**SVEUČILIŠTE U SPLITU**

**PRIRODOSLOVNO-MATEMATIČKI FAKULTET**

**DOKUMENTACIJA PROJEKTA**

OBJEKTNO ORIJENTIRANO PROGRAMIRANJE

VIDEO IGRA - METAL SLUG

**NAZIV KOLEGIJA:** Objektno orijentirano programiranje **STUDENT:** Rei Krstić

**PROFESORI:** Saša Mladenović i Divna Krpan **NAZIV STUDIJA:** Informatika 2

Sadržaj:

[1 TEMELJNA IDEJA 1](#_Toc170332106)

[2 PRIČA 1](#_Toc170332107)

[3 CILJ, CILJEVI 1](#_Toc170332108)

[4 LOGIKA IGRE 1](#_Toc170332109)

[5 MEHANIKA 2](#_Toc170332110)

[6 BODOVI 2](#_Toc170332111)

[7 NAPREDOVANJE 2](#_Toc170332112)

[8 OKRUŽENJE 2](#_Toc170332113)

[9 LIKOVI 2](#_Toc170332114)

[10 LIKOVI BEZ MOGUĆNOSTI UPRAVLJANJA 3](#_Toc170332115)

[11 MOGUĆE PRIKUPLJANJE RESURSA 3](#_Toc170332116)

[12 NAČIN KONTROLE U IGRI 3](#_Toc170332117)

[13 KRAJ IGRE/SIMULACIJE, APLIKACIJE 3](#_Toc170332118)

[14 OBLIKOVANJE KLASA 1](#_Toc170332119)

[14.1 Prva verzija dijagrama klasa 1](#_Toc170332120)

[14.2 Završna verzija dijagrama klasa (statičke klase – zeleno, apstraktne klase - crveno) 2](#_Toc170332121)

[14.3 Opis klasa 1](#_Toc170332122)

[14.3.1 Postavke (statička klasa) 1](#_Toc170332123)

[14.3.2 Character (apstraktna klasa) 2](#_Toc170332124)

[14.3.3 Marco 2](#_Toc170332125)

[14.3.4 Rebel (apstraktna klasa) 3](#_Toc170332126)

[14.3.5 RifleRebel 4](#_Toc170332127)

[14.3.6 PlaneRebel 4](#_Toc170332128)

[14.3.7 Platform 5](#_Toc170332129)

[14.3.8 NoGravItem 5](#_Toc170332130)

[14.3.9 Bullet 5](#_Toc170332131)

[14.3.10 HMGBullet 6](#_Toc170332132)

[14.3.11 FSBullet 6](#_Toc170332133)

[14.3.12 RLBullet 6](#_Toc170332134)

[14.3.13 RifleRebelBullet 7](#_Toc170332135)

[14.3.14 PlaneRebelBullet 7](#_Toc170332136)

[14.3.15 POW 7](#_Toc170332137)

[14.3.16 PowerUp 7](#_Toc170332138)

[14.3.17 GoSign 7](#_Toc170332139)

# TEMELJNA IDEJA

Ovaj projekt iz Objektno orijentiranog programiranja (skraćeno OOP) biti će utemeljen na konceptima po prvoj igri u Metal Slug seriji (prva igra se punim imenom naziva Metal Slug: Super Vehicle-001). Kao i originalna igra, projektna igra će biti žanrova: 2D (dvodimenzionalnog) platformer, te „Run 'n gun“ (hrv. „trčanje i pucanje“).

# PRIČA

U prvoj igri, igrači preuzimaju uloge vojnika Peregrine Falcon Squad-a iz Regularne vojske, Marco Rossija (našeg glavnog igrača u projektu) i Tarme Rovinga, i moraju se suočiti s negativcem Generalom Mordenom i njegovom Pobunjeničkom vojskom (naših glavnih neprijatelja u projektu, eng. Rebel army), koji želi eliminirati vlade i uspostaviti novi poredak, sve to potaknuto njegovim bijesom uzrokovanom smrću svoje obitelji koja bi mogla biti spriječena da Regularna vojska nije korumpirana.

# CILJ, CILJEVI

Primarni cilj u prvoj "Metal Slug" igri (kao i u cijeloj seriji igara) je prolazak kroz različite razine ispunjene neprijateljskim vojnicima i drugim preprekama, izbjegavajući napade neprijatelja i skupljajući „power-upove“ koje nam daju ratni zarobljenici. Dodatno, igrači ciljaju postići visoke rezultate (eng. „high scores“) efikasnim uništavanjem neprijatelja, spašavanjem ratnih zarobljenika i što bržim završetkom razina.

