Листинг программы

Содержимое файла «main.cpp»:

#define SDL\_MAIN\_HANDLED

#include <SDL.h>

#include <SDL\_TTF.h>

#include <SDL\_mixer.h>

#include <stdio.h>

#include <math.h>

#include <windows.h>

#include <stdlib.h>

#include <time.h>

#include <vector>

#include "blocks.h"

#include "bblock.h"

#include "shar.h"

#include "image.h"

using namespace std;

int main(int argc, char\* args[])

{

int tstart=SDL\_GetTicks();

int n=0, gameover=0, begin=0, score=0;

Mix\_Music \*music = NULL;

SDL\_Event event;

int mx, my;

double ti=1;

srand(time(0));

if (SDL\_Init(SDL\_INIT\_EVERYTHING) != 0)

{

printf("SDL\_Init Error: %s\n", SDL\_GetError());

return 1;

}

if(TTF\_Init()!=0) printf("Error ttf");

SDL\_Window \*win = SDL\_CreateWindow("Shalnoi", 20, 100, 640, 480, SDL\_WINDOW\_SHOWN);

if (!win)

{

printf("SDL\_CreateWindow Error: %s\n", SDL\_GetError());

return 1;

}

SDL\_Renderer \*ren = SDL\_CreateRenderer(win, -1, SDL\_RENDERER\_ACCELERATED | SDL\_RENDERER\_PRESENTVSYNC);

if (!ren)

{

printf("SDL\_CreateRenderer Error: %s\n", SDL\_GetError());

return 1;

}

blocks bl;

shar sh;

b\_block bblock(ren);

image D(ren);

if( Mix\_OpenAudio( 22050, MIX\_DEFAULT\_FORMAT, 2, 4096 ) == -1 )

{

return 2;

}

music = Mix\_LoadMUS("shrek.wav");

if(!music)

{

return 1;

}

SDL\_RenderClear(ren);

if(Mix\_PlayMusic(music, -1)==-1) return 1;

while(1)

{

while(1)

{

D.draw\_menu(n);

SDL\_RenderPresent(ren);

SDL\_GetMouseState(&mx, &my);

if(mx>=450 && mx<=600 && my<=130 && my>=100) n=0;

else if(mx>=450 && mx<=600 && my<=170 && my>=140) n=1;

if(SDL\_PollEvent(&event))

{

if (event.type == SDL\_MOUSEBUTTONDOWN && event.button.button==SDL\_BUTTON\_LEFT && mx<=600 && mx>=450 && my>=100 && my<=170)

{

if(n==0)

{

bl.load\_level();

break;

}

else if(n==1)

{

Mix\_FreeMusic(music);

SDL\_DestroyRenderer(ren);

SDL\_DestroyWindow(win);

Mix\_CloseAudio();

TTF\_Quit();

SDL\_Quit();

return 3;

}

}

}

}

while(!gameover)

{

if(!begin)

{

D.draw\_background();

bblock.Draw\_Block();

bl.Draw\_Blocks(ren);

sh.Draw(ren);

D.draw\_begin();

SDL\_RenderPresent(ren);

while(1)

{

if(SDL\_PollEvent(&event))

{

if(event.type == SDL\_MOUSEBUTTONDOWN && event.button.button == SDL\_BUTTON\_LEFT) break;

}

}

begin=1;

}

SDL\_GetMouseState(&mx, NULL);

if(mx>=0 && mx<=540) bblock.setpos(mx);

if(SDL\_PollEvent(&event))

{

if(event.type == SDL\_KEYDOWN && event.key.keysym.sym == SDLK\_ESCAPE)

{

D.draw\_pause();

SDL\_RenderPresent(ren);

while(1)

{

if(SDL\_PollEvent(&event))

{

if(event.type == SDL\_KEYDOWN && event.key.keysym.sym == SDLK\_RETURN)

{

gameover=1;

break;

}

else if(event.type == SDL\_KEYDOWN && event.key.keysym.sym == SDLK\_ESCAPE) break;

}

}

}

}

if(begin)

{

SDL\_ShowCursor(SDL\_DISABLE);

D.draw\_background();

if(sh.rety()>=470)

{

D.draw\_end();

SDL\_RenderPresent(ren);

while(1)

{

if(SDL\_PollEvent(&event) && event.type == SDL\_KEYDOWN && event.key.keysym.sym == SDLK\_RETURN)

{

gameover=1;

break;

}

}

}

if(sh.retx()<=630 && sh.retx()>=10 && sh.rety()>=10)

