

Projektna naloga

Objektno usmerjena analiza in razvoj informacijskih sistemov

Izdelava igre Domesday

Avtor: Nik Kovačič Škofja Loka, 13.6.2019

**POVZETEK**

V projektni nalogi opisujemo razvoj računalniške igre. V njej najprej opredeljujemo problem s katerim smo se soočili. Cilje, ki smo jih želeli doseči ter programe oziroma orodja, ki smo jih uporabili pri izdelavi. Poleg tega v njej tudi opisujemo izbrano metodologijo ter obrazložimo idejo igre ter njeno delovanje. Na primeru izdelane igre prikazujemo tudi tri UML diagrame.

**KLJUČNE BESEDE:**

* Igra
* UML diagram
* Python
* Vesoljska ladja
* Vesoljec

Vsebina

[**1.** **UVOD** 4](#_Toc11333491)

[**1.1.** **OPREDELITEV PROBLEMA** 5](#_Toc11333492)

[**1.2.** **OPREDELITEV CILJEV NALOGE** 5](#_Toc11333493)

[**1.3.** **UPORABLJENA ORODJA** 5](#_Toc11333494)

[**2.** **METODOLOGIJA** 5](#_Toc11333495)

[**3.** **EKSPERIMENT** 6](#_Toc11333496)

[**3.1.** **IDEJA** 6](#_Toc11333497)

[**3.2.** **IGRA** 6](#_Toc11333498)

[**4.** **UML DIAGRAMI** 8](#_Toc11333499)

[**4.1.** **DIAGRAM PRIMERA UPORABE** 8](#_Toc11333500)

[**4.2.** **DIAGRAM RAZREDOV** 11](#_Toc11333501)

[**4.3.** **DIAGRAM ZAPOREDJA** 13](#_Toc11333502)

[**5.** **DISKUZIJA** 14](#_Toc11333503)

[**6.** **ZAKLJUČEK** 14](#_Toc11333504)

[**7.** **LITERATURA IN VIRI** 15](#_Toc11333505)

**KAZALO SLIK**

[Slika 1: Začetni zaslon igre 6](#_Toc11333506)

[Slika 2: Prikaz aktivne igre 7](#_Toc11333507)

[Slika 3: Diagram uporabe 8](#_Toc11333508)

[Slika 4: Zagon igre 9](#_Toc11333509)

[Slika 5: Diagram razredov 11](#_Toc11333510)

[Slika 6: Diagram zaporedja 13](#_Toc11333511)

# **UVOD**

Za projektno nalogo pri predmetu Objektno orientirana analiza in razvoj informacijskih sistemov smo izdelali igro v programskem jeziku Python (verzija 3.7.) s pomočjo knjižnice pygame. Izvorno kodo smo spisali v urejevalniku kode (code editor) JetBrains PyCharm (Community Edition 2018.3.4). Celotni projekt je sestavljen iz devetih python datotek (Python file), tekstovne datoteke (text file) in mape, ki vsebuje slike (/images). V python datotekah je izvorna koda, ki omogoča delovanje igre. Domesday.py vsebuje funkcijo run\_game, ki s pomočjo »while« zanke omogoča nemoteno potekanje igre. Vanjo so vključeni (import) tudi vsi potrebni razredi preostalih python datotek. To so game\_functions.py, spaceship.py, settings.py, game\_stats.py, scoreboard.py, play\_button.py. Preostali razredi kot so bullet.py in alien.py pa so vključeni v druge datoteke, ki se uporabljajo v datoteki domesday.py. Kot smo že omenili projekt vsebuje tudi tekstovno datoteko poimenovano score.txt, ki beleži najvišji doseženi rezultat v igri s pomočjo zapisa v to datoteko tudi, v primeru, da igro igralec ugasne.

Namen izdelane igre je doseči čim večji rezultat, ki pa ga dosežemo tako, da sestrelimo čim večje število vesoljcev, ki se spuščajo proti dnu ekrana. Streljamo jih z naboji, ki prihajajo iz naše vesoljske ladje na dnu ekrana. Pozicijo naše ladje nadzorujemo z tipkami levo in desno, streljamo pa z »SPACE« tipko. Igra se konča, ko 3 vesoljci pridejo na dno ekrana oziroma se zaletijo v našo vesoljsko ladjo. Da je igra bolj zanimiva se vsakič, ko sestrelimo vse vesoljce na ekranu te ponovno prikažejo in se pričnejo hitreje spuščati ter posledično tudi hitreje premikati iz leve proti desni. Da lahko uspešno nadaljujemo z igranjem igre se poveča tudi hitrost naše ladje in izstreljenih nabojev. Povečajo se tudi točke, ki jih pridobimo na vsakega sestreljenega vesoljca, saj je le-te zaradi povečane hitrosti tudi težje zadeti. Ker je bila igra v začetni fazi precej preprosta tudi ob povečani hitrosti zaradi možnosti držanja tipke »SPACE« in posledično neomejenega streljanja smo v kodi omejili število nabojev prikazanih na ekranu (na tri). S tem smo dosegli, da se igralec potrudi za vsak izstreljeni naboj, poleg tega pa je igra postala tudi bolj zahtevna. Ob igranju igre imamo prikazane tudi nekatere koristne informacije, kot so trenutni rezultat, najboljši doseženi rezultat, nivo (s številko) in preostala življenja, ki so ponazorjena z vesoljskimi ladjami v levem zgornjem kotu ekrana.

