LEGEND OF ZELDA

Realizacija igre Legend of Zelda na specijalnoj grafičkoj

Vesna Isić RA41/2015

Milorad Marković RA162/2015

Jelena Boroja RA22/2015

Ovaj projekat je rađen u okviru predmeta LPRS 2. Dizajniran je za FPGA platformu – Microblaze procesor. Korišćen je postojeći HW od Super Mario-a uz potrebne izmene, a Gameplay je prilagođen gorepomenutoj igrici.

29.05.2018

1 Sadržaj

1. Opis igre i Gameplay

Igra *Legend of Zelda* je video igra dizajnirana i publikovana od strane Nintenda 1987. godine. Igra je akciono-avanturističkog karaktera. Priča se dešava u imaginarnom svetu *Hyrule* i glavni protagonista igre jeste *Link.* Njegov zadak je da krećući se kroz svet pobedi neprijatelje I sakupi 8 delova takozvanog *Triforce of Wisdom* da bi spasao princezu *Zeldu*, koju je oteo *Ganon*.

Na samom početku igre, *Link* se pojavljuje samo sa štitom i iznad njega se vidi ulaz u pećinu. Nakon ulaska u pećinu, od ljubaznog starca dobija mač sa kojim može da se bori protiv neprijatelja. Realizovana je kompletna originalna *Overworld* mapa. Pritiskom na tastere realizuje se kretanje Linka u sva četiri smera. Mapa se iscrtava deo po deo (po frejmovima[1]). Kada *Link* dođe do ivica frejma, prelazi na naredni i na taj način obilazi čitavu mapu. Takođe, realizovana je detekcija prepreka, te tako *Link* i neprijatelji, koji se pojavljuju na određenim frejmovima, mogu da se kreću samo po dozvoljenim oblastima. Pritiskom odgovarajućeg tastera pojavljuje se mač u pravcu u kom je *Link* okrenut. Ukoliko se protivnik dodirne mačem – on umire. Ako mač prethodno nije pokupljen *Link* neće imati čime da izvrši napad. Ukoliko se *Link* primakne ivicama frejma sa korespodentnih strana (leva ivica – *Link* okrenut nalevo) napad će biti onemogućen.

Pored kretanja i interakcije sa okolinom, obuhvaćeno je i formiranje zaglavlja u kome se nalaze informacije o *Linku*.

[1] Frejm (eng. *frame*) – trenutni ekran koji je vidljiv igraču. Mapa se sastoji iz 8x16 frejmova

2. OPIS RADA

2.1 Image Processing

Na samom početku rada osnovni problem je bio iscrtavanje mape. Glavni zadatak *Image Processing Python* programa je da izračuna matrice indeksirane boje korišćenih sprajtova[2] - a u slučaju mape, korišćenih *tile*[3]-ova koji grade mapu, odnosno njene frejmove. Ove matrice se koriste tokom generisanja HDL koda za iscrtavanje *sprite*-ova na ekran. Generisanje HDL koda je odrađeno u *getHDLReadable* projektu. Korišćen je ovakav način generisanja slike da bi se uštedilo što više memorije.

Matrice se računaju pomoću *online* dostupne slike koje sadrže sve *sprite*-ove korišćene u igri i podeljene u grupe (*Enemies*, *Link*, *Map Tiles*, itd.)

Za sve slike je bila potrebna priprema. Ona obuhvata podešavanje dimenzija slike i uklanjanje razmaka između *sprite*-ova sadržanih unutar te slike tako da oni mogu biti pročitani.

U slučaju mape, svi *tile*-ovi se, nakon provere da li se nalaze u bazi *tile*-ova dobijenoj iz dostupne slike, koriste da se indeksira matrica (s tim da svaki *tile* ima svoj odgovarajući indeks).

Sami *tile*-ovi i *sprite*-ovi se sastoje od matrica koje indeksiraju njihove boje. Prvo se prolazi kroz sliku svih trenutno obrađivanih *sprite*-ova i pamte se sve boje koje se nalaze u njoj. Taj niz boja se potom koristi u prolazu kroz zasebne *sprite*-ove da se izgradi matrica boja koja će biti iskorišćena tokom iscrtavanja *sprite*-ova na ekran.

Unutar *ImageProcessing* dela projekta je takodje obrađena sva konverzija boja u slučaju oštećenih početnih slika (prvobitna boja slike nije bila odgovarajuća originalnim bojama igre), kao i iscrtavanja trenutno obrađivanih *sprite*-ova i slika korišćenjem *OpenCV* biblioteke.

[2] Sprajt (eng. *sprite*) – sličiča veličine 16x16 piksela
[3] *Tile* eng. – Sprajt korišćen za izradu mape i njene frejmove

2.2 VHDL readable

U prethodnom poglavlju opisanim programom se dobijaju nizovi koje je potrebno napisati u obliku koji se koristi u vdhl kodu. Radi razumevanja funkcionalnosti ovog programa, potrebno je razjasniti realizaciju hardvera.

