Korištenje C# programskog jezika u izradi računalne igre za dva igrača

Bareš, Mateo

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

Preddiplomski studij računarstva

Korištenje C# programskog jezika u izradi računalne igre za dva igrača

Završni rad

Mateo Bareš

Osijek, 2016.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

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

Osijek, 06.07.2016.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada

Ime i prezime studenta:	Mateo Bareš
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3536, 30.07.2013.
OIB studenta:	72779015464
Mentor:	Doc.dr.sc. Časlav Livada
Sumentor:	
Naslov završnog rada:	Korištenje C# programskog jezika u izradi računalne igre za dva igrača
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 Postignuti rezultati u odnosu na složenost zadatka: 3 Jasnoća pismenog izražavanja: 3 Razina samostalnosti: 3
Datum prijedloga ocjene mentora:	06.07.2016.
Datum potvrde ocjene Odbora:	28.09.2016.

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, 28.09.2016.

Ime i prezime studenta:	Mateo Bareš
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3536, 30.07.2013.
Ephorus podudaranje [%]:	0

Ovom izjavom izjavljujem da je rad pod nazivom**: Korištenje C# programskog jezika u izradi računalne igre za dva igrača**

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

Cilj ovog završnog rada je izrada računalne igre za dva igrača korištenjem C# programskog jezika. Igra sadržava 2D grafiku i namijenjena je za igru na jednom računalu upotrebom tipkovnice. Tema igre je borba jednog igrača protiv drugog i oba igrača protiv računalno kontroliranog neprijatelja s ciljem stjecanja vrijednog blaga. Igra je podržana na Windows, Linux i Mac OS operacijskim sustavima. Izrada igre je ostvarena u Unity *game engineu*. Upotrebom C# programskog jezika napisani su kodovi koji određuju mehaniku izvođenja igre, funkciju pojedinih objekata u igri, upravljanje likovima i slično. Grafika je izrađena pomoću web aplikacije Piskel [1]. Sve tehnologije korištene za izradu igre i bitni elementi igre opisani su u narednim poglavljima ovog završnog rada.

1.1. Zadatak završnog rada

U ovom završnom radu potrebno je napraviti računalnu igru za dva igrača korištenjem C# programskog jezika. C# programski jezik se koristi u pisanju koda unutar Unity *game enginea* u čijem okruženju se igra izrađuje. Napisani kod treba omogućiti svu funkcionalnost jedne računalne igre za dva igrača.

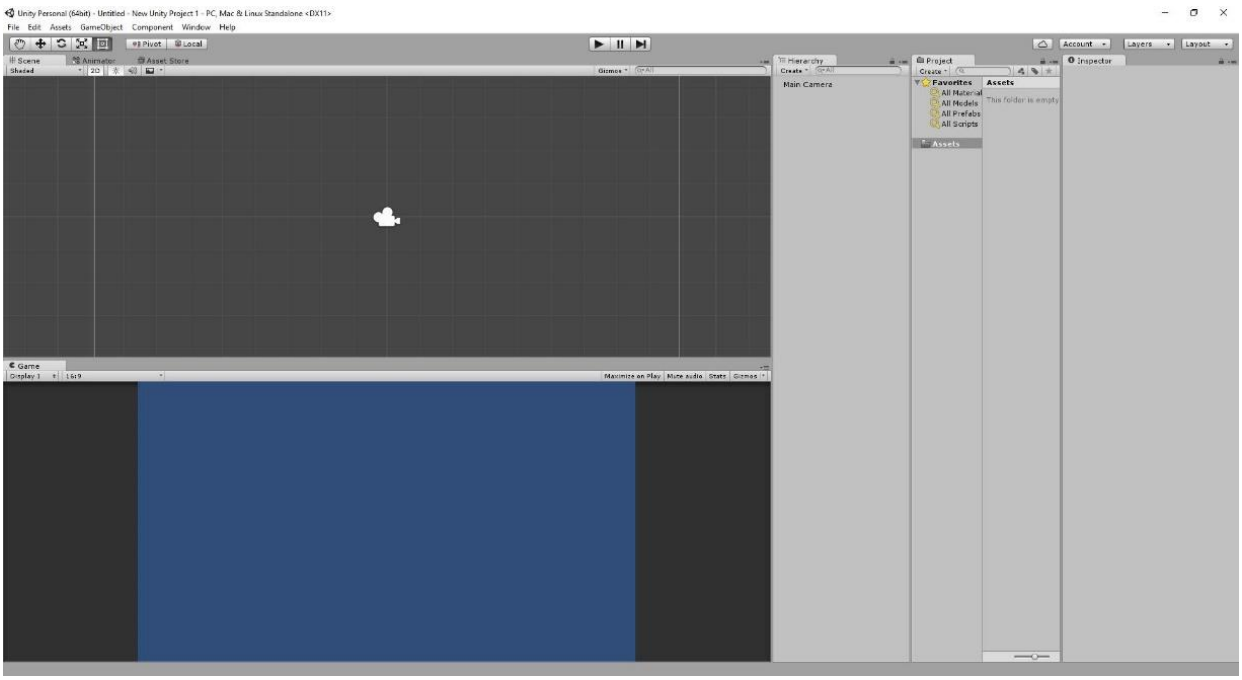
2. KORIŠTENE TEHNOLOGIJE I ALATI

2.1 C#

C# je objektno orijentirani programski jezik koji omogućuje izradu raznovrsnih aplikacija pokretanih .NET Frameworkom. Sintaksa jezika je slična kao i u C, C++ i Java programskim jezicima. Sintaksa C# pojednostavljuje mnoge kompleksnosti C++ i pruža dodatne mogućnosti koje nisu dostupne u Java programskom jeziku. Kao objektno orijentirani programski jezik, C# podržava koncepte učahurivanja ili enkapsulacije, nasljeđivanja i višeobličja ili polimorfizma [2].

2.2 Unity

Unity je *game engine* kojeg je razvila tvrtka Unity Technologies 2005. godine. Koristi se za razvoj 2D i 3D videoigara i interaktivnog sadržaja koji se mogu pokretati na više od 20 platformi na računalima, konzolama, mobilnim uređajima i web stranicama [3]. Izrada igre odvija se unutar Unity uređivača u kojem se izrađuju scene i manipuliraju objekti i grafičko sučelje. Uređivač podržava razne slikovne, audio, video i tekstualne formate koji se mogu uvesti i koristiti kao nove objekte, teksture, zvukove i slično [4].

