СОУ „Јосип Броз – Тито“ – Битола

ПРОЕКТНА ЗАДАЧА ПО ПРОГРАМСКИ ЈАЗИЦИ

ТЕМА: Игра во C++

Ментор:
Проф. Мирјана Соклевска

Изработил:
Мартин Петковски

Содржина

Вовед

Почетокот на компјутерските игри датира уште од времето по Втората светска војна кога сеуште не бил откриен микропроцесорот и за еден компјутер била потребна цела просторија. Првата копмјутерска игра била адаптација на позната игра „Икс точка“ и била направена во 1952 година, а потоа почнале да се прават и покомплексни игри кои имале анимирана графика. Во овој проект јас сакав да направам моја верзија на една многу позната и безброј пати програмирана игра – Понг. Играта е напишана во програмскиот јазик C++, а за графиката е користена специјалната и нестандардна библиотека за графика Алегро. Играта е наменаета за два играчи: човек против компјутер, човек против човек или компјутер против компјутер. Целта на играта е со одбивање на едно топче со помош на палка за да се постигне поен. Со единаесет освоени поени еден од играчите победува во мечот.

Јазикот C++ е одбран како основен јазик во кој е напишана играта бидејќи е најпопуларен јазик за програмирање во оваа област и нуди можност за објектно ориентирано програмирање кое драстично ја зголемува прегледноста на кодот при програмирање на игра. Библиотеката Алегро е одбрана бидејќи нуди брз пристап до функции и класи специфично програмирани за дводимензинално програмирање на игри. Исто така оваа библиотека нуди и останати функции за вметнување на мултимедијални ресурси во играта како што се музиката, фонтовите и огрмоен спектар на графички формати.

Се одлучив за програмирање на игра од фамилијата „Понг“ бидејќи од графички аспект не е многу сложена, но сепак е многу забавна и заразна. Оваа игра се смета како почеток при навлегување подлабоко во областа на програмирањето на компјутерски игри.

Технички дел

Играта е напишана во програмскиот јазик C++. C++ е програмски јазик создаден во 1983 година од данскиот научник Бјарне Строуструп кој имал намера да го надогради програмскиот јазик C кој бил претходно напишан од Денис Ричи. C++ од C се разликува во тоа што во него е воведено објектно ориентирано програмирање кое имало цел да ја зголеми прегледноста на кодот, а и да го олесни пишувањето на огромни програми со користење на објекти. Овој програмски јазик се смета за јазик од средно ниво бидејќи комбинира елементи од програмирање на високо и ниско ниво. Денес C++ е еден од најкористените и најпопуларните програмски јазици кој се користи за програмирање на секојдневни апарати и на апарати кои ги користат вселенските агенции.

Алегро е библиотека со отворен код, првично создадена од Шон Харгривс во средината на девеесетите години, но сега таа е создавана од неколку стотици луѓе. Нејзиното име е кратенка од „**A**llegro **L**ow **L**evel **G**ame **Ro**utines“ што во превод значи „Алегро рутини за игри од ниско ниво“ и е создадена со цел за програмирање на игри на ниско ниво. Библиотеката Алегро е платформски независна што значи дека компајлирањето на кодот на Алегро може да биде за сите попопуларни оперативни системи меѓу кои и Андроид и iOS. Играта „Понг“ е компајлирана само за оперативниот систем Windows. Алегро има и неколку мултимедијални додатоци како што се вметнување на слики од многу формати како .png; .jpg; .gif; .bmp итн.; вметнување на аудио или аудиостримови од формати како .mp3; .wav; .ogg (Vorbis); вметнување на фонтови од true-type форматите .otf и .ttf.

Во играта „Понг“ покрај стандардните C++ библиотеки и библиотеката Алегро се искористени и следниве додатоци за библиотеката Алегро: allegro_audio, allegro_image и allegro_ttf.

Со цел да се олесни програмирањето на играта напишана е и дополнителна библиотека над библиотеката Алгеро која го олеснува цртањето на одредени елементи на играта како на пример менијата и лентите за вредност.

Функционалност

Играта започнува со главниот титл на играта по што се отвора главното мени. Главното мени содржи 6 опции за одбирање. Главното мени има позадинска музика која е специјлано направена за навигацијата во менијата.

Првата опција за одбирање е игра со еден играч против компјутер. Откако ќе се одбере оваа опција се отвора ново мени кое му дава опција на играчот да одбира колку е силен неговиот противник. Во ова мени има 5 опции за одбирање: Прелесно, Лесно, Така-така, Тешко и Невозможно. По одбирање на една опција играчот влегува во самата игра и на екранот забележува две палки и едно топче. Музиката во овој дел е различна. Во горниот дел од екранот на средината се наоѓа вредносната лента која покажува колку од силата на компјутерот е истрошена. Ако таа не е истрошена тогаш е обоена жолто и компјутерот сеуште има сила за да го победи играчот. Ако вредносната линија е целосно пополнета и црвена тогаш силата на компјутерот е скоро истрошена и само е прашање на време кога тој ќе изгуби. Силата на компјутерот се намалува со зголемување на брзината на топчето. Брзината на топчето се зголемува со секоја колизија со една од двете палки. Под оваа вредностна лента се наоѓа моменталниот резултат. Ако еден од играчите стигне до 11 поени тогаш тој е победник. Играта започнува со притискање на копчето „SPACE“ при што топчето почнува да се движи во една насока одбрана со рандом функција. Целта е секој од играчите да го одбие топчето за тоа да не стигне зад палката. Ако топчето стигне зад палката тогаш поен добива спротивниот играч. Палката на играчот се придвижува со притискање на стрелките на тастатурата. Со притискање на копчето „ESCAPE“ играчот се враќа во главното мени.

Втората опција од главното мени е игра со два играчи еден против друг, без компјутер. Оваа опција е слична како првата со неколку разлики: нема сила на компјутерот бидејќи нема палка контролирана по вештачки пат и за да нема недоразбирања меѓу двата играчи само првиот пат топчето почнува да се движи со притискање на копчето „SPACE“, а до крајот на мечот тоа автоматски се „фрла“ по освојување на поен на некој од играчите. Едниот играч ја придвижува својата палка на стрелките, а другиот на копчињата „W“ и „S“, соодветно за горе и долу.

Третата опција од главното мени е игра со два играчи контролирани од копмјутерот. Нивото на јачина на компјутерот е наместено некаде на средината за еден од играчите да може да изгуби, инаку играта ќе трае бесконечно.

Четвртата опција од главното мени е статистиката на сите игри досега изиграни во првата опција од главното мени.

Петтата опција е помош која на екранот ги печати сите контроли во играта и дава информации за креаторот на играта.

Шестата опција е излез од играта.

Скриен power-up

Во играта е додаден и скриен power-up кој многу тешко може да се освои. Направен е специјален отвор во алгоритамот за детекција на колизијата меѓу палката и топчето кој ако биде прецизно погоден му дава поголема брзина на топчето со множење на моменталната брзина. Ова го пореметува компјутерскиот алгоритам за детектирање на местото на колизија со што создава можност да се освои поен против компјутерот и на „Невозможно“ ниво.

Вештачка интелегенција

Со цел играчот да може да игра против компјутерот напишан е и алгоритам за автоматско движење на палката. Во другите игри од овој тип овој пробем е решен на начин што на палката на компјутерот му се назначува вредноста на координатата на топчето со што компјутерската палка секогаш се движи синхронизирано со топчето. Овој пристап јас го сметам за лош поради две причини: Првата причина е неефикасноста бидејќи компјутерската палка се движи и кога нема потреба од тоа и прави циклуси кои се вишок и втората причина е неферската игра на компјутерот на кој му се овозможува поголема брзина на палката одколку на играчот. При големи брзини на топчето ова прави голема разлика. Јас напишав алгоритам со кој компјутерската палка секогаш има константна брзина која е еднаква со онаа со која се движи палката на играчот и сепак секогаш може да стигне да го одбие топчето. Овој алгоритам функционира на начин што се креира еден објект од иста класа како и топчето, но за разлика од топчето што го гледаме ова топче е виртуелно и тоа служи само за правење на пресметки. Тоа исто се одбива како и топчето што го гледаме но е секогаш неколку пиксели во зависност од брзината пред топчето што можеме да го видиме. Ова виртуелно топче кога ќе удри на местото каде треба да се помести палката создава „жешка точка“ која откако ќе се создаде ја повикува функцијата која ја придвижува компјутерската палка со константна брзина до „жешката точка“ во временски период помал од разликата на удирањето на виртуелното и реалното топче во „жешката точка“. На овој начин се добива алгоритам за придвижување на компјутерската палка кој е оптимизиран и е фер во однос на другиот играч кој ја контролира палката преку тастатурата.

