|  |  |  |
| --- | --- | --- |
|  | УНИВЕРЗИТЕТ У НОВОМ САДУ  **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ** |  |

Танасковић Александар

Тешић Борис

Стојановић Стефан

**Имплементација Др. Марио игре у 4-бојном текстуалном моду на „E2LP” платформи**

ИСПИТНИ РАД

- Логичко пројектовање рачунарских система 2 -

Ментор: Проф. Доц. Др. Душан Мајсторовић

Нови Сад, 2016

**Sadržaj**

[1. Uvod 1](#_Toc454025415)

[2. Implementacija hardverskog dela 2](#_Toc454025416)

[2.1 Dodavanje 4-boje u tekstualni mod 2](#_Toc454025417)

[2.2 Dodavanje interrupt-a 2](#_Toc454025418)

[2.3 Promena charrom-a 2](#_Toc454025419)

[3. Implementacija softverskog dela 3](#_Toc454025420)

[3.1 Main() 3](#_Toc454025421)

[3.2 Pomoćne funkcije 4](#_Toc454025422)

# Uvod

Implementacija Dr. Maria igre zahteva i hardwersku i programsku promenu/implementaciju već postojećeg koda. Sam zadatak što se hardverskog dela tiče, zahteva da se na postojeci projekat dodaju 4 boje, da se implementira interrupt koji ce sluziti za sinhronizaciju softwer-skog dela kod iscrtavanja slike ali i kao vremenski brojač s obzirom da će se interrupt “okidati“ na tačno 25 milisekundi, i promena char rom definisanih znakova koji će se koristiti za isctavanje elemenata igre.

# Implementacija hardverskog dela

Za sam projekat nasledjena je implementacija hardwera za igru tetris koja se mora menjati.

## Dodavanje 4-boje u tekstualni mod

Kod user\_logic je proširena širina signala TEXT\_MEM\_DATA\_WIDTH sa 6 na 8 bita tako da će 2 gornja bita sadržati informacije o bojama. Taj signal je spušten na vga\_top. Istovremeno su dodati 4 signala na foreground\_color za svaku od različitih boja. U vga\_top je taj signal od 8 bit-a slicovan tako da se gornja 2 bita prosledjuju na color\_mux koji preko multipleksera postavljaju 4 signala za foreground\_color. Preostalih 6 bit-a će ostati nepromenjeno tj. biti prosledjeno kao adresa znaka koji treba da se ispiše.

## Dodavanje interrupt-a

Dodavanje interrupt-a je uradjeno korišćenjem upustva za labaratorijsku vežbu 6 i 3. U user\_logic je implementiran timer. Kreirana je instance interrupt kontrolera preko EDK-a i izvršena konekcija interrupt kontrolera sa timerom.

## Promena charrom-a

Za promenu char roma korišćena je labaratorijska vežba 2. Prvo je promenut izgled karaktera koji se nalaze u char\_rom\_def\_mem.coe fajlu. Onda je preko Core-generatora “izbildovan” .ngc fajl koji je ubačen u projekat.

# Implementacija softverskog dela

Sama igra je realizovana preko stanja koji kontrolišu tok igre u main()-u, kao i uz upotrebu callback funkcije za interrupt koja osim iscrtavanja ekrana menja stanja brojača koji kontrolišu vremensku sinhronizaciju animacije i toka igre.

## Main()

Sam glavni deo programa će preko promene stanja kontrolisati tok igre.

* **OFF** – To je stanje u kojem se vrše podešavaju vrednosti promenjivih za startovanje igre kao i iscrtavanje pozadine i poruka igre koja se neće menjati tokom celog aktivnog toka igre. Postavlja se kao sledeće aktivno stanje stanje ON.
* **ON** – U ovom stanju se priprema tabela koja služi za predstavljanje ubačenih pilula kao I virusa. Prvo se tabela čisti, dodaju se virusi i kreira se trenutne pilule kao i sledeće pilule. Kao sledeće stanje postavlja se kao aktivno stanje PADANJE.
* **PADANJE** – To je stanje koje kontroliše padanje pilula tj. sinhronizacija brzine padanja pilule zajedno sa kontrolom dali pilula ima mogućnost da padne i to preko provere da li je mesto ispod pilule slobodno shodno orjentaciji same pilule. Ovo stanje će biti aktivno samo kad interrupt brojač predje odredjenu vrednost. Ta vrednost u kombinaciji sa promenjivom speed kontroliše brzinu padanja pilula. Kad se detektuje sudar tj. pilula je “udarila” u prepreku aktivira se stanje SREDJIVANJE.
* **SREDJIVANJE** – U ovom stanju se proverava dali igde na tabeli došlo do formiranja bilo koje kombinacije oblika i boje koji se može poništiti i bodovati. Poziva se funkcija sredjivanje koja detektuje i poništava bilo koji odgovarajući oblik. U slučaju da nema ništa za sredjivanje formira se nova pilula i postavlja kao sledeće aktivno stanje stanje PADANJE. U slučaju da je došlo do detekcije poništavanja odredjenih elemenata tabele čisti se tabela od pomoćnog znaka i aktivira se kao sledeće stanje SREDJIVANJE\_PADANJE.
* **SREDJIVANJE\_PADANJE** – Kad dodje do poništavanja odredjenog broja elemenata potrebno je proveriti dali svaki odgovarajući element može da ostane na toj poziciji ili mora da “propadne”. Da bi se sinhoniyovalo padanje sama aktivnost ovog stanja je povezana sa interrupt brojačem koji na takav način kontroliše brzinu padanja elemenata tabele koja je znatno brža od običajenog padanja samih pilula. U slučaju da je došlo do “propadanja” bilo kojeg elementa ostaje kao aktivno ovo stanje a u suprotnom kontrola se predaje stanju SREDJIVANJE koji treba da proveri da li je nakon “padanja” došlo do formiranja odgovarajućeg oblika za poništavanje.
* **KRAJ** – Ovo je krajnje stanje igre koje čuva na ekranu trenutni rezultat ili postoji mogućnost da se igra resetuje tj. prebaci u stanje OFF.

