

SUPER MARIO: PYTHON

TEHNIČKA DOKUMENTACIJA



Autor: Dino Gržinić

Mentor: prof. Goran Boneta

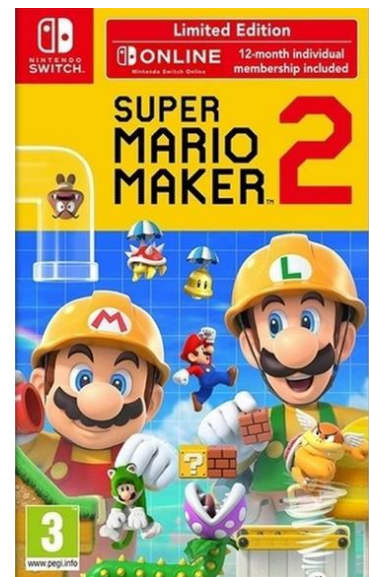
SADRŽAJ

1. UVOD	2
1.1 ŠTO JE SUPER MARIO?	2
1.2 IDEJA PROJEKTA	2
2. OPIS IZRADE PROGRAMA	3
2.1 PYGAME	3
2.2 SPRITES	3
2.3 POSTAVLJANJE LIKA MARIA	4
2.4 ZVUČNI EFEKTI	5
2.5 POSTAVLJANJE NPC-A	5
2.6 INTERAKCIJA LIKOVA	6
2.7 OPENING CINEMATIC	6
3. OBJAŠNJENJE KODA	7
3.1 VIDEOPLAYER.py	7
3.2 SUPER MARIO.py	8
4. UPUTE ZA KORIŠTENJE	12
4.1 CUSTOM MAPA	13
4.2 PRAVILA IGRE	13
4.3 KONTROLE	13
4.4 TEHNIČKI PODATCI	13

1.1. ŠTO JE SUPER MARIO?

A screenshot from the video game Super Mario Bros. The scene is set against a black background. In the top left corner, the text "SCORE 000300" is displayed in white. In the top center, "HIGH SCORE 750000" is shown. In the top right, "TIME 031" is visible next to a speaker icon. Below the score, the text "LEVEL: 1" is displayed. The game area features several horizontal platforms made of red brick with a white 'X' pattern. Two tall, thin ladders made of blue rungs are positioned in the upper left. A Goomba enemy, a small brown creature with a white belly, is on the top platform. A small blue pipe with a flame on top is on the bottom left. A small blue box with the number "100" is on the bottom right. A small blue character, likely a Koopa, is on the bottom right platform. There are several green coins scattered across the platforms.

Miyamoto. Prvi put se pojavljuje na arkadnim strojevima 1981. u igri Donkey Kong. Od tada se pojavljuje u više od 200 igara te raznim animiranim filmovima i stripovima. U svojoj popularnosti je nadmašio čak i Disneyjevog Mickey Mouse-a. Od njegovog



1.2 IDEJA PROJEKTA

A side-by-side comparison of a Koopa from the Mario franchise. On the left is a pixelated, 8-bit style Koopa, and on the right is a smooth, 3D rendered Koopa. Both are brown with a tan face and a black mustache. The 3D version has a more expressive, angry facial expression with large, white eyes and a small, downturned mouth.

Slika 1.4: Mario iz 1985. i Mario sada

2. OPIS IZRADE PROGRAMA

2.1 PYGAME

Kada sam se odlučio za programiranje video igre, znao sam da će mi Pygame biti nužna Python knjižnica. No u početku nisam znao kako Pygame radi. Stoga sam učio i istraživao o Pygame-u preko raznih internetskih stranica. Kada sam shvatio osnove, počeo sam vježbati na osnovnom programu, na primjer, kako nacrtati pravokutnik i kako se može micati.

2.2 SPRITES

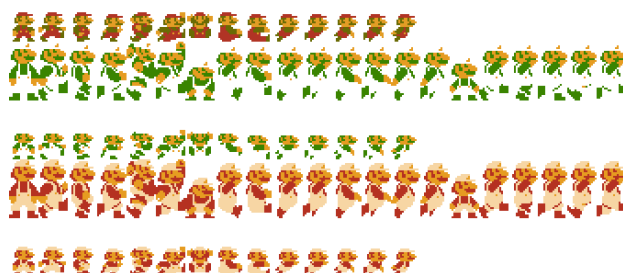
Kada sam naučio kako micati pravokutnik unutar prozora, nisam htio da moja verzija Super Maria ne bude par mičućih pravokutnika na ekranu – trebali su mi pravi likovi. *Sprite* je slika koja se pojavljuje unutar veće grafičke scene (prozora igre). Prvo mi je trebala pozadina. Za nju sam iskoristio sliku Super Mario nivoa s interneta. Zatim je bilo vrijeme da ispunim pozadinu likovima. Na internetu sam pronašao izgled svih originalnih likova korištenih u Super Mario Bros igri poput Goombe i Maria.



