```
unit GUI;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,
  Dialogs, OpenGL, DGLUT, Textures, Mesh, Resurce,
ExtCtrls,Command,GFonts,math,
  GameLavels, Vcl.MPlayer, unit2;

 const
  Pi =3.14;
  size = 20;
type
   TVector = record  //Вектор
   X,Y,Z:GLfloat;
   end;


   TCamera = record
   Pos: Tvector;        //Позиция камеры
   PhiY: single;        //вертикальный улол поворота
камеры
   PhiX: single;        //горизонтальный
   Speed: glFloat;      //Скорость камеры
   end;


  TForm1 = class(TForm)
    DrawGrGL: TTimer;
    PhizProcess: TTimer;
    MediaPlayer1: TMediaPlayer;
    procedure PhizProcessTimer(Sender: TObject);
    procedure Timer2Timer(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormPaint(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
  HRC : HGLRC;
  MouseMove1: boolean;
  glLightPos: array[0..3] of glFloat = (0,0,100,1);
  TempX1,TempY1,SingX,SingY,TempX2,TempY2,k :
integer;
  Point: Tpoint;
  FPS,FP:integer;
  P:FPlayer;
  TObj : TGLMultyMesh;
   Cursor1,W,S,D,FontF: Uint;

   provrka,WX,WY:integer;
   cetest:byte;
   Sky,Grass,board:BBoxT;
   Blocks:array[0..255] of BBoxT;
   Ttextobj : array[0..256] of Uint;
   Blocs:integer;
   bclick,lmb,rmb:boolean;
   MyBlock:byte;
   Mass:array[0..1024,0..1024,0..1024] of Byte;


   MenuStay:integer;
/// функция для установки текущей текстуры по
идентификатору texture
procedure glBindTexture(target: GLenum; texture: GLu-
int); stdcall; external opengl32;
function Summae(a,b:real):real;  external  'MIZEENG';
implementation

{$R *.dfm}
//действия с блоками
procedure BoxManager();
  var x,y,z:real;
    dist, VX,VY,VZ,oldX,oldY,oldZ:integer;
begin
  if (lmb=true) or (rmb=true) then //считываем началь-
ное положение головы персонажа
  begin
  x:=p.X-size/2;
  y:=p.Y+p.h/2-size/2;
  z:=p.Z-size/2;
  dist:=0;
  while dist<100 do   //чем больше dist тем дальше
можно создавать кубики
    begin
    dist:=dist+1;
    if abs(tan(p.AngleY/180*Pi))>1 then
    begin
    x:=x-
sin(p.angleX/180*pi)/abs(tan(p.AngleY/180*Pi));
VX:=round(x/size);

y:=y+tan(p.AngleY/180*Pi)/abs(tan(p.AngleY/180*Pi));
VY:=round(y/size);
    z:=z-
cos(p.angleX/180*pi)/abs(tan(p.AngleY/180*Pi));
VZ:=round(z/size);
    end
    else
    begin
    x:=x-sin(p.angleX/180*pi); VX:=round(x/size);
    y:=y+tan(p.AngleY/180*Pi); VY:=round(y/size);
    z:=z-cos(p.angleX/180*pi); VZ:=round(z/size);
    end;

   if (Check(VX,VY,VZ)<>0) then begin  //проверка
если столкнулись с боксом
      if lmb=true then mass[VX,VY,VZ]:=0;           //если
левая то удаляем кубик по его координатам
      if (rmb=true) and (Check(oldX,oldY,oldZ)=0) then
begin  //если правая ставим выбранный блок на наве-
дённый координат
      if (dist>60) and ((tan(p.AngleY/180*Pi)<-8) or
(tan(p.AngleY/180*Pi)>4)) then
mass[oldX,oldY,oldZ]:=MyBlock;
```

```
    if (dist>25) and (tan(p.AngleY/180*Pi)>=-8) and
(tan(p.AngleY/180*Pi)<=4) then
mass[oldX,oldY,oldZ]:=MyBlock;
    end;

    lmb:=false;
    rmb:=false;
   end;
   oldX:=VX; oldY:=VY; oldZ:=VZ;
   end;
 end;
end;
//формат пикселей
procedure SetDCPixelFormat ( hdc : HDC );
 var
  pfd : TPixelFormatDescriptor;
  nPixelFormat : Integer;
 begin
  FillChar (pfd, SizeOf (pfd), 0);
  pfd.dwFlags  := PFD_DRAW_TO_WINDOW or
PFD_SUPPORT_OPENGL or PFD_DOUBLEBUFFER;
   nPixelFormat := ChoosePixelFormat (hdc, @pfd);
  SetPixelFormat (hdc, nPixelFormat, @pfd);
 end;

//создание мира
procedure TForm1.FormCreate(Sender: TObject);
var
 I,J,Y,G,K:integer;
 U:TBitmap;
 Color:TColor;
 R:byte;
begin
ShowCursor(false);
MediaPlayer1.filename:=('DATA\Music.wav');
Mediaplayer1.Play;
U:= TBitmap.Create;
 U.LoadFromFile('Data\Map.bmp');
for I := 0 to 127 do
  for J := 0 to 127 do
   for Y := 0 to 127 do
   begin
     Mass[I,J,Y]:=0;
   end;

 for G := 0 to 127 do
  for K := 0 to 127 do
     begin
       Color:=U.Canvas.Pixels[K,G];
       R:=GetBValue(Color);

