Informatika

Diplomska naloga

Igra “Cryptic Dungeons”

Program: Informatika

Šolsko leto: 2020/21

Mentor: Aleksandar Lazarević

Lokacija: Ljubljana

Maks Jurkas

Vpisna številka: 1S00190598

Ljubljana, 24.5.2021

**Kazalo**

[**1 POVZETEK** 5](#_Toc72777222)

[**2 UVOD** 6](#_Toc72777223)

[**2.1 Opredelitev teme in razlogi za izbiro** 6](#_Toc72777224)

[**2.2 Predstavitev problemov** 7](#_Toc72777225)

[**2.3 Cilji projekta** 8](#_Toc72777226)

[**2.4 Uporabljena sredstva** 8](#_Toc72777227)

[**2.5 Reševanje glavnih problemov** 9](#_Toc72777228)

[**2.5.1 Prenašanje podatkov** 9](#_Toc72777229)

[**2.5.2 Optimizacija** 9](#_Toc72777230)

[**2.5.3 Shranjevanje in nalaganje igre** 10](#_Toc72777231)

[**3 POTEK PROGRAMA** 11](#_Toc72777232)

[**3.1 Začetni zaslon** 11](#_Toc72777233)

[**3.2 Domači zaslon** 12](#_Toc72777234)

[**3.3 Igralčevi podatki** 14](#_Toc72777235)

[**3.4 Zemljevid** 17](#_Toc72777236)

[**3.5 Glavni spopad** 19](#_Toc72777237)

[**3.6 Neskončni spopad** 22](#_Toc72777238)

[**3.7 Končni nasprotnik** 25](#_Toc72777239)

[**3.8 Rudarjenje** 26](#_Toc72777240)

[**3.9 Lov** 27](#_Toc72777241)

[**3.10 Trgovina** 28](#_Toc72777242)

[**3.11 Kovač** 30](#_Toc72777243)

[**4 ZAKLJUČEK** 32](#_Toc72777244)

[**5 PRILOGE** 33](#_Toc72777245)

[**5.1 Nasprotniki** 33](#_Toc72777246)

[**5.2 Oklepi** 34](#_Toc72777247)

[**5.3 Orožja** 35](#_Toc72777248)

**Kazalo slik**

[Slika 1: Začetni zaslon 10](#_Toc72777191)

[Slika 2: Domači zaslon 11](#_Toc72777192)

[Slika 3: Zemljevid 16](#_Toc72777193)

[Slika 4: Glavni spopad - izbira 18](#_Toc72777194)

[Slika 5: Glavni spopad - bitka 19](#_Toc72777195)

[Slika 6: Neskončni spopad 21](#_Toc72777196)

[Slika 7: Končni nasprotnik 24](#_Toc72777197)

[Slika 8: Rudarjenje 25](#_Toc72777198)

[Slika 9: Lov 26](#_Toc72777199)

[Slika 10: Prikaz trgovine 27](#_Toc72777200)

[Slika 11: Prikaz kovača 29](#_Toc72777201)

[Slika 12: Nasprotniki 2. Nivoja 32](#_Toc72777202)

[Slika 13: Nasprotniki 3. Nivoja 32](#_Toc72777203)

[Slika 14: Končni nasprotnik 32](#_Toc72777204)

[Slika 15: Železni oklep, Bronasti oklep, "Pickle Armor" 33](#_Toc72777205)

[Slika 16: Goblin oklep, Opičji oklep, Dinozaver oklep 33](#_Toc72777206)

[Slika 17: Lesen meč, Železen meč, "Pickle Sword" 34](#_Toc72777207)

[Slika 18: Goblin meč, Opičji meč, Dinozaver meč 34](#_Toc72777208)

# **1 POVZETEK**

Izdelava RPG (Role-Playing Game) igre z uporabo programskega jezika C# in Microsoft-ovega framework-a Windows Forms, ter Newtonsoft JSON framework-a za shranjevanje.

V šolskem letu 2020/21 smo pri predmetu Programiranje 2 delali s programskim jezikom C# in sicer vizualno programiranje preko Windows Forms-ov. Pri odločanju za seminarsko nalogo sem se odločil izdelati nekaj svojega, in sicer igro, kjer se spopadaš z raznimi nasprotniki, za to, da lahko ustvariš njihov oklep in orožje, ter postaneš močnejši. Igralec začne z začetno opremo, kjer se mora spopadati z močnejšimi nasprotniki, da pridobi kovance in točke izkušenj in si s tem izboljša možnosti zmage nad končnim nasprotnikom prvega nivoja. Po prvi zmagi nad njim, igralec pridobi material, katerega lahko po ponovnih spopadih s končnim nasprotnikom uporabi za izdelavo njegovega orožja in oklepa.

Igralec se nato počasi prebija skozi nivoje, nadgrajuje svojo opremo in zvišuje svojo moč in obrambo, dokler ne pride do zadnjega nivoja, kjer se lahko spopade z končnim nasprotnikom igre in odklene način igre, kjer se prikažejo vsi nasprotniki iz prejšnih nivojev. Ko izdela še zadnji meč in oklep, se igralcu odklene »Hard Mode« oz. težki način, kjer postanejo nasprotniki močnejši in igralcu dajo nove materiale, katere mora uporabiti za izdelavo končnega orožja in oklepa.

S tem projektom sem se odločil, da bom nadaljeval razvoj iger in poskušal narediti nekaj, kar bi pritegnilo pozornost publike in mi dalo veselje izdelave, ter nadaljnjih projektov.

Moram priznati da je bil projekt bolj uspešen v določenih področjih in rahlo nezadovoljiv v drugih, ampak bi ga z veseljem ponovil.

# **2 UVOD**

## **2.1 Opredelitev teme in razlogi za izbiro**

Odločil sem se za izdelavo “Role-Playing” igre, ki bo temeljila na igralčevi opremi in delovala na princip “boljša oprema = močnejši igralec”.

Igro sem izdelal s pomočjo programa Visual Basic, v C# programskem jeziku, s pomočjo Windows Form-ov. Visual Basic je eden izmed najbolj priljubljenih progamov za programiranje (vsaj za Microsoftove programske jezike. C# je relativno popularen programski jezik, katerega se uporablja za več popolnoma različnih stvari, z močno podporo razvijanju iger za Windows okolja, poleg tega se je pa v zadnjih letih nabrala podpora za razvijanje iger za pametne telefone. Windows Form-i so v trenutnem stanju že precej stari in se jih uporablja vse manj, ampak so kljub temu še zmeraj precej uporabni in priljubljeni, saj so dokaj preprosti za naučiti.

Za izdelavo igre sem se odločil, ker sem se že mnogo let želel spraviti k izdelavi in razvoju iger, vendar nisem vedel kje začeti. Nato smo pri predmetu Programiranje 2 delali z Windows Form-i in sem si zamislil neko igro, ki bi jo lahko naredil za seminarsko nalogo, po pogovoru s profesorjem sem se pa odločil to igro razširiti in iz nje izdelati diplomsko nalogo.

Že dolga leta sem oboževalec RPG (Role-Playing Game) žanre, kot so na primer: World of Warcraft, Monster Hunter, Slay The Spire, in podobne, ampak glede na to, da nimam nobenih izkušenj iz razivanja iger, sem se odločil za nekaj bolj preprostega, kjer bi iz tistih iger črpal le osnovne lastnosti (izdelovanje opreme, z materiali, ki jih igralec dobi od nasprotnikov, nakupovanje opreme, nadgradnja opreme), brez 3D grafike in sveta, kjer bi se igralec premikal in spopadal z nasprotniki.

Zelo me navdušujejo igre, kjer se igralec premika po raznih različnih nivojih, z različnimi nasprotniki, vse za to, da bi si lahko nadgradil svojo opremo in šel proti težjim in težjim nasprotnikom.

