Izrada edukativne 3D Ecodrome u alatu Unity

Ilić, Anton

Sveučilište u Rijeci – Odjel za informatiku

Preddiplomski jednopredmetni studij informatike

Anton Ilić

Izrada edukativne 3D igre Ecodrome u alatu Unity

Završni rad

Mentorica: izv. prof. dr. sc. Marina Ivašić-Kos

Rijeka, rujan 2019.

Zadatak za završni rad

Sažetak

U ovom radu opisana je izrada 3D edukativne igre *Ecodrome* namijenjene djeci osnovnoškolskog uzrasta.

Igra se sastoji od dva djela od kojih jedan predstavlja fazu učenja s tri razine, dok drugi predstavlja fazu provjere usvojenog znanja. Cilj prvog djela igre je skupiti znanje na temu ekologije i problema koje stvaraju zagađenje i ljudski nemar za prirodu u okruženju sa i bez prepreka. Drugi dio igre se sastoji od provjere znanja u obliku kviza.

Ključne riječi

Unity, program, 3D, edukativna, igra, učenje, platformer, infinite, endless, runner, quiz, C#, prepreke, prefab

Sadržaj

1. Uvod

Od osamdesetih godina dvadesetog stoljeća do danas, industrija videoigara se zbog informatizacije društva i velike dostupnosti uređaja probila u svaku sferu naših života. Objedinjujući elemente slike, zvuka, kompleksnih ponašanja i kontrola, videoigre su postale jedan od sofisticiranijih i najraznovrsnijih oblika zabave.

Videoigre dolaze u raznim formatima od kojih svaki zahtijeva savladavanje i usvajanje određenih vještina ovisno o žanru koji ih definira. Suprotno vjerovanju mnogih koji igre smatraju gubitkom vremena, brojni primjeri prikazuju benefite koje određeni žanrovi nose. Uzmimo za primjer neke od najpopularnijih žanrova igara. Da bi igrač bio uspješan u igrama akcijskog tipa [1], mora poboljšati svoje motoričke sposobnosti te koordinaciju ruku i očiju u svrhu izbjegavanja prepreka, borbi, pucanja i brzih scena. Što se tiče strateških igara [2], igrač mora promišljeno i planirano rasporediti resurse i vojsku da bi izgurao bitku i ostvario cilj. Budući da su takve igre povijesne tematike, igrača možemo zainteresirati za proširenje znanja iz tog područja. Kod MMORPG (*Engl. Massively multiplayer online role-playing game*) igara [3], potrebno je surađivati s ostalim igračima radi ostvarenja zajedničkog cilja što pridonosi sposobnostima timskog rada i uvažavanja kolega.

Žanrovi realizirani u ovom projektu su platformer [4] i endless runner [5]. U platformer igrama cilj igre je pomoću mehanizama slobodnog kretanja, skakanja i penjanja navigirati po mapi na kojoj nerijetko nalazimo zagonetke, prepreke i neprijatelje. S druge strane, endless runner igre su podvrsta platformerskih igara u kojima se igrač automatski kreće naprijed po proceduralno generiranom terenu. Kroz igru nailazimo na komponente platformera u vidu prepreka i neprijatelja, ali uz ograničene kontrole za kretanje likom.

Ukoliko takvim igrama pridodamo edukativnu komponentu, uz zabavne elemente koje igre nose, možemo proširiti znanje igrača u raznim područjima. Na primjeru igre obrađene u ovom projektu moguće je stvoriti okruženje za prikupljanje novog znanja. Koristeći veliku bazu informacija, igrač u više pokušaja igranja iste igre može prikupiti veliki broj novih informacija pritom učeći na svojim greškama. Kod tradicionalnih metoda učenja, osoba koja padne test nema ponovnu priliku iskazati svoje znanje dok računalne igre dozvoljavaju neograničeni broj pokušaja savladavanja gradiva. Da bi kod takvih igara postigli provjeru naučenog, u nju mogu biti implementirane pitalice s bodovima pomoću kojih se može mjeriti uspjeh [6].

Neke od poznatijih igara koje sadrže navedeni princip učenja su Assassin's Creed saga i Math Blaster. Assassin's Creed je akcijska igra čija se tematika događa za vrijeme važnih povijesnih razdoblja te igraču povećava interes za dodatno prikupljanje informacija i povijesnih činjenica. Nadalje, igra Math Blaster je 2D arkada [7] edukativnog karaktera u kojoj igrač pomoću svog svemirskog broda prikuplja smeće odbačeno u svemiru uz periodičke provjere raznih matematičkih zadataka [8].

2. Razvojni alati

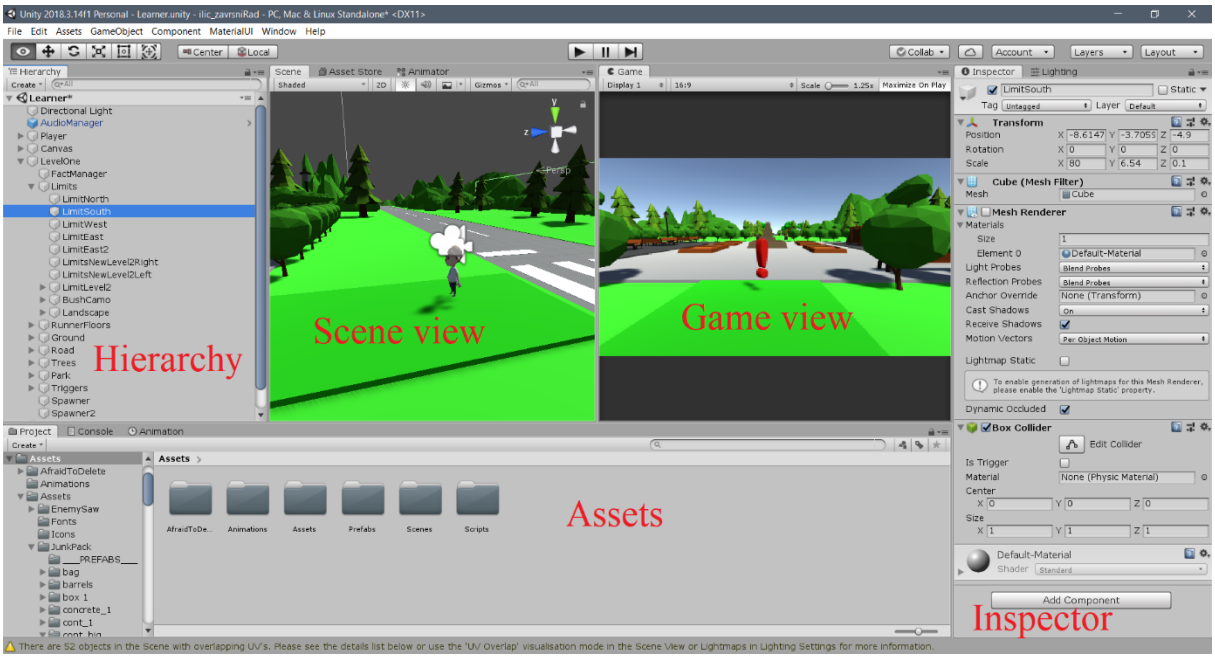
Za izradu igre Ecodrome korišteni su razni alati za pisanje programskog koda, 3D modeliranja te obrade slike i zvuka.

2.1. Unity

Unity je višeplatformski alat za izradu videoigara razvijen od strane Unity Technologies. Objavljen je u lipnju 2005. godine kao platforma za izradu OS-X igara. Zbog svoje velike popularnosti i brojnih mogućnosti do 2018. godine je proširen te pruža mogućnost izrade igara za više od 25 platformi [9]. Neke od brojnih mogućnosti koje Unity pruža, osim izrade videoigara, su 2D i 3D simulacije virtualne stvarnosti, animiranje i snimanje. Danas, Unity nalazi široku primjenu u mnogim industrijama. U ovom poglavlju su opisane neke od značajki sučelja.

2.1.1. Sučelje alata Unity

Zbog opširnosti samog alata Unity u ovom će poglavlju biti prikazane i opisane osnovne komponente čije je korištenje neophodno pri izradi videoigara. Te komponente su su Hierarchy, Assets, Inspector, Scene view i Game view. Elementi su vidljivi na slici 1.



Slika 1: Unity sučelje

Hierarchy panel sadrži popis svih objekata koji se trenutno nalaze na sceni. Razmještanjem objekata više ili niže u hijerarhiji možemo postići efekt dubine objekata. Objekte u hijerarhiji možemo grupirati čime stvaramo Parent-Child odnos pri čemu objekti djece poprimaju poziciju i svojstva roditelja. Također, Hierarchy pruža mogućnost aktivacije i deaktivacije objekata što pruža brojne mogućnosti kroz igru u vidu otvaranja novih razina, stvaranja objekata i čišćenja memorije.

Assets panel je prikaz mape koja se generira unutar Unity projekta. Uključuje sve skripte, uvezene datoteke i assete. Koristimo ju za odvlačenje objekata na scenu te pomoću nje možemo držati projekt urednim i sortiranim. Da bi se olakšala navigacija kroz Assets mapu, u gornjem desnom uglu pruža se mogućnost pretrage i filtera za sortiranje.

Inspector panel služi za opću manipulaciju svojstvima objekata u projektu. Pomoću njega možemo upravljati veličinom, pozicijom i rotacijom objekata pritom koristeći precizne vrijednosti na više decimala. Uz navedeno, objektima je u inspectoru moguće pridodati oznake na koje se pozivamo u skriptama te dodavati skripte i gotove komponente dostupne od strane Unitya.

Scene view alata Unity pruža mogućnost dodavanja i manipulacije svojstvima veličine, pozicije i rotacije objekata pomoću miša i tipkovnice. Omogućava nam razmještanje objekata po sceni i slobodu kretanja čime dobivamo mogućnost sagledavanja izvedbe scene iz više kutova što pomaže kod otkrivanja dizajnerskih grešaka.

