcija i mogućnosti, opisat ćemo one osnovne, odnosno one koje smo koristili za potrebe ove igre.



Slika 3: Detaljniji prikaz praznog projekta u alatu Unity (1) (slika autora, 2019.)

Sa Slike 3 prikazane iznad, izdvojit ćemo sljedeće objekte:

1) Verzija Unity-a, naziv scene, naziv projekta te platforme koje podržavaju ovaj projekt
2) Alatna traka za biranje opcija za rad s trenutnom datotekom, uređivanje sadržaja, dodavanje sredstava, objekata u igri, pomoćnih komponenata, pomoćnih prozora te otvaranje pomoći za korištenje Unity-a
3) Gumbi za pomoć pri radu s objektima (kretanje kroz scenu, pomicanje objekta, rotiranje objekta, promjena dimenzija objekta, promjena dimenzija dvodimenzionalnog objekta te višefunkcionalni objekt)
4) Promjena izvora rotacije (rotiranje oko trenutno odabranog objekta, rotiranje u sceni)
5) Scena (eng. Scene) – trenutni prikaz razine u kojoj se korisnik nalazi. Ovdje korisnik dodaje objekte, pomiče ih, rotira ili im mijenja veličinu
6) Igra (eng. Game) – prikaz kako trenutna razina izgleda iz očiju igrača kada je razina izrađena
7) Gumbi za pokretanje igre u uređivaču (igra se pokreće u istom prozoru kao i uređivač, u alatnom protoru „Igra“), pauziranje igre te prebacivanje igre u sljedeći okvir (eng. Frame)
8) Naprave (eng. Gizmos) – izbornik u kojem korisnik bira koje objekte želi vidjeti na sceni (fizički objekti, zvukovi, kamere, čestice, animacije, sudarače i slično)



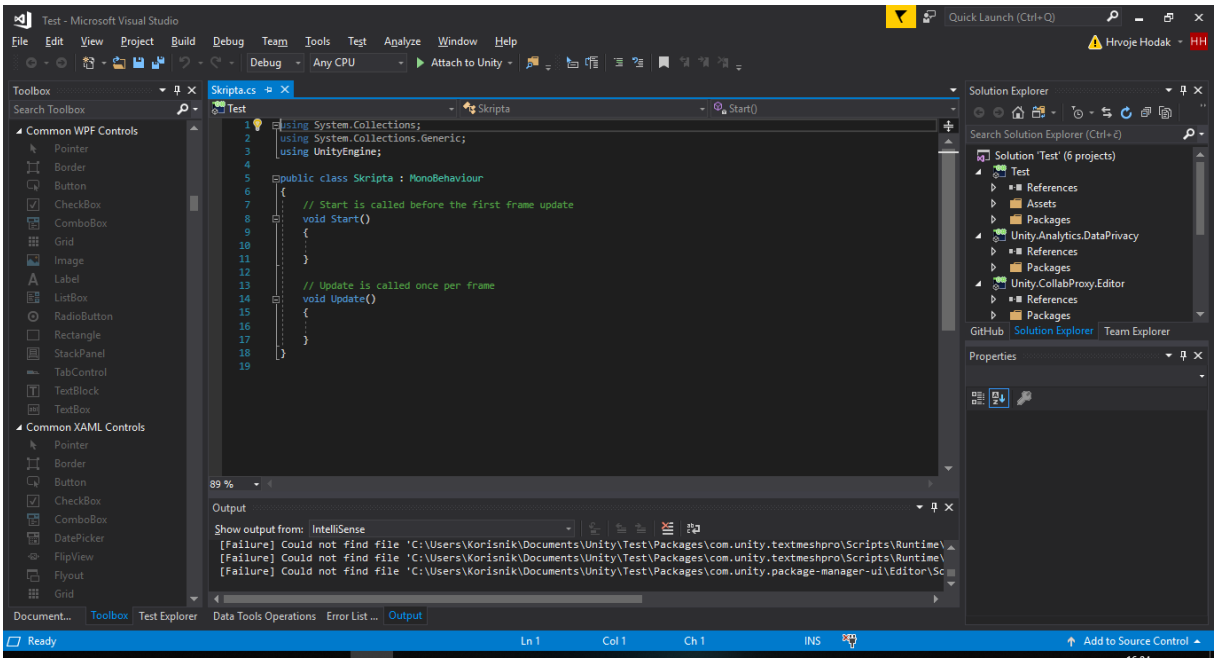
Slika 4: Detaljniji prikaz praznog projekta u alatu Unity (2) (slika autora, 2019.)

Sa Slike 4 prikazane iznad izdvojit ćemo sljedeće objekte:

1) Projektna mapa – prikaz svih datoteka povezanih s trenutnim projektom kako bi korisniku bilo lakše dodavati i mijenjati objekte te kako bi pristup različitim pomoćnim objektima bio organiziraniji. Svi objekti prikazuju se onako kako se nalaze u mapama spremljenim na računalu
2) Gumbi za pomoć prilikom surađivanja na projektu s drugim ljudima ili dohvaćanje drugih projekata s drugog servera
3) Inspektor (eng. Inspector) – u ovom prozoru prikazuju se sve detaljnosti svakog odabranog objekta na sceni (atributi i funkcionalnosti)
4) Konzolni prikaz (eng. Console) – prozor za prikaz konzolnog ispisa koji može biti od velike pomoći prilikom uređivanje projekta (ne vidi se kad je projekt gotov i isporučen)
5) Osvjetljenja (eng. Lighting) – prozor s opcionalnostima za osvjetljenja kako bi igra izgledala bolje i stvarnije
6) Hijerarhija (eng. Hierarchy) – ovdje su prikazani svi objekti koji se trenutno nalaze na sceni bez obzira bili oni akrivni ili ne (mogu li se vidjeti na sceni). Ovdje također možemo dodati i nove objekte pritiskom na padajući izbornik „Create“

2.2. Visual Studio

Visual Studio je razvojno okruženje u kojem se pišu programski kodovi za računalne, mobile i web programe. Za izradu ove igre korištena je besplatna licenca dobivena od Fakulteta organizacije i informatike. Tijekom razvoja korišten je Visual Studio 2017. Budući da u Unity-u možemo samo čitati skripte, odnosno programske kodove (eng. Script), potreban nam je vanjski uređivač koda pa će tako, ako želimo promijeniti programski kod, Unity automatski pokrenuti Visual Studio kao predefinirani program, osim ako korisnik ne odabere drugačije.



Slika 5: Prikaz predefiniranog koda u alatu Visual Studio 2017 koji se generira kada napravimo novu skriptu u Unity-u (slika autora, 2019.)

Prilikom stvaranje nove skripte, automatski se dodaje biblioteka *UnityEngine* koja sadrži mnoge funkcionalnosti za rad s objektima u Unity-u. Nova klasa koja je napravljena, automatski nasljeđuje klasu *MonoBehaviour*. Nova klasa dolazi s automatski izgenerirane dvije metode: Start i Update. Start je metoda koja nema povratnu vrijednost, a poziva se samo jednom, i to u trenutnku kad je objekt omogućen, odnosno inicijaliziran je na sceni. Update je također metoda koja nema povratnu vrijednost, a poziva se prilikom prikaza svakog okvira što, ovisno o računalu, može biti i do nekoliko desetaka puta u jednoj sekundi.

