Razvoj avanturističko-logičke računalne igre

Katić, Ana-Marija

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

RAZVOJ AVANTURISTIČKO-LOGIČKE RAČUNALNE IGRE

Završni rad

Ana-Marija Katić

Osijek, 2021.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 04.09.2021.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Ana-Marija Katić
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4216, 24.07.2018.
OIB studenta:	65275575583
Mentor:	Izv. prof. dr. sc. Josip Balen
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Razvoj avanturističko-logičke računalne igre
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	04.09.2021.
Datum potvrde ocjene Odbora:	08.09.2021.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 09.09.2021.

Ime i prezime studenta:	Ana-Marija Katić
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4216, 24.07.2018.
Turnitin podudaranje [%]:	4

Ovom izjavom izjavljujem da je rad pod nazivom**: Razvoj avanturističko-logičke računalne igre**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Josip Balen

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Razvoj računalnih igara započeo je krajem 1940-ih kao način predstavljanja novih mogućnosti računala, a prva igra koja je predstavljana javnosti je „Tenis za dvoje“ (engl. „*Tennis for two*“) iz 1958. godine. Današnje tržište računalnih igara ima značajnu ulogu u zabavnoj industriji što ga čini važnim gospodarskim čimbenikom mnogih zemalja [1]. Računalne igre, odnosno videoigre, predstavljaju interaktivne igre koje se definiraju prema korištenoj digitalnoj platformi (engl. *computing/digital platform*), gdje se razlikuju osobna računala, igraće konzole, mobiteli, igraći automati, sustavi virtualne i proširene stvarnosti. Videoigre su raspodijeljene u širok raspon žanrova ovisno o tipu vrste igranja i njihovoj svrsi [2].

Za razvoj avanturističko-logičke računalne igre potrebna su znanja programiranja i korištenja računalnih tehnologija. Avanturističko-logička računalna igra se temelji na rješavanju logičkih zadataka koji otvaraju nove logičke zagonetke i proširuju priču igre. Uz zabavu, svrha takvih igara je rješavanje logičkih slagalica što pomaže igračima razviti sposobnosti za brže rješavanja problema i sposobnosti logičkog zaključivanja [3].

U završnom radu opisan je razvoj avanturističko-logičke igre Crystal Hunter. Cilj igre je rješavajući logičke slagalice sakupiti tri kristala i izaći iz špilja u kojima se oni nalaze. Na putu kroz špilje igrač, odnosno lovac, nailazi na vodiče koji mu daju savjete za prolazak kroz špilje i pronalazak traženih kristala. Prolaskom kroz špilje igrač nailazi na kamene blokove i čudovišta koje može pomicati i lokve koje mu otežavaju put. Prolazak kroz razine ograničen je brojem koraka koje lovac može napraviti te težina raste prolaskom kroz svaku špilju. Igra je inspirirana strateškom igrom šah, serijalom igara „Fireboy and Watergirl“ koje je razvio Oslo Albet i igrom „Helltaker“ koju je razvio Łukasz Piskorz. Ove igre odlikuje potreba za strateškim razmišljanjem i sposobnosti logičkog zaključivanja što se također htjelo postići razvojem igre Crystal Hunter.

Računalna igra Crystal Hunter razvijena je pomoću više programa, kao što su Unity, GIMP i Microsoft Visual Studio. Oni su teorijski objašnjeni u drugom poglavlju. Objašnjen je razlog njihova korištenja i način primjene. U trećem poglavlju prikazan je postupak i navedena svrha razvoja avanturističko-logičke igre Crystal Hunter. Opisan je postupak izrade vizualnog dizajna objekata i pozadina, dizajna logičkih slagalica, grafičkog sučelja te implementacije logičke strukture igre. Četvrto poglavlje sadrži prikaz cjelokupne priče na kojoj se temelji avanturističko-logička igre Crystal Hunter. Također, prikazan je postupak igranja igre, odnosno prikazane su sve razine igre u nasumičnim trenucima igranja.

1.1. Zadatak završnog rada

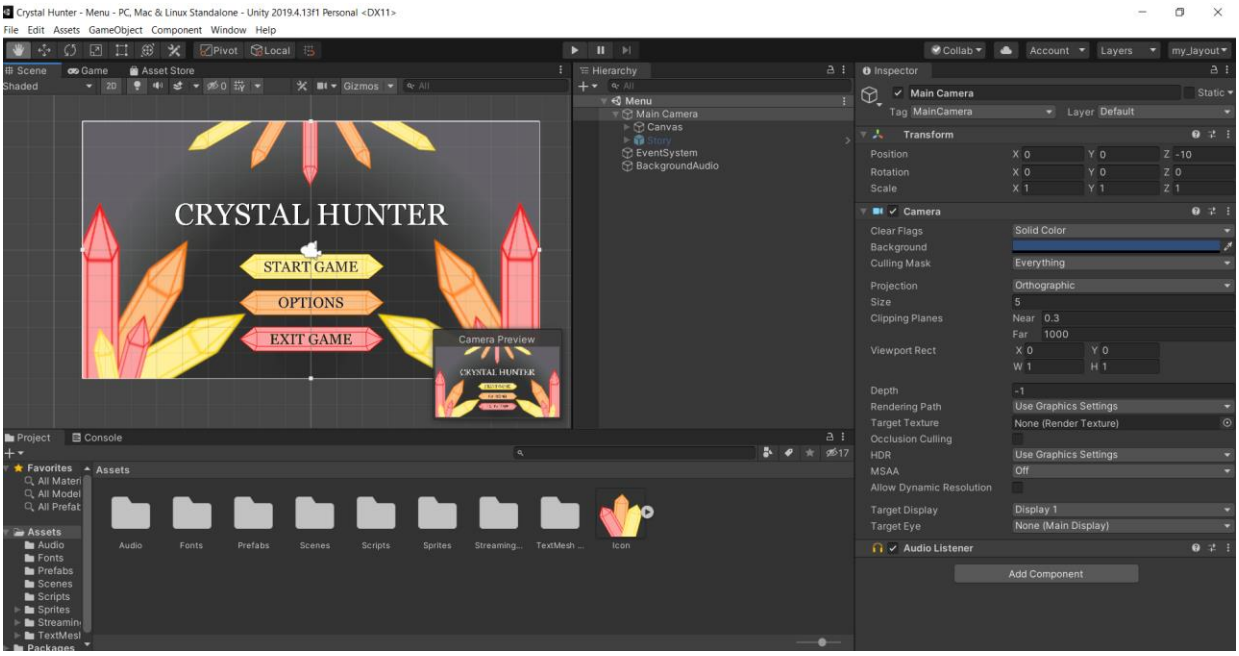
Zadatak završnog rada je osmisliti, opisati i razviti avanturističku (pustolovnu) logičku igru za osobna računala koristeći Unity i programski jezik C#. Igra se treba temeljiti na rješavanju logičkih slagalica (engl. *puzzle*) prolazeći kroz različite težinske razine sve do cilja. Potrebno je osmisliti i izraditi cjeloviti vizualni dizajn poput objekata i pozadina te kompletnu logičku strukturu računalne igre.

2. KORIŠTENE TEHNOLOGIJE

Razvoj računalne igre popraćen je korištenjem tehnologija kao što su Unity, Microsoft Visual Studio i GIMP programi koji su teorijski opisani u ovom poglavlju.

2.1. Unity višeplatformski softver za razvoj računalnih igara

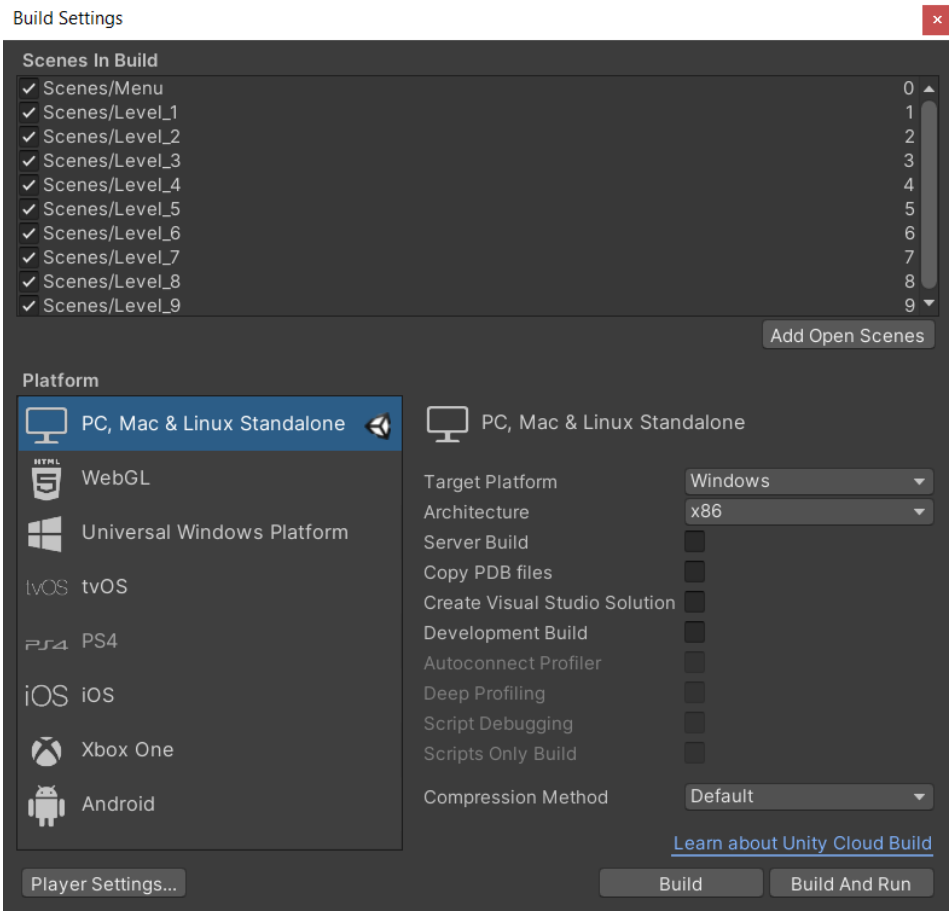
Unity je višeplatformski (engl. *cross-platform*) softver za razvoj igara kojeg je 2005. godine razvila tvrtka Unity Technologies [4]. Kreiranje igara pomoću Unityja, koristeći C# programski jezik, podržano je za više od 20 različitih platformi. Zbog velikog broja podržanih platformi Unity se koristi za razvoj različitih vrsta sadržaja. Uz stvaranje 2D i 3D sadržaja Unity omogućava stvaranje sadržaja virtualne i proširene stvarnosti [5]. Osim za razvoj igara, Unity se koristi za 3D dizajn i simulacije u drugim industrijama kao što su filmska, automobilska industrija, u arhitekturi, te za stvaranje 60% svih sadržaja za iskustvo proširene i virtualne stvarnosti [6]. Popularnosti Unityja pridonosi njegova cijena, odnosno on je besplatan za sve samostalne korisnike koji ne zarađuju više od tisuću američkih dolara godišnje koristeći softver i za korisnike koji ga koriste u edukacijske svrhe. Za korištenje Unityja u komercijalne svrhe potrebna je kupovina licence [5]. Slika 2.1. predstavlja snimku zaslona koja prikazuje izgled rasporeda stavki Unity sučelja.



Slika 2.1. Izgled Unity sučelja

Unity je korišten kao glavni program za razvoj računalne igre jer sadrži različite značajke poput komponenti, imenika i razreda, koje pojednostavljuju razvoj računalne igre i smanjuju količinu programskog koda [7]. U njemu su ukomponirani svi elementi potrebni za razvoj igre, poput

medijskih datoteka. Računalna igra Crystal Hunter napravljena je u 2D načinu stvaranja projekta za Windows, Mac OS i Linux operacijske sustave (Slika 2.2). Korišteni objekti izrađeni su pomoću programa GIMP, a skripte su napisane u razvojnom okružju Microsoft Visual Studio.

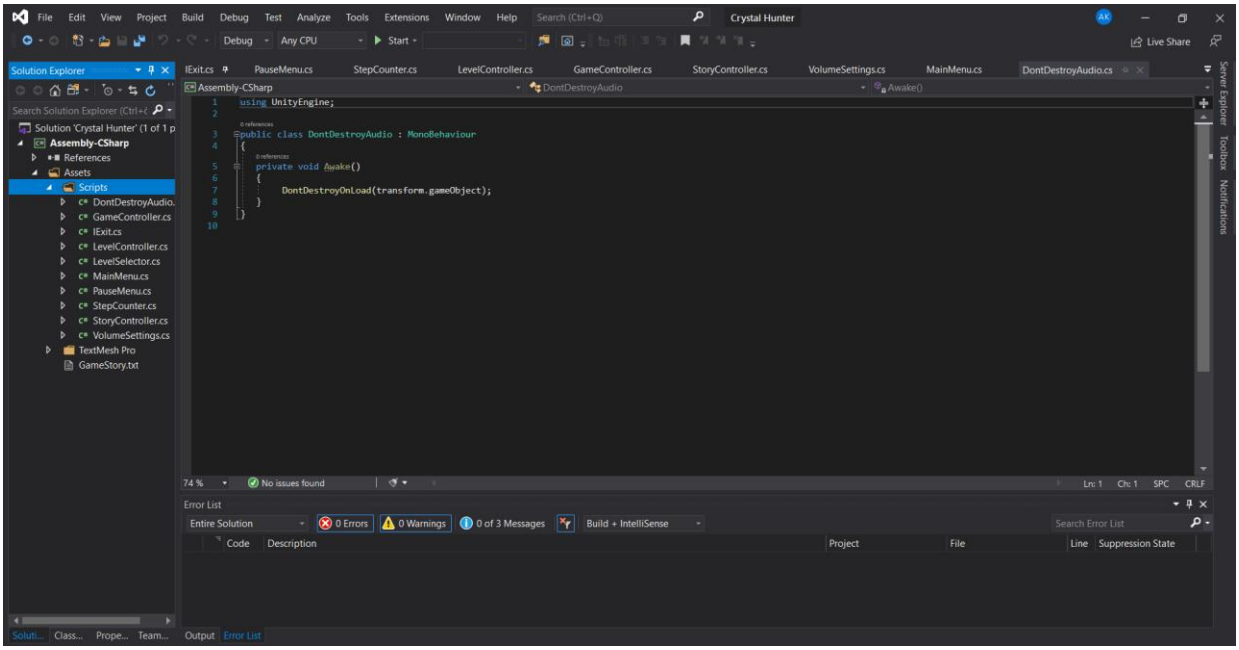


Slika 2.2. Odabir razvoja za Windows, Mac OS i Linux operacijske sustave

2.2. Microsoft Visual Studio

Microsoft Visual Studio je integrirano razvojno okruženje (engl. *integrated development environment*) kojeg je razvila tvrtka Microsoft. Služi za razvoj računalnih programa, mrežnih i mobilnih aplikacija, te mrežnih usluga i stranica. Visual Studio podržava 36 različitih programskih jezika i omogućava podržavanje ostalih jezika putem dodataka. Neki od ugrađenih jezika su C, C++, C#, JavaScript, XML, HTML, CSS. Najjednostavnija verzija Visual Studija, koja je dostupna studentima i pojedinačnim programerima, besplatna je za korištenje, dok se dodaci trebaju platiti [8].