Slika 2.1: Pozadina igre

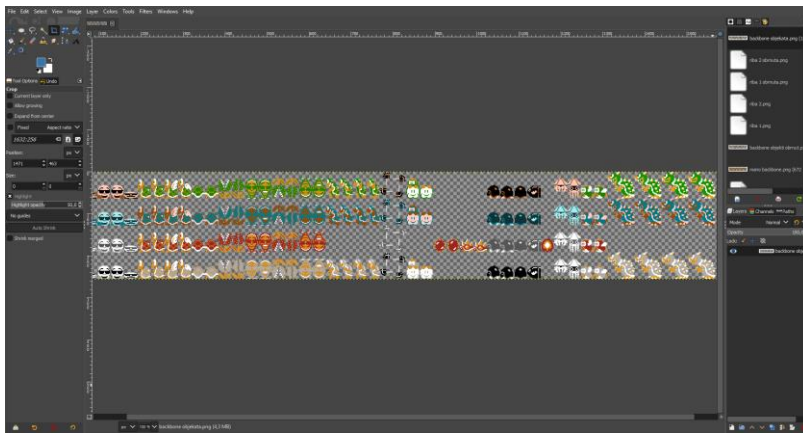


Slika 2.2: Svi *sprite*-ovi korišteni u Super Mario Bros igri



2.3 POSTAVLJANJE LIKA MARIA

Prvo sam samo *sprite* Maria trebao 'izrezati' od ostatka slike. Kako bih to učinio, koristio sam se besplatnim softverom za manipulaciju slika GIMP. Prvo sam

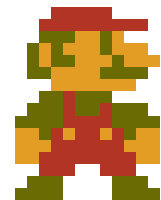


Slika 2.2: Program GIMP

dodao Maria, no nije izgledalo realistično pošto ne izgleda kao da se Mario kreće, već samo slika koja lebdi unutar prozora. Trebao sam ograničiti njegovo kretanje tako da ne može 'otići' preko ruba prozora

te da se može

kretati po vodoravnoj liniji. No i dalje ne izgleda realistično pošto nema animacije hodanja. Kako bih to napravio, na ekranu se treba na određeno vrijeme prikazati određeni *sprite* Maria kako bi se dobio dojam hodanja. Sve potrebne *spriteove* sam izrezao iz slike 2.2 te sam ih spremio kao pojedinačne slike. Pošto se Mario