Poleg izdelave igre smo za potrebe projektne naloge na omenjenem primeru igre izdelali UML diagrame.

## **OPREDELITEV PROBLEMA**

Za potrebe projektne naloge, pri predmetu Objektno usmerjena analiza in razvoj informacijskih sistemov bomo izdelali računalniško igro v programskem jeziku Python. Cilj igre je doseči čim večje število točk, ki jih pridobimo z sestrelitvijo vesoljcev, ki se premikajo levo in desno ter spuščajo proti dnu ekrana. V igri ima igralec na voljo tri življenja, ki se zmanjšajo vedno, ko se vesoljec dotakne igralčeve ladje ali spusti na dno ekrana. Kodo smo spisali v urejevalniku PyCharm, v njej pa smo uporabil knjižnico pygame, ki je namenjena programiranju iger. Za potrebe te projektne naloge bomo narisal tudi UML diagrame s pomočjo programa JDeveloper. Naredili bomo diagram uporabe (angl. Use-Case), diagram razredov (angl. Class Diagram) in diagram zaporedja (angl. Sequence Diagram). Glavni problem naloge mi predstavlja izdelava igre, saj le-ta zahteva znaten časovni vložek. Poleg tega pa menim, da mi bo velik izziv predstavljala tudi izdelava vseh potrebnih UML diagramov, saj z njimi nimam veliko praktičnih izkušenj.

## **OPREDELITEV CILJEV NALOGE**

Cilji naloge so izdelati delujočo igro, narisati vse potrebne diagrame, spisati ustrezno dokumentacijo ter uspešno predstaviti nalogo.

## **UPORABLJENA ORODJA**

Pri izdelavi projektne naloge smo uporabil tri programe. To so:

1. PyCharm – v omenjenem programu smo spisali vso potrebno kodo za delovanje igre.
2. JDeveloper – s pomočjo tega programa smo izdelali UML diagrame.
3. Slikar – program smo uporabil za urejanje slik (sprememba velikosti, barve ozadja,..)

# **METODOLOGIJA**

Pri izdelavi seminarske naloge smo uporabil metodologijo računalništvo uporabnikov (EUC – End User Computing Methodology). Zanjo je značilno, da je oseba, ki izdela aplikacijo tudi njen uporabnik. Dokumentacije pri tej metodologiji ni veliko, saj uporabnik načeloma ve, kje lahko spremeni določene nastavitve aplikacije, ker jo je izdelal sam. Ta metodologija lahko podpira kompleksnost vendar je veliko drugih metodologij primernejših za kompleksne projekte. Rezultat načeloma ni zanesljiv, saj večino oseba, izdelava aplikacij ni primarna naloga osebe, ki je aplikacijo naredila. Nad izvedbo ni omogočena dobra preglednost so pa zanjo značilni nizki stroški.

V tem primeru je bila uporabljena ta metodologija, saj je igra izdelana le za potrebe projektne naloge in ni namenjena trženju ter uporabi drugih posameznikov.

# **EKSPERIMENT**

V tem poglavju vam predstavljam na kakšen način se mi je porodila ideja za izdelavo igre ter kratek opis igre ter njene uporabe.