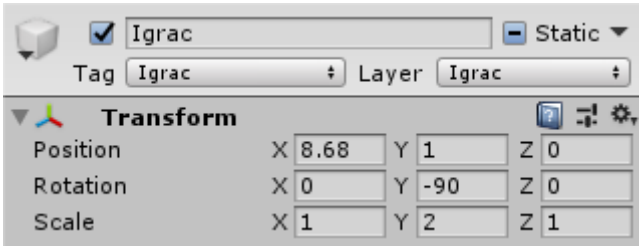
3. Opis mehanika

Trodimenzionalni objekti u Unity-u sadrže mnogo različitih komponenti koje igračima igru približuju stvarnom svijetu. Svaki objekt može biti drugačije pozicioniran na sceni, imati svoj zvuk, reagirati u doticaju s drugim objektima ili se ponašati prema određenim zakonima fizike. Također, dodavanjem programskog koda na svaki objekt, sam korisnik može kontrolirati određene događaje i akcije za svaki objekt. Osim toga, u Unity-u postoji i mogućnost izrade predložaka (eng. Prefab) kojima korisnik jednom napravljeni objekt može spremiti i koristiti ga više puta na sceni ili u cijeloj igri bez potrebe da se cijeli objekt ponovno pravi ispočetka.

3.1. Inspektor

Svaku komponentu koju objekt sadrži moguće je pregledati i urediti u inspektoru. Komponente koje su najčešće korištene u ovom projektu su preinaka (eng. Transform), kruto tijelo (eng. Rigidbody), sudarač (eng. Collider), izvor zvuka (eng. Audio Source) i programski kod, odnosno skripta (eng. Script). Korisnik može dodati novu komponentu tako da u inspektoru odabere gumb Dodaj Komponentu (eng. Add Component) te odabere komponentu iz ponuđenog izbornika. Komponente se mogu pretražiti upisivanjem naziva ili pretraživanjem po vrstama komponenata.

3.1.1. Preinaka



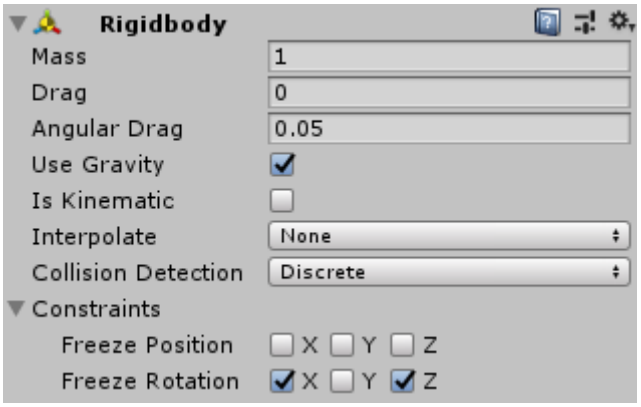
Slika 6: Prikaz osnovnih informacija o objektu i preinaka u inspektoru (slika autora, 2019.)

Svaki objekt u inspektoru sadrži naziv objekta, oznaku je li objekt trenutno aktivan na sceni (kvačica ispred naziva), oznaka objekta (eng. Tag) i sloj (eng. Layer) koji služi za označavanje redosljeda prilikom izrade slike na zaslon.

Komponenta preinaka omogućuje promjenu pozicije, rotacije i veličine objekta. Pozicija (eng. Position) sadrži koordinate (x, y, z) koje omogućuju korisniku promjenu pozicije objekta u odnosu na izvorište scene ili objekt-roditelj. Rotacija (eng. Rotation) omogućuje promjenu kuta objekta u odnosu na koordinatne osi scene ili objekta-roditelja.

Veličina (eng. Scale) omogućuje promjenu veličine objekta prema koordinatnim osima u odnosu na originalnu veličinu objekta.

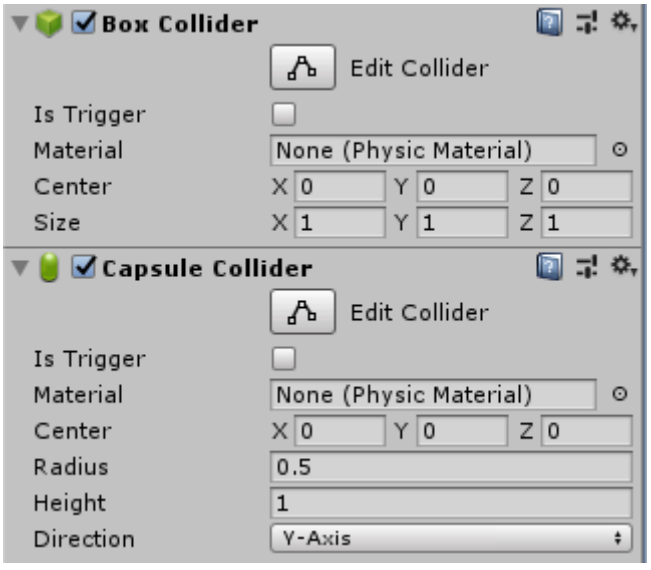
3.1.2. Kruto tijelo



Slika 7: Prikaz komponente krutog tijela u inspektoru (slika autora, 2019.)

Komponenta krutog tijela koristi se kako bi se objekt u sceni ponašao prema zakonima fizike. Korisnik može mijenjati masu objekta (eng. Mass), trenje (eng. Drag) i kutno trenje (eng. Angular Drag), omogućiti utjecaj gravitacije na objekt (opcija Use Gravity), isključiti fiziku na objekt (opcija Is Kinematic), promjenu vrste interpolacije (opcija Interpolate) koja se u nekim situacijama koristi za popravljanje razlike u fizici i prikazu slike na ekranu, promjenu vrste očitavanja sudara (eng. Collision Detection) te ograničenja za poziciju i rotaciju (eng. Constraints) što onemogućuje promjenu pozicije i/ili rotacije za određenu os objekta.

3.1.3. Sudarač

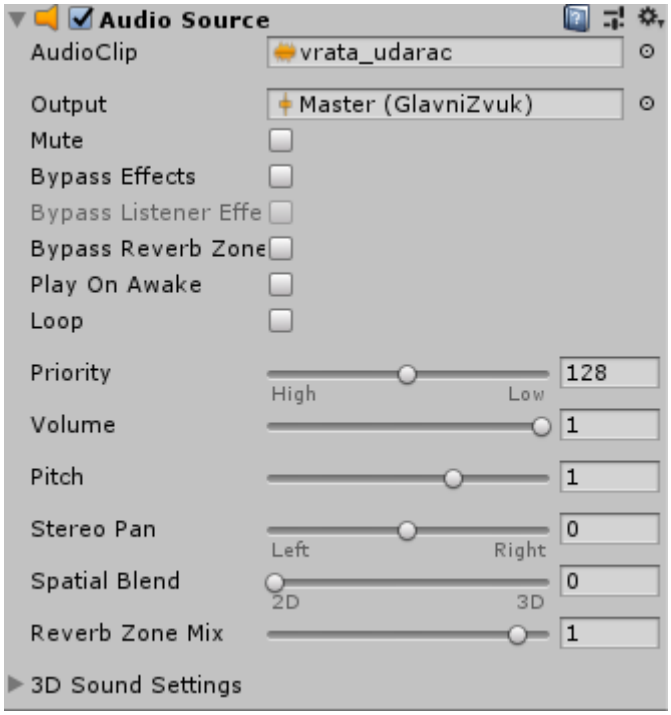


Slika 8: Prikaz komponente sudarača u inspektoru (slika autora, 2019.)