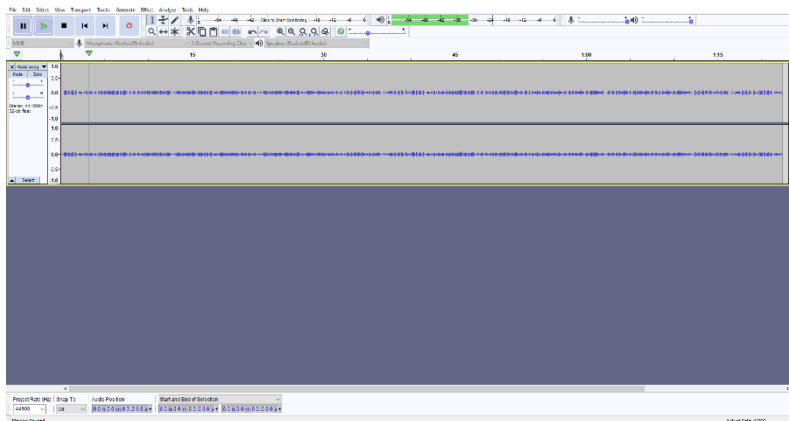


Slika 2.3: Jedan od Mario *spriteova*

kreće i lijevo i desno, ne mogu koristiti iste *spriteove* za oba kretanja ako želim zadržati realističnost. Zbog toga sam napravio dvije liste sa slikama (jedna se desnim hodom, druga s lijevom). Ovisno koju tipku igrač pritisne (A-lijevo, D-desno), koristit će se određena lista. Dokle god je tipka pritisnuta, vrijednost tzv. 'brojača hoda' će se povećavati te će se pomoću *for* petlje prikazivati jedan za drugim *spriteovi* iz liste. Koliko dugo će biti prikazan pojedini *sprite* ovisi o FPS-u kojega sam postavio na 60 jer igra nema veliki utjecaj na CPU računala. Idući korak je bio omogućiti Mariu da skače. Puta skoka dok je Mario u pokretu treba izgledati kao parabola okrenuta prema dolje. Može ju se napraviti u koordinatnom sustavu koji je unutar prozora igre zapravo širina (x os) i visina (y os) samog prozora. Naknadno sam dodao posebne *spriteove* Maria kako skače radi realističnosti.

2.4 ZVUČNI EFEKTI

++Također sam kod igrača htio zabaviti ne samo osjetilo vida, već i sluha. Kako bih to postigao, koristio sam se zvučnim efektima iz Super Maria (pozadinski zvuk igre, zvuk skoka, zvuk kraja igre, zvuk pucanja). Za puštanje zvuka koristio sam se s *pygame.mixer.music* funkcijom. No kasnije sam otkrio da ona može puštati samo jednu po jednu .mp3 datoteku. Stoga sam tu funkciju koristio samo za puštanje pozadinskog zvuka igre te sam stavio da se konstantno ponavlja nakon završetka. Ostale zvučne datoteke sam trebao obraditi u programu za



Slika 2.4: Program Audacity

manipulaciju zvukom, *Audacity*. Prvo sam trebao datoteke 'izrezati' kako bi se čuo samo dio koji želim čuti te sam ih zatim *exportao* kao .wav datoteke kako bih mogao više zvučnih efekata puštati istovremeno pomoću funkcije

pygame.mixer.Sound. Zatim sam te zvukove trebao povezati s određenim radnjama poput pritiska tipke (W-skok => puštanje zvučnog efekta skoka).

2.5 POSTAVLJANJE NPC-A

NPC je skraćenica za *Non-Player Character*. To je lik čije akcije nisu kontrolirane od strane igrača, već od njegovog koda. NPC-jevi su se također koristili u *Super Mario Bros* igri, poput gljive.



Slika 2.6: Sprite

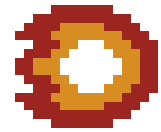
Postupak je sličan izradi Maria. Izrezao sam potrebne slike, napravio da se izmjenjuju kako bi se činilo da hoda, u kodu sam napisao obrazac njenog kretanja. No još uvijek likovi nisu mogli interagirati jedan s drugim. Naknadno sam ubacio i drugi NPC – leteću ribu. Ona proljeće iznad gljive i Maria, svako toliko. Ubacio sam ju radi ograničavanja Mariovog kretanja, no kao i izvor dodatnih bodova.



Slika 2.5: Sprite gljive

2.6 INTERAKCIJA LIKOVA

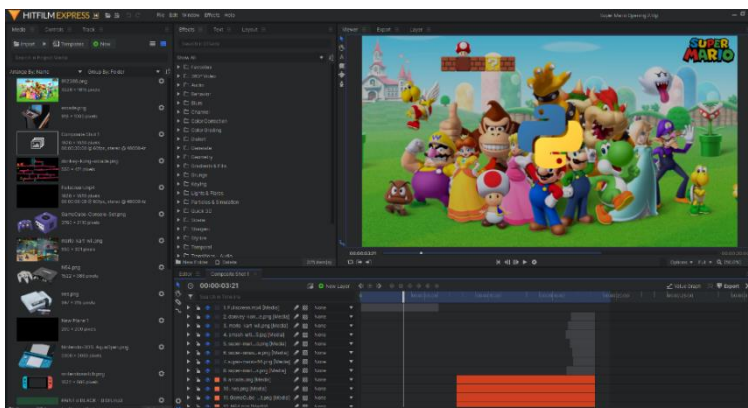
Prvo sam Mariu htio omogućiti da može pucati gljivu *fireballom*. Izrezao sam *fireball sprite* te sam omogućio da se s Mariovih koordinata ispali *fireball* pritiskom tipke u smjeru u kojem Mario gleda. Ako se koordinate *fireballa* i NPC-a poklapaju, *fireball* će nestati te će program zabilježiti pogodak. Ako igrač pogodi gljivu, dobit će 1 bod. Ako pogodi ribu, koja se rjeđe pojavljuje na ekranu i brza je te treba skočiti da ju se pogodi, onda dobiva 5 bodova. No kako igrač ne bi konstantno pritiskao paljbu, ograničio sam njegovo pucanje tako da može ispucati sljedeći *fireball* tek kad se prethodni sudari s NPC-om ili otiđe s ekrana. To znači da što je Mario bliže NPC-u, to će moći ostvariti više pogodaka, no riskirat će sudar s NPC-jem. Tako će ujedno i igra završiti, ako se Mario i NPC dotaknu. Taj trenutak se može identificirati tako da se likovima nacrtaju *hitboxi*. Korištenje okvira samih slika kao *hitboxa* ne bi bilo efikasno jer likovi ne zauzimaju cijelu površinu slike, već treba ručno odrediti položaj, veličinu i oblik *hitboxa*. Kada se bridovi *hitboxa* NPC-a i Maria dotaknu, to će reći programu da je došlo do sudara te da izvrši kod namijenjen za tu situaciju.