Game view developeru pruža mogućnost testiranja trenutne scene bez izgradnje igre. Budući da izgradnja kompleksnih igara može trajati više sati, Game view značajno skraćuje proces testiranja. Dok koristimo Game view, imamo mogućnost razmještaja objekata u realtime-u što može biti korisno pri realizaciji ideja koje traže aktivaciju nekih ponašanja u igri. Izlaskom iz Game view-a brišu se sve modifikacije u sceni.

2.2. Ostali alati

Za izradu računalne igre u viziji developera, neke od elemenata je potrebno izraditi po vlastitoj intuiciji. Željene elemente je teško pronaći u gotovom obliku na način da su krojeni po mjeri za igru na kojoj radimo.

Iako Unity nudi vlastiti editor za pisanje koda, za izradu je zbog veće fleksibilnosti korišten Microsoft Visual Studio [10]. Za potrebe ove igre neki su od zvučnih elemenata ostvareni snimanjem zvuka pametnim telefonom te obradom u programu Audacity [11]. Budući da se radi o 3D igri, neki od tekstualnih elemenata su modelirani u programu Blender [12] dodavanjem treće dimenzije tekstu. Pojedini grafički elementi su nacrtani u alatu Adobe Photoshop [13].

3. Razvoj igre

Poglavlje opisuje inspiraciju za igru te žanrove koji je definiraju. U daljnjem tekstu je opisana implementacija i opis kodova najvažnijih dijelova

3.1. Ideja

Igra Ecodrome je inspirirana kratkim edukativnim igrama koje su učenici osnovne škole u prošlom desetljeću dobivali na poklon uz kupovinu udžbenika ili edukativnih časopisa. Osnovna ideja izrade je skupiti što više različitih žanrova igara, spojiti ih u jednu igru uz dodavanje edukativne komponente. Zbog toga, žanr igre nije moguće jednoznačno odrediti. Prvi dio igre ima element slobodnog kretanja igrača uz skupljanje objekata, kasnije prvi dio igre poprima oblik platformera. Kroz igru se nude misije koju definira endless runner i trivia [14] žanr.

Prvi dio igre je informativnog tipa, igrač poprima ulogu djeteta u parku okruženom šumicom i opasnom cestom za koju je poznato da je puna nemarnih vozača. Dijete mora skupiti informacije o ekologiji i ugroženosti planeta Zemlje koje su skrivene u obliku upitnika po cijeloj mapi. Skupljanje može biti otežano s opasnostima koje nastaju kod prelaska ceste i neodgovornim ostavljanjem informacija na cesti. Kada dijete skupi sve informacije, primjećuje da mapa sadrži još jedan dio koji predstavlja novi izazov. Dijete mora taktički savladati prepreke koje ga dijele od elemenata slagalice koja prikazuje razliku između zagađenog i nezagađenog područja. Nakon savladavanja te dvije misije, jedino što djetetu preostaje je igranje u parku koje zna postati monotono ili inspiriran skupljenim informacijama može ići čistiti šumu od divljeg odlaganja otpada. U novoj misiji mora izbjegavati prepreke i prikupiti što više smeća. U cijeloj igri ne saznajemo informaciju gdje se nalaze djetetovi roditelji ni zašto je sam došao u park, ali osnovna je ideja čuvati ga od mogućih opasnosti koje ga u igri čekaju. Kada djetetu dosadi repetativno skupljanje informacija preostaje mu izići iz igre učenja i prijeći u igru kviza koja mu pruža šansu provjeriti znanje prikupljeno kroz igru.

3.2. Scena Main Menu

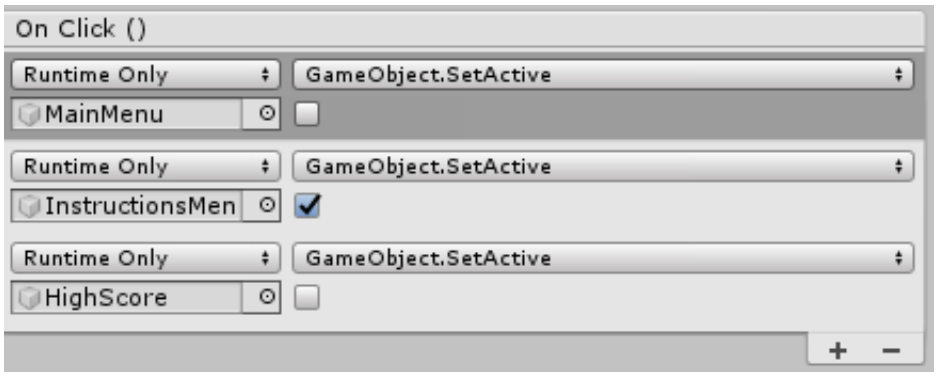
Otvaranjem igre prikazuje se početna scena koja predstavlja glavni izbornik. Navigacija i kretanje izbornikom realizirano je pomoću interaktivnih gumbova za pokretanje scena, uviđaj u informacije o igri te upute za igranje. S desne strane izbornika nalazi se najbolji rezultat misije scene „Runner“.

Za osnovne funkcionalnosti gumba, na objektu MainMenu je umetnuta skripta s klasom *MenuManager*. Klasa sadrži javne metode pomoću kojih dohvaća imena scena koje želimo pokrenuti (Slika 2). Za aktiviranje scene pomoću gumba, na gumb je potrebno dodati OnClick() [15] akciju kojoj dodamo objekt koji uključuje metodu za prijelaz između scena vidljivu na priloženoj slici. Klikom na gumb, metoda se izvršava.

```
27      public void PlayLearner()
28      {
29          SceneManager.LoadScene("Learner");
30      }
```

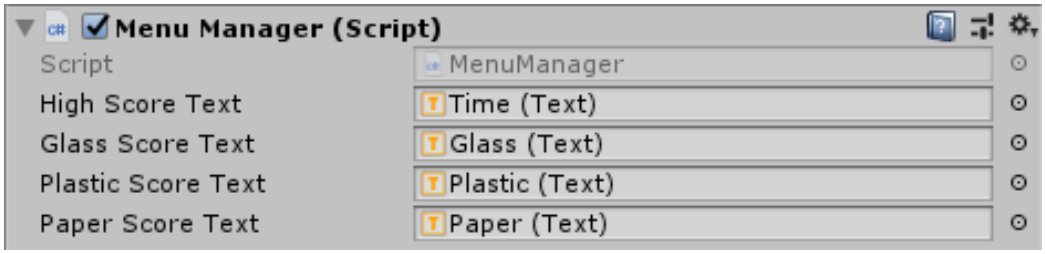
Slika 2: Primjer jedne od metoda za pokretanje scene

Akciju definiramo u inspectoru (Slika 3) tako da u padajućem izborniku pronađemo skriptu koja sadrži željenu metodu. Za primjer možemo uzeti tranziciju iz glavnog izbornika pomoću objekta InstructionsButton. Klikom na gumb objekti MainMenu i HighScore su deaktivirani metodom SetActive [16], dok je objekt InstructionsMenu aktiviran istom metodom ali istinitom vrijednosti.



Slika 3: OnClick() pozivanje metoda za objekte

Za prikaz najboljeg rezultata (*Eng. Highscore*) u klasi *MenuManager* su uvedene četiri javna tekstualna objekta odabranih u inspectoru Unity-a (Slika 4). Tim su objektima pridodane vrijednosti učitane iz PlayerPrefs [17] kojima dohvaćamo vrijednosti definirane u drugim sesijama igranja (Slika 5). Te se vrijednosti (ukoliko postoje) pojavljuju na ekranu prilikom pokretanja igre.



Slika 4: Dodavanje tekstualnih objekata u Inspector

```
14      void Start()
15      {
16          highScoreText.text = "Time:       "     +     (int)PlayerPrefs.GetFloat("HighScore");
17          glassScoreText.text = "Glass:       " +     (int)PlayerPrefs.GetFloat("GlassScore");
18          plasticScoreText.text = "Plastic:   " +     (int)PlayerPrefs.GetFloat("PlasticScore");
19          paperScoreText.text = "Paper:       " +     (int)PlayerPrefs.GetFloat("PaperScore");
20      }
```

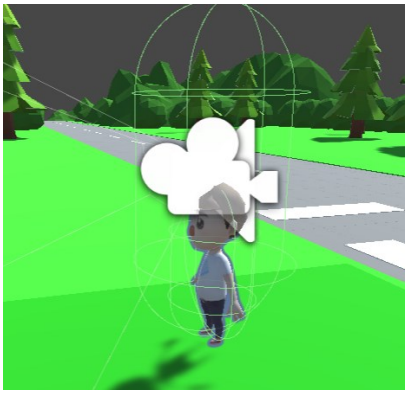
Slika 5: Dohvaćanje i upisivanje u Highscore

3.3. Scena Learner

Scena Learner je edukativni dio igre u kojem igrač slobodnim kretanjem upoznaje svoju okolinu, pritom učeći o važnosti ekologije i problematici zagađenja.

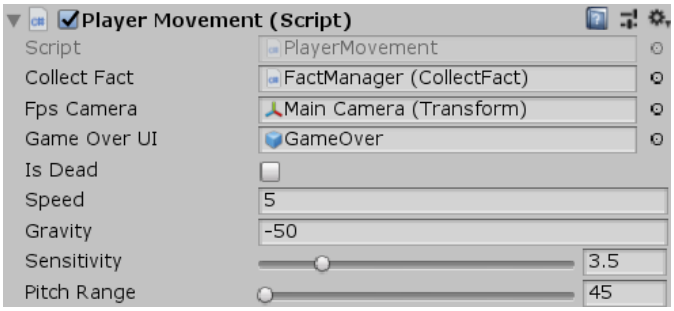
3.3.1. Igrač

Budući da je Learner dio igre izrađen u prvom licu, igrač je ostvaren pomoću praznog objekta (Slika 6) na kojem je dodano svojstvo Capsule Collider [18] i objekt kamere. Capsule Collider je kapsula dodana na objekt pomoću koje objektu dajemo čvrstoću i mogućnost sudaranja s ostalim objektima koji imaju neki od dostupnih oblika collidera. Da bi objekt igrača imao efekt sjene, dodan je prefab [19] korišten u sceni Runner. Prefab je unaprijed stvoren objekt sa svim potrebnim komponentama i svojstvima koji je moguće koristiti više puta kroz izradu igre. Za kretanje i svojstva gravitacije lika dodane su komponente *Character Controller* [20] i skripta *PlayerMovement* [21] preuzeta s interneta. Zbog opsežnosti skripte *PlayerMovement,* u nastavku je objašnjeno upravljanje pojedinostima skripte.