Заклучок

Човекот одсекогаш тежнеел надворешниот свет да го претопи во виртуелниот. Да создаде игра која ќе биде блиску до реалноста. Која ќе му овозможи на обичниот човек да прави работи за кои може само да сонува.

Играта „Понг“ иако многу пати досега била преработувана и секој пат надоградувана вреди уште еднаш да се „посети“ како основа и почеток на еден вид копирање на релниот свет во виртуелниот. Овој пат нејзината вештачка интелегенција е надоградена како никогаш досега. Воведен е нов начин на движење на палката кој овозможува фер натпревар помеѓу играчот и компјутерот и оптимизирано движење на компјутерската палка.

Користена литература:

1. **Bjarne Stroustrup: „The C++ Programming Language“, Reading, Mass. : Addison-Wesley, 1997**

2. **Пол Дејтел и Харви Дејтел: „Како се програмира“, Арс Ламина, Скопје, 2010**

3. **Stanely B. Lippman: „Osnove jezika C++“ , CET Computer Equipment and Trade, Beograd, 2000**

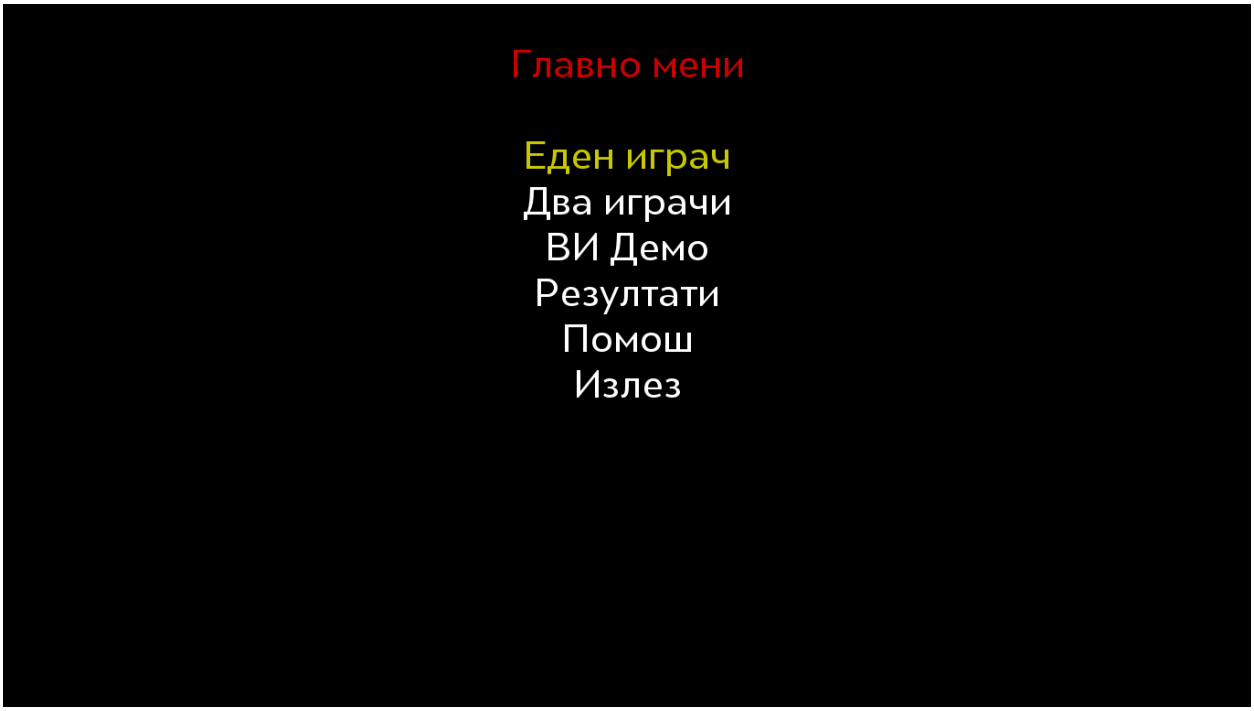
4. **http://www.cplusplus.com/reference/**

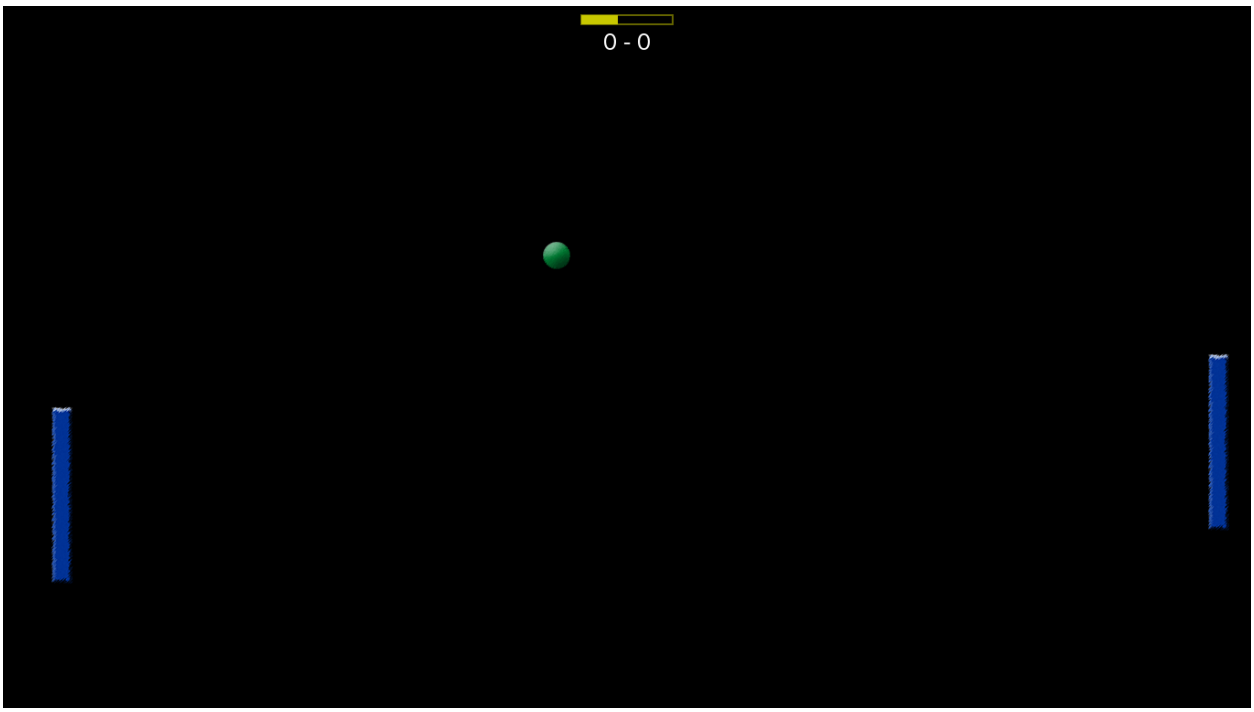
5. **http://alleg.sourceforge.net/a5docs/refman/**