Sudarač je komponenta koju objekt ima ako je bitan njegov odnos s drugim objektima. To je „kostur“ nekog objekta za kojeg računalo računa volumen te na taj način detektira u kakvom su međusobnom odnosu dva objekta (ne dodiruju se, dodiruju se, jedan objekt je unutar drugog objekta i slično).

Objekt na kojeg je postavljen sudarač može napraviti određene akcije ako se nađe u kontaktu (ili ako prestane kontakt) s drugim objektom, no tada je potrebno uključiti opciju Okidač (eng. Is Trigger) na sudaraču. Sudaraču se proizvoljno mogu mijenjati dimenzije (opcija Edit Collider) te tako on ne mora biti veličine kao i objekt na kojeg je postavljen. Također, ako je sudarač neki od predefiniranih oblika, kao primjerice kutija ili kapsula kao na Slici 8 (eng. Box Collider, eng. Capsule Collider) moguće je mijenjati veličinu, radijus, visinu ili usmjerenje (eng. Size, Radius, Height, Direction) te promijeniti odstupanje koordinata sudarača od objekta na kojem se nalazi (opcija Center).

3.1.4. Izvor zvuka



Slika 9: Prikaz komponente izvor zvuka u inspektoru (slika autora, 2019.)

Svaki objekt u Unity-u može imati svoj izvor zvuka koji se programski može pokrenuti u određenim situacijama. Komonenta izvora zvuka ima široku paletu opcija, a mi ćemo objesniti one najvažnije koje su korištene u projektu. Zvučni okvir (eng. AudioClip) je zvučna datoteka koja se pokreće prilikom pokretanja izvora zvuka. Izlaz (eng. Output) služi za odabir zvučnog izlaza u igri, budući da korisnik može napraviti više izlaza, primjerice

jedan za govor likova u igri, drugi za pozadinske zvukove i slično. Glasnoćom izlaza može se upravljati u uređivaču igre ili programskim putem. Zvuk se također može zanijemiti (eng. Mute), postaviti da se pokreće prilikom inicijalizacije objekta (opcija Play On Awake) ili postaviti da se pokreće u petlji, odnosno automatski ponovno pokrene jednom kada završi (eng. Loop). Ostale opcije vrijedne izdvajanja su prioritet puštanja (eng. Priority), glasnoća (eng. Volume) te frekvencija kojom se zvuk može ubrzati ili usporiti (eng. Pitch).

Zvukovi korišteni u ovoj videoigri preuzeti su s web stranice ZapSplat. Programski su uređeni (ubrzani ili usporeni) kako bi odgovarali potrebama igre, a sljedeća tablica sadrži popis svih zvukova koji su korišteni.

Tablica 1: Popis zvukova korištenih u projektu

Naziv u projektu	Trajanje	Preuzeto s
kljuc.mp3	0:01	https://www.zapsplat.com/music/bunch-of-keys-throw-fly-through-air-then-catch/ 20. srpnja 2019.
platforma_pomicanje_2.mp3	0:25	https://www.zapsplat.com/music/conveyor-belt-operating/ 19. srpnja 2019.
prekidac.mp3	0:00	https://www.zapsplat.com/music/switch-light-6/ 4. prosinca 2018.
vrata.mp3	0:03	https://www.zapsplat.com/music/metal-gate-squeak-groan-open-or-close-4/ 4. prosinca 2018.
vrata_level.mp3	0:02	https://www.zapsplat.com/music/glass-patio-sliding-door-slide-1/ 19. srpnja 2019.
vrata_udarac.mp3	0:07	https://www.zapsplat.com/music/metal-impact-loud-large-and-heavy/ 4. prosinca 2018.

3.1.5. Programski kod (skripta)

Slika 10: Prikaz skripte i javnih atributa u inspektoru (slika autora, 2019.)

Osim svih komponenti čije su promjene vidljive na sceni, objektu je moguće dodati i programski kod koji će utjecati na ponašanje objekta kada se ispune određene akcije ili uvjeti. Svi programski kodovi pisani u ovom projektu, napisani su u programskom jeziku C#. Kao što je navedeno, Visual Studio je korišten kao vanjski uređivač koda.

Dodavanjem skripte na objekt, dodaje se komponenta Script koji dvoklikom na naziv skripte otvara programski kod u definiranom uređivaču koda. Svaki javni atribut u klasi koju korisnik napiše za objekt bit će vidljiv u inspektoru. Skripta „IgracKontrole“ na Slici 10 sadrži javne atribute Tlo (tip podataka LayerMask), ImaKljuc (tip podataka Toggle) te TekstPomoc (tip podataka Text) koje korisnik može referencirati tako da jednostavno dovuče odgovarajući objekt u polje pokraj naziva atributa ili ručno upisati vrijednost atributa ukoliko je to moguće.

3.2. Programski kodovi

Za potrebe ovog projekta, napisano je 26 programskih kodova koji su dodani na razne objekte u igri. Sljedeća tablica prikazuje nazive skripti te kratak opis njihove uloge:

Tablica 2: Programski kodovi i njihove uloge

Skripta	Opis
BrojacMase	Računanje mase postavljenu na platformu u Razini 2
DetekcijaKutije	Detekcija kutije postavljenu na platformu u Razini 4
GumbKutije	Inicijalizacija kutije u Razini 3 i pokretanje mehanizma za izbacivanje kutije
GumbPlatforma	Pritisak gumba u Razini 4 i pokretanje zvuka
GumbVrata	Aktivacija vrata u Razini 1 i promjena boje gumba
IgracKontrole	Kontrole objekta kojim upravlja igrač
IgracMis	Kontrola kamere koja je postavljena na igrača, odnosno upravljanje mišem
Izbornik	Funkcionalnosti izbornika
Kljuc	Funkcionalnosti ključa
KombiniranjeBoja	Funkcionalnosti za dobivanje kutija sekundarnih boja u Razini 3
KrajLevela	Metode koje se pozivaju kad je razina gotova
NosacPlatforme	Funkcionalnost električnog nosača pomičnih platformi u Razini 4
ObjektiUIzborniku	Skripta za rotiranje objekata u izborniku Pomoć
PlatformaBoja	Funkcionalnosti obojanih platformi u Razini 3
PlatformaCrna	Funkcionalnosti crnih platformi u Razini 3
Ploca	Funkcionalnosti ploče koja izbacuje kutije u Razini 3
PomicanjePlatforme	Metode za kretanje pomičnih platformi u Razini 4
Postavke	Upravljanje postavkama
UpraviteljBoja	Skripta za kontorlu obojenih platformi u Razini 3

UpravljacVrata	Skripta za kontrolu vrata u Razini 4
UzmiKutiju	Funkcionalnosti kutija
VrataLevel	Metode vrata koje je potrebno proći za prolazak razine
VrataLevel1	Metode vrata koje je potrebno proći dobivanje ključa u Razini 1
VrataLevel2	Metode vrata koje je potrebno proći dobivanje ključa u Razini 2
VrataLevel3	Metode vrata koje je potrebno proći dobivanje ključa u Razini 3
VrataLevel4	Metode vrata koje je potrebno proći dobivanje ključa u Razini 4

U nastavku ćemo izdvojiti ajvažnije programske kodove na ovom projektu. Prilikom kreiranja svake klase automatski se dodaju biblioteke System.Colletcions, System.Collections.Generic i UnityEngine. Svaka klasa je javna i nasljeđuje klasu MonoBehaviour. Također sve klase dolaze s praznim privatnim metodama Start i Update. Sve ostalo dodano je ili uklonjeno u svrhu izrade ove videoigre. Pomoć prilikom pisanja ovih programskih kodova pronađena je na YouTube kanalima Holistic3d (youtube.com, 2016.) i Brackeys (youtube.com, 2017.) te službenim stranicama Unity-a (learn.unity.com, 2019.)

