Izrada point-and-click avanture u Unityu

Hlavsa, Nikola

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij računarstva

IZRADA POINT-AND-CLICK AVANTURE U UNITYU

Završni rad

Nikola Hlavsa

Osijek, 2018.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

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

Osijek, 24.09.2018.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada

Ime i prezime studenta:	Nikola Hlavsa
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3638, 27.09.2017.
OIB studenta:	16634214951
Mentor:	Doc.dr.sc. Časlav Livada
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Izrada point-and-click avanture u Unityu
Znanstvena grana rada:	**Obradba informacija (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: 2 bod/boda Razina samostalnosti: 2 razina
Datum prijedloga ocjene mentora:	24.09.2018.
Datum potvrde ocjene Odbora:	26.09.2018.

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, 28.09.2018.

Ime i prezime studenta:	Nikola Hlavsa
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3638, 27.09.2017.
Ephorus podudaranje [%]:	4

Ovom izjavom izjavljujem da je rad pod nazivom**: Izrada point-and-click avanture u Unityu**

izrađen pod vodstvom mentora Doc.dr.sc. Časlav Livada

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

Tema ovog završnog rada je izrada 2D point-and-click avanture. Point-and-click avanture su žanr video igara nastale sredinom 80-ih godina prošlog stoljeća. Temelje se na pripovijedanju interaktivne priče bazirane oko jednog ili više protagonista koje kontrolira igrač, istraživanju svijeta igre te rješavanju raznih zagonetki. Nastale su kao evolucija tekstualnih avantura zamjenom pisanih naredbi s point-and-click mehanikom. Point-and-click je mehanika u igri koja se bazira na pokazivaču. Igrač pokazivačem pomiče lika, skuplja predmete, kombinira ih te rješava razne zagonetke. Neke od poznatijih igara iz zlatnog doba žanra su: Monkey Island serijal, Sam and Max serijal, Day of the Tentacle, Full Throttle itd. Popularizacijom pucačina početkom 21. stoljeća žanr je počeo umirati. Međutim, doživio je preporod 2012. nakon što je tvrtka Telltale Games izdala epizodičnu point-and-click avanturu The Walking Dead.

Naziv igre je Debt Star: Episode I, a izrađena je u Unityu korištenjem C# objektno-orijentiranog programskog jezika. Unity je multi-platformni game engine koji podržava 2D i 3D grafiku, drag-and-drop funkcionalnost i C# skripte. Igra je napravljena s 2D grafikom, te je za izradu 2D spriteova i pozadine korišten Aseprite, program za izradu grafičkih elemenata baziranih na pojedinim pikselima. Zvučni efekti i pozadinska glazba su preuzeti sa stranica s besplatnim sadržajima.

1.1. Zadatak završnog rada

Zadatak završnog rada je izrada 2D point-and-click avanture. Potrebno je ispuniti sve zahtjeve i funkcionalnosti za igru tog žanra. Za izradu je potrebno koristiti Unity i programski jezik C#.

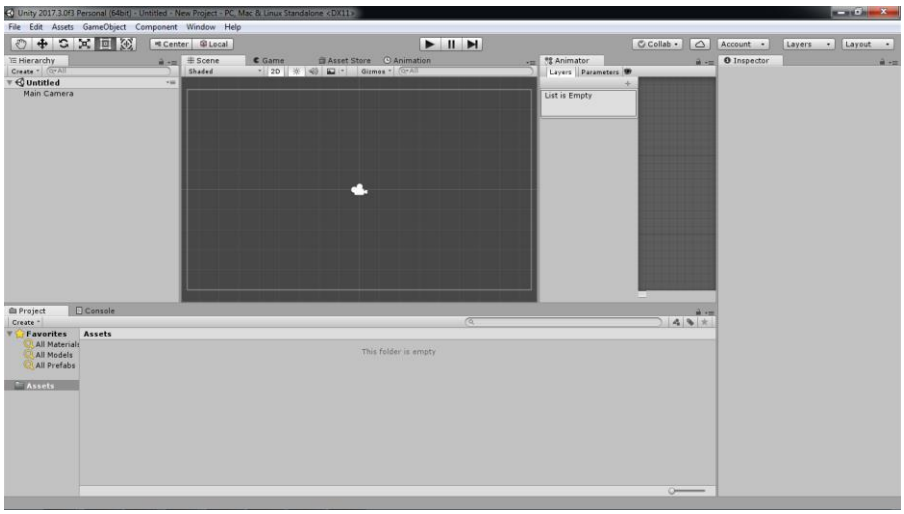
2. KORIŠTENE TEHNOLOGIJE

2.1. UNITY

Unity [1] je multi-platformni game engine koji je razvila tvrtka Unity Technologies. Glavna svrha je razvoj 2D i 3D video igara i simulacija za računala, konzole i mobilne uređaje. Podržava drag-and-drop funkcionalnost i skripte u C# programskom jeziku. Osim C# bili su podržani i još dva programska jezika: Boo, koji je obustavljen izdavanjem Unityja 5 i JavaScript koji je obustavljen nakon izdavanja Unitya 2017.1.

Unity cilja sljedeće grafičke API-je: Direct3D na Windows i Xbox One; OpenGL na Linux, macOS i Windows; OpenGL ES na Androidu i iOS-u; WebGL na webu; i vlasnički API-ji na konzolama za videoigre. Osim toga, Unity podržava nisko-razinske API-je Metal na iOS i Vulkan na Androidu, Linuxu i Windows, kao i Direct3D 12 n Windows i Xbox One.

Unutar 2D igara Unity omogućuje uvoz sprite-ova i napredni render 2D svijeta. Za 3D igre, Unity omogućuje specifikaciju kompresije teksture, mipmaps i postavke razlučivosti za svaku platformu koju podražava engine te podržava bump mapping, reflection mapping, parallax mapping, screen space ambient occlusion(SSAO), dinamične sjene, renderiranje tekstura i efekte postprocesiranja na cijelm zaslonu. Unity također nudi usluge programerima: Unity Ads, Unity Analytics, Unity Certification, Unity Cloud Build, Unity Everyplay, Unity IAP, Unity Multiplayer, Unity Performance Reporting i Unity Collaborate. Unity podržava stvaranje prilagođenih vrhova, fragmenta, piksela, tesselacije, izračunavanje shadera pomoću Cg, izmjenjene verzije Microsoftovog visoko-razinskog jezika za shadere.



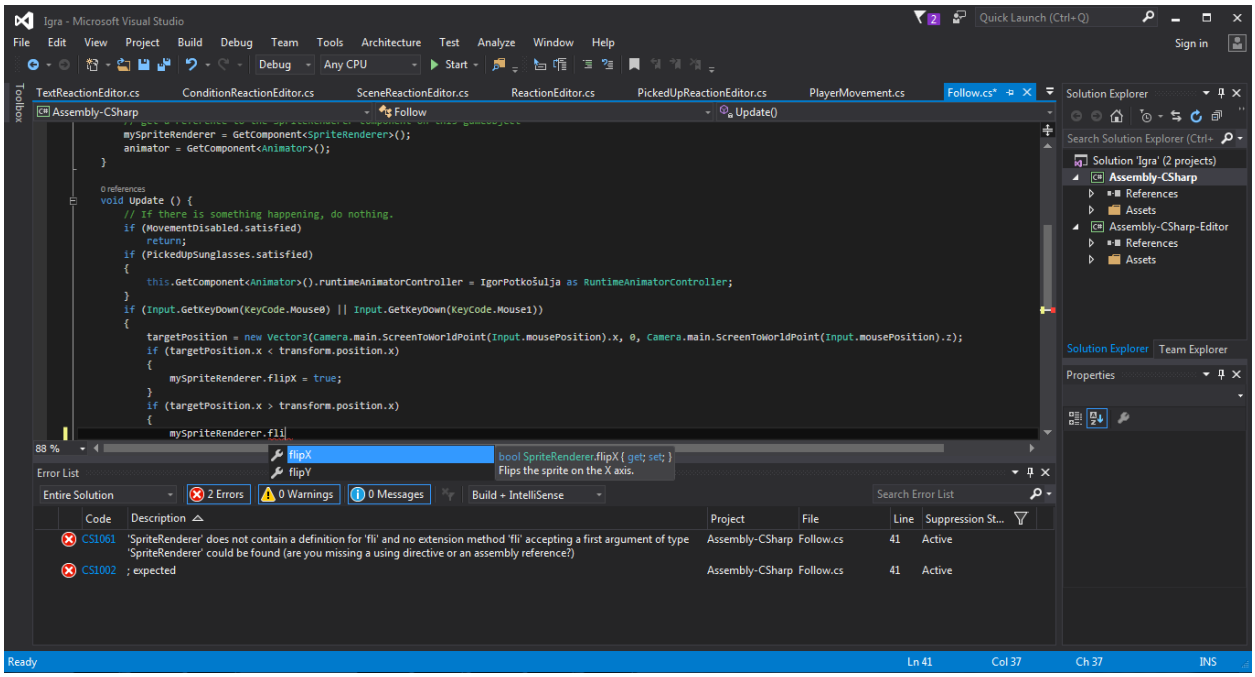
Sl. 2.1. Unity sučelje

Prilikom izrade novog projekta Unity nudi izbor između 2D ili 3D igre, nakon čega se otvara Unity sučelje (sl. 2.1) koje se sastoji od: scene(engl. *Scene*), pregled igre (engl. *Game*), kartica projekta (engl. *Project*), inspektora (engl. *Inspector*), kartica za hijerarhiju objekata (engl. *Hierarchy)* konzole (engl. *Console*), kartica za animacije(engl. *Animation*), animatora(engl. *Animator*). Postoji još raznih kartica sa različitim funkcionalnostima te se sučelje može prilagoditi potrebama korisnika.