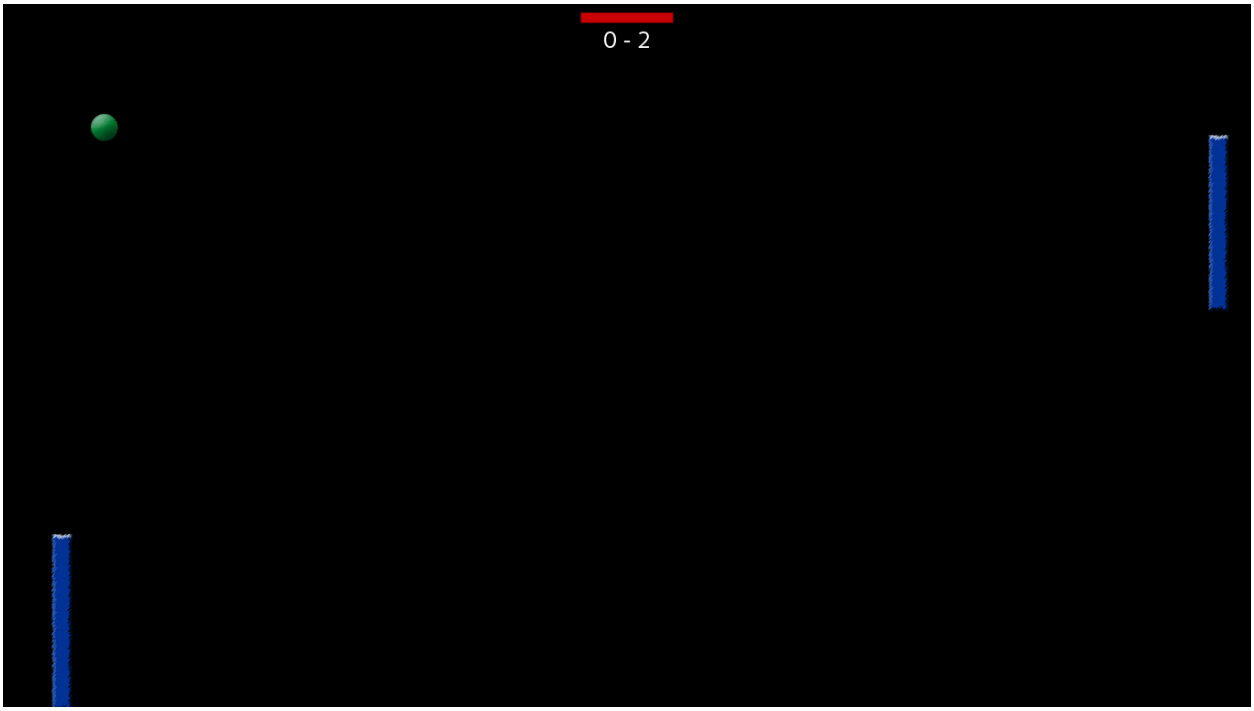
6. **https://www.allegro.cc/manual/5/**

7. **Во играта е искористена песната „Ова не е диско“ од македонскиот бенд The John**

Прилог:


Главно мени
Еден играч
Два играчи
ВИ Демо
Резултати
Помош
Излез


0 - 0



Еден играч

[Стрелка нагоре] - Движење на палката нагоре
[Стрелка надолу] - Движење на палката надолу
[SPACE] - Фрлање на топчето

Победник во играта е оној кој прв ќе постигне 11 поени.
Статистиката влегува во резултатите.

Два играчи

[W] - Движење на левата палката нагоре
[S] - Движење на левата палката надолу
[Стрелка нагоре] - Движење на десната палката нагоре
[Стрелка надолу] - Движење на десната палката надолу
Топчето се фрла автоматски по постигнување на поен

Победник во играта е оној кој прв ќе постигне 11 поени.

За Понг

Играта Понг е направена од Мартин Петковски за неговата матурска проектна задача.
Графиката на играта и музиката во менито е направена од Мартин Петковски.
Музиката во играта е „Ова не е диско“ од Д'Џон и е користена само за презентација.
Фонтовите се превземени бесплатно од страната на МИО.