Slika 2.7: *Sprite fireball-a*

2.7 OPENING CINEMATIC

Opening cinematic je film od par sekundi koji se pojavljuje prije početka igre koji služi da pripremi igrača na ono što slijedi. Stoga sam se i ja okušao u izrađivanju jedne. Za to mi je trebao program za uređivanje videozapisa. Za to sam koristio besplatni program *Hitfilm Express*. Kad sam napravio cinematic, trebao sam



Slika 2.8: Program Hitfilm Express

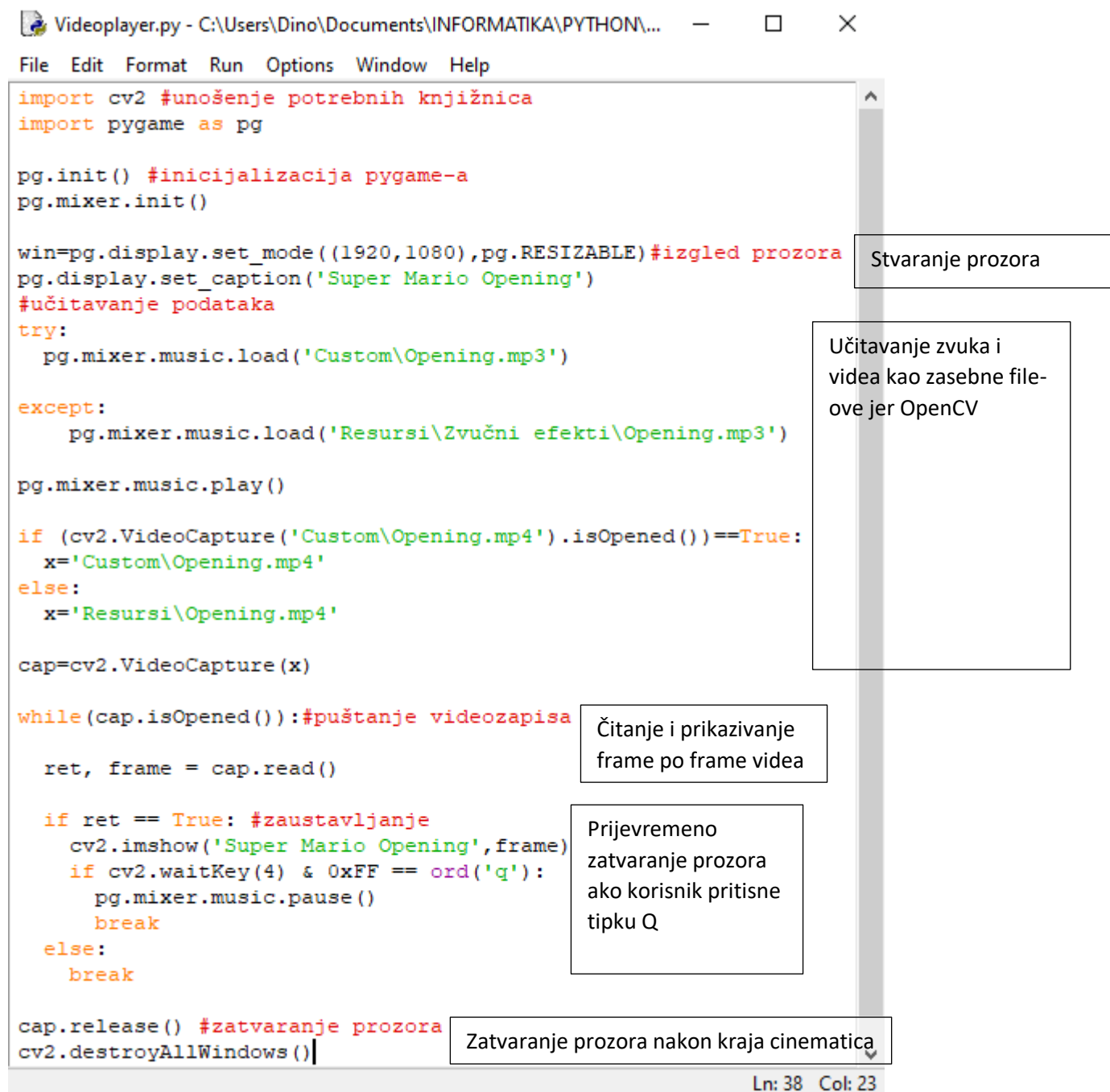
Također sam dodao opciju preskakanja cinemática da ne dosadi igraču nakon svakog ulazanja u igru. Kako bi se igra pokrenula poslije cinemática, importao sam file s video reprodukcijom u glavni.