2.2. MICROSOFT VISUAL STUDIO

Kako bi se skripte koje se nalaze unutar Unity-a mogle uređivati, korišten je Microsoft Visual Studio [2]. Visual Studio je moguće posebno preuzeti s Microsoft-ove službene stranice, no moguće ga je preuzeti i prilikom preuzimanja Unity-a. Unutar Visual Studia su napravljene sve skripte koje su korištene u završnom radu.

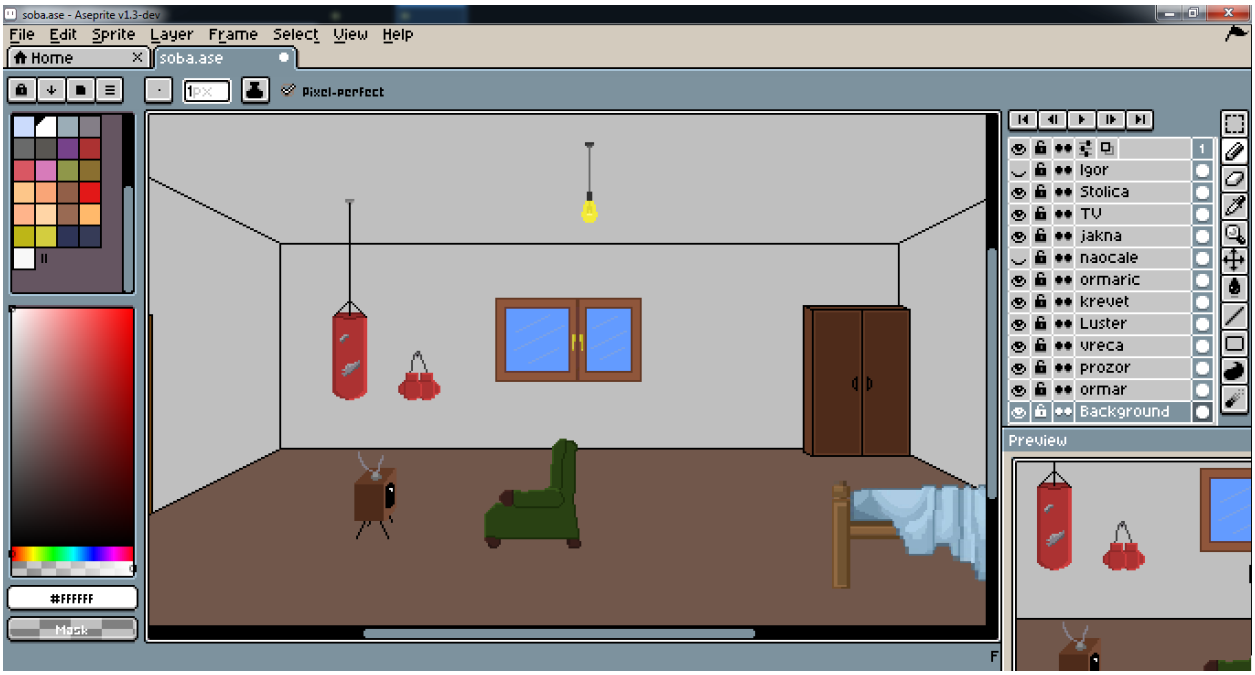
Razlog radi kojeg je izabran Visual Studio kao razvojno okruženje je taj što ima IntelliSence, odnosno mogućnost dovršavanja naredbi. IntelliSence funkcionira tako da korisnik utipka prvih par slova funkcije ili neke varijable koju koristi u svom kodu te mu onda Visual Studio ponudi sve funkcije ili varijable koje se koriste unutar koda koje sadrže tih prvih par slova koji su utipkani.



Sl. 2.2. Korištenje IntelliSense-a u Microsoft Visual Studiu

2.3. ASEPRITE

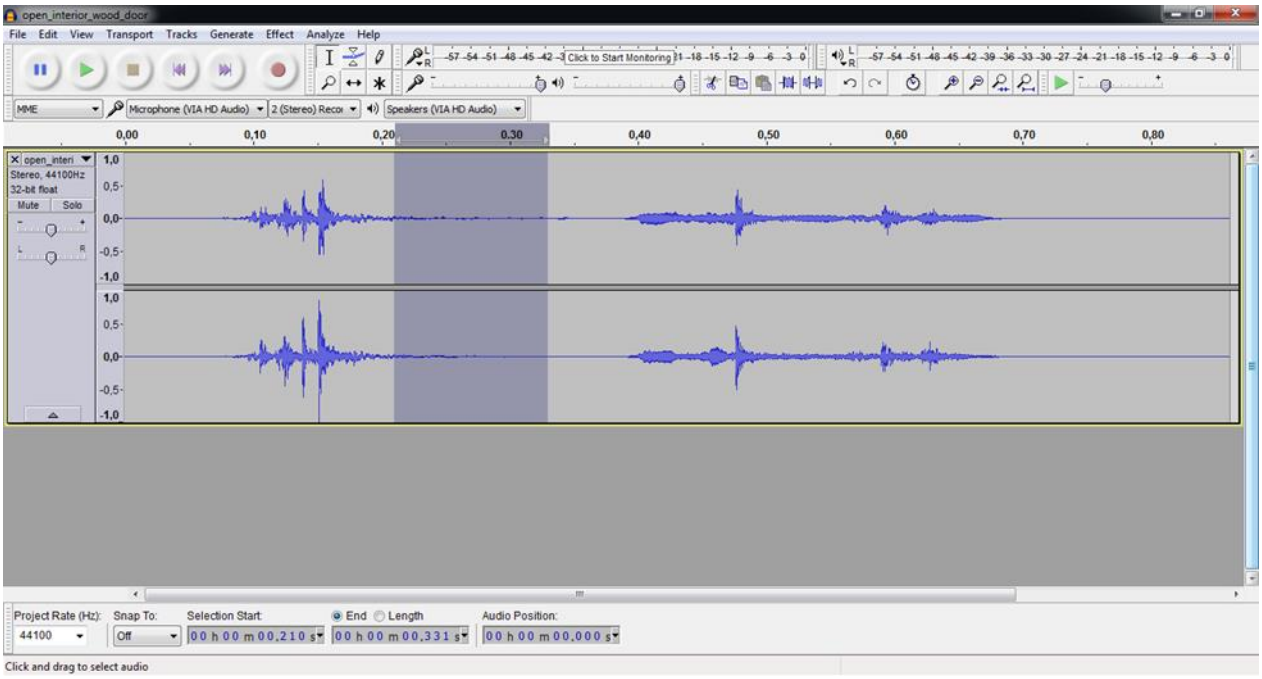
Aseprite [3] je program koji svojim korisnicima omogućava kreiranje grafičkog sadržaja koji se može koristiti prilikom izrade grafičkih animacija ili slika. Za ovaj završni rad Aseprite je izabran radi svog jednostavnog korisničkog sučelja, te svoje lagane izvedbe, odnosno male zahtjevnosti resursa. Sve što je potrebno kako bi se naučilo raditi unutar Aseprite-a je moguće pronaći na njihovoj službenoj stranici gdje se isto tako mogu pronaći i napredne tehnike crtanja i animiranja. Gotov program nije besplatan, međutim source kod je dostupan besplatno za skinuti te ga se može vlastoručno kompajlirati i koristiti. Sve što je vidljivo unutar igre je nacrtano pomoću Aseprite-a.



Sl. 2.3. Aseprite sučelje

2.4. AUDACITY

Audacity [4] je besplatan program koji se koristi za obradu zvuka prilikom izrade zvučnih datoteka. Audacity je cross-platform program što znači da ima podršku za rane operacijske sustave koji uključuju Windows i MacX i Linux. Audacity je kao program veoma jednostavan za korištenje, no omogućava jako složene i napredne operacije. Program je u ovom završnom korišten za skraćivanje i uređivanje glazbenih datoteka koje su korištene unutar igre kao što su na primjer glazba u glavnom izborniku, glazba unutar igre, zvučni efekti i slično. Program osim obrade glazbenih datoteka omogućava i snimanje te izvoz glazbenih datoteka u različitim formatima. U ovom završnom je korišten mp3 format.



Sl. 2.4. Audacity sučelje

3. RAZVOJ IGRE

3.1. Priča

Radnja je smještena u Osijeku 1984. godine. Prati život Igora, 27-godišnjeg propalog glazbenika, koji još živi s majkom. Zbog divljeg života isprepletanog pićem, zabavama i ženama Igor je zapao u velike dugove. Jednog dana mu pred vrata dolaze kamatari te mu daju rok od tjedan dana da skupi novce koje duguje inače će ga doći glave. Igor mora skupiti staru ekipu te pronaći način kako skupiti novce u tako malo vremena.

3.2. Likovi

3.2.1. Igor

Igor je glavni lik ove igre. Igrač kontrolira njegove kretnje i radnju klikom miša. Igor može međudjelovati sa svijetom u igri, skupljati stvari te ih kombinirati za rješavanje različitih problema i zagonetki.

Igor je svirač usne harmonike te je bio frontmen svojeg bivšeg benda *Vile i Kobile*. Kao tinejdžer je želio postati poznati glazbenik no zbog mladosti, neiskustva i luksuznog života je zapao u ogromne dugove.