# LOGIKA IGRE

Prilikom gubljenja svih života, glavnog se igrača šalje na početak prve misije. Inače se samo stvori na ~~(mjestu gdje je „poginuo“)~~ početku te misije. Tri „power-upa“ imaju različite atribute: Heavy Machine gun, odnosno „mitraljez“ puca metke brže od početnog pištolja, Flame shot puca velike količine vatre koje rade više „damagea“, odnosno štete od primjerice pištolja, no zato može pogoditi „metu“ samo „s blizine“~~(, te puca sporije)~~, a Rocket launcher ili bacač raketa puca rakete koje ~~(su mješavina Heavy machine guna i Flame shota)~~ su sporije, ali zato rade više damagea. Posebna oružja, odnosno „power-upovi“ imaju svoju određenu količinu municije. Ako smo dovoljno blizu neprijatelja prilikom „pucanja“ onda ga umjesto pucanja, bodemo našim nožem.

# MEHANIKA

Naš protagonist može trčati lijevo/desno, te može skakati. Bitan dio igre je, očito, i pucanje. Može se pucati prema gore, dolje, lijevo i desno. Na početku puca iz pištolja, no može skupiti tri različita „power-upa“ koja mu daju različita oružja, odnosno municiju (eng. Heavy machine gun, Flame shot i Rocket laucher). Također možemo koristiti i nož. „Power-upove“ dobijamo tako što spašavamo ratne zarobljenike, odnosno POW (eng. Prisoners Of War). Naš igrač ima ~~(tri)~~ pet života. No naš igrač umire u jednom metku od neprijatelje~~(, baš kao i sami neprijatelji (u većini slučajeva)).~~ Skupljamo poene što bržim završavanjem razina, ubijanjem neprijatelja i spašavanjem „POW-a“ i još ako imamo dodatnih života pri pobjedi to se također ubraja.

# BODOVI

Rezultati na kraju završene igre ovise o: broju spašenih ratnih zarobljenika i ubijenih neprijatelja (također ovisi o tome koliko je jak koji neprijatelj), te o brzini završetka pojedinog levela i još ako imamo dodatnih života pri pobjedi to se također ubraja. Bodovi (eng. score) se prikazuju na kraju igre.

# NAPREDOVANJE

„Power-upovi“ su nam jedini oblik napredovanja, te možemo birati koji želimo, no samim tim izgubimo onaj koji smo imali. Kao što je prije rečeno, izgubimo „power-up“ (posebno oružje) tako što potrošimo sve njegove metke, ~~(no također gubimo „power-up“ koji smo imali ako izgubimo život)~~.

# OKRUŽENJE

Za naše okruženje (mapu) koristit ćemo mjesta iz prve misije iz originalne igre (u blizini hrama i džungla/močvara). Bitno je naznačiti kako kroz različite platforme u levelima metci prolaze.

# LIKOVI

Ovdje imamo našeg lika kao protagonista Metal Sluga.

# LIKOVI BEZ MOGUĆNOSTI UPRAVLJANJA

Ovo su, u pravilu, neprijatelji koje moramo poraziti. Rebel army vojnici i njihov zrakoplov („boss“) kojima je general glavni „negativac“ Morden, te ratne zarobljenike koje trebamo spasiti i koji nam zatim daju posebna oružja.

# MOGUĆE PRIKUPLJANJE RESURSA

Baš kao i kod napredovanja, „power-upovi“ su nam jedini oblik resursa koje možemo prikupiti, te neposredno poslije i koristiti. Skupljamo ih na način da ih nam daju ratni zatvorenici nakon što ih spasimo time što upucamo ili prerežemo uže kojim su zavezani.

# NAČIN KONTROLE U IGRI

Glavnog lika se kontrolira tipkovnicom, specifično: ~~(„AWSD“ tipke)~~ strelice se koriste za kretanje i ciljanje, ~~(„J“)~~ „A“ za skakanje (čak i kroz platforme) i ~~(„K“)~~ „S“ za pucanje. Kombinacijom tipki možemo pucati u zraku, te mijenjati smjer u kojem pucamo.

# KRAJ IGRE/SIMULACIJE, APLIKACIJE

Dođemo li na kraj ~~(drugog)~~ trećeg levela (misije) i uništimo neprijateljski zrakoplov, igra tada „završava“. Nakon toga nam se prikazuje rezultat (score) koji smo skupili kroz ta prethodna ~~(dva)~~ tri levela, te koliko nam je života ostalo i konačno vrijeme.