3.2.1. IgracKontrole.cs

Klasa IgracKontrole sadrži funkcionalnosti potrebne za upravljanje objaktom igrača ten neke elemente korisničkog sučelja. Za početak, dodane su biblioteke UnityEngine.UI za podršku korisničkog sučelja te UnityEngine.SceneManagement za upravljanje scenama u igri. U Start metodi, uklanjamo pomoćni tekst, dohvaćamo komponente krutog tijela i sudarača, ograničavamo poziciju miša na sredinu zaslona te isključujemo sklopku (eng. Toggle) za ključ. Metoda FixedUpdate dolazi uz klasu MonoBehaviour i koristi se pri računanju fizičkih promjena za kruto tijelo (docs.Unity3d.com ,2019.) pa su funkcionalnosti za kretnje igrača sadržane upravo u ovoj metodi. Ovdje se provjerava unos tipki za kretanje, skakanje i brzo kretanje te se temeljem toga usmjerava igrača u željeni pravac. Osim toga detektiraju se i tipke za savjet i ponovno pokretanje razine te izlazak iz igre. Metoda OnTriggerStay provjerava je li sudarač igrača u koliziji sa sudaračem ključa, te ako jesu i ako je pritisnuta tipka “E”, object ključa poziva svoju metodu KljucZvuk koja pušta zvuk ključa, sklopka ključa se uključuje te s odgodom od 0.25 sekundi ključ postaje neaktivan, odnosno ne vidi se na zaslonu. Metoda ResetirajLevel ponovno pokreće trenutnu razinu koristeći se klasom SceneManager i metodom LoadScene. Uzemljen je metoda koja vraća bool tip podataka ovisno o tome dodiruje li igračev kapsulni sudarač tlo ili ne. Ova metoda koristi se prilikom

provjere može li igrač skočiti ili ne. Metoda PrikaziPoruku ispisuje savjet igraču na ekran temeljem razine na kojoj se nalazi i uz pomoć metode PrikaziTekst. Nakon čekanja od 5 sekundi, poruka nestaje sa zaslona. Ovo je prikaz cjelovitog koda:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class IgracKontrole : MonoBehaviour
{
    private Rigidbody rbIgrac;
    private float pomakHorizontalno;
    private float pomakVertikalno;
    float brzina = 6.0f;
    float snagaSkoka = 6.0f;
    public LayerMask tlo;
    CapsuleCollider igracSudarac;
    public Toggle imaKljuc;
    public Text tekstPomoc;

    void Start()
    {
        tekstPomoc.text = "";
        rbIgrac = GetComponent<Rigidbody>();
        Cursor.lockState = CursorLockMode.Locked;
        imaKljuc.isOn = false;
        igracSudarac = GetComponent<CapsuleCollider>();
    }

    void FixedUpdate()
    {
        pomakHorizontalno = Input.GetAxis("Horizontal");
        pomakVertikalno = Input.GetAxis("Vertical");

        if (Uzemljen() && Input.GetKeyDown(KeyCode.Space))
        {
            rbIgrac.AddForce(Vector3.up * snagaSkoka, ForceMode.Impulse);
        }

        if (Input.GetKey("left ctrl"))
            brzina = 10.0f;
        else
        {
            brzina = 6.0f;
        }

        Vector3 pomak = new Vector3(pomakHorizontalno, 0.0f, pomakVertikalno);

        rbIgrac.transform.Translate(pomak * brzina * Time.deltaTime);

        if (Input.GetKeyDown("escape"))
        {
            Cursor.lockState = CursorLockMode.None;
            SceneManager.LoadScene(0);
        }

        if (Input.GetKeyDown(KeyCode.R))
        {
            ResetirajLevel();
        }

        if (Input.GetKeyDown(KeyCode.H))
        {
            StartCoroutine(PrikaziPoruku());
        }
    }

    private void OnTriggerStay(Collider kolizija)
```

```
    {
        if (kolizija.gameObject.tag == "KljucTag" && Input.GetKeyDown(KeyCode.E))
        {
            kolizija.gameObject.GetComponent<Kljuc>().KljucZvuk();
            imaKljuc.isOn = true;
            kolizija.gameObject.GetComponent<Kljuc>().Invoke("KljucUkloni", 0.25f);
        }
    }

    void ResetirajLevel()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
    }

    bool Uzemljen()
    {
        return Physics.CheckCapsule(igracSudarac.bounds.center, new
Vector3(igracSudarac.bounds.center.x, igracSudarac.bounds.min.y, igracSudarac.bounds.center.z),
igracSudarac.radius * 0.9f, tlo);
    }

    IEnumerator PrikaziPoruku()
    {
        PrikaziTekst();
        yield return new WaitForSeconds(5.0f);
        tekstPomoc.text = "";
    }

    void PrikaziTekst()
    {
        switch (SceneManager.GetActiveScene().buildIndex)
        {
            case 1: tekstPomoc.text = "Vrata ne ostaju otvorena. Potrebno ih je zadržati."; break;
            case 2: tekstPomoc.text = "Platforme detektiraju promjenu mase. Mogli bi to iskoristiti.";
break;
            case 3: tekstPomoc.text = "Trebat ćemo primjeniti znanje o primarnim i sekundarnim
bojama."; break;
            case 4: tekstPomoc.text = "Neke platforme treba 'zaključati' u mjestu, a druge pomaknuti.";
break;
        }
    }
}
```

3.2.2. IgracMis.cs

U klasi IgracMis povezujemo objekt igrača s kamerom kroz koju igrač „gleda“ svijet. U metodi Start postavili smo objekt igrača kao roditelja objekta kamere, što znači da će kamera pratiti kretnje igrača. U metodi Update bilježimo igračev pomak mišem te joj promjenimo vrijednosti definiranim atributima za osjetljivost i glatkoću pokreta. Lerp metodom se te vrijednosti izglađuju kako ne bi imali nagle pokrete, a zatim se kut zakretanja miša prenosi na igrača i kameru kako si smo dobili efekt okretanja igrača u prostoru. Metoda OnGUI nam dodaje „ciljnik“ na zaslon kako bi korisnik preciznije mogao odabirati akcije tokom igre.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class IgracMis : MonoBehaviour
{
    Vector2 pomakMisa;
    Vector2 glatkiPomak;
    float osjetljivost = 3.0f;
```

```
    float glatkoca = 1.8f;

    GameObject Igrac;

    void Start()
    {
        Igrac = this.transform.parent.gameObject;
    }

    void Update()
    {
        var promjenaMisa = new Vector2(Input.GetAxisRaw("Mouse X"), Input.GetAxisRaw("Mouse Y"));

        promjenaMisa = Vector2.Scale(promjenaMisa, new Vector2(osjetljivost * glatkoca, osjetljivost *
glatkoca));
        glatkiPomak.x = Mathf.Lerp(glatkiPomak.x, promjenaMisa.x, 1f / glatkoca);
        glatkiPomak.y = Mathf.Lerp(glatkiPomak.y, promjenaMisa.y, 1f / glatkoca);
        pomakMisa += glatkiPomak;
        pomakMisa.y = Mathf.Clamp(pomakMisa.y, -90f, 90f);

        transform.localRotation = Quaternion.AngleAxis(-pomakMisa.y, Vector3.right);
        Igrac.transform.localRotation = Quaternion.AngleAxis(pomakMisa.x, Igrac.transform.up);
    }

    void OnGUI()
    {
        GUI.Box(new Rect(Screen.width / 2, Screen.height / 2, 10, 10), "");
    }
}
```