shvatiti kako reproducirati video u Pythonu. Za to su mi trebale knjižnice OpenCV, Numpy i Pygame. Napravio sam zaseban Python file za reprodukciju videa jer se izvodi u zasebnom prozoru te se nakon kraja videa zatvara i igra počinje.

3. OPIS KODA

3.1 VIDEOPLAYER.py

Videoplayer.py je odvojeni Python file od ostatka koda koji služi za reprodukciju opening cinematica u zasebnom fullscreen prozoru.



The screenshot shows a Python IDE window titled "Videoplayer.py - C:\Users\Dino\Documents\INFORMATIKA\PYTHON\...". The code is as follows:

```
import cv2 #unošenje potrebnih knjižnica
import pygame as pg

pg.init() #inicijalizacija pygame-a
pg.mixer.init()

win=pg.display.set_mode((1920,1080),pg.RESIZABLE)#izgled prozora
pg.display.set_caption('Super Mario Opening')
#učitavanje podataka
try:
    pg.mixer.music.load('Custom\Opening.mp3')
except:
    pg.mixer.music.load('Resursi\Zvučni efekti\Opening.mp3')
pg.mixer.music.play()

if (cv2.VideoCapture('Custom\Opening.mp4').isOpened())==True:
    x='Custom\Opening.mp4'
else:
    x='Resursi\Opening.mp4'
cap=cv2.VideoCapture(x)

while(cap.isOpened()):#puštanje videozapisa
    ret, frame = cap.read()

    if ret == True: #zaustavljanje
        cv2.imshow('Super Mario Opening',frame)
        if cv2.waitKey(4) & 0xFF == ord('q'):
            pg.mixer.music.pause()
            break
        else:
            break

cap.release() #zatvaranje prozora
cv2.destroyAllWindows()
```

Annotations on the right side of the code:

- Stvaranje prozora**: Points to the `pg.display.set_mode` and `pg.display.set_caption` lines.
- Učitavanje zvuka i videa kao zasebne file-ove jer OpenCV**: Points to the `try/except` block for loading audio and video files.
- Čitanje i prikazivanje frame po frame videa**: Points to the `while` loop and `cap.read()` line.
- Prijevremeno zatvaranje prozora ako korisnik pritisne tipku Q**: Points to the `if ret == True` block and the `break` statement.
- Zatvaranje prozora nakon kraja cinematica**: Points to the `cap.release()` and `cv2.destroyAllWindows()` lines.

At the bottom right, the status bar shows "Ln: 38 Col: 23".