# OBLIKOVANJE KLASA

## Prva verzija dijagrama klasa

Bullet

FlameShot

RocketLauncher

HeavyMG

PlaneRebel

RifleRebel

Enemy

Marco

Character

PowerUp

Sprite

## Završna verzija dijagrama klasa (statičke klase – zeleno, apstraktne klase - crveno)

NoGravItem

Platform

POW

PowerUp

GoSign

Bullet

Item

Sprite

Character

Marco

Rebel

RifleRebelBullet

PlaneRebelBomb

FSBullet

HMGBullet

RLBullet

RifleRebel

PlaneRebel

Box

Postavke

## Opis klasa

### Postavke (statička klasa)

**POLJA:**

* **marco:** koristit ćemo statičku varijablu jer imamo samo jedno playera
* **rebels:** stavljamo ih u niz kako bi im lakše u game loopu mogli „pristupiti“
* **platforms:** također ih držimo u nizu za lakši setup levela
* **powerUp:** trenutni power-up (tip metaka) koji trenutno posjedujemo
* **currentPowerUp:** naziv power-upa koji posjedujemo
* **bullets:** stavljamo ih u niz kako bi im lakše u game loopu mogli „pristupiti“
* **#lives:** privatna varijabla – naš broj života
* **#ammo:** privatna varijabla - postavljamo na Infinity (hrv. beskonačno) za početak jer će obični pištolj imati toliko metaka
* **#points:** privatna varijabla - **„**score“ koji trenutno imamo
* **time:** vrijeme koje brojimo kroz igru
* **intervalId:** varijabla kojoj doslovno pridjeljujemo interval od jedne sekunde kako bi se simulirao timer

**SVOJSTVA:**

* **­get/set points:** brine se da points ne idu ispod nule
* **get/set ammo:** vraćamo se na pištolj ako nam ponestane metaka u „posebnom“ oružju
* **get/set lives:** ovo se svojstvo brine za to da se sve postavke playera i igre vrate na početno stanje pri gubitku svih života

**METODE:**

* **timeCounterStarts():** metoda koja služi za pokretanje in-game „štoperice“ koja počinje klikom na „START“ button
* **timeCounterStops():** metoda koja služi za zaustavljanje in-game „štoperice“ – zaustavlja se klikom na „STOP“ button
* **newMissionReset():** metoda koja se poziva svakim novim setupom bilo koje misije – brisanje svih neprijatelja, platformi i metaka iz njihovih listi
* **gameEndReset():** poziva se pri pobjedi i gubitku, samo postavlja nekoliko varijabli na njihovo početno stanje
* **totalPointCalculation():** izračuna završne points pri pobjedi ovisno o vremenu završetka i preostalim životima
* **removeBullet():** briše metak iz liste bullets

### Character (apstraktna klasa)

**POLJA:**

* **frame\_sets:** ovo za sad samo inicijaliziramo kao prazan objekt
* **layer:** naziv layera u Tiledu
* **visible:** odmah postavljamo na true kako ne bismo to trebali raditi svaki put pri inicijalizaciji novog objekta

**METODE:**

* **collideBox(box):** metoda koja omogućuje svim objektima koji nasljeđuju Character klasu da „prolaze kroz platforme“ (box parametar je „konkretna“ platforma o kojoj se trenutno radi)
* **switchFrameSet(direction):** s ovom metodom pojednostavljujemo mijenjanje spriteova u igri
* **currentShotSetup(b):** postavlja neka osnovna svojstva koja moramo pridijeliti metku pri pucanju – b je metak (bullet)

### Marco

**POLJA:**

* **frame\_sets:** dodjeljujemo spriteove glavnom liku
* **isShooting:** boolean vrijednost koja ovisi o tome pucamo li ili ne (važno kako ne bismo cijelo vrijeme pucali ako držimo tipku za pucanje ako to ne želimo)
* **direction:** stupnjevi za koje je okrenut lik, te gdje će pucati (0, 90, 180, 270)

**SVOJSTVA:**

* **get centerX:** dohvaćaju se koordinate sredine lika po x osi (nije nam potreban setter jer nam nije potrebno mijenjati formulu za dohvaćanje)
* **get topY:** dohvaćaju se koordinate vrha lika po y osi (nije nam potreban setter jer nam nije potrebno mijenjati formulu za dohvaćanje)
* **get bottomY:** dohvaćaju se koordinate dna lika po y osi (nije nam potreban setter jer nam nije potrebno mijenjati formulu za dohvaćanje)