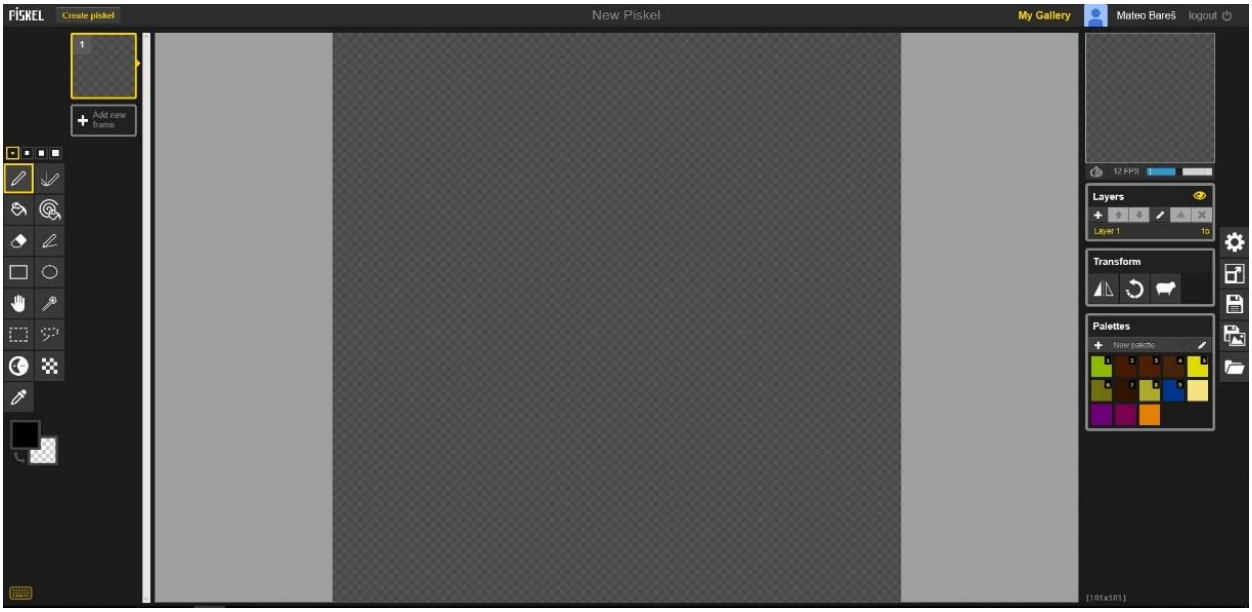


Sl. 2.1. Izgled sučelja Unity uređivača

Slika 2.1. prikazuje sučelje Unity uređivača. Prostor u kojemu se stvara igra dodavanjem i manipuliranjem objekata naziva se scena. Većina igara je sastavljeno od više scena, one mogu predstavljati različite razine igre ili izbornike. U Unityju kodovi se pišu u skripte koje se onda dodaju kao atribut objekta koji je dodan u scenu. Kodovi se mogu pisati u UnityScript ili C# programskom jeziku. U ovom završnom radu koristi se C# programski jezik.

2.3 Piskel

Piskel je web aplikacija za izradu 2D pikselizirane grafike i animiranje izrađene grafike. Aplikacija podržava Chrome, Firefox i Microsoft Edge pretraživače.



Sl. 1.2. Izgled sučelja Piskela

Slika 2.2. prikazuje sučelje Piskel aplikacije. Postoji i *offline* verzija aplikacije napravljena za računala. Piskel je aplikacija otvorenog koda [1]. Svi *spriteovi* za ovu igru su napravljeni upotrebom Piskela. *Sprite* je 2D grafički objekt koji se koristi za likove, projektile i ostale elemente 2D okružja [5].

3. IZRADA IGRE

3.1. Ideja

Srž ove računalne igre je međusobna borba dva igrača i njihova borba protiv računalno upravljanog neprijatelja. Igrači imaju izbor od tri različita lika, svaki sa svoje dvije posebne sposobnosti i posebnim napadom.

Borba se odvija horizontalno i vertikalno, u prostoru koji sadrži platforme različitih veličina i na različitim visinama. U tom prostoru postoje lava, šiljci i padajuće vatrene kugle koje oštećuju igračevog lika ako on dođe u doticaj s njima.

Tijekom borbe računalno kontrolirani neprijatelj se pojavljuje na nasumičnim mjestima, gdje koristi svoj poseban napad u jednom od četiri smjera (gore, dolje, lijevo, desno) koji je nasumično odabran i nakon nasumičnog vremenskog perioda neprijatelj nestaje i pojavljuje se opet nakon nekog vremenskog perioda ponavljajući objašnjen proces.

Igrači i neprijatelj imaju svoje životne bodove koji su brojčano jednaki i iznose sto bodova. Ako životni bodovi jednog od igrača ili neprijatelja dostignu nulu ili prođu ispod nule, igrač ili računalno neprijatelj se smatra pobijeđenim. Igra je završena kada samo jedan igrač ili samo neprijatelj ostane živ.

Likovi koje igrač može izabrati su vitez (engl. *knight*), čarobnjak (engl. *mage*) i kauboj (engl. *cowboy*). Oba igrača mogu izabrati različite, ali i iste likove. Računalno kontrolirani neprijatelj je zmaj, odnosno zmajeva glava koja se pojavljuje i riga vatru koja oštećuje igračeve likove ako su u dodiru s njom. Igrači mogu oštetiti zmaja jedino napadajući njegovu glavu.

Svaki lik ima svoj napad koji se razlikuje od ostalih po izgledu, brzini i dometu. Vitez ima napad s najkraćim dometom, kaubojev napad ima srednji domet, dok čarobnjakov napad ima najveći domet. Svaki napad oduzima pet životnih bodova drugom igračevom liku ili zmaju ako ga dodirne. Projektili ispaljeni tijekom napada, a i sami likovi imaju svoju masu i mogu odgurnuti igračevog lika tijekom sudara, tako dajući mogućnost igračima da se međusobnu guraju na nepovoljne prepreke kao što su lava, šiljci i vatrene kugle. Igrači ne mogu odgurnuti zmaja, samo ga mogu oštetiti.

Svaki lik ima i dvije posebne sposobnosti koje nakon što ih igrač iskoristi imaju određeno vrijeme punjenja tijekom kojega se ne mogu opet iskoristiti.

Cjelokupna igra je sastavljena od tri scene. Jedna scena za glavni izbornik, jedna za izbor igračevih likova i jedna za arenu za borbu.