       for I := 0 to Round(R/4)-1 do
        begin
          If I<Round(R/4)-1 then Mass[K,I,G] := 9 ;
          If I=Round(R/4)-1 then Mass[K,I,G] := 1 ;
        end;
end;
U.Free;

  SetDCPixelFormat(Canvas.Handle);
  hrc := wglCreateContext(Canvas.Handle);
  wglMakeCurrent(Canvas.Handle, hrc);

  Load_First_Tex_settings();
  Load_Game_Textures();
  Load_Game_Models();

{ for I := 0 to 19 do
  for J := 0 to 19 do
   for Y := 0 to 19 do
   begin
     K:=random(150);
     If (K mod 140 <> 0)  then Mass[I,J,Y]:=0;
     if (K mod 140 = 0)  then Mass[I,J,Y]:=1;
     if J=0 then Mass[I,J,Y]:=1;
   end; }
     P.create(400,700,400);
    MyBlock:=1;
  provrka:=2;
  MenuStay:=0;
end;
//отрисовка
procedure TForm1.FormPaint(Sender: TObject);
var i : integer;
    j : integer;
    l : integer;
begin
  IdenTifiWindow(ClientWidth, ClientHeight);
 Active_UnActive_system();
    if (GetAsyncKeyState(VK_ESCAPE)<>0) then
     begin
       ShowCursor(true);
       form2.show;
       form1.close;
     end;
      if (GetAsyncKeyState(VK_LButton)=0) and (GetA-
syncKeyState(VK_RButton)=0) and (bclick=true) then
       begin
         bclick:=false;
       end;
GluLookAt(P.X,P.Y+P.h/2,P.Z,   P.X-
sin(P.angleX/180*Pi),
    P.Y+P.h/2+tan(P.AngleY/180*Pi),P.Z-
cos(P.angleX/180*Pi),0,1,0);
MainGame(ClientWidth, ClientHeight);

FP:=FP+1;
SwapBuffers(Canvas.Handle); //для обновления со-
держимого холста на экране
end;

//движение персонажа
procedure TForm1.PhizProcessTimer(Sender: TObject);
var I,J,Y,O:integer;
dir:real;
begin
    if (Commande=false) then
      begin
        Mouse_Move();
      end;
```

```
   P:=Key_Move(P);
 P.dy:=P.Dy-0.2;
 if P.dy<-3 then P.dy:=-3;

for I := 1 to 127 do
  for J := 1 to 127 do
   for Y := 1 to 127 do
   begin
        dir:=sqrt(sqr(p.X-size*i+size/2)+sqr(p.Y-
size*j+size/2)+sqr(p.z-size*y+size/2));

     If (Mass[I,J,Y]<>0) and (abs(dir)<80) then begin
P:=NueThon(size*i+size/2, size*j+size/2, size*y+size/2,
     size/2,size/2,size/2,P);
     end;
   end;

if P.Colizt>0 then begin
P.onGround:=true;
end else begin
  P.onGround:=false;
end;

 p.Colizt:=0;
 P.X:=P.X+P.dx;
 P.Y:=P.Y+P.dy;
 P.Z:=P.Z+P.dz;

 BoxManager();
       P.dx:=0;
       P.dz:=0;
end;

procedure TForm1.Timer2Timer(Sender: TObject);
begin
Form1.Handle;
end;

end.
unit Unit2;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils,
System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls,
Vcl.ExtCtrls,
  Vcl.Buttons, Vcl.ComCtrls, Vcl.Menus,
Vcl.Imaging.pngimage, shellAPI;

type
  TForm2 = class(TForm)
    SpeedButton1: TSpeedButton;
    ProgressBar1: TProgressBar;
    Timer1: TTimer;
    Image1: TImage;
    SpeedButton2: TSpeedButton;
    MainMenu1: TMainMenu;
    N3: TMenuItem;
    N4: TMenuItem;
    N5: TMenuItem;
    procedure FormCreate(Sender: TObject);
    procedure SpeedButton1Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure SpeedButton2Click(Sender: TObject);
    procedure N4Click(Sender: TObject);
    procedure N5Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;
implementation
{$R *.dfm}
uses GUI;



procedure TForm2.FormCreate(Sender: TObject);
begin
showcursor(true);
Image1.Align := alClient;
Image1.Stretch := True;
end;

procedure TForm2.N4Click(Sender: TObject);
begin
ShellExecute(0, PChar ('Open'), PChar ('spavka2.chm'),
nil, nil, SW_SHOW);
end;

procedure TForm2.N5Click(Sender: TObject);
begin
close;
end;

procedure TForm2.SpeedButton1Click(Sender:
TObject);
begin
showcursor(false);
Form1.Show;
Form2.Hide;
end;

procedure TForm2.SpeedButton2Click(Sender:
TObject);
begin
close;
end;

procedure TForm2.Timer1Timer(Sender: TObject);
begin
ProgressBar1.Position:=Progressbar1.Position+20;
if ProgressBar1.position=100 then
begin
```

```
Timer1.enabled:=False;
SpeedButton1.Visible:=true;
SpeedButton2.Visible:=true;
ProgressBar1.Visible:=false;
N3.Visible:=true;
N4.Visible:=true;
N5.Visible:=true;
end;
end;

end.