Slika 6: Igrač iz scene pogleda

Igrač upravlja likom koristeći tipke W,S,A,D za kretanje te tipku Space za skakanje. Koristeći se mišem, igrač može mijenjati smjer kretanja i razgledavati okolinu. U Inspectoru (Slika 7) Unity sučelja možemo upravljati brzinom kretanja, snagom gravitacije i osjetljivosti miša. Da bi mogli upravljati kretanjem kamere pomoću miša, objekt kamere mora biti dodan u skriptu. Skripta sadrži metodu OnCollisionEnter() [23] koja služi za detekciju sudara s automobilom te skriptu OnTriggerEnter() koja služi za interakciju s objektima koji prenose informacije i upravljanje brojačem skupljenih informacija.



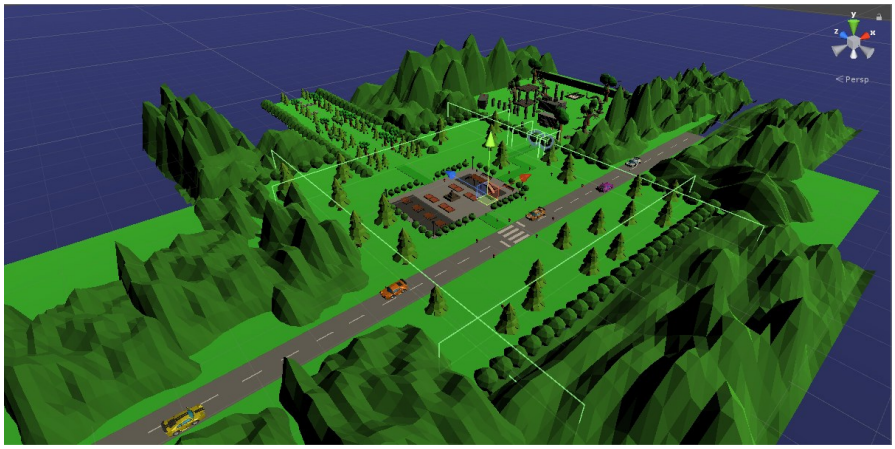
Slika 7: Upravljanje postavkama kretanja igrača

3.3.2. Mapa

Mapa dijela igre koji je zadužen za učenje sastoji se od 3D Low Poly [24] asseta koji su u Scene načinu rada Unity alata posloženi ručno. Sastoji se od dvije razine u kojima su korišteni različiti elementi u svrhu raznolikih zadataka.

Prvi dio mape se sastoji od parka s fontanom i spravama za igru. Park je okružen šumarkom a iz njega vode tri jasno označena puta čija je svrha voditi igrača do obilježja igre: nivo trčanja i skupljanja smeća, drugi nivo i cesta s automobilima koja predstavlja oblik prepreke. Ovaj dio mape uključuje i informacije koje igrač mora skupljati. Drugi dio mape se sastoji od raznih poligona koji predstavljaju statične i pomične platforme gdje igrač mora skupljati i raspoređivati elemente.

Na sve dijelove mape su dodani Box Collideri [25] pomoću kojih igrač dobiva dojam o interakciji sa svojom okolinom. Tim putem su ostvarene i granice igre koje možemo dijeliti na barijere i vizualne granice. Barijere su fizičke granice ostvarene pomoću izduženih kocki koje su isključenjem Renderera [26] učinjene nevidljivima (Slika 8), dok su vizualne granice realizirane pomoću planina, grmlja i mora.



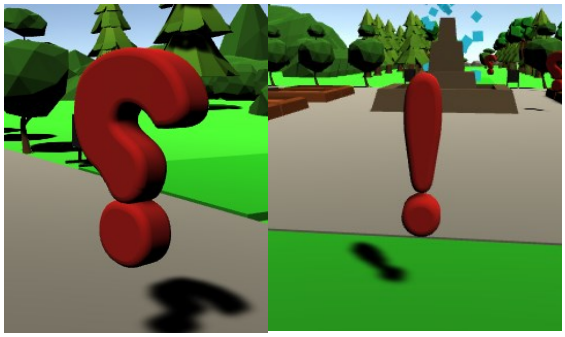
Slika 8: Mapa s uključenim granicama

3.3.3. Informacije

Za prikupljanje informacija u igri uvedene su dvije vrste objekata. Prva vrsta je realizirana u obliku uskličnika dok je druga realizirana pomoću upitnika. Uskličnici igraču služe za prenošenje uputa o njegovim zadacima, dok upitnici služe za učenje i prikupljanje informacija za kviz.

Obje vrste objekata su realizirane 3D modeliranjem teksta u alatu Blender [12] te imaju svojstvo kontinuirane rotacije.

Kroz igru, igrač se susreće s dva objekta u obliku 3D upitnika (Slika 9) koji nose uputu o trenutnom zadatku. Za realizaciju interakcije između igrača i upute, na objekt je dodano svojstvo Box Collider s okidačem događaja.



Slika 9: Uskličnik za prijenos uputa

Za prikaz UI elementa zadužena je klasa *TriggerInstructions.* Prilikom ulaska u zonu sudara između objekta i igrača koji nosi oznaku „Player“, metoda OnTriggerEnter() aktivira grafičko sučelje s odgovarajućom uputom za igrača, paralelno deaktivirajući nepotrebne grafičke elemente. Pri izlasku iz zone sudara, metoda OnTriggerExit() [27] deaktivira elemente grafičkog sučelja zbog mogućih preklapanja s ostalim funkcionalnostima sučelja (Slika 10).

```
18      private void OnTriggerEnter(Collider player)
19      {
20          if(player.gameObject.tag == "Player")
21          {
22              uiObject.SetActive(true);
23              uiText.SetActive(true);
24              factText.SetActive(false);
25              FindObjectOfType<SoundManager>().Play("LearnSound");
26          }
27      }
        0 references
28      private void OnTriggerExit(Collider player)
29      {
30          if (player.gameObject.tag == "Player")
31          {
32              uiObject.SetActive(false);
33              uiText.SetActive(false);
34              factText.SetActive(true);
35          }
36      }
```

Slika 10: Metode za aktivaciju i deaktivaciju objekta za prijenos uputa

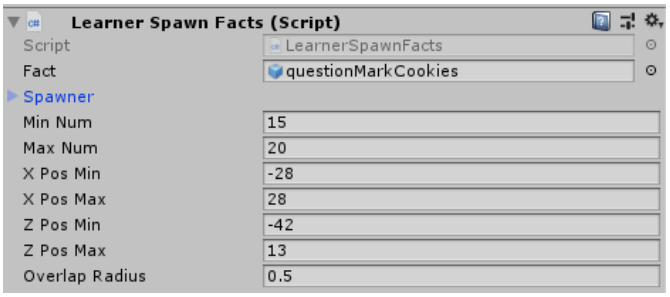
Pri pokretanju nivoa učenja, igrač se susreće s objektima čiji je zadatak prijenos informacija vezanih za učenje i pripremu za kviz znanja Objekt sadrži dva svojstva Box Collider-a od kojih jedan ima svrhu kolizije, dok drugi ima svrhu okidača. Prilikom sudara igrača i objekta aktivira se grafičko sučelje koje sadrži informaciju.

Za generiranje upitnika na mapi zadužena je skripta *LearnerSpawnFacts*. Pri svakom pokretanju igre, radi raznolikosti svakog pokušaja igranja pozicije su drukčije, a informacije različite. Metoda Awake() [28] (Slika 11) prvo određuje random broj informacija koje se spawnaju između minimalnih i maksimalnih koordinata na osima X i Z. Zatim se pomoću Physics.CheckSphere() [29] provjerava postoji li preklapanje između objekata na mapi i objekta koji instanciramo. Ukoliko preklapanja nema, klon objekta se istancira pomoću metode Instantiate() [30].

```
19  void Awake()
20  {
21      int factCount = Random.Range(minNum, maxNum);
22
23      spawner = new GameObject[factCount];
24
25      for (int i = 0; i < spawner.Length; i++)
26      {
27          Vector3 position = new Vector3(xPos, 1, zPos);
28          xPos = Random.Range(xPosMin, xPosMax);
29          zPos = Random.Range(zPosMin, zPosMax);
30
31          if (Physics.CheckSphere(position, overlapRadius))
32          {
33              Debug.Log("Overlap detected!");
34          }
35          else
36          {
37              GameObject clone = Instantiate(fact, position, Quaternion.identity);
38
39              spawner[i] = clone;
40          }
41      }
42  }
```

Slika 11: Metoda za generiranje informacija po mapi

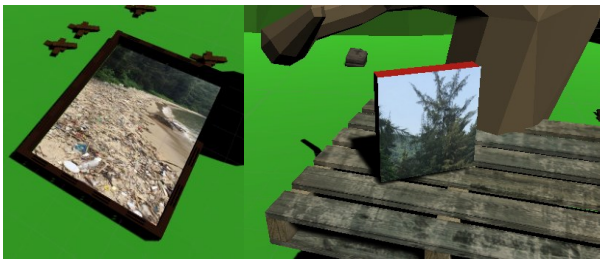
Vrijednosti klase *LearnerSpawnFacts* upisujemo u inspectoru Unity sučelja(Slika 12) na objektu *FactManager*. Potrebno je upisati minimalne i maksimalne vrijednosti broja informacija, X koordinati, Z koordinati te radijus Physics.CheckSphere() koja provjerava postoje li preklapanja s drugim objektima.