Slika 3.1: Kod za Videoplayer.py

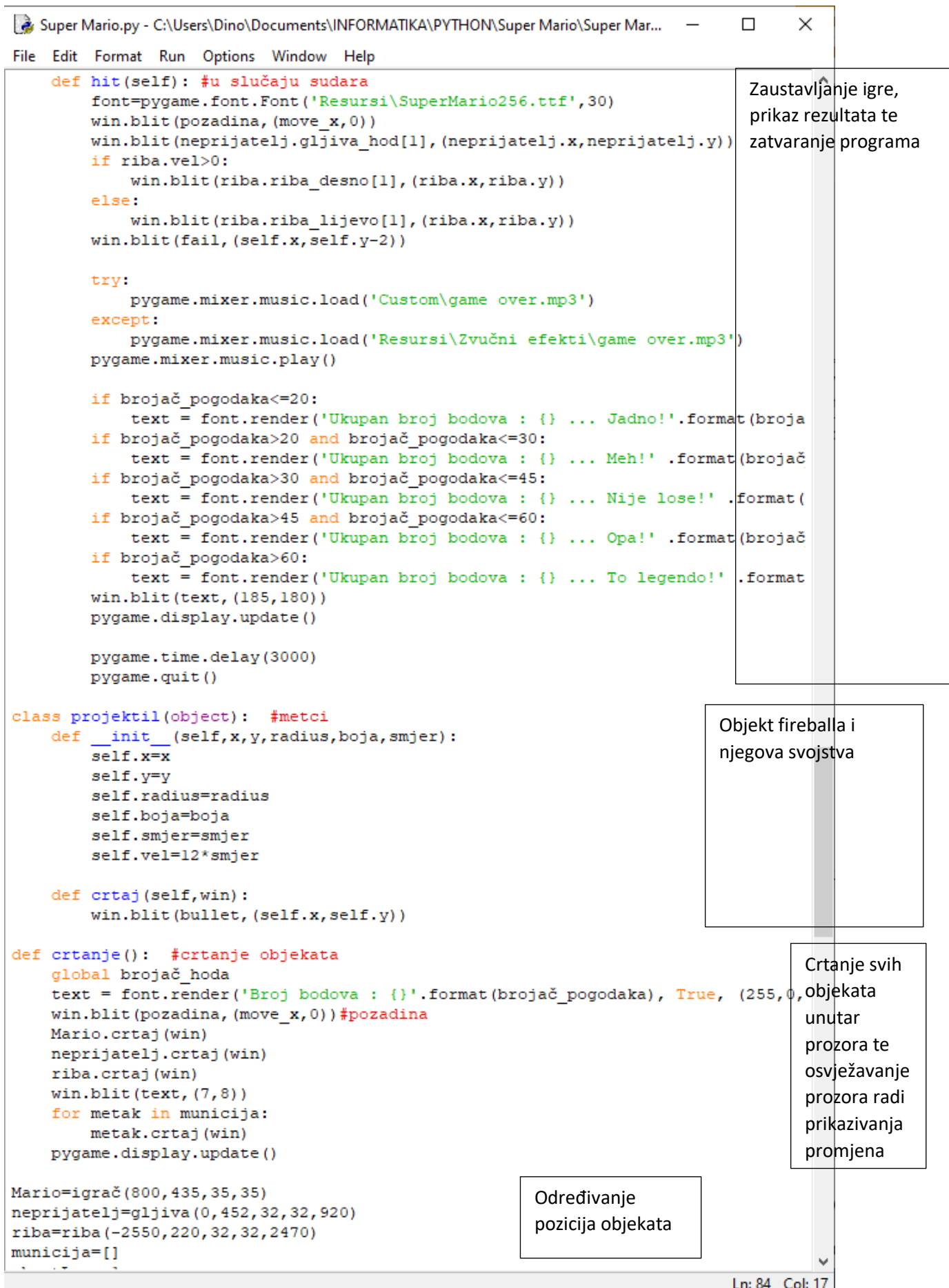
3.2 SUPER MARIO.py



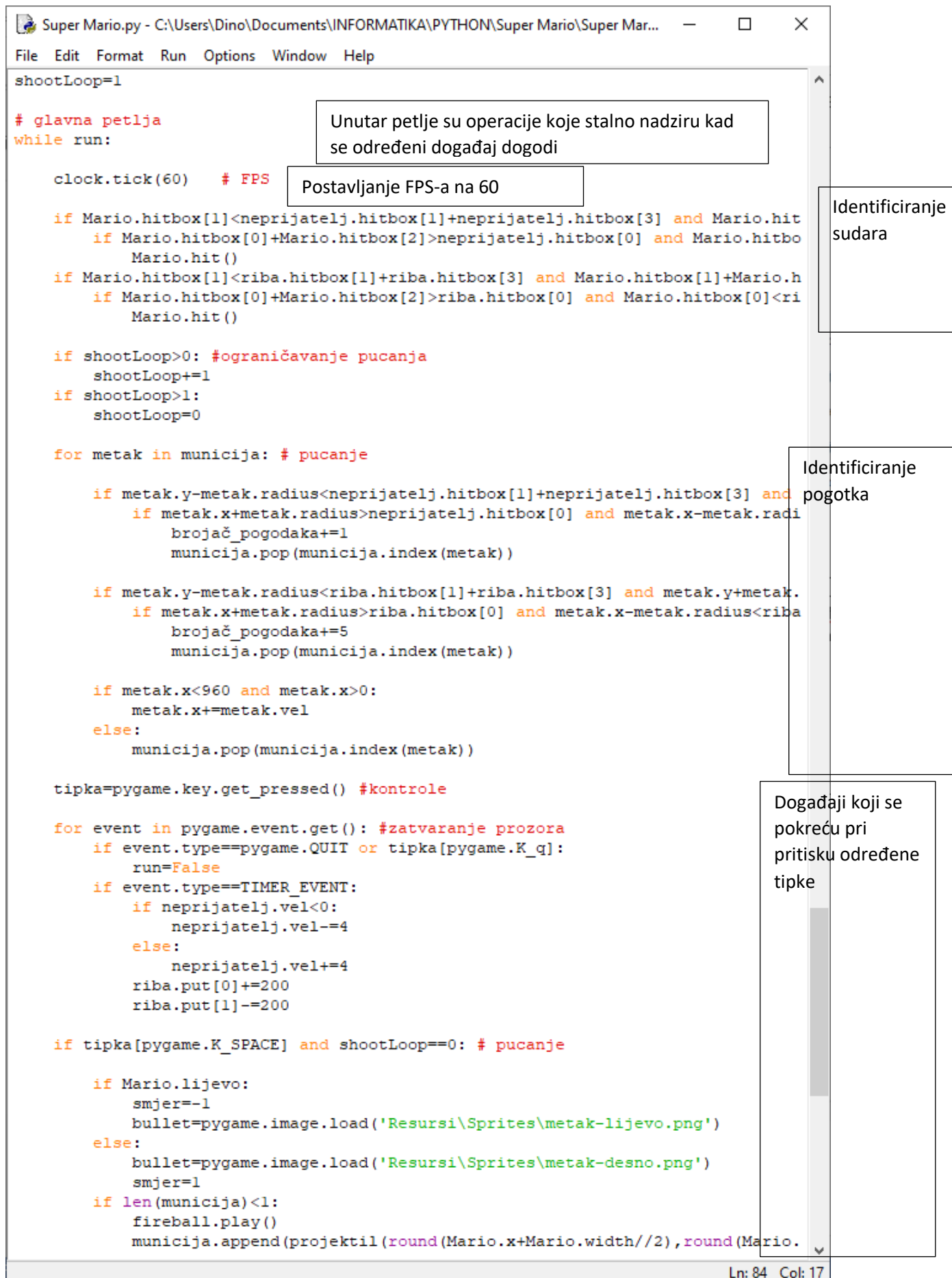
Slika 3.2: Super Mario.py 1. dio



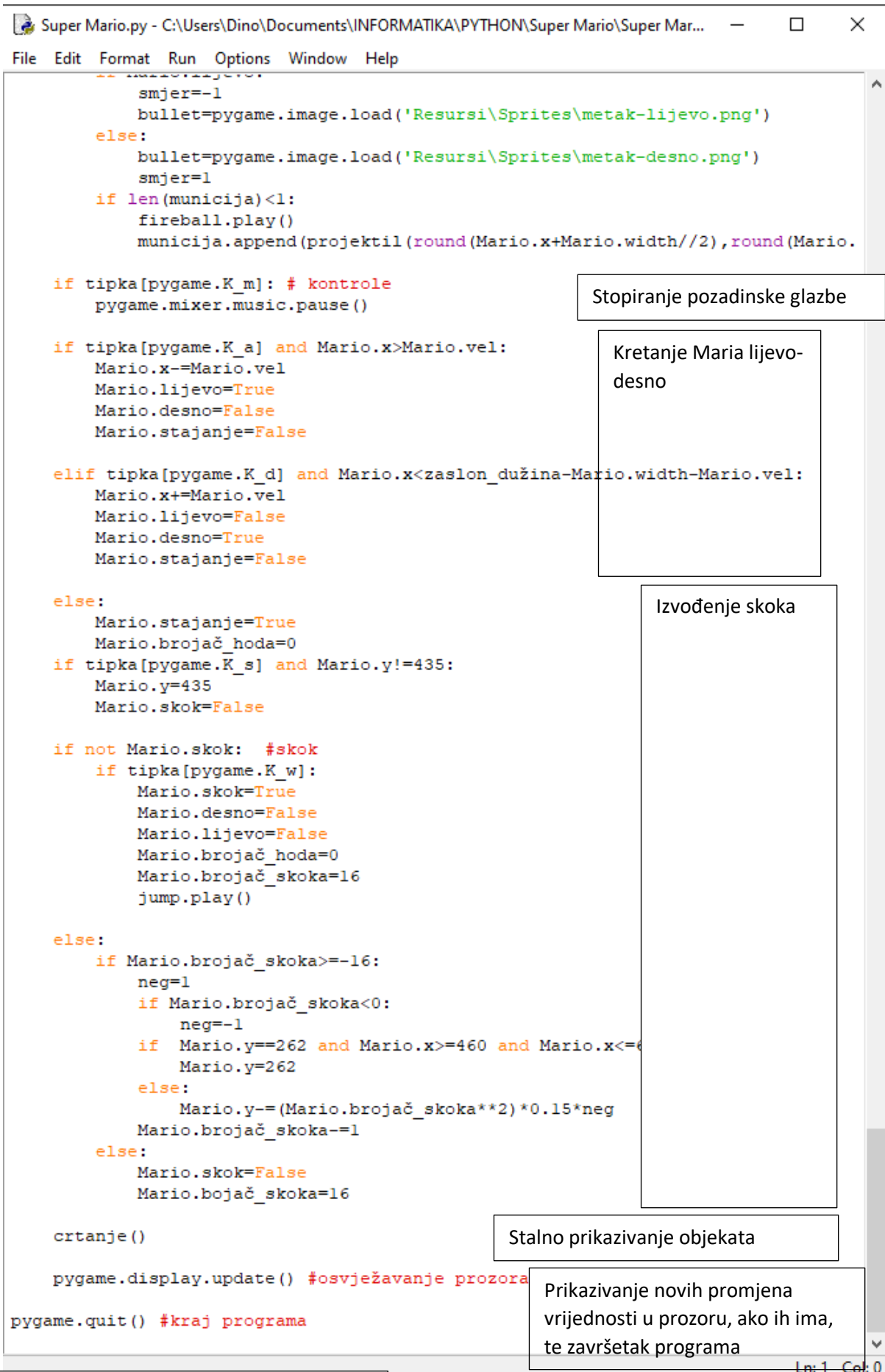
Slika 3.3: Super Mario.py 2. dio



Slika 3.4: Super Mario.py 3. dio



Slika 3.5: Super Mario.py 4. dio



Slika 3.6: Super Mario.py 5. dio

4. UPUTE ZA KORIŠTENJE

4.1 CUSTOM MAPA

Ako bi igrač htio promijeniti neke resurse koje sam koristio, to može učiniti u Custom mapi. To je prazna mapa u koju korisnik stavi file s određenim nazivom i formatom te će program umjesto zadanog