unit Resurce;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,
  Dialogs, OpenGL, DGLUT, Textures,Mesh, com-
mand,GFonts;



procedure Load_First_Tex_settings();
procedure Load_Game_Textures();
procedure Load_Game_Models();

Procedure RenderBox();
procedure Render-
Sprite(PX,PY,SX,SY,Rot:real;Pict:Uint);
function SpriteButton(PX,PY,SX,SY:real;Pict:Uint;
SN,SP,Key:integer):integer;
procedure RenderFlore(x,y,size:integer; Texture:Uint);
procedure RenderGMSModel(x,y,z,size:integer; mod-
el:TGLMultyMesh; Texture:Uint);

implementation
uses GUI;
//отрисовку спрайта на экране
procedure Render-
Sprite(PX,PY,SX,SY,Rot:real;Pict:Uint);
begin

  glTranslatef(PX,PY,0);
  glRotatef(Rot,0,0,1);
  glPushMatrix;  //панель нижняя с блоками
    //  glColor(255,255,255);
  glBindTexture(GL_TEXTURE_2D, Pict);
   glBegin(GL_Quads);
     glTexCoord2d(0.0, 1.0);  glVertex2d(-sx,-sy);
     glTexCoord2d(0.0, 0.0);  glVertex2d(-sx,+sy);
     glTexCoord2d(1.0, 0.0);  glVertex2d(+sx,+sy);
     glTexCoord2d(1.0, 1.0);  glVertex2d(+sx,-sy);
   glEnd;
      // glColor(255,255,255);
  GLpOPmATRIX;
  glRotatef(-Rot,0,0,1);
  glTranslatef(-PX,-PY,0);
end;

//проверка положения курсора относительно кнопки
и определение действия при нажатии на кнопку мы-
шью
function SpriteButton(PX,PY,SX,SY:real;Pict:Uint;
SN,SP,Key:integer):integer;
var X,Y:integer;
begin
  GetCursorPos(Point);
  X:=-Form1.Left +Point.X;
  Y:=-Form1.Top-15  +point.Y;
  RenderSprite(PX,PY,SX,SY,0,Pict);
  if (Point.X>Form1.Left) and
(Point.X<Form1.Left+Form1.Width) and
     (point.Y>Form1.Top-15) and
(point.Y<Form1.Top+Form1.Height) and
      (X<px+sx) and (X>px-sx) and
      (Y<py+sy) and (Y>py-sy) then begin
      if(GetAsyncKeyState(VK_LBUTTON)<>0) and
(MBL=false) then
        begin
          result:=Key;
          MBL:=true;
        end else   begin
          result:=SP;
        end;
     end else begin
       result:=SN;

     end;



end;
//параметры текстур перед их использованием
procedure Load_First_Tex_settings();
begin
  glEnable(GL_DEPTH_TEST); // включаем проверку
разрешения фигур (впереди стоящая закрывает фи-
гуру за ней)
  glDepthFunc(GL_LEQUAL);  //тип проверки
  glEnable(GL_TEXTURE_2D);   //Вкелючаем ржим
наложения текстур
  glEnable(GL_ALPHA_TEST);     //Разрешаем альфа
тест (прозрачность текстур)
  glAlphaFunc(GL_GREATER,0.025);
  glEnable (GL_BLEND);         //Включаем режим
смешивания цветов
  glDepthMask(GL_True);

  glTexEnvi( GL_TEXTURE_ENV,
GL_TEXTURE_ENV_MODE, GL_REPLACE );



  glBlendFunc (GL_SRC_ALPHA,
GL_ONE_MINUS_SRC_ALPHA) ; //Тип смешивания
  glTexParameter (GL_TEXTURE_2D,
GL_TEXTURE_MAG_FILTER, GL_NEAREST);
//Параметры наложения текстуры
  glTexParameter (GL_TEXTURE_2D,
GL_TEXTURE_MIN_FILTER, GL_NEAREST);
//Параметры наложения текстуры
```

```
  glTexParameter (GL_TEXTURE_2D,
GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
  glTexParameter (GL_TEXTURE_2D,
GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
end;
//загрузка текстур с каждой стороны
procedure Load_Game_Textures();
begin
   LoadTexture('TEXTURES\skybox\left.tga',W,false);
   LoadTexture('TEXTURES\skybox\right.tga',S,false);
   LoadTex-
ture('TEXTURES\skybox\bottom.tga',D,false);

  LoadTexture('GRE\Stex\SCursor.tga', Cursor1, false);
  LoadTexture('GRE\Stex\F.tga', FontF, false);


    LoadTexture('data\cursor.tga',Ttextobj[0],false);

    LoadTexture('data\image\1.tga',Ttextobj[1],false);
    LoadTexture('data\image\2.tga',Ttextobj[2],false);
    LoadTexture('data\image\3.tga',Ttextobj[3],false);
    LoadTexture('data\image\4.tga',Ttextobj[4],false);
    LoadTexture('data\image\5.tga',Ttextobj[5],false);
    LoadTexture('data\image\6.tga',Ttextobj[6],false);
    LoadTexture('data\image\7.tga',Ttextobj[7],false);
    LoadTexture('data\image\8.tga',Ttextobj[8],false);
    LoadTexture('data\image\9.tga',Ttextobj[9],false);
    LoadTexture('data\image\10.tga',Ttextobj[10],false);

    LoadTex-
ture('data\image\enter.tga',Ttextobj[200],false);

    LoadTex-
ture('data\Grass\LRBT.tga',Blocks[0].Texture[0],false);
    LoadTex-
ture('data\Grass\LRBT.tga',Blocks[0].Texture[1],false);
    LoadTex-
ture('data\Grass\LRBT.tga',Blocks[0].Texture[2],false);
    LoadTex-
ture('data\Grass\LRBT.tga',Blocks[0].Texture[3],false);
    LoadTex-
ture('data\Grass\dn.tga',Blocks[0].Texture[4],false);
    LoadTex-
ture('data\Grass\up.tga',Blocks[0].Texture[5],false);


    LoadTex-