3.2.3. UzmiKutiju.cs

Klasa UzmiKutiju dodana je na objekte kutija i koristi se kako bi igrač mogao uzeti kutiju, nositi je i postaviti na željeno mjesto. U Start metodi povezujemo trenutni objekt s igračem koji će služiti kao „vodič“ kutijama, odnosno igrač će biti objekt-roditelj kutiji. Update računa udaljenost između igrača i objekta te provjerava drži li igrač još uvijek kutiju ili ne te shodno tome uključuje ili isključuje fiziku za taj objekt. OnMouseDown provjerava je li igrač dovoljno blizu objektu da ga uzme u ruke, dok ga metoda OnMouseUp ispušta iz ruku. OnCollisionEnter provjerava je li sudarač kutije u kontaktu s nekim drugim objektom te ga ispušta ukoliko je (zaštitna mjera kako objekt ne bi „prošao kroz zid“).

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class UzmiKutiju : MonoBehaviour
{
    public GameObject stvar;
    public GameObject trenutniRoditelj;
    public Transform vodic;
    public bool drziStvar = false;
    float udaljenost;
    float snaga = 600;

    void Start()
    {
        trenutniRoditelj = GameObject.Find("vodic");
        vodic = trenutniRoditelj.GetComponent<Transform>();
    }

    void Update()
    {
        udaljenost = Vector3.Distance(stvar.transform.position, vodic.transform.position);
```

```
        if (drziStvar == true)
        {
            stvar.GetComponent<Rigidbody>().useGravity = false;
            stvar.GetComponent<Rigidbody>().detectCollisions = true;
            stvar.transform.parent = trenutniRoditelj.transform;
            stvar.transform.position = vodic.transform.position;
            if (Input.GetMouseButtonDown(1))
            {
                stvar.GetComponent<Rigidbody>().AddForce(vodic.transform.forward * snaga);
                drziStvar = false;
            }
        }
        else
        {
            stvar.GetComponent<Rigidbody>().useGravity = true;
            stvar.transform.parent = null;
        }
    }

    void OnMouseDown()
    {
        if (udaljenost <= 1.5f)
        {
            drziStvar = true;
        }
    }

    void OnMouseUp()
    {
        drziStvar = false;
    }

    private void OnCollisionEnter(Collision collision)
    {
        drziStvar = false;
    }
}
```

3.2.4. VrataLevel.cs

Klasa VrataLevel koristi se na svim vratima koje igrač mora proći prilikm završetka razine. U Start metodi vratima se isključuje gravitacija kako bi ostala statična te svjetlo oko njih postaje crveno, pokazujući igraču da nisu ispunjeni uvjeti prolaska. Update metoda provjerava ima li igrač ključ te shodno tome mijenja svjetlo u zeleno. Metoda OtvoriVrata vraća gravitaciju vratima te koristi silu i pušta zvuk kako bi se storio dojam laganog otvaranja. OnTriggerStay detektira jesu li sudarač vrata u koliziji s drugim objektom. Ako je, dodatno se provjerava je li taj objekt igrač, je li pritisnuta tipka „F“, ima li igrač ključ te jesu li vrata zatvorena. Ako su svi parametri ispunjeni, poziva se metoda za otvaranje vrata.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class VrataLevel : MonoBehaviour
{
    public Rigidbody desnoKrilo;
    public Rigidbody lijevoKrilo;
    public GameObject lijevoSvjetlo;
    public GameObject desnoSvjetlo;
    public GameObject gornjeSvjetlo;
```

```
    public Material crveno;
    public Material zeleno;
    public GameObject igrac;
    float brzina = 50.0f;
    bool desnoKriloOtvoreno;
    bool lijevoKriloOtvoreno;
    Renderer svjetloLijevo;
    Renderer svjetloDesno;
    Renderer svjetloGornje;

    void Start()
    {
        desnoKrilo.useGravity = false;
        lijevoKrilo.useGravity = false;
        desnoKriloOtvoreno = false;
        lijevoKriloOtvoreno = false;
        svjetloLijevo = lijevoSvjetlo.GetComponent<Renderer>();
        svjetloLijevo.sharedMaterial = crveno;
        svjetloDesno = desnoSvjetlo.GetComponent<Renderer>();
        svjetloDesno.sharedMaterial = crveno;
        svjetloGornje = gornjeSvjetlo.GetComponent<Renderer>();
        svjetloGornje.sharedMaterial = crveno;
    }

    void Update()
    {
        if (igrac.GetComponent<IgracKontrole>().imaKljuc.isOn)
        {
            svjetloLijevo.sharedMaterial = zeleno;
            svjetloDesno.sharedMaterial = zeleno;
            svjetloGornje.sharedMaterial = zeleno;
        }
        else
        {
            svjetloLijevo.sharedMaterial = crveno;
            svjetloDesno.sharedMaterial = crveno;
            svjetloGornje.sharedMaterial = crveno;
        }
    }

    void OtvoriVrata()
    {
        desnoKrilo.useGravity = true;
        Vector3 pomakDesno = new Vector3(1.0f, 0.0f, 0.0f);
        desnoKrilo.AddForce(pomakDesno * brzina);
        desnoKrilo.GetComponent<AudioSource>().pitch = 0.8f;
        desnoKrilo.GetComponent<AudioSource>().Play();
        desnoKriloOtvoreno = true;

        lijevoKrilo.useGravity = true;
        Vector3 pomakLijevo = new Vector3(-1.0f, 0.0f, 0.0f);
        lijevoKrilo.AddForce(pomakLijevo * brzina);
        lijevoKrilo.GetComponent<AudioSource>().pitch = 0.8f;
        lijevoKrilo.GetComponent<AudioSource>().Play();
        lijevoKriloOtvoreno = true;
    }

    private void OnTriggerStay(Collider kolizija)
    {
        if (kolizija.gameObject.tag == "Igrac" && Input.GetKeyDown(KeyCode.F) &&
igrac.GetComponent<IgracKontrole>().imaKljuc.isOn && desnoKriloOtvoreno == false && lijevoKriloOtvoreno
== false)
        {
            OtvoriVrata();
        }
    }
}
```