{

sh.next\_step();

for(int i=0; i<bl.bsize(); i++)

{

if((sh.rety()-10-sh.retalf()\*sh.retgy()<=bl.rety(i)+15 && sh.rety()-sh.retalf()\*sh.retgy()>=bl.rety(i)+15 || sh.rety()+10-sh.retalf()\*sh.retgy()>=bl.rety(i) && sh.rety()-sh.retalf()\*sh.retgy()<=bl.rety(i)) && sh.retx()-10<=bl.retx(i)+50 && sh.retx()+10>=bl.retx(i))

{

sh.setgy();

bl.minus(i);

if(bl.retvar(i)==0)

{

bl.del(i);

score++;

}

}

else if((sh.retx()+10+2\*sh.retgx()\*sh.retfx()>=bl.retx(i) && sh.retx()+2\*sh.retgx()\*sh.retfx()<=bl.retx(i) || sh.retx()-10+2\*sh.retgx()\*sh.retfx()<=bl.retx(i)+50 && sh.retx()+2\*sh.retgx()\*sh.retfx()>=bl.retx(i)+50) && sh.rety()-10<=bl.rety(i)+15 && sh.rety()+10>=bl.rety(i))

{

sh.setgx();

bl.minus(i);

if(bl.retvar(i)==0)

{

bl.del(i);

score++;

}

}

}

if(sh.retx()>=bblock.retx() && sh.retx()<=bblock.retx()+100 && sh.rety()+13>=bblock.rety())

{

if(sh.retx()<=bblock.retx()+52 && sh.retx()>=bblock.retx()+48)

{

sh.setfx(0);

}

else

{

sh.setfx(1);

if(sh.retx()<=bblock.retx()+50)

{

if(sh.retgx()>0) sh.setgx();

if(sh.retx()<=bblock.retx()+10) sh.setalf(2);

else if(sh.retx()>bblock.retx()+10 && sh.retx()<=bblock.retx()+30) sh.setalf(3);

else sh.setalf(4);

}

else

{

if(sh.retgx()<0) sh.setgx();

if(sh.retx()>=bblock.retx()+90) sh.setalf(2);

else if(sh.retx()<bblock.retx()+90 && sh.retx()>=bblock.retx()+70) sh.setalf(3);

else sh.setalf(4);

}

}

sh.setgy();

}

}

else

{

if((sh.retx()>=630 && sh.rety()<=10) || (sh.retx()<=10 && sh.rety()<=10) || (sh.retx()<=10 && sh.rety()>=470) || (sh.retx()>=630 && sh.rety()>=470))

{

sh.setgx();

sh.setgy();

}

else if(sh.retx()<=10 || sh.retx()>=630) sh.setgx();

else if(sh.rety()<=10 || sh.rety()>=470) sh.setgy();

sh.next\_step();

}

bblock.Draw\_Block();

bl.Draw\_Blocks(ren);

sh.Draw(ren);

SDL\_RenderPresent(ren);

}

if(bl.bsize()==0)

{

ti=SDL\_GetTicks()-tstart;

ti/=1000;

D.level\_cleared(ti/score);

SDL\_RenderPresent(ren);

while(1)

{

if(SDL\_PollEvent(&event) && event.type == SDL\_KEYDOWN && event.key.keysym.sym == SDLK\_RETURN)

{

gameover=1;

break;

}

}

}

}

SDL\_ShowCursor(SDL\_ENABLE);

begin=0;

gameover=0;

score=0;

tstart=SDL\_GetTicks();

bl.setmain();

sh.setmain();

bblock.setmain();

}

Mix\_FreeMusic(music);

SDL\_DestroyRenderer(ren);

SDL\_DestroyWindow(win);

Mix\_CloseAudio();

TTF\_Quit();

SDL\_Quit();

return 0;

}

Содержимое файла «shar.h»:

#ifndef SHAR\_H

#define SHAR\_H

#include <SDL.h>

#include <SDL\_TTF.h>

#include <SDL\_mixer.h>

#include <stdio.h>

#include <math.h>

#include <windows.h>

#include <vector>

class shar

{

SDL\_Rect RECT;

int gy, gx, alf, fx, radius;

public:

shar();

void setgy();

void setgx();

void setfx(int x);

void setalf(int x);

void setmain();

void Draw(SDL\_Renderer \*rend);

void SDL\_RenderFillCircle(SDL\_Renderer\* rend, int rad);

int rety();

int retx();

int retgx();

int retalf();

int retgy();

int retfx();

void next\_step();