## **2.2 Predstavitev problemov**

Problemi, na katere sem naletel pri pisanju programa, so bili rahlo različni od problemov, ki sem si jih zastavil pred začetkom programiranja.

Pričakoval sem, da bodo problemi pri uporabljanju različnih lastnosti opreme skozi različne nivoje igre in pri shranjevanju podatkov pridobljenih po spopadu z nasprotnikom.

Predvideval sem, da bom naletel na številne probleme glede optimiziranja napisane kode, saj je bilo moje prvotno razmišljanje o uporabi opreme popolnoma različno in neoptimizirano glede na to kar sem kasneje ugotovil in kako sem stvari na koncu zapisal.

Poleg ostalih problemov, sem bil zaskrbljen s tem, kako bom shranjeval podatke, katere bi se kasneje dalo klicati tudi iz drugega sistema. Rabil sem se odločiti med serializacijo podatkov in shranjevanjem podatkov v podatkovno bazo (tu bi se moral odločiti še za ustrezno bazo podatkov, npr. MySQL, SQL, SQLite, itd.).

Manjši problemi, ki sem jih pričakoval so bili uporaba posebnih efektov od določenih nasprotnikov in določene igralčeve opreme, prikaz likov (igralec, orožje, oklep, nasprotniki, ozadja), odklepanje in zaklepanje prisvojene opreme in izvajanje spopada med igralcem in nasprotnikom.

## **2.3 Cilji projekta**

Na začetku projekta sem si zastavil cilj, da ustvarim neko aplikacijo oz. igro, ki bi imela nekatere principe svetovno znanih RPG (Role-Playing Game) iger.

Želel sem uporabiti številne lastnosti iz raznih iger, ki sem jih igral in jih združiti v eno igro, brez 3D grafik, premikanja po svetu, animacij, itd.

Prvotno sem nameraval implementirati pridobivanje opreme na način igre World of Warcraft, kjer bi igralec pridobil opremo, ko bi premagal končnega nasprotnika nivoja, ampak sem se kasneje odločil za drug pristop k pridobivanju opreme, in sicer, na metodo igre Monster Hunter, kjer igralec po zmagi nad končnim nasprotnikom nivoja pridobi materiale za izdelavo orožja in oklepa od njega.

Premikanje skozi nivo, premagovanje nasprotnikov in premikanje do končnega nasprotnika sem si zamislil na način igre Slay the Spire, kjer ima igralec več možnosti pristopa k končnemu nasprotniku (bojevanje, preskakovanje boja…), s tem, da ima vsaka možnost svoje prednosti in slabosti.

Za like sem se odločil, da bodo 8-bitne sličice, katere bi sam izdelal.

## **2.4 Uporabljena sredstva**

Za izdelovanje likov sem uporabil program Adobe Illustrator, kjer sem s pomočjo mreže izdelal vse like, orožja, oklepe, gumbe in ozadja.

Za pisanje programa sem uporabil program Visual Basic in programski jezik C#.

V Visual Basic-u sem uporabil Windows Form-e za prikaz programa in Newtonsoft JSON framework za serializacijo oz. shranjevanje ter nalaganje igre.

## **2.5 Reševanje glavnih problemov**

### **2.5.1 Prenašanje podatkov**

Problemi pri tej nalogi so bili številni, največji problem je bil, ugotoviti kako prenašati podatke o igralcu in nasprotnikih iz ene Windows Form-e v drugo, kajti se nova forma odpre s klicem »New«, ki pa kreira novo instanco Windows Form-a, kar pomeni, da so vsi podatki ponastavljeni, ta problem sem odrešil s pomočjo kazalcev oz. referenc, s tem da sem vse igralčeve podatke shranil v posebej dokument, iz katerega sem jih nato klical, poleg tega sem pa iz ene forme v drugo pošiljal določene podatke (npr. izbran oklep in orožje) in sem tako odpravil problem shranjevanja oz.   
ne-shranjevanja podatkov, tako sem lahko po parih dneh ugotavljanja končno nadaljeval izdelavo svojega programa.

### **2.5.2 Optimizacija**

Ključni problem pri večini programov je optimizacija, z drugimi besedami, kako bi program zapisal s čim manjšim številom vrstic in ponavljanja, da bi delal hitreje in bolje kot prej.

Pri optimizaciji je bila uporaba kazalcev ključna, saj bi bil moj progam dolg več tisoč vrstic kode brez njih. Tako, pa je ponavljanje kode minimalno in uporabljeno le ko je absoluto potrebno.

Kazalci delujejo tako, da v eni datoteki oz. eni formi kličemo podatke iz druge datoteke. V mojem primeru so bili vsi podatki shranjeni v eni datoteki (Data.cs) in vse forme črpajo podatke iz te datoteke.

V objektu oz. class-u Podatki so shranjeni vsi podatki o igralcu, poleg tega pa so tu shranjeni tudi vsi podatki o orožjih, oklepih in zalogi materialov, ki jih je igralec pridobil pri igranju.

### **2.5.3 Shranjevanje in nalaganje igre**

Poleg problema prenašanja podatkov, medtem ko igralec igra, je bil eden izmed večjih problemov še shranjevanje igre, tako da bi lahko igralec nadaljeval svojo prejšno sejo igre in se mu ne bi vsi podatki in pridobljena oprema zbrisala ob zaustavitvi programa.

Tu sem se potreboval odločiti za način shranjevanja in nalaganja igre, imel sem izbiro med serializacijo (shranjevanje objektov v neko datoteko) in uporabo podatkovne baze (MySQL, SQL, SQLite).

Najprej sem se odločil za uporabo podatkovne baze in sem se odloči za MySQL, saj sem bil s to bazo že seznanjen. Po številnih urah ugotavljanja, kako bi svoj progam povezal z bazo podatkov in iz baze bral oz. spreminjal vrednosti, se mi je celoten program obrnil na glavo in sem porabil par ur, da sem vse nazaj zgradil v stanje pred izgubo podatkov. Naslednji dan sem ponovno poskusil vzpostaviti povezavo in uporabo baze, tokrat mi je uspelo, ampak se je pojavil nov problem; veliko kode in malo optimizacije, kar bi pomenilo, da bi bil program veliko bolj počasen, z dodajanjem elementov in objektov v igro.

Po tem, ko sem uspešno vzpostavil povezavo s podatkovno bazo, sem to idejo tudi zavrgel in se odločil za uporabo serializacije.

Serializacija je bila dokaj enostavna, saj je tehnologija pogosto uporabljena in dobro dokumentirana, kar je zelo poenostavilo problem shranjevanja.

Odločil sem se za Newtonsoft JSON serializacijo, kar shranjuje objekt Podatki v dokument z .json končnico.

Newtonsoft JSON je že precej razširjen in uporabljen »framework«, z leti razvijanja in optimiziranja in z dobro opisano dokumentacijo je njegova uporaba relativno enostavna.

# **3 POTEK PROGRAMA**

## **3.1 Začetni zaslon**

Ob zagonu programa se nam prikaže začetni zaslon, kjer lahko uporabnik izbira med »Start«, »Load Game« in »Exit«.



Slika 1: Začetni zaslon

Če se uporabnik odloči začeti novo igro pritisne na »Start«, kar bo programu povedalo, da se uporabijo privzeti podatki in nastavitve.

Če je uporabnik igro zagnal že kadarkoli prej in je shranil svoj napredek, lahko izbere opcijo »Load Game«, nakar se odpre pogovorno okno, kjer uporabnik poišče kreirano datoteko s končnico .json.

Ko uporabnik izbere željeno datoteko, se mu zažene igra z vsemi predhodno pridobljenimi materiali in napredki v igri.

Tretja možnost na začetnem zaslonu je še »Exit«, ki aplikacijo oz. program zapre.