Битола, Февруари/Март 2013

```cpp
//SOURCE.CPP

#include <allegro5/allegro.h>
#include <allegro5/allegro_image.h>
#include <allegro5/allegro_primitives.h>
#include <allegro5/allegro_font.h>
#include <allegro5/allegro_ttf.h>
#include <allegro5/allegro_audio.h>
#include <allegro5/allegro_acodec.h>
#include <fstream>
#include <vector>
#include <string>
#include "bam.h"
#include "objects.h"

//===============================================
// PROMENLIVI
//===============================================
int WIDTH, HEIGHT;

bool keys[] = {false, false, false, false, false};
enum KEYS{UP, DOWN, W, S, SPACE, ENTER};
bool done = false;
bool render = false;

int check_win;

ALLEGRO_DISPLAY *display = NULL;
ALLEGRO_EVENT_QUEUE *event_queue = NULL;
ALLEGRO_TIMER *timer;
ALLEGRO_FONT *font_big = NULL, *font_medium = NULL, *font_small = NULL;
ALLEGRO_DISPLAY_MODE disp_data;

//bitmapi
ALLEGRO_BITMAP *topce_image;
ALLEGRO_BITMAP *palka_image;
ALLEGRO_BITMAP *pozadina;

ALLEGRO_SAMPLE *menu_music;
ALLEGRO_SAMPLE *game_music;
ALLEGRO_SAMPLE *dang;

fstream hs;
string hsdata[10], hstmp;

int itohsdata[10];

global_game_settings settings;

string stringToIntAndBack(string oldValue, int modifyBy)
{
        char hsbuf[33];
        string stmp;
        int tmp = atoi(oldValue.c_str());
        tmp += modifyBy;
        itoa(tmp, hsbuf, 10);
        stmp = hsbuf;
        return stmp;
}


void single_player(int level)
{

        done = false;
        bool playstate = false;
```

```
topce t(WIDTH, HEIGHT,true,topce_image);
topce v(WIDTH, HEIGHT,false,topce_image);

palka player(WIDTH,HEIGHT,false,palka_image);
palka cpu(WIDTH,HEIGHT,true,palka_image);

settings.scorecpu = 0;
settings.scorep = 0;

al_clear_to_color(al_map_rgb(0,0,0));

event_queue = al_create_event_queue();
timer = al_create_timer(1.0 / 60);

al_register_event_source(event_queue, al_get_timer_event_source(timer));
al_register_event_source(event_queue, al_get_keyboard_event_source());

al_start_timer(timer);
while(!done)
{
        ALLEGRO_EVENT ev;
        al_wait_for_event(event_queue, &ev);

        //=============================================
        //INPUT
        //=============================================
        if(ev.type == ALLEGRO_EVENT_KEY_DOWN)
        {
                switch(ev.keyboard.keycode)
                {
                case ALLEGRO_KEY_ESCAPE:
                        done = true;
                        break;
                case ALLEGRO_KEY_UP:
                        keys[UP] = true;
                        break;
                case ALLEGRO_KEY_DOWN:
                        keys[DOWN] = true;
                        break;
                case ALLEGRO_KEY_SPACE:
                        keys[SPACE] = true;
                        break;
                }
        }
        else if(ev.type == ALLEGRO_EVENT_KEY_UP)
        {
                switch(ev.keyboard.keycode)
                {
                case ALLEGRO_KEY_ESCAPE:
                        done = true;
                        break;
                case ALLEGRO_KEY_UP:
                        keys[UP] = false;
                        break;
                case ALLEGRO_KEY_DOWN:
                        keys[DOWN] = false;
                        break;
                case ALLEGRO_KEY_SPACE:
                        keys[SPACE] = false;
                        break;
                }
        }

        //=============================================
        //UPDATE
        //=============================================
        else if(ev.type == ALLEGRO_EVENT_TIMER)
        {
                if(playstate)
```

```
                        {
                                if(v.bounce == 2)
                                        t.speed = 10;
                                if(v.bounce == 0)
                                        v.speed = 10;
                        }

                        if(t.speed > level*10)
                        {
                                t.speed = level*10;
                        }

                        if(keys[UP])
                                player.move(false);
                        if(keys[DOWN])
                                player.move(true);
                        if(keys[SPACE])
                                playstate = true;

                        t.move();
                        v.move();

                        if(v.target != -1)
                                cpu.ai_move(v.target);

                        cpu.screen_bound();

                        v.palka_bound(player.x + player.w);

                        int check_win = t.screen_bound();
                        if(check_win == 1)
                        {
                                t.reset();
                                v.reset();
                                v.dx = t.dx; v.dy = t.dy;
                                settings.scorecpu++;
                                playstate = false;

                                if(settings.scorecpu == 11)
                                {

        al_draw_text(font_medium,al_map_rgb(200,0,0),WIDTH/2,HEIGHT/2,ALLEGRO_ALIGN_CENTER,"Компјутерот
победи!");
                                        al_flip_display();

                                        hs.open("hs.sav");

                                        int hsi = 0;
                                        while(!hs.eof())
                                        {
                                                getline(hs,hsdata[hsi]);
                                                hsi++;
                                        }

                                        hs.close();
                                        hs.open("hs.sav",ios_base::trunc|ios_base::app);

                                        hs << hsdata[0];

                                        hstmp = stringToIntAndBack(hsdata[1],1);
                                        hs << hstmp;

                                        hstmp = stringToIntAndBack(hsdata[2],settings.dopir1);
                                        hs << hstmp;

                                        hstmp = stringToIntAndBack(hsdata[3],settings.dopir2);
                                        hs << hstmp;
```

```
                                        hstmp = stringToIntAndBack(hsdata[4],settings.dopir1 +
settings.dopir2);
                                        hs << hstmp;

                                        hs.close();

                                        al_rest(3);
                                        break;
                                }
                                //poen za kompjuterot
                        }
                        else if(check_win == 2)
                        {
                                t.reset();
                                v.reset();
                                v.dx = t.dx; v.dy = t.dy;
                                settings.scorep++;
                                playstate = false;

                                if(settings.scorep == 11)
                                {

        al_draw_text(font_medium,al_map_rgb(200,0,0),WIDTH/2,HEIGHT/2,ALLEGRO_ALIGN_CENTER,"Играчот
победи!");
                                        al_flip_display();
                                        al_rest(3);
                                        break;
                                }
                                //poen za igrachot
                        }
                        else {}//nikomu nishto

                        player.screen_bound();

                        if(phy_check_collision(t.x -
t.r,t.y,t.r,t.r,player.x,player.y,player.w,player.h))
                        {
                                if(t.dx == 0)
                                        t.dx = 1;
                                else
                                        t.dx = 0;

                                t.speed++;

                                al_play_sample(dang, 0.5, 0.0,1.0,ALLEGRO_PLAYMODE_ONCE,NULL);
                        }

                        if(phy_check_collision(t.x + t.r,t.y,t.r,t.r,cpu.x,cpu.y,cpu.w,cpu.h))
                        {
                                if(t.dx == 0)
                                        t.dx = 1;
                                else
                                        t.dx = 0;

                                v.target = -1;
                                t.speed++;
                                al_play_sample(dang, 0.5, 0.0,1.0,ALLEGRO_PLAYMODE_ONCE,NULL);
                        }

                        if(phy_check_collision(v.x - v.r,v.y,v.r,v.r,player.x,0,player.w,HEIGHT))
                        {
                                v.dx--;
                                v.bounce++;
                                v.speed++;
                        }

                        if(phy_check_collision(v.x + v.r,v.y,v.r,v.r,cpu.x,0,cpu.w,HEIGHT))
                        {
                                v.dx++;
```

```
                                v.bounce++;
                                v.target = v.y;