ture('data\board\bt.tga',Blocks[1].Texture[0],false);
    LoadTex-
ture('data\board\bt.tga',Blocks[1].Texture[1],false);
    LoadTex-
ture('data\board\bt.tga',Blocks[1].Texture[2],false);
    LoadTex-
ture('data\board\bt.tga',Blocks[1].Texture[3],false);
    LoadTex-
ture('data\board\bt.tga',Blocks[1].Texture[4],false);
    LoadTex-
ture('data\board\bt.tga',Blocks[1].Texture[5],false);

    LoadTex-
ture('data\Case\cb.tga',Blocks[2].Texture[0],false);
    LoadTex-
ture('data\Case\cb.tga',Blocks[2].Texture[1],false);
    LoadTex-
ture('data\Case\cf.tga',Blocks[2].Texture[2],false);
    LoadTex-
ture('data\Case\cb.tga',Blocks[2].Texture[3],false);
    LoadTex-
ture('data\Case\cuw.tga',Blocks[2].Texture[4],false);
    LoadTex-
ture('data\Case\cuw.tga',Blocks[2].Texture[5],false);

    LoadTex-
ture('data\brick\br.tga',Blocks[3].Texture[0],false);
    LoadTex-
ture('data\brick\br.tga',Blocks[3].Texture[1],false);
    LoadTex-
ture('data\brick\br.tga',Blocks[3].Texture[2],false);
    LoadTex-
ture('data\brick\br.tga',Blocks[3].Texture[3],false);
    LoadTex-
ture('data\brick\br.tga',Blocks[3].Texture[4],false);
    LoadTex-
ture('data\brick\br.tga',Blocks[3].Texture[5],false);

    LoadTex-
ture('data\stone\Brick.tga',Blocks[4].Texture[0],false);
    LoadTex-
ture('data\stone\Brick.tga',Blocks[4].Texture[1],false);
    LoadTex-
ture('data\stone\Brick.tga',Blocks[4].Texture[2],false);
    LoadTex-
ture('data\stone\Brick.tga',Blocks[4].Texture[3],false);
    LoadTex-
ture('data\stone\Brick.tga',Blocks[4].Texture[4],false);
    LoadTex-
ture('data\stone\Brick.tga',Blocks[4].Texture[5],false);

    LoadTex-
ture('data\tree\cora.tga',Blocks[5].Texture[0],false);
    LoadTex-
ture('data\tree\cora.tga',Blocks[5].Texture[1],false);
    LoadTex-
ture('data\tree\cora.tga',Blocks[5].Texture[2],false);
    LoadTex-
ture('data\tree\cora.tga',Blocks[5].Texture[3],false);
    LoadTex-
ture('data\tree\corat.tga',Blocks[5].Texture[4],false);
    LoadTex-
ture('data\tree\corat.tga',Blocks[5].Texture[5],false);

    LoadTex-
ture('data\tree\list.tga',Blocks[6].Texture[0],false);
    LoadTex-
ture('data\tree\list.tga',Blocks[6].Texture[1],false);
    LoadTex-
ture('data\tree\list.tga',Blocks[6].Texture[2],false);
    LoadTex-
ture('data\tree\list.tga',Blocks[6].Texture[3],false);
```

```
   LoadTex-
ture('data\tree\list.tga',Blocks[6].Texture[4],false);
   LoadTex-
ture('data\tree\list.tga',Blocks[6].Texture[5],false);

   LoadTex-
ture('data\sand\sand.tga',Blocks[7].Texture[0],false);
   LoadTex-
ture('data\sand\sand.tga',Blocks[7].Texture[1],false);
   LoadTex-
ture('data\sand\sand.tga',Blocks[7].Texture[2],false);
   LoadTex-
ture('data\sand\sand.tga',Blocks[7].Texture[3],false);
   LoadTex-
ture('data\sand\sand.tga',Blocks[7].Texture[4],false);
   LoadTex-
ture('data\sand\sand.tga',Blocks[7].Texture[5],false);

   LoadTex-
ture('data\Grass\dn.tga',Blocks[8].Texture[0],false);
   LoadTex-
ture('data\Grass\dn.tga',Blocks[8].Texture[1],false);
   LoadTex-
ture('data\Grass\dn.tga',Blocks[8].Texture[2],false);
   LoadTex-
ture('data\Grass\dn.tga',Blocks[8].Texture[3],false);
   LoadTex-
ture('data\Grass\dn.tga',Blocks[8].Texture[4],false);
   LoadTex-
ture('data\Grass\dn.tga',Blocks[8].Texture[5],false);

   LoadTex-
ture('data\window\wind.tga',Blocks[9].Texture[0],false);
   LoadTex-
ture('data\window\wind.tga',Blocks[9].Texture[1],false);
   LoadTex-
ture('data\window\wind.tga',Blocks[9].Texture[2],false);
   LoadTex-
ture('data\window\wind.tga',Blocks[9].Texture[3],false);
   LoadTex-
ture('data\window\wind.tga',Blocks[9].Texture[4],false);
   LoadTex-
ture('data\window\wind.tga',Blocks[9].Texture[5],false);


   LoadTex-
tu-
re('data\Blocktex\bluecloud_rt.jpg',Sky.Texture[0],false);
   LoadTex-
tu-
re('data\Blocktex\bluecloud_lf.jpg',Sky.Texture[1],false);
   LoadTex-
tu-
re('data\Blocktex\bluecloud_bk.jpg',Sky.Texture[2],false
);
   LoadTex-
tu-
re('data\Blocktex\bluecloud_ft.jpg',Sky.Texture[3],false);
   LoadTex-
tu-
re('data\Blocktex\bluecloud_dn.jpg',Sky.Texture[4],false
);
   LoadTex-
tu-
re('data\Blocktex\bluecloud_up.jpg',Sky.Texture[5],false
);
end;

procedure Load_Game_Models();
begin
  TObj := TGLMultyMesh.Create;
  TObj.LoadFromFile('GRE\sphere.gms');
  TObj.Extent := true;
  TObj.fSmooth := false; // Установить в фасеты
end;


Procedure RenderBox();
begin
glPushMatrix;
glBegin(GL_QUADS);
   glNormal3f( 0.0, 0.0, 1.0);