Home h = new Home(weapon, armor, ref p);

h.Width = this.Width;

h.Height = this.Height;

h.StartPosition = FormStartPosition.Manual;

h.Location = new Point(this.Location.X, this.Location.Y);

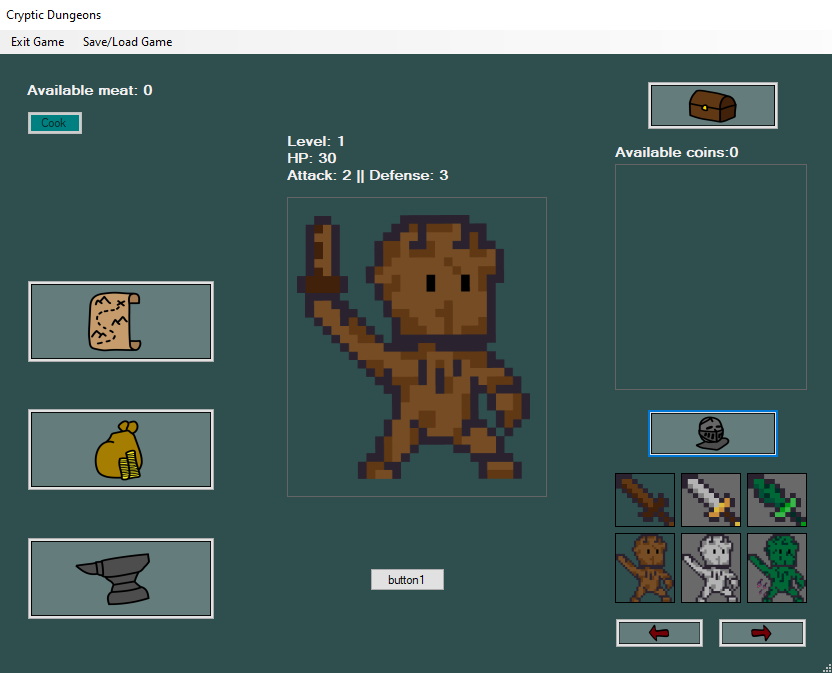
this.Visible = false;

h.ShowDialog();

Zgornja programska koda prikazuje odpiranje novega okna, tak zapis se ponavlja skozi celoten program, kadar se odpira novo okno (prehod iz enega okna v drugega).

Tak zapis je namenjen temu, da se novo okno odpre na mestu trenutnega okna, z isto velikostjo.

## **3.2 Domači zaslon**



Slika 2: Domači zaslon

Na domačem zaslonu na sredini vidimo igralca opremljenega s trenutno izbrano opremo, nad njim so podatki o igralčevih atributih (nivo igralca, življenjske točke, moč in obramba igralca).

Na levi strani vidimo gumbe za zemljevid, trgovino in kovača.

Na desni strani je prikazan še vsebina igralčeve torbe in kosi opreme, ki imajo sivo ozadje dokler jih igralec ne pridobi.

Na vrhu okna vidimo »Exit Game« in »Save/Load Game« kar se uporablja za izhod iz igre in shranjevanje oz. nalaganje prejšnje seje igre.

Zanimivost pri kodi za to okno leži pri prikazovanju in skrivanju igralčeve opreme.

public void hideAvailableEquipment()

{

foreach (Control x in this.Controls)

{

if (x is Button && x.Visible == true)

{

if ((string)x.Tag == "Equipment" || (string)x.Tag == "Equipment1" ||

(string)x.Tag == "Equipment2")

{

x.Visible = false;

}

}

}

}

Ko uporabnik pritisne na gumb za opremo (železna čelada na desni strani zaslona) se izvede zgornja zanka, pri kateri gre program skozi vse kontrolne komponente programa in preveri če je komponenta gumb, ter če je gumb trenutno skrit.

Nato pregleda še če ima gumb značko »Equipment«, ta značka je podana vsem komponentam, ki se uporabijo za prikaz opreme, ter ga skrije oziroma prikaže, če trenutno ni viden.

Domači zaslon je tudi edini zaslon iz katerega lahko uporabnik shranjuje, nalaga in zapira igro.

* Shranjevanje igre:

p.saveName = Path.GetFullPath(saveFileDialog1.FileName);

File.WriteAllText(p.saveName, JsonConvert.SerializeObject(p));

Kot sem omenil že prej, sem za shranjevanje uporabil JSON serializacijo, kar pomeni, da se vsi željeni podatki shranijo v datoteko s končnico .json.

* Nalaganje igre:

p.saveName = Path.GetFullPath(openFileDialog1.FileName);

JsonConvert.PopulateObject(File.ReadAllText(p.saveName), this.p);

Prvi stavek je podoben stavku za shranjevanje igre, njegova funkcionalnost je shranjevanje poti datoteke, da lahko kasneje uporabnik shranjuje v isto datoteko brez odpiranja dodatnega pogovornega okna.

Drugi stavek pretvori .json datoteko v podatke, ki spodrinejo trenutne uporabljene podatke.

## **3.3 Igralčevi podatki**

Vse podatke, ki se uporabljajo skozi igro sem zapisal v ločeno datoteko, iz katere jih program nato vleče in kamor jih tudi zapisuje, ko se spremenijo.

Razred sem poimenoval »Podatki« in vsaka forma oz. datoteka ima na začetku zapisan kazalec, ki programu pove od koder naj črpa podatke.

Inicializacija forme:

public Home(Weapons selectedWeapon, Armor selectedArmor, ref Podatki p)

{

InitializeComponent();

this.p = p;

* »Home« je domači zaslon, v oklepaju so pa zapisani podatki, ki se prenašajo v to formo iz prejšnjega odprtega okna.
* Zadnji del oklepaja (ref Podatki p) je kazalec na igralčeve podatke, zapisane v drugi datoteki.
* »InitializeComponent();« nakazuje zagon trenutnega okna in vseh njegovih komponent.
* »this.p = p;« pa pove programu nakaže spremenljivko, ki se uporablja za črpanje podatkov

Vsi podatki o orožjih, oklepih, nasprotnikih so zasebni, da jih igralec ne mora spreminjati in si prilagajati igre po svojih željah.

public class WoodenSword : Weapons

{

public WoodenSword()

{

weaponName = "Wooden Sword";

weaponDmgMax = 3;

weaponDmgMin = 2;

isUnlocked = true;

weaponImage = Properties.Resources.WoodenSword\_Equip;

}

}

Zgornji zapis kode prikazuje javne lastnosti, medtem ko so podatki zasebni.

V taki obliki so zapisani vsi objekti v programu, torej, javni razred objekta, ki ima podane podatke iz razreda, kjer so podatki zasebni.

Vsak objekt ima tu zapisane vse lastnosti, poleg tega pa še sliko objekta.

Spremenljivka »isUnlocked« se uporablja za prikaz opreme, ki jo je uporabnik pridobil (če je vrednost spremenljivke »true« pomeni, da je igralec kos opreme pridobil in mu je odklenjen).

Zapis orožij v objekt vseh orožij:

public class Weapon

{

public WoodenSword woodenS = new WoodenSword();

public IronSword ironS = new IronSword();

public PickleSword pickleS = new PickleSword();

public GoblinSword goblinS = new GoblinSword();

public MonkeMallet monkeM = new MonkeMallet();

public DinoSword dinoS = new DinoSword();

}

Po prvotnem zapisu željenega dela opreme v objekt, ga moramo za tem še kreirati in vstaviti v polje vseh orožij.

Zapis vseh orožij v polje, katerega program nato uporablja za klic željenega orožja:

public Weapon weapons = new Weapon();

Vsi materiali ustvarjeni na enak način, zapis v polje za klic je pa drugačno, ker je njihova uporabnost drugačna.