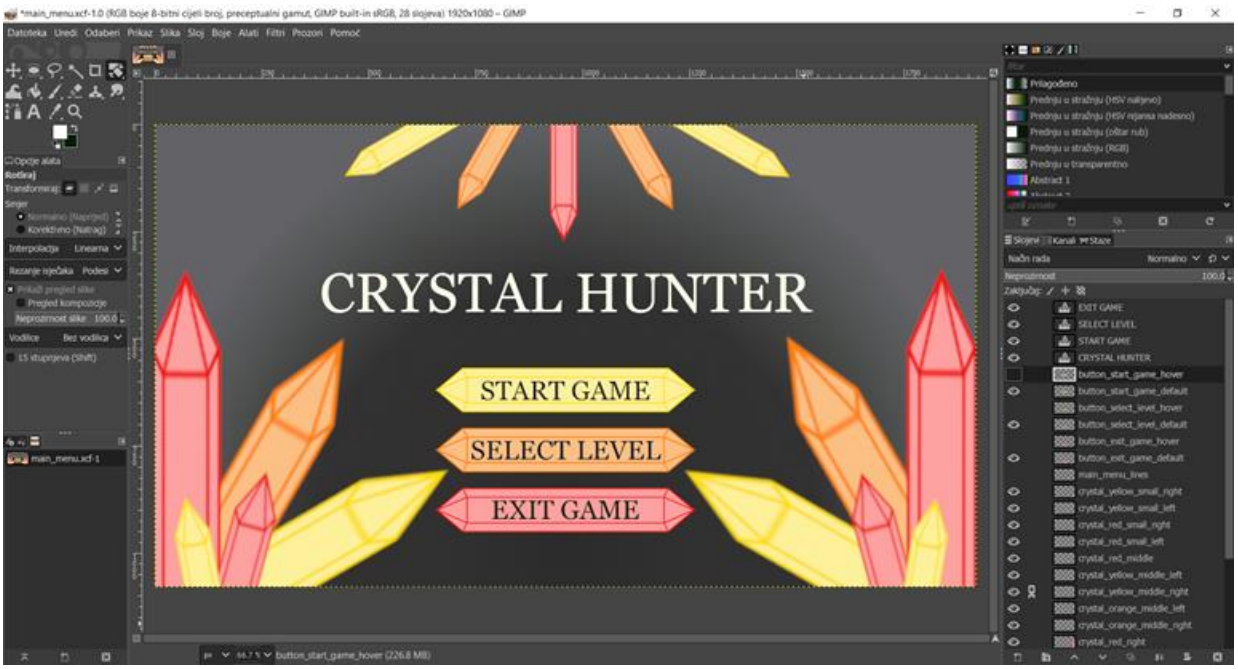
Microsoft Visual Studio je korišten za izradu skripti koje se koriste u Unityju za rukovanje objektima i opisivanje logičke strukture igre. Programski kod skripti pisan je objektno orijentiranim programskim jezikom C#. Slika 2.3. prikazuje snimku zaslona Visual Studio sučelja.



Slika 2.3. Izgled Visual Studio sučelja

2.3. GIMP višeplatformski uređivač slika

GNU Image Manipulation Program (GIMP) je višeplatformski uređivač slika [9]. GIMP je besplatan i omogućava retuširanje i uređivanje slika, slobodno crtanje, mijenjanje formata slikovnih datoteka i druge specijalizirane zadatke. Neki od formata koje GIMP podržava su PNG, GIF, JPEG, BMP [10]. Snimkom zaslona prikazan je rasporeda stavki GIMP sučelja (Slika 2.4.).



Slika 2.4. Izgled GIMP sučelja

GIMP posjeduje alate za visokokvalitetno uređivanje slika i omogućava umjetnicima transformiranje slika u jedinstvena umjetnička djela. Uz uređivanje i crtanje GIMP se koristi za izradu ikona, elemenata grafičkog dizajna i za dizajn komponenata korisničkog sučelja [9]. U ovom radu GIMP je korišten za grafički dizajn korisničkog sučelja, izradu modela objekata i pozadine. U GIMP-u je napravljena paleta boja koja je korištena za dizajn svih modela, a oni su u obliku 2D grafičkih objekata (engl. *sprites*) implementirani u Unity.

3. RAZVOJ RAČUNALNE IGRE

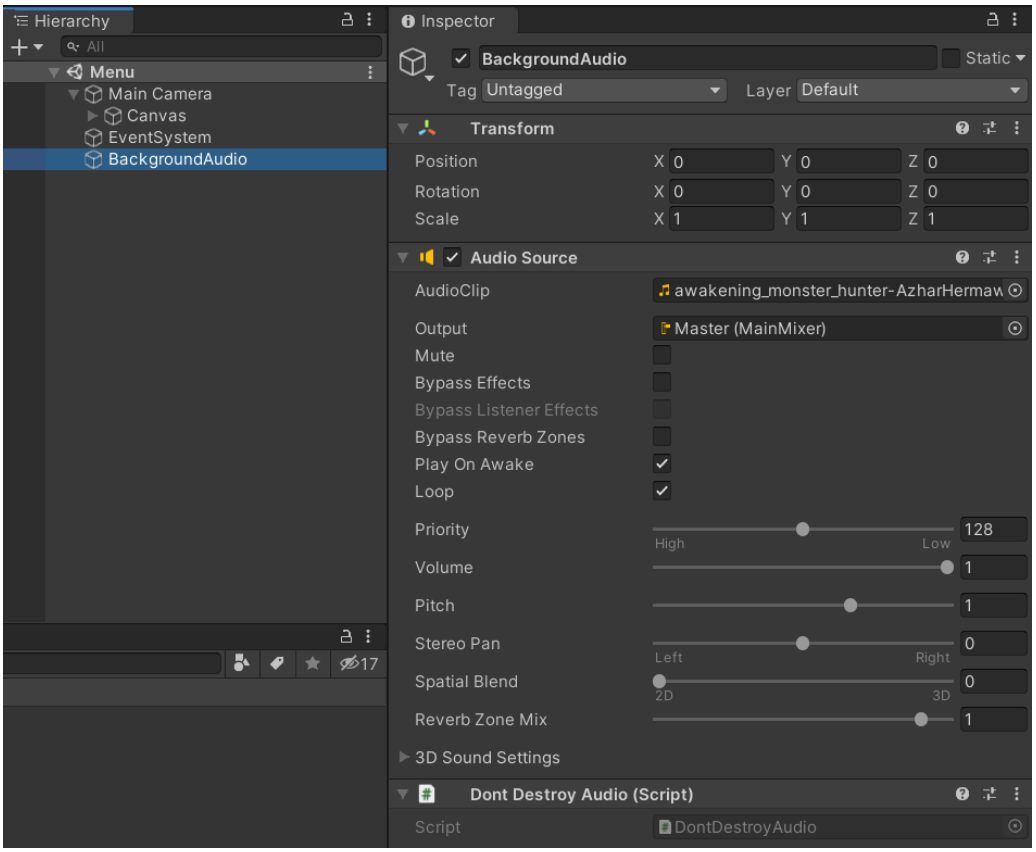
3.1. Ideja računalne igre Crystal Hunter

Osnovna ideja u avanturističko-logičkoj igri Crystal Hunter je sakupiti tri kristala koji se nalaze u različitim špiljama. Glavni lik, kojim igrač (engl. *player*) upravlja, predstavlja lovca na kristale (engl. *crystal hunter*) koji prolazi kroz špilje s ciljem sakupljanja svih kristala. Pokušavajući sakupiti kristale susreće vodiče (engl. *guides*) koji mu pomažu u njihovom pronalasku i pokazuju izlaz iz špilja. Put kroz špilje nije jednostavan jer igrač mora riješiti devet logičkih zagonetki. Igra se sastoji od ukupno tri špilje, a svaka špilja se sastoje od tri razine. Težina rješenja logičkih slagalica postepeno raste prolaskom lovca kroz špilju. Lovac ima određeni broj koraka koji se smanjuje kada igrač pritisne tipke A, W, S, D ili tipke sa strelicama. Lovac se ne može dijagonalno kretati već se isključivo kreće okomito ili vodoravno. Prolazeći kroz razine igrač se susreće s novim objektima koji čine slagalice. Ulaskom u špilju igrač susreće vodiče koji mu daju savjete potrebne za rješavanje logičkih slagalica i pronalazak kristala. U prvim dijelovima špilja igrač nailazi na kamene blokove koji se mogu pomicati. Pomičući kamene blokove lovcu se broj koraka dodatno smanjuje zbog snage koju mora utrošiti kako bi ih pomaknuo. Lovac kamene blokove može pomicati samo ako se iza njih ništa ne nalazi. Daljnjim kretanjem kroz špilju lovac nailazi na kamena čudovišta. Kamena čudovišta se ponašaju slično kamenim blokovima, ali kada su gurnuta u druge objekte ona se skamene što omogućava lovcu da pređe preko njih. Guranjem čudovišta lovac također gubi dodatan korak. Rješavanjem razina s kamenim blokovima i čudovištima lovac pronalazi kristale i preostaje mu samo pronaći izlaz iz špilje. Tražeći izlaz, osim što mora gurati kamene blokove i čudovišta, na tlu špilje pronalazi lokve mu otežavaju kretanje. Osim što ulaskom i izlaskom lovac gubi jedan korak, on gubi korake provodeći vrijeme u lokvi, odnosno gubi dva koraka ako ga igrač ostavi u lokvi tijekom pauziranja i povratka u igru. Zbog toga igrač treba obratiti pažnju gdje uzima pauzu dok rješava razine. Ako lovac ostane bez koraka on je vraćen na početak razine. Svrha izrade ove igre je omogućiti igraču da se zabavi rješavajući logičke slagalice te igrajući razvija svoje sposobnosti logičkog zaključivanja i bržeg rješavanja logičkih problema.

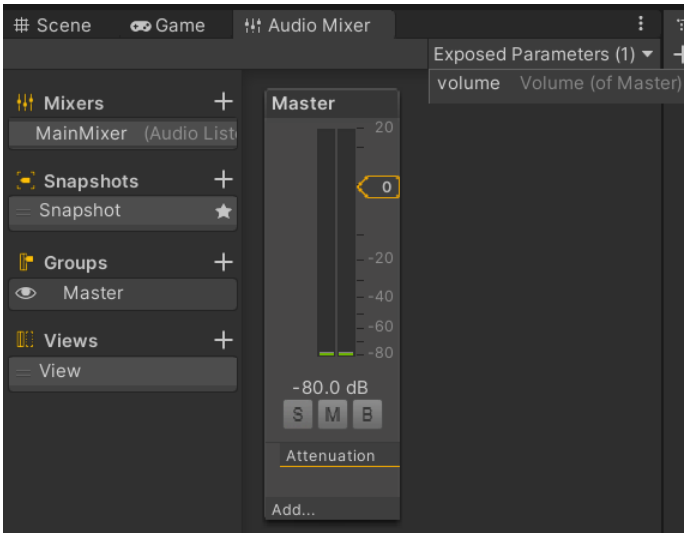
3.2. Korisničko sučelje

Za izradu grafičkog korisničkog sučelja korišten je GIMP, program za obradu slika. Pomoću GIMP-a dizajnirani su glavni izbornik, izbornik opcija, izbornik pauze te sve razine i objekti. U igru je umetnuta pozadinska glazba koja svira u svim izbornicima i razinama tijekom igranja igre.

Sceni *Menu* dodan je objekt pozadinske glazbe kojemu je pridružena komponenta izvor zvuka (engl. *audio source*) (Slika 3.1.). Kao audio isječak korištena je skladba „Awakening Monster Hunter“ koju je izdao Azhar Hermawan [11]. Za izlaz (engl. *output*) zvuka postavljen je audio mikser naziva *MainMixer* (Slika 3.2.). Parametar glasnoće zvuka audio miksera postavljen je kao javan kako bi bio dostupan za promjene.