Slika 12: Unos public atributa klase LearnerSpawnFacts

3.3.4. Slagalica

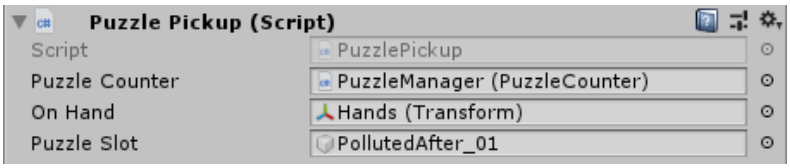
Nakon prolaska prvog nivoa igre, igrača dočeka 3D platformer razina čiji je glavni element slika zagađene obale. Zadatak je preći izazove koje nose platforme te složiti šest komada slagalice koji čine sliku [31] očišćene obale (Slika 13).



Slika 13: Slagalica i komad slagalice

Za navedenu funkcionalnost zadužan je objekt PuzzleManager pomoću klase *PuzzleCounter*. Aktivacijom drugog nivoa, Start() [32] metoda klase *PuzzleCounter* nalazi i prebrojava sve objekte kojima je pridodana oznaka „Puzzle“ koje zatim svrstava u polje. Zatim atributu pomoću svojstva Length [33] provjeravamo veličinu polja čime dobivamo broj komada slagalice koje moramo složiti. Update() [34] metoda prati broj složenih komada te u grafičkom sučelju ispisuje odgovarajuće poruke o broju preostalih komada.

Da bi igrač mogao dohvatiti i prenositi komad slagalice, objekti slagalice sadrže klasu *PuzzlePickup*. U inspectoru Unity alata je dodan objekt PuzzleManager (Slika 14) koji služi za manipulaciju brojača složenih komada, Transform [35] onHand kojim mijenjamo poziciju te odgovarajuća pozicija slagalice.



Slika 14: Umetanje javnih objekata u klasu PuzzlePickup

Prilikom stiskanja lijeve tipka miša poziva se metoda OnMouseDown() [36] kojom igrač dohvaća komad slagalice (Ukoliko isti već nije složen), slagalica postaje „Child“ objekta onHand koji je dio objekta Player. Ukoliko igrač pusti lijevu tipku miša aktivira se metoda OnMouseUp() [37] te slagalica prestaje biti „Child“ objekt igrača. Metode su vidljive na slici 15.

```
13      private void OnMouseDown()
14      {
15          if (pieceFixed == false)
16          {
17              GetComponent<Rigidbody>().useGravity = false;
18          this.transform.position = onHand.transform.position;
19          this.transform.parent = GameObject.Find("Player").transform;
20          }
21      }
22
        0 references
23      private void OnMouseUp()
24      {
25          this.transform.parent = null;
26          GetComponent<Rigidbody>().useGravity = true;
27      }
```

Slika 15: Metode za dohvaćanje i puštanje slagalice

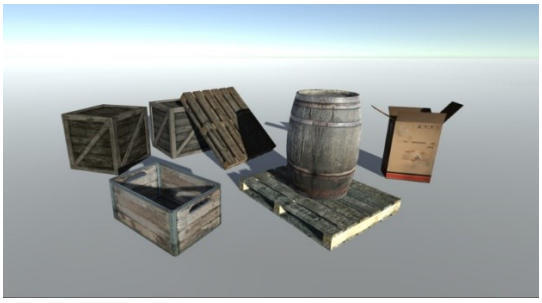
Na svakom odgovarajućem mjestu slagalice nalazi se Box Collider s okidačem koji prepoznaje odgovarajući komad slagalice uspoređujući imena pozicije i komada slagalice. Ukoliko je uvjet ispunjen, odgovarajući komad poprima identičnu poziciju i rotaciju kao i mjesto na koje treba biti složen te se zatim deaktivira. Zatim se na objektu PuzzleSlot aktivira komponenta SpriteRenderer [38] koji sadrži unaprijed postavljenu sliku s komada slagalice čime postižemo vizualni efekt da se komad složio na svoje mjesto. Nadalje, brojač klase *PuzzleCounter* se smanji za jedan, a komad se fiksira te ga više nije moguće pomaknuti niti podignuti (Slika 16).

```
28      private void OnTriggerEnter(Collider other)
29      {
30          if (other.gameObject.name == gameObject.name)
31          {
32              gameObject.transform.position = other.transform.position;
33              gameObject.transform.rotation = other.transform.rotation;
34
35              gameObject.SetActive(false);
36              puzzleSlot.GetComponent<SpriteRenderer>().enabled = true;
37              puzzleCounter.GetComponent<PuzzleCounter>().puzzleCount--;
38              pieceFixed = true;
39
40              FindObjectOfType<SoundManager>().Play("LearnSound");
41          }
42      }
```

Slika 16: Metoda za postavljanje slagalice na željenu poziciju

3.3.5. Platforme

Neki od elemenata koje igrač kroz igru mora savladati su statične i pomične platforme (Slika 17). Da bi igrač na njih mogao kročiti dodano je svojstvo Box Collider koji omogućava sudar s igračem i sprječava pojavu u kojoj igrač prolazi kroz objekt.



Slika 17: Asseti korišteni za izradu platformi

Pomične platforme su vrsta prepreka koje igrač koristi za prijelaz do inače nedostupnih dijelova mape. Za potrebe mehanizma pomicanja platformi zadužena je klasa *MovePlatform* [39] čije isječke možemo vidjeti na priloženim slikama u nastavku.

Putanja pomičnih platformi određena je poljem praznih objekata Waypoints koji služe kao krajnje točke odredišta platforme koje uz brzinu micanja platformi dodjeljujemo u inspectoru.

U Update() (Slika 18) metodi provjeravamo uvjet koji prati udaljenost trenutne udaljenosti između trenutne krajnje točke i pomične platforme. Ukoliko je uvjet ostvaren, bira se sljedeća nasumično odabrana krajnja točka. Kretanje platforme između krajnjih točaka je ostvareno pomoću metode MoveTowards() koja pomiče trenutnu platformu do sljedeće točke zadanom brzinom.

Istim principom je realizirano kretanje automobila po mapi. Jedina razlika je u rutini koja je zadužena za uništavanje objekta automobila nakon određenog broja sekundi. Automobili sadrže i oznaku „Car“ pomoću koje registriramo koliziju između igrača i automobila što označava kraj tekuće runde i aktivaciju Game Over izbornika.

```
7   public GameObject[] waypoints;
8   private int current = 0;
9   public float speed;
10  private float WPradius = 1;
11
    0 references
12  void Update()
13  {
14      if(Vector3.Distance(waypoints[current].transform.position, transform.position) < WPradius)
15      {
16          current = Random.Range(0, waypoints.Length);
17          if(current >= waypoints.Length)
18          {
19              current = 0;
20          }
21      }
22      transform.position = Vector3.MoveTowards(transform.position, waypoints[current].transform.position, Time.deltaTime * speed);
23  }
```

Slika 18: Klasa MovePlatform

Kako bi spriječili situaciju u kojoj igrač padne s platforme na koju je skočio, na objektima su postavljeni dodatni Box Collideri koji služe kao okidači događaja. Takvi objekti gube svojstva kolizije i služe kao okidači metode OnTriggerEnter() (Slika 19).

Kada igrač kroči na platformu, poziva se metoda OnTriggerEnter() kojom objekt Player postaje „Child“ objekt platforme čime poprima putanju platforme.

```
25    private void OnTriggerEnter(Collider other)
26    {
27        if(other.gameObject.tag == "Player")
28        {
29            other.transform.parent = transform;
30        }
31    }
```

Slika 19: Metoda za fiksiranje igrača na platformu

Prilikom silaska s platforme, igrač izlazi iz područja okidača čime se poziva metoda OnTriggerExit() (Slika 20). Tada igrač prestaje biti „Child“ objekt platforme te više ne ovisi o kretanju platforme.

```
33    private void OnTriggerExit(Collider other)
34    {
35        if (other.gameObject.tag == "Player")
36        {
37            other.transform.parent = null;
38        }
39    }
```

Slika 20: Metoda za kraj fiksiranja igrača na platformu

3.4. Scena Runner

Scena Runner je eduativni dio igre u kojem igrač mora saznati što više informacija na temu ekologije izbjegavajući prepreke i čišćeći okoliš.

3.4.1. Igrač

Objekt kojim upravlja igrač (Slika 21) je uveden iz Asset store-a, kao prefab dolazi sa svim kontrolerima i animacijama. Za potrebe ove igre, sve mogućnosti osim animatora [40] i Capsule Collidera su onesposobljene. Jedina animacija koju igrač koristi je ona za trčanje, budući da se koncept igre svodi na trčanje, idle stanje igrača je namješteno na animaciju trčanja.

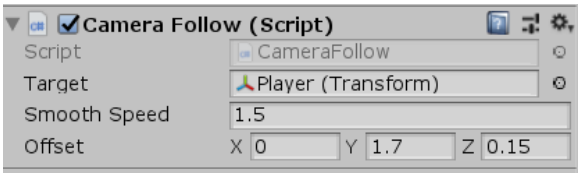
Slika 21: Objekt Player

Na objekt igrača dodana je klasa *PlayerMotion* [41] koja upravlja kretanjem i metodom za detekciju sudara. Frontalno kretanje igrača je realizirano pomoću controller.Move() u Update() metodi te igrač nema kontrolu nad njime. Za horizontalno kretanje, odnosno kretanje po x osi omogućeno je pomoću metode Input.GetAxisRaw() [42] što igraču dozvoljava kretanje samo pomoću tipki A i D. Kod potreban za kretanje u igri je prikazan na slici 22.

```
49          moveVector.x = Input.GetAxisRaw("Horizontal")*speed;
50          moveVector.y = verticalVelocity;
51          moveVector.z = speed;
52          controller.Move(moveVector * Time.deltaTime);
```

Slika 22: Kretanje u igri Runner

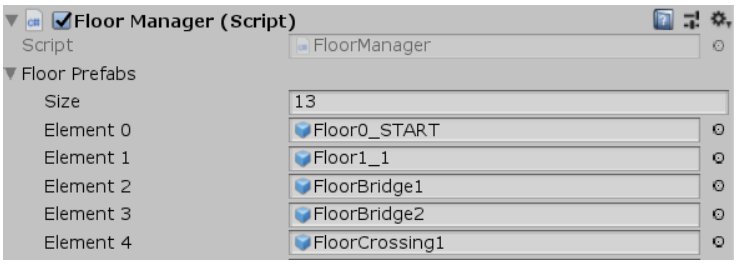
Za postizanje efekta da kamera [43] kroz igru prati igrača, pridodana je skripta *CameraFollow* [44]. Skripti pridodajemo objekt igrača te zadajemo brzinu kretanja i koordinate.