   glTexCoord2f(0.0, 0.0); glVertex3f(-1.0, -1.0,  1.0);
   glTexCoord2f(1.0, 0.0); glVertex3f( 1.0, -1.0,  1.0);
   glTexCoord2f(1.0, 1.0); glVertex3f( 1.0,  1.0,  1.0);
   glTexCoord2f(0.0, 1.0); glVertex3f(-1.0,  1.0,  1.0);
  glEnd; glPopMatrix;  glPushMatrix;
  glBegin(GL_QUADS);
   glNormal3f( 0.0, 0.0,-1.0);
   glTexCoord2f(1.0, 0.0); glVertex3f(-1.0, -1.0, -1.0);
   glTexCoord2f(1.0, 1.0); glVertex3f(-1.0,  1.0, -1.0);
   glTexCoord2f(0.0, 1.0); glVertex3f( 1.0,  1.0, -1.0);
   glTexCoord2f(0.0, 0.0); glVertex3f( 1.0, -1.0, -1.0);
  glEnd;    glPopMatrix;  glPushMatrix;
  glBegin(GL_QUADS);
   glNormal3f( 0.0, 1.0, 0.0);
   glTexCoord2f(0.0, 1.0); glVertex3f(-1.0,  1.0, -1.0);
   glTexCoord2f(0.0, 0.0); glVertex3f(-1.0,  1.0,  1.0);
   glTexCoord2f(1.0, 0.0); glVertex3f( 1.0,  1.0,  1.0);
   glTexCoord2f(1.0, 1.0); glVertex3f( 1.0,  1.0, -1.0);
  glEnd();  glPopMatrix;  glPushMatrix;
  glBegin(GL_QUADS);
   glNormal3f( 0.0,-1.0, 0.0);
   glTexCoord2f(1.0, 1.0); glVertex3f(-1.0, -1.0, -1.0);
   glTexCoord2f(0.0, 1.0); glVertex3f( 1.0, -1.0, -1.0);
   glTexCoord2f(0.0, 0.0); glVertex3f( 1.0, -1.0,  1.0);
   glTexCoord2f(1.0, 0.0); glVertex3f(-1.0, -1.0,  1.0);
  glEnd();  glPopMatrix;
            glPushMatrix;
  glBegin(GL_QUADS);
   glNormal3f( 1.0, 0.0, 0.0);
   glTexCoord2f(1.0, 0.0); glVertex3f( 1.0, -1.0, -1.0);
   glTexCoord2f(1.0, 1.0); glVertex3f( 1.0,  1.0, -1.0);
   glTexCoord2f(0.0, 1.0); glVertex3f( 1.0,  1.0,  1.0);
   glTexCoord2f(0.0, 0.0); glVertex3f( 1.0, -1.0,  1.0);
  glEnd();  glPopMatrix;

  glBegin(GL_QUADS);  glPushMatrix;
   glNormal3f(-1.0, 0.0, 0.0);
```

```
   glTexCoord2f(0.0, 0.0); glVertex3f(-1.0, -1.0, -1.0);
   glTexCoord2f(1.0, 0.0); glVertex3f(-1.0, -1.0,  1.0);
   glTexCoord2f(1.0, 1.0); glVertex3f(-1.0,  1.0,  1.0);
   glTexCoord2f(0.0, 1.0); glVertex3f(-1.0,  1.0, -1.0);
  glEnd(); glPopMatrix;
end;

procedure RenderFlore(x,y,size:integer; Texture:Uint);
begin
glTranslatef(x,y,-10);
glPushMatrix;
glBindTexture(GL_TEXTURE_2D,Texture);
glBegin(GL_QUADS);
   //glNormal3f( 0.0, 0.0, 1.0);
   glTexCoord2f(0.0, 0.0); glVertex3f(-1.0*size, -
1.0*size,  1.0);
   glTexCoord2f(1.0, 0.0); glVertex3f( 1.0*size, -
1.0*size,  1.0);
   glTexCoord2f(1.0, 1.0); glVertex3f( 1.0*size,
1.0*size,  1.0);
   glTexCoord2f(0.0, 1.0); glVertex3f(-1.0*size,
1.0*size,  1.0);
  glEnd;
glPopMatrix;
glTranslatef(-x,-y,+10);
end;

procedure RenderGMSModel(x,y,z,size:integer; mod-
el:TGLMultyMesh; Texture:Uint);
begin
       glBindTexture(GL_TEXTURE_2D,Texture);
      glTranslatef(x,y,z);
       glPushMatrix;
         glScalef(size,size,size);
         model.Draw;
       glPopMatrix;
      glTranslatef(-x,-y,-z);
end;

end.
unit Command;

interface
uses
  Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,
  Dialogs, OpenGL, DGLUT, Textures,mesh, math;

const
  GL_CLAMP_TO_EDGE =$812F;

  VK_W = $57;   VK_1 = $31;
  VK_S = $53;   VK_2 = $32;
  VK_D = $44;   VK_3 = $33;
  VK_A = $41;   VK_4 = $34;
  VK_R = $52;

  VK_5 = $35;   VK_6 = $36;
  VK_7 = $37;   VK_8 = $38;
  VK_9 = $39;   VK_0 = $30;
  VK_YO    =  $C0;
  VK_SPACE = $20;

type
  RObject = record
    x,y,sx,sy:integer;
    cls:byte;
    model:TGLMultyMesh;
    texture:Uint;
  end;
Type
  BBoxT =record
     Texture:array[0..5] of Uint;
  end;


//////----------------
**********************************************
******----------------------------////

Type
  FPlayer = record
    X,Y,Z:Real;
    dx,dy,dz:real;
    w,h,d:real;
    Colizt:integer;
    angleX,AngleY: Single;
    onGround:boolean;
    Speed:real;
    public
    procedure create(X0,Y0,Z0:real); //векторы переме-
щения по 3 осям
  end;
var
   mbl:boolean;
   Button:array[0..11] of Uint;
   SkyBox:array[0..5] of Uint;
   Commande, KeyClick:boolean;
   fogColor : array[0..3] of GLfloat = (0.14, 0.52, 0.89,
0.5); //цвет тумана
procedure IdenTifiWindow(ClientWidth, Clien-
tHeight:integer);
procedure Mouse_Move();
function Key_Move(G:Fplayer):Fplayer;
function check(x,y,z:integer):Byte;
function Nue-
Thon(fx,fy,fz,fsx,fsy,fsz:real;D:FPlayer):FPlayer;
procedure DrawBox(X,Y,Z,Size:real; Texture:BBoxT);
procedure Active_UnActive_system();
procedure Fog();


implementation
uses GUI;

//туман
procedure Fog();
begin
```

```
  glFogi(GL_FOG_MODE, GL_NEAREST ); // задаем
закон смешения тумана
  glHint(GL_FOG_HINT, GL_NEAREST);
  glFogf(GL_FOG_START , 10); // начало тумана