Slika 3.1. Prikaz objekta BackgroundAudio i njegove komponente Audio Source



Slika 3.2. Izgled audio miksera MainMixer

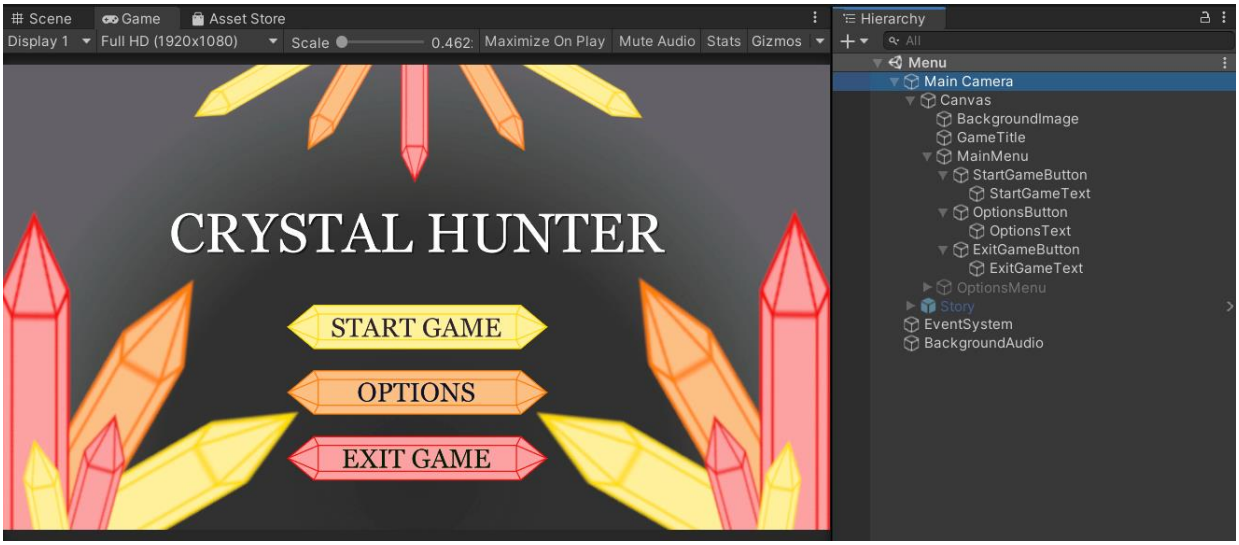
Objektu *BackgroundAudio* pridružena je skripta *DontDestroyAudio* koja omogućava nastavak audio isječka neovisno o sceni koja je u određenom trenutku učitana. Pridruživanjem te skripte spriječen je prekid i ponovno pokretanje audio isječka nakon učitavanja nove scene. Pokretanjem igre poziva se metoda *Awake* koja sadrži funkciju *DontDestroyOnLoad*. Ona omogućava nastavak sviranja audio isječka nakon učitavanja nove scene (Programski kod 3.1.).

```
1   using UnityEngine;
2
    0 references
3   public class DontDestroyAudio : MonoBehaviour
4   {
        0 references
5       private void Awake()
6       {
7           DontDestroyOnLoad(transform.gameObject);
8       }
9   }
```

Programski kod 3.1. Skripta DontDestroyAudio

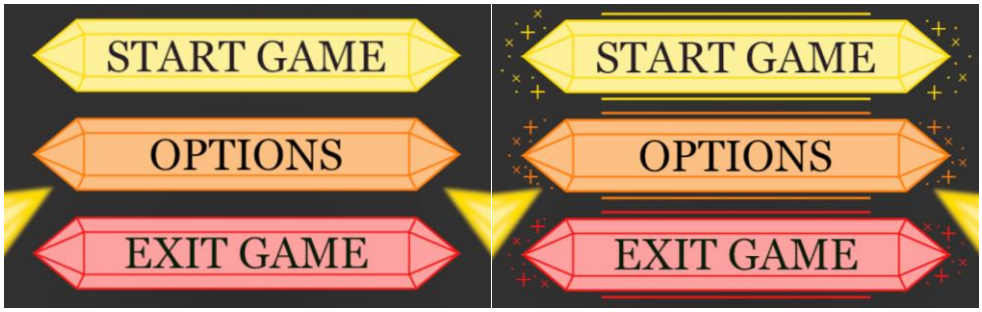
3.2.1. Glavni izbornik

Prva scena predstavlja scenu izbornika i nazvana je *Menu*, odnosno izbornik. Ona se prva otvara kada se pokrene igra i sadrži dva izbornika od kojih je vidljiv glavni izbornik. Osim izbornika, prva scena uključuje objekt koji sadrži početak priče. Glavni izbornik (engl. *main menu*) omogućava pokretanje igre, otvaranje izbornika opcija i napuštanje igre (Slika 3.3.).



Slika 3.3. Izgled glavnog izbornika i prikaz hijerarhije objekata

Funkcije koje omogućava glavni izbornik implementirane su pomoću tipki. Glavni izbornik se sastoji od tipki *START GAME*, *OPTIONS* i *EXIT GAME*. Na slici 3.4. vidi se promjena stanja svih tipki kada se pokazivačem prijeđe preko njih.



Slika 3.4. Lijevo izgled obične tipke, desno izgled tipke kada se pokazivačem pređe preko nje

Pritiskom tipke *START GAME* pokreće se igra, odnosno objekt priče postaje vidljiv. Pritiskom tipke *EXIT GAME* napušta se igra. Tipki *EXIT GAME* je pridružena skripta *MainMenu* s metodom *ExitGame* (Programski kod 3.2.). U skriptu *MainMenu* je uključen imenik *Assets.Scripts* kako bi se mogla primijeniti implementacija sučelja.

```
1   using UnityEngine;
2   using Assets.Scripts;
3
    0 references
4   public class MainMenu : MonoBehaviour, IExit
5   {
        4 references
6       public void ExitGame()
7       {
8           Debug.Log("QUIT!");
9           Application.Quit();
10      }
11  }
```

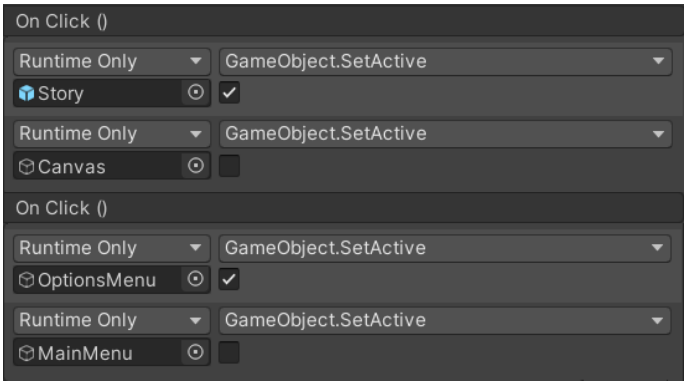
Programski kod 3.2. Skripta MainMenu

Klasa *MainMenu* implementira sučelje *IExit*, a ono definira da sve klase koje ga implementiraju moraju imati metodu *ExitGame* [12] (Programski kod 3.3.). Klikom na tipku *EXIT GAME* poziva se metoda *ExitGame* koja služi za napuštanje igre. Zbog toga što tijekom razvoja igre nije vidljivo napuštanje igre korišten je *Debug.Log* s porukom za dobivanje povratne informacije.

```
1   namespace Assets.Scripts
2   {
        3 references
3       public interface IExit
4       {
            4 references
5           void ExitGame();
6       }
7   }
```

Programski kod 3.3. Skripta IExit

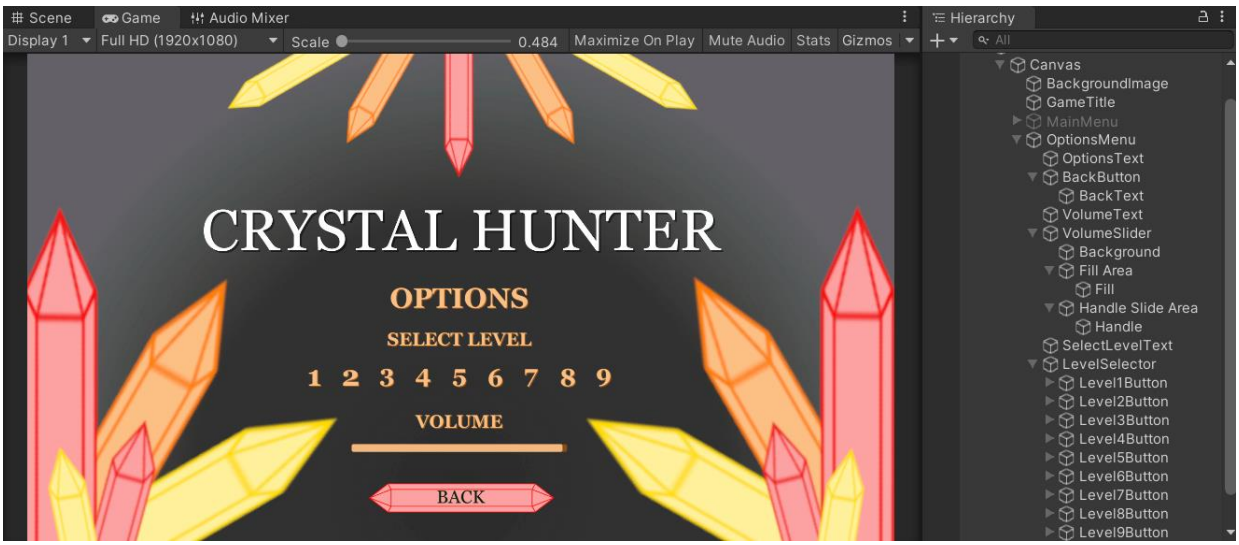
Za postavljanje funkcionalnosti tipki *START GAME* i *OPTIONS* korištena je ugrađena značajka Unityja koja omogućava pridruživanje funkcije *SetActive* objektu *GameObject*. Pritiskom tipke *START GAME* objekt *Story* se postavlja kao aktivan, vidljiv, a objekt *Canvas*, koji obuhvaća sve izbornike, postavlja kao neaktivna, nevidljiv. U slučaju pritiska tipke *OPTIONS* objekt *OptionsMenu* se postavlja kao aktivan, odnosno postaje vidljiv, a objekt *MainMenu* postaje nevidljiv, neaktivan (Slika 3.5.).



Slika 3.5. Postavljanje funkcionalnosti tipki START GAME i OPTIONS

3.2.2. Izbornik opcija

Klikom na tipku *OPTIONS* objekt *OptionsMenu* postaje vidljiv, aktivan. U izborniku opcija igrač može izabrati razinu koju želi igrati ako ne želi početi od početka. Odabirom razine preskače monolog vodiča te odmah odlazi na razinu. Osim odabira razine, igrač može podesiti glasnoću pozadinske glazbe te se može vratiti u glavni izbornik (Slika 3.6.).

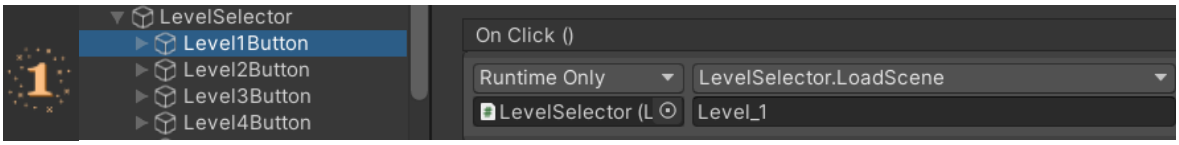


Slika 3.6. Izgled izbornika opcija i prikaz hijerarhije objekata

Za odabir razine objektu *LevelSelector* pridružena je istoimena skripta (Programski kod 3.4.). Tipke predstavljene brojevima razina mijenjaju izgled kada se pokazivačem pređe preko njih (Slika 3.7.). Tim tipkama pridružena je metoda *LoadScene* za učitavanje scena. Ona prima parametar ime scene, a poziva se klikom na tipku broja razine u kojoj je postavljena vrijednost imena scene. Uključen je imenik *UnityEngine.SceneManagement* za upravljanje scenama. Imena scena postavljena su za sve tipke s brojevima razina prema primjeru na slici 3.7.

```
1  using UnityEngine;
2  using UnityEngine.SceneManagement;
3
   0 references
4  public class LevelSelector : MonoBehaviour
5  {
       0 references
6      private void LoadScene(string sceneName)
7      {
8          SceneManager.LoadScene(sceneName);
9      }
10 }
```

Programski kod 3.4. Skripta LevelSelector



Slika 3.7. Izgled tipke za odabir razine kada se pokazivačem pređe preko nje i postavljanje vrijednosti imena scene

Uz odabir razine u izborniku opcija igrač može prilagoditi glasnoću pozadinske glazbe koristeći klizač kojemu je pridružena skripta *VolumeSettings* (Programski kod 3.5.). Vrijednost klizača postavljena je na raspon vrijednosti od 0.0001 do 1 gdje je početna vrijednost postavljena na 1. Te vrijednosti su odabrane kako bi se provedbom skripte dobile vrijednosti koje ljudi mogu čuti.

```
1  using UnityEngine;
2  using UnityEngine.Audio;
3
   0 references
4  public class VolumeSettings : MonoBehaviour
5  {
6      public AudioMixer mAudioMixer;
7
       0 references
8      private void SetVolume(float volume)
9      {
10         mAudioMixer.SetFloat("volume", Mathf.Log10(volume) * 20);
11     }
12 }
```

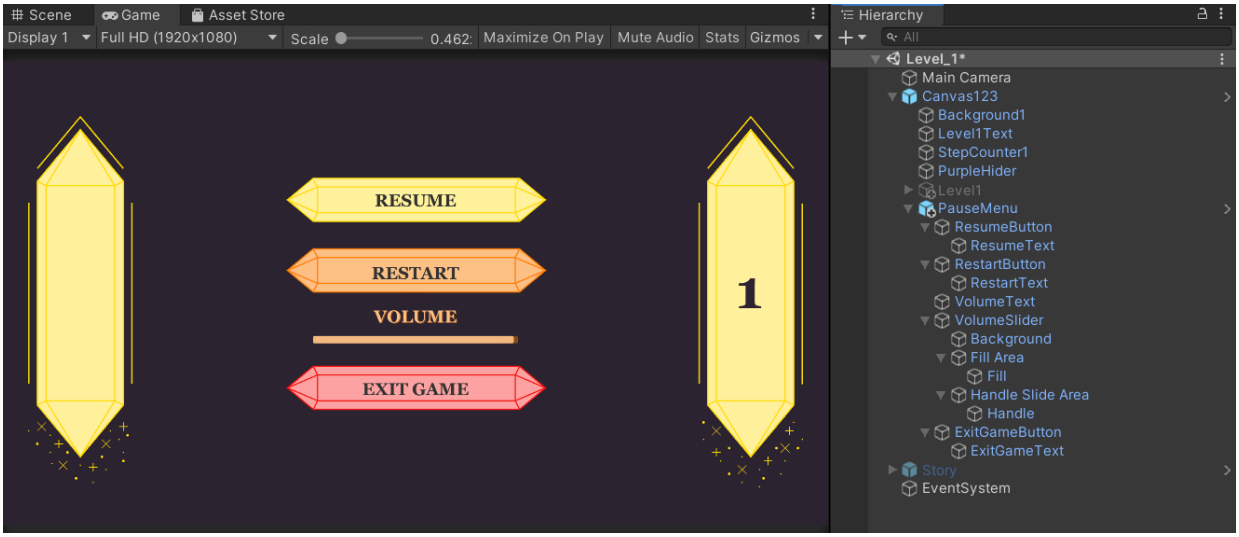
Programski kod 3.5. Skripta VolumeSettings

U skriptu je uključen imenik *UnityEngine.Audio* kako bi se mogla promijeniti glasnoća glazbe. Na promjenu vrijednosti na klizaču poziva se metoda *SetVolume* koja kao parametar uzima *volume* koji je javni parametar audio miksera. Vrijednost parametra *volume* postavlja se pomoću logaritamske funkcije koja je množena s dvadeset kako bi se dobile vrijednosti koje osoba može čuti.