3.2. Upravljač

Upravljač (engl. *controller*) je objekt koji postoji u svakoj sceni igre i njemu je dodana skripta kojom se upravlja glavnim pozadinskim događajima unutar igre kao što su promjena scene, promjena pozadinske glazbe, odabir likova i prijenos tog odabira u sljedeću scenu te provjera ostvarenja cilja igre. Zbog čestog povezivanja funkcija skripte upravljača i ostalih skripti, neke bitnije funkcije će biti objašnjene u daljnjem tekstu.

```
public void ChangeScene(int scene)
    {
        SceneManager.LoadScene(scene);
        endGame = false;
        paused = false;
        done = false;
        if (scene == 2)
        {
            Time.timeScale = 1;
            Invoke("InstantiateP1", 0.1f);
            Invoke("InstantiateP2", 0.1f);
        }
        if(scene == 0 || scene == 1)
        {
            player1pick = 0;
            player2pick = 0;
        }
    }
```

Sl. 3.1. Kod funkcije ChangeScene u Controller skripti

Funkcija *ChangeScene* prikazana na slici 3.1. prima cjelobrojnu vrijednost koja predstavlja broj scene koju će potom učitati. U ovoj igri postoje tri scene označene brojevima 0, 1 i 2. Scena 0 predstavlja scenu glavnog izbornika, scena 1 predstavlja scenu odabira lika, scena 2 predstavlja scenu arene za borbu. Funkcija još provjerava koja scena je odabrana i ako se radi o sceni 2, odnosno areni za borbu, pozivaju se funkcije *InstantiateP1* i *InstantiateP2* koje stvaraju igračeve likove u toj sceni.

```
private void InstantiateP1()
    {
        if (player1pick == 0)
        {
            knight.tag = "player1";
            Instantiate(knight, new Vector3(-5.7f, -6.1f, 0),
Quaternion.identity);
        }
        else if (player1pick == 1)
        {
            mage.tag = "player1";
            Instantiate(mage, new Vector3(-5.7f, -6.1f, 0),
Quaternion.identity);
        }
        else if (player1pick == 2)
        {
            cowboy.tag = "player1";
            Instantiate(cowboy, new Vector3(-5.7f, -6.1f, 0),
Quaternion.identity);
        }
    }
```

Sl. 3.2. Kod funkcije InstantiateP1 u Controller skripti

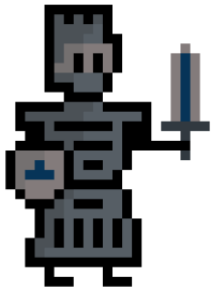
Funkcija *InstantiateP1* prikazana na slici 3.2. provjerava vrijednost varijable *player1pick*, koja ovisi o igračevom odabiru lika i ovisno o njenoj vrijednosti stvara objekt odabranog lika na sceni kojim upravlja prvi igrač. Postoji i funkcija *InstantiateP2* koja se razlikuje u tome da stvara objekt lika koji je izabrao drugi igrač i njime će i upravljati drugi igrač.

3.3. Likovi

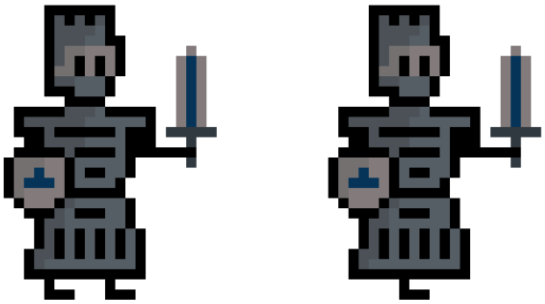
U igri postoje tri lika kojima igrači mogu upravljati i postoji računalno kontrolirani lik koji predstavlja glavnog neprijatelja. Likovi kojima igrači mogu upravljati su vitez (engl. *knight*), čarobnjak (engl. *mage*) i kauboj (engl. *cowboy*), a računalno kontrolirani neprijatelj je zmaj (engl. *dragon*). Svaki lik ima svoje *spriteove*, animacije i skriptu sa funkcijama njihovih napada i sposobnosti. Napadi i sposobnosti svih likova kojima igrači mogu upravljati imaju određeno vrijeme punjenja kada se opet mogu iskoristiti.

3.3.1. Vitez

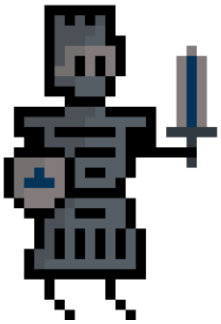
Vitez je prvi od tri lika kojim igrači mogu upravljati. Vitez ima najkraći domet napada, ali i najbolju obrambenu sposobnost od sva tri lika.



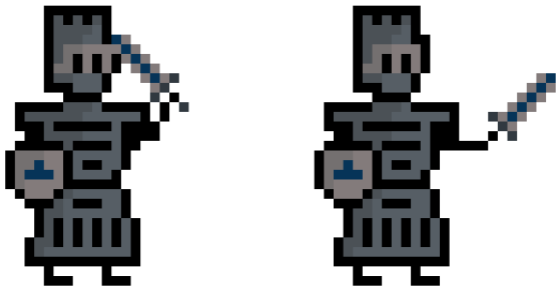
Sl. 3.3. Sprite viteza



Sl. 3.4 Spriteovi animacije hoda viteza



Sl. 3.5 Sprite animacije skoka viteza

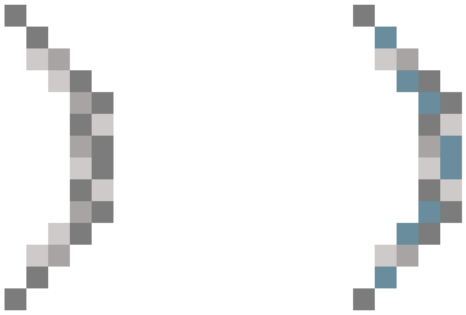


Sl. 3.6 Spriteovi animacije napada viteza

Na slici 3.3 je prikazan osnovni *sprite* viteza. Objekt viteza sadrži više animacija koje su sastavljane od nekoliko *spriteova*. Postoje animacije za hod (Sl. 3.4.), skok (Sl. 3.5.) i napad (Sl. 3.6.).

```
void SwordSlash()
    {
        this.GetComponent<AudioSource>().Play();
        animator.SetTrigger("attack");
        timeStampAttack = Time.time + attackCooldown;
        Rigidbody2D sword_slashInstance = Instantiate(sword_slash,
attack.transform.position, attack.transform.rotation) as Rigidbody2D;
        if (attack.transform.rotation == Quaternion.Euler(0, 0, 0))
            sword_slashInstance.velocity = new Vector2(speed, 0);
        else
            sword_slashInstance.velocity = new Vector2(-speed, 0);
    }
void Rush()
    {
        timeStampRush = Time.time + rushCooldown;
        gameObject.GetComponent<Player>().moveSpeed = 10;
        gameObject.GetComponent<Rigidbody2D>().mass = 10;
        Invoke("RushDown", 3f);
    }
void Shield()
    {
        timeStampShield = Time.time + shieldCooldown;
        shield.SetActive(true);
        attackEnabled = false;
        Invoke("ShieldDown", 2f);
    }
```

Sl. 3.7. Kod funkcija napada i sposobnosti viteza



Sl. 3.8. Spriteovi animacije reza mača

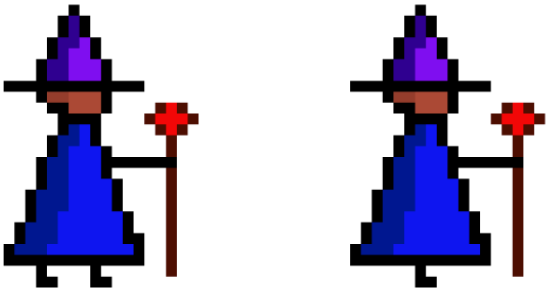
Sl. 3.9. Vitez i štit koji se pojavi kada se iskoristi Shield sposobnost

Funkcije vitezovog napada i sposobnosti prikazane su na slici 3.7. Vitezov napad aktivira se pozivanjem funkcije *SwordSlash*, koja stvara objekt, koji prikazuje rez mača (Sl. 3.8.) i daje mu određenu brzinu u smjeru prema kojem je lik okrenut. Ako rez mača zahvati lika drugog igrača ili zmaja, zahvaćeni lik gubi 5 životnih bodova. Prva vitezova sposobnost aktivira se pozivom funkcije *Rush*, koja privremeno vitezu povećava brzinu i masu na 3 sekunde. Ta sposobnost omogućava brže kretanje po areni i veći otpor na guranje, ali isto tako i veću snagu guranja drugog igrača. Druga vitezova sposobnost aktivira se pozivom funkcije *Shield*, koja privremeno aktivira objekt štita ispred lika na 2 sekunde (Sl. 3.9.). Štit blokira napade drugog igrača.