**METODE:**

* **updatePosition():** metoda koja se konstanto „refresha“ u game loopu, ovdje je overrideana kako bi se moglo šetati na platformama kroz koje se prolazi, te za provjeru pada sa mape
* **start():** početna pozicija protagonista pri setupu
* **moveLeft():** metoda za micanje lijevo (prilagođena brzina)
* **moveRight():** metoda za micanje desno (prilagođena brzina)
* **aimUp():** metoda za ciljanje gore
* **aimDown():** metoda za ciljanje dolje
* **jump():** metoda za skakanje
* **loseLife():** metoda sa gubljenje života glavnog lika
* **shoot():** metoda za pucanje u kojoj se kreira objekt - „potrebni bullet“, ova metoda može biti overloadana sa posebnim power-upom za različite „načine i efekte“ pucanja
* **bulletStop():** metoda koja briše metak pri dodiru s metom (osim u slučaju flame shota)
* **knife():** metoda koja se poziva kad se „približimo“ neprijatelju, te uz pomoć koje odmah ubijemo tog samog neprijatelja
* **defaultFrameSets():** metoda koja postavlja prvotne spriteove na glavnog lika

### Rebel (apstraktna klasa)

**POLJA:**

* **x:** x koordinata neprijatelja
* **y:** y koordinata neprijatelja
* **frame\_sets:** ovo za sad samo inicijaliziramo kao prazan objekt
* **lastShotTime:** polje koje koristimo kako bi neprijatelji mogli pucati u intervalima

**SVOJSTVA:**

* **get centerX:** dohvaćaju se koordinate sredine lika po x osi (nije nam potreban setter jer nam nije potrebno mijenjati formulu za dohvaćanje)
* **get centerY:** dohvaćaju se koordinate sredine lika po y osi (nije nam potreban setter jer nam nije potrebno mijenjati formulu za dohvaćanje)

**METODE:**

* **aimLeft():** okretanje prema lijevo (direction)
* **aimRight():** okretanje prema desno (direction)
* **inititalizeShoot():** metoda koju se može overloadati, te koja omogućava kreiranja pravog projektila pri pucanju neprijatelja
* **intervalShooting():** metoda koju se može overloadati, te koja omogućava pucanje neprijatelja u korisnički zadanim intervalima
* **takeDamage():** što će se dogoditi pri dobivanju damagea, odnosno oduzimanju damagea metka od healtha neprijatelja (smrt neprijatelja, te mijenjanje spritea, o kojem se metku i samim tim damageu radi…)

### RifleRebel

**POLJA:**

* **x:** x koordinata neprijatelja
* **y:** y koordinata neprijatelja
* **frame\_sets:** dodjeljujemo spriteove neprijatelju tipa RifleRebel
* **width:** posebno označavamo širinu objekta
* **height:** posebno označavamo visinu objekta
* **direction:** stupnjevi za koje je okrenut neprijatelj, te gdje će pucati (90, 270)
* **health:** koliko damagea može ta istanca neprijatelja primiti prije „smrti“

**METODE:**

* **updatePosition():** overrideamo ovu metodu iz bazne klase i činimo ju praznom kako bi ovaj tip neprijatelja zanemario gravitaciju
* **isFacingPlayer():** ova metoda ne prima ništa, a vraća formulu kojom su dobiva informacija o tome hoće li glavni lik biti u vidnom polju ovog enemya
* **shoot():** metoda za pucanje
* **enemyPlayerInteractions():** ponašanje neprijatelja pri interakciji sa playerom, npr. mijenjanje spriteova pri uočavanju samog playera, gdje će neprijatelj biti okrenut itd.

### PlaneRebel

**POLJA:**

* **frame\_sets: :** dodjeljujemo spriteove neprijatelju tipa PlaneRebel
* **width:** posebno označavamo širinu objekta
* **height:** posebno označavamo visinu objekta
* **direction:** stupnjevi za koje je okrenut neprijatelj, te gdje će pucati, što je u ovom slučaju uvijek 180 stupnjeva
* **health:** koliko damagea može ta istanca neprijatelja primiti prije „uništenja“, odnosno „umiranja“
* **flyingLeft:** s obzirom da ne možemo mijenjat direction, na ovaj ćemo onda način provjeravati kreće li se avion lijevo ili desno (bool vrijednost)

