Министерство образования и науки РФ

Новосибирский государственный технический университет

Кафедра общей физики

Отчет по проекту

Виртуальная физическая лаборатория

«Cтоячие волны на непрерывной струне»

Факультет: ПМИ

Группа: ПМ-02

Авторы: Сарычев Виктор

Вайцель Сергей

Руководитель: к.ф.м.н., доц. Баранов Александр Викторович

Новосибирск

2012

Оглавление

[1. Цель проекта 3](#_Toc327981950)

[