Osim odabira razine i promjene glasnoće glazbe igrač se može vratiti u glavni izbornik klikom na tipku *BACK*. Funkcionalnost tipke *BACK* postavljena je na isti način kao za tipku *OPTIONS*. Razlikuju se po tome što pritiskom tipke *BACK* objekt *MainMenu* postaje aktivan, a objekt *OptionsMenu* neaktivan.

3.2.3. Izbornik pauze

Izborniku pauze igrač pristupa pritiskom tipke *Escape* dok se nalazi u scenama razina igre. Izbornik pauze (engl. *pause menu*) omogućava nastavak igranja razine, ponovno pokretanje razine, promjenu glasnoće pozadinske glazbe i napuštanje igre. Objekt *PauseMenu* napravljen je kao *prefab* objekt što znači da se može koristiti u drugim scenama te je jednak za sve scene. Sastoji se od tipki *RESUME*, *RESTART* i *EXIT GAME* i klizača *VOLUME* (Slika 3.8.).



Slika 3.8. Izgled izbornika pauze i prikaz hijerarhije objekata

Funkcionalnost tipke *RESUME* za nastavak igranja razine jednako se postavlja kao i prethodno spomenute tipke *START GAME*, *OPTIONS* i *BACK*. Pritiskom tipke *RESUME* objekt *Level1* se postavlja kao aktivan, a objekt *PauseMenu* kao neaktivan. Ta funkcionalnost vrijedi za sve razine, a razlikuju se samo imena objekta koji predstavlja razinu. Ako igrač pritisne tipku *RESUME* vraća se na razinu čiji je raspored objekata jednak rasporedu u trenutku otvaranja izbornika pauze.

Pritiskom tipke *RESTART* za ponovno pokretanje razine poziva se metoda *LoadLevel* na objektu *Canvas* unutar kojega se nalaze svi ostali objekti vezani za razinu. Pozivom te metode razina se ponovno učitava (Programski kod 3.6.). Ako se igrač pomaknu do određenog položaja on se vraća na početak razine i broj koraka se vraća na početni broj koraka. Za ponovno učitavanje razine korištena je metoda *LoadScene* kojoj je predan indeks trenutno aktivne scene. Za provedbu te metode uključen je imenik *UnityEngine.SceneManagement*.

```
using UnityEngine;
using UnityEngine.SceneManagement;
using Assets.Scripts;

0 references
public class PauseMenu : MonoBehaviour, IExit
{
    public GameObject mPauseMenuUI;
    public GameObject mLevel;

    0 references
    private void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
                mPauseMenuUI.SetActive(true);
                mLevel.SetActive(false);
        }
    }

    0 references
    private void LoadLevel()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
    }

    4 references
    public void ExitGame()
    {
        Debug.Log("QUIT!");
        Application.Quit();
    }
}
```

Programski kod 3.6. Skripta PauseMenu

Osim nastavka igranja i ponovnog pokretanja razine, u izborniku pauze igrač može promijeniti glasnoću pozadinske glazbe. Mijenjanjem vrijednosti klizača glasnoće zvuka poziva se već spomenuta metoda *SetVolume* iz skripte *VolumeSettings* jer je korišten kao *prefab* objekt. Uz promjenu glasnoće pozadinske glazbe ponovno je javlja tipka za napuštanje igre. Pritiskom tipke *EXIT GAME* na objektu *Canvas* poziva se metoda *ExitGame*. U skriptu je uključen imenik *Assets.Scripts* jer klasa *PauseMenu* implementira sučelje *IExit* (Programski kod 3.6.).

Metoda *Update* se stalno pokreće i provjerava pritisak tipke *Escape* na tipkovnici igrača. Kada se dogodi pritisak tipke objekt *PauseMenu* se postavlja kao aktivan, a objekt *Level1* neaktivan. To vrijedi za sve razine čiji objekti razine imaju drugačija imena, dok se pritiskom tipke *Escape* u izbornicima ništa ne događa. Objekti skripte pridruženi su objektima *mPauseMenuUI* i *mLevel* u Unityju.

3.3. Dizajn pozadine razina

Pozadine razina jednake su za sve razine, a razlikuju se po bojama koje se koriste. Sve pozadine se sastoje od pozadinske slike s dva kristala na kojima pišu brojevi razine i brojevi koraka. Te pozadine su korištene i u *prefab* objektu *Story*. Objektu *Canvas* pridružena je skripta *StepCounter* koja se koristi za postavljanje i izmjenu broja koraka (Programski kod 3.7.). Učitavanjem scene s razinom pokreće se metoda *Start* koja poziva metodu *SetSteps* za postavljanje koraka.

```
1   using UnityEngine;
2   using UnityEngine.SceneManagement;
3   using UnityEngine.UI;
4
    1 reference
5   public class StepCounter : MonoBehaviour
6   {
7       public Text mStepsText;
8       private int mStepNumber = 0;
9
        0 references
10      private void Start()
11      {
12          SetSteps();
13      }
```

Programski kod 3.7. Skripta StepCounter metode Start

Za mijenjanje teksta objekata scene uključen je imenik *UnityEngine.UI*, a za dobivanje indeksa scene *UnityEngine.SceneManagement*. U metodi *SetSteps* prvo se pronalazi indeks scene, koji se uspoređuje s brojevima od jedan do devet i vrijednost koraka se postavlja na određenu vrijednost. Razine imaju drugačiji broj koraka zbog čega je potrebno provjeriti indeks razine. Kada se odredi broj koraka taj se broj postavi u objekt koji se prikazuje u sceni (Programski kod 3.8.).

Skripta *StepCounter* također ima metodu za smanjivanje, odnosno promjenu broja koraka (Programski kod 3.9.). Metoda *RemoveStep* se poziva tijekom igre, kada se lovac pomiče po razini. Broj poziva metode RemoveStep ovisi o smjeru kretanja lovca na kristale i njegovoj interakciji s ostalim objektima. Ona smanjuje broj koraka za jedan i tu vrijednost pridružuje objektu koji se prikazuje u sceni. Zbog toga igrač u svakom trenutku zna koliko mu je koraka ostalo i to uzima u obzir pri odlučivanju o daljnjem kretanju kroz razinu.

```
1 reference
private void SetSteps()
{
    int sceneIndex = SceneManager.GetActiveScene().buildIndex;
    if (sceneIndex == 1)
    {
        mStepNumber = 11;
    }
    else if (sceneIndex == 2)
    {
        mStepNumber = 13;
    }
    else if (sceneIndex == 3 || sceneIndex == 4 || sceneIndex == 6)
    {
        mStepNumber = 14;
    }
    else if (sceneIndex == 5)
    {
        mStepNumber = 15;
    }
    else if (sceneIndex == 7)
    {
        mStepNumber = 25;
    }
    else if (sceneIndex == 8)
    {
        mStepNumber = 16;
    }
    else if (sceneIndex == 9)
    {
        mStepNumber = 27;
    }
    mStepsText.text = mStepNumber.ToString();
}
```

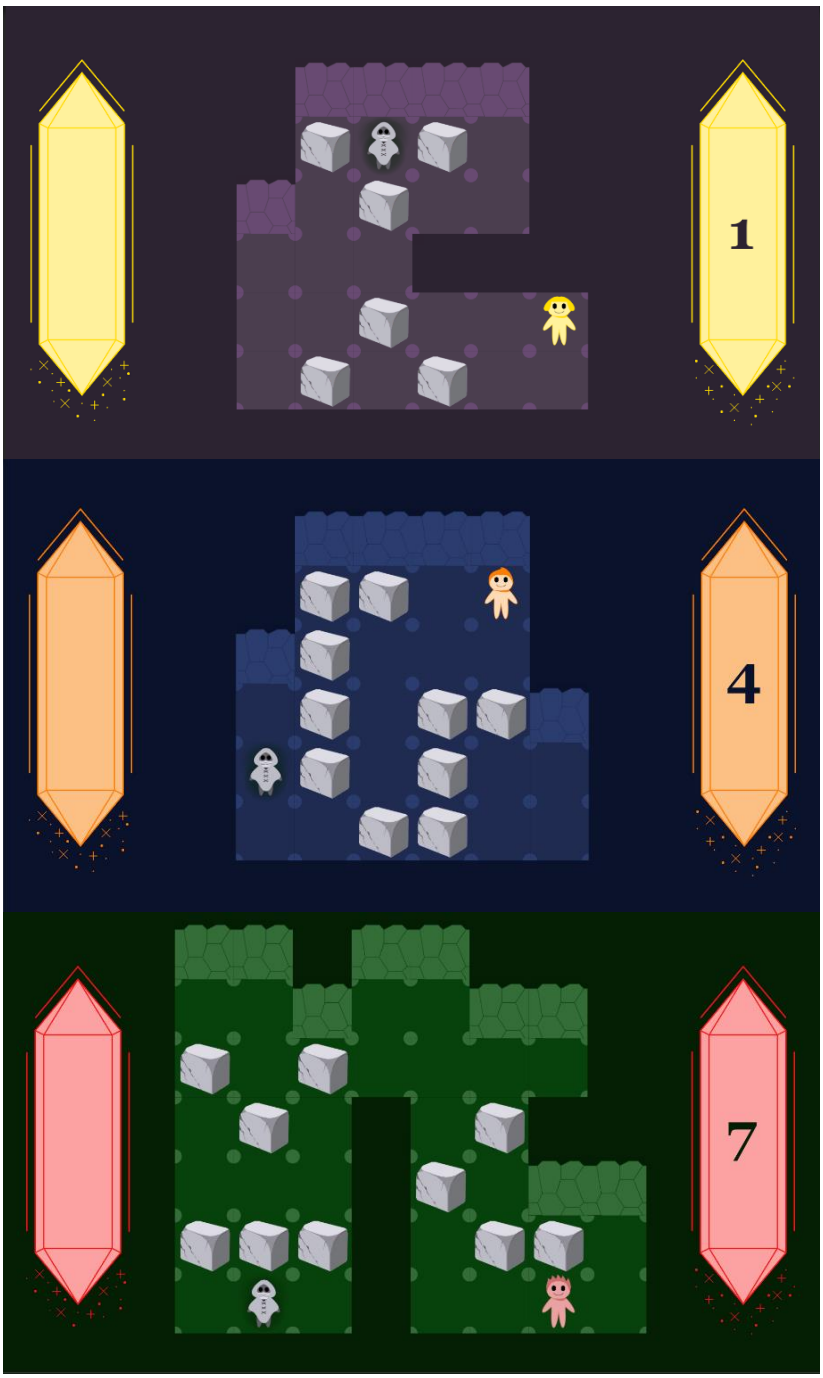
Programski kod 3.8. Skripta StepCounter metoda SetSteps

```
7 references
public void RemoveStep()
{
    mStepNumber = int.Parse(mStepsText.text);
    if (mStepNumber == 0)
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
    }
    else
    {
        mStepNumber--;
        mStepsText.text = mStepNumber.ToString();
    }
}
}
```

Programski kod 3.9. Skripta StepCounter metoda RemoveStep

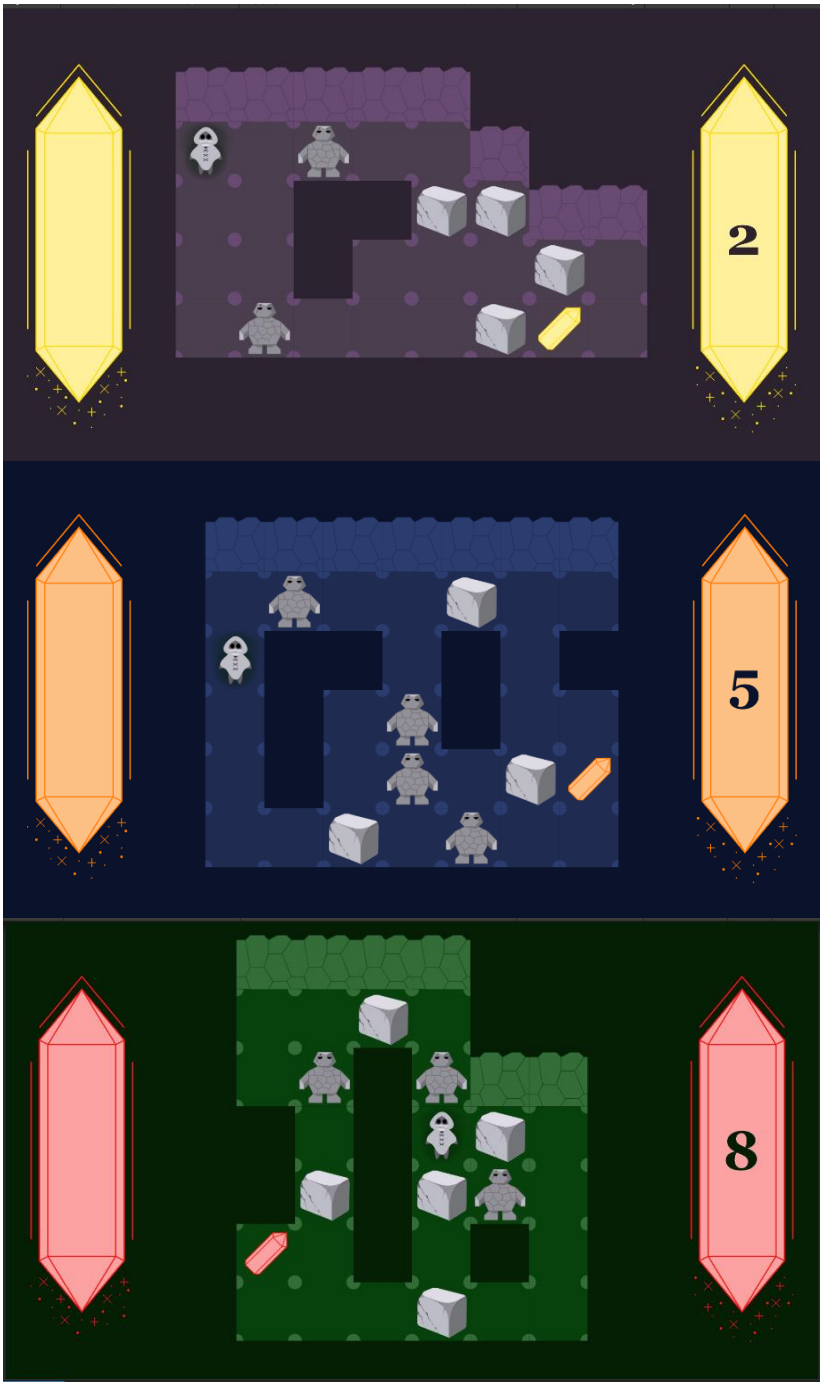
3.4. Dizajn razina i objekata

Razine se međusobno razlikuju po broju objekata i težini rješenja logičkih slagalica. Svi modeli objekata i razina su izrađeni u GIMP-u. Kako je igra podijeljena u tri špilje s tri razine, prve razine u špiljama su najlakše, dok su ostale teže. Objekti koje se nalaze u svim razinama su tlo, zidovi, granice, kameni blokovi i lovac, dok se ostali objekti izmjenjuju ovisno o razini. Prve, druge i treće razine u svim špiljama imaju slične karakteristike. Slika 3.9. prikazuje prvu, četvrtu i sedmu razinu.