Sl. 3.1. Igor

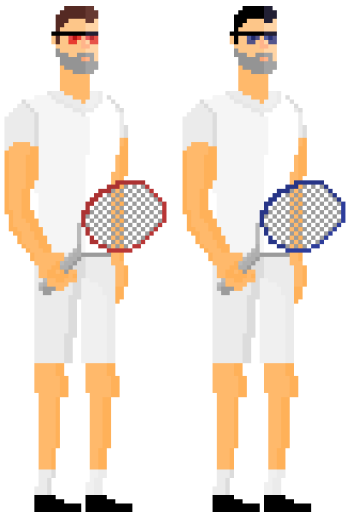
3.2.2. Boris

Boris je glavni antagonist ove igre te pokretač cijele radnje. Boris je šef osječkog podzemlja koji je Igoru pozajmio znatan iznos, te je došao po naplatu duga. Visokom inteligencijom i vezama po legalnim i ilegalnim dijelovima Osijeka sagradio je svoje carstvo skriveno od oka javnosti.

Sl. 3.2. Boris

3.2.3. Nele i Jole

Nele i Jole su bivši „prvaci“ u tenisu u paru. Diskvalificirani su zbog korištenja nedozvoljenih supstanci tijekom natjecanja. Također su dobili doživotnu zabranu natjecanja jer su sudcu slomili obje noge reketima. Vidjevši njihov „talent“, Boris ih je uzeo pod svoje krilo kao glavne ljude za prljave poslove. U slobodno vrijeme se bave reketom.



Sl. 3.3. Nele i Jole

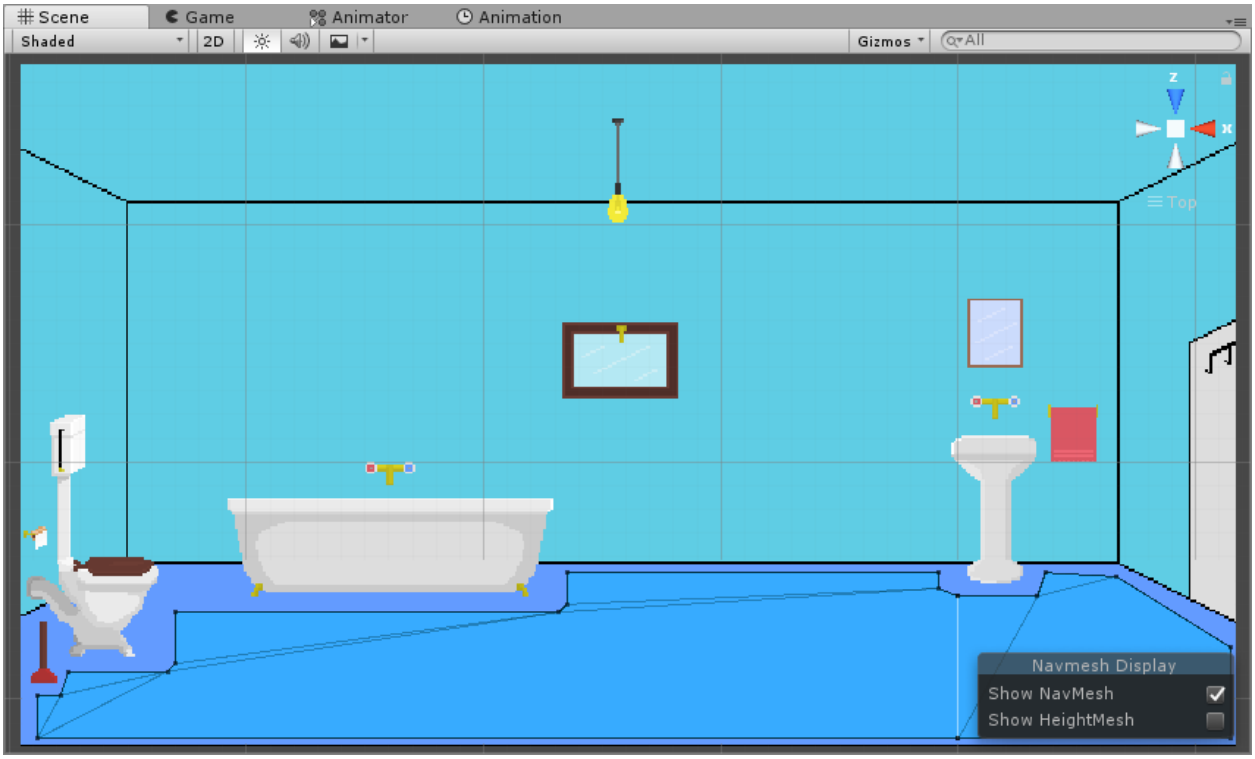
3.3. Kretanje lika po sceni

Kako bi se lik mogao kretati po sceni potrebno je imati NavMesh, NavMeshAgent te skriptu koja će upravljati njime.

3.3.1. Navigation system

Navigation system [5] omogućuje stvaranje likova koji se mogu inteligentno kretati po svijetu igre pomoću NavMesh-eva koji se automatski kreiraju iz geometrije scene.

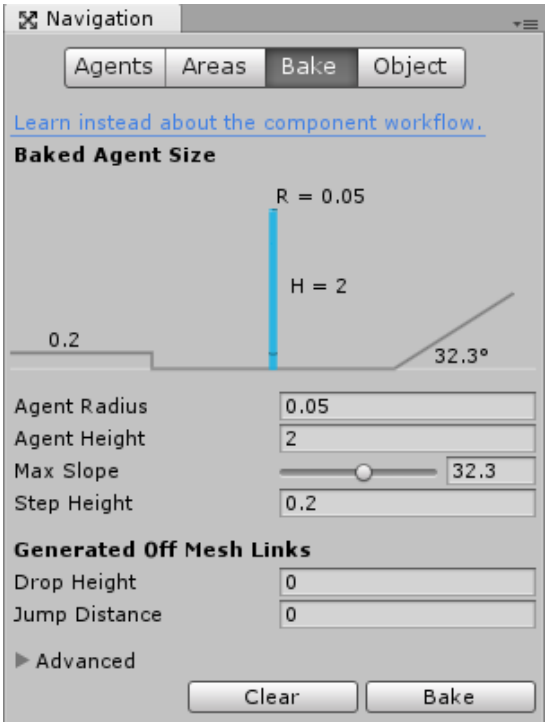
NavMesh [6] je klasa koja se može upotrijebiti za provođenje prostornih upita, kao što je testiranje putanja (pathfinding) i testove prohodnosti (walkability tests), postavljanje troškova pronalaženja staza za određene vrste područja i ugađanje globalnog ponašanja pronalaženja i izbjegavanja. U ovo slučaju je korišten za izbjegavanje prepreka u sceni. Da bi se koristili prostorni upiti, prvo se morate „ispeći“ (bake) NavMesh za scenu.



Sl. 3.4. NavMesh u sceni

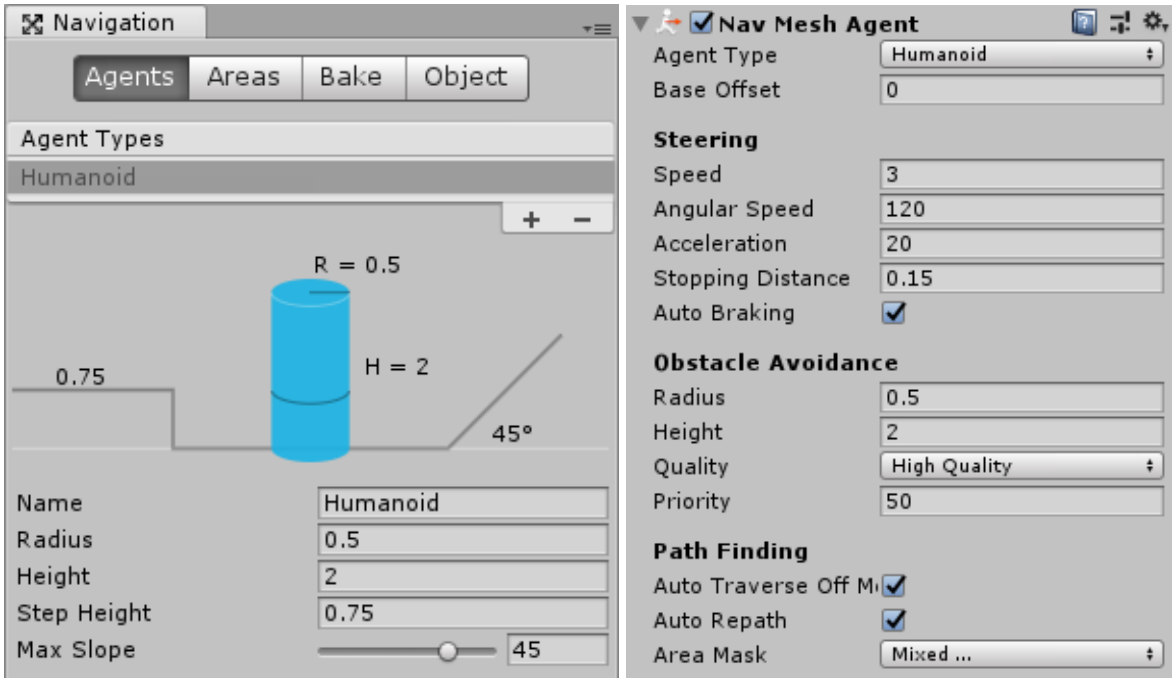
NavMesh baking [7] je proces stvaranja NavMesh-a iz geometrije scene. Postupak sakuplja Render Mesh-eve u sceni koji su označeni kao Navigation static, a zatim ih obrađuje

kako bi se stvorio NavMesh koji se procjenjuje prohodne površine scene. U Unity-ju, NavMesh baking se upravlja preko Navigation prozora (Window> AI> Navigation)



Sl. 3.5. NavMesh baking opcije

Nav mesh agent [8] komponenta se pridodaje mobilnom liku u igri kako bi mu omogućila navigaciju na sceni pomoću NavMesh-a.