                                v.speed++;
                        }

                        cout<<t.speed<<" "<<v.speed<<endl;
                        render = true;
                }

                //============================================
                //RENDER
                //============================================
                if(render && al_is_event_queue_empty(event_queue))
                {
                        t.draw();
                        //v.draw();
                        player.draw();
                        cpu.draw();
                        settings.draw_highscore(font_small,WIDTH);

                        if(t.speed < level*10)
                                gui_draw_value_bar(t.speed, level*10 ,WIDTH/2 -
50,10,100,10,al_map_rgb(200,200,0),al_map_rgb(100,100,0));
                        else
                                gui_draw_value_bar(level*10, level*10 ,WIDTH/2 - 50,
10,100,10,al_map_rgb(200,0,0),al_map_rgb(100,0,0));

                        render = false;

                        al_flip_display();
                        al_clear_to_color(al_map_rgb(0,0,0));
                }
        }
}


void multiplayer()
{
        done = false;
        bool playstate = false;

        topce t(WIDTH, HEIGHT,true,topce_image);
        topce v(WIDTH, HEIGHT,false,topce_image);

        palka player(WIDTH,HEIGHT,false,palka_image);
        palka cpu(WIDTH,HEIGHT,true,palka_image);

        settings.scorecpu = 0;
        settings.scorep = 0;

        al_clear_to_color(al_map_rgb(0,0,0));

        event_queue = al_create_event_queue();
        timer = al_create_timer(1.0 / 60);

        al_register_event_source(event_queue, al_get_timer_event_source(timer));
        al_register_event_source(event_queue, al_get_keyboard_event_source());

        al_start_timer(timer);
        while(!done)
        {
                ALLEGRO_EVENT ev;
                al_wait_for_event(event_queue, &ev);

                //============================================
                //INPUT
                //============================================
                if(ev.type == ALLEGRO_EVENT_KEY_DOWN)
                {
```

```
        switch(ev.keyboard.keycode)
        {
        case ALLEGRO_KEY_ESCAPE:
                done = true;
                break;
        case ALLEGRO_KEY_W:
                keys[W] = true;
                break;
        case ALLEGRO_KEY_S:
                keys[S] = true;
                break;
        case ALLEGRO_KEY_UP:
                keys[UP] = true;
                break;
        case ALLEGRO_KEY_DOWN:
                keys[DOWN] = true;
                break;
        case ALLEGRO_KEY_SPACE:
                keys[SPACE] = true;
                break;
        }
}
else if(ev.type == ALLEGRO_EVENT_KEY_UP)
{
        switch(ev.keyboard.keycode)
        {
        case ALLEGRO_KEY_ESCAPE:
                done = true;
                break;
        case ALLEGRO_KEY_W:
                keys[W] = false;
                break;
        case ALLEGRO_KEY_S:
                keys[S] = false;
                break;
        case ALLEGRO_KEY_UP:
                keys[UP] = false;
                break;
        case ALLEGRO_KEY_DOWN:
                keys[DOWN] = false;
                break;
        case ALLEGRO_KEY_SPACE:
                keys[SPACE] = false;
                break;
        }
}

//=============================================
//UPDATE
//=============================================
else if(ev.type == ALLEGRO_EVENT_TIMER)
{

        if(playstate && t.speed < 10)
                t.speed = 10;

        if(keys[W])
                player.move(false);
        if(keys[S])
                player.move(true);

        if(keys[UP])
                cpu.move(false);
        if(keys[DOWN])
                cpu.move(true);

        if(keys[SPACE])
                playstate = true;
```

```
                        t.move();
                        v.move();

                        player.screen_bound();
                        cpu.screen_bound();

                        int check_win = t.screen_bound();
                        if(check_win == 1)
                        {
                                t.reset_multi();
                                settings.scorecpu++;
                                playstate = false;
                                if(settings.scorecpu == 11)
                                {

        al_draw_text(font_medium,al_map_rgb(200,0,0),WIDTH/2,HEIGHT/2,ALLEGRO_ALIGN_CENTER,"Вториот играч
победи!");
                                        al_flip_display();
                                        al_rest(3);
                                        break;
                                }
                                //poen za kompjuterot
                        }
                        else if(check_win == 2)
                        {
                                t.reset_multi();
                                settings.scorep++;
                                playstate = false;
                                if(settings.scorep == 11)
                                {

        al_draw_text(font_medium,al_map_rgb(200,0,0),WIDTH/2,HEIGHT/2,ALLEGRO_ALIGN_CENTER,"Првиот играч
победи!");
                                        al_flip_display();
                                        al_rest(3);
                                        break;
                                }
                                //poen za igrachot
                        }
                        else {}//nikomu nishto

                        if(phy_check_collision(t.x -
t.r,t.y,t.r,t.r,player.x,player.y,player.w,player.h))
                        {
                                if(t.dx == 0)
                                        t.dx = 1;
                                else
                                        t.dx = 0;

                                t.speed++;
                                al_play_sample(dang, 0.5, 0.0,1.0,ALLEGRO_PLAYMODE_ONCE,NULL);
                        }

                        if(phy_check_collision(t.x + t.r,t.y,t.r,t.r,cpu.x,cpu.y,cpu.w,cpu.h))
                        {
                                if(t.dx == 0)
                                        t.dx = 1;
                                else
                                        t.dx = 0;

                                v.target = -1;
                                t.speed++;
                                al_play_sample(dang, 0.5, 0.0,1.0,ALLEGRO_PLAYMODE_ONCE,NULL);
                        }

                        render = true;
                }

                //============================================
```

```
                //RENDER
                //=============================================
                if(render && al_is_event_queue_empty(event_queue))
                {
                        t.draw();
                        //v.draw();
                        player.draw();
                        cpu.draw();
                        settings.draw_highscore(font_small,WIDTH);

                        //gui_draw_value_bar(t.speed, 80 ,WIDTH/2 -
50,10,100,10,al_map_rgb(200,200,0),al_map_rgb(100,100,0));

                        render = false;

                        al_flip_display();
                        al_clear_to_color(al_map_rgb(0,0,0));
                }
        }
}

void demoplay()
{
        done = false;
        topce t(WIDTH, HEIGHT,true,topce_image);
        topce v(WIDTH, HEIGHT,false,topce_image);

        palka player(WIDTH,HEIGHT,false,palka_image);
        palka cpu(WIDTH,HEIGHT,true,palka_image);

        settings.scorecpu = 0;
        settings.scorep = 0;

        al_clear_to_color(al_map_rgb(0,0,0));

        event_queue = al_create_event_queue();
        timer = al_create_timer(1.0 / 60);

        al_register_event_source(event_queue, al_get_timer_event_source(timer));
        al_register_event_source(event_queue, al_get_keyboard_event_source());

        al_start_timer(timer);
        while(!done)
        {
                ALLEGRO_EVENT ev;
                al_wait_for_event(event_queue, &ev);

                //=============================================
                //INPUT
                //=============================================
                if(ev.type == ALLEGRO_EVENT_KEY_DOWN)
                {
                        switch(ev.keyboard.keycode)
                        {
                        case ALLEGRO_KEY_ESCAPE:
                                done = true;
                                break;
                        case ALLEGRO_KEY_UP:
                                keys[UP] = true;
                                break;
                        case ALLEGRO_KEY_DOWN:
                                keys[DOWN] = true;
                                break;
                        case ALLEGRO_KEY_SPACE:
                                keys[SPACE] = true;
                                break;
                        }
                }
                else if(ev.type == ALLEGRO_EVENT_KEY_UP)
```

```
		{
			switch(ev.keyboard.keycode)