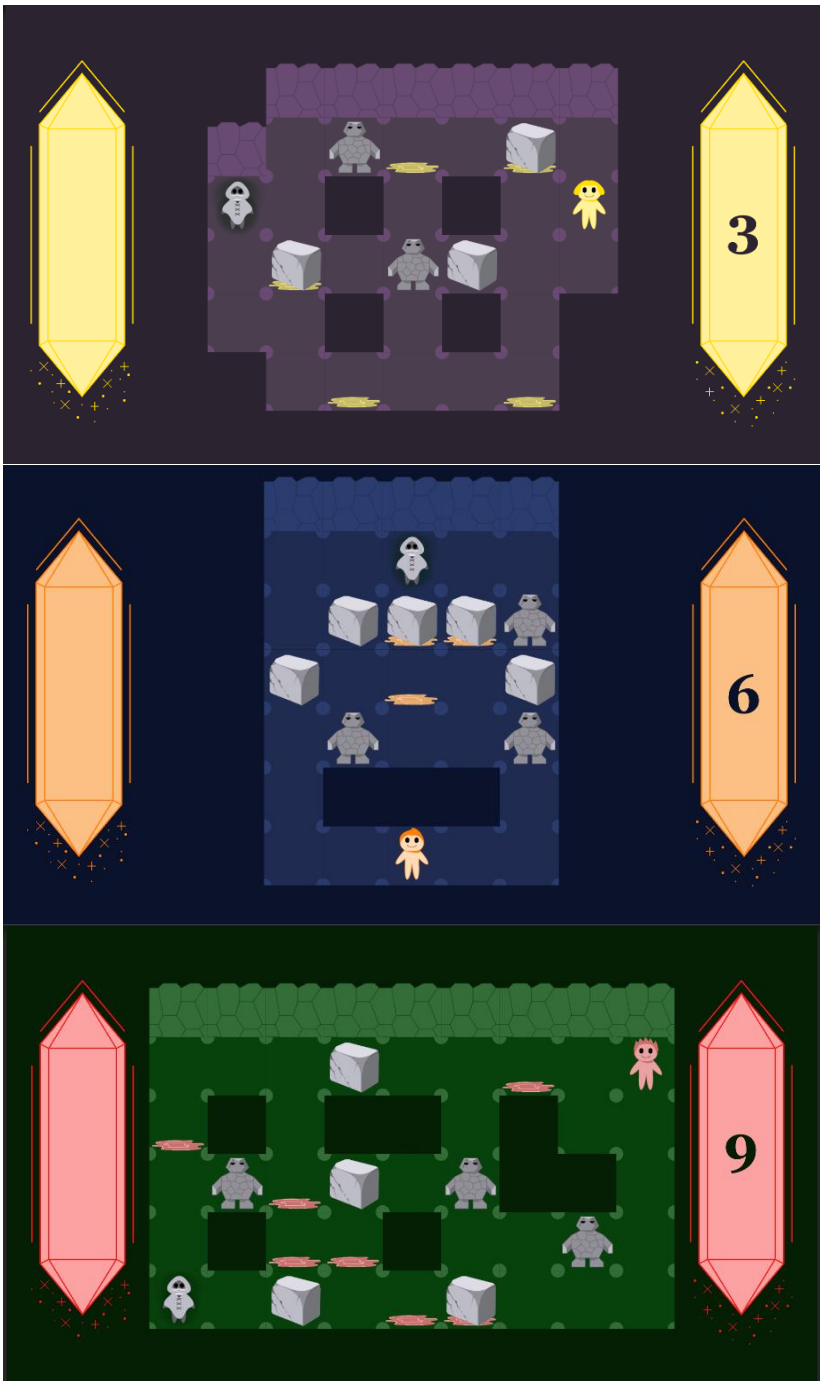
Slika 3.9. Prva, četvrta i sedma razina

U prvim razinama lovac treba pomicanjem kamenih blokova doći do vodiča, koji mu pokazuje put kroz špilju. Zbog veličine kamenih blokova lovac gudi dodatan korak jer troši više energije. Pronalazeći vodiča lovac prelazi u druge razine, odnosno drugu, petu i osmu. U drugim razinama lovac na kristale treba pomicati kamene blokove i čudovišta kako bi sakupio kristale (Slika 3.10.). Treba uzeti u obzir da pomicanjem kamenih blokova i čudovišta lovac gubi dodatan korak jer je za pomicanje tih objekata potrebno uložiti dodatnu energiju.



Slika 3.10. Druga, peta i osma razina

Cilj drugih razina je doći do kristala i sakupiti ih. Nakon što lovac sakupi kristal u špilji on pokušava naći izlazak rješavajući treću razinu. Uz ostale objekte, treće razine karakteriziraju lokve, koje otežavaju kretanje lovca (Slika 3.11.). Kako lokve otežavaju kretanje one oduzimaju barem dva dodatna koraka lovcu ako prođe kroz njih. Cilj lovca je doći do vodiča i slušajući njihove savjete pronaći izlazak iz špilja te tako nastaviti potragu za ostalim kristalima.



Slika 3.11. Treća, šesta i deveta razina

3.4.1. Glavni lik u računalnoj igri

Glavni lik računalne igre Crystal Hunter je lovac na kristale koji ih sakuplja prolazeći kroz špilje (Slika 3.12). Izgled glavnog lika se ne mijenja prolaskom kroz razine kako bi ga igrač lakše uočio. Oko glavnog lika se nalazi zatamnjeni prostor kako bi se isticao prelaskom preko ostalih objekata u razini.

Slika 3.12. Model lovca

U prilogu 3.1. se nalazi skripta *GameController* koja sadrži metodu *Update* kojom se provjeravaju ograničenja kretanja glavnog lika. Njegovo kretanje je ograničeno na okomito i vodoravno kretanje u smjerovima u kojima se ne nalaze drugi objekti. U slučaju guranja objekta glavni lik ga može pomicati sve dok se iza njega u smjeru guranja ne nalazi drugi objekt. To se provjerava koristeći slojeve na koje su postavljeni objekti. Provjerava se nalazi li se u željenom smjeru kretanja lovca kameni blok ili kameno čudovište iza kojih se nalazi drugi objekt. Ako je taj uvjet istinit, dalje se provjerava nalazi li se drugi objekt iza kamenog čudovišta. U istinitom slučaju, objekt kamenog čudovišta se uništava, odnosno ono se skameni te se poziva metoda *RemoveStep* za smanjenje koraka. U slučaju kada se iza objekata kamenih blokova ili kamenih čudovišta ne nalazi ništa onda se poziva metoda *RemoveStep* i objekt lovca pomiče drugi objekt koji se nalazi u smjeru kretanja. Ako su sva ograničenja zadovoljena svakim korakom odnosnom pritiskom tipki za kretanje poziva se metoda *Move* (Programski kod 3.10.).

```
89
        2 references
90      private void Move(Vector3 vector3)
91      {
92          Vector3 newPosition = mMovePoint.position + vector3;
93
94          if (!Physics2D.OverlapCircle(newPosition, 0.2f, mBorderColliders))
95          {
96              mStepCounter.RemoveStep();
97              mMovePoint.position = newPosition;
98
99          }
100     }
```

Programski kod 3.10. Skripta GameController metoda Move

Metodi *Move* predaje se objekt *Vector3* koji označava željeni smjer kretanja koji se pridodaje trenutnom položaju. U metodi se provjerava prelazi li novi željeni položaj kretanja granice razine ili praznina i u slučaju kada ne prelazi smanjuje se broj koraka i objekt lovca se pomiče. Programski kod provjere kretanja nalazi se u prilogu 3.1.

3.4.2. Objekti koji reagiraju na sudare ili okidače

Pomičući se kroz razine, lovac dolazi u kontakt s drugim objektima koji reagiraju na sudare ili na okidače. Te karakteristike su ugrađene u komponente Unityja pod nazivima *Box Collider 2D* i *Rigidbody 2D*. Komponentom *Box Collider 2D* definira se reagira li objekt na sudare ili okidače, a komponentom *Rigidbody 2D* definira se tip tijela objekta.

Objekti kamenih blokova reagiraju na sudare, što je postavljeno u komponenti *Box Collider 2D*. Komponentom *Rigidbody 2D* tip tijela objekta postavljen na dinamičan (engl. *dinamic*) što znači da se objekt pomiče zbog sudara (Slika 3.13.). To omogućava lovcu da pomiče kamene blokove po razini.

Slika 3.13. Model kamenog bloka

Objekti kamenih čudovišta su jednako postavljeni kao objekti kamenih komponenti, ali se oni uništavaju kada dođu u sudare sa drugim objektima (Slika 3.14.). To je prikazano u igrici kao skamenjivanje čudovišta na mjestu gdje se nalaze. Zbog toga lovac može preći preko njih. Skamenjivanje kamenih čudovišta ne ovisi o tome jesu li oni bili pomaknuti prije skamenjivanja već nalazi li se neki drugi objekt iza njih.

Slika 3.14. Model kamenog čudovišta

Objekti lokvi reagiraju na okidače, što je postavljeno u komponenti *Box Collider 2D*. U komponenti *Rigidbody 2D* tip tijela objekta postavljen na kinematički (engl. *kinematic*) što znači da se objekt ne pomiče kod sudara (Slika 3.15.). Kada dođe do sudara lovca s lokvom on ju ne pomiče nego prelazi preko nje.

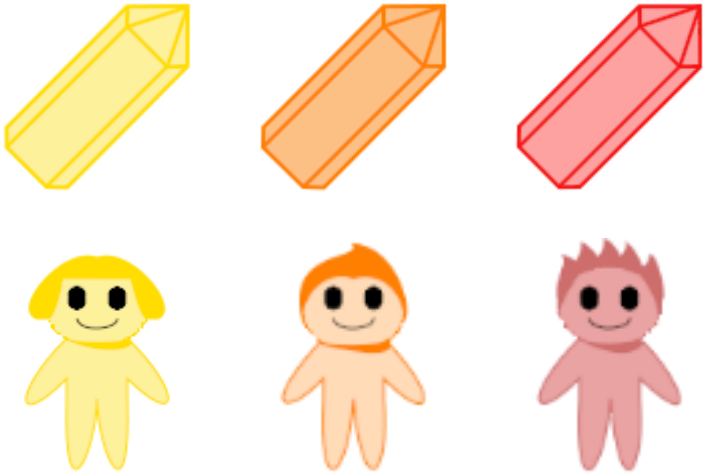
Slika 3.15. Modeli lokvi

U slučaju sudara s lokvama, na okidač se pozivaju dvije metode *OnTriggerEnter2D* i *OnTriggerExit2D* iz skripte *GameController* (Programski kod 3.11.). Te metode imaju istu funkcionalnost, odnosno provjeravaju ako se dogodio sudar s objektom lokva i ulazi li se ili izlazi iz sudara, što izaziva okidač. Na ulazu i izlazu iz sudara, smanjuje se broj koraka. Ako se u slučaju pauziranja igre lovac nalazi u lokvi, obje se metode pokreću što znači da se igrač vraća u igru s dva koraka manje. Zbog toga igrač treba paziti u kojem trenutku otvara izbornik pauze i s kojim ciljem. Ako želi ponovno pokrenuti razinu onda nije bitno gdje ostavi lovca jer će se pritiskom tipke *RESTART* objekti razine i broj koraka vratiti na početno stanje.

```
101
        0 references
102     private void OnTriggerEnter2D(Collider2D collider)
103     {
104         if (collider.gameObject.CompareTag("Puddle"))
105         {
106             mStepCounter.RemoveStep();
107         }
108     }
109
        0 references
110     private void OnTriggerExit2D(Collider2D collider)
111     {
112         if (collider.gameObject.CompareTag("Puddle"))
113         {
114             mStepCounter.RemoveStep();
115         }
116     }
117 }
```

Programski kod 3.11. Skripta GameController metode OnTriggerEnter2D i OnTriggerExit2D

Objekti vodiča i kristali reagiraju na okidače što im je postavljeno u komponenti *Box Collider 2D*. Komponentom *Rigidbody 2D* tip tijela im je postavljen na dinamičan (engl. *dinamic*), što znači da se objekti pomiču zbog sudara (Slika 3.16.). Iako se ovi objekti pomiču zbog sudara to se ne uočava tijekom igre jer sudar izaziva završetak razine.