Sl. 3.6. NavMesh Agent komponenta

3.3.2. Skripta za kretanje po sceni

Unity zasad ne podržava baking NavMesha za 2D igre zato su svi sprite-ovi i kamera rotirani za +90 stupnjeva po X osi. Lik kojeg igrač kontrolira je zapravo nevidljivi 3D model koji se kreće po sceni pomoću NavMesha, a sprite-u kojeg igrač vidi je pridodana skripta kojom prati nevidljivi 3D model. Kretnja lika se kontrolira klikom miša po sceni.

```
public class PlayerMovement : MonoBehaviour
{
    private void Start()
    {
        agent.updateRotation = false;
        inputHoldWait = new WaitForSeconds(inputHoldDelay);
        string startingPositionName = "";
        playerSaveData.Load(startingPositionKey, ref startingPositionName);
        Transform startingPosition =
StartingPosition.FindStartingPosition(startingPositionName);
        transform.position = startingPosition.position;
        transform.rotation = startingPosition.rotation;
        destinationPosition = transform.position;
    }

    private void Update()
    {
        if (agent.pathPending)
            return;
        float speed = agent.desiredVelocity.magnitude;
        if (agent.remainingDistance <= agent.stoppingDistance * stopDistanceProportion)
            Stopping(out speed);
        else if (agent.remainingDistance <= agent.stoppingDistance)
            Slowing(out speed, agent.remainingDistance);
    }

    private void Stopping(out float speed)
    {
        agent.isStopped = true;
        transform.position = destinationPosition;
        spriteAnimator.SetBool("isWalking", false);
        speed = 0f;

        if (currentInteractable)
        {
            currentInteractable.Interact();
            currentInteractable = null;
            StartCoroutine(WaitForInteraction());
        }
    }

    private void Slowing(out float speed, float distanceToDestination)
    {
        agent.isStopped = true;
        float proportionalDistance = 1f - distanceToDestination / agent.stoppingDistance;
        transform.position = Vector3.MoveTowards(transform.position, destinationPosition,
slowingSpeed * Time.deltaTime);
        speed = Mathf.Lerp(slowingSpeed, 0f, proportionalDistance);
    }
```

```
public void OnGroundClick(BaseEventData data)
    {
        if (!handleInput)
            return;
        spriteAnimator.SetBool("isWalking", true);
        currentInteractable = null;
        PointerEventData pData = (PointerEventData)data;
        NavMeshHit hit;
        if (NavMesh.SamplePosition(pData.pointerCurrentRaycast.worldPosition, out hit,
navMeshSampleDistance, NavMesh.AllAreas))
            destinationPosition = hit.position;
        else
            destinationPosition = pData.pointerCurrentRaycast.worldPosition;
        agent.SetDestination(destinationPosition);
        agent.isStopped = false;
    }

    public void OnInteractableClick(Interactable interactable)
    {
        if (MovementDisabled.satisfied)
            return;
        if (!handleInput)
            return;
        currentInteractable = interactable;
        destinationPosition = currentInteractable.interactionLocation.position;
        agent.SetDestination(destinationPosition);
        agent.isStopped = false;
    }

    private IEnumerator WaitForInteraction()
    {
        handleInput = false;
        yield return inputHoldWait;
        while (animator.GetCurrentAnimatorStateInfo(0).tagHash != hashLocomotionTag)
        {
            yield return null;
        }
        handleInput = true;
    }
}
```

Slika 3.7. Skripta za kretanje glavnog lika

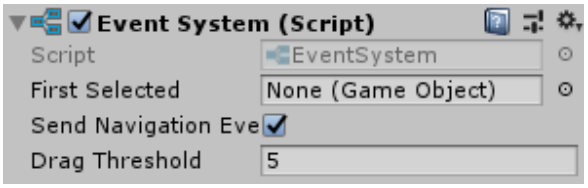
U ovu skriptu je ujedno implementirano i što se dogodi kada igrač klikne na interaktivni objekt s funkcijoma OnInteractableClick(Interactable).

3.4. Interakcija lika s okolinom

Kao glavna mehanika igre je interakcija lika s okolinom. Igrač može međudjelovati sa stvarima oko sebe, uzimati ih te ih spremiti u inventory, razgovarati s drugim likovima itd. Zato je napisana Interactable skripta. Ona se dodaje na objekte te kada igrač igrač stisne na njih dogodi se jedna ili više reakcija, koje mogu biti u obliku animacije, teksta, zvuka, spremanja itema-a u inventory, gubljenja itema iz inventory-ja ili promjene scene. To je napravljeno pomoću EventSystema i EventTriggera.

3.4.1. Event System

Event System [9] je način slanja eventa (događaja) na objekte u igri na temelju unosa, bilo da se radi o tipkovnici, mišu ili prilagođenom unosu. Event System se sastoji od nekoliko komponenti koje rade zajedno kako bi poslale event.



Sl. 3.8. Event System komponenta

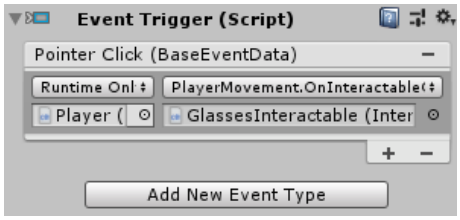
Raycasters

Raycasteri se koriste kako bi se saznalo iznad čega je pokazivač. Ulazni moduli ih često koriste. Unity pruža 3 vrste Raycastera: Graphic Raycaster (koristi se za UI elemente), Physics 2D Raycaster (koristi se za 2D fizičke elemente), Physics Raycaster (koristi se za 3D fizičke elemente)

Sl. 3.9. Physics Raycaster komponenta

3.4.2. Event Trigger

Event Trigger [10] prima evente iz Event Systema i poziva zabilježene funkcije za svaki event. Event Trigger se može koristiti za određivanje funkcija koje se žele pozvati za svaki event Event Systema. Jednom eventu se može dodijeliti više funkcija i svaki put kada Event Trigger primi taj event pozvat će te funkcije.



Sl. 3.10. Event Trigger komponenta

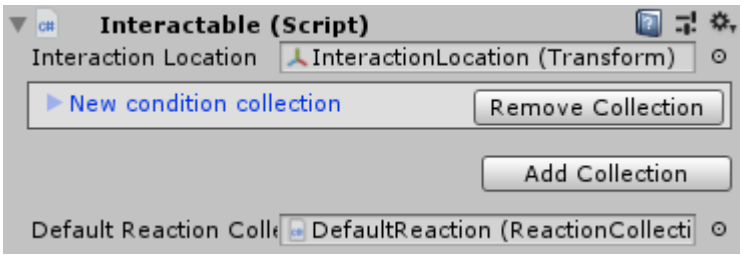
3.4.3. Skripte za interakciju

Nakon što igrač klikne na interaktivni objekt EventTrigger poziva funkciju OnInteractableClick() unutar skripte PlayerMovement koja uzima Intaractable komponentu kao input.

```
public class Interactable : MonoBehaviour
{
    public Transform interactionLocation;
    public ConditionCollection[] conditionCollections = new ConditionCollection[0];
    public ReactionCollection defaultReactionCollection;

    public void Interact()
    {
        for (int i = 0; i < conditionCollections.Length; i++)
        {
            if (conditionCollections[i].CheckAndReact())
                return;
        }
        defaultReactionCollection.React();
    }
}
```

Sl. 3.11. Interactable skripta

U Interactable skripti Interact() funkcija prolazi kroz uvjete iz ConditionCollectionsa te ih redom provjerava i izvode se određene reakcije ako je uvjet zadovoljen. Ako nijedan uvjet nije zadovoljen provode se default reakcije.



Sl. 3.12. Interactable komponenta s jednim Condition Collectionom

```
public class ConditionCollection : ScriptableObject
{
    public string description;.
    public Condition[] requiredConditions = new Condition[0];
    public ReactionCollection reactionCollection;

    public bool CheckAndReact()
    {
        for (int i = 0; i < requiredConditions.Length; i++)
```

```
{
            if (!AllConditions.CheckCondition(requiredConditions[i]))
                return false;
        }
        if (reactionCollection)
            reactionCollection.React();
        return true;
    }
}
```

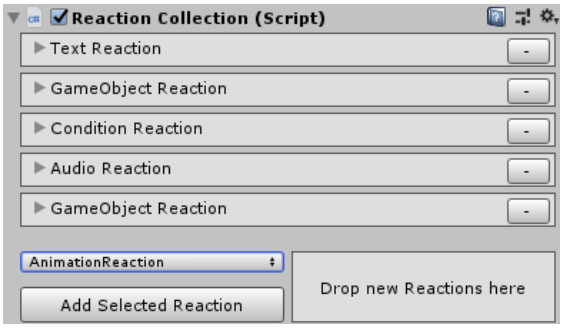
Sl. 3.13. ConditionCollection skripta

Condition Collection služi Interactable komponenti tako što provjerava jesu li uvjeti zadovoljeni te izvodi određene reakcije. Novi Condition Collection se dodaje pritiskom na gumb *Add Collection*, a uklanja pritiskom na gumb *Remove Collection*. Uvjeti se dodaju pritiskom na gumb +, a uklanjaju pritiskom na gumb -.

```
public class ReactionCollection : MonoBehaviour
{
    public Reaction[] reactions = new Reaction[0];

    private void Start()
    {
        for (int i = 0; i < reactions.Length; i++)
        {
                reactions[i].Init();
        }
    }

    public void React()
    {
        for (int i = 0; i < reactions.Length; i++)
        {
                reactions[i].React(this);
        }
    }
}
```

Sl. 3.14. ReactionCollection skripta

Reaction Collection služi Condition Collection, odnosno Interactable komponenti kako bi provela zadane reakcije.