4. Tijek igre

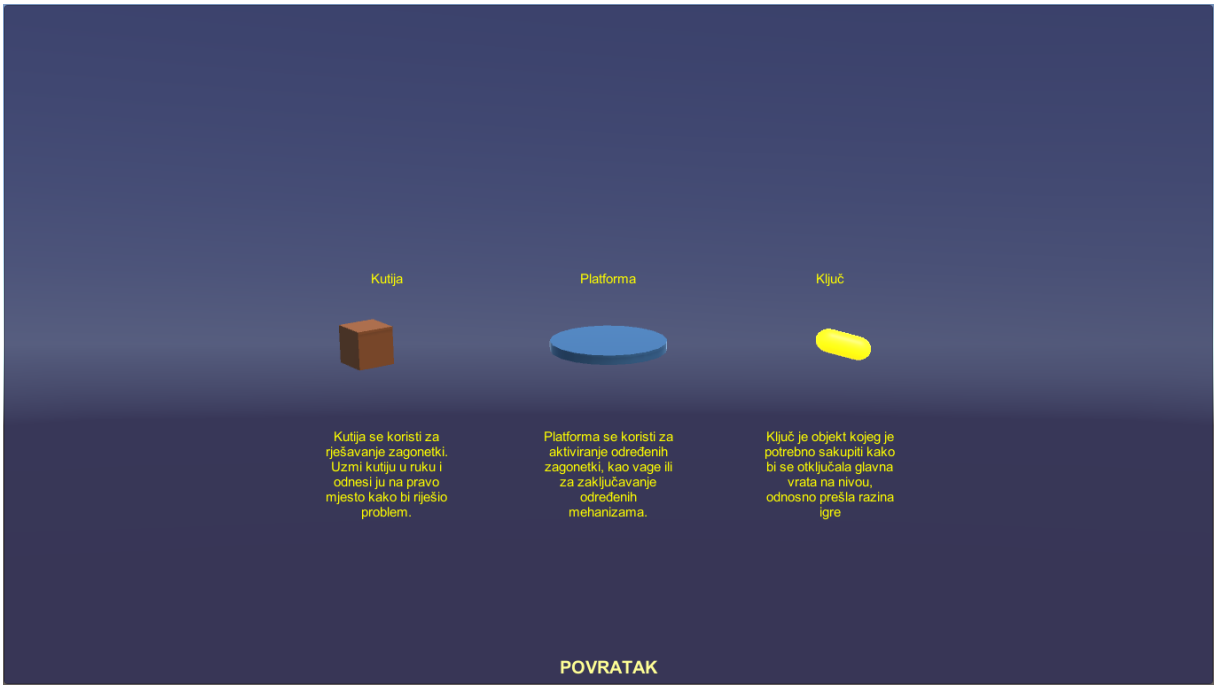
Cilj ove igre je logičko rješavanje zagonetki. Kako bi igrač napredovao kroz razine potrebno je riješiti određeni problem. Cilj svake razine je sakupljanje ključa koji omogućuje otvaranje glavnih vrata koja u konačnici omogućuju završetak nivoa. Ključ se nalazi iza određenih vrata koja je potrebno otvoriti rješavanjem zagonetke ili problema. Korisnik se kroz nivo kreće tipkama „W“, „A“, „S“ i „D“ (naprijed, lijevo, nazad i desno), dok se mišem okreće u prostoru. Lijevom klikom miša može nositi kutije (ako se drži) odnosno može kliknuti na gumbe koji se nalaze na određenim razinama. Desni gumb miša može se koristiti za bacanje kutija koje se trenutno nalaze „u rukama“. Tipka „E“ koristi se za sakupljanje ključeva. Tipka „F“ služi za otvaranje glavnih vrata, dok je tipka „H“ pomoćna tipka koja igraču prikazuje savjet kako prijeći trenutnu razinu. Tipkom „R“ igrač može ponovno pokrenuti trenutnu razinu. Ukoliko igrač želi napustiti trenutnu igru, to može učiniti tipkom „Esc“ koja ga vraća u izbornik.

Slika 11: Izbornik (slika autora, 2019.)

Prilikom pokretanja igre otvara se glavni izbornik koji nudi četiri opcije: Igraj, Postavke, Pomoć, Izlaz. Opcija „Igraj“ pokreće igru i korisnik počinje na razini 1. Pritiskom na opciju „Postavke“ otvaraju se postavke za zvuk gdje korisnik može upravljati glasnoćom zvukova u igri, zatim postavke za sliku gdje korisnik može igru izbaciti iz punog zaslona i obratno, promjena rezolucije zaslona te promjena kvalitete prikaza slike. Također je prikazan i popis kontrola u igri. Odabirom opcije „Pomoć“ prikazuju se objekti na koje bi korisnik

trebao obratiti pažnju prilikom igre (kutija, platforma i ključ). Ukoliko korisnik odabere gumb „Izlaz“, aplikacija se zatvara.

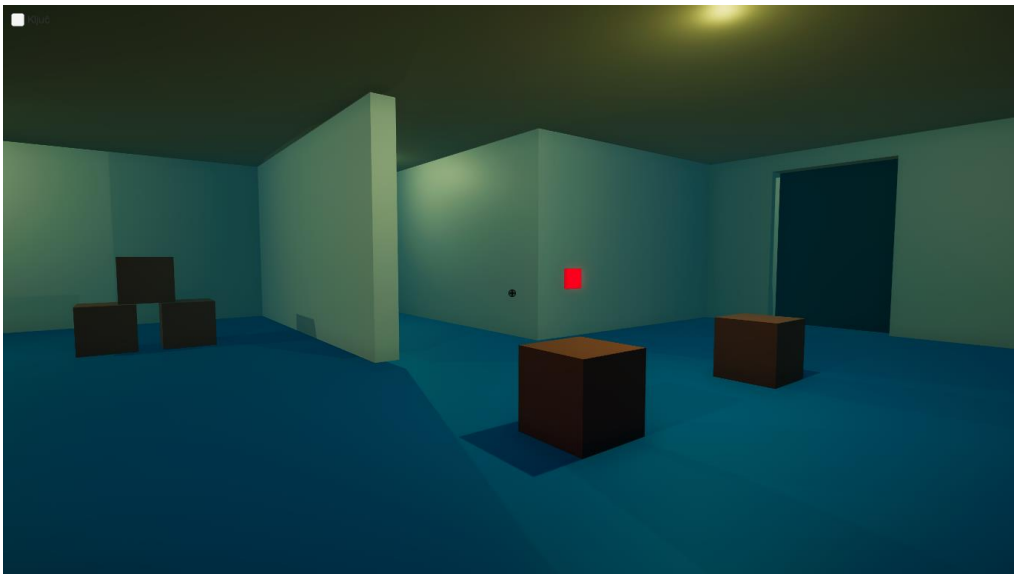
Slika 12: Postavke (slika autora, 2019.)



Slika 13: Pomoć (slika autora, 2019.)