			{
			case ALLEGRO_KEY_ESCAPE:
				done = true;
				break;
			case ALLEGRO_KEY_UP:
				keys[UP] = false;
				break;
			case ALLEGRO_KEY_DOWN:
				keys[DOWN] = false;
				break;
			case ALLEGRO_KEY_SPACE:
				keys[SPACE] = false;
				break;
			}
		}

		//===========================================
		//UPDATE
		//===========================================
		else if(ev.type == ALLEGRO_EVENT_TIMER)
		{
			if(v.bounce == 2)
				t.speed = 10;
			if(v.bounce == 0)
				v.speed = 10;

			t.move();
			v.move();

			if(v.target != -1)
				cpu.ai_move(v.target);

			if(v.targetdemo != -1)
				player.ai_move(v.targetdemo);

			cpu.screen_bound();
			player.screen_bound();

			v.palka_bound(player.x + player.w);

			int check_win = t.screen_bound();
			if(check_win == 1)
			{
				t.reset();
				v.reset();
				v.dx = t.dx; v.dy = t.dy;
				settings.scorecpu++;

				if(settings.scorecpu == 11)
				{

	al_draw_text(font_medium,al_map_rgb(200,0,0),WIDTH/2,HEIGHT/2,ALLEGRO_ALIGN_CENTER,"Вториот играч победи!");
					al_flip_display();
					al_rest(3);
				}
				//poen za kompjuterot
			}
			else if(check_win == 2)
			{
				t.reset();
				v.reset();
				v.dx = t.dx; v.dy = t.dy;
				settings.scorep++;

				if(settings.scorep == 11)
				{
```

```
        al_draw_text(font_medium,al_map_rgb(200,0,0),WIDTH/2,HEIGHT/2,ALLEGRO_ALIGN_CENTER,"Првиот играч
победи!");
                                        al_flip_display();
                                        al_rest(3);
                                }
                                //poen za igrachot
                        }
                        else {}//nikomu nishto

                        if(phy_check_collision(t.x -
t.r,t.y,t.r,t.r,player.x,player.y,player.w,player.h))
                        {
                                if(t.dx == 0)
                                        t.dx = 1;
                                else
                                        t.dx = 0;

                                v.targetdemo = -1;
                                t.speed++;
                                al_play_sample(dang, 0.5, 0.0,1.0,ALLEGRO_PLAYMODE_ONCE,NULL);
                        }

                        if(phy_check_collision(t.x + t.r,t.y,t.r,t.r,cpu.x,cpu.y,cpu.w,cpu.h))
                        {
                                if(t.dx == 0)
                                        t.dx = 1;
                                else
                                        t.dx = 0;

                                v.target = -1;
                                t.speed++;
                                al_play_sample(dang, 0.5, 0.0,1.0,ALLEGRO_PLAYMODE_ONCE,NULL);
                        }

                        if(phy_check_collision(v.x - v.r,v.y,v.r,v.r,player.x,0,player.w,HEIGHT))
                        {
                                v.dx--;
                                v.bounce++;
                                v.targetdemo = v.y;
                                v.speed++;
                        }

                        if(phy_check_collision(v.x + v.r,v.y,v.r,v.r,cpu.x,0,cpu.w,HEIGHT))
                        {
                                v.dx++;
                                v.bounce++;
                                v.target = v.y;
                                v.speed++;
                        }

                        cout<<t.speed<<" "<<v.speed<<endl;
                        render = true;
                }

                //============================================
                //RENDER
                //============================================
                if(render && al_is_event_queue_empty(event_queue))
                {
                        t.draw();
                        //v.draw();
                        player.draw();
                        cpu.draw();
                        settings.draw_highscore(font_small,WIDTH);

                        /*
                        if(t.speed < level*10)
```

```
                        gui_draw_value_bar(t.speed, level*10 ,WIDTH/2 -
50,10,100,10,al_map_rgb(200,200,0),al_map_rgb(100,100,0));
                        else
                        gui_draw_value_bar(level*10, level*10 ,WIDTH/2 - 50,
10,100,10,al_map_rgb(200,0,0),al_map_rgb(100,0,0));*/

                        render = false;

                        al_flip_display();
                        al_clear_to_color(al_map_rgb(0,0,0));
                }
        }
}

int pomosh()
{
        done = false;
        event_queue = al_create_event_queue();
        timer = al_create_timer(1.0 / 60);

        al_register_event_source(event_queue, al_get_timer_event_source(timer));
        al_register_event_source(event_queue, al_get_keyboard_event_source());

        al_start_timer(timer);

        while(!done)
        {
                ALLEGRO_EVENT ev;
                al_wait_for_event(event_queue, &ev);

                //=============================================
                //INPUT
                //=============================================
                if(ev.type == ALLEGRO_EVENT_KEY_DOWN)
                {
                        switch(ev.keyboard.keycode)
                        {
                        case ALLEGRO_KEY_ESCAPE:
                                return 0;
                        }
                }
                else if(ev.type == ALLEGRO_EVENT_KEY_UP)
                {
                        switch(ev.keyboard.keycode)
                        {
                        case ALLEGRO_KEY_ESCAPE:
                                return 0;
                        }
                }
                else if(ev.type == ALLEGRO_EVENT_TIMER)
                {
                        render = true;
                }

                if(render && al_is_event_queue_empty(event_queue))
                {
                        al_clear_to_color(al_map_rgb(0,0,0));
                        al_draw_text(font_medium,al_map_rgb(0,128,255),50,50,ALLEGRO_ALIGN_LEFT,"Еден
играч");

        al_draw_text(font_small,al_map_rgb(240,240,240),50,120,ALLEGRO_ALIGN_LEFT,"[Стрелка нагоре] -
Движење на палката нагоре");

        al_draw_text(font_small,al_map_rgb(240,240,240),50,150,ALLEGRO_ALIGN_LEFT,"[Стрелка надолу] -
Движење на палката надолу");

        al_draw_text(font_small,al_map_rgb(240,240,240),50,180,ALLEGRO_ALIGN_LEFT,"[SPACE] - Фрлање на
топчето");
```

```
        al_draw_text(font_small,al_map_rgb(240,240,240),50,240,ALLEGRO_ALIGN_LEFT,"Победник во играта е
оној кој прв");
                        al_draw_text(font_small,al_map_rgb(240,240,240),50,270,ALLEGRO_ALIGN_LEFT,"ќе
постигне 11 поени. ");

        al_draw_text(font_small,al_map_rgb(240,240,240),50,300,ALLEGRO_ALIGN_LEFT,"Статистиката влегува во
резултатите.");

                        al_draw_text(font_medium,al_map_rgb(0,128,255),WIDTH -
50,50,ALLEGRO_ALIGN_RIGHT,"Два играчи");
                        al_draw_text(font_small,al_map_rgb(240,240,240),WIDTH -
50,120,ALLEGRO_ALIGN_RIGHT,"[W] - Движење на левата палката нагоре");
                        al_draw_text(font_small,al_map_rgb(240,240,240),WIDTH -
50,150,ALLEGRO_ALIGN_RIGHT,"[S] - Движење на левата палката надолу");
                        al_draw_text(font_small,al_map_rgb(240,240,240),WIDTH -
50,180,ALLEGRO_ALIGN_RIGHT,"[Стрелка нагоре] - Движење на десната палката нагоре");
                        al_draw_text(font_small,al_map_rgb(240,240,240),WIDTH -
50,210,ALLEGRO_ALIGN_RIGHT,"[Стрелка надолу] - Движење на десната палката надолу");
                        al_draw_text(font_small,al_map_rgb(240,240,240),WIDTH -
50,240,ALLEGRO_ALIGN_RIGHT,"Топчето се фрла автоматски по постигнување на поен");
                        al_draw_text(font_small,al_map_rgb(240,240,240),WIDTH -
50,300,ALLEGRO_ALIGN_RIGHT,"Победник во играта е оној кој прв");
                        al_draw_text(font_small,al_map_rgb(240,240,240),WIDTH -
50,330,ALLEGRO_ALIGN_RIGHT,"ќе постигне 11 поени.");


        al_draw_text(font_medium,al_map_rgb(0,128,255),WIDTH/2,420,ALLEGRO_ALIGN_CENTER,"За Понг");

        al_draw_text(font_small,al_map_rgb(240,240,240),WIDTH/2,490,ALLEGRO_ALIGN_CENTER,"Играта Понг е
направена од Мартин Петковски за неговата матурска проектна задача.");

        al_draw_text(font_small,al_map_rgb(240,240,240),WIDTH/2,520,ALLEGRO_ALIGN_CENTER,"Графиката на
играта и музиката во менито е направена од Мартин Петковски.");

        al_draw_text(font_small,al_map_rgb(240,240,240),WIDTH/2,550,ALLEGRO_ALIGN_CENTER,"Музиката во
играта е „Ова не е диско“ од Д’Џон и е користена само за презентација.");

        al_draw_text(font_small,al_map_rgb(240,240,240),WIDTH/2,580,ALLEGRO_ALIGN_CENTER,"Фонтовите се
превземени бесплатно од страната на МИО.");

        al_draw_text(font_small,al_map_rgb(240,240,240),WIDTH/2,610,ALLEGRO_ALIGN_CENTER,"");

        al_draw_text(font_small,al_map_rgb(240,240,240),WIDTH/2,640,ALLEGRO_ALIGN_CENTER,"Битола,
Февруари/Март 2013");

        al_draw_text(font_small,al_map_rgb(240,240,240),WIDTH/2,670,ALLEGRO_ALIGN_CENTER,"");
                        al_flip_display();
                }
        }