Pored 6 implementiranih stanja postoji kod koji kontroliše kretanje i rotiranje pilule preko očitavanja prekidača sa logikom koja će registrovani pritisak samo jednom pri pritisku sve dok se taster ne otpusti.

Da bi se kontrolisala animacija virusa proverava se stanje posebnog brojača interrupt-a koji menja vrednost promenjive koja kod iscrtavanja diktira koji predefinisan znak virusa će biti iscrtan.

Kod poništavanja svih virusa a da nije došlo do kraja igre vrši se povećanja levela igre i samim tim i brojnost virusa a kao sledeće stanje se postavlja stanje OFF bez obzira u kojem stanju igre se trenutno igrač nalazio.

## Pomoćne funkcije

Realizacija bilinearne interpolacije će se raditi preko funkcije:

* void **initializingPlatform**(); - funkcija koja inicijalizuje platformu i postavlja odgovarajuće vrednosti.
* void **creatingCurrentNextPills**(); - formira i postavlja trenutnu pilulu u tabelu i formira sledeću pilulu.
* void **newPillF**(); - smešta sledeću pilulu na tablu kao trenutnu i formira novu sledeću pilulu
* void **fillTableWBugs**(); - na osnovu random genereratora i broja virusa shodno trenutnom levelu formira na tabeli viruse.
* void **clearTable**(); - briše sve sa tabele i priprema tabelu za startovanje igre.
* unsigned char **sredjivanje**(); - To je osnovna funkcija koja vrši kontrolu pozivom drugih funkcija da li postoji neki oblik za poništavanje i ako postoji da ih poništi istovremeno vodeći računa da li je došlo do poništavanja što i vraća kao vrednost. Sama funkcija vrši proveru prvo vertikalno a onda horizontalno.
* unsigned char **proveraDole**(unsigned char kordY, unsigned char kordX, unsigned char bojaProvere); - na osnovu parametara početnih kordinata i boje proverava koliko elemenata tabele ima sa tom istom bojom vertikalno na dole od pocetne pozicije i tu vrednost vraća.
* unsigned char **proveraDesno**(unsigned char kordY,unsigned char kordX, unsigned char bojaProvere); - na osnovu početnih kordinata i boje proverava koliko ima elemenata u tabeli horizontalno prema desno od početne pozicije i tu vrednost vraća.
* void **ponistavanje**(unsigned char kordYp, unsigned char kordXp, unsigned char kordYk, unsigned char kordXk); - shodno početnoj poziciji kao i krajnoj poništava sve elemente na tim pozicijama pretvarajući element u poseban znak i zadržavajući istu boju. Sama funkcija radi samo vertikalno i horizontalno poništavanje. Isto tako shodno elementu koji poništava pretvara susedne elemente koji su povezani sa poništenim u odgovarajući novi element uz održavanje boje. Poseban znak je uoptrebljen da bi se omogućilo vertikalna i horizontalna provera cele tabele ali i zbog animacije samog uništavanja elemenata tabele. U slučaju da je uništen virus dodaju se bodovi na trenutni rezultat.
* void **clearingPill6**(); - služi da posle poništavaja očisti celu tabelu od upotrebljenog pomoćnog znaka koji je služio kao ispomoć za poništavanja ali i animaciju uništavanja.
* void my\_timer\_interrupt\_handler(void \* baseaddr\_p); - callback funkcija koja se aktivira na svaku aktivaciju interrupt –a. U njoj se vrši upis u tekstualnu memoriju za ispis na ekran.
* void drawTable(); - funkcija se upotrebljava za ispis trenutne vrednosti tabele tokom svakog interrupt-a.
* void drawBackground(); - iscrtava “grafičku“ pozadinu igre i poziva se samo jednom u stanju OFF.
* void drawGameState(); - ispisuje trenutno stanje statistike igre i to trenutni level, koliko je preostalo virusa na ekranu i trenutni score. Isto kao i drawTable poziva se na svaki interrupt.
* void drawStaticGameMessages(); - pomocna funkcija koja iscrtava pojedine tekstualne segmente pozadine i to samo jednom u stanju OFF.