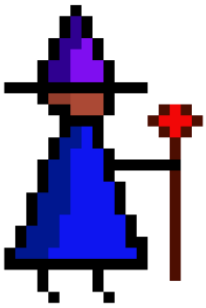
3.3.2. Čarobnjak

Čarobnjak je drugi od tri lika kojim igrači mogu upravljati. Čarobnjak ima najduži domet napada i najbolju sposobnost izmicanja opasnosti.

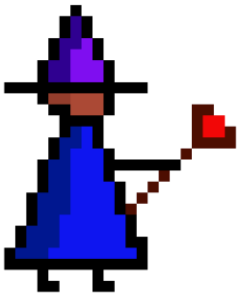
Sl. 3.10. Sprite čarobnjaka



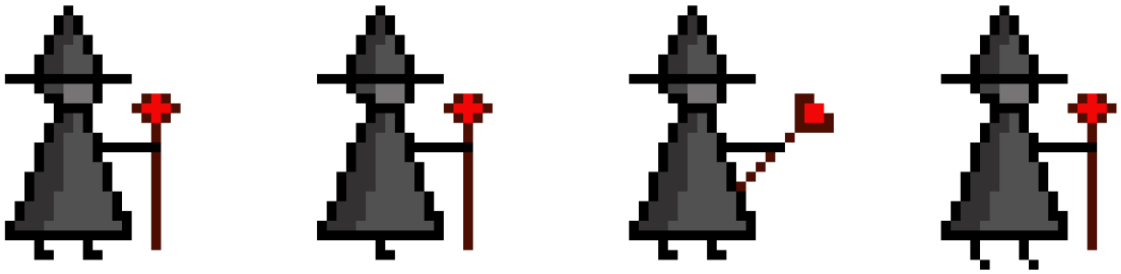
Sl. 3.11. Spriteovi animacije hoda čarobnjaka



Sl. 3.12. Sprite animacije skoka čarobnjaka



Sl. 3.13. Sprite animacije napada čarobnjaka



Sl. 3.14. Spriteovi animacija kada čarobnjak iskoristi Stoneform sposobnost

Na slici 3.10. prikazan je osnovni *sprite* čarobnjaka. Objekt čarobnjaka sadrži više animacija koje su sastavljane od nekoliko *spriteova*. Postoje animacije za hod (Sl. 3.11.), skok (Sl. 3.12.), napad (Sl. 3.13.) i *Stoneform* sposobnost (Sl. 3.14.).

```
void Fireball()
    {
        this.GetComponent<AudioSource>().Play();
        animator.SetTrigger("attack");
        timeStampAttack = Time.time + attackCooldown;
        Rigidbody2D fireballInstance = Instantiate(fireball,
attack.transform.position, attack.transform.rotation) as Rigidbody2D;
        if (attack.transform.rotation == Quaternion.Euler(0, 0, 0))
            fireballInstance.velocity = new Vector2(speed, 0);
        else
            fireballInstance.velocity = new Vector2(-speed, 0);
    }
void Teleport()
    {
        timeStampTeleport = Time.time + teleportCooldown;
        if (this.transform.rotation == Quaternion.Euler(0, 0, 0))
            this.transform.position = new Vector2(transform.position.x +
3, transform.position.y);
        else
            this.transform.position = new Vector2(transform.position.x -
3, transform.position.y);
    }
void Stoneform()
    {
        timeStampStone = Time.time + stoneCooldown;
        GetComponent<Rigidbody2D>().mass = 1000f;
        animator.SetBool("isStone", true);
        Invoke("StoneformDown", 2f);
    }
```

Sl. 3.15. Kod funkcija napada i sposobnosti čarobnjaka

Sl. 3.16. Spriteovi animacije vatrene kugle

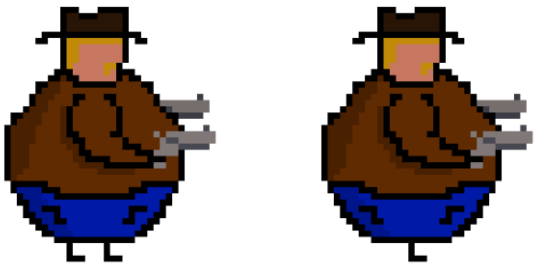
Funkcije čarobnjakovog napada i sposobnost prikazane su na slici 3.15. Čarobnjakov napad aktivira se pozivanjem funkcije *Fireball* koja stvara objekt, vatrenu kuglu (Sl. 3.16.) i daje mu određenu brzinu u smjeru prema kojem je lik okrenut. Ako vatrena kugla zahvati lika drugog igrača ili zmaja, zahvaćeni lik gubi 5 životnih bodova. Prva čarobnjakova sposobnost aktivira se pozivom funkcije *Teleport* koja premješta lika na nedaleku lokaciju u smjeru prema kojem je lik okrenut.

Druga čarobnjakova sposobnost aktivira se pozivom funkcije *Stoneform* koja privremeno daje liku kameni izgled, povećava mu masu i time ga čini otpornijim na guranje, a i daje mu puno veću snagu guranja. Ta sposobnost traje 2 sekunde.

3.3.3. Kauboj

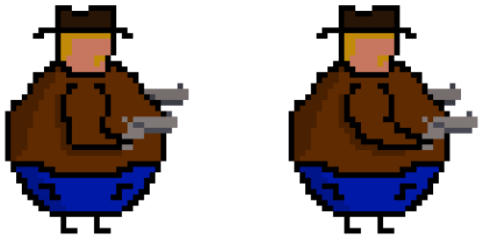
Kauboj je treći od tri lika kojim igrači mogu upravljati. Kauboj ima srednji domet napada, ali njegov napad je dvostruk.