Slika 23: Podešavanje kamere koja prati igrača u sceni Runner

3.4.2. Beskonačna mapa

Da bi se postigao efekt beskonačnog trčanja u svrhu postizanja najboljeg rezultata igranja, endless runner igre imaju obilježje beskonačnog generiranja mape. Beskonačno generiranje mape je omogućeno pomoću objekta FloorManager koji sadrži skriptu *FloorManager* [45]. Front end skripte *FloorManager* (Slika 23) prikazuje polje javnih objekata proizvoljne veličine u koje dodajemo prefabove unaprijed modeliranih dijelova mape iste dužine i širine. Zbog složenosti skripte objasniti ćemo samo metodu za generiranje i brisanje isječaka mape.



Slika 24: Postavljanje isječaka mape u sceni Runner

Za generiranje isječaka zadužena je metoda SpawnFloor() (Slika 24) koja učitava random element iz liste isječaka mape provjeravajući uvjet radi li se o početku igre ili je igra već počela.

Ukoliko je igra tek počela, instancirati će se nulti (početni) isječak mape, ukoliko je prvi isječak postavljen generirati će se nasumično odabrani isječak. Pozicija generiranja sljedećeg isječka pomaknuta je za duljinu svakog isječka, duljina je definirana atributom spawnZ.

```
41      private void SpawnFloor(int prefabIndex = -1)
42      {
43          GameObject go;
44
45          if (prefabIndex == -1)
46              go = Instantiate(floorPrefabs[RandomPrefabIndex()]) as GameObject;
47          else
48              go = Instantiate(floorPrefabs[prefabIndex]) as GameObject;
49          go.transform.SetParent(transform);
50          go.transform.position = Vector3.forward * spawnZ;
51          spawnZ += floorLenght;
52          activeFloors.Add(go);
53      }
```

Slika 25: Metoda za istanciranje isječaka mape

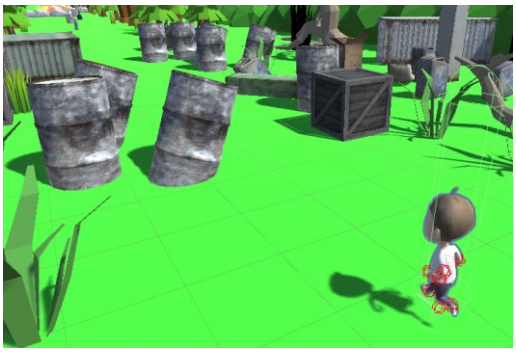
Zbog velikog broja istanciranih isječaka mape, može se javiti problem s manjkom memorije računala. Radi rješavanja tog problema, uvedena je metoda DeleteFloor() (Slika 25) koja provjerava listu objekata activeFloors te briše objekt koji je trenutno na nultoj poziciji.

```
60      private void DeleteFloor()
61      {
62          Destroy(activeFloors[0]);
63          activeFloors.RemoveAt(0);
64      }
```

Slika 26: Čišćenje memorije

3.4.3. Prepreke

Kao jedan od izazova s kojim se igrač igrajući endless runner igru mora suočiti su prepreke. U kontekstu ove igre, prepreke su statički elementi kojima se igrač mora izmicati te ih taktički zaobići. Napravljene su od raznih 3D objekata od kojih neki predstavljaju kamenje, bačve, kontejnere, provalije s mostom,… Neke od tih prepreka su vidljive na slici 26.



Slika 27: Prepreke u sceni „Runner“

Za realizaciju prepreka, na svaku od njih je nužno dodati Box Collider te objekt označiti oznakom „Obstracle“. Prilikom sudara igrača s preprekom dolazimo do kraja tekuće runde. Da bi ostvarili taj mehanizam igre, odgovarajući kod je najbolje staviti u skriptu zaduženu za kretanje igrača, odnosno klasu *PlayerMotion* zbog mogućnosti prekidanja kretanja igrača.

Na slici 27 u nastavku su metode zadužene za mehanizam kraja tekuće igre. Metoda OnControllerColliderHit() [46] se poziva ukoliko se igrač kontrolirajući svog lika sudari s objektom. U ovom kontekstu, ukoliko prepreka nosi oznaku „Obstracle“ i dogodi se frontalni sudar na osi Z (uvjet služi za sprječavanja situacija u kojima prepreku dotaknemo bočnom stranom), poziva se metoda Death(). Pozivom metode Death() prekidaju se zvukovi trčanja, a varijabla isDead poprima istinitu vrijednost čime u metodi Update() pomoću uvjeta prekidamo mogućnost kretanja igrača (Slika 28). Na kraju, iz klase *Score* poziva se metoda OnDeath() koja upravlja sustavom konačnih rezultata.

```
62  private void OnControllerColliderHit(ControllerColliderHit hit)
63  {
64      if ((hit.gameObject.tag == "Obstracle") && (hit.point.z > transform.position.z + controller.radius))
65          Death();
66  }
67
    1 reference
68  private void Death()
69  {
70      isDead = true;
71      FindObjectOfType<SoundManager>().StopPlaying("Running");
72      FindObjectOfType<SoundManager>().Play("crashSound");
73      GetComponent<Score>().OnDeath();
74  }
```

Slika 28: Metode koje aktiviraju kraj runde

```
    0 references
25  void Update()
26  {
27      if (isDead)
28          return;
```

Slika 29: Prekid rada skripte PlayerMotion koja je odgovorna za kretanje

3.4.4. Smeće

Igrač uz savladavanje prepreka mora skupiti što više odbačenog smeća. Smeće se dijeli na tri vrste koje čine tri vrste score-a, a to su: staklo, plastika i papir.

Za generiranje smeća na mapi, zadužena je klasa *SpawnTrash* [47]. U metodi Spawn() (Slika 29) prvo se u polje tipa float učitavaju tri vrijednosti xPos koje predstavljaju zadane koordinate spawnanja objekata na osi X. Zatim se iz polja učitavaju nasumično odabrani elementi iz polja koja čine prefabovi smeća i elementi xPos. Za realizaciju pozicija generiranja uveden je Vector3 [48] hposition čiju X koordinatu čine elementi xPos, Y koordinatu čini visina generiranja dok Z čini trenutna pozicija igrača udaljena za 60 čime postižemo kontinuirano instanciranje smeća ispred igrača i efekt beskonačnosti igre.

Da bi spriječili preklapanje generiranih objekata koristi se Physics.CheckSphere koji provjerava nalazi li se u zadanom radijusu neki drugi objekt. Ukoliko je došlo do preklapanja, igra šalje odgovarajuću poruku, ukoliko nije došlo do preklapanja, smeće se nesmetano generira pomoću metode Instantiate(). Nakon generiranja, poziva se IEnumerator [49] spawnTrash() koji predstavlja odgodu između generiranja novog smeća.

```
24    void Spawn()
25    {
26        float[] xPos = new float[3];
27        xPos[0] = 0f;
28        xPos[1] = 1f;
29        xPos[2] = 2.2f;
30
31        int randomTrash = Random.Range(0, trash.Length);
32        int RandomXposition = Random.Range(0, xPos.Length);
33
34        Vector3 hposition = new Vector3(xPos[RandomXposition], 0.2f, player.position.z + 60);
35        if (Physics.CheckSphere(hposition, sphereRadius))
36        {
37            Debug.Log("Overlap detected!");
38        }
39        else
40        {
41            Instantiate(trash[randomTrash], hposition, trash[randomTrash].transform.rotation);
42        }
43        StartCoroutine(spawnTrash());
44    }
```

Slika 30: Metoda za generiranje smeća po mapi

3.4.5. Score

U svrhu mjerenja postignuća i napretka kroz igru, uveden je sustav bilježenja rezultata (*Eng. Score*). Score se sastoji od brojača vremena te tri brojača koji prate skupljanje tri vrste smeća. Ova funkcionalnost omogućuje klasa *Score* [50].

Brojač vremena osim za praćenje vremena provedenog u sesiji igre služi i za povećavanje težine, odnosno brzine kojom se kreće kroz šumu. Ukoliko trenutna razina težine nije jednaka maksimalnoj predefiniranoj težini, težina se povećava na svakoj desetoj sekundi (Slika 30).

```
65    void LevelUp()
66    {
67        if (difficultyLevel == maxDifficultyLevel)
68            return;
69
70        scoreToNextLevel *= 2;
71        difficultyLevel++;
72
73        GetComponent<PlayerMotion>().SetSpeed(difficultyLevel);
74    }
```

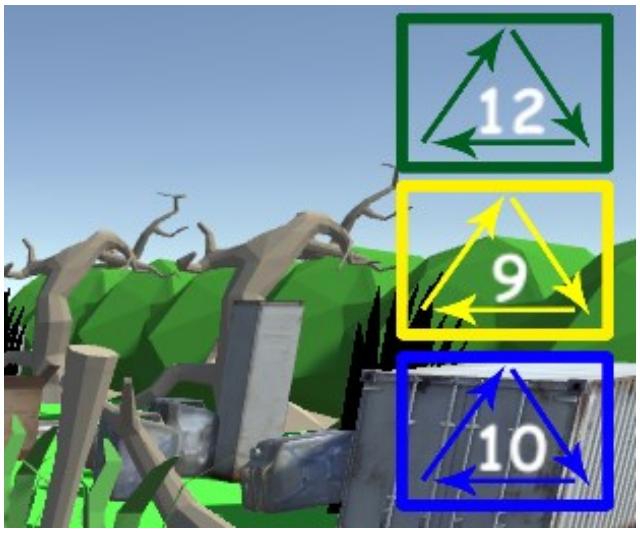
Slika 31: Metoda za povećanje težine igre

Bilježenje skupljenog smeća se u score upisuje prilikom sudara između igrača i smeća pomoću metode OnTriggerEnter(). Ulaskom u zonu okidača smeća, score se povećava, a objekt se briše radi uštede memorije (Slika 31).

```
41      void OnTriggerEnter(Collider other)
42      {
43          if (other.gameObject.tag == "Glass")
44          {
45              scoreGlass++;
46              Destroy(other.gameObject);
47              FindObjectOfType<SoundManager>().Play("CollectSound");
48          }
```

Slika 32: Primjer skupljanja staklenog otpada

Kroz igranje je u svakom trenutku prikazan brojač skupljenog smeća (Slika 32) koji je ostvaren pomoću javnih tekstualnih objekata kojima su dodijeljeni atributi scoreGlass, scorePlastic i scorePaper koji su pretvoreni u tekstualnu vrijednost.