## **IDEJA**

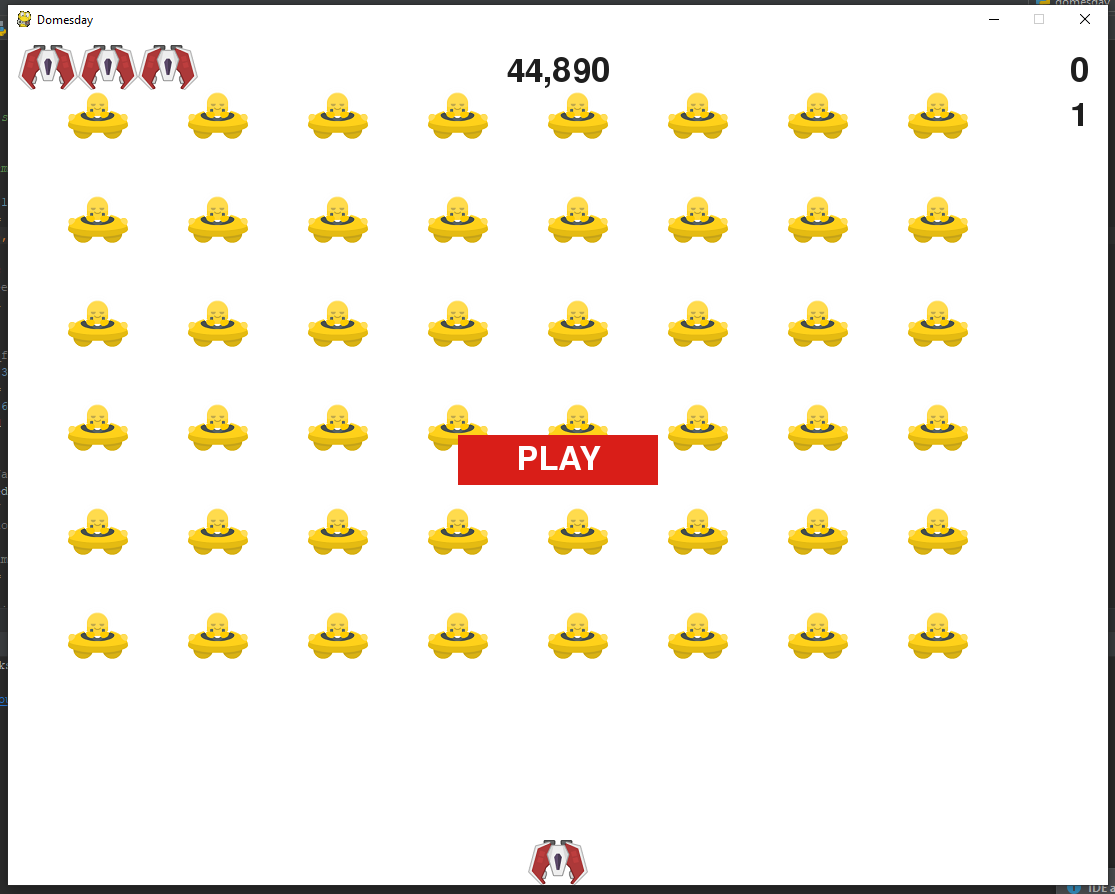
Kot otrok sem izredno rad igral arkadne igre. Najbolj v spomin se mi je vtisnila igra (imena se žal ne spomnim) v kateri si upravljal vesoljsko ladjo, ter z njo streljal na vesoljce, ki so se pomikali proti dnu ekrana. Na voljo si imel različne izstrelke, ki so povzročili drugačne efekte (večji naboji, laser, hitrejše streljanje, več izstrelkov na enkrat, itd.). Te si lahko pridobil z sestrelitvijo vesoljca ter pobiranjem zabojev na ekranu. V levem zgornjem kotu si imel prikazano življenje, na sredini zgoraj pa rezultat. Nasprotniki v igri so se prav tako razlikovali, nekateri so te lahko streljali nazaj, bili bolj odporni, ali pa zgolj preprosti nasprotniki. Glede na težavnost nasprotnika so se razlikovale tudi točke, ki si jih pridobil z njegovo sestrelitvijo. Igra je bila razdeljena na nivoje, na koncu vsakega nivoja pa te je pričakal »Boss«, ki je bil vedno najtežji nasprotnik v nivoju.

Na podlagi tega spomina sem pridobil idejo za izdelavo poenostavljene različice te igre.

## **IGRA**

V izdelani igri s pomočjo ladje streljaš na vesoljce, ki se pomikajo proti dnu ekrana. Na voljo imaš tri življenja, ki ti jih lahko odvzamejo vesoljci, ko se dotaknejo ladje ali pa dna ekrana. Igra teče vedno hitreje, kar jo naredi bolj zanimivo. S povečano hitrostjo in s tem tudi težavnostjo pa za vsakega sestreljenega vesoljca pridobiš večje število točk. Igra beleži tudi statistike in jih prikazuje na ekranu.

Slika 1 prikazuje začetni zaslon, ki se prikaže ob zagonu igre. Na sredini ekrana je prikazan rdeč gumb z napisom »PLAY«, ki zažene igro. To pa tako, da nastavi »flag« game\_active=True.



Slika 1: Začetni zaslon igre

Slika 2 nam prikazuje že aktivno igro v kateri je igralec že sestrelil večino vesoljcev na ekranu v prvem nivoju. Na omenjeni sliki sem s pomočjo slikarja označil pomembne dele igre. S pravokotniki so označene statistike igre, z krogi pa objekti.

Z rdečim pravokotnikom sem na sliki označil število življenj, ki jih ima uporabnik še na voljo. Z vsakim dotikom vesoljca ladje ali dna ekrana se izbriše ena ladja, ki predstavlja eno življenje.

Z zelenim pravokotnikom je označen najboljši dosežen rezultat, ki ga igra pridobi iz tekstovne datoteke (score.txt).

Z modrim pravokotnikom označujem trenutni rezultat, ki ga je dosegel igralec. Z vsakim sestreljenim vesoljcem se ta rezultat poveča za točke, ki mu jih je prinesel vesoljec. Točke se povečujejo z vsakim višjim nivojem.

V črnem pravokotniku je številka nivoja. Od nivoja je odvisna hitrost ter s tem tudi težavnost igre ter kot že omenjeno točke, ki jih igralec pridobi s sestrelitvijo vesoljca.