Sl. 3.17. Sprite kauboja

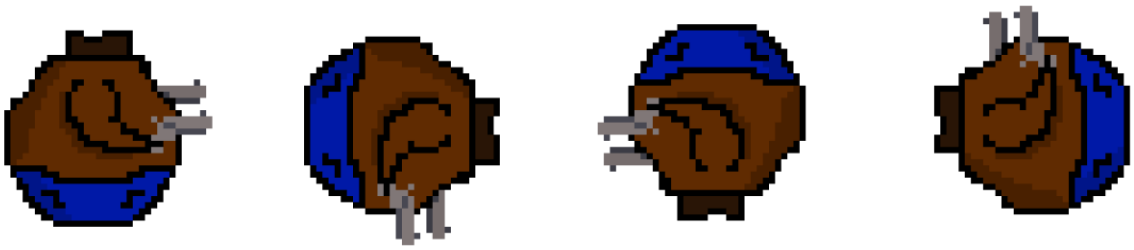


Sl. 3.18. Spriteovi animacije hoda kauboja

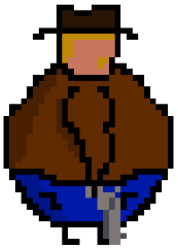
Sl. 3.19. Sprite animacije skoka kauboja



Sl. 3.20. Spriteovi animacije napada kauboja



Sl. 3.21. Spriteovi animacije Hayball sposobnosti



Sl. 3.22. Sprite animacije Jumpshot sposobnosti

Na slici 3.17. prikazan je osnovni sprite kauboja. Objekt kauboja sadrži više animacija koje su sastavljane od nekoliko *spriteova*. Postoje animacije za hod (Sl. 3.18.), skok (Sl. 3.19.), napad (Sl. 3.20.), *Hayball* (Sl. 3.21.) i *Jumpshot* (Sl. 3.22.) sposobnosti.

```
void Fire()
    {
        Rigidbody2D bulletInstance = Instantiate(bullet,
attack.transform.position, attack.transform.rotation) as Rigidbody2D;
        this.GetComponent<AudioSource>().Play();
        if (attack.transform.rotation == Quaternion.Euler(0, 0, 0))
            bulletInstance.velocity = new Vector2(speed, 0);
        else
            bulletInstance.velocity = new Vector2(-speed, 0);
    }
void Jumpshot()
    {
        this.GetComponent<AudioSource>().PlayOneShot(jumpshotSound);
        timeStampJumpshot = Time.time + jumpshotCooldown;
        animator.SetTrigger("jumpshot");
        Vector2 leftBullet = new Vector2(this.transform.position.x - 1f,
this.transform.position.y);
        Vector2 rightBullet = new Vector2(this.transform.position.x + 1f,
this.transform.position.y);
        Rigidbody2D bulletInstance = Instantiate(bullet, rightBullet,
Quaternion.identity) as Rigidbody2D;
        bulletInstance.velocity = new Vector2(speed, 0);
        Rigidbody2D bulletInstance2 = Instantiate(bullet, leftBullet,
Quaternion.identity) as Rigidbody2D;
        bulletInstance2.velocity = new Vector2(-speed, 0);
        GetComponent<Rigidbody2D>().velocity = new
Vector2(GetComponent<Rigidbody2D>().velocity.x, 10);
    }

void Hayball()

    {
        timeStampHayball = Time.time + hayballCooldown;
        animator.SetBool("hayball", true);
        canAttack = false;
        this.gameObject.GetComponent<Player>().canJump = false;
        GetComponent<Rigidbody2D>().mass = 2000f;
        Invoke("HayballDown", 3f);
    }
```

Sl. 3.23. Kod funkcija napada i sposobnosti kauboja

Sl. 3.24. Sprite metka

Funkcije kaubojevog napada i sposobnosti prikazane su na slici 3.23. Kaubojev napad aktivira se pozivom funkcije *Fire* koja stvara objekt, metak (Sl. 3.24.), i daje mu određenu brzinu u smjeru prema kojem je lik okrenut. Ako metak pogodi lika drugog igrača ili zmaja, pogođeni lik gubi 5 životnih bodova. Prva kaubojeva sposobnost aktivira se pozivom funkcije *Jumpshot*. Upotrebom te sposobnosti lik skoči vertikalno prema gore više nego običnim skokom i prije skoka ispali dva metka, jedan sa svoje lijeve strane, a drugi sa svoje desne. Druga kaubojeva sposobnost aktivira se pozivom funkcije *Hayball*. Upotrebom te sposobnosti kauboj postaje rotirajuća kugla na 3 sekunde, ima najveću masu od svih likova, ali ne može skakati, ni napadati dok ta sposobnost traje.

3.3.4. Zmaj

Zmaj je jedini računalno upravljani lik. Zmaj se pojavljuje svakih nekoliko sekundi i aktivira svoj napad, koji šteti bilo kojeg igrača kojeg zahvati.

Sl. 3.25. Sprite zmaja

Sl. 3.26. Spriteovi animacija zmajeve glave okrenuta gore i dolje

Sl. 3.27. Spriteovi animacija zmajeve glave okrenute lijevo i desno

Na slici 3.25. prikazan je osnovni *sprite* zmaja. Objekt zmaja sadrži više animacija koje su sastavljane od nekoliko *spriteova*. Postoje animacije za gore i dolje (Sl. 3.26.) te lijevo i desno okrenutu glavu (Sl. 3.27.).

```
void FireBreath()
    {
        int rand = Random.Range(0, 4);
        Vector2 left = new Vector2(this.transform.position.x-0.56f,
this.transform.position.y-0.93f);
        Vector2 right= new Vector2(this.transform.position.x+0.32f,
this.transform.position.y-0.98f);
        Vector2 up = new Vector2(this.transform.position.x+0.41f,
this.transform.position.y + 2.33f);
        Vector2 down = new Vector2(this.transform.position.x-0.43f,
this.transform.position.y - 0.5f);
        Quaternion leftQ = new Quaternion(this.transform.rotation.x, 180,
this.transform.rotation.z, this.transform.rotation.w);
        if (rand == 0)
        {
            animator.SetBool("headLeft", true);
            flame = Instantiate(flameBreath, left, leftQ);
        }
        else if(rand == 1)
        {
            animator.SetBool("headRight", true);
            flame = Instantiate(flameBreath, right, Quaternion.identity);
        }
        else if(rand ==2)
        {
            animator.SetBool("headUp", true);
            this.GetComponent<Collider2D>().offset=new Vector2(0, 3f);
            flame = Instantiate(flameBreath, up,
Quaternion.identity*Quaternion.Euler(0,0,90));
        }
        else
        {
            animator.SetBool("headDown", true);
            flame = Instantiate(flameBreath, down,
Quaternion.identity*Quaternion.Euler(0,0,-90));
        }
    }

void DragonFireball()
    {
        if(Controller.paused)
        {
            return;
        }
        Instantiate(dragonFireball, new Vector2(Random.Range(-13.05f,
3.12f), 8.66f),dragonFireball.transform.rotation);
    }
```

Sl. 3.28. Kod funkcija napada zmaja

Sl. 3.29. Spriteovi animacije vatrenog daha

Funkcije napada zmaja prikazane su na slici 3.28. Zmaj ima napad koji se aktivira pozivom funkcije *FireBreath*, koja stvara objekt, koji prikazuje vatreni dah (Sl. 3.29.). Koristeći funkciju koja stvara nasumičan broj između određenih vrijednosti, zmajev napad će biti nasumično uperen prema gore, dolje, lijevo ili desno. Drugi zmajev napad se aktivira pozivom funkcije *DragonFireball*, koja stvara objekt, koji predstavlja vatrenu kuglu, na vrhu arene i onda pada do dna pritom oštećujući igračeve likove pri doticaju. Ovaj napad ne ovisi je li zmaj trenutno aktivan ili ne, napad se ponavlja od početka do kraja igre.