};

#endif

Содержимое файла «shar.cpp»:

#include "shar.h"

#include <SDL.h>

#include <SDL\_TTF.h>

#include <SDL\_mixer.h>

#include <stdio.h>

#include <math.h>

#include <windows.h>

#include <vector>

shar::shar()

{

RECT.x=320;

RECT.y=448;

gy=1;

gx=2;

radius=0;

alf=4;

fx=0;

}

void shar::Draw(SDL\_Renderer \*rend)

{

radius=10;

SDL\_SetRenderDrawColor(rend, 192, 192, 192, 255);

SDL\_RenderFillCircle(rend, radius);

SDL\_SetRenderDrawColor(rend, 255, 0, 0, 255);

radius=8;

SDL\_RenderFillCircle(rend, radius);

}

void shar::setgy()

{

gy=-gy;

}

void shar::setgx()

{

gx=-gx;

}

int shar::retfx()

{

return fx;

}

int shar::retalf()

{

return alf;

}

void shar::setfx(int x)

{

fx=x;

}

void shar::setalf(int x)

{

alf=x;

}

void shar::SDL\_RenderFillCircle(SDL\_Renderer\* rend, int rad)

{

int x = rad;

int y = 0;

int radiusError = 1 - x;

while (x >= y)

{

SDL\_RenderDrawLine(rend, x + RECT.x, y + RECT.y, -x + RECT.x, y + RECT.y);

SDL\_RenderDrawLine(rend, y + RECT.x, x + RECT.y, -y + RECT.x, x + RECT.y);

SDL\_RenderDrawLine(rend, -x + RECT.x, -y + RECT.y, x + RECT.x, -y + RECT.y);

SDL\_RenderDrawLine(rend, -y + RECT.x, -x + RECT.y, y + RECT.x, -x + RECT.y);

y++;

if (radiusError < 0)

radiusError += 2 \* y + 1;

else

{

x--;

radiusError += 2 \* (y - x + 1);

}

}

}

int shar::rety()

{

return RECT.y;

}

int shar::retx()

{

return RECT.x;

}

int shar::retgx()

{

return gx;

}

int shar::retgy()

{

return gy;

}

void shar::setmain()

{

RECT.x=320;

RECT.y=448;

gy=1;

gx=2;

radius=0;

alf=4;

fx=0;

}

void shar::next\_step()

{

RECT.x+=2\*gx\*fx;

if(!fx) RECT.y-=alf\*gy\*(fx+1);

else RECT.y-=alf\*gy;

}

Содержимое файла «bblock.h»:

#ifndef BBLOCK\_H

#define BBLOCK\_H

#include <SDL.h>

#include <SDL\_TTF.h>

#include <SDL\_mixer.h>

#include <stdio.h>

#include <math.h>

#include <windows.h>

#include <vector>

class b\_block

{

SDL\_Rect nblock;

SDL\_Surface \*img;

SDL\_Texture \*timg;

SDL\_Renderer \*render;

public:

b\_block(SDL\_Renderer \*);

~b\_block();

int rety();

void setmain();

int retx();

void Draw\_Block();

void setpos(int);

};

#endif

Содержимое файла «bblock.cpp»:

#include "bblock.h"

#include <SDL.h>

#include <SDL\_TTF.h>

#include <SDL\_mixer.h>

#include <stdio.h>

#include <math.h>

#include <windows.h>

#include <vector>

b\_block::b\_block(SDL\_Renderer \*ren)

{

render=ren;

img=SDL\_LoadBMP("2621.bmp");

SDL\_SetColorKey(img, 1, SDL\_MapRGB(img->format, 255, 255, 255));

timg=SDL\_CreateTextureFromSurface(render, img);

SDL\_FreeSurface(img);

nblock.x=270;

nblock.y=460;

nblock.w=100;

nblock.h=10;

}

b\_block::~b\_block()

{

SDL\_DestroyTexture(timg);

}

int b\_block::rety()

{

return nblock.y;

}

void b\_block::setmain()

{

nblock.x=270;

nblock.y=460;

nblock.w=100;

nblock.h=10;

}

int b\_block::retx()

{

return nblock.x;

}

void b\_block::Draw\_Block()

{

SDL\_RenderCopy(render, timg, NULL, &nblock);

}

void b\_block::setpos(int x)

{

nblock.x=x;

}

Содержимое файла «blocks.h»:

#ifndef BLOCKS\_H

#define BLOCKS\_H

#include <SDL.h>

#include <SDL\_TTF.h>

#include <SDL\_mixer.h>

#include <stdio.h>

#include <math.h>

#include <windows.h>

#include <vector>

using namespace std;

class blocks

{

struct qwe

{

SDL\_Rect r;

int var;

};

vector <qwe> a;

qwe temp;

int map[26][13];

public:

blocks();

void Draw\_Blocks(SDL\_Renderer \*ren);

int rety(int i);

int retx(int i);

int retvar(int i);

void load\_level();

void minus(int i);

void setmain();

int bsize();

void del(int i);

};

#endif

Содержимое файла «blocks.cpp»:

#include <SDL.h>

#include <SDL\_TTF.h>

#include <SDL\_mixer.h>

#include <stdio.h>

#include <math.h>

#include <windows.h>

#include <vector>

#include <stdlib.h>

#include <time.h>

#include <fstream>

#include "blocks.h"

using namespace std;

blocks::blocks()

{

for(int i=0 ; i<26 ; i++)

{

for(int j=0 ; j<13 ; j++)

{

map[i][j]=0;

}

}

}

int blocks::rety(int i)

{

return a[i].r.y;

}

int blocks::retx(int i)

{

return a[i].r.x;

}

int blocks::bsize()

{

return (int)a.size();

}

void blocks::del(int i)

{

a.erase(a.begin()+i);

}

void blocks::setmain()

{

a.clear();

}

void blocks::load\_level()

{

int i,j, k;

temp.r.h=15;

temp.r.w=50;

for(i=0 ; i<15 ; i++)

{

for(j=0 ; j<13 ; j++)

{

k=rand()%5;

if(k<=2) map[i][j]=k;

else map[i][j]=0;

}

}

for(i=0 ; i<26 ; i++)

{

for(j=0 ; j<13 ; j++)

{

if(map[i][j]==1)

{

temp.r.y=i\*15;

temp.r.x=j\*50;

temp.var=1;

a.push\_back(temp);

}

else if(map[i][j]==2)

{

temp.r.y=i\*15;

temp.r.x=j\*50;

temp.var=2;

a.push\_back(temp);

}

}

}

}

int blocks::retvar(int i)

{

return a[i].var;

}

void blocks::minus(int i)

{

a[i].var--;

}

void blocks::Draw\_Blocks(SDL\_Renderer \*ren)

{

SDL\_Rect r;

for(int i=0 ; i<(int)a.size(); i++)

{

if(a[i].var==1) SDL\_SetRenderDrawColor(ren, 202, 255, 112, 255);

else if(a[i].var==2) SDL\_SetRenderDrawColor(ren, 255, 202, 112, 255);

SDL\_RenderFillRect(ren, &a[i].r);

}

for(int i=0 ; i<(int)a.size(); i++)

{

if(a[i].var==1) SDL\_SetRenderDrawColor(ren, 0, 255, 0, 255);

else if(a[i].var==2) SDL\_SetRenderDrawColor(ren, 255, 0, 0, 255);

r.x=a[i].r.x+2;

r.y=a[i].r.y+2;

r.h=13;

r.w=48;

SDL\_RenderFillRect(ren, &r);

}

}

Содержимое файла «image.h»:

#ifndef IMAGE\_H

#define IMAGE\_H

#include <SDL.h>

#include <SDL\_TTF.h>

#include <stdio.h>

#include <windows.h>

#include <stdlib.h>

class image

{

SDL\_Surface \*background, \*menu, \*text;

SDL\_Texture \*tbackground, \*tmenu, \*ftext;

SDL\_Renderer \*render;

public:

image(SDL\_Renderer \*ren);

~image();

void draw\_begin();

void draw\_pause();

void draw\_background();

void level\_cleared(double a);

void draw\_end();

void draw\_menu(int menu);

};

#endif

Содержимое файла «image.cpp»:

#include <SDL.h>

#include <SDL\_TTF.h>

#include <stdio.h>

#include <windows.h>

#include <stdlib.h>

#include "image.h"

image::image(SDL\_Renderer \*ren)

{

render=ren;

background=SDL\_LoadBMP("back.bmp");

tbackground=SDL\_CreateTextureFromSurface(ren, background);

SDL\_FreeSurface(background);

menu=SDL\_LoadBMP("menu.bmp");

tmenu=SDL\_CreateTextureFromSurface(ren, menu);

SDL\_FreeSurface(menu);

}

image::~image()

{

SDL\_DestroyTexture(tmenu);