Materiali so zapisani v razredu »Inventory«, kjer so shranjene tudi funkcije za pridobitev in uporabo materialov, poleg tega pa še funkciji za prikaz vseh igralčevih materialov ter zahtevano število materialov za izdelavo opreme.

public Item[] items = { new PickleJuice(), new PickleSkin(), new GoblinHide(), new GoblinSkull(), new MonkeBones(), new MonkeCarapace(),

new DinoTeeth(), new DinoClaws(), new DinoTail(),

new IronOre(), new MythrilOre(), new EnhancementStone()};

Zapis materialov je narejen tako, da se vse zapiše v neko polje, kjer ima vsak posamezni material svojo identifikacijsko številko, ki se v programu uporablja za pridobitev in porabo materiala.

public void incrementQuantity(int i)

{

itemQuantity += i;

}

public void decrementQuantity(int i)

{

itemQuantity -= i;

}

V razredu za predmete sta zapisani zgornji funkciji, prva se uporablja za dodajanje, druga pa za odvzemanje predmetov iz igralčeve torbe.

Na koncu imamo še razred z recepti za orožja in oklepe, in sicer, vsak recept ima zapisano potrebno količino materialov za izdelavo.

V razredu za recepte se nahajajo še funkcije za preverjanje, če ima igralec dovolj materialov za izdelavo opreme, izdelavo želene opreme in pa odklep izdelanega kosa opreme.

Podatki, ki niso potrebni za shranjevanje imajo »[JsonIgnore]« pred njimi, kar programu pove naj preskoči dan podatek, ko izdeluje kopijo trenutnega stanja igre.

public class PickleJuice : Item

{

readonly public static int ID = 0;

public PickleJuice()

{

itemName = "Pickle Juice";

itemID = ID;

itemQuantity = 0;

}

}

Tako so naslovljeni materiali, prvi podatek je identifikacijska številka, ki se uporablja za klic teh materialov, nato je spodaj nastavljen še itemID na to številko, saj je zgornji podatek le za branje.

public int coins = 0;

public int points = 0;

public int str = 0;

public int def = 0;

public int playerMaxHP = 30;;

public double exp = 0;

public int maxExp = 10;

public int level = 1;

public int newLevel = 2;

public bool didPlayerEat = false;

Zgoraj so navedeni igralčevi podatki, in sicer:

* Kovanci
* Točke (za nadgrajevanje moči in obrambe)
* Moč
* Obramba
* Igralčeve življenjske točke
* Točke izkušenj
* Prag za točke izkušenj
* Igralčev nivo
* Igralčev nov nivo (uporabljeno za dodajanje novega nivoja)
* Ali je igralec pojedel (pred misijo)

## **3.4 Zemljevid**



Slika 3: Zemljevid

S pritiskom na gumb z ikono zemljevida na domačemu zaslonu nas program popelje do zemljevidnega okna, kjer lahko igralec izbira kam bo šel.

Ko začnemo novo igro, imamo odklenjen le prvi nivo »Seabed«, kjer se igralec spopada z morskimi nasprotniki, poleg tega ima pa na voljo še »LowRank Grind«, kjer hitreje pridobiva točke izkušenj in kovance, ter postane močnejši, da gre lahko probati premagati končnega nasprotnika prvega nivoja.

Po izdelavi orožja in oklepa iz prvega nivoja (Pickle Sword in Pickle Armor) se uporabniku odklene naslednji nivo »Forest« ter neskončni spopad drugega nivoja, »MidRank Grind«.

Igra se nadaljuje iz nivoja v nivo, pri katerem je vsaka naslednja stopnja težja od prejšnje.

Poleg stopnjevanja težavnosti se s odklepom naslednjega nivoja igralcu onemogoči pridobivanje kamnov za nadgrajevanje opreme v prejšnjih nivojih.

Ko igralec premaga vse 3 nivoje in izdela vso opremo iz vseh nivojev, se mu odklene arena »Boss«, kamor se gre uporabnik nato spopasti s končnim nasprotnikom igre.

Če mu ga uspe premagati, se mu odklene še zadnji nivo »Endless«, v katerem se pred igralcem prikažejo vsi nasprotniki, ki nastopajo v igri.

Poleg prvega nivoja ima igralec odklenjen tudi »Mining« oz. rudarjenje in »Hunting«, kjer gre igralec na lov za hrano.

* Na vrhu sedi gumb za povračilo na prejšnje okno.
* Pod njim so nivoji, od zgoraj navzdol so: misija (glavni spopad), neskončni spopad nivoja in neskončni spopad vseh nasprotnikov, na desni strani je končni nasprotnik igre.
* Na dnu sta pa še gumba za lov in rudarjenje.

Pri rudarjenju igralec pridobi rudo, ki jo potrebuje za izdelavo močnejše opreme.

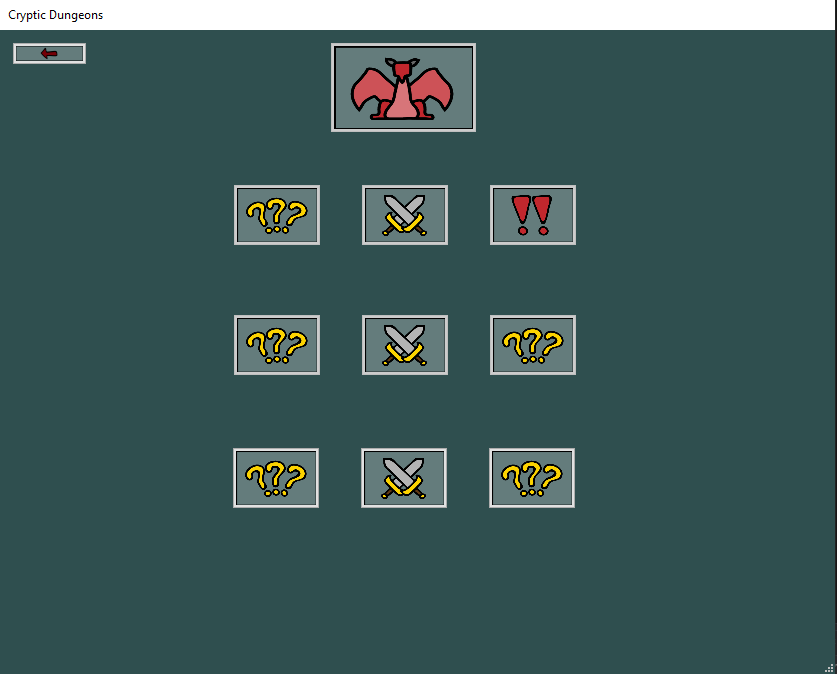
Pri lovu igralec pridobi meso, ki mu zviša napad za izbrano misijo.

Koda za pošiljanje podatkov v drugo formo je tu drugačna, in sicer:

STS s = new STS(selectedWeapon, selectedArmor, boss, mob1, mob2, mob3, specialMob, ref p);

Poleg kazalca na igralčeve podatke, se iz tega okna pošljejo še vsi nasprotniki, ki nastopajo na določenem nivoju.