3.4. Glavni izbornik

Glavni izbornik je prva scena koja se pojavljuje kada se igra pokreće. U glavnom izborniku postoji pozadina, naslov igre, tri gumba i tablica sa opisom kontrola kojim igrač upravlja likom.

Sl. 3.30. Završni izgled glavnog izbornika

Na slici 3.30. mogu se vidjeti svi nabrojani dijelovi glavnog izbornika. Gumbi i tekst su posebni objekti koji pripadaju u grafičko sučelje igre. Gumbima se može mijenjati izgled i tekst na njima te postaviti funkciju koju pozivaju kada su stisnuti.

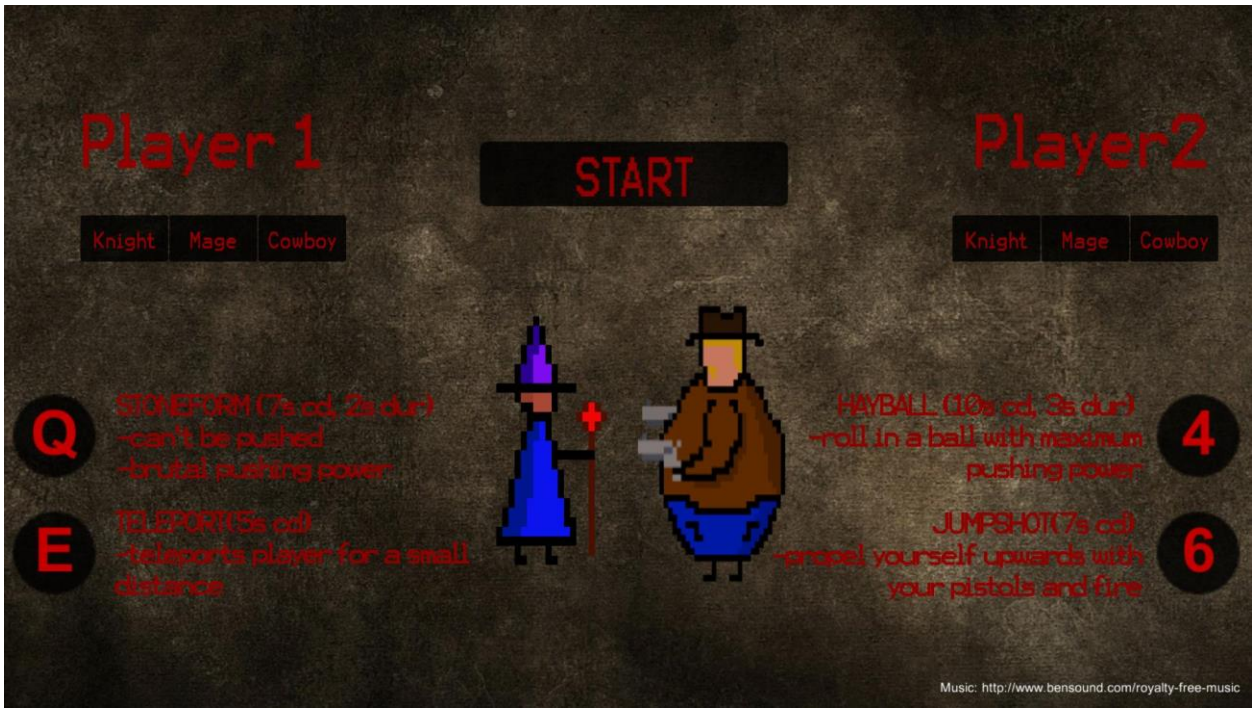
```
public void StartGame()
    {

GameObject.FindGameObjectWithTag("GameController").GetComponent<Controller
>().ChangeScene(1);
    }

public void HowToPlay()
    {
        if (howToPlayPanel.activeInHierarchy)
        {
            howToPlayPanel.SetActive(false);
        }
        else
        {
            howToPlayPanel.SetActive(true);
        }
    }

public void QuitToDesktop()
    {
        Application.Quit();
    }
```

Sl. 3.31. Kod funkcija koje pozivaju gumbi u glavnom izborniku

Pritiskom na gumb s natpisom *quit* poziva se funkcija *QuitToDesktop* i igra se zatvara. Pritiskom na gumb s natpisom *how to play* poziva se funkcija *HowToPlay* i pojavljuje se tablica s kontrolama koja se može vidjeti na slici 3.30. u donjem dijelu, desno od gumba, ponovnim pritiskom na gumb *how to play* tablica nestaje. Pritiskom na gumb s natpisom *start game* poziva se funkcija *StartGame* i učitava se scena odabira lika. Na slici 3.31. mogu se vidjeti funkcije *StartGame*, *HowToPlay* i *QuitToDesktop* koje se pozivaju pritiskom na određeni gumb. Funkcija *StartGame* radi tako da pronalazi objekt upravljača na trenutnoj sceni, poziva funkciju *ChangeScene*, koja se nalazi na objektu upravljača, predaje joj cjelobrojnu vrijednost 1 time učitavajući scenu odabira lika.

3.5. Odabir lika

Scena koja predstavlja odabir lika uvijek dolazi poslije scene glavnog izbornika ili prije scene arene za borbu. Svrha ove scene je da igrači odaberu lika s kojim žele igrati i da mogu vidjeti ukratko opisane sposobnosti svakog lika.



Sl. 3.32. Završni izgled scene odabira lika

Na slici 3.32. može se vidjeti da je scena sastavljena od dvije skupine po tri gumba za odabir lika, jednog gumba s natpisom *start*, koji pokreće sljedeću scenu, tekstualnog opisa sposobnosti lika uz prikaz tipke kojom se aktiviraju, teksta koji prikazuje koja strana predstavlja kojeg igrača i *spriteova* koji prikazuju izgled odabranog lika.