Slika 3.16. Modeli kristala i vodiča

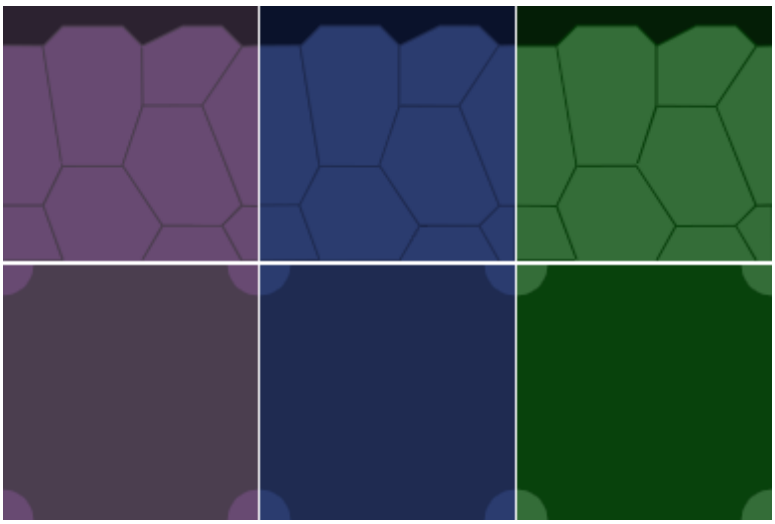
Skripta *LevelController* pridružena je objektima vodiča i kristala (Programski kod 3.12.). Skripta sadrži metodu *OnTriggerEnter2D* koja se poziva u slučaju sudara s objektima koji reagiraju na okidač. U metodi se provjerava sudar objekta s objektom lovca koji ima oznaku *Player*. Provjerava se objekt s kojim se lovac sudario te ako je taj objekt kristal i ima oznaku *Crystal* onda se objekt kristala prestaje prikazivati. To je u igri predstavljeno kao sakupljanje kristala. Objekt razine postavlja se kao neaktivan, nevidljiv, a objekt priče kao aktivan, vidljiv.

```
1   using UnityEngine;
2
    0 references
3   public class LevelController : MonoBehaviour
4   {
5       public GameObject mStory;
6       public GameObject mLevel;
7
        0 references
8       private void OnTriggerEnter2D(Collider2D collider)
9       {
10          if (collider.gameObject.CompareTag("Player"))
11          {
12              if (gameObject.CompareTag("Crystal"))
13              {
14                  gameObject.GetComponent<Renderer>().enabled = false;
15              }
16              mLevel.SetActive(false);
17              mStory.SetActive(true);
18          }
19      }
20  }
```

Programski kod 3.12. Skripta LevelController

3.4.3. Objekti tla i zidova

Objekti tla i zidova su napravljeni u GIMP-u i u scene su složeni prema dizajnu logičkih slagalica (Slika 3.17.). Zidovi i praznine u tlu služe kao granice kretanja lovca. Lovac se ne može kretati izvan granica razine, odnosno špilje. Pomicanjem kamenih čudovišta u zidove, praznine i nevidljive granice razina izaziva njihovo skamenjivanje.



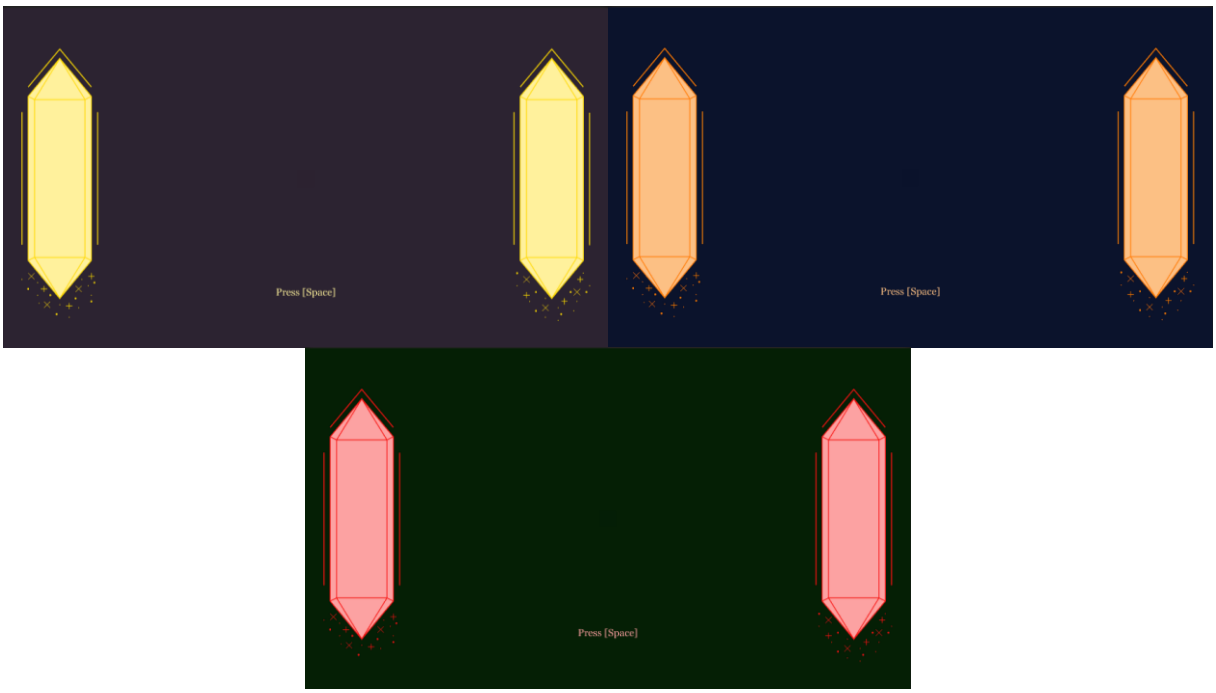
Slika 3.17. Modeli zidova i tla

4. OPIS RAČUNALNE IGRE

4.1. Upute vodiča avanturističko-logičke računalne igre

Upute vodiča predstavljaju priču igre, odnosno monolog koji pomaže igraču u rješavanju logičkih slagalica. Vodiči daju upute lovcu na kristale, ali te upute zapravo predstavljaju savjete koji su upućeni igraču i objašnjavaju pojedinu razinu. Monolog vodiča je napisan engleskim jezikom kako bi igra bila razumljiva većem broju igrača.

Objekti priče se sastoje od pozadine, koja se izmjenjuje prema špilji u kojoj se lovac nalazi, te objekta *StoryText* u kojeg se upisuje tekst iz tekstualne datoteke. U prilogu 4.1. se nalazi cjelokupni tekst iz tekstualne datoteke koji se koristi kao priča igre. Na slici 4.1. prikazane su pozadine objekata priče za različite razine.



Slika 4.1. Pozadine objekata priče

Objekti priče pojavljuju se na kraju scene, odnosno u sceni razine se pojavljuje priča za sljedeću razinu. Nakon što igrač pritisne tipku *START GAME* u glavnom izborniku objekt priče postaje vidljiv. To se također dogodi kada lovac kojim igrač upravlja dođe u kontakt s kristalom ili vodičem. Ukoliko se pristupa sceni iz izbornika opcija priča za tu razinu se ne prikazuje već igrač vidi razinu, a to se događa i ako se iz izbornika pauze igrač vrati na aktivnu scenu. Postavljanjem objekta priče na početak scene on bi se prikazivao svakim novim pokretanjem scene što bi moglo biti frustrirajuće u slučaju višestrukog ponavljanja iste razine. Svaki put kada bi igrač

neuspješno riješio razinu vratio bi se na objekt priče umjesto na početak razine. Objektu priče pridružena je skripta *StroyController* koja definira postupak postavljanja priče i upravlja scenama (Programski kod 4.1.).

```
using System.IO;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using Assets.Scripts;

0 references
public class StoryController : MonoBehaviour, IExit
{
    public Text mStoryText;

    1 reference
    private void SetStory()
    {
        string filePath = Application.streamingAssetsPath +
            "/Recall_Chat/" + "GameStory" + ".txt";
        int sceneIndex = SceneManager.GetActiveScene().buildIndex;

        try
        {
            using (StreamReader inputFile = new StreamReader(filePath))
            {
                for(int i = 1; i < sceneIndex + 1; i++)
                {
                    inputFile.ReadLine();
                }
                mStoryText.text = inputFile.ReadLine();
            }
        }
        catch (IOException e)
        {
            Debug.LogError("Error reading from file.");
            Debug.LogError(e.Message);
        }
    }
```

Programski kod 4.1. Skripta StoryController metoda SetStory

Metoda *SetStory* postavlja tekst u objekt *StoryText*. Za rad s datotekama potrebno je bilo uključiti imenik *System.IO*, a za pronalazak indeksa scene *UnityEngine.SceneManagement*. Kod za čitanje iz datoteke stavljen je u *try-catch* blok za hvatanje iznimki kako bi se upravljalo greškama prilikom čitanja iz datoteke. Korištena je *for* petlja za prolazak kroz redove teksta dok se ne dođe do traženog i taj se tekst postavi u objekt *StoryText*. Kako bi se tekst mogao postaviti u objekt uključen je imenik *UnityEngine.UI*. Metoda *SetStory* poziva se u metodi *Update* koja se stalno pokreće (Programski kod 4.2.). Metodom *Update* provjerava se vidljivost objekta priče i indeks scene.

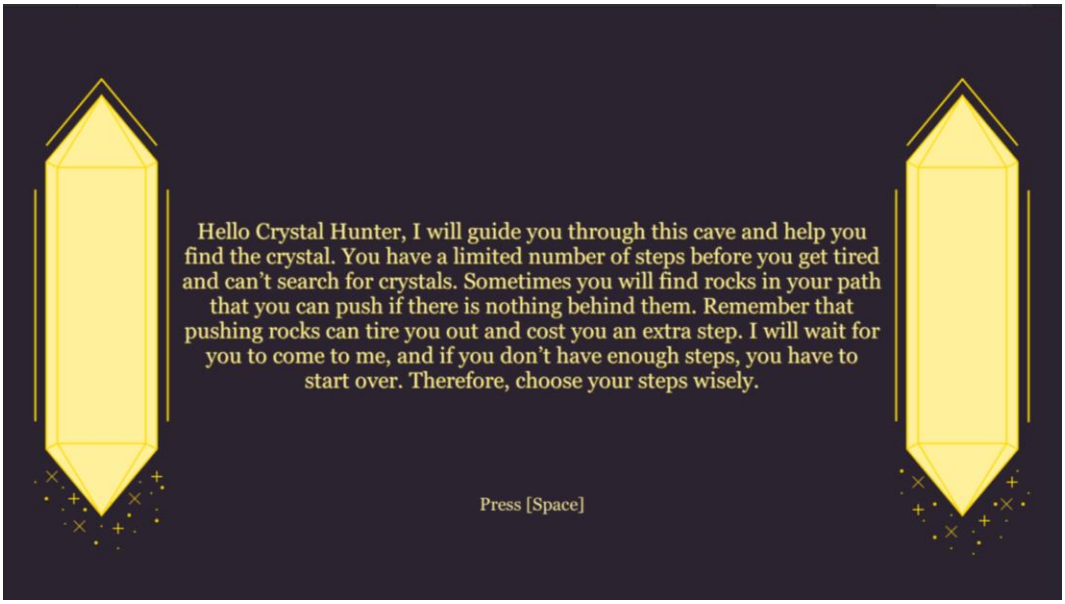
```
34
        0 references
35      private void Update()
36      {
37          int sceneIndex = SceneManager.GetActiveScene().buildIndex;
38          if (gameObject.activeInHierarchy && sceneIndex != 9)
39          {
40              SetStory();
41          }
42          if (Input.GetKeyDown(KeyCode.Space))
43          {
44              if(sceneIndex != 9)
45              {
46                  SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex
47                      + 1);
48              }
49              else
50              {
51                  ExitGame();
52              }
53          }
54      }
55
        4 references
56      public void ExitGame()
57      {
58          Debug.Log("QUIT!");
59          Application.Quit();
60      }
61  }
```

Programski kod 4.1. Skripta StoryController metode Update i ExitGame

Ako je indeks scene različit od devet onda se poziva metoda *SetStory*. Stalno se provjerava pritisak tipke *Space* te ukoliko je tipka pritisnuta i indeks scene je različit od devet poziva se metoda *LoadScene* pomoću koje se učitava sljedeća razina. U suprotno slučaju, kada je indeks jednak devet poziva se metoda *ExitGame* koja služi za napuštanje igre. Klasa *StoryController* implementira sučelje *IExit* za što je potrebno uključiti imenik *Assets.Scripts*.

4.2. Prikaz priče računalne igre

Objekt priče nije vidljiv u Unityju sve dok se ne pokrene igra te se postavi tekst objekta. Pritiskom tipke *START GAME* prikazuje se prvi savjet vodiča, a ostali se savjeti prikazuju točnim rješavanjem razina. Na slici 4.2. prikazan je izgled savjeta vodiča za prvu razinu. Savjeti za ostale razine su slični, odnosno razlika je u paleti boja koja je korištena kako bi odgovarala pojedinoj razini. Žuti vodič savjetuje lovca na kristale kako bi on uspješno pronašao kristale. Igrač dobiva objašnjenje da ima ograničen broj koraka te da lovac može pomicati kamene blokove, ali to dodatno smanjuje broj koraka koje može napraviti. Zbog ograničenog broja koraka igrač mora pažljivo pomicati lovca kako bi riješio logičku slagalicu.



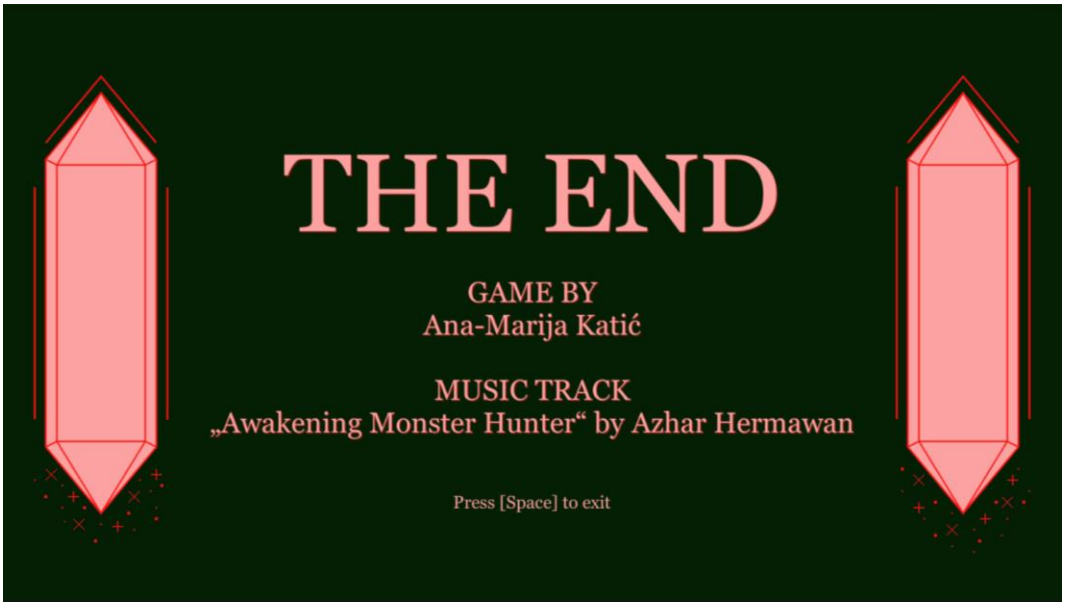
Slika 4.2. Priča žutog vodiča za prvu razinu

Nakon rješavanja logičke slagalice prve razine žuti vodič objašnjava postojanje novih objekata, kamenih čudovišta. Ako se lovac susretne s kamenim čudovištima, treba ih odgurnuti u druge objekte kako bi se oni skamenili i kako bi mogao proći pored njih i pronaći kristal. Igrač doznaje da pomicanje kamenih čudovišta dodatno smanjuje broj koraka.