  glFogf(GL_FOG_END , 1000); // конец тумана
  glFogfv(GL_FOG_COLOR, @fogColor); // цвет
дымки6
  glFogf(GL_FOG_DENSITY, 0.8); // плотность
тумана
end;



//стартовое положение персонажа
procedure Fplayer.create(X0: Real; Y0: Real; Z0: Real);
begin
  x:=X0;  y:=y0;  z:=Z0;
  dx:=0;  dy:=0; dz:=0;
  w:=5; h:=20; d:=5; speed:=2;
  onGround:=false;
end;
//обработка курсора
procedure Active_UnActive_system();
begin
if (GetAsyncKeyState(VK_YO)<>0) then
  begin
    Commande:=true;
  end;
if (GetAsyncKeyState(VK_R)<>0) then
  begin
    SetCursorPos(screen.Width div 2,screen.Height div
2);
    Commande:=false;
  end;
end;
// физика
function Nue-
Thon(fx,fy,fz,fsx,fsy,fsz:real;D:FPlayer):FPlayer;
  var
    DX,DY,DZ:real;
    RX,RY,RZ:boolean;
    YAY:real;
begin
DX:=D.X-fx;
DZ:=D.Z-fz;
DY:=D.Y-fy;
  if (abs(DX)<fsx+D.w) and (abs(DZ)<fsz+d.d) and
(abs(DY)<fsy+d.h) then
    begin
      if (DX>0) then  RX:=true;
      if (DX<0) then  RX:=false;
      if (DZ>0) then  RZ:=true;
      if (DZ<0) then  RZ:=false;
      if (DY>0) then  RY:=true;
      if (DY<0) then  RY:=false;

          if (RY=true) or (RY=false) then begin
            if DY-D.h>fsy*0.7 then begin
               if D.dy<0 then begin
                d.dy:=0;
                D.Colizt:=D.Colizt+1;
                D.Y:=fy+D.H+fsy*0.95;
               end;
              end else
              begin
              if Rx=true then begin
                if (DZ-D.d<fsz*0.8) and (DZ+d.d>-fsz*0.8)
and (DX-d.w>fsx*0.7) then
                  begin
                    D.X:=fx+d.d+fsx;
                  end;
                end;
                if Rx=false then begin
                if (DZ-D.d<fsz*0.8) and (DZ+d.d>-fsz*0.8)
and (DX+d.w<-fsx*0.7) then
                  begin
                    D.X:=fx-d.d-fsx;
                  end;
                end;
                if Rz=true then begin
                  if (DX-D.w<fsx*0.8) and (DX+D.d>-fsx*0.8)
and (Dz-d.d>fsz*0.7) then
                    begin
                      D.Z:=fz+d.w+fsz;
                    end;
                end;
                if Rz=false then begin
                  if (Dx-D.w<fsx*0.8) and (Dx+D.w>-fsx*0.8)
and (Dz+d.d<-fsz*0.7)  then
                    begin
                      D.z:=fz-d.w-fsz;
                    end;
                end;
                if DY+D.h<-fsy*0.7 then begin
                  if D.dy>0 then begin
                    d.dy:=0;
                    D.Y:=fy-D.H-fsy*0.95;
                  end;
end; end; end; end;
result:=D;
end;

function check(x,y,z:integer):byte;
begin
  if ((x<0) or (X>127) or
      (y<0) or (Y>127) or
      (z<0) or (Z>127)) then
      begin
        result:= 0;
      end else begin
        result:=mass[x,y,z];
      end;

end;



//управление персонажа
function Key_Move(G:Fplayer):Fplayer;
var Gamer:Fplayer;
begin
```

```
  Gamer:=G;
    if Gamer.AngleY  >89 then Gamer.AngleY  :=89;
    if Gamer.AngleY  <-89 then Gamer.AngleY  :=-89;


  if (GetAsyncKeyState(VK_W)<>0) then
    begin
      Gamer.dx:= -
Sin(Gamer.angleX/180*Pi)*Gamer.Speed;
      Gamer.dz:= -
cos(Gamer.angleX/180*Pi)*Gamer.Speed;
    end;
  if (GetAsyncKeyState(VK_S)<>0) then
    begin
      Gamer.dx:=
Sin(Gamer.angleX/180*Pi)*Gamer.Speed;
      Gamer.dz:=
cos(Gamer.angleX/180*Pi)*Gamer.Speed;
    end;

  if (GetAsyncKeyState(VK_D)<>0) then
    begin
      Gam-
er.dx:=sin((Gamer.angleX+90)/180*Pi)*Gamer.Speed;
      Gam-
er.dz:=cos((Gamer.angleX+90)/180*Pi)*Gamer.Speed;
    end;
  if (GetAsyncKeyState(VK_A)<>0) then
    begin
      Gamer.dx:=sin((Gamer.angleX-
90)/180*Pi)*Gamer.Speed;
      Gamer.dz:=cos((Gamer.angleX-
90)/180*Pi)*Gamer.Speed;
    end;

  if (GetAsyncKeyState(VK_Space)<>0) and (Gam-
er.onGround=true) then
    begin
      Gamer.dy:=4;
    end;
      if (GetAsyncKeyState(VK_LButton)<>0) and
(bclick=false) then
        begin
          lmb:=true;
          bclick:=true;
        end else
        begin
          lmb:=false;
        end;

      if (GetAsyncKeyState(VK_RButton)<>0) and
(bclick=false) then
        begin
          rmb:=true;
          bclick:=true;
        end
        else
        begin
          rmb:=false;
        end;

  if (GetAsyncKeyState(VK_1)<>0) then MyBlock:=1;
  if (GetAsyncKeyState(VK_2)<>0) then MyBlock:=2;
  if (GetAsyncKeyState(VK_3)<>0) then MyBlock:=3;
  if (GetAsyncKeyState(VK_4)<>0) then MyBlock:=4;
  if (GetAsyncKeyState(VK_5)<>0) then MyBlock:=5;