## **3.5 Glavni spopad**



Slika 4: Glavni spopad - izbira

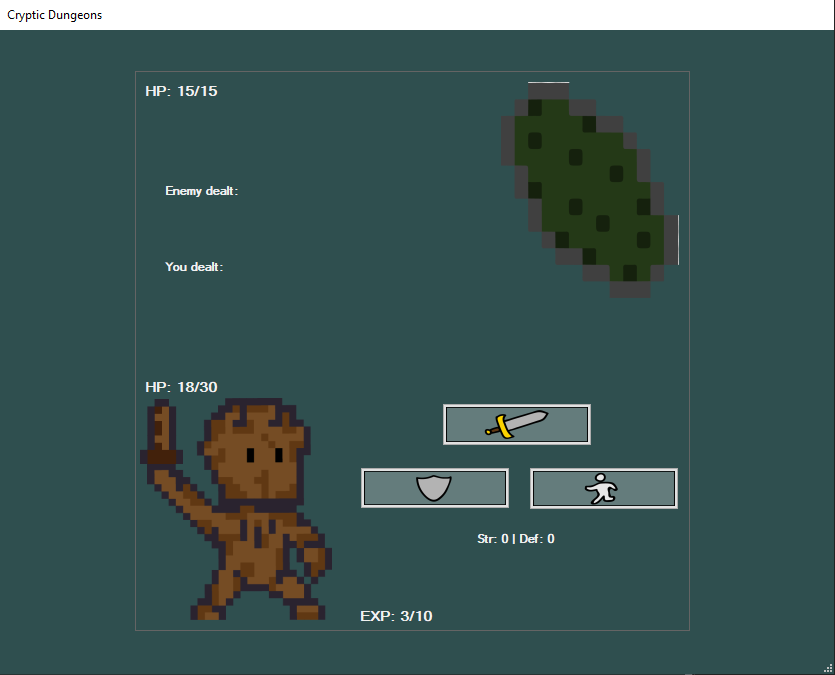
Ko igralec pride na misijo, se mu odpre zgornja slika.

Tu ima nato izbiro med spopadom in naključnim izborom.

Naključni izbor ima 30% možnost preskoka stopnje, 20% možnost da se mu zvišajo življenjske točke za 10 in 50% možnost spopada.

Na tretji stopnji ima igralec možnost spopada s posebnim nasprotnikom, ki je sicer močnejši, ampak je tudi več vreden in ima možnost puščanja kamnov za nadgrajevanje za sabo.

Zadnja stopnja misije je spopad s končnim nasprotnikom, ki pusti za sabo materiale, če ga igralec premaga.



Slika 5: Glavni spopad - bitka

Če igralec izbere bitko, se mu prikaže ta zaslon.

Na vrhu se vidi nasprotnikove življenjske točke, pod njimi se izpisuje igralčev in nasprotnikov napad, pod tem so pa še igralčeve življenjske pike.

Levo spodaj je prikaz igralca, desno zgoraj pa prikaz nasprotnika.

Desno od igralca vidimo 3 gumbe: gumb za napad, gumb za obrambo in gumb za beg iz boja.

Pod gumbi vidimo še atribute igralca, na dnu zaslona pa nam prikazuje še naše točke izkušenj.

Igra deluje tako, da igralec izbere enega izmed treh gumbov in poskuša zbiti nasprotnikove življenjske točke na 0, medtem ko sam proba preživeti spopad.

Če igralec nasprotnika premaga, se mu odklene naslednja stopnja, vse do končnega spopada, kjer ob zmagi pridobi materiale, katere lahko kasneje uporabi za izdelavo opreme.

Ob izbiri spopada se izvede funkcija za prikaz borbenega prizora, ki je podobna kot funkcija za prikaz in skrivanje opreme na domačem zaslonu, in sicer gre program skozi zanko, kjer preverja če so vsi argumenti pravilni in prikaže, kar je potrebno.

Ob vsaki zmagi se izvede funkcija, ki igralcu dodeli priigrane kovance in točke izkušenj, hkrati pa še preveri, če je igralec pridobil dovolj točk izkušenj za zvišanje njegovega nivoja.

Poleg pridobivanja zasluženih točk in kovancev program še zaklene trenutni izbor in odklene naslednji izbor stopnje (naključni izbor/spopad/posebni nasprotnik).

Če je igralec premagal posebnega nasprotnika, pridobi še kamen za nadgrajevanje opreme.

Če igralec premaga končnega nasprotnika nivoja pa pridobi naključno število izbranega materiala, ki ga lahko kasneje uporabi za izdelavo opreme.

Pri spopadu je igralčev napad izračunan s seštevanjem polovice igralčeve moči, naključnim številom med najnižjim in najvišjim napadom izbranega orožja in dodatnimi napadnimi točkami pridobljenimi iz uporabe mesa iz lova:

x = p.str / 2 + r.Next(selectedWeapon.weaponDmgMin, selectedWeapon.weaponDmgMax + 1) + foodDmg;

Nasprotnikov napad pa z naključnim številom med najnižjim in najvišjim možnim nasprotnikovim napadom, od tega se pa še odšteje polovična igralčeva obramba in tretjina obrambe izbranega oklepa:

p.y = r.Next(selectedMob.minMobDmg, selectedMob.maxMobDmg + 1);

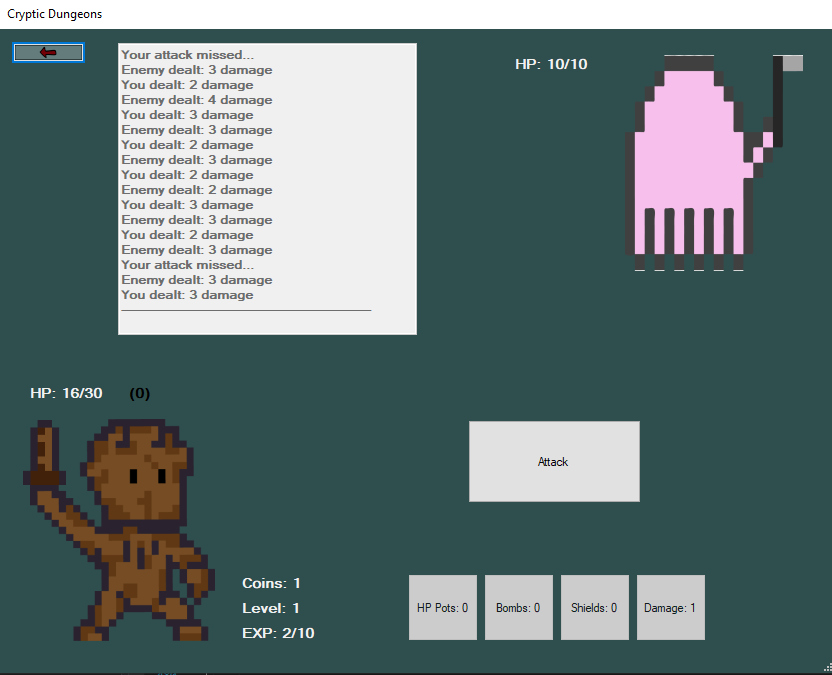
p.y = p.y - p.def / 2 - selectedArmor.armorDefense / 3;

Boj je zgrajen tako, da se takoj po igralčevem napadu izvrši še nasprotnikov napad.

Poleg možnosti za napad lahko igralec izbere obrambo, ob kateri ima možnost preslikati nasprotnikov napad nazaj k njemu, ter zniža nasprotnikov za polovico izbranega igralčevega oklepa.

Tretja možnost pri spopadu je še pobeg iz bojišča, kar pa ima 30% možnost uspeha.

## **3.6 Neskončni spopad**



Slika 6: Neskončni spopad

Uporabnik pride do zgornjega okna, če izbere katerokoli »Grind« opcijo iz zemljevida.

Za spremembo od glavnega spopada, tu igralec ne more priti do končnega nasprotnika nivoja in ne more pridobiti njegovih materialov.

Ta način igranja je tudi drugačen od glavnega spopada, saj tu igralec nima možnosti obrambe ali bega iz bojišča, ampak pridobi druge funkcije.

Pri neskončnem spopadu ima igralec na voljo 4 dodatne možnosti:

* Življenjski napitek (za zvišanje življenjskih pik)
* Eksploziv (za močne nasprotnike – naredi veliko večji napad)
* Ščit (se doda k igralčevim življenjskim pikam in nasprotnik lahko napade le ščit)
* Dodatne napadalne točke (zvišajo igralčev naslednji napad)

Za razliko od glavnega spopada, lahko igralec tu hitreje pridobi točke izkušenj in kovance, saj so spopadi hitrejši.