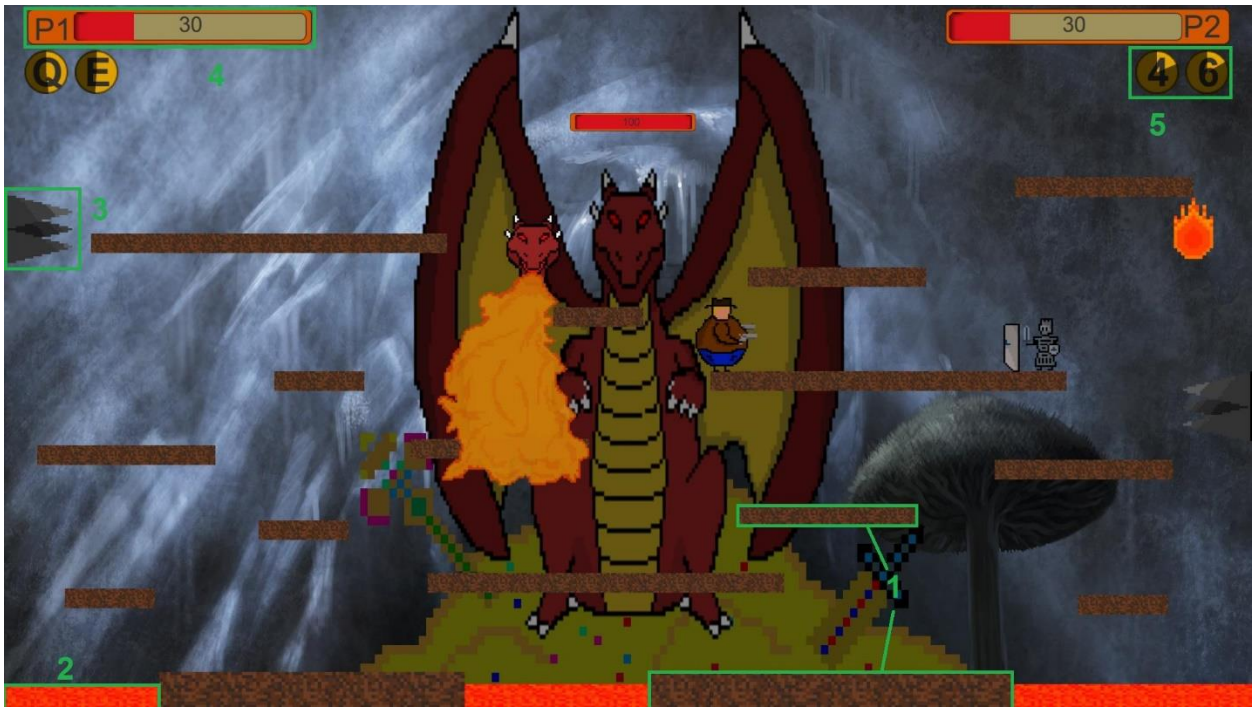
```
public void Select(int index)
    {
        if (index == selectionIndex)
            return;
        if (index < 0 || index >= models.Count)
            return;
        models[selectionIndex].SetActive(false);
        selectionIndex = index;
        models[selectionIndex].SetActive(true);
        if(index ==0)
        {
            ability1text.text="SHIELD (5s cd, 1s dur) \n-blocks damage \n-
push enemies with it";
            ability2text.text = "RUSH(8s cd, 3s dur) \n-double movement
speed \n-higher pushing power";
        }
         if(index ==1)
        {
            ability1text.text="STONEFORM (7s cd, 2s dur) \n-can't be
pushed \n-brutal pushing power";
            ability2text.text="TELEPORT(5s cd) \n-teleports player for a
small distance";
        }
        if(index ==2)
        {
            ability1text.text = "HAYBALL (10s cd, 3s dur) \n-roll in a
ball with maximum pushing power";
            ability2text.text = "JUMPSHOT(7s cd) \n-propel yourself
upwards with your pistols and fire";
        }
    }

    public void StartGame()
    {
        startGame = true;
    }
```

Sl. 3.33. Kod funkcija koje se izvode pritiskom na gumbe u sceni izbora lika

Gumbi za izbor likova sa natpisima *knight*, *mage* i *cowboy* pozivaju funkciju *Select*, prikazanu na slici 3.33., i šalju joj cjelobrojnu vrijednost. Ovisno o poslanoj vrijednosti mijenja se prikazani *sprite* i tekst sposobnosti odabranog lika. Gumb s natpisom *start* poziva funkciju *StartGame* prikazanu na slici 3.33. koja daje vrijednost *true* javnoj varijabli *startGame* koju provjerava skripta *Controller* pridružena objektu upravljača i ako je ona istinita dolazi do promjene scene na scenu arene za borbu i do stvaranja odabranih likova.

3.6. Arena za borbu

Arena za borbu je treća scena u igri u kojoj se odvija borba igrača međusobno i sa računalno kontroliranim neprijateljem. Ova scena se uvijek pokreće nakon scene odabira lika.



Sl. 3.304. Završni izgled arene za borbu

Na slici 3.34. može se vidjeti da scena sadrži pozadinske slike, platforme (1), lavu (2), šiljke (3), životne trake (4) i ikone sposobnosti (5). Pozadinske slike i platforme ne zahtijevaju nikakve kodove za svoju funkciju, ti objekti su statični. Lava i šiljci nemaju nikakve skripte dodane na sebi, ali u skripti dodanoj na igračeve likove, postoje linije koda koje aktiviraju gubitak životnih bodova igrača ako dođu u doticaj s njima.

```
void HealthBar()
    {
            if (player.tag == "player1" || player.tag == "player2")
            {
                statusText.text =
player.GetComponent<Player>().HP.ToString();
                fillAmount = player.GetComponent<Player>().HP / 100;
                content.fillAmount = fillAmount;
                if(player.GetComponent<Player>().isDead)
                {
                    statusText.text = "DEAD";
                }
            }
            else
            {
                statusText.text =
player.GetComponent<Boss>().HP.ToString();
                fillAmount = player.GetComponent<Boss>().HP / 100;
                content.fillAmount = fillAmount;
                if (player.GetComponent<Boss>().isDead)
                {
                    statusText.text = "DEAD";
                }
            }
      }

void CoolingDown()
      {
            if(coolingDown)
        {
            if (done == false)
            {
                content.fillAmount = 1.0f;
                done = true;
            }
            content.fillAmount -= 1.0f / cooldownTime * Time.deltaTime;
            if(content.fillAmount==0)
            {
                done=false;
                coolingDown=false;
            }
        }
      }
```