Slika 33: Brojač smeća

Prilikom sudara s preprekom što označava kraj runde, poziva se metoda OnDeath() (Slika 33) koja je zadužena za zapisivanje i pohranjivanje rezultata. Za ostvarivanje bilježenja rezultata zadužen je PlayerPrefs, njegova zadaća je spremanje podataka između sesija igranja. Ukoliko je rezultat bolji od najvećeg zapisanog, rezultat se zapisuje u highscore u glavnom izborniku pomoću metode ToggleEndMenu().

```
76      public void OnDeath()
77      {
78          isDead = true;
79
80          if((PlayerPrefs.GetFloat("HighScore") < score) &&
81              (PlayerPrefs.GetFloat("GlassScore") < scoreGlass) &&
82               (PlayerPrefs.GetFloat("PlasticScore") < scorePlastic) &&
83                  (PlayerPrefs.GetFloat("PaperScore") < scorePaper))
84          {
85              PlayerPrefs.SetFloat("HighScore", score);
86              PlayerPrefs.SetFloat("GlassScore", scoreGlass);
87              PlayerPrefs.SetFloat("PlasticScore", scorePlastic);
88              PlayerPrefs.SetFloat("PaperScore", scorePaper);
89          }
90          gameOver.ToggleEndMenu(score, scoreGlass, scorePlastic, scorePaper);
91      }
```

Slika 34: Metoda za upis rezultata u ispis najboljeg rezultata

3.5. Grafičko sučelje

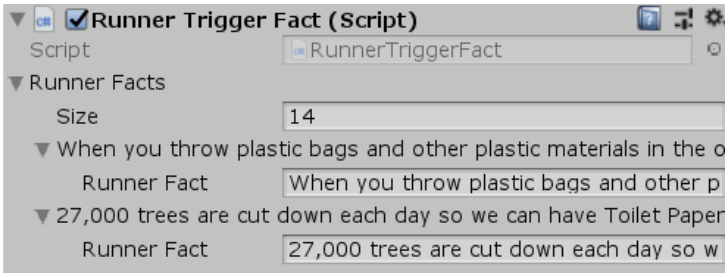
Kroz scene učenja javljaju se tri vrste grafičkog sučelja koja su dio Canvasa [51] koje se tiču prikazivanja informacija za učenje, pauziranja i gubitka u igri.

3.5.1. Prikazivanje informacija

Osnovni element igre Ecodrome je prikupljanje informacija i učenje pojmova na temu ekologije. U scenama učenja igrač dolazi u interakciju s objektima za prijenos informacija, u sudaru s tim objektima aktivira se klasa *RunnerTriggerFact*. Grafičko sučelje aktivirano od strane klase je prikazano na slici 34

Slika 35: Sučelje s informacijom

Unos informacija se vrši u inspectoru (Slika 35) na objektu Player. Da bi višestruki unos informacija bio moguć, klasa *RunnerTriggerFact* sadrži objekt instance klase *RunnerFacts.* Objekt je svrstan u listu iz koje metoda SetFact() uzima nasumično odabranu informaciju pomoću Random.Range() [52] pritom ju ispisujući u tekstualni objekt factText.



Slika 36: Unos novih informacija u inspectoru

Klasa se aktivira pri sudaru s objektom pomoću metode OnTriggerEnter() prikazane na slici 36 koja aktivira grafičko sučelje koje sadrži činjenicu pomoću metode SetFact() pritom zaustavljajući vrijeme.

```
36      void SetFact()
37      {
38          int randomFactIndex = Random.Range(0, undisplayedFacts.Count);
39          currentFact = undisplayedFacts[randomFactIndex];
40
41          factText.text = currentFact.runnerFact;
42      }
43
            0 references
44      private void OnTriggerEnter(Collider player)
45      {
46          if (player.gameObject.tag == "Fact")
47          {
48              FindObjectOfType<SoundManager>().Play("LearnSound");
49              Time.timeScale = 0;
50              SetFact();
51              uiObject.SetActive(true);
52
53              StartCoroutine(HideFact());
54          }
55      }
```

Slika 37: Postavljanje i okidanje informacije

Da bi igrač nastavio igru nakon što pročita informaciju, u Update() metodi postoji jednostavan uvjet koji provjerava je li stisnuta tipka F, ukoliko je stisnuta, igra se nastavlja, a IEnumerator u predefiniranom vremenu od 0.1 sekunde skriva grafičko sučelje s informacijom.

3.5.2. Pause izbornik

Igrač u svakom trenutku misije učenja pritiskom na tipku Escape može pauzirati igru (Slika 37) i nastaviti igru.

Slika 38: Pause izbornik

Izbornik za pauziranje sadrži gumbove za nastavak igre, prijelaz u glavni izbornik te izlaz iz igre. Za funkcionalnosti glavnog izbornika zadužena je klasa *PauseFreeze* koja sadrži metode Pause() i Resume(). Pritiskom na tipku Escape što je omogućeno pomoću Input.GetKeyDown() [53] aktivira se metoda Pause() kojom zaustavljamo vrijeme postavljajući vrijednost Time.timeScale [54] na nulu. Prilikom nastavka igre vrijednost Time.timeScale vraćamo na jedan pri čemu se tok igre nastavlja. Klasa *PauseFreeze* ima i akcije pridodane gumbovima za prijelaz u glavni izbornik te izlaz iz igre.

3.5.3. Game Over izbornik

Izbornik za kraj igre (Slika 38) sastoji se od gumbova za ponovni pokušaj igranja, prijelaz u glavni izbornik te izlaz iz igre. Izbornik se aktivira u sceni Learner, kada se dogodi sudar između igrača i automobila. Tada izbornik prikazuje upozorenje igraču u svrhu učenja prometnih pravila i prelaska preko zebre. U sceni Runner, izbornik se aktivira pri sudaru s preprekom. U obje scene su za ovo ponašanje zadužene skripte za kretanje, odnosno *PlayerMovement* i *PlayerMotion* pomoću odgovarajućih Collidera.

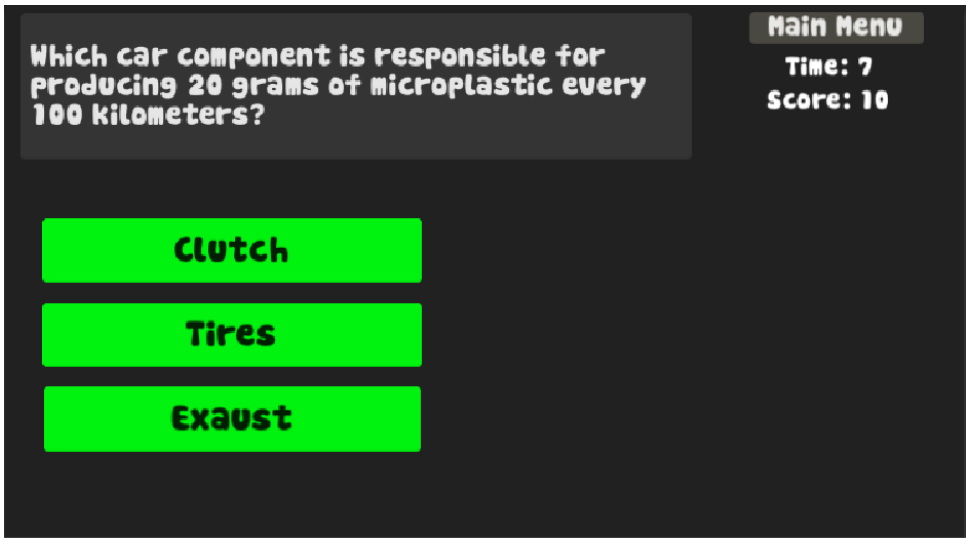
Slika 39: Grafičko sučelje izbornika

3.6. Scena Quiz

Cilj scene Quiz je provjeriti svoje znanje skupljeno u igri tijekom misija učenja.

3.6.1. Sučelje

Sučelje scene Quiz (Slika 39) sastoji se od tekstualnog polja s pitanjem, tri gumba koji predstavljaju odgovore, tajmer i brojač postignutog rezultata. U svakom trenutku igranja, igrač može kliknuti gumb MainMenu koji služi kao izlaz iz igre.



Slika 40: Sučelje kviza

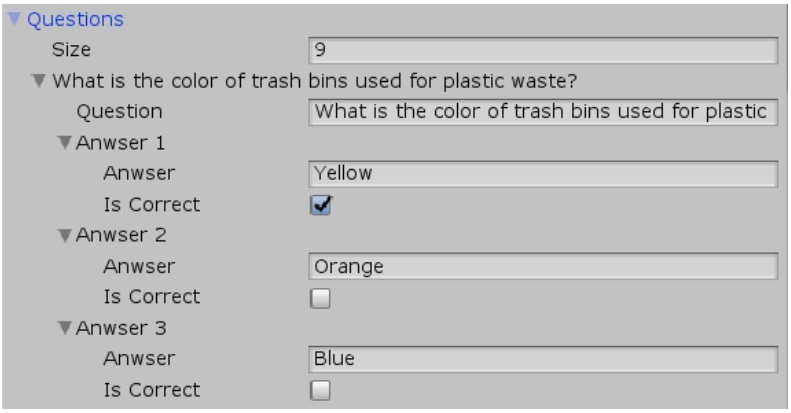
Ukoliko je odgovor na pitanje netočan, otvara se Game Over dio sučelja (Slika 40) koji traži tekstualni upis imena igrača koji se klikom na gumb unosi u JSON [55] tablicu koja prikazuje listu 5 najboljih rezultata kviza.