Fajl *ram.vhd* predstavlja sadržaj memorije. Na početku se nalaze palete boja koje se koriste za razne sprajtove. Nakon paleta, sledi niz sprajtova. Oni obuhvataju *tile*-ove *overworld* mape, sličice protivnika, Linka, itd. Jedno polje sprajta (32-bitna vrednost) sadrži indekse boja četiri uzastopna piksela. Svako polje ima svoju adresu. Sprajtovi su sličice od po 16x16 piksela, grupisani u delove od po 4, te svaka pojedinačna sličica zauzima 64 adrese. Prilikom iscrtavanja *sprite*-ova, hardver prolazi kroz ove adrese i iščitava indekse (adrese) boja odgovarajućih piksela i iscrtava ih. Samo adresiranje sprajta se dešava na dva načina: pomoću registara ili čitanjem mape, koja se nalazi na kraju datoteke *ram.vhd.*

Program *getVHDLReadable* sadrži funkcije koje generišu odgovarajući oblik parova adresa i polja, kako se duge sekvence ovih parova ne bi kucale ručno. U okviru funkcija, vrši se računanje adresa i mapiranje sprajtova na odgovarajuće boje, odnosno mapiranje same mape na sprajtove. Napomena u vezi mapiranja *overworld* mape na matricu *tile*-ova: matrica sadrži tri bloka sličica koje se podudaraju, tj. jedina razlika je boja. Funkcijom se druga dva bloka matrice premapiraju na prvi deo, kako bi se uštedilo na memorijskom prostoru. U okviru softverske realizacije je obezbeđena promena palete, tako da se održava podudarnost sa originalnom mapom.

Pored nizova namenjenih za *ram.vhd* datoteku, u projektu se nalazi i funkcija koja generiše *overworld* mapu u matričnom obliku namenjenu za C programski jezik. *Overworld* je matrica *frame*-ova predstavljena u obliku niza, a *frame*-ovi su matrice tile-ova (tj. njihovih adresa u okviru ram.vhd), takođe predstavljeni u formi niza. Pored *overworld*-a, za softverski deo zadatka se generiše i niz adresa za sprajtove od interesa, poput, na primer, sprajtova slova.

2.3 Battle city

U datoteci *battle_city.c* nalazi se glavni deo programa. Prilikom pokretanja igre podešavaju se inicijalne vrednosti *Linka* i iscrtavaju se prvi frejm i info-bar. Čitav program nalazi se u beskonačnoj petlji kako bi se *Gameplay* omogućio.

Učitavanje frejma: u zavisnosti od toga sa koje strane *Link* ulazi na novi frejm – učitava se odgovarajući deo mape upisivanjem pomenutog u memoriju. Odrađeno je i ažuriranje boja, te tako u zavisnosti od položaja frejma u mapi menja se paleta. U ovom delu se proverava i da li na frejmu postoje neprijatelji i ukoliko postoje oni se inicijalizuju.

Zaglavlje frejma: na zaglavlju frejma nalazi se mini mapa koja pokazuje trenutnu poziciju na kom frejmu u mapi se *Link* nalazi. Desno od mini mape pojavljuju se *item*-i kako ih *Link* sakupi i u desnom uglu nalazi se preostali *Health*.

Pećina: prilikom nailaska na vrata, ulazi se u pećinu. Ukoliko je reč o ulazu na inicijalnom frejmu, na ekranu se ispisuje tekst, pojavljuju se deda, mač i animacija vatre. Potrebno je da *Link* priđe maču kako bi ga pokupio.

***Linkovo* kretanje:** pritiskom tastera na E2LP ploči dešava se kretanje *Linka*. Podržana je animacija kretanja, odnosno promena *sprite*-ova ukoliko se duže drži taster da bi se dobila iluzija hodanja. Na početku *Link* je bez mača i nije mu omogućen napad. Da bi ga dobio potrebno je da ode u pećinu. Napad mačem onemogućen je na ivicama frejmova jer bi u protivnom mač izlazio iz okvira frejma, te tako se ne bi dobio osećaj da se igra dešava unutar frejma. U ovom delu je takođe urađeno da se, ukoliko se *Link* nađe na ivicama frejma, pozove funkcija za učitavanje korespodentnog dela mape. Kada je mač pokupljen, on postaje aktivan i pritiskom srednjeg tastera na ploči aktivira se napad. Mač se pojavljuje u smeru u kom je *Link* okrenut. Ukoliko dođe do preklapanja vrha *sprite*-a mača i neprijatelja – neprijatelj nestaje. Ubijanjem neprijatelja u zaglavlju frejma se pojavljuje neki od *item*-a (bomba/novčić). *Linkovo* kretanje je omogućeno samo po dozvoljenim oblastima, odnosno – postoji funkcija koja proverava da li se njegova pozicija poklapa sa *tile*-ovima mape na kojima je logički dozvoljeno kretanje (pesak, zemlja, itd.).

Kretanje neprijatelja: kod učitavanja frejma inicijalizuju se neprijatelji, ukoliko postoje na tom frejmu. Poredi se indeks na kom se nalazi frejm i u zavisnosti od njega podešavaju se inicijalne pozicije neprijatelja. Od neprijatelja podržani su *Octorozi* i duhovi. Njihovo kretanje je određeno na pseudo-slučajan način. Zbog nemogućnosti korišćenja *rand()* funkcije, napisana je funkcija koja na neki način vraća slučajan broj i u zavisnosti od njega generiše se novi smer u kom će se kretati neprijatelji. Kao i kod kretanja *Linka* omogućena je detekcija prepreka. Neprijatelji nestaju kada se dodirnu mačem.