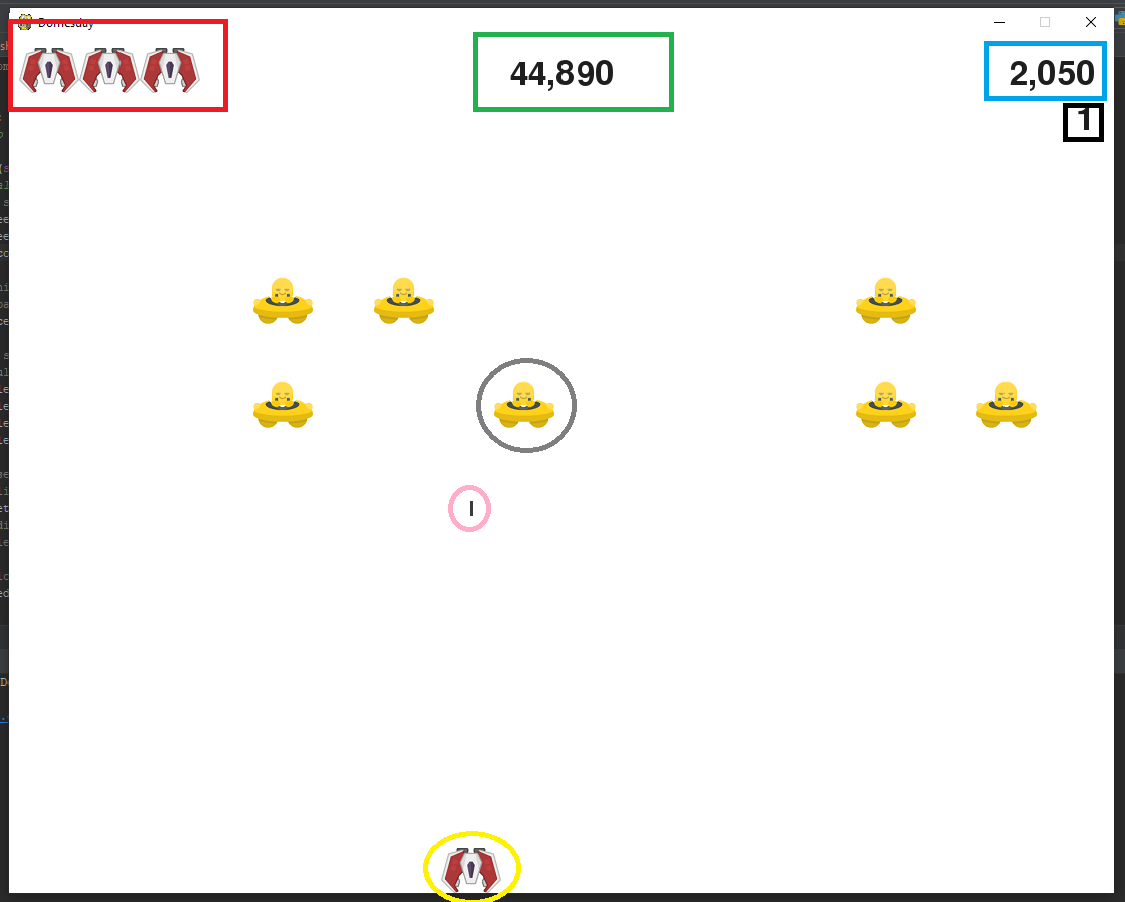
Z sivim krogom sem označil enega vesoljca v floti, ki predstavlja igralcu nasprotnika.

Z roza krogom je označen izstrelek, s katerim lahko uporabnik sestreli vesoljca. Izstrelek potuje iz sredine vesoljske ladje po ekranu navzgor, ter v primeru da zgreši vse vesoljce na ekranu izgine na vrhu ekrana. Da igra ni preveč preprosta sem omejil število izdelkov na ekranu. Ob istem času so na ekranu lahko le trije izdelki. Menim, da sem s tem uporabnika prisilil v bolj natančno streljanje ter se izognil samo hitremu udarjanju po preslednici.

V igri nastopa še vesoljska ladja, ki je označena z rumenim krogom. Njo upravlja uporabnik ter jo lahko premika levo, desno ter iz nje izstreli naboj. Kontrole s katerimi jo upravlja so:

* Tipka desno – premakni ladjo v desno
* Tipka levo – premakni ladjo v levo
* Preslednica – izstrel naboj

Uporabniku sta na voljo še dve kontroli, ki zaključita igro in zapreta ekran. To sta tipki Q in ESC.