Slika 41: Unos imena i scoreboard

3.6.2. Quiz Manager

Za veći dio upravljanja scenom Quiz zadužen je objekt QuizManager s klasom *QuizManager* [56] a čiju je izvedbu inspiracija preuzeta iz serije tutoriala. Klasa je zadužena za učitavanje pitanja i odgovora, izmjenu tekstualnih i grafičkih elemenata sučelja, dodavanje i zbrajanje score-a. Zbog svoje opsežnosti, u nastavku su objašnjene samo neke od komponenata.

Pitanja koja se provjeravaju u kvizu su ostvarena pomoću skripti *AnwserData* i *QuestionData*. QuizManager sadrži polje objekata klase *QuestionData* koja uključuje polje objekata *AnwserData*. Time dobivamo mogućnost dodavanja pitanja i odgovora u Inspectoru (Slika 41)



Slika 42: Unos pitanja

Pitanja sadržana u klasi *QuestionData* su svrstana u listu te se elementi odabiru u metodi SetCurrentQuestion() koja iz polja odabire nasumično odabrano pitanje i odgovore pomoću Random.Range().

Za provjeru točnosti odgovora za svaki gumb je uvedena jedna javna metoda UserSelectTrue(). Svaka od tih metoda ima jednostavan uvjet koji provjerava istinitost vrijednosti isCorrent koja je pridodana točnom odgovoru. Ukoliko je odgovor točan, poziva metoda RoundWin(), ako je odgovor netočan poziva se metoda RoundGameOver();

Metoda RoundWin() je zadužena za povećavanje score-a dohvaćajući metodu klase *QuizScore* AddScore() koja povećava trenutni score za 10. Uz to, obavještava igrača o ispravnom odgovoru i zamrzava gumbove metodom FreezeButtons() koja mijenja interaktivnost gumba.

Ukoliko se radi o krivom odgovoru, poziva se metoda RoundGameOver() koja uz zamrzavanje gumbova otvara InputField[57] za unos imena koje će biti pridodano trenutnom ostvarenom rezultatu u scoreboardu.

3.6.3. Scoreboard

Scoreboard je lista imena i rezultata čija je svrha igraču dati uvid u prethodne pokušaje igranja u svrhu uspoređivanja s trenutnim pokušajem igranja. Rad scoreboarda je osigurava klasa *Scoreboard* [58] te dodatne tri klase zadužene za postavljanje novih vrijednosti, spremanje rezultata u JSON datoteku i prikaz UI-a.

Scena Quiz se učitava ispočetka pri svakom prijelazu na novo pitanje. Da bi score tekuće runde sačuvali, postavljen je kao statični [59] atribut. Ukoliko igrač ponudi krivi odgovor, otvara se izbornik za unos imena. Klikom na gumb pozivaju se metode StoreName() iz klase *QuizScore* i AddNewEntry() iz klase *Scoreboard*. Zatim klasa *Scoreboard* poziva metodu GetSavedScores() koja učitava i na ekranu prikazuje prošle rezultate iz .json datoteke, a ukoliko prošli rezultati ne postoje, stvara novu .json datoteku.

Metoda AddNewEntry() uključuje poziv metode AddEntry() (Slika 42) pri čemu su atributi scoreName i score ulazne vrijednosti proslijeđene pomoću konstruktora klase *ScoreboardEntryData*. Zatim metoda provjerava uvjete koji određuju na koje mjesto pripada određeni score. Prvi uvjet pomoću for petlje prolazi kroz listu svih prethodnih rezultata, ukoliko je trenutni rezultat veći od tekućeg elementa s indeksom i, score se upisuje na njegovo mjesto. Drugi uvjet se odnosi na maksimalni broj unosa u tablicu koji je u ovoj igri ograničen na pet unosa. Ukoliko je broj dosadašnjih unosa manji od maksimalnog broja unosa, rezultat se nesmetano sprema. Treći uvjet služi za izbacivanje najmanjeg elementa liste. Ukoliko je trenutni broj upisa veći od maksimalnog broja upisa, metodom RemoveRange() [60] brišemo najmanji, odnosno najslabiji rezultat igranja. Zatim se lista osvježava i prikazuje novi poredak rezultata.

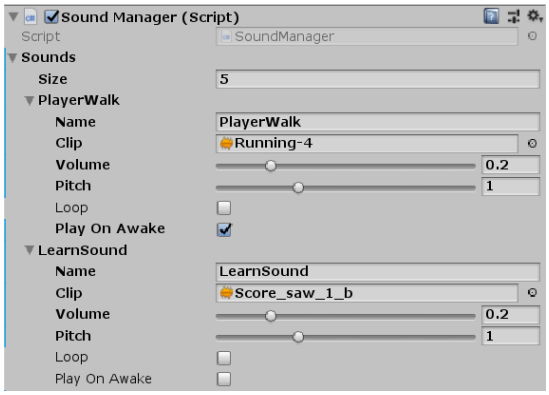
```
36  public void AddEntry(ScoreboardEntryData scoreboardEntryData)
37      {
38          ScoreboardSaveData savedScores = GetSavedScores();
39          bool scoreAdded = false;
40
41          for(int i=0; i < savedScores.highscores.Count; i++)
42          {
43              if(scoreboardEntryData.entryScore > savedScores.highscores[i].entryScore)
44              {
45                  savedScores.highscores.Insert(i, scoreboardEntryData);
46                  scoreAdded = true;
47                  break;
48              }
49          }
50
51          if(!scoreAdded && savedScores.highscores.Count < maxScoreboardEntries)
52          {
53              savedScores.highscores.Add(scoreboardEntryData);
54          }
55
56          if(savedScores.highscores.Count > maxScoreboardEntries)
57          {
58              savedScores.highscores.RemoveRange(maxScoreboardEntries,
59                  savedScores.highscores.Count - maxScoreboardEntries);
60          }
61          UpdateUI(savedScores);
62          SaveScores(savedScores);
63      }
```

Slika 43: Metoda za unos podataka u scoreboard

3.7. Zvuk

Osnovni element ugođaja svake videoigre je zvuk. Zvukovi u igri služe za obavještavanje igrača o akcijama koje se događaju, zaduženi su za ambijent, a pridonose atmosferi koja može biti napeta, opuštajuća, zagonetna,…

Za potrebe igre Ecodrome, u svakoj sceni je dodan objekt SoundManager koji sadrži skriptu *SoundManager* [61]. SoundManager stvara objekt instance klase *Sound* koja uključuje ime, glasnoću, visinu tona te elemente zadužene za vrijeme pokretanje i ponavljanje koje se instancira kao polje. Na taj način dobivamo slobodu dodavanja i namještanja zvukova na jednom mjestu u inspectoru. Nakon postavljanja zvuka, pozivamo ga u ostalim klasama što je vidljivo u kodovima ovog rada, a izvedeno je metodama FindObjectOfType() [62] i Play() [63].



Slika 44 Unos zvukova za cijelu scenu skriptom SoundManager

3.8. Assets

Asseti korišteni u izradi 3D igre Ecodrome su tipa Low Poly, zbog velike količine izvora nije bilo praktično navoditi svaki paket koji je korišten pri izradi i realizaciji mapa. Paketi korišteni u izradi su:

- Character Pack: Free Sample [64] – Lik igrača
- Low Poly Props [65] – Prepreke, smeće i platforme
- Free LowpolyCar [66] – Automobili
- Low-Poly Park [67] – Elementi parka
- Trash Low Poly Cartoon Pack [68] – Smeće u sceni Runner
- Free Low Poly Nature Project Pack 2 Swamp [69] – Ukrasni elementi
- Lowpoly Style Free Rocks and Plants [70] – Ukrasni elementi

Neki od elemenata zvuka su snimljeni pomoću pametnog telefona. Primjer: zvuk za skupljanje smeća po šumi je ostvaren gužvanjem najlonske vrećice.

Pozadinska glazba je kreditirana u samoj igri u rubrici „Credits“ glavnog izbornika u skladu s pravima korištenja zadanim od strane autora.

4. Zaključak

Ovim radom prikazana je izrada edukativne 3D igre u Unity alatu koja se sastoji od tri odvojene igre različitih žanrova. Objašnjeni su elementi kretanja igrača, skupljanja i generiranja objekata, bilježenje postignuća i prenošenje informacija igraču. Veći dio asseta je preuzet s Unity Asset Store-a.

Što se tiče same izrade, ideje i budućnosti igre, moguća proširenja su velika. U konkretnoj izradi moguće je ugraditi veći broj nivoa. Moguće je implementirati kviz u glavni dio igre čime bi igrač mogao otključavati nove dijelove mape ili skrivena mjesta u igri. Nadalje, moguće je dodati i NPC-eve (*Engl. Non-player character*) [71] koji igraču nude odgovarajuće misije čime bi igra dobila elemente RPG (*Engl. Role-playing game*) [72] igre. Glavna ideja igre je bila mape modelirati u obliku Terrain-a [73] što bi uključivalo veći broj detalja i raznolike nivoe ali zbog slabog hardvera korištenog kroz izradu to nažalost nije bilo moguće.

Kroz izradu 3D video igre u Unity alatu uočene su nove mogućnosti u raznim sferama modeliranja i programiranja. Te mogućnosti svaki programer može iskoristiti kao priliku za stjecanje novih vještina ili usavršavanje svojih sposobnosti.

5. Literatura

[1] [Mrežno]. https://en.wikipedia.org/wiki/List_of_video_game_genres#Action [Pristupano 3. Rujan 2019].