Poleg hitrejšega pridobivanja moči pa mora biti igralec tu tudi bolj pazljiv, saj lahko izgubi točke izkušenj, če izgubi vse življenjske pike.

Igralec pa lahko iz spopada pobegne nazaj na domač zaslon kadar želi.

Neskončni spopadi so določeni za vsak nivo in vsak ima drugačne nasprotnike.

Izbira nasprotnikov se izvede glede na izbrano stopnjo:

Type[] chooseMob = Assembly.GetAssembly(typeof(MobLowRank)).GetTypes().Where(TheType =>

TheType.IsClass && !TheType.IsAbstract &&

TheType.GetInterfaces().Contains(typeof(MobLowRank))).ToArray();

x = r.Next(chooseMob.Length);

ConstructorInfo ctor = chooseMob[x].GetConstructor(new Type[0]);

selectedMob = (Mob)ctor.Invoke(new object[] { });

Prvi stavek v tej kodi poišče nasprotnike, ki imajo značko »MobLowRank« pri zapisu. V temu primeru bodo izbrani nasprotniki za prvi nivo.

Naslednja dva stavka naključno izbereta enega izmed teh nasprotnikov in ga shranita.

Zadnji stavek zapiše izbranega nasprotnika v spremenljivko »selectedMob«, ki se uporabi za prikaz nasprotnika na zaslonu.

Po vsaki zmagi igralec pridobi nov predmet, ki mu pomaga pri naslednjem boju (napitek/eksploziv/ščit/dodatni napad), lahko pa uporabi le enega na spopad.

Ta način igre je namenjen pridobivanju izkušenj, da lahko uporabnik premaga končnega nasprotnika v glavnemu spopadu.

Ko igralčeve točke izkušenj presežejo prag, se zviša igralčev nivo. S tem igralec pridobi 2 točki, ki ju lahko spravi v moč ali obrambo, poleg tega se pa igralcu še zvišajo življenjske pike za 5 pik.

Zadnji nivo, ki ga igralec odklene je neskončni spopad vseh nasprotnikov, kjer se mu prikažejo vsi nasprotniki, ki so nastopili v igri. Narejen je na isti koncept kot neskončni spopad nivoja, le da se tam prikažejo vsi nasprotniki in igralec lahko pridobi več točk izkušenj.Po vsaki zmagi se izvede naslednja funkcija:

public void dropItems()

{

x = r.Next(12);

if (x >= 0 && x <= 2)

p.items.healthPot.incrementQuantity(1);

else if (x >= 3 && x <= 5)

p.items.shield.incrementQuantity(1);

else if (x >= 6 && x <= 8)

p.items.bomb.incrementQuantity(1);

else

p.items.sword.incrementQuantity(1);

updateItems();

}

Kar se tu zgodi je, program naključno dodeli igralcu nek predmet, ki ga lahko uporabi v naslednjem spopadu.

Program tu namesto dodajanja količine izbranega predmeta preko npr. x = x + 1, uporabi funkcijo »incrementQuantity()«, ki je zapisana v datoteki s podatki.

Po dodelitvi predmeta igralcu, se izvede še funkcija »updateItems()«, ki posodobi igralčeve predmete. Ta posodobitev se zgodi tudi po uporabi predmeta.

Predmet se odklene, če je njegova količina večja od 0.

Moč predmeta se prilagaja glede na izbran nivo. Primer:

private void btnShield\_Click(object sender, EventArgs e)

{

if (p.items.shield.itemQuantity > 0)

{

p.items.shield.decrementQuantity(1);

lockItems();

updateItems();

if (p.stageRank == 1)

p.shields += 6;

else if (p.stageRank == 2)

p.shields += 10;

else if (p.stageRank == 3)

p.shields += 15;

lblShields.Text = "(" + p.shields + ")";

}

}

Pri uporabi predmeta se zniža njegova količina s funkcijo »decrementQuantity()«, takoj za tem se zaklenejo vsi predmeti, ter se posodobijo njihove količine.

Za tem program preveri kateri nivo je izbran in prilagodi moč predmeta glede nanj.

## **3.7 Končni nasprotnik**



Slika 7: Končni nasprotnik

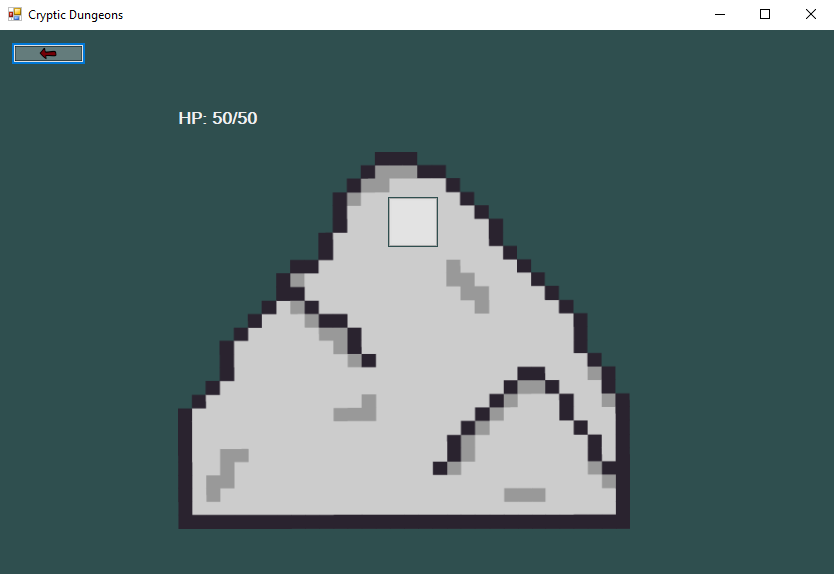
Spopad s končnim nasprotnikom je mešanica glavnega borbenega prizora in neskončnega spopada.

Igralec ima tu na voljo 3 življenjske napitke, poleg tega pa ima na voljo še možnost napada in obrambe.

Ko igralec premaga in izdela orožje in oklep od končnega nasprotnika, se mu odklene še neskončni spopad z vsemi nasprotniki.

Končno orožje ima še posebni učinek, ki se lahko naključno sproži ob napadu in doda uporabniku dodatnih 5 napadalnih točk za 3 napade.

## **3.8 Rudarjenje**



Slika 8: Rudarjenje

Pri pritisku na »Mining« iz zemljevida, je uporabnik popeljan na to okno.

Tu mora igralec pritisniti na gumbe, ki se prikazujejo na zaslonu, da pridobi rudo, potrebno za izdelavo določene opreme.

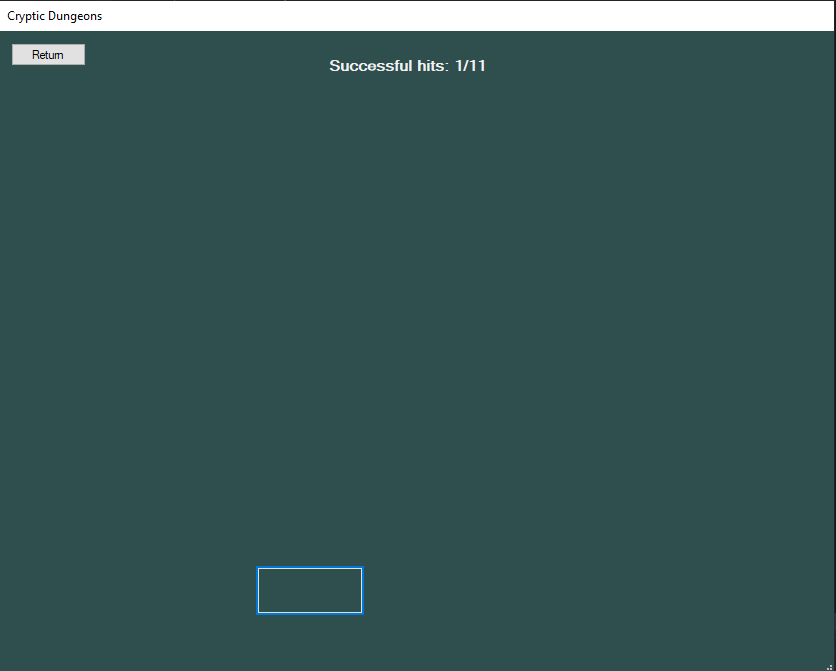
Ko igralec zadene gumb dostikrat, se kamen poruši in igralec pridobi material.