Nakon uspješnog sakupljanja kristala žuti vodič upozorava na lokve koje otežavaju kretanje lovca kroz razinu i smanjuju njegov broj koraka. Igrač doznaje da ne bi trebao ostavljati lovca u lokvama kako ne bi izgubio dodatne korake jer gubi dodatan korak ulaskom i izlaskom iz lokve. Zbog toga igrač u trećoj razini mora paziti u kojem se smjeru kreće kako bi uspješno riješio logičku slagalicu.

Ulaskom u drugu špilju narančasti vodič savjetuje lovca na sličan način kao što je to učinio žuti vodič. Ne postoje velike razlike između prve tri i druge tri razine, jedina razlika je u tome što zahtjevnost razina raste. To ne bi trebalo predstavljati veliki problem igraču tijekom igranja jer je prelaskom prve tri razine naučio način rješavanja logičkih slagalica s kojim će se susresti u sljedećim razinama.

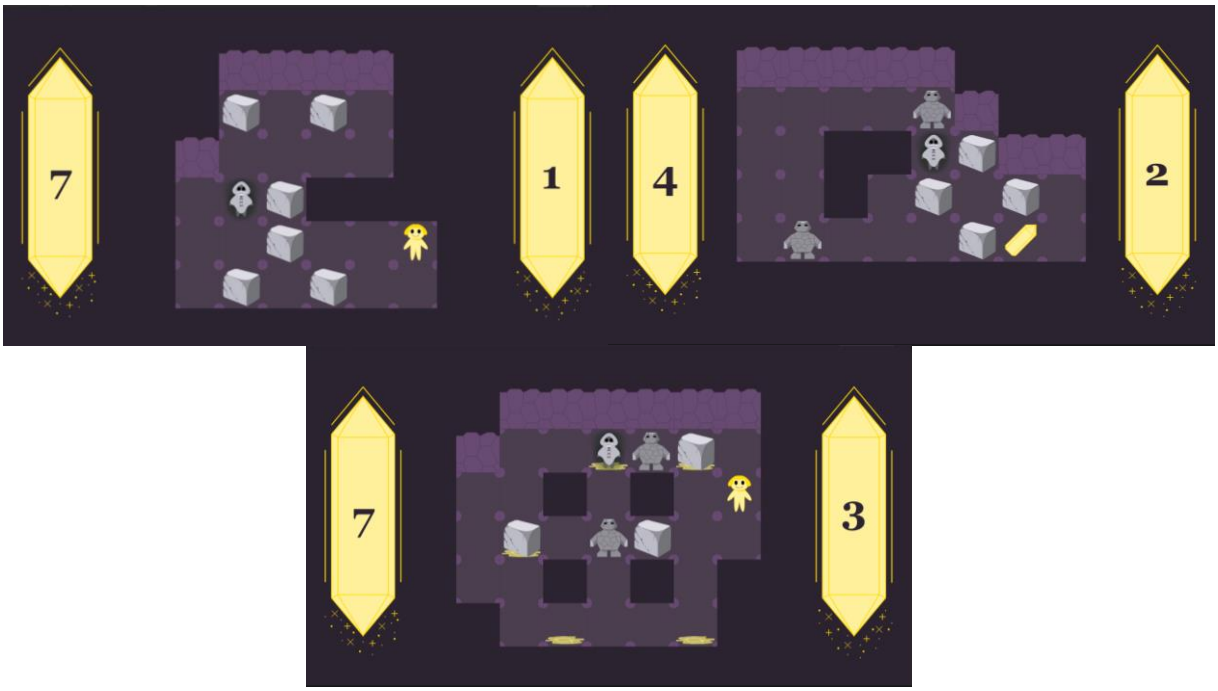
Crveni vodič daje slične savjete kao dva prethodna vodiča. Time podsjeća igrača na što treba paziti rješavajući logičke slagalice. Razine treće špilje predstavljaju najsloženije razine ove avanturističko-logičke računalne igre. Nakon uspješnog rješavanja devete razine umjesto savjeta vodiča prikazuje se objekt kraja računalne igre (Slika 4.3.). Njime je prikazano ime i prezime programera igre i skladatelja pjesme korištene u računalnoj igri. Pritiskom tipke *Space* napušta se igra.



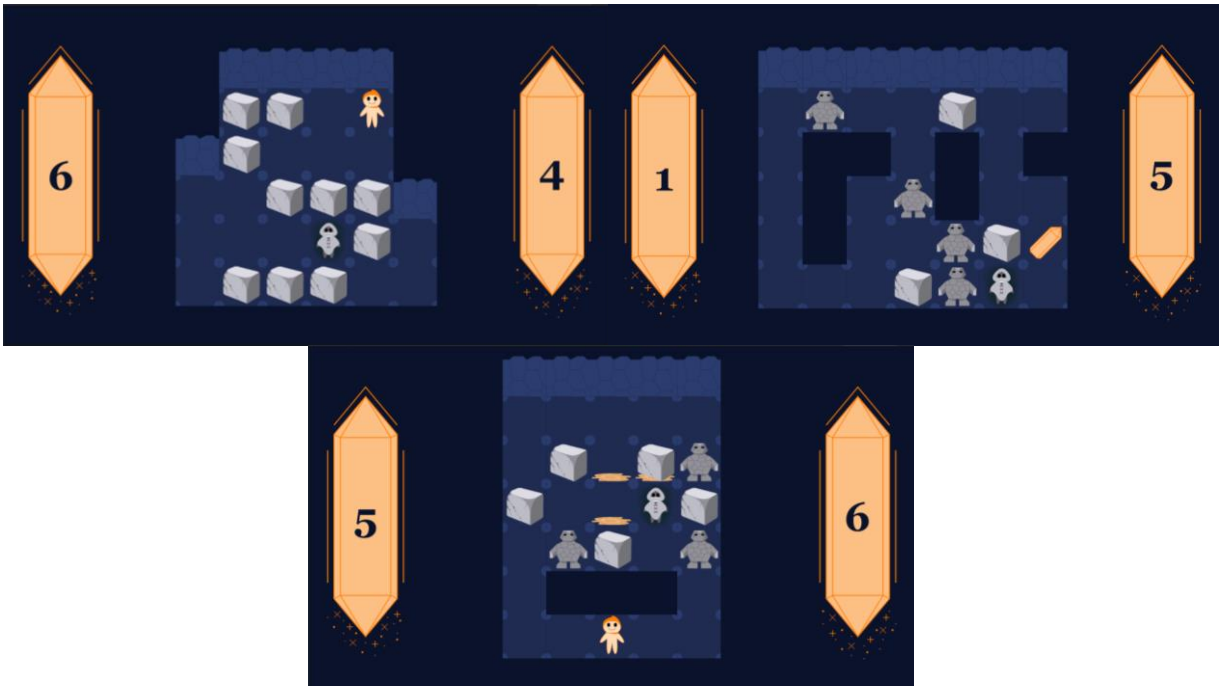
Slika 4.3. Snimka zaslona kraja igre

4.3. Opis tijeka računalne igre

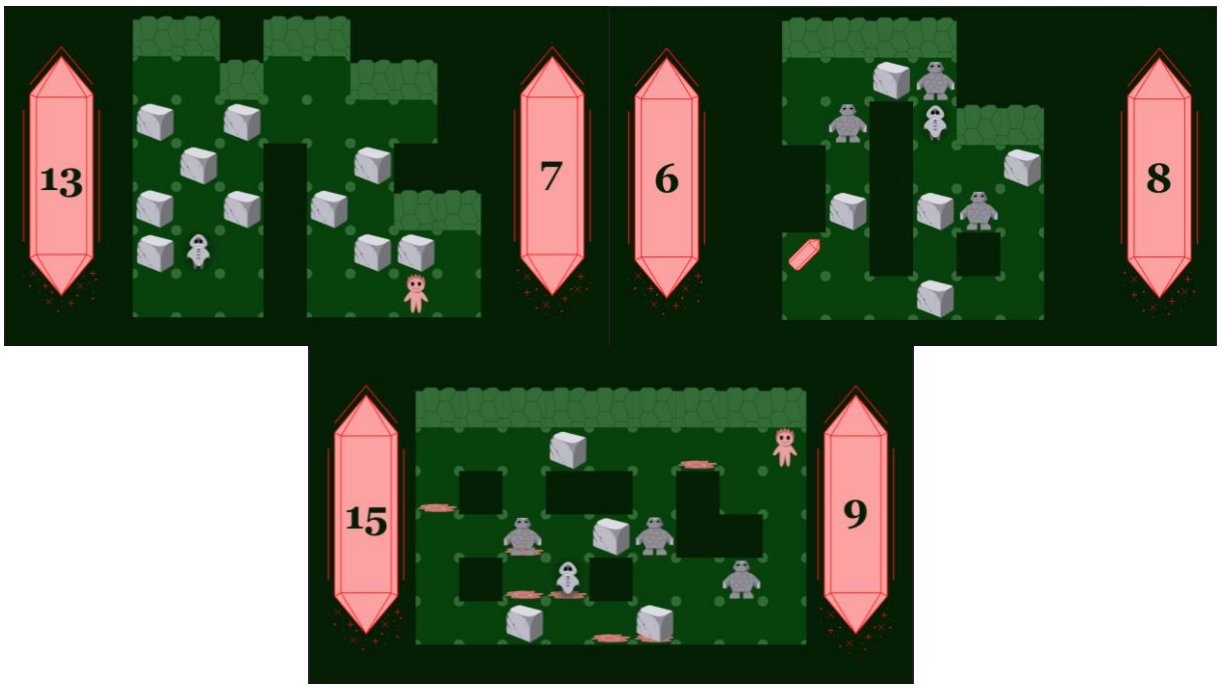
U nastavku su prikazane snimke zaslona tijekom igranja igre (Slike 4.4., 4.5. i 4.6.). Zaslon je sniman u nasumičnim trenutcima tijekom rješavanja logičkih slagalica. Na taj način je omogućen prikaz igranja računalne igre bez otkrivanja rješenja logičkih slagalica.



Slika 4.4. Snimke zaslona tijekom igranja prve tri razine



Slika 4.5. Snimke zaslona tijekom igranja razina u drugoj špilji



Slika 4.6. Snimka zaslona tijekom igranja zadnje tri razine

5. ZAKLJUČAK

Završnim radom prikazan je postupak razvoja avanturističko-logičke računalne igre Crystal Hunter. Opisane su korištene tehnologije za razvoj igre i dan je razlog njihova korištenja. Objašnjena je temeljna ideja igre i svrha njezine izrade. Težnja igre je omogućiti igraču zabavu tijekom rješavanja logičkih slagalica, koje potiču razvoj sposobnosti logičkog zaključivanja i rješavanja logičkih problema. GIMP program je korišten za dizajn svih modela korištenih u stvaranju igre. Izrađena je paleta boja kako bi modeli objekata i pozadina bili komplementarni. Tako je izgled igre objedinjen u skladnu cjelinu. Ti modeli su korišteni u Unityju, gdje su djelomično definirana njihova ponašanja, a ostatak njihove funkcionalnosti određen je C# skriptama u Visual Studiju. Razvojem korisničkog sučelja koje je lako razumljivo igračima poboljšano je iskustvo igre. Upute igre su elegantno ukomponirane u priču igre, odnosno savjete vodiča. Savjeti služe kao pomoć u rješavanju razina, koje su postupno sve zahtjevnije. Rezultati razvoja avanturističko-logičke računalne igre Crystal Hunter prikazanu su u obliku snimki zaslona tijekom njezina igranja.

Jedna od mogućnosti nadogradnje završnog rada je primjena animacija na objektima korištenim u igri. To bi moglo poboljšati iskustvo korisnika, ali je u radu izbjegnuto kao stilska odluka. Animacije bi se mogle izraditi u programima za obradu slika ili programima za animaciju i jednostavno implementirati u Unity pomoću ugrađenih značajki za upravljanje animacijama. Uz dodavanje animacija, daljnjim razvojem igre mogao bi se povećati broj razina i njihova složenost. Uključivanjem novih objekata s drugačijim funkcionalnostima promijenila bi se dinamika razina i povećala njihova kompleksnost.

LITERATURA

[1] Leksikografski zavod Miroslav Krleža, "Hrvatska enciklopedija, mrežno izdanje," 2021. [Online]. Dostupno na: https://www.enciklopedija.hr/natuknica.aspx?ID=68642. [10. srpnja 2021.].

[2] Wikipedia, "Video game," 2021. [Online]. Dostupno na: https://en.wikipedia.org/wiki/Video_game. [10. srpnja 2021.].

[3] Društvo za komunikacijsku i medijsku kulturu, brošura Svijet videoigara, "7 prednosti igranja videoigara i 4 nedostatka," Medijska pismenost - abeceda za 21. stoljeće, 2018. [Online]. Dostupno na: https://www.medijskapismenost.hr/7-prednosti-igranja-videoigara-i-4-nedostatka/. [10. srpnja 2021.].

[4] Wikipedia, "Unity (game engine)," 2021. [Online]. Dostupno na: https://en.wikipedia.org/wiki/Unity_(game_engine). [11 srpnja 2021.].

[5] M. Dealessandri, "What is the best game engine: is Unity right for you?," 2020. [Online]. Dostupno na: https://www.gamesindustry.biz/articles/2020-01-16-what-is-the-best-game-engine-is-unity-the-right-game-engine-for-you. [11. srpnja 2021.].