}


int main(int argc, char **argv)
{
        //===============================================
        // INICIJALIZACIJA NA ALLEGRO
        //===============================================

        if(!al_init())
                return -1;

        al_get_display_mode(al_get_num_display_modes() - 1, &disp_data);

        al_set_new_display_flags(ALLEGRO_FULLSCREEN);
        WIDTH = disp_data.width;
        HEIGHT = disp_data.height;
        display = al_create_display(WIDTH, HEIGHT);
```

```
if(!display)
        return -1;

//============================================
// INICIJALIZACIJA NA DODATOCI
//============================================

al_install_keyboard();
al_init_image_addon();
al_init_font_addon();
al_init_ttf_addon();
al_init_primitives_addon();

if(!al_install_audio()){
        fprintf(stderr, "failed to initialize audio!\n");
        return -1;
}

if(!al_init_acodec_addon()){
        fprintf(stderr, "failed to initialize audio codecs!\n");
        return -1;
}

if (!al_reserve_samples(5)){
        fprintf(stderr, "failed to reserve samples!\n");
        return -1;
}

//============================================
//INICIJALIZACIJA NA PROMENLIVI
//============================================
font_big = al_load_font("skola.otf", 60, 0);
al_clear_to_color(al_map_rgb(0,0,0));
al_draw_text(font_big,al_map_rgb(200,0,0),200,200,0,"Вчитување ...");
al_flip_display();

font_medium = al_load_font("skola.otf", 40, 0);
font_small = al_load_font("skola.otf", 24, 0);

vector<string> options;
options.push_back("Еден играч");
options.push_back("Два играчи");
options.push_back("ВИ Демо");
options.push_back("Резултати");
options.push_back("Помош");
options.push_back("Излез");

vector<string> levels;
levels.push_back("Прелесно");
levels.push_back("Лесно");
levels.push_back("Така-така");
levels.push_back("Тешко");
levels.push_back("Невозможно");

topce_image = al_load_bitmap("ball.png");
if(!topce_image) {
        fprintf(stderr, "failed to create topce bitmap!\n");
        al_destroy_display(display);
        al_destroy_timer(timer);
        return -1;
}

palka_image = al_load_bitmap("bat.png");
if(!palka_image) {
        fprintf(stderr, "failed to create palka bitmap!\n");
        al_destroy_display(display);
        al_destroy_timer(timer);
        return -1;
}
```

```
	menu_music = al_load_sample("menu_music.wav");
	if(!menu_music)
	{
		fprintf(stderr, "failed to create menu music!\n");
		return -1;
	}

	game_music = al_load_sample("thejohn.wav");
	if(!game_music)
	{
		fprintf(stderr, "failed to create game music!\n");
		return -1;
	}

	dang = al_load_sample("dang.wav");
	if(!dang)
	{
		fprintf(stderr, "failed to create dang sound!\n");
		return -1;
	}

	//=============================================
	// OPCII
	//=============================================
	int selection, level;

	while(1)
	{
		al_stop_samples();
		al_play_sample(menu_music, 0.7, 0.0,1.0,ALLEGRO_PLAYMODE_LOOP,NULL);
		selection = gui_generate_menu(options,"Главно
мени",font_medium,al_map_rgb(200,0,0),al_map_rgb(255,255,255),al_map_rgb(200,200,0),al_map_rgb(0,0,0),WIDT
H/2,100,50);

		if(selection == 1)
		{
			al_clear_to_color(al_map_rgb(0,0,0));
			level = gui_generate_menu(levels,"Одбери
ниво",font_medium,al_map_rgb(200,0,0),al_map_rgb(255,255,255),al_map_rgb(200,200,0),al_map_rgb(0,0,0),WIDT
H/2,100,50);

			al_stop_samples();
			al_play_sample(game_music, 1.0, 0.0,1.0,ALLEGRO_PLAYMODE_LOOP,NULL);
			single_player(level);
		}
		else if(selection == 2)
		{
			al_stop_samples();
			al_play_sample(game_music, 1.0, 0.0,1.0,ALLEGRO_PLAYMODE_LOOP,NULL);
			multiplayer();
		}
		else if(selection == 3)
		{
			al_stop_samples();
			al_play_sample(game_music, 1.0, 0.0,1.0,ALLEGRO_PLAYMODE_LOOP,NULL);
			demoplay();
		}
		else if(selection == 5)
			pomosh();

		else if(selection == 6)
			break;
	}

	//=============================================
	//UNISHTUVANJE NA ALLEGRO OBJEKTI
	//=============================================
```

```
        al_destroy_font(font_big);
        al_destroy_timer(timer);
        al_destroy_event_queue(event_queue);
        al_destroy_display(display);
        al_destroy_sample(menu_music);

        return 0;

}

//BAM.H

#include<string>
#include<iostream>
#include<vector>
#include<allegro5/allegro.h>
#include<allegro5/allegro_image.h>
#include<allegro5/allegro_primitives.h>
#include<allegro5/allegro_font.h>
#include<allegro5/allegro_ttf.h>
using namespace std;

//========================//
//     STATIC EFFECTS     //
//========================//

void fx_static_fadein(ALLEGRO_COLOR fade_from, int screen_w, int screen_h, int duration, ALLEGRO_BITMAP
*image, int ix, int iy);
void fx_static_fadeout(ALLEGRO_COLOR fade_to, int screen_w, int screen_h, int duration, ALLEGRO_BITMAP
*image, int ix, int iy);
void fx_static_fadetoggle(ALLEGRO_COLOR fade_from, ALLEGRO_COLOR fade_to, int screen_w, int screen_h, int
duration, ALLEGRO_BITMAP *image, int ix, int iy, int pause);

//==============================//
//     DYNAMIC EFFECTS #TODO    //
//==============================//

//====================//
//     GUI TWEAKS     //
//====================//

void gui_draw_value_bar(double current_value, double max_value, int x , int y, int w, int h, ALLEGRO_COLOR
color, ALLEGRO_COLOR border_color);
int gui_generate_menu(vector<string> options, string title, ALLEGRO_FONT *font, ALLEGRO_COLOR title_color,
ALLEGRO_COLOR default_color,
                                        ALLEGRO_COLOR focus_color, ALLEGRO_COLOR background_color, int
margin_left, int margin_top, int options_padding,
                                        ALLEGRO_BITMAP *image, int theme);

//========================//
//     PHYSICS ENGINE     //
//========================//

bool phy_check_collision(int b1_x, int b1_y, int b1_w, int b1_h, int b2_x, int b2_y, int b2_w, int b2_h);
bool phy_check_collision_top(int x1,int w1,int y1, int h1, int x2, int y2, int w2, double thickness);
bool phy_check_collision_bottom(int x1,int w1,int y1, int h2, int x2, int y2, int w2, double thickness);
bool phy_check_collision_left(int x1,int w1,int y1, int h2, int x2, int y2, int h1, double thickness);
bool phy_check_collision_right(int x1,int w2,int y1, int h2, int x2, int y2, int h1, double thickness);
double phy_add_gravity(double speed, double max_speed, int fps);


//=========================================================================//
//                                                                         //
//                           STATIC EFFECTS                                //
//                                                                         //
//=========================================================================//
```

```
void fx_static_fadein(ALLEGRO_COLOR fade_from, int screen_w, int screen_h, int duration, ALLEGRO_BITMAP
*image = NULL,int ix = NULL, int iy = NULL)
{
        int alpha = 255;
        al_clear_to_color(fade_from);

        while(alpha >= 0)
        {
                if(image != NULL)
                        al_draw_bitmap(image,ix,iy,0);
                al_draw_filled_rectangle(0,0,screen_w,screen_h,al_map_rgba(0,0,0,alpha));
                al_flip_display();
                al_rest(duration/255);
                alpha--;
        }
}

void fx_static_fadeout(ALLEGRO_COLOR fade_to, int screen_w, int screen_h, int duration, ALLEGRO_BITMAP
*image = NULL,int ix = NULL, int iy = NULL)
{
        int alpha = 0;
        al_clear_to_color(fade_to);

        while(alpha <= 255)
        {
                if(image != NULL)
                        al_draw_bitmap(image,ix,iy,0);
                al_draw_filled_rectangle(0,0,screen_w,screen_h,al_map_rgba(0,0,0,alpha));
                al_flip_display();
                al_rest(duration/255);
                alpha++;
        }
}

void fx_static_fadetoggle(ALLEGRO_COLOR fade_from, ALLEGRO_COLOR fade_to, int screen_w, int screen_h, int
duration,
                                                ALLEGRO_BITMAP *image, int ix, int iy, int pause)
{
        fx_static_fadein(fade_from,screen_w,screen_h,duration,image,0,0);
        al_rest(pause);
        fx_static_fadeout(fade_to,screen_w,screen_h,duration,image,0,0);
}


//========================================================================//
//        #TODO                                                           //
//               DYNAMIC EFFECTS (MULTITHREADING NEEDED)                  //
//                                                                        //
//========================================================================//


//========================================================================//
//                                                                        //
//                              GUI TWEAKS                                //
//                                                                        //
//========================================================================//

void gui_draw_value_bar(double current_value, double max_value, int x, int y, int w, int h, ALLEGRO_COLOR
color, ALLEGRO_COLOR border_color)
{