Sl. 3.15. Reaction Collection komponenta

Nova reakcija se odabire padajućim izbornikom te se dodaje pritiskom na gumb *Add Selected Reaction*. Reakcije se uklanjaju pritiskom na gumb -.

3.5. Inventory

Igrač može pokupiti neke objekte iz okoline te ih staviti u svoj inventory. Te objekte (iteme) kasnije može upotrijebiti s ostalim objektima ili dati ih drugim likovima.

Sl. 3.16. Prazan i pun item slot

```
public class Inventory : MonoBehaviour
{
    public Image[] itemImages = new Image[numItemSlots];
    public Item[] items = new Item[numItemSlots];
    public const int numItemSlots = 4;

    public void AddItem(Item itemToAdd)
    {
        for (int i = 0; i < items.Length; i++)
        {
            if (items[i] == null)
            {
                items[i] = itemToAdd;
                itemImages[i].sprite = itemToAdd.sprite;
                itemImages[i].enabled = true;
                return;
            }
        }
    }

      public void RemoveItem(Item itemToRemove)
      {
          for (int i = 0; i < items.Length; i++)
          {
              if (items[i] == itemToRemove)
              {
                  items[i] = null;
                  itemImages[i].sprite = null;
                  itemImages[i].enabled = false;
                  return;
              }
          }
      }
  }
```

Sl. 3.17. Inventory skripta

Inventory skripta sadrži dvije funkcije, AddItem(Item) i RemoveItem(Item). Additem(Item) prolazi kroz itemslotove dok ne naiđe na prazan slot te ga zauzima itemom. RemoveItem(Item) prolazi kroz itemslotove dok ne naiđe na traženi item te ga uklanja sa slota.

Sl. 3.18. Inventory komponenta

Svaki item slot ima ItemImage element i Item element.ItemImage element predstavlja sprite koji će se prikazati na ekranu u tom item slotu, dok Item element predstavlja item u tom slotu u memoriji.

3.6. Spremanje podataka

Promjenom scene te njenim ponovnim učitavanjem stanje scene svega u njoj se vraća na početnu vrijednost. U point and click igrama je to nedopustivo jer tada radnja ne bi mogla napredovati. Stoga su napravljeni Savedata i Saver komponente koje spremaju trenutnu stanje scene te kada se scena ponovno učita bit će u spremljenom stanju.

3.6.1. Trajna scena

Trajna scena je učitana cijelo vrijeme dok je pokrenuta igra te služi kako bi sve zajedničke komponente iz ostalih scena držala na jednom mjest tijekom cijele igre. Ona sadrži trajni Canvas (služi za prikazivanje Inventoryja i teksta na ekranu), EventSystem(upravlja eventima), Pause Canvas (služi za prikaz Pause Menu-a) i Scene Controller (učitava scene na trajnu scenu).

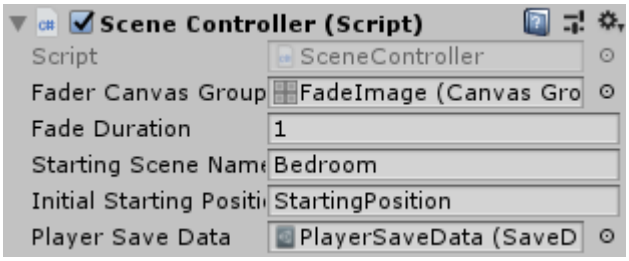
3.6.2. Scene Controller

Scene Contoller je iznimno bitan objekt u igri. Postoji u trajnoj sceni i upravlja sadržajem pri učitavanju scene. Radi na načelu da će se prvo učitati trajna scena, a zatim učitava scene koje sadrže igrača i druge vizualne elemente kada su potrebni.

```
public class SceneController : MonoBehaviour
{
    private IEnumerator Start()
    {
        faderCanvasGroup.alpha = 1f;
        playerSaveData.Save(PlayerMovement.startingPositionKey, initialStartingPositionName);
        yield return StartCoroutine(LoadSceneAndSetActive(startingSceneName));
        StartCoroutine(Fade(0f));
    }

    public void FadeAndLoadScene(SceneReaction sceneReaction)
    {
        if (!isFading)
        {
            StartCoroutine(FadeAndSwitchScenes(sceneReaction.sceneName));
        }
    }

    private IEnumerator FadeAndSwitchScenes(string sceneName)
    {
        yield return StartCoroutine(Fade(1f));
        if (BeforeSceneUnload != null)
            BeforeSceneUnload();
        yield return SceneManager.UnloadSceneAsync(SceneManager.GetActiveScene().buildIndex);
        yield return StartCoroutine(LoadSceneAndSetActive(sceneName));
        if (AfterSceneLoad != null)
            AfterSceneLoad();
        yield return StartCoroutine(Fade(0f));
    }

    private IEnumerator LoadSceneAndSetActive(string sceneName)
    {
        yield return SceneManager.LoadSceneAsync(sceneName, LoadSceneMode.Additive);
        Scene newlyLoadedScene = SceneManager.GetSceneAt(SceneManager.sceneCount - 1);
        SceneManager.SetActiveScene(newlyLoadedScene);
    }

    private IEnumerator Fade(float finalAlpha)
    {
        isFading = true;
        faderCanvasGroup.blocksRaycasts = true;
        float fadeSpeed = Mathf.Abs(faderCanvasGroup.alpha - finalAlpha) / fadeDuration;
        while (!Mathf.Approximately(faderCanvasGroup.alpha, finalAlpha))
        {
            faderCanvasGroup.alpha = Mathf.MoveTowards(faderCanvasGroup.alpha, finalAlpha,
                fadeSpeed * Time.deltaTime);
            yield return null;
        }
        isFading = false;
        faderCanvasGroup.blocksRaycasts = false;
    }
}
```

Sl. 3.19. SceneController skripta

Prva stvar koju Scene Controller učini pri promjeni scene je postepeno nestajeanje (engl. fade in/out) ekrana. Time sakriva oka igrača ono što se događa u sceni. Zatim blokira Raycaste kako igrač ne bi mogao upravljati likom i okolinom dok se ne učita druga scena. Potom unloada trenutnu scenu, učita početnu poziciju igrača, učita drugu scenu te nakon svega toga makne fade s ekrana i odblokira Raycaste kako bi igrač ponovno mogao upravljati likom.



Sl. 3.20. Scene Controller komponenta

3.6.3. Saveri

Saveri spremaju podatke u SaveDate-ove. Ovisno o tipu savera spremaju se različiti tipovi podataka: pozicija lika, stanje objekata...

```
public abstract class Saver : MonoBehaviour
{
    public string uniqueIdentifier;
    public SaveData saveData;
    protected string key;
    private SceneController sceneController;

    private void Awake()
    {
        sceneController = FindObjectOfType<SceneController>();
        key = SetKey();
    }

    private void OnEnable()
    {
        sceneController.BeforeSceneUnload += Save;
        sceneController.AfterSceneLoad += Load;
    }

    private void OnDisable()
    {
        sceneController.BeforeSceneUnload -= Save;
        sceneController.AfterSceneLoad -= Load;
    }

    protected abstract string SetKey();

    protected abstract void Save();

    protected abstract void Load();
}
```

Sl. 3.21. Saver skripta

Ovo je apstraktna klasa koja je osnovna za sve ostale klase koje spremaju podatke između scena.

Sl. 3.22. Game Object Activity Saver komponenta

3.6.4. SaveData

Instanca ove klase se može stvoriti kao asset. Svaka instanca sadrži kolekcije podataka iz Savera na koji imaju referencu. Budući da asseti postoje izvan scene podaci će ostati spremni za ponovno učitavanje sljedeći put kada se scena učita.

Sl. 3.23. Save Data Asset

3.7. Vrste korištenih objekata tijekom igre

3.7.1. Game Object

Game Object [11] je temeljni objekt u Unityju koji može predstavljati likove, rekvizite i okolinu. Sami ne postižu puno nego djeluju kao spremnici za druge komponete, koji implementiraju stvarnu funkcionalnost. Na primjer, Light objekt se stvara dodavanjem Light komponente u Game Object, a čvrsti objekt kocke (cube) ima Mesh Filter i Mesh Renderer komponentu, koji služe za crtanje površine kocke i Box Collider komponentu koja predstavlja čvrst volumen objekta u fizičkom smislu. Game Object uvijek ima pridodanu Transform

komponentu (koja predstavlja položaj i orijentaciju) i nije ju moguće ukloniti. Druge komponente koje daju objektu funkcionalnost mogu se dodati iz izbornika u Editor-u ili iz skripte.