[6] E. Peckham, "How Unity built the world's most popular game engine," 2019. [Online]. Dostupno na: https://techcrunch.com/2019/10/17/how-unity-built-the-worlds-most-popular-game-engine/. [11. srpnja 2021.].

[7] Unity, "Unity Manual," 2020. [Online]. Dostupno na: https://docs.unity3d.com/Manual/UnityasaLibrary.html. [10. srpnja 2021.].

[8] Wikipedia, "Microsoft Visual Studio," 2021. [Online]. Dostupno na: https://en.wikipedia.org/wiki/Microsoft_Visual_Studio#History. [11. sprnja 2021.].

[9] "GIMP," [Online]. Dostupno na: https://www.gimp.org/. [11. srpnja 2021.].

[10] Wikipedia, "GIMP," 2021. [Online]. Dostupno na: https://en.wikipedia.org/wiki/GIMP. [11. srpnja 2021.].

[11] A. Hermawan, "Awakening Monster Hunter," [Online]. Dostupno na: https://pixabay.com/music/search/hunter/. [10. srpnja 2021.].

[12] Skupina autora, "Fundamentals of Computer Programming with C#," Vol. 16, pp. 643, Sofia, 2013.

SAŽETAK

Završnim radom opisan je postupak razvoja avanturističko-logičke računalne igre Crystal Hunter. U radu su objašnjene korištene tehnologije te proces od stvaranja temeljne ideje igre do njezinog igranja. Svi objekti i pozadine su osmišljeni i dizajnirani u GIMP-u. Većina funkcionalnosti objekata je definirana pomoću C# skripti u Visual Studiju, dok je dio ponašanja objekata postavljan pomoću ugrađenih značajki Unityja. Objekti razvijeni pomoću GIMP-a korišteni su za razvoj korisničkog sučelja i razina, koje sadrže logičke slagalice. Igra se sastoji od devet razina, a svaka razina sadrži jednu logičku slagalicu. Složenost logički slagalica varira ovisno o razini, gdje se najjednostavnije nalaze na početku špilja. Svaka špilja sadrži tri razine. Kompleksnost logičkih slagalica raste prolaskom kroz špilju, ali ovisi i o špilji u kojoj se lovac na kristale nalazi. Najzahtjevnije slagalice se nalaze u zadnjoj špilje sa sedmom, osmom i devetom razinom. Jednostavnost korisničkog sučelja i upute uključene u savjete vodiča poboljšavaju iskustvo igranja igre. Rezultat ovog završnog rada je avanturističko-logička računalna igra, koja uz zabavu, potiče razvoj sposobnosti rješavanja problema i logičkog zaključivanja.

Ključne riječi: Avanturističko-logička računalna igra, GIMP, Razvoj igara, Unity

ABSTRACT

Development of an adventure-logic computer game

The final paper describes the process of developing the adventure-logic computer game Crystal Hunter. The paper explains the used technologies and the process from creating the basic idea of the game to playing it. All objects and backgrounds are conceived and designed in GIMP. Most object functionality is defined using C # scripts in Visual Studio, while part of the object behaviour is set using built-in Unity features. Objects developed using GIMP were used to develop the user interface and levels, which contain logic puzzles. The game consists of nine levels, and each level contains one logic puzzle. The complexity of logical puzzles varies depending on the level, where the simplest are located at the beginning of the caves. Each cave contains three levels. The complexity of logic puzzles grows as you go through the cave, but it also depends on the cave in which the crystal hunter is located. The most challenging puzzles are located in the last cave with the seventh, eighth, and ninth levels. The simplicity of the user interface and the instructions included in the guide tips enhance the gaming experience. The result of this final paper is an adventure-logic computer game, which, in addition to fun, encourages the development of problem-solving skills and logical reasoning.

Keywords: Game development, GIMP, Logic adventure computer game, Unity

ŽIVOTOPIS

Ana-Marija Katić rođena je 18.1.2000. godine u Osijeku. Od 2014. do 2018. godine pohađa Prirodoslovno-matematičku gimnaziju u Osijeku gdje se susreće s programiranjem i stječe znanja o programiranju s programskim jezicima C i Python. Koristeći stečena znanja izrađuje jednostavnu računalnu igru koristeći Python što potiče njeno daljnje učenje programiranja. Godine 2018. upisuje preddiplomski sveučilišni studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek gdje proširuje svoje znanje o programiranju učeći druge programske jezike među kojima su C++, C# i Java.

Ana-Marija Katić

PRILOZI

P.3.0. Projekt računalne igre Crystal Hunter

Dostupan na: https://gitlab.com/Ana-Marija1/crystalhunter

P.3.1. Skripta *GameController* metoda *Update*

Linija ***Kod***

```
1:      private void Update()
2:      {
3:        transform.position = Vector3.MoveTowards(transform.position,
4:        mMovePoint.position, mMoveSpeed * Time.deltaTime);
5:        if (Vector3.Distance(transform.position, mMovePoint.position <=
6:        0.05f)
7:        {
8:          if (Mathf.Abs(Input.GetAxisRaw("Horizontal")) == 1f)
9:          {
10:           if ((Physics2D.OverlapCircle(mMovePoint.position + new
11:           Vector3(Input.GetAxisRaw("Horizontal") * mPixelPerUnit, 0, 0),
12:           0.2f, mRocks) || Physics2D.OverlapCircle(mMovePoint.position
13:           + new Vector3(Input.GetAxisRaw("Horizontal") * mPixelPerUnit,
14:           0, 0), 0.2f, mMonsters)) &&
15:           Physics2D.OverlapCircle(mMovePoint.position(+ new
16:           Vector3(Input.GetAxisRaw("Horizontal") * mPixelPerUnit *
17:           2, 0, 0), 0.2f, mBorderColliders) ||
18:           Physics2D.OverlapCircle(mMovePoint.position + new
19:           Vector3(Input.GetAxisRaw("Horizontal") * mPixelPerUnit *
20:           2, 0, 0), 0.2f, mRocks) ||
21:           Physics2D.OverlapCircle(mMovePoint.position + new
22:           Vector3(Input.GetAxisRaw("Horizontal") * mPixelPerUnit *
23:           2, 0, 0), 0.2f, mGuides) ||
24:           Physics2D.OverlapCircle(mMovePoint.position + new
25:           Vector3(Input.GetAxisRaw("Horizontal") * mPixelPerUnit *
26:           2, 0, 0), 0.2f, mMonsters) ||
27:           Physics2D.OverlapCircle(mMovePoint.position + new
28:           Vector3(Input.GetAxisRaw("Horizontal") * mPixelPerUnit *
29:           2, 0, 0), 0.2f, mCrystals)))
30:           {
31:             if (Physics2D.OverlapCircle(mMovePoint.position +
32:            new Vector3(Input.GetAxisRaw("Horizontal") * mPixelPerUnit,
33:            0, 0), 0.2f, mMonsters) &&
34:            (Physics2D.OverlapCircle(mMovePoint.position + new
35:            Vector3(Input.GetAxisRaw("Horizontal") * mPixelPerUnit * 2,
36:            0, 0), 0.2f, mBorderColliders) ||
```

Linija Kod

```
37:             Physics2D.OverlapCircle(mMovePoint.position + new
38:             Vector3(Input.GetAxisRaw("Horizontal") * mPixelPerUnit * 2,
39:             0, 0), 0.2f, mRocks) ||
40:             Physics2D.OverlapCircle(mMovePoint.position + new
41:             Vector3(Input.GetAxisRaw("Horizontal") * mPixelPerUnit * 2,
42:             0, 0), 0.2f, mMonsters)))
43:             {
44:               Destroy(Physics2D.OverlapCircle(mMovePoint.position + new
45:               Vector3(Input.GetAxisRaw("Horizontal") * mPixelPerUnit,
46:               0, 0), 0.2f, mMonsters));
47:               mStepCounter.RemoveStep();
48:             }
49:           }
50:           else
51:           {
52:             if (Physics2D.OverlapCircle(mMovePoint.position + new
53:             Vector3(Input.GetAxisRaw("Horizontal") * mPixelPerUnit, 0,
54:             0), 0.2f, mRocks) ||
55:             Physics2D.OverlapCircle(mMovePoint.position + new
56:             Vector3(Input.GetAxisRaw("Horizontal") * mPixelPerUnit, 0,
57:             0), 0.2f, mMonsters))
58:               {
59:                 mStepCounter.RemoveStep();
60:               }
61:             Move(new Vector3(Input.GetAxisRaw("Horizontal") *
62:             mPixelPerUnit, 0, 0));
63:           }
64:         }
65:         else if (Mathf.Abs(Input.GetAxisRaw("Vertical")) == 1f)
66:         {
67:           if ((Physics2D.OverlapCircle(mMovePoint.position + new
68:           Vector3(0, Input.GetAxisRaw("Vertical") * mPixelPerUnit, 0),
69:           0.2f, mRocks) || Physics2D.OverlapCircle(mMovePoint.position +
70:           new Vector3(0, Input.GetAxisRaw("Vertical") * mPixelPerUnit,
71:           0), 0.2f, mMonsters)) &&
72:           (Physics2D.OverlapCircle(mMovePoint.position + new
73:           Vector3(0, Input.GetAxisRaw("Vertical") * mPixelPerUnit * 2,
74:           0), 0.2f, mBorderColliders) ||
75:           Physics2D.OverlapCircle(mMovePoint.position + new Vector3(0,
76:           Input.GetAxisRaw("Vertical") * mPixelPerUnit * 2, 0), 0.2f,
77:           mRocks) || Physics2D.OverlapCircle(mMovePoint.position + new
78:           Vector3(0, Input.GetAxisRaw("Vertical") * mPixelPerUnit * 2,
79:           0), 0.2f, mGuides) ||
80:           Physics2D.OverlapCircle(mMovePoint.position + new
```

Linija Kod

```
81:             Vector3(0, Input.GetAxisRaw("Vertical") * mPixelPerUnit * 2,
82:             0), 0.2f, mMonsters) ||
83:             Physics2D.OverlapCircle(mMovePoint.position + new
84:             Vector3(0, Input.GetAxisRaw("Vertical") * mPixelPerUnit * 2,
85:             0), 0.2f, mCrystals)))
86:             {
87:               if (Physics2D.OverlapCircle(mMovePoint.position + new
88:               Vector3(0, Input.GetAxisRaw("Vertical") * mPixelPerUnit, 0),
89:               0.2f, mMonsters) &&
90:               (Physics2D.OverlapCircle(mMovePoint.position + new
91:               Vector3(0, Input.GetAxisRaw("Vertical") * mPixelPerUnit *
92:               2, 0), 0.2f, mBorderColliders) ||
93:               Physics2D.OverlapCircle(mMovePoint.position + new Vector3(0,
94:               Input.GetAxisRaw("Vertical") * mPixelPerUnit * 2, 0), 0.2f,
95:               mRocks) || Physics2D.OverlapCircle(mMovePoint.position + new
96:               Vector3(0, Input.GetAxisRaw("Vertical") * mPixelPerUnit * 2,
97:               0), 0.2f, mMonsters)))
98:               {
99:                 Destroy(Physics2D.OverlapCircle(mMovePoint.position + new
100:                Vector3(0, Input.GetAxisRaw("Vertical") * mPixelPerUnit,
101:                0), 0.2f, mMonsters));
102:                mStepCounter.RemoveStep();
103:              }
104:            }
105:            else
106:            {
107:              if (Physics2D.OverlapCircle(mMovePoint.position + new
108:              Vector3(0, Input.GetAxisRaw("Vertical") * mPixelPerUnit, 0),
109:              0.2f, mRocks) ||
110:              Physics2D.OverlapCircle(mMovePoint.position + new
111:              Vector3(0, Input.GetAxisRaw("Vertical") * mPixelPerUnit, 0),
112:              0.2f, mMonsters))
113:              {
114:                mStepCounter.RemoveStep();
115:              }
116:              Move(new Vector3(0, Input.GetAxisRaw("Vertical") *
117:              mPixelPerUnit, 0));
118:            }
119:          }
120:        }
121:      }
```

P.4.1. Tekst priče igre na engleskom jeziku

„Hello Crystal Hunter, I will guide you through this cave and help you find the crystal. You have a limited number of steps before you get tired and can't search for crystals. Sometimes you will find rocks in your path that you can push if there is nothing behind them. Remember that pushing rocks can tire you out and cost you an extra step. I will wait for you to come to me, and if you don't have enough steps, you have to start over. Therefore, choose your steps wisely.

I see you found your way through the first part of this cave. Look for the crystal in this part of the cave, and once you find it, I will help you get out of the cave. Be careful, and if you find a stone monster, push it to other object to petrify it. You can then walk past it, but keep in mind that pushing requires energy, and you will lose an extra step.

Congratulations, you found the crystal! If you follow me, I will show you the way out of the cave. Watch out for puddles along the way. Moving through them is more exhausting, so you will lose one extra step by stepping in and out of them. Also, I don't recommend standing in puddles, they seem to draw more of your energy. I will wait for you at the exit.

Hi Crystal Hunter, I can see you already have the yellow crystal with you. I can help you find another one, but it will be a little harder. Remember to think before you take a step and don't push rocks if there is something behind them. They will not move. So, don't waste your time and follow me!

You see, it wasn't that much harder. Don't be afraid to push stone monsters because sometimes they stand in the way of finding the crystal. Pushing monsters is exhausting, but they could stand in the quickest way to the crystals. As a reminder, pushing monsters usually creates new paths to take while pushing stones blocks them.

Now that you have found another crystal, I will show you the way out of this cave. Think about everything I have taught you, and watch where you step. Even if walking through puddles is exhausting at times, it is unavoidable.

Hi there! I don't think that you need much help to find the red crystal. I noticed that you already have yellow and orange crystals. You know that moving stones in a certain way creates new paths, and sometimes there is more than one correct path.

You probably already know what to do, so I will just wait for you. But in case you forgot, you have to push stone monsters to other objects to petrify them or push stones out of the way to get to the crystal.

Now that you have collected the last crystal, all you need to do is to find a way out of this cave. On your way, there are stones, puddles, and stone monsters. Choose carefully which ones to move and which ones to avoid. By making the right decisions, you will find the way out.“

P.5.1. Datoteka s modelima dizajniranim u GIMP-u

Dostupna u datoteci GIMP modeli na: https://gitlab.com/Ana-Marija1/crystalhunter