        al_draw_rectangle(x,y,x+w,y+h,border_color,2);
        al_draw_filled_rectangle(x,y,x + current_value*w/max_value,y+h,color);
}


int gui_generate_menu(vector<string> options, string title, ALLEGRO_FONT *font, ALLEGRO_COLOR title_color,
ALLEGRO_COLOR default_color, ALLEGRO_COLOR focus_color, ALLEGRO_COLOR background_color,
                                        int margin_left, int margin_top, int options_padding)
{
```

```
int position = 1;
int numOptions = options.size();

ALLEGRO_EVENT_QUEUE *menu_event_queue = NULL;
ALLEGRO_TIMER *menu_timer;

bool render = false;
bool keys[] = {false, false, false, false, false};
enum KEYS{UP, DOWN, LEFT, RIGHT, SPACE};

menu_event_queue = al_create_event_queue();
menu_timer = al_create_timer(1/13.0);

al_register_event_source(menu_event_queue, al_get_timer_event_source(menu_timer));
al_register_event_source(menu_event_queue, al_get_keyboard_event_source());

al_start_timer(menu_timer);
while(1)
{
	ALLEGRO_EVENT menu_ev;
	al_wait_for_event(menu_event_queue, &menu_ev);

	//================================================
	//INPUT
	//================================================
	if(menu_ev.type == ALLEGRO_EVENT_KEY_DOWN)
	{
		switch(menu_ev.keyboard.keycode)
		{
		case ALLEGRO_KEY_UP:
			keys[UP] = true;
			break;
		case ALLEGRO_KEY_DOWN:
			keys[DOWN] = true;
			break;
		case ALLEGRO_KEY_ENTER:
			keys[SPACE] = true;
			break;
		}
	}
	else if(menu_ev.type == ALLEGRO_EVENT_KEY_UP)
	{
		switch(menu_ev.keyboard.keycode)
		{
		case ALLEGRO_KEY_UP:
			keys[UP] = false;
			break;
		case ALLEGRO_KEY_DOWN:
			keys[DOWN] = false;
			break;
		case ALLEGRO_KEY_ENTER:
			keys[SPACE] = false;
			break;
		}
	}

	//================================================
	//UPDATE
	//================================================
	else if(menu_ev.type == ALLEGRO_EVENT_TIMER)
	{
		if(keys[UP])
			position--;
		else if(keys[DOWN])
			position++;
		else if(keys[SPACE])
			return position;

		if(position < 1)
```

```
                        position = numOptions;
                if(position > numOptions)
                        position = 1;

                render = true;
        }

        //=============================================
        //RENDER
        //=============================================
        if(render && al_is_event_queue_empty(menu_event_queue))
        {
                //cout<<"success"<<endl;
                al_draw_text(font,title_color,margin_left,50,ALLEGRO_ALIGN_CENTER,
title.c_str());

                for(int i = 1; i<=numOptions; i++)
                {
                        if(i == position)
                        {
                                al_draw_text(font, focus_color, margin_left,
margin_top+i*options_padding, ALLEGRO_ALIGN_CENTER, options[i-1].c_str());
                        }
                        else
                        {
                                al_draw_text(font, default_color, margin_left,
margin_top+i*options_padding, ALLEGRO_ALIGN_CENTER, options[i-1].c_str());
                        }
                }

                al_flip_display();
                al_clear_to_color(al_map_rgb(0,0,0));

                render = false;
        }

    }
}

//============================================================================//
//                                                                            //
//                          PHYSICS ENGINE                                    //
//                                                                            //
//============================================================================//

bool phy_check_collision(int b1_x, int b1_y, int b1_w, int b1_h, int b2_x, int b2_y, int b2_w, int b2_h)
{
    if ((b1_x > b2_x + b2_w - 1) || (b1_y > b2_y + b2_h - 1) || (b2_x > b1_x + b1_w - 1) || (b2_y >
b1_y + b1_h - 1))
            return false;
    else return true;
}

bool phy_check_collision_top(int x1,int w1,int y1, int h1, int x2, int y2, int w2, int h2, double
thickness)
{
    if(thickness < 5) thickness = 5;
    if((x1+w1 > x2 && x1 < x2+w2) && (y1+h1 >= y2 && y1+h1 <= y2+h2-10))
            return true;
    return false;
}

bool phy_check_collision_bottom(int x1,int w1,int y1, int h2, int x2, int y2, int w2, double thickness)
{
    if(thickness < 5) thickness = 5;
    if((x1+w1 > x2 && x1 < x2+w2) && (y1 <= y2+h2 && y1+thickness >= y2+h2))
            return true;
    return false;
}
```

```
bool phy_check_collision_left(int x1,int w1,int y1, int h2, int x2, int y2, int h1, double thickness)
{
	if(thickness < 5) thickness = 5;
	if((y1 < y2 + h2 && y1 + h1 > y2) && (x1 + w1 >= x2 && x1 + w1 <= x2 + thickness))
		return true;
	return false;
}

bool phy_check_collision_right(int x1,int w2,int y1, int h2, int x2, int y2, int h1, double thickness)
{
	if(thickness < 5) thickness = 5;
	if((y1 < y2 + h2 && y1 + h1 > y2) && (x1 <= x2 + w2 && x1 + thickness >= x2 + w2 ))
		return true;
	return false;
}

double phy_add_gravity(double speed, double max_speed, int fps)
{
	const double acceleration = 9.81;
	double timeElapsed = 1 / (double)fps;

	if(speed > max_speed)
		return max_speed;
	else return speed + acceleration * timeElapsed;
}


//OBJECTS.H

class global_game_settings
{
public:
	int scorep, scorecpu;
	int dopir1, dopir2;

	void draw_highscore(ALLEGRO_FONT *font,int screen_w)
	{
		al_draw_textf(font,al_map_rgb(255,255,255),screen_w/2,30,ALLEGRO_ALIGN_CENTER,"%d -
%d",scorep,scorecpu);
	}
};

class topce
{
private:
	int w,h;
	bool iv;
	ALLEGRO_BITMAP *img;
public:
	int x, y, r, dx, dy, speed, bounce;
	int target, targetdemo;

	int grn(int min, int max)
	{
		srand(time(NULL));
		return(rand()%(max-min)+min);
	}

	topce(int screen_w, int screen_h, bool isnt_virtual, ALLEGRO_BITMAP *image)
	{
		r = 15;
		x = screen_w/2 - r;
		y = screen_h/2 - r;

		w = screen_w;
		h = screen_h;
		dx = grn(0,1); dy = grn(0,1);
		speed = 0;
```

```
        iv = isnt_virtual;

        bounce = 0;
        target = -1;
        targetdemo = -1;

        img = image;
}

void reset()
{
        r = 15;
        x = w/2 - r;
        y = h/2 - r;
        //dx = grn(0,1); dy = grn(0,1);
        speed = 0;

        bounce = 0;
        target = -1;
        targetdemo = -1;
}

void reset_multi()
{
        r = 15;
        x = w/2 - r;
        y = h/2 - r;
        //dx = grn(0,1); dy = grn(0,1);
        speed = 10;
}

int screen_bound()
{
        if(x - r < 0)
                return 1;
        if(x + r > w)
                return 2;
        if(y - r < 0)
        {
                y = 0 + r;
                dy--;
                return 0;
        }
        if(y + r > h)
        {
                y = h - r;
                dy++;
                return 0;
        }
}

int palka_bound(int bound)
{
        if(y - r < 0)
        {
                y = 0 + r;
                dy--;
                bounce ++;
                return 0;
        }
        if(y + r > h)
        {
                y = h - r;
                dy++;
                bounce ++;
                return 0;
        }
}
```

```
        void draw()
        {
                if(iv)
                {
                        //al_draw_circle(x,y,r,al_map_rgb(0,200,0),5);
                        al_draw_bitmap(img,x-r,y-r,0);
                }
                else
                        al_draw_circle(x,y,r,al_map_rgb(200,0,0),4);
        }

        void move()
        {
                if(dx == 0)
                        x += speed;
                else
                        x-= speed;
                if(dy == 0)
                        y += speed;
                else
                        y -= speed;
        }
};

class palka
{
        private:
        int sw,sh;
public:
        int x, y, w,h, dx, dy, speed;
        ALLEGRO_BITMAP *img;

        palka(int screen_w, int screen_h, bool cpu, ALLEGRO_BITMAP *image)
        {
                sw = screen_w;
                sh = screen_h;
                w = 20;
                h = sh/4;

                if(cpu)
                        x = sw - 50;
                else
                        x = 50;

                y = sh/2 - h/2;
                dx = 0; dy = 1;
                speed = 10;
                img = image;
        }

        void move(bool direction)
        {
                if(direction)
                        y += speed;
                else
                        y -= speed;
        }

        void ai_move(int target)
        {
                if(target < y + 50)
                        y-=speed;
                else if(target > y + h - 50)
                        y+=speed;
        }

        void screen_bound()
        {
                if(y < 0)
```

```
                        y = 0;
                if(y+h > sh)
                        y = sh - h;
        }

        void draw()
        {
                //al_draw_filled_rectangle(x,y,x+w,y+h,al_map_rgb(0,0,200));
                al_draw_bitmap(img,x,y,0);
        }
};

class powerup
{
public:
        int xx,yy,t;

        powerup(int x,int y,int type)
        {
                xx = x;
                yy = y;
                t = type;
        }
};
```