**METODE:**

* **updatePosition():** overrideamo ovu metodu iz bazne klase i „namještamo“ ju na način da gravitacija ne utječe na ovog enemya, međutim i tako da se još uvijek može kretati lijevo/desno
* **updateAnimation():** overrideano kako bi se promijenio način rada, ovo se učinilo zbog nemogućnosti korištenja direction varijable
* **moveLeft():** metoda za „letenje“ lijevo (prilagođena brzina)
* **moveRight():** metoda za „letenje“ desno (prilagođena brzina)
* **flying():** metoda u kojoj obrađujemo logiku letenja neprijatelja
* **shoot():** metoda za pucanje

### Platform

**POLJA:**

* **okvir:** ovo polje odmah postavljamo na false kako ne bismo imali crni okvir oko svake platforme
* **mozeVan:** ovim poljem (bool) označavamo da ovaj objekt može ići van granica mape (canvasa)

**METODE:**

* **platformCreation():** u ovoj metodi se nalazi sva logika za kreiranje platformi (kroz koje se može proći „od dolje“) koja se poziva pri „postavljanju“ svakog levela

### NoGravItem

**POLJA:**

* **visible:** odmah postavljamo na true kako se poslije ne bismo „trebali misliti“ zašto nešto nije vidljivo, iako je inicijalizirano

**METODE:**

* **updatePosition():** overrideamo ovu metodu iz bazne klase i činimo ju praznom kako bi ovaj tip objekta zanemario gravitaciju

### Bullet

**POLJA:**

* **#distance:** privatna varijabla – daljina koju metak prelazi
* **visible:** odmah postavljamo na true kako se poslije ne bismo „trebali misliti“ zašto nešto nije vidljivo, iako je inicijalizirano
* **move:** bool provjera – kreće li se metak
* **\_collidedPlatform:** vraća string – s koje je strane metak pogodio „klasični“ tile, odnosno platformu
* **mozeVan:** ovim poljem (bool) označavamo da ovaj objekt može ići van granica mape (canvasa)

**SVOJSTVA:**

* **get/set distance:** brine se o tome da se projektil koji je izašao van granica mape, odnosno canvasa izbriše
* **get/set collidedPlatform:** svojstvo kojim se briše metak ako je pogodio „klasičnu“ platformu

**METODE:**

* **updatePosition():** logika kretanja projektila – u kojem smjeru, koliko brzo itd. (cijelo vrijeme se poziva)
* **stop():** zaustavljanje metka koje pozivamo pri određenom uvjetu
* **shotEnemy():** metoda u kojoj provjeravamo je li metak pogodio neprijatelja, te ako jest koji su dalje koraci npr. primanje određenog damagea
* **shotPlayer():** metoda u kojoj provjeravamo je li metak pogodio „našeg“ playera, te ako jest koji su dalje koraci npr. gubljenje jednog lifea

### HMGBullet

**POLJA:**

* **damage:** postavlja se damage za svaki metak tipa Heavy Machine Gun
* **bulletSpeed:** postavlja se brzina za svaki metak tipa Heavy Machine Gun

### FSBullet

**POLJA:**

* **#distance:** privatna varijabla - daljina koju Flame Shot metak prelazi
* **damage:** postavlja se damage za svaki metak tipa Flame Shot
* **bulletSpeed:** postavlja se brzina za svaki metak tipa Flame Shot

**SVOJSTVA:**

* **get/set distance:** brine se o tome da se projektil koji je prešao određenu daljinu izbriše

### RLBullet

**POLJA:**

* **damage:** postavlja se damage za svaki metak, odnosno raketu tipa Rocket Launcher
* **bulletSpeed:** postavlja se brzina za svaki metak, odnosno raketu tipa Rocket Launcher

### RifleRebelBullet

**POLJA:**

* **bulletSpeed:** postavlja se brzina za svaki metak koji je ispucao enemy tipa RifleRebel

### PlaneRebelBullet

**POLJA:**

* **bulletSpeed:** postavlja se brzina za svaki metak, odnosno bombu koji je ispucao enemy tipa PlaneRebel

### POW

**METODE:**

* **freePOW():** metoda koja se poziva kad se oslobodi Prisoner Of War (hrv. ratni zarobljenik), npr. dodavanje pointova na konačni rezultat

### PowerUp

**POLJA:**

* **visible:** odmah postavljamo na false jer će tek nakon oslobađanja P.O.W.a biti vidljiv
* **ammo:** postavlja se ammo koju taj power-up „nosi“

### GoSign

**METODE:**

* **nextMissionAdvancementCheck():** prvo se provjerava diramo li uopće „cilj“, te ako jesmo, među ostalim, „postavljamo“ idući level