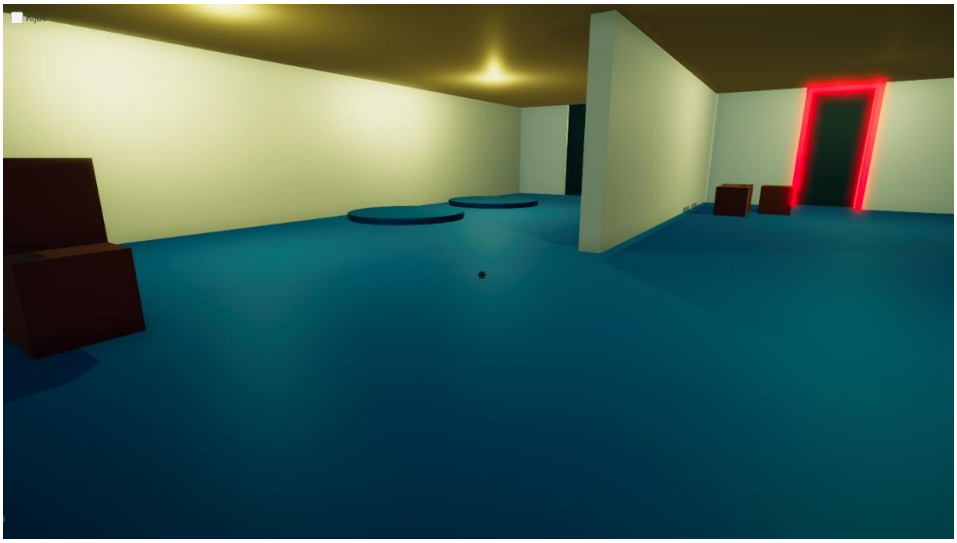
Razina 1 pozicionira igrača u prostoriju u kojoj se nalazi pet kutija, dvoja vrata i jedan gumb kojeg korisnik može stisnuti. Dok su vrata zatvorena ili u procesu zatvaranja, gumb će

emitirati crveno svjetlo, a dok su otvorena ili u procesu otvaranja, gumb emitira zeleno svjetlo. Specifičnost ove razine je ta što se vrata zatvaraju odmah nakon što se u potpunosti otvore, što igraču onemogućuje dolazak do ključa. Cilj razine je postavljanje kutija ispod vrata kako se ona ne bi zatvorila.



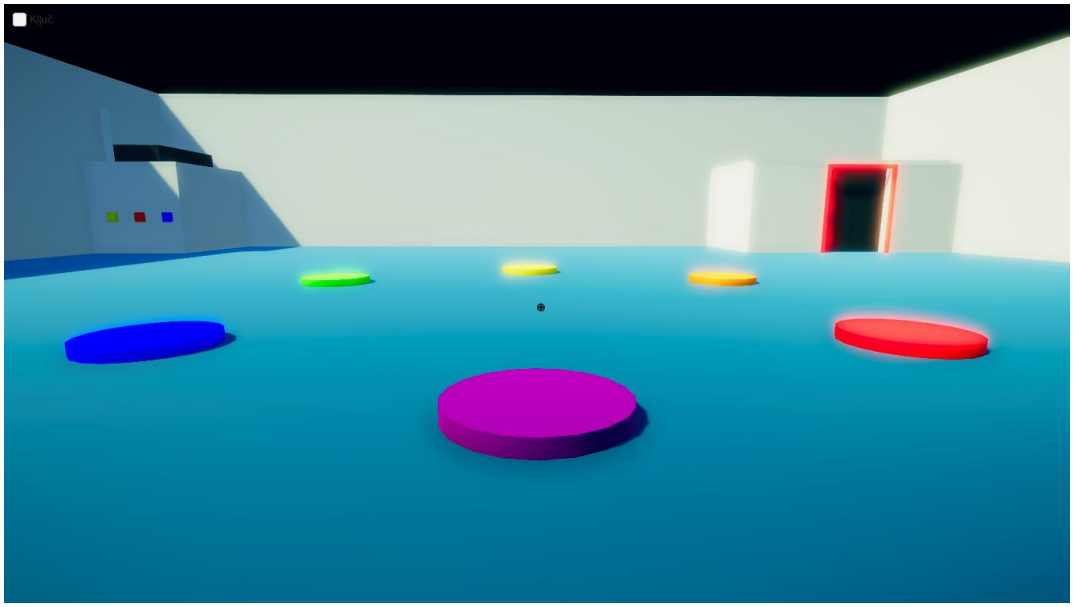
Slika 14: Razina 1 iz igračeve perspektive (slika autora, 2019.)

Razina 2 postavlja igrača u prostoriju u kojoj se nalazi deset kutija, dvije platforme i dvoja vrata. Specifičnost ove razine je ta što se vrata iza kojih se nalazi ključ otvaraju pomoću platformi koje reagraju na masu. Također, na razini 2, nisu sve kutije jednake mase. Cilj ove razine je postaviti dovoljno kutija na svaku od platformi kako bi se vrata u potpunosti otvorila i omogućila igraču sakupljanje ključa.



Slika 15: Razina 2 iz igračeve perspektive (slika autora, 2019.)

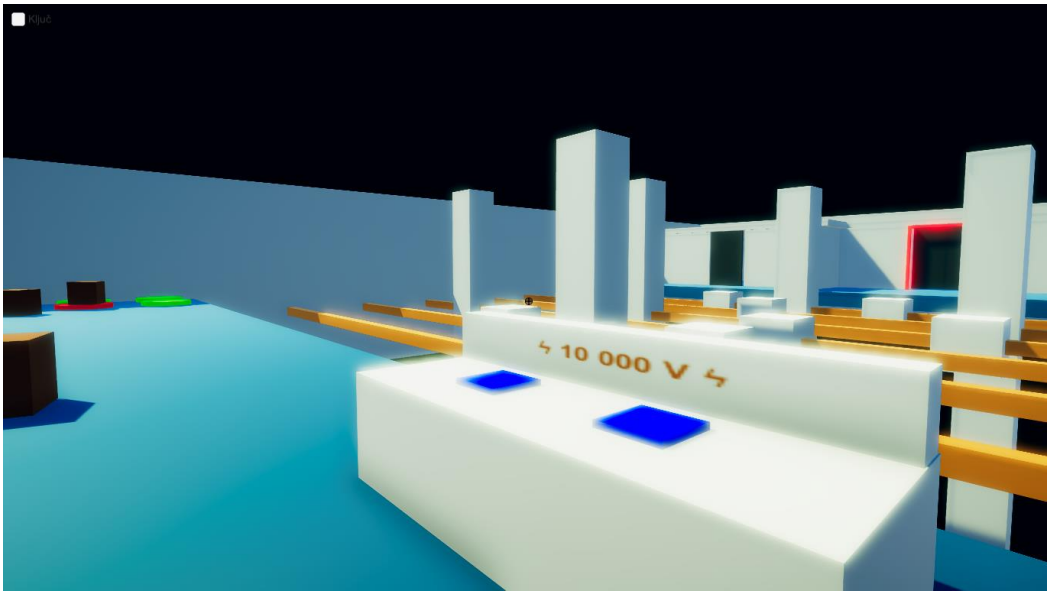
Razina 3 postavlja igrača u veliku prostoriju u kojoj se nalaze dvoja vrata, osam platformi (žuta, narančasta, crvena, ljubičasta, plava, zelena i dvije crne) te tri gumba (žuti, crveni i plavi). Specifičnost ove razine su kutije koje dolaze u različitim bojama i koje korisnik sam mora napraviti. Žute, crvene i plave kutije korisnik može napraviti pritiskom na odgovarajući gumb, dok ostale (naračnaste, ljubičaste i zelene) kutije igrač može dobiti postavljajući po dvije kutije na crne platforme. Primjerice, za zelenu kutiju potrebno je postaviti jednu žutu kutiju na jednu crnu platformu i jednu plavu kutiju na drugu crnu platformu. Cilj razine je postaviti kutiju na platformu tako da boje međusobno odgovaraju (žuta kutija na žutu platformu, narančasta kutija na narančastu platformu i tako dalje). Postavljanjem svih šest kutija na šest platformi u boji otvaraju se vrata iza kojih se nalazi ključ.