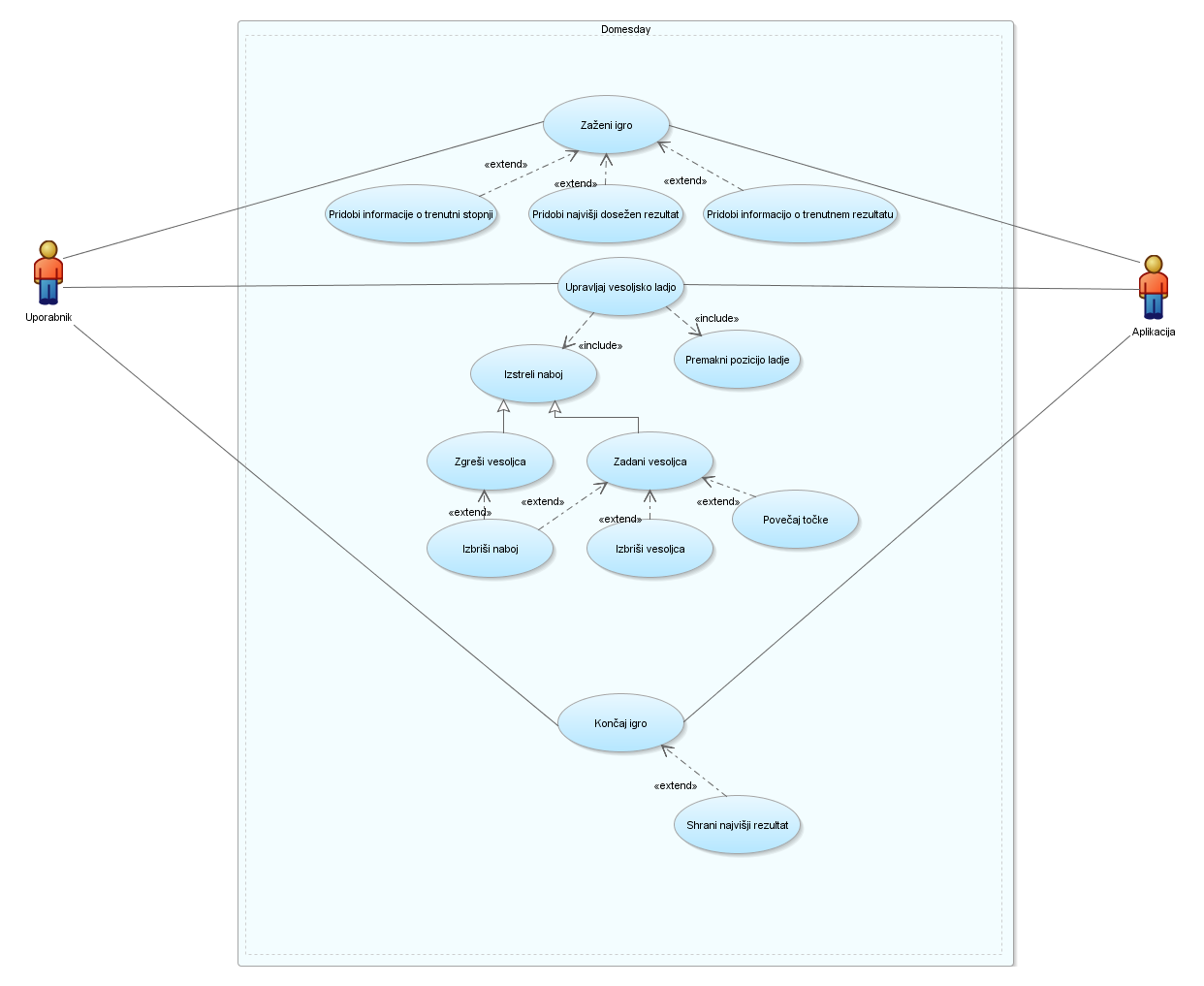


Slika 2: Prikaz aktivne igre

# **UML DIAGRAMI**

V tem poglavju vam prikazujem diagrame izdelane na primeru igre. Prikazani so diagram primera uporabe (Use-Case Diagram), diagram razredov (Class Diagram) in diagram zaporedja (Sequence Diagram).

## **DIAGRAM PRIMERA UPORABE**



Slika 3: Diagram uporabe

**OPIS DIAGRAMA UPORABE:**

Na sliki 1, je prikazan diagram uporabe (use-case diagram) za izdelano aplikacijo (igro Domesday). Sestavljen je iz igralcev, ki jih predstavljata:

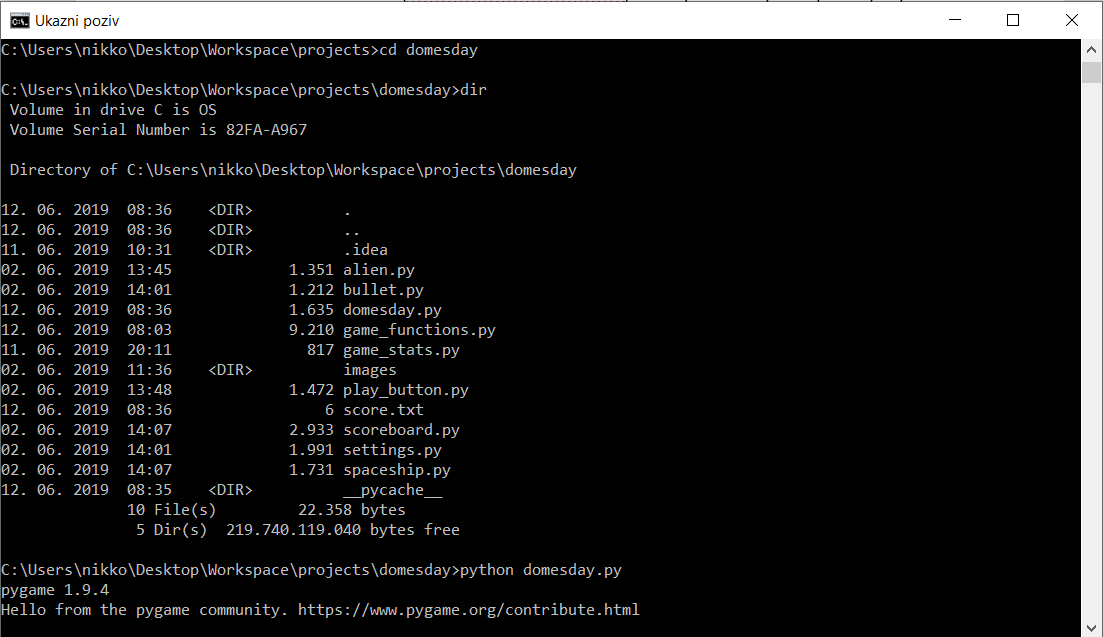
* Uporabnik – oseba, ki igra igro
* Aplikacija – sistem, ki zagotavlja nemoteno igranje igre