SDL\_DestroyTexture(tbackground);

}

void image::draw\_begin()

{

TTF\_Font \*fnt=NULL;

SDL\_Rect r;

SDL\_Color color;

fnt=TTF\_OpenFont("Casper\_B.ttf", 30);

if (!fnt) printf("Unable to load font \n");

color.r=255;

color.g=185;

color.b=15;

text=TTF\_RenderText\_Blended(fnt, "Press left mouse button to begin", color);

ftext=SDL\_CreateTextureFromSurface(render, text);

r.x=130;

r.y=200;

r.h=text->h;

r.w=text->w;

SDL\_RenderCopy(render, ftext, NULL, &r);

SDL\_FreeSurface(text);

SDL\_DestroyTexture(ftext);

TTF\_CloseFont(fnt);

}

void image::draw\_pause()

{

TTF\_Font \*fnt=NULL;

SDL\_Rect r;

SDL\_Color color;

fnt=TTF\_OpenFont("Casper\_B.ttf", 30);

if (!fnt) printf("Unable to load font \n");

color.r=255;

color.g=185;

color.b=15;

text=TTF\_RenderText\_Blended(fnt, "Game is paused", color);

ftext=SDL\_CreateTextureFromSurface(render, text);

r.x=230;

r.y=150;

r.h=text->h;

r.w=text->w;

SDL\_RenderCopy(render, ftext, NULL, &r);

text=TTF\_RenderText\_Blended(fnt, "Esc - continue", color);

ftext=SDL\_CreateTextureFromSurface(render, text);

r.x=230;

r.y=180;

r.h=text->h;

r.w=text->w;

SDL\_RenderCopy(render, ftext, NULL, &r);

text=TTF\_RenderText\_Blended(fnt, "Return - quit", color);

ftext=SDL\_CreateTextureFromSurface(render, text);

r.x=230;

r.y=210;

r.h=text->h;

r.w=text->w;

SDL\_RenderCopy(render, ftext, NULL, &r);

SDL\_FreeSurface(text);

SDL\_DestroyTexture(ftext);

TTF\_CloseFont(fnt);

}

void image::draw\_background()

{

SDL\_RenderCopy(render, tbackground, NULL, NULL);

}

void image::level\_cleared(double a)

{

TTF\_Font \*fnt=NULL;

SDL\_Rect r;

SDL\_Color color;

char buf[200];

fnt=TTF\_OpenFont("Casper\_B.ttf", 20);

if (!fnt) printf("Unable to load font \n");

color.r=255;

color.g=185;

color.b=15;

sprintf(buf, "Level cleared, your rank: %.2lf, press Enter to return in main menu", a);

text=TTF\_RenderText\_Blended(fnt, buf, color);

ftext=SDL\_CreateTextureFromSurface(render, text);

r.x=40;

r.y=130;

r.h=text->h;

r.w=text->w;

SDL\_RenderCopy(render, ftext, NULL, &r);

SDL\_FreeSurface(text);

SDL\_DestroyTexture(ftext);

TTF\_CloseFont(fnt);

}

void image::draw\_end()

{

TTF\_Font \*fnt=NULL;

SDL\_Rect r;

SDL\_Color color;

fnt=TTF\_OpenFont("Casper\_B.ttf", 20);

if (!fnt) printf("Unable to load font \n");

color.r=255;

color.g=185;

color.b=15;

text=TTF\_RenderText\_Blended(fnt, "Game Over, press Enter to return in main menu", color);

ftext=SDL\_CreateTextureFromSurface(render, text);

r.x=120;

r.y=130;

r.h=text->h;

r.w=text->w;

SDL\_RenderCopy(render, ftext, NULL, &r);

SDL\_FreeSurface(text);

SDL\_DestroyTexture(ftext);

TTF\_CloseFont(fnt);

}

void image::draw\_menu(int menu)

{

TTF\_Font \*fnt=NULL;

SDL\_Rect r;

SDL\_Color color;

fnt=TTF\_OpenFont("Casper\_B.ttf", 30);

if (!fnt) printf("Unable to load font \n");

SDL\_RenderCopy(render, tmenu, NULL, NULL);

if(menu==0)

{

color.r=255;

color.g=185;

color.b=15;

text=TTF\_RenderText\_Blended(fnt, "New Game", color);

ftext=SDL\_CreateTextureFromSurface(render, text);

r.x=450;

r.y=100;

r.h=text->h;

r.w=text->w;