Ob uničenju, se izvede stavek:

p.inventory.addItem(selectedOre, 1);

Ta stavek uporabi funkcijo »addItem()«, ki je zapisana v datoteki s podatki.

## **3.9 Lov**



Slika 9: Lov

Ko želi igralec iti na lov, pritisne na zemljevidu gumb »Hunting«.

Ta igra deluje tako, da se na zaslonu prikaže gumb, ki se prestavi, ko ga igralec pritisne.

Ko igralec uspešno zadene gumb 11-krat je konec igre in uporabnik pridobi meso, ki ga lahko nato uporabi za zvišanje napada za izbrano misijo.

public void randomizeButtonLocations()

{

xx = r.Next(20, 720);

yy = r.Next(100, 580);

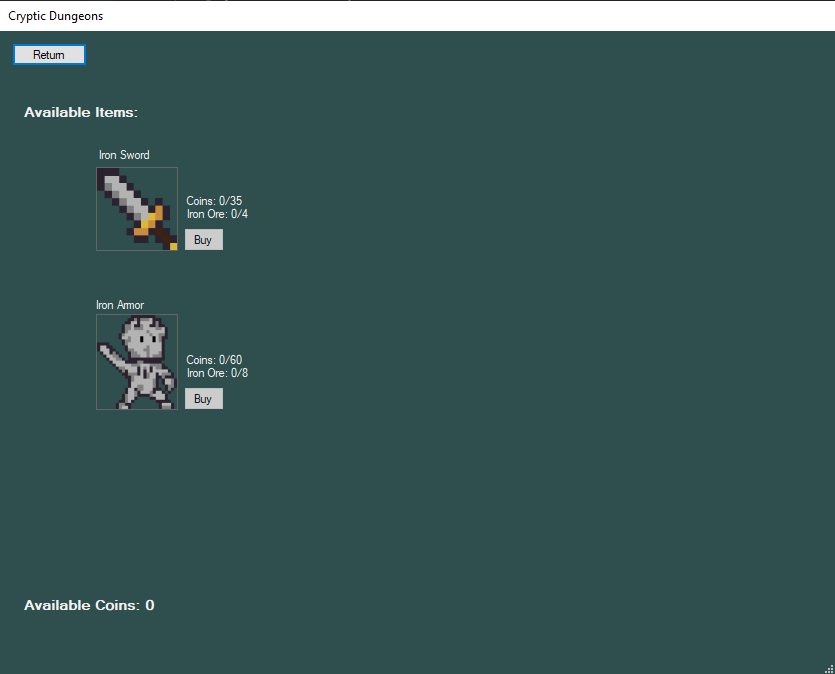
button1.Location = new Point(xx, yy);

}

Zgornji zapis kode se uporabi za naključno dodeljevanje naslednje pozicije gumba.

Spremenljivka »xx« predstavlja X koordinato, spremenljivka »yy« pa Y koordinato v programu.

## **3.10 Trgovina**



Slika 10: Prikaz trgovine

Ko pritisnemo na gumb za trgovino na našem domačem zaslonu se odpre nov Windows Form, ki prikazuje orožje in oklep, katerega lahko nakupimo s kovanci in železom.

Na zgornji sliki vidimo opremo, ki se odklene za nakup, ko so izpolnjeni vsi pogoji (pravilna količina kovancev in železa).

* Na vrhu je gumb »Return«, ki nas vrne na domači ekran
* Pod njim je oprema, ki jo lahko nakupimo (gumb »Buy« se odklene, ko ima igralec vse materiale)
* Na dnu se nam prikaže naše število kovancev.

Koda za pregledovanje razpoložljivosti izdelka:

public void isAvailable()

{

if (p.coins >= p.weapons.ironS.coinValue && p.weapons.ironS.isObtained ==

false && p.inventory.requiredMaterials(9) >= 4)

btnBuyIronS.Enabled = true;

Ta funkcija se izvede ob inicializaciji forme in preveri če ima igralec dovolj kovancev in rude za nakup želenega kosa opreme, ter odklene gumb za nakup, če so izpolnjeni vsi pogoji.

public void updateMaterials()

{

lblIronS.Text = "Coins: " + p.coins + "/35" + Environment.NewLine + "Iron

Ore: " + p.inventory.requiredMaterials(9) + "/4";

lblIronA.Text = "Coins: " + p.coins + "/60" + Environment.NewLine + "Iron

Ore: " + p.inventory.requiredMaterials(9) + "/8";

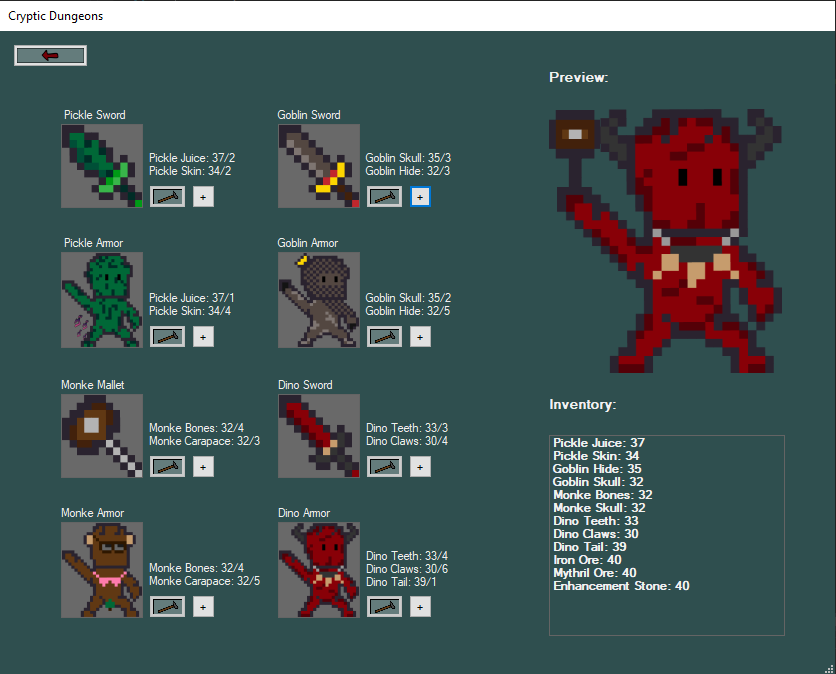
}

Zgornja koda se uporabi za posodobitev materialov, ki jih ima igralec v svoji torbi.

Ukaz »Enviroment.NewLine« se uporabi za preskok v novo vrstico.

V kodi se za prikaz potrebnih materialov uporabi funkcija »requiredMaterials()« iz datoteke s podaki, ki uporabi identifikacijsko številko materiala, da programu pove kateri material naj preverja.

## **3.11 Kovač**



Slika 11: Prikaz kovača

Ob pritisku na gumb z ikono nakovala na domačemu zaslonu se nam odpre okno, kjer lahko izdelujemo in nadgrajujemo opremo.

* Na vrhu zopet vidimo gumb za povrnitev na prejšnje okno
* Pod njim imamo orožja in oklepe, ki jih lahko izdelamo v igri. Oprema je zaklenjena dokler igralec ne pridobi vseh potrebnih materialov za izdelavo. Materiale igralec pridobiva na misijah, ti se shranijo v igralčevo torbo, prikažejo se pa tudi nad gumbom za izdelavo dela opreme, kar nam pove koliko materialov imamo in koliko jih še potrebujemo.
* Na desni strani imamo izgled igralca in igralčevo torbo z vso vsebino.