  if (GetAsyncKeyState(VK_6)<>0) then MyBlock:=6;
  if (GetAsyncKeyState(VK_7)<>0) then MyBlock:=7;
  if (GetAsyncKeyState(VK_8)<>0) then MyBlock:=8;
  if (GetAsyncKeyState(VK_9)<>0) then MyBlock:=9;
  if (GetAsyncKeyState(VK_0)<>0) then MyBlock:=10;

result:=Gamer;

end;



//поворот камеры
procedure Mouse_Move();
begin
  try
    if MouseMove1 = false then
    begin
      GetCursorPos(Point);
      TempX1 := Point.X;
      TempY1 := Point.Y;
      MouseMove1 := true;
    end;
  finally
    if Form1.Active then
    begin
      SetCursorPos(Screen.Width div 2, Screen.Height div
2);
    end;
    GetCursorPos(Point);
    if MouseMove1 = true then
    begin
      TempX2 := Point.X;
      TempY2 := Point.Y;
      SingX := TempX1 - TempX2;
      SingY := TempY1 - TempY2;
      P.AngleY := P.AngleY + (-SingY / 8);
      P.AngleX := P.AngleX + (-SingX / 4);
      TempX1 := 0;
      TempY1 := 0;
      TempX2 := 0;
      TempY2 := 0;
      SingX := 0;
      SingY := 0;
      MouseMove1 := false;
    end;
  end;
end;

//////----------------
**********************************************
******----------------------------////
```

```
procedure IdenTifiWindow(ClientWidth, Clien-
tHeight:integer);
begin
  glViewport(0, 0, ClientWidth, ClientHeight);
//выделяем область куда будет выводиться наш бу-
фер
  glMatrixMode ( GL_PROJECTION ); //переходим в
матрицу проекции
  glLoadIdentity;  //сбрасываем текущую матрицу

gluPerspective(60,ClientWidth/ClientHeight,0.1,10000);
//Область видимости
  glMatrixMode ( GL_MODELVIEW ); //переходим в
модельную матрицу
  glLoadIdentity;//сбрасываем текущую матрицу
end;

//отрисовка блоков
procedure DrawBox(X,Y,Z,Size:real; Texture:BBoxT);
begin
   glTranslatef(X,Y,Z);
     glPushMatrix;
              glBindTexture(GL_TEXTURE_2D, Tex-
ture.Texture[0]);
              glBegin(GL_QUADS);
                        //front
          glTexCoord2f(0, 0);   glVertex3f(-size, -size, -
size);
          glTexCoord2f(1, 0);   glVertex3f(size,  -size, -
size);
          glTexCoord2f(1, 1);   glVertex3f( size,  size, -
size);
          glTexCoord2f(0, 1);   glVertex3f( -size, size, -
size);
        glEnd();
    glPopmatrix;
     glPushMatrix;
              glBindTexture(GL_TEXTURE_2D, Tex-
ture.Texture[1]);
              glBegin(GL_QUADS);
                              //back
          glTexCoord2f(0, 0); glVertex3f(size, -size, size);
          glTexCoord2f(1, 0); glVertex3f(-size,  -size,
size);
          glTexCoord2f(1, 1); glVertex3f( -size,  size,
size);
          glTexCoord2f(0, 1); glVertex3f( size, size, size);
       glEnd();
    glPopmatrix;
     glPushMatrix;
                    glBindTexture(GL_TEXTURE_2D,
Texture.Texture[2]);
              glBegin(GL_QUADS);
                              //left
          glTexCoord2f(0, 0); glVertex3f(-size, -size,
size);
          glTexCoord2f(1, 0); glVertex3f(-size, -size, -
size);
          glTexCoord2f(1, 1); glVertex3f(-size,  size, -
size);
          glTexCoord2f(0, 1); glVertex3f(-size,  size,
size);
        glEnd();
    glPopmatrix;
     glPushMatrix;
                    glBindTexture(GL_TEXTURE_2D,
Texture.Texture[3]);
              glBegin(GL_QUADS);
                              //right
          glTexCoord2f(0, 0); glVertex3f(size, -size, -size);
          glTexCoord2f(1, 0); glVertex3f(size,  -size, size);
          glTexCoord2f(1, 1); glVertex3f(size,  size,  size);
          glTexCoord2f(0, 1); glVertex3f(size, size,  -size);
        glEnd();
    glPopmatrix;
     glPushMatrix;
                    glBindTexture(GL_TEXTURE_2D,
Texture.Texture[4]);
              glBegin(GL_QUADS);
                              //bottom
          glTexCoord2f(0, 0); glVertex3f(-size, -size,
size);
          glTexCoord2f(1, 0); glVertex3f(size, -size, size);
          glTexCoord2f(1, 1); glVertex3f( size, -size, -
size);
          glTexCoord2f(0, 1); glVertex3f( -size, -size,  -
size);
        glEnd();
    glPopmatrix;
     glPushMatrix;
              glBindTexture(GL_TEXTURE_2D, Tex-
ture.Texture[5]);
              glBegin(GL_QUADS);
                              //top
          glTexCoord2f(0, 0); glVertex3f(-size, size,  -
size);
          glTexCoord2f(1, 0); glVertex3f(size, size, -size);
          glTexCoord2f(1, 1); glVertex3f( size, size, size);
          glTexCoord2f(0, 1); glVertex3f( -size, size,
size);
        glEnd();
    glPopmatrix;
   glTranslatef(-X,-Y,-Z);
end;

end.