SDL\_RenderCopy(render, ftext, NULL, &r);

color.r=255;

color.g=52;

color.b=179;

text=TTF\_RenderText\_Blended(fnt, "Quit", color);

ftext=SDL\_CreateTextureFromSurface(render, text);

r.x=450;

r.y=140;

r.h=text->h;

r.w=text->w;

SDL\_RenderCopy(render, ftext, NULL, &r);

SDL\_FreeSurface(text);

}

else if(menu==1)

{

color.r=255;

color.g=52;

color.b=179;

text=TTF\_RenderText\_Blended(fnt, "New Game", color);

ftext=SDL\_CreateTextureFromSurface(render, text);

r.x=450;

r.y=100;

r.h=text->h;

r.w=text->w;

SDL\_RenderCopy(render, ftext, NULL, &r);

color.r=255;

color.g=185;

color.b=15;

text=TTF\_RenderText\_Blended(fnt, "Quit", color);

ftext=SDL\_CreateTextureFromSurface(render, text);

r.x=450;

r.y=140;

r.h=text->h;

r.w=text->w;

SDL\_RenderCopy(render, ftext, NULL, &r);

SDL\_FreeSurface(text);

}

SDL\_DestroyTexture(ftext);

TTF\_CloseFont(fnt);

}

Содержимое файла «shar.h»:

#ifndef SHAR\_H

#define SHAR\_H

#include <SDL.h>

#include <SDL\_TTF.h>

#include <SDL\_mixer.h>

#include <stdio.h>

#include <math.h>

#include <windows.h>

#include <vector>

class shar

{

SDL\_Rect RECT;

int gy, gx, alf, fx, radius;

public:

shar();

void setgy();

void setgx();

void setfx(int x);

void setalf(int x);

void setmain();

void Draw(SDL\_Renderer \*rend);

void SDL\_RenderFillCircle(SDL\_Renderer\* rend, int rad);

int rety();

int retx();

int retgx();

int retalf();

int retgy();

int retfx();

void next\_step();

};

#endif

Содержимое файла «shar.cpp»:

#include "shar.h"

#include <SDL.h>

#include <SDL\_TTF.h>

#include <SDL\_mixer.h>

#include <stdio.h>

#include <math.h>

#include <windows.h>

#include <vector>

shar::shar()

{

RECT.x=320;

RECT.y=448;

gy=1;

gx=2;

radius=0;

alf=4;

fx=0;

}

void shar::Draw(SDL\_Renderer \*rend)

{

radius=10;

SDL\_SetRenderDrawColor(rend, 192, 192, 192, 255);

SDL\_RenderFillCircle(rend, radius);

SDL\_SetRenderDrawColor(rend, 255, 0, 0, 255);

radius=8;

SDL\_RenderFillCircle(rend, radius);

}

void shar::setgy()

{

gy=-gy;

}

void shar::setgx()

{

gx=-gx;

}

int shar::retfx()

{

return fx;

}

int shar::retalf()

{

return alf;

}

void shar::setfx(int x)

{

fx=x;

}

void shar::setalf(int x)

{

alf=x;

}

void shar::SDL\_RenderFillCircle(SDL\_Renderer\* rend, int rad)

{

int x = rad;

int y = 0;

int radiusError = 1 - x;

while (x >= y)

{

SDL\_RenderDrawLine(rend, x + RECT.x, y + RECT.y, -x + RECT.x, y + RECT.y);

SDL\_RenderDrawLine(rend, y + RECT.x, x + RECT.y, -y + RECT.x, x + RECT.y);

SDL\_RenderDrawLine(rend, -x + RECT.x, -y + RECT.y, x + RECT.x, -y + RECT.y);

SDL\_RenderDrawLine(rend, -y + RECT.x, -x + RECT.y, y + RECT.x, -x + RECT.y);

y++;

if (radiusError < 0)

radiusError += 2 \* y + 1;

else

{

x--;

radiusError += 2 \* (y - x + 1);

}

}

}

int shar::rety()

{

return RECT.y;

}

int shar::retx()

{

return RECT.x;

}

int shar::retgx()

{

return gx;

}

int shar::retgy()

{

return gy;

}

void shar::setmain()

{

RECT.x=320;

RECT.y=448;

gy=1;

gx=2;

radius=0;

alf=4;

fx=0;

}

void shar::next\_step()

{

RECT.x+=2\*gx\*fx;

if(!fx) RECT.y-=alf\*gy\*(fx+1);

else RECT.y-=alf\*gy;

}