Primer uporabe aplikacije

Uporabnik s klikom na gumb »PLAY« zažene igro. V tem trenutku se prične igra, ki prikazuje tudi trenutni doseženi rezultat, trenutno stopnjo in najvišji doseženi rezultat, ki ga pridobi iz tekstovne datoteke score.txt. Uporabnik nato z tipkovnico lahko upravlja vesoljsko ladjo. Lahko spreminja njeno pozicijo z tipkami levo in desno ter izstreli naboje ob kliku na preslednico. V primeru, da uporabnik izstreli naboj lahko zadane ali pa zgreši enega izmed vesoljcev prikazanih na ekranu. Če vesoljca z nabojem zgreši, se naboj na vrhu ekrana izbriše, uporabnik pa za to ne pridobi nobenih točk. V nasprotnem primeru, ko uporabnik zadane vesoljca se izvrši funkcija, ki z ekrana izbriše zadetega vesoljca ter uporabljeni naboj, poleg tega pa tudi poveča trenutni rezultatu izpisan na ekranu oziroma doda točke (check\_bullet\_alien\_collisions). Omenjena funkcija tudi preveri, če je na ekranu prikazan še kakšen vesoljec. V primeru, da na ekranu ni več vesoljca se poveča hitrost igre, stopnja prikazana zgoraj desno na ekranu ter na ekran postavi nova grupa vesoljcev. Uporabnik lahko konča igro, ko zapre pojavno okno, klikne »tipko Q« ali pa ko izgubi vsa življenja. V tem trenutku se doseženi rezultat primerja z najvišjim rezultatom. V tekstovno datoteko se zapiše najvišji rezultat (write\_high\_score & check\_high\_score).

Pogoji za izvedbo

Aplikacija deluje na računalniku, ki ima nameščen programski jezik Python 3. Za zagon aplikacije priporočamo, da ima uporabnik nameščen tudi PyCharm, preko katerega lahko enostavno zažene igro. Lahko pa jo zažene tudi tako, da se uporabnik v terminalnem oknu postavi v mapo domesday in vpiše »python domesday.py«. Primer omenjenega načina zagona igre je prikazan na sliki 2.



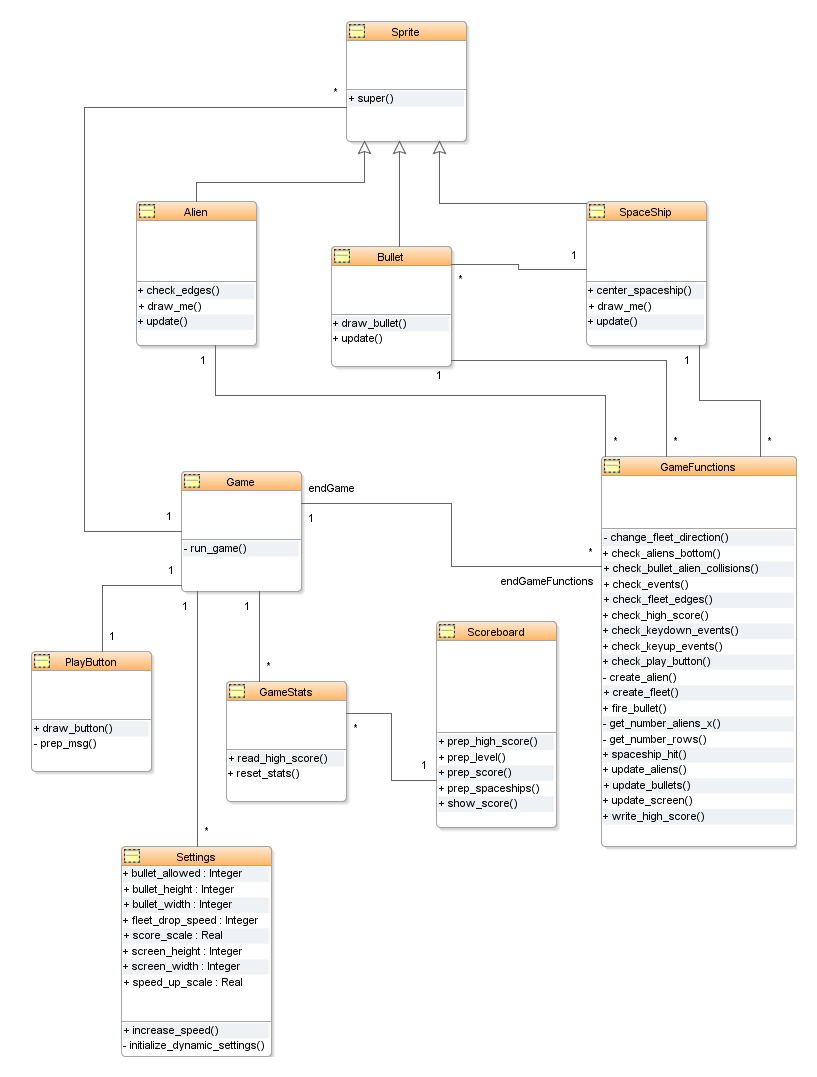
Slika 4: Zagon igre

Možne razširitve

Igro bi lahko dodatno izboljšal na mnogo različnih načinov, sam prelagam naslednje:

* Sprememba ozadja.
* Zvočni efekti.
* Težji nasprotniki (vesoljci lahko streljajo nazaj, potrebno jih je večkrat ustreliti…).
* Zapisom najboljših 10-ih rezultatov in imen igralcev, ki so jih dosegli.
* Z različnimi izstrelki.

## **DIAGRAM RAZREDOV**



Slika 5: Diagram razredov

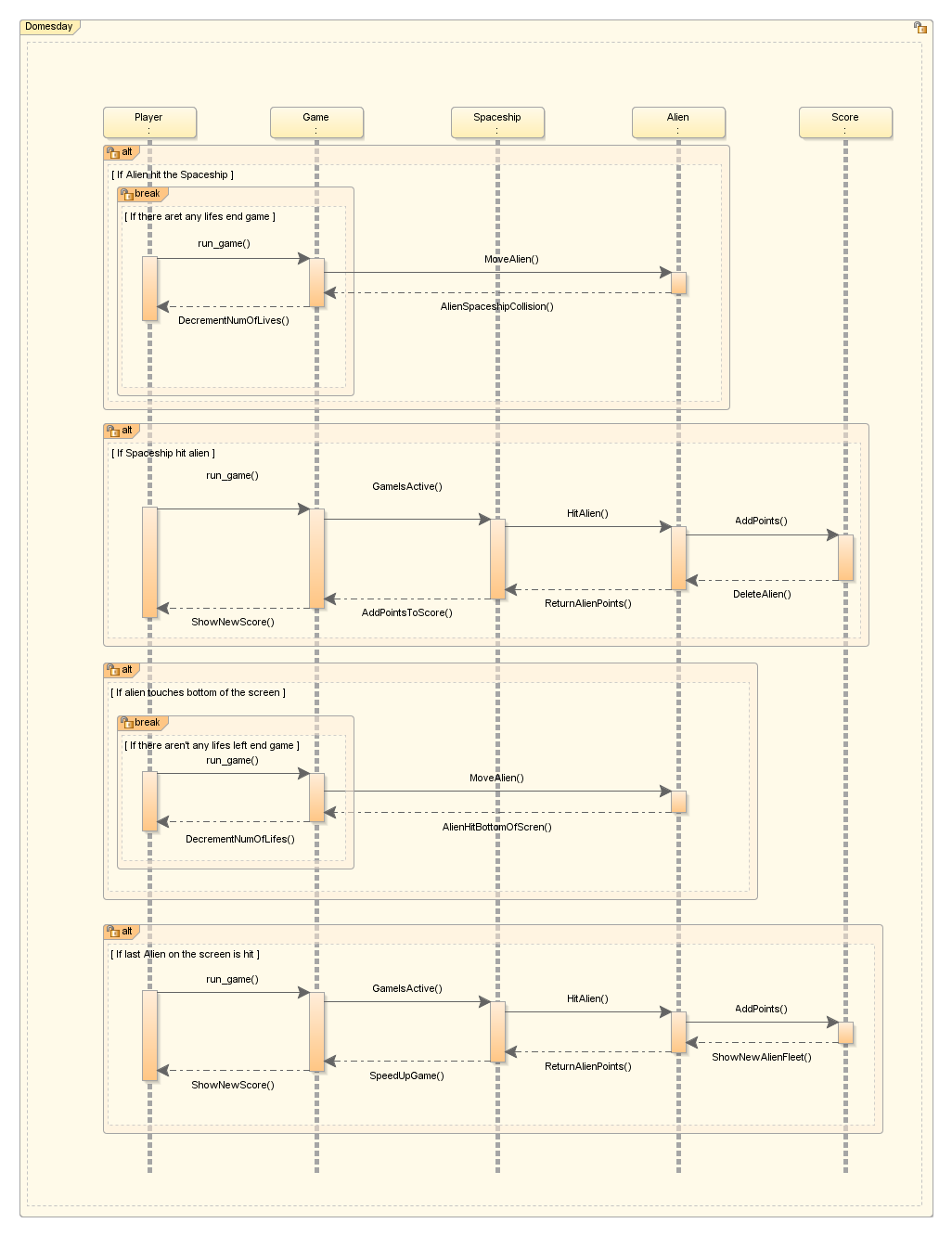
V izdelani aplikaciji nastopajo naslednji razredi:

* Sprite – je razred uvožen iz knjižnice pygame. Zagotavlja nam startno točko za kodo, ki je podobna pri veliki večini iger napisanih v programskem jeziku python. Sestavlja ga veliko število že definiranih funkcij oziroma metod, v našem primeru pa je bila koristna funkcija super(), ki vrne proxy objekt, ki posreduje klice metod razredu starševskega ali sorodnega tipa.
* Alien – je razred, ki predstavlja posameznega vesoljca v floti vesoljcev na ekranu. Sestavljajo ga na sliki naštete funkcije. Razred Alien podeduje funkcije razreda Sprite.
* Bullet – predstavlja posamezni izstrelek na ekranu, prav tako kot razred Alien podeduje funkcije razreda Sprite.
* SpaceShip – razred predstavlja vesoljsko ladjo, ki jo upravlja igralec. Sestavljajo ga na sliki 3 prikazane funkcije, ki določajo lokacijo na ekranu, omogočajo spremembo smeri in jo prikažejo na igralni površini (v pojavnem oknu). Tudi ta razred podeduje funkcije oziroma metode razreda Sprite.
* Game – omenjeni razred vsebuje metodo run\_game, ki zažene igro.
* GameFunctions – ta razred vsebuje veliko število metod oziroma funkcij, ki so potrebne za uspešno delovanje igre. Večina le-teh je javnega tipa, saj se uporabljajo tudi za potrebe drugih razredov.
* PlayButton – to je razred, ki prikaže gumb »PLAY« na ekranu. Z klikom na ta gum se prične nova igra.
* Settings – razred, ki shranjuje vse nastavitve igre Domesday.
* GameStats – razred, ki je namenjen sledenju statističnih podatkov igre.
* Scoreboard – skrbi za poročanje trenutnih statističnih podatkov igralcu (prikaz preostalih življenj, trenutnega rezultata, najvišjega rezultata ter trenutne stopnje).

Iz slike 3 je razvidno, da je diagram razredov v mojem primeru precej obsežen, saj vsebuje deset razredov, med katerimi pa je eden izmed njih del pygame knjižnice. Na sliki so razvidni atributi, ki so podani le za razred Settings, ki shranjuje pomembne nastavitve igre. Vsi podani atributi so javnega tipa, kar pomeni, da do njih lahko dostopamo tudi v preostalih razredih. Na sliki je prikazanih tudi mnogo povezav oziroma relacij med razredi. Zaradi velikega števila teh povezav sem se odločil, da bom opisal le nekatere izmed njih.

* Relacija Game-GameFunctions: omenjena povezava je tipa ena proti mnogo. Kar pomeni, da ena igra vsebuje več funkcij iz razreda GameFunctions.
* Relacija Game-PlayButton: omenjena povezava je tipa ena proti ena, kar pomeni, da igra vsebuje samo en »gumb PLAY«.
* Relacija Sprite-Alien: ta povezava prikazuje da razred Alien podeduje lastnosti (atribute in metode) razreda Sprite. V tem primeru je razred Sprite »superclass«, razred Alien pa »subclass«.

## **DIAGRAM ZAPOREDJA**



Slika 6: Diagram zaporedja

Na sliki 4 vam prikazujem diagram zaporedja, ki je prikazan glede na različne alternative, ki predstavlja potek dogodkov. Razdelil sem ga na različne alternative, ki se lahko dogodijo v igri.

# **DISKUZIJA**

Dandanes imamo samo na spletu ogromno iger, ki so dostopne praktično vsakomur. Preden sem izdelal to seminarsko nalogo sem menil, da je večin prosto dostopnih iger zelo nedovršena ter, da so narejene »na hitro« in brez tehtnega premisleka. Sedaj po končani projektni nalogi vidim, da je izdelava preprosti 2D iger daleč od tega, da bi bila preprosta. Že v izdelavo preproste igre je potrebno vložiti ogromno truda in časa, sedaj mnogo bolj cenim delo vseh, ki so izdelali igre ter jih ponudili na spletu zastonj, ter zaradi tega niso v zameno pridobili nobenega dobička.

To spoznanje je povišalo moje zanimanje za izdelavo iger ter drugih aplikacij. Predvsem me zanima, koliko truda, časa, energije in denarja je potrebno vložiti v izdelavo obsežnejših iger in aplikacij.

# **ZAKLJUČEK**

Z izdelavo projektne naloge sem kljub temu, da je bila izjemno zahtevna in mi je že v začetku predstavljala velik izziv izredno zadovoljen. Menim, da sem skozi njeno izdelavo pridobil teoretičnega in praktičnega znanja, za katerega ne dvomim, da mi bo v prihodnosti koristil.

Bolje sem se spoznal z principi objektnega programiranja ter tako nadgradil svoje znanje programiranja. Pri izdelavi sem bolje spoznal tudi uporabo različnih programov, ter osvojil osnove risanja UML diagramov, ki so izjemno koristni za dosego boljšega razumevanja med poslovno skupino in IT ekipo (informatiki).

# **LITERATURA IN VIRI**

* Built-In Functions. (13.6.2019). Pridobljeno na: <https://docs.python.org/2/library/functions.html#super>
* Eric Matthes. 2017. *Python Crash Course*. Sixth printing.
* Kenney. (13.6.2019). Pridobljeno na: <https://kenney.nl/>
* Pygame documentation. (13.6.2019). Pridobljeno na: <https://www.pygame.org/docs/>
* PythonForBeginners. (13.6.2019). Pridobljeno na: <https://www.pythonforbeginners.com/files/reading-and-writing-files-in-python>