[2] [Mrežno]. https://en.wikipedia.org/wiki/List_of_video_game_genres#Strategy [Pristupano 3. Rujan 2019].

[3] [Mrežno]. https://en.wikipedia.org/wiki/List_of_video_game_genres#MMORPG [Pristupano 3. Rujan 2019].

[4] [Mrežno]. https://en.wikipedia.org/wiki/List_of_video_game_genres#Platform_games [Pristupano 3. Rujan 2019].

[5] [Mrežno]. https://en.wikipedia.org/wiki/Platform_game#Endless_running_game [Pristupano 3. Rujan 2019].

[6] [Mrežno]. https://elearningindustry.com/rise-of-educational-games [Pristupano 3. Rujan 2019].

[7] [Mrežno]. https://en.wikipedia.org/wiki/Arcade_game [Pristupano 3. Rujan].

[8] [Mrežno]. http://www.dorkly.com/post/87115/educational-games-ranked [Pristupano 3. Rujan 2019].

[9] [Mrežno]. https://en.wikipedia.org/wiki/Unity_(game_engine) [Pristupano 3. Rujan 2019].

[10] [Mrežno]. https://visualstudio.microsoft.com/ [Pristupano 3. Rujan 2019].

[11] [Mrežno]. https://www.audacityteam.org/ [Pristupano 3. Rujan 2019].

[12] [Mrežno]. https://www.blender.org/ [Pristupano 3. Rujan 2019].

[13] [Mrežno]. https://www.adobe.com/products/photoshop.html# [Pristupano 3. Rujan 2019].

[14] [Mrežno]. https://en.wikipedia.org/wiki/List_of_video_game_genres#Trivia_game [Pristupano 3. Rujan 2019].

[15] [Mrežno]. https://docs.unity3d.com/530/Documentation/ScriptReference/UI.Button-onClick.html [Pristupano 3. Rujan 2019].

[16] [Mrežno]. https://docs.unity3d.com/ScriptReference/GameObject.SetActive.html [Pristupano 3. Rujan 2019].

[17] [Mrežno]. https://docs.unity3d.com/ScriptReference/PlayerPrefs.html [Pristupano 3. Rujan 2019].

[18] [Mrežno]. https://docs.unity3d.com/Manual/class-CapsuleCollider.html [Pristupano 3. Rujan 2019].

[19] [Mrežno]. https://docs.unity3d.com/Manual/Prefabs.html [Pristupano 3. Rujan 2019].

[20] [Mrežno]. https://docs.unity3d.com/ScriptReference/CharacterController.html [Pristupano 3. Rujan 2019].

[21] [Mrežno]. https://www.youtube.com/watch?v=riPZtrWNGzc [Pristupano 3. Rujan 2019].

[22] [Mrežno]. https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnCollisionEnter.html [Pristupano 3. Rujan 2019].

[23] [Mrežno]. https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnTriggerEnter.html [Pristupano 3. Rujan 2019].

[24] [Mrežno]. https://en.wikipedia.org/wiki/Low_poly [Pristupano 3. Rujan 2019].

[25] [Mrežno]. https://docs.unity3d.com/Manual/class-BoxCollider.html [Pristupano 3. Rujan 2019].

[26] [Mrežno]. https://docs.unity3d.com/ScriptReference/Renderer.html [Pristupano 3. Rujan 2019].

[27] [Mrežno]. https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnTriggerExit.html [Pristupano 3. Rujan 2019].

[28] [Mrežno]. https://docs.unity3d.com/ScriptReference/MonoBehaviour.Awake.html [Pristupano 3. Rujan 2019].

[29] [Mrežno]. https://docs.unity3d.com/ScriptReference/Physics.CheckSphere.html [Pristupano 3. Rujan 2019].

[30] [Mrežno]. https://docs.unity3d.com/ScriptReference/Object.Instantiate.html [Pristupano 3. Rujan 2019].

[31] [Mrežno]. http://www.arounddb.com/news/hong-kongers-join-forces-for-massive-beach-clean-up-effort-in-discovery-bay/ [Pristupano 3. Rujan 2019].

[32] [Mrežno]. https://docs.unity3d.com/ScriptReference/MonoBehaviour.Start.html [Pristupano 3. Rujan 2019].

[33] [Mrežno]. https://docs.unity3d.com/ScriptReference/Array-length.html [Pristupano 3. Rujan 2019].

[34] [Mrežno]. https://docs.unity3d.com/ScriptReference/MonoBehaviour.Update.html [Pristupano 3. Rujan 2019].

[35] [Mrežno]. https://docs.unity3d.com/ScriptReference/Transform.html [Pristupano 3. Rujan 2019].

[36] [Mrežno]. https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnMouseDown.html [Pristupano 3. Rujan 2019].

[37] [Mrežno]. https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnMouseUp.html [Pristupano 3. Rujan 2019].

[38] [Mrežno]. https://docs.unity3d.com/ScriptReference/SpriteRenderer.html [Pristupano 3. Rujan 2019].

[39] [Mrežno]. https://www.youtube.com/watch?v=Jwoo1UJf4bQ [Pristupano 3. Rujan 2019].

[40] [Mrežno]. https://docs.unity3d.com/ScriptReference/Animator.html [Pristupano 3. Rujan 2019].

[41] [Mrežno]. https://youtu.be/w3nywthMvQA [Pristupano 3. Rujan 2019].

[42] [Mrežno]. https://docs.unity3d.com/ScriptReference/Input.GetAxisRaw.html [Pristupano 3. Rujan 2019].

[43] [Mrežno]. https://docs.unity3d.com/ScriptReference/Camera.html [Pristupano 3. Rujan 2019].

[44] [Mrežno]. https://youtu.be/MFQhpwc6cKE [Pristupano 3. Rujan 2019].

[45] [Mrežno]. https://youtu.be/HIsEqKPoJXM [Pristupano 3. Rujan 2019].

[46] [Mrežno]. https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnControllerColliderHit.html [Pristupano 3. Rujan 2019].

[47] [Mrežno]. https://youtu.be/daWbDGhXScM [Pristupano 3. Rujan 2019].

[48] [Mrežno]. https://docs.unity3d.com/ScriptReference/Vector3.html [Pristupano 3. Rujan 2019].

[49] [Mrežno]. https://docs.unity3d.com/ScriptReference/MonoBehaviour.StartCoroutine.html [Pristupano 3. Rujan 2019].

[50] [Mrežno]. https://youtu.be/G9jHLRqdWyU [Pristupano 3. Rujan 2019].

[51] [Mrežno]. https://docs.unity3d.com/Manual/UICanvas.html [Pristupano 3. Rujan 2019].

[52] [Mrežno]. https://docs.unity3d.com/ScriptReference/Random.Range.html [Pristupano 3. Rujan 2019].

[53] [Mrežno]. https://docs.unity3d.com/ScriptReference/Input.GetKeyDown.html [Pristupano 3. Rujan 2019].

[54] [Mrežno]. https://docs.unity3d.com/ScriptReference/Time-timeScale.html [Pristupano 3. Rujan 2019].

[55] [Mrežno]. https://docs.unity3d.com/Manual/JSONSerialization.html [Pristupano 3. Rujan 2019].

[56] [Mrežno]. https://youtu.be/g_Ff1SPhidg [Pristupano 3. Rujan 2019].

[57] [Mrežno]. https://docs.unity3d.com/Manual/script-InputField.html [Pristupano 3. Rujan 2019].

[58] [Mrežno]. https://youtu.be/FSEbPxf0kfs [Pristupano 3. Rujan 2019].

[59] [Mrežno]. https://unity3d.com/fr/learn/tutorials/topics/scripting/statics [Pristupano 3. Rujan 2019].

[60] [Mrežno]. https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1.removerange?view=netframework-4.8 [Pristupano 3. Rujan 2019].

[61] [Mrežno]. https://youtu.be/6OT43pvUyfY [Pristupano 3. Rujan 2019].

[62] [Mrežno]. https://docs.unity3d.com/ScriptReference/Object.FindObjectOfType.html [Pristupano 3. Rujan 2019].

[63] [Mrežno]. https://docs.unity3d.com/ScriptReference/AudioSource.Play.html [Pristupano 3. Rujan 2019].

[64] [Mrežno]. https://assetstore.unity.com/packages/3d/characters/humanoids/character-pack-free-sample-79870 [Pristupano 27. Kolovoz 2019].

[65] [Mrežno]. https://assetstore.unity.com/packages/3d/props/industrial/low-poly-props-pack-53751 [Pristupano 27. Kolovoz 2019].

[66] [Mrežno]. https://assetstore.unity.com/packages/3d/vehicles/free-lowpolycar-127921 [Pristupano 27. Kolovoz 2019].

[67] [Mrežno]. https://assetstore.unity.com/packages/3d/environments/urban/low-poly-park-61922 [Pristupano 27. Kolovoz 2019].

[68] [Mrežno]. https://assetstore.unity.com/packages/3d/trash-low-poly-cartoon-pack-66229 [Pristupano 27. Kolovoz 2019].

[69] [Mrežno]. https://assetstore.unity.com/packages/3d/environments/landscapes/free-low-poly-nature-project-pack-2-swamp-123765 [Pristupano 27. Kolovoz 2019].

[70] [Mrežno]. https://assetstore.unity.com/packages/3d/environments/landscapes/lowpoly-style-free-rocks-and-plants-145133 [Pristupano 27. Kolovoz 2019].

[71] [Mrežno]. https://www.techopedia.com/definition/1920/non-player-character-npc [Pristupano 27. Kolovoz 2019].

[72] [Mrežno]. https://en.wikipedia.org/wiki/List_of_video_game_genres#Role-playing [Pristupano 27. Kolovoz 2019].

[73] [Mrežno]. https://docs.unity3d.com/Manual/script-Terrain.html [Pristupano 27. Kolovoz 2019].

6. Popis slika

7. Prilozi

U prilogu rada je CD koji sadrži ovaj dokument, Unity projekt i zadnju verziju buildane igre.