unit GFonts;

interface
uses
  Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,
  Dialogs, OpenGL, DGLUT, Textures,command;


type
  FFont = record
    x,y,sim,Leg,a: integer;
```

```
 end;


var
  cis:integer;
  Sim:array[1..196] of string;
  FontI:array[1..196] of FFont;

procedure R3D_To_2D(ClientWidth, Clien-
tHeight:integer);
procedure R2D_To_3D();


implementation
uses GUI,Resurce;



procedure Enable_Atest();
begin
  glEnable(GL_DEPTH_TEST);     // Включает тест
глубины.
  glEnable(GL_NORMALIZE);      // Включает норма-
лизацию нормалей.
  glEnable(GL_COLOR_MATERIAL); // Включает
цветные материалы.
  glShadeModel(GL_SMOOTH);     // Устанавливает
сглаживание (плавное изменение цветов).
end;

procedure Disable_Atest();
begin
  glDisable(GL_NORMALIZE);      // Отключает нор-
мализацию нормалей.
  glDisable(GL_COLOR_MATERIAL); // Отключает
цветные материалы.
  glDisable(GL_DEPTH_TEST);     // Отключает тест
глубины.
end;

procedure R3D_To_2D(ClientWidth, ClientHeight: inte-
ger);
begin
  glPushMatrix;                 // Сохраняет текущую мат-
рицу.
  glLoadIdentity;               // Сбрасывает текущую мат-
рицу.
  glMatrixMode(GL_PROJECTION);   // Переключает
в режим работы с проекционной матрицей.
  glPushMatrix;                 // Сохраняет проекционную
матрицу.
  glLoadIdentity;               // Сбрасывает проекцион-
ную матрицу.
  gluOrtho2D(0, ClientWidth, ClientHeight, 0); // Уста-
навливает ортогональную проекцию.
  glMatrixMode(GL_MODELVIEW);    // Переключает
в режим работы с модельно-видовой матрицей.
end;

procedure R2D_To_3D();
begin
  glMatrixMode(GL_PROJECTION); // Переключает в
режим работы с проекционной матрицей.
  glPopMatrix;                 // Восстанавливает предыду-
щую проекционную матрицу.
  glMatrixMode(GL_MODELVIEW);  // Переключает в
режим работы с модельно-видовой матрицей.
  glPopMatrix;                 // Восстанавливает предыду-
щую модельно-видовую матрицу.
end;


end.
unit Gamelavels;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,
  Dialogs, OpenGL, DGLUT, Textures,Mesh, com-
mand,GFonts,resurce;


procedure MainGame(ClientWidth, Clien-
tHeight:integer);

implementation
uses GUI;
procedure MainGame(ClientWidth, Clien-
tHeight:integer);
var I,J,Y:integer;
CW,CH:integer;
EN:integer;
Dir:real;
begin
CW:=ClientWidth;
CH:=ClientHeight;
  glClearColor(0.1, 0.1, 0.1, 0.0); // цвет фона
  glClear (GL_COLOR_BUFFER_BIT or
GL_DEPTH_BUFFER_BIT); // очистка буфера цвета

  glEnable(GL_DEPTH_TEST);
  glEnable(GL_NORMALIZE);
  glEnable(GL_COLOR_MATERIAL);
  glShadeModel(GL_SMOOTH);
    Fog();

 for I := 1 to 127 do
  for J := 1 to 127 do
   for Y := 1 to 127 do
   begin
     dir:=sqrt(sqr(p.X-size*i+size/2)+sqr(p.Y-
size*j+size/2)+sqr(p.z-size*y+size/2));

     If (Mass[I,J,Y]>0) and (abs(dir)<500) then
       begin
         if (Mass[I+1,J,Y]>0) and (Mass[I-1,J,Y]>0)and
             (Mass[I,J+1,Y]>0) and (Mass[I,J-1,Y]>0) and
```

```
                (Mass[I,J,Y+1]>0) and (Mass[I,J,Y-1]>0)
then
                  begin
                end
             else begin
            DrawBox(size*i+size/2, size*j+size/2,
size*y+size/2,size/2,Blocks[Mass[I,J,Y]-1]);
         end;

     end;
  end;
    DrawBox(P.x,p.y,p.z,2000,Sky);
      glBindTexture(GL_TEXTURE_2D, Ttextobj[0]);
   glpushmatrix();
   R3D_To_2D(ClientWidth, ClientHeight);

   RenderSprite(CW/2,CH/2,25,25,0,Ttextobj[0]);
   RenderSprite(CW/2-240,CH-50,30,30,0,Ttextobj[1]);
   RenderSprite(CW/2-180,CH-50,30,30,0,Ttextobj[2]);
   RenderSprite(CW/2-120,CH-50,30,30,0,Ttextobj[3]);
   RenderSprite(CW/2-60,CH-50,30,30,0,Ttextobj[4]);
   RenderSprite(CW/2,CH-50,30,30,0,Ttextobj[5]);
   RenderSprite(CW/2+60,CH-50,30,30,0,Ttextobj[6]);
   RenderSprite(CW/2+120,CH-50,30,30,0,Ttextobj[7]);
   RenderSprite(CW/2+180,CH-50,30,30,0,Ttextobj[8]);
   RenderSprite(CW/2+240,CH-50,30,30,0,Ttextobj[9]);
   RenderSprite(CW/2+300,CH-
50,30,30,0,Ttextobj[10]);
   RenderSprite(CW/2-240+((MyBlock-1)*60),CH-
50,35,35,0,Ttextobj[200]);
   R2D_To_3D();
   glpopmatrix();
  glPopMatrix;


glPopMatrix;

R3D_To_2D(ClientWidth, ClientHeight);
end;

end.
```