Slika 16: Razina 3 iz igračeve perspektive (slika autora, 2019.)

Razina 4 postavlja korisnika u prostoriju koja sadrži troja vrata, dvanaest kutija, dvanaest platformi, dva gumba i deset pomičnih platformi. Na početku, na mjestu na kojem se nalazi korisnik, nalazi se deset platformi koje „zaključavaju“ ili „otključavaju“ pomične platforme, odnosno omogućuju im pomicanje. Platforme se zaključavaju postavljanjem kutije na njih nakon čega one mijenjaju boju iz zelene (otključano) u crveno (zaključano). Pomične platforme mogu se pomicati lijevo i desno pritiskom na lijevi ili desni gumb plave boje koji se nalazi na kontrolnoj ploči ispred igrača. Pomične platforme nalaze se na narančastim vodičima (prikazano da su pod naponom 10 000 volta) koji će automatski ponovno pokrenuti razinu ukoliko ih korisnik dotakne. Također, korisnik ne smije pasti s pomičnih platformi

budući da će tako pasti u bezdan (ako padne potrebno je ponovno pokrenuti razinu). Cilj ove razine je pomaknuti platforme tako da one budu formirane u nizu koji bi omogućio igraču dolazak do desnih vrata. Ondje je potrebno aktivirati jednu od platformi čime bi se otključala lijeva vrata iza kojih se nalazi ključ. Igrač nakon toga mora formirati novi niz ispred lijevih vrata kako bi pokupio ključ. Na kraju, potrebno je formirati novi, posljednji niz, koji bi korisniku omogućio dolazak do glavnih vrata i završetak razine te u konačnici i igre.



Slika 17: Razina 4 iz igračeve perspektive (slika autora, 2019.)

5. Zaključak

Igra je trebala sadržavati jednu kompletnu i igrivu razinu u kojoj bi se objasnile neke osnovne značajke Unity alata. Također, bilo je potrebno opisati neke osnovne mehanike kao što su kretnje igrača i upravljanje mišem. Videoigra se u konačnici sastoji od četiri razine i izbornikom s postavkama za sliku i zvuk. Opisane su sve osnovne mehanike kretnje igrača kao i osnovne mehanike specifične za ovu igru.

Najveći problem bio je sinkroniziranje određenih elemenata zbog velike brzine kojom se izvodi Update metoda te postavljanje sudarača na određene objekte. Igra ne sadrži teksture i uvezene modele ali se kvaliteta Unity-a prikazala korištenjem njihovih osnovnih objekata (kocke, valjci, kapsule i slično) te dodavanjem i modificiranjem svjetlosnih elemenata.

Igra bi se mogla poboljšati ponekim optimizacijama u kodu, dodavanjem dodatnih svjetlosnih i zvučnih elemenata te određenim vizualnim poboljšanjima (kao što je navedeno teksturama i 3D modelima) budući da je za potrebe projekta ta komponenta svedena na minimum. U konačnici, dodavanje novih razina ili poboljšanje postojećih bi pozitivno utjecalo na ovu videoigru.

6. Popis izvora

[1] croteam.com (2014.) *The Talos Principle*. Preuzeto 27. kolovoza 2019. s http://www.croteam.com/talosprinciple/

[2] docs.Unity3d.com (2019.) *MonoBehaviour.FixedUpdate().* Preuzeto 27. kolovoza 2019. s https://docs.unity3d.com/ScriptReference/MonoBehaviour.FixedUpdate.html

[3] learn.unity.com (2019.) *Roll-a-ball.* Preuzeto 2. veljače 2019. s https://learn.unity.com/project/roll-a-ball-tutorial

[4] playdead.com (2019.) *Playdeads LIMBO*. Preuzeto 27. kolovoza 2019. s https://playdead.com/games/limbo/

[5] thinswithportal.com (2011.) *Portal 2*. Preuzeto 27. kolovoza 2019. s http://www.thinkwithportals.com

[6] unity3d.com (2019.) *The world's leading real-time creation platform.* Preuzeto 23. kolovoza 2019. s https://unity3d.com/unity?_ga=2.53564712.1833743957.1566547268-1619419147.1563216329

[7] youtube.com (2016.) *How to construct a simple First Person Controller with Camera Mouse Look in Unity 5.* Preuzeto s kanala Holistic3d 31. ožujka 2018. s https://www.youtube.com/watch?v=blO039OzUZc&t=608s

[8] youtube.com (2017.) *How to make a Video Game in Unity - BASICS (E01).* Preuzeto s kanala Brackeys 15. veljače 2018. s https://www.youtube.com/watch?v=IlKaB1etrik

[9] youtube.com (2017.) *How to make a Video Game in Unity - PROGRAMMING (E02).* Preuzeto s kanala Brackeys 17. veljače 2018. s https://www.youtube.com/watch?v=9ZEu_I-ido4

[10] youtube.com (2017.) *How to make a Video Game in Unity - MOVEMENT (E03).* Prezueto s kanala Brackeys 19. veljače 2018. s https://www.youtube.com/watch?v=Au8oX5pu5u4

[11] youtube.com (2017.) *How to make a Video Game in Unity - CAMERA FOLLOW (E04).* Preuzeto s kanala Brackeys 19. veljače 2018. s https://www.youtube.com/watch?v=HVB6UVcb3f8

[12] youtube.com (2017.) *SETTINGS MENU in Unity*. Preuzeto s kanala Brackeys 24. ožujka 2019. s https://www.youtube.com/watch?v=YOaYQrN1oYQ&t=757s

7. Popis slika

Slika 1: Prikaz izrade videoigre u tri dimenzije u alatu Unity (slika autora, 2019.) 1
Slika 2: Početni prikaz u alatu Unity prilikom otvaranja novog praznog projekta (slika autora, 2019.) 2
Slika 3: Detaljniji prikaz praznog projekta u alatu Unity (1) (slika autora, 2019.) 3
Slika 4: Detaljniji prikaz praznog projekta u alatu Unity (2) (slika autora, 2019.) 4
Slika 5: Prikaz predefiniranog koda u alatu Visual Studio 2017 koji se generira kada napravimo novu skriptu u Unity-u (slika autora, 2019.) 5
Slika 6: Prikaz osnovnih informacija o objektu i preinaka u inspektoru (slika autora, 2019.) .. 6
Slika 7: Prikaz komponente krutog tijela u inspektoru (slika autora, 2019.) 7
Slika 8: Prikaz komponente sudarača u inspektoru (slika autora, 2019.) 7
Slika 9: Prikaz komponente izvor zvuka u inspektoru (slika autora, 2019.) 8
Slika 10: Prikaz skripte i javnih atributa u inspektoru (slika autora, 2019.) 9
Slika 11: Izbornik (slika autora, 2019.) 17
Slika 12: Postavke (slika autora, 2019.) 18
Slika 13: Pomoć (slika autora, 2019.) 18
Slika 14: Razina 1 iz igračeve perspektive (slika autora, 2019.) 19
Slika 15: Razina 2 iz igračeve perspektive (slika autora, 2019.) 19
Slika 16: Razina 3 iz igračeve perspektive (slika autora, 2019.) 20
Slika 17: Razina 4 iz igračeve perspektive (slika autora, 2019.) 21

8. Popis tablica

Tablica 1: Popis zvukova korištenih u projektu 9
Tablica 2: Programski kodovi i njihove uloge 10