Pri kovaču je pregled razpoložljivosti izdelave drugačen kot pri trgovini, saj se tu uporablja še recept za izdelavo opreme.

V programu je zapisana funkcija »isCraftable()«, ki preverja, če ima igralec dovolj materialov za izdelavo želene opreme:

if (p.recipes.dinoS.isCraftable(ref p) == true && p.weapons.dinoS.isObtained == false)

{

btnCraftDinoS.Enabled = true;

btnCraftDinoS.BackColor = Color.DarkSlateGray;

}

V temu stavku program kliče funkcijo iz datoteke s podatki, ki primerja igralčeve materiale z materiali potrebnimi za izdelavo dela opreme.

Poleg tega preveri še, če je uporabnik orožje ali oklep že pridobil oz. izdelal in mu prepreči ponovno izdelavo.

Ko je pa igralec enkrat kos opreme izdelal, se mu prikaže gumb za nadgradnjo, če ima dovolj kamnov za nadgradnjo in kovancev.

Določen del opreme se lahko nadgradi do 5. stopnje.

Pri orožjih se zviša napad, pri oklepu pa obramba.

Koda za preverjanje razpoložljivosti za nadgradnjo:

if (p.weapons.pickleS.isUnlocked == true && p.weapons.pickleS.enchLevel < 5 &&

p.inventory.requiredMaterials(11) >= 1 && p.coins >= 10)

btnUpgradePS.Visible = true;

else

btnUpgradePS.Visible = false;

Tu program preveri, če je igralec opremo že pridobil, poleg tega pa še preveri, če je nivo nadgradnje manjši 5, če ima igralec dovolj kamnov za nadgradnjo, ter če ima dovolj kovancev.

Če so vsi pogoji zadovoljeni, se prikaže gumb »+« za nadgradnjo.

# **4 ZAKLJUČEK**

Projekt sem začel brez kakršnegakoli znanja razvijanja iger, poznal sem le določeno količino programskega jezika in večino kode, ki sestavlja program, sem moral naknadno ugotoviti s pomočjo interneta.

Na srečo je splet že toliko razvit, da sem našel vse z minimalnim trudom in brez pomoči StackOverflow spletne strani (namenjena pomoči pri pisanju programske kode) mi projekta ne bi uspelo narediti, vsaj ne tako optimizirano kot je končni produkt.

Skozi grajenje projekta sem ugotovil, da sem se pravilno odločil, saj mi je projekt razširil obzorje.

Če bi moral projekt ponovno začeti bi spremenil par stvari.

* Spremenil bi program za izrisovanje likov. Za ta projekt sem uporabljal Adobe Illustrator, saj je relativno enostaven za uporabo, mu pa manjkajo pripomočki za izdelovanje 8-bitnih likov. Če bi moral like izdelati na novo, bi uporabil program Aseprite, ki je namenjen za tako grafiko.
* Namesto izdelave programa, ki odpira razna okna, bi igro izdelal za HTML5 (naredil bi spletno aplikacijo), saj je grafika in stil igranja bolj primeren za splet.
* Od samega začetka bi razmišljal bolj v smeri optimizacije, saj sem za projekt porabil število ur, da sem spremenil zadeve, ki bi lahko že prvotno drugače zapisal.
* Pred začetkom izdelave programa, bi se bolj pozanimal o izvedbi programa in bi si zapisal bolj podroben načrt, da bi izdelava projekta potekala bolj tekoče.

Kljub vsem problemov, na katere sem naletel, je bil projekt precej informativen in sem se veliko naučil. Zahvaljujoč predmetu Programiranje 2 in profesorju Aleksandaru Lazareviću bom nadaljeval z razvijanjem iger in v upanju izdelal nekaj, kar bom lahko delil z ostalimi.

# **5 PRILOGE**

## **5.1 Nasprotniki**



Slika 12: Nasprotniki 2. Nivoja

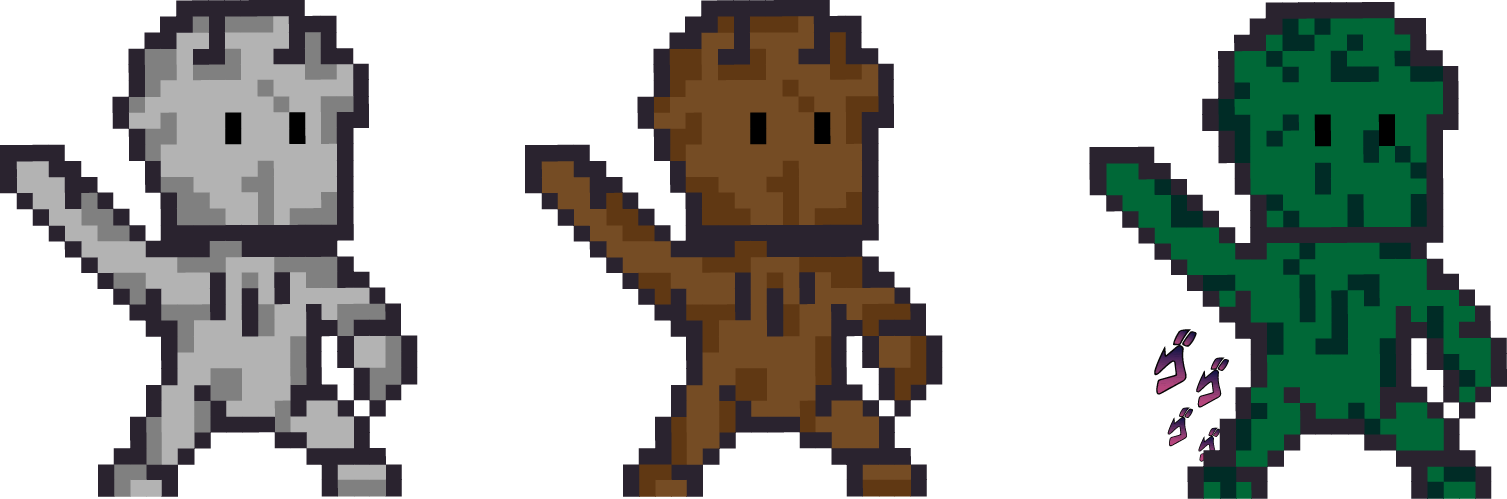


Slika 13: Nasprotniki 3. Nivoja



Slika 14: Končni nasprotnik

## **5.2 Oklepi**

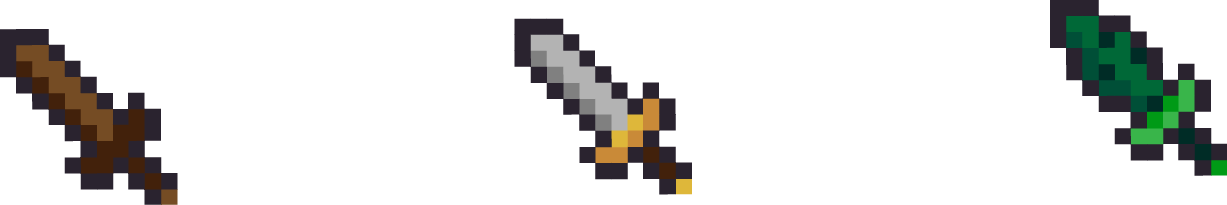


Slika 15: Železni oklep, Bronasti oklep, "Pickle Armor"



Slika 16: Goblin oklep, Opičji oklep, Dinozaver oklep

## **5.3 Orožja**



Slika 17: Lesen meč, Železen meč, "Pickle Sword"



Slika 18: Goblin meč, Opičji meč, Dinozaver meč