Projektni rad

Smer: Računarstvo i Automatika

Predmet: Arhitektura Računara

Projekat: 2D Pucačina

Autor:

Nenad Palinkašević

Mentor:

Lazar Stričević

1) Uvod

U igri “Shoot Stuff” cilj vam je što više ubijenih neprijatelja i samim tim i veći skor. Napravljena je po uzoru “malih” flash igra koje sam igrao za vreme časova informatike u srednjoj školi, otuda i ideja za ovakvom igrom.

2) Instalacija

Jedna od najpopularnijh biblioteka za rad sa grafikom u C/C++ jezicima jeste SDL (Simple Directmedia Layer), upravo iz tog razloga sam i izabrao tu biblioteku. Mnogobrojni tutorijali I dokumentacije, veliki broj korisnika i godine testiranja i rada na SDL su je učinili laganom i zabavnom za rad.

Pored same SDL biblioteke, u igri se koriste još dve pomoćne biblioteke SDL_image i SDL_ttf, za učitavanje raznih formata I lakšeg rukovanja sa slikama i učitavanje fontova za ispis teksta u igri. Ove dve pomoćne biblioteke poprilično su čest dodatak samoj SDL biblioteci i takodje su dobro dokumentovane i lake za korišćenje.

Samo komplajiranje igre se svodi na pozivanje naredbe “make” iz terminale koja pokreće makefile i komplajira aplikaciju bez ikakvih poteškoća. Naravno moramo imati gore navede biblioteke instalirane na sistemu kao i gcc.

3) Implementacija

Bilo koj projekat, pa čak i manji kao što je ova igra, mora imati neku smislenu organizaciju koda radi lakšeg rada i uopšte funkcionisanje igre. Sam izvršni kod sam podelio na različite fajlove, gde se u .h nalaze deklaracije a u .c sam kod. Svaki fajl rešava svoj svojstveni problem, odnosno u svakoj datoteci imamo implementiran neki deo igre i tako igru rastavljamo na vise lakših delova, koji posle funkcionišu kao jedna celina.

Glavni fajl kao što je to običaj za C jezik jeste “main.c” u kom se nalazi samo glavna struktura igre. Gde imam Inicijalizaciju, glavnu petlju gde vršimo crtanje I hvatanje dogadjaja I na kraju oslobadjanje memorije.

```c
#include "Game.h"
#include "SDL/SDL.h"

int main(int argc, char* argv[])
{
    Init(800, 600, "Distro war");

    while(IsRunning())
    {
        Update();
        Draw();
    }

    Destroy();

    return 0;
}
```

Kao što je već rečeno projekat je podeljen na više fajlova, jedini fajl kog pozivamo u “main.c” jeste “Game.c”, gde smo zapravo i implementirali igru i gde se pozivaju svi drugi fajlovi. Čitava igra je podeljena na više stanja (“state”), na pocetku igre se nalazimo u “MENU_STATE”, gde je dalje napravljeno da se stanja menjaju prilikom nekih dogadjaja kao što je pritisak dugmeta.

```
 //state switch

switch(globals.curr_state)
{
    case MENU_STATE:
        MenuDraw(globals.screen);
        break;
    case GAME_STATE:
        GameDraw(globals.screen, &(globals.curr_state));
        break;
    case CREDITS_STATE:
        CreditsDraw(globals.screen);
        break;
    case HIGHSCORE_STATE:
        HighScoreDraw(globals.screen);
        break;
    case END_STATE:
        EndSceneDraw(globals.screen);
        break;
    case QUIT_STATE:   //exit game
            globals.running = 0;
            break;
};
```

Pored različitih stanja u igri, nalaze se i implementacije samog igrača, gde je uradjeno na poprilično standardan način kretanje i pucanje. Bitno je i napomenuti da se za generisanje neprijatelja i metkova ne koriste dinamičke strukture podataka, već statički niz ali zbog “recikliranja” ne dolazi do prekoračenje u memoriji.

```
int i;
for(i = 0; i < MAX_ENEMIES; ++i)
{
     if(enemies[i].state == DEAD)
     {
         enemies[i].state = ALIVE;
         break;
     }
}
```

Takodje je uradjena i animacija, gde učitavamo sliku sa više različitih frejmova, gde ih “seckamo” u kodu i prikazujemo jedan za drugim. Pošto možemo imati više različitih animacija na jednoj slici, postoji funkcija koja se poziva i odredjujemo od kog do kog frejma je animacija I dajemo joj ime, gde posle preko tog imena pozivamo tu animaciju za crtanje.

```
void AnimLoop(Anim* a, char* name, SDL_Surface* screen)
{
    int i,b,e, n;
    for(i = 0; i < MAX_ANIM_STATE; ++i)
    {
        if(strcmp(a->anim_data[i].name, name) == 0)
        {
            b = a->anim_data[i].begin;
            e = a->anim_data[i].end;
            n = a->anim_data[i].nivo;
            break;
        }
    }

       a->frame_index++;
       if(a->frame_index == 12){
           a->frame_index = 0;
        if(a->p < e)
            a->p++;

        if(a->p == e)
            a->p = 0;}

     ImageDrawSheet(screen, a->img_anim, a->w, a->h, (b*a->w) * a->p, n * ((b*a->h)));

}
```