resursa koristiti korisnikov. Mogu se promijeniti: cinematic, Mario, NPC-jevi, pozadina, fireball, zvučni efekti.

4.2 PRAVILA IGRE

Kao Mario, tvoj zadatak je da vatrenim loptama što više puta pogodiš gljivu ili leteću ribu. Radi sprječavanja konstantnog pucanja, novi fireball se neće moći ispaliti dokle god prethodni ne dođe do ruba prozora ili ne pogodi gljivu. To znači, što si bliže gljivi, više pogodaka, ali i veći rizik od sudara. Leteća riba, nakon određenog vremena, prolazi preko ekrana. Pogodak gljive vrijedi 1 bod, dok pogodak leteće ribe vrijedi 5 bodova. Imaš izbor, nastaviti gađati gljivu ili riskirati gađanjem leteće ribe. No ako te gljiva ili leteća riba dotaknu, igra završava. Kako igra prolazi, gljiva se kreće sve brže i brže te se riba sve češće pojavljuje.

4.3 KONTROLE

W – skok

A – lijevo

D – desno

S – brz povratak na tlo iz skoka

SPACE - pucanje vatrenih lopti

M – isključivanje glazbe

Q – preskoči cutscenu/izađi iz igre

4.4 TEHNIČKI PODATCI

Veličina cijele igre: 6,57 MB

Veličina resursa: 6,55 MB

Veličina samog koda: 14,9 KB

Broj linija koda: 448

Korišćenje CPU-a: 0,45 GHz

Korišćenje RAM-a: 40 MB

Ne preporuča se pokretati na starijim laptopima.