Sl. 3.35. Kod funkcija za životnu traku i ikone sposobnosti

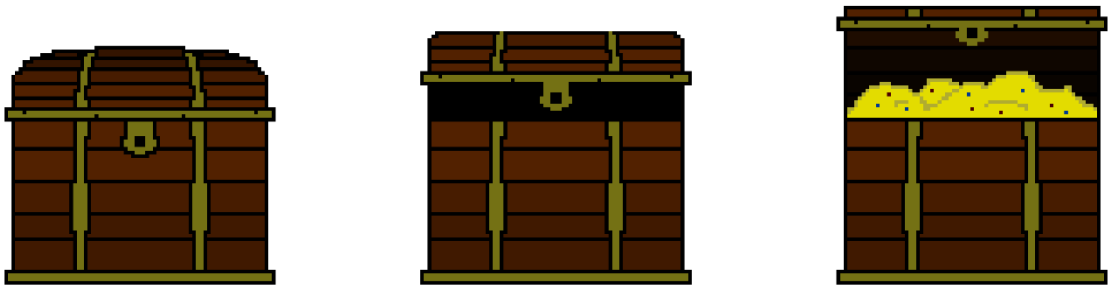
Životne trake su povezane sa igračevim likovima i sa zmajem. Pomoću funkcije *HealthBar*, prikazane na slici 3.35. provjerava se trenutno stanje životnih bodova lika i onda se prenosi u tekstualni oblik na životnoj traci, ali i životna traka se prazni ovisno o izgubljenim životnim bodovima. Ikone sposobnosti ispod životnih traka, prikazuju tipku koja aktivira sposobnost i stanje sposobnosti. Ako ikona nije zatamnjena, sposobnost je spremna za korištenje, kada se sposobnost

iskoristi ikona se zatamni i to zatamnjenje se počinje kružno prazniti ovisno o vremenu punjenja sposobnosti. To se ostvaruje funkcijom *CoolingDown* prikazanoj na slici 3.35.

Sl. 3.35. Scena arene za borbu sa aktivnim escape menijem

Sl. 3.36. Scena arene za borbu sa aktivnim end menijem

U ovoj sceni postoje još *escape* meni i *end* meni. *Escape* meni se otvara pritiskom tipke *escape* na tipkovnici i igra se pauzira, ponovnim pritiskom meni se zatvara i igra se nastavlja. *Escape* aktivira tri gumba, gumb za nastavljanje igre, gumb za odlazak na glavni izbornik i gumb za izlazak iz igre. *Escape* meni je vidljiv na slici 3.35. *End* meni se pokreće nakon završetka igre, odnosno kada su zmaj i jedan igrač pobijeđeni ili kada su oba igrača pobijeđena. *End* meni isto zadrži tri gumba, od kojih se jedino prvi razlikuje, umjesto gumba za nastavak igre, postoji gumb za povratak na scenu odabira lika. *End* meni je vidljiv na slici 3.36.



Sl. 3.37. Spriteovi animacije škrinje s blagom

Sl. 3.318. Spriteovi škrinje s blagom i nagradom ovisno o odabranom liku

Sl. 3.39. Scena arene za borbu i nagrade za pobjedu čarobnjaka

Sl. 3.40. Scena arene za borbu i nagrade za pobjedu kauboja

Sl. 3.41. Scena arene za borbu i nagrade za pobjedu viteza

Kada se ispuni uvjet kraja igre, odnosno kada samo jedan od igrača ili neprijatelj ostane nepobijeđen, igra se zaustavlja, pojavljuje se poruka u sredini scene koja informira tko je pobjednik. Ako jedan od igrača pobjedi, pojavljuje se škrinja s blagom (Sl. 3.37.) koja se otvara i prikazuje jedno od oružja ovisno o izabranom liku igrača koji je pobijedio (Sl. 3.38.). Slike 3.39., 3.40. i 3.41. pokazuju završnu scenu kada jedan od igrača pobjedi.

4. ZAKLJUČAK

U ovom radu izrađena je računalna igra za dva igrača koristeći C# programski jezik. Igra je rađena u okruženju *game enginea* Unity, a korišteni *spriteovi* za 2D objekte igre su napravljeni korištenjem Piskel web aplikacije. Kodovi pisani u C# se nalaze u skriptama koje se dodaju na objekte unutar scena u Unity uređivaču i tako omogućavaju ostvarivanje zamišljene funkcionalnosti tih objekata. Ideja igre je međusobna borba igrača uz dodatak računalno kontroliranog neprijatelja koji se pojavljuje u nasumičnim vremenskim periodima. Igra je napravljena tako da se lagano može nadograđivati bez mijenjanja kodova, objekata ili njihovih atributa. Uvijek je moguće dodati nove likove, one kojima igrači mogu upravljati, a i računalno kontrolirane. Moguće je i promijeniti ili dodati nove posebne sposobnosti i napade likova. Isto tako postoji mogućnost dodavanja novih scena arene za borbu sa drugačijim preprekama i platformama.

LITERATURA

[1] Piskel github readme

https://github.com/juliandescottes/piskel, 23.6.2016.

[2] Introduction to C# language and .NET framework

https://msdn.microsoft.com/en-us/library/z1zx9t92.aspx, 23.6.2016.

[3] Unity-Fast Facts

https://unity3d.com/public-relations, 23.6.2016.

[4] Unity Editor

https://unity3d.com/unity/editor, 23.6.2016.

[5] Unity Documentation – Sprite

https://docs.unity3d.com/ScriptReference/Sprite.html, 24.6.2016.

SAŽETAK

U ovom završnom radu pokazuje se izrada 2D računalne igre za dva igrača, korištenjem C# programskog jezika. Srž igre je međusobna borba dva igrača i zajedničkog neprijatelja, koji je računalno kontroliran. Igra je rađena u Unity *game engine* okruženju, a kodovi se pišu unutar skripti, koje se dodaju objektima u Unity uređivaču i time im daju funkcionalnost. 2D grafika, koja se koristi u igri je napravljena u Piskelu. U radu je detaljnije objašnjen proces izrade glavnih elemenata igre. Jedan od tih elemenata je upravljač koji osigurava izvođenje bitnih funkcija nevidljive igračima. Bitni elementi su i likovi sa svojim napadima i posebnim sposobnostima i tri scene koje čine ovu igru. Gotova igra pokazuje da je upotrebom Unity *game enginea* i C# programskog jezika moguće efikasno izraditi 2D igru za dva igrača koja se i kasnije može nadograđivati.

Ključne riječi: 2D računalna igra, C#, Unity, skripta, objekt, scena, *game engine*, funkcija

ABSTRACT

Using C# programming language in the making of two player computer game

This final paper shows the making of a 2D computer game for two players, using C# programming language. Core of the game is a mutual combat between two players and a common enemy, which is computer controlled. The game is made in Unity game engine environment and codes are written inside the scripts which, are added to objects in Unity editor, giving them functionality. 2D graphics, which is used in the game, is made in Piskel. The paper detaily explains the process of making the main elements of the game. One of these elements is the controller which ensures execution of important functions that are invisible to players. Characters, with their attacks and special abilities and scenes that make the game are also important elements. In the game there are three playable characters and one computer controlled character. Completed game shows that with the use of Unity game engine and C# programming language it is possible to efficiently make a 2D game for two players that can later be upgraded.

Keywords: 2D computer game, C#, Unity, script, object, scene, game engine, function

ŽIVOTOPIS

Mateo Bareš je rođen 4.8.1994 u Osijeku. Od 2001. do 2009. pohađa OŠ Grigor Vitez. Nakon toga upisuje Isusovačku klasičnu gimnaziju u Osijeku koju završava 2013. godine. Iste godine redovno upisuje preddiplomski studij računarstva na Elektrotehničkom fakultetu u Osijeku, gdje je i sada student 3. godine.