3.7.2. Sprite

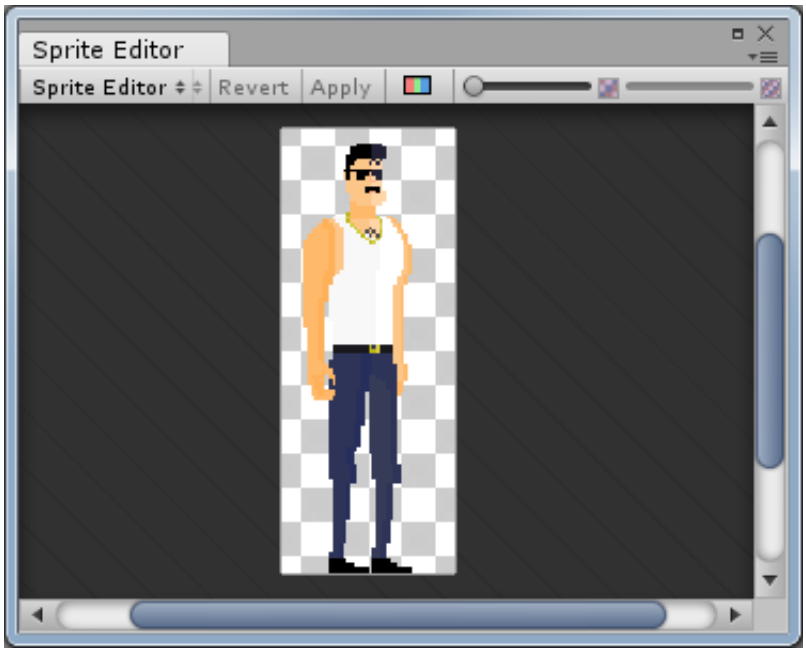
Sprite [12] je 2D grafičko objekt. U usporedbi s 3D objektima, Sprite-ovi su u osnovi samo standardne teksture, ali postoje posebne tehnike za kombiniranje i upravljanje sprite teksturama za učinkovitost i praktičnost tijekom razvijanja igre. Unity pruža Sprite Creator, ugrađeni Sprite Editor, Sprite Renderer i Sprite Packer.

Sprite Creator

Sprite Creator se koristi za stvaranje placeholder Sprite-ova koji rezerviraju mjesta u projektu, tako da se može nastaviti s razvojem bez potrebe za čekanjem na grafiku.

Sprite Editor

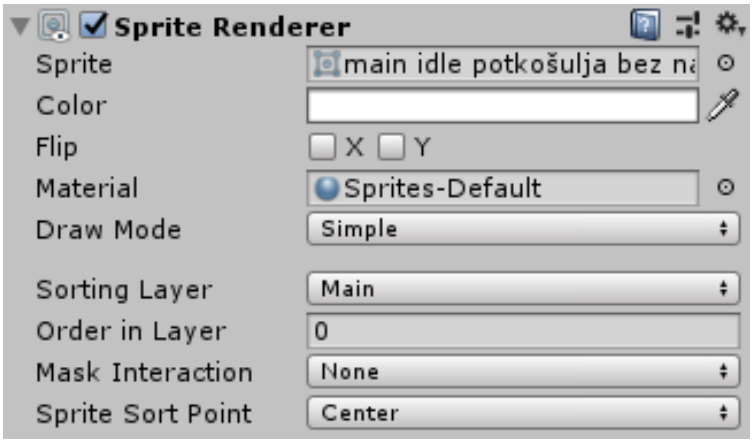
Sprite Editor omogućuje izdvajanje Sprite grafike s veće slike i uređivanje brojnih komponenti slike unutar jedne teksture. Može se upotrijebiti, na primjer, kako bi držali ruke, noge i tijelo lika kao odvojene elemente unutar jedne slike.



Sl. 3.24. Sprite Editor

Sprite Renderer

Sprite-ovi se prikazuju s Sprite Renderer komponentom umjesto Mesh Renderer komponente koja se koristi za prikazivanje 3D objekata. Služi za prikazivanja Sprite-ova i u 2D i 3D scenama.



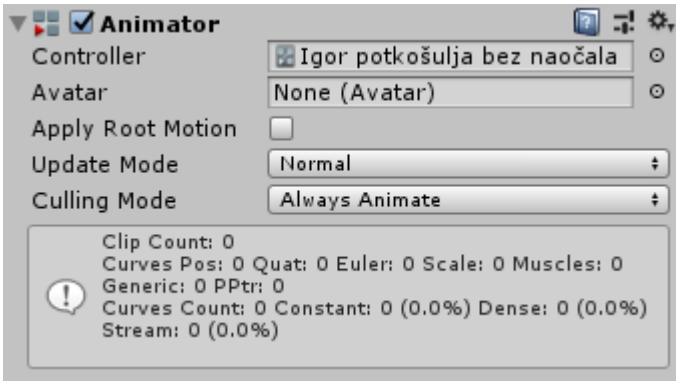
Sl. 3.25. Sprite Renderer komponenta

Sprite Packer

Sprite Packer se koristi za optimizaciju uporabe i performansu video memorije igre. Značajan dio teksture Sprite-ove često će zauzimati prazni prostor između grafičkih elemenata i taj će prostor rezultirati uzalud utrošenom video memorijom. Za optimalne performanse, najbolje je pakirati grafiku iz nekoliko sprite tekstura zajedno, bez previše praznog prostora, unutar jedne teksture poznate kao atlas.

3.7.3. Animator

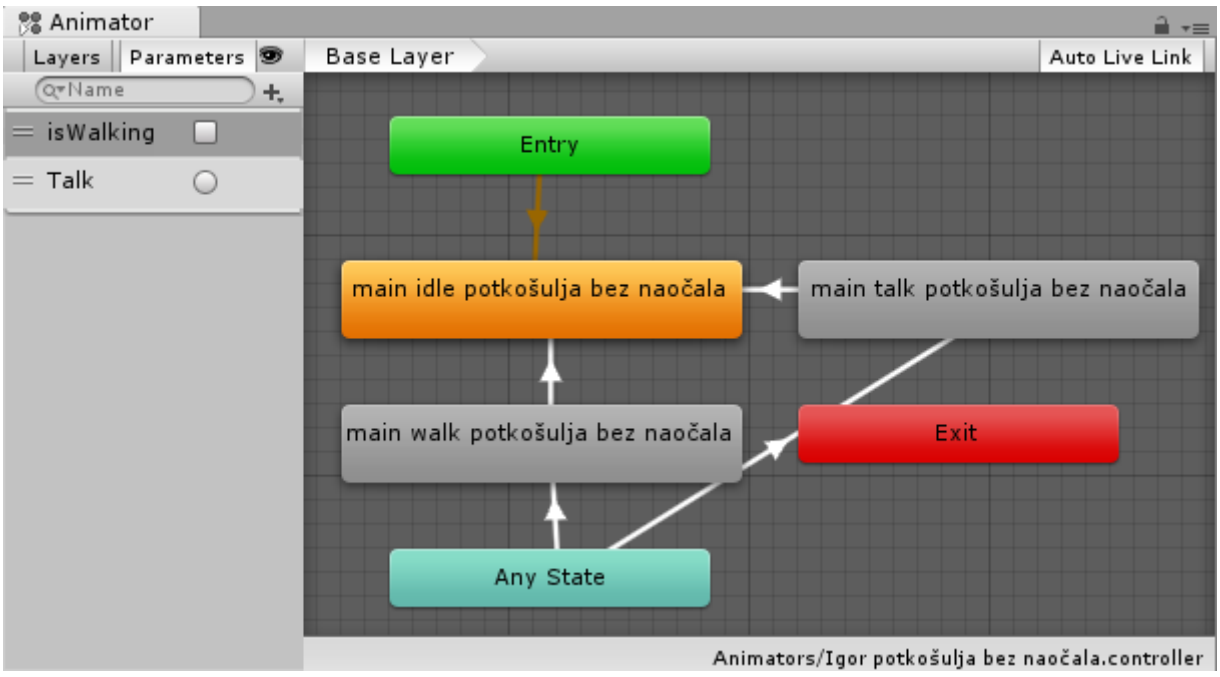
Animator [13] komponenta se koristi za dodjeljivanje animacije GameObject-u u sceni. Animator komponenta zahtijeva referencu na Animator Controller koji definira koji će se animacijski isječci koristiti i kontrolira kada i kako napraviti prijelaz između njih.



Sl. 3.26. Animator komponenta

Animator Contoller

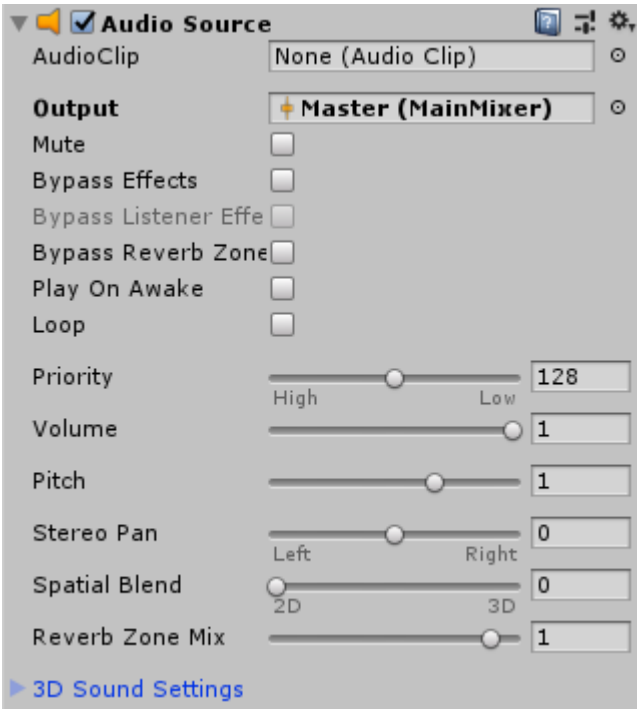
Animator Controller [14] omogućuje organiziranje skupa animacijskih isječaka i prijelaza između animacija za lika ili objekta. U većini slučajeva, normalno je imati više animacija i prebacivati se između njih kada se pojave zadovolje određeni uvjeti igre. Na primjer, može se prebaciti iz animacije za hodanje u animaciju za skok svaki put kad se pritisne razmaknica. Međutim, čak i ako imate samo jednu animacijsku isječak, još uvijek ga treba staviti u Animator Controller da bi se mogao koristiti na Game Objectu.



Sl. 3.27. Animator Contoller

3.7.4. Audio Source

Audio Source [15] je priključen na GameObject za reprodukciju zvukova u 3D okruženju. Za reproduciranje 3D zvukova, također treba imati Audio Listener. Audio Listener je obično priključen na kameru koja se koristi. Reproducira li se zvuk u 3D ili 2D okruženju određuju postavke u Audio Importer-u. Pojedini zvučni isječci se mogu reproducirati pomoću Play, Pause and Stop. Može se prilagoditi i glasnoća tijekom reprodukcije pomoću Volume svojstva. vrijeme. Višestruki zvukovi se mogu reproducirati na jednom Audio Source-u koristeći PlayOneShot. Može se reproducirati isječak na statičnom položaju u 3D prostoru koristeći PlayClipAtPoint.



Sl. 3.28. Audio Source komponenta

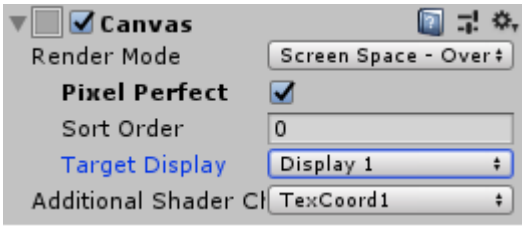
Audio Listener

Audio Listener [16] komponenta implementira uređaj sličan mikrofonu. Bilježi zvukove oko sebe i reproducira ih putem zvučnika uređaja. U jednoj sceni se može imati samo jedan Audio Listener.

3.7.5. Canvas

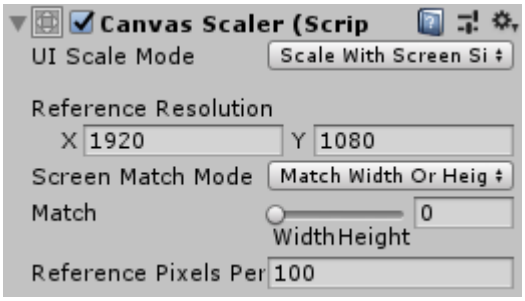
Canvas [17] komponenta predstavlja apstraktni prostor u kojem je korisničko sučelje (UI) postavljeno i izvedeno. Svi elementi UI-ja moraju biti djeca (children) GameObject-a koji ima pridodanu komponentu Canvas. Kada je UI elementa stvoren iz izbornika (GameObject> Create

UI), Canvas objekt će biti kreiran automatski ako već nema jednog na sceni. Jedna Canvas koponenta je dovoljna za sve elemente korisničkog sučelja, ali više Canvas komponenti na sceni je moguće. Također je moguće upotrijebiti ugniježđene Canvas-e, gdje je jedan Canvas postavljen kao dijete drugog u svrhu optimizacije. Ugniježđeni Canvas koristi isti Render Mode kao njegov roditelj.



Sl. 3.29. Canvas komponenta

Canvas Scaler [18] komponenta se koristi za kontrolu cjelokupne veličine i gustoće piksela UI elemente na Canvas-u. Ta skala utječe na sve što je ispod platna, uključujući veličine fonta i granice slike.

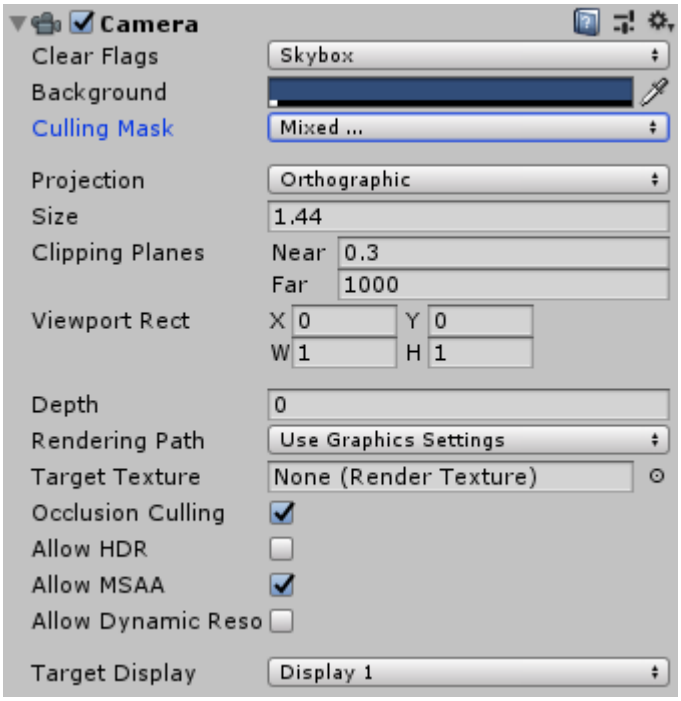


Sl. 3.30. Canvas Scaler komponenta

3.7.6. Camera

Kamere (Camera) [19] su uređaji koji prikazuju svijet igraču. Prilagođavanjem i manipuliranjem kamera, prezentacija igre se može učiniti jedinstvenom. Može se imati neograničen broj kamera u sceni. Moguće ih je postaviti na prikazivanje bilo kojim redoslijedom, bilo gdje na zaslonu ili samo na određenim dijelovima zaslona. Projekcija može biti Orthographic ili Perspective. Korišten je Orthographic projection jer je igra u 2D okruženju.

Kamere su neophodne za prikaz igre igraču. Mogu se prilagoditi, napraviti skriptu za njih, ili postaviti kao parent drugim objektima kako bi se postigao bilo kakav efekt zamisliv. Za igre sa zagonetkama, može se držati statička kamera za puni prikaz zagonetke. Za pucačine u prvom licu (First-person shooter) kamera će se postaviti kao parent na lik igrača i postaviti je na razinu očiju lika. Za trkaću igru kamera bi slijedila vozilo igrača.



Sl. 3.31. Camera komponenta

3.7.7. Pixel Perfect Camera

Pixel Perfect Camera [20] komponenta osigurava da svježinu i jasnoću Sprite-ova pri različitim rezolucijama te njihovu stabilnost u pokretu. Čini sve izračune potrebne za skaliranje okvira s promjenama razlučivosti, uklanjajući gnjavažu korisniku. Korisnik može prilagoditi definiciju Sprite-ova prikazanu u okviru kamere kroz postavke komponenata, kao i pregledati sve promjene odmah u prikazu igre pomoću značajke Pokreni u načinu rada za uređivanje.

Sl. 3.32. Pixel Perfect Camera komponenta

3.8. Grafičko korisničko sučelje (GUI)

Grafičko korisničko sučelje [21] (engl. Graphical User Interface - GUI) je vrsta korisničkog sučelja koje korisnicima omogućuje interakciju s elektroničkim uređajima pomoću grafičkih ikona i vizualnih pokazatelja kao što je sekundarna notacija, umjesto korisničkog sučelja temeljenog na tekstu, upisivanjem zapovjednih linija ili tekstualne navigacije. Grafička korisnička sučelja su nastala zbog strmu krivulju učenja za korištenje sučelja naredbenog retka (engl. Command-line Interface - CLIs), koji zahtijevaju da se naredbe upisuju na tipkovnici računala. Većina današnjih programa pa tako i Unity imaju grafičko sučelje koje znatno olakšava i ubrzava rad u njemu. Projektiranje grafičkog korisničkog sučelja važan je dio programiranja aplikacija u području interakcije čovjek-računalo. Njegov je cilj povećati učinkovitost i jednostavnost korištenja za temeljni logički dizajn pohranjenog programa.

3.8.1. Glavni Izbornik

Pri pokretanju igre prva scena koja se učita je glavni izbornik. Na glavnom izborniku se nalazi ime igre, pozadinska slika i 3 gumba: Play, Options i Quit. Kao što se može vidjeti na slici 6.8.1.1. Glavni izbornik je stvoren korištenjem Canvasa i Buttona objekata te posebnom skriptom koja upravlja njima.

Sl. 3.33. Glavni izbornik

Gumb Play je povezan skriptom MainMenu koja učitava prvu igrivu scenu tj. započinje novu igru. Gumb Options otvara izbornik opcija u kojem je moguće odabrati razinu glasnoće igre, dok pritiskom na gumb Quit izlazimo iz igre.

Izbornik opcija sadrži Volume slider koji je povezan s Audio Mixerom te upravlja glasnoćom u igri.

3.8.2. Pauzirajući izbornik

Dok je igra u tijeku, pritiskom na tipku Escape, moguće ju je pauzirati te otvoriti pauzirajući izbornik. U pauzirajućem izborniku se nalaze 4 gumba: Resume, Options, Menu i Quit. Ponovnim pritiskom na Escape ili Resume, pauzirajući izbornik se zatvara te se igra nastavlja. Kao i u glavnom izborniku gumb Options otvara izbornik opcija u kojem se nalazi Volume slider za određivanje razine glasnoće. Gumb Menu vraća igrača nazad na glavni izbornik, pritiskom dok gumb Quit izlazimo iz igre. Pauzirajući izbornik rađen je na Pause Canvasu koji se nalazi u trajnoj sceni kako bi mu igrač mogao pristupiti neovisno u kojoj se sceni nalazi. Gumbi su povezani PauseMenu skriptom.

Sl. 3.34. Pauzirajući izbornik

4. ZAKLJUČAK

Ideja ovog završnog je bila izrada 2D Point and click avanture u Unity engine-u, a pritom koristeći C# programski jezik. Stvoren je sustav kretanja po sceni i sustav interaktiranja s okolinom te Inventory sustav za skupljanje stvari iz okoline. Za sve to je također korišten Event System u Unitiy-u koji upravlja događajima ovisno o odabranim funkcijama. Opisani su svi dijelovi igre kao što su priča, grafika, izbornici, kretanje lika te skripte potrebne za njihovo pokretanje. Za crtanje grafike i animacija korišten je program Aseprite, a glazba i zvučni efekti su preuzeti s interneta te obrađeni besplatnim programom Audacity za obradu zvučnih datoteka. Sve skripte su napisane u Microsoft Visual studio programskom ukruženju.

LITERATURA

[1] Unity (game engine) – 10.9.2018.

https://en.wikipedia.org/wiki/Unity_(game_engine)

[2] Microsoft Visual Studio – 15.8.2018.

https://visualstudio.microsoft.com/

[3] Aseprite (Animated Sprite Editor & Pixel Art) – 29.8.2018.

https://www.aseprite.org/

[4] Audacity (Free, open sourece, cross-platform audio software) – 1.9.2018.

https://www.audacityteam.org/

[5] Navigation System in Unity – 14.9.2018.

https://docs.unity3d.com/Manual/nav-NavigationSystem.html

[6] NavMesh – 14.9.2018.

https://docs.unity3d.com/ScriptReference/AI.NavMesh.html

[7] NavMesh baking – 14.9.2018.

https://unity3d.com/learn/tutorials/topics/navigation/navmesh-baking

[8] NavMesh Agent – 14.9.2018.

https://docs.unity3d.com/ScriptReference/AI.NavMeshAgent.html

[9] Event System – 14.9.2018.

https://docs.unity3d.com/Manual/EventSystem.html

[10] Event Trigger – 14.9.2018.

https://docs.unity3d.com/Manual/script-EventTrigger.html

[11] Game Object – 14.9.2018.

https://docs.unity3d.com/Manual/class-GameObject.html

[12] Sprites – 14.9.2018.

https://docs.unity3d.com/Manual/Sprites.html

[13] Animator – 14.9.2018.

https://docs.unity3d.com/Manual/class-Animator.html

[14] Animator Controller – 14.9.2018.

https://docs.unity3d.com/Manual/class-AnimatorController.html

[15] Audio Source – 14.9.2018.

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

[16] Audio Listener – 14.9.2018.

https://docs.unity3d.com/Manual/class-AudioListener.html

[17] Canvas – 14.9.2018.

https://docs.unity3d.com/Manual/UICanvas.html

[18] Canvas Scaler – 14.9.2018.

https://docs.unity3d.com/Manual/script-CanvasScaler.html

[19] Camera – 14.9.2018.

https://docs.unity3d.com/Manual/class-Camera.html

[20] Pixel Perfect Camera – 14.9.2018.

https://github.com/Unity-Technologies/2d-pixel-perfect

[21] Graphic user interface (GUI) – 15.9.2018.

https://en.wikipedia.org/wiki/Graphical_user_interface

[22] Adventure Game tutorial - Unity – 8.9.2018.

https://unity3d.com/learn/tutorials/projects/adventure-game-tutorial

[23] Brackeys – 13.9.2018.

https://www.youtube.com/channel/UCYbK_tjZ2OrIZFBvU6CCMiA

SAŽETAK

Ideja ovog rada je izrada 2D point-and-click igre u Unity engine-u. Igra se zasniva na međudjelovanju okolinom i rješavanjem zagonetki kako bi se nastavila priča. Glavni lik igre je Igor, a igrač njime upravlja mišem. Igor se može kretati po sceni te djeluje s objektima u sceni koji utječu na radnju. Cilj mu je skupiti dovoljno novaca kako bi vratio dug koji duguje Borisu, šefu Osječkog podzemlja. Svi sprite-ovi i animacije u igri su nacrtani grafičkim programom Aseprite. Glazba i zvučni efekti preuzeti su s interneta i obrađeni u besplatnom programu za obradu zvučnih datoteka Audacity. Unity omogućuje korištenje C# skripti za upravljanje scenama i objektima u igri te je korišteno Microsoft Visual Studio okruženje za pisanje tih skripti. Igra sadrži 6 scena. Prve dvije scene su glavni izbornik i trajna scena, dok ostale su 4 scene nivoi u igri.

Ključne riječi: 2D point-and-click avantura, Unity Engine, C#, Igor, Boris, Aseprite, Audacity, Microsoft Visual studio

ABSTRACT

Title: Development of 2D point-and-click adventure game

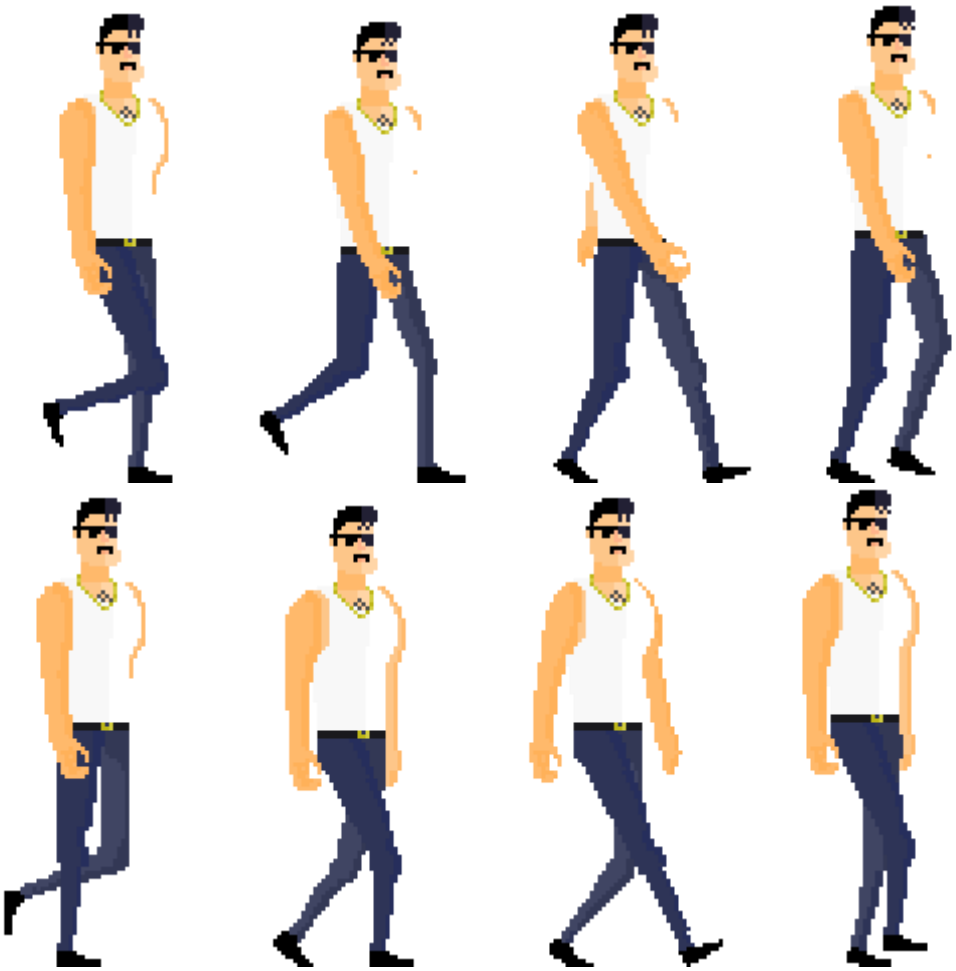
The idea of this final paper is to create 2D point-and-click games in Unity Engine. The game is based on interacting with the environment and solving riddles to keep the story going. The main character of the game is Igor, and the player controlls him with the mouse. Igor can move around the scene and interact with objects in the scene that affect the game. His goal is to raise enough money to repay the debt owed to Boris, the head of the Osijek underground. All sprites and animations in the game are drawn with the graphic program Aseprite. Music and sound effects are downloaded from the Internet and processed in a free program for audio processing Audacity. Unity allows the use of C# scripts to manage scenes and objects in the game, so Microsoft Visual Studio was used as an environment to write those scripts. The game contains 6 scenes. The first two scenes are the main menu and the persistent scene, while the other four scenes are levels in the game.

ŽIVOTOPIS

Nikola Hlavsa je rođen 20.11.1995. godine u Našicama. Trenutno prebiva u Brezovici kod Marijanaca. Završio je Osnovnu školu Matije Gupca u Magadenovcu, gdje je išao na županijska natjecanja iz matematike i kemije. Nakon toga je upisao prirodoslovno-matematičku gimnaziju u Srednjoj školi Isidora Kršnjavog u Našicama. Završetkom srednje škole upisuje se na preddiplomski studij računarstvna na Elekrotehnički fakultet u Osijeku.

PRILOZI

Sl. 1. Igor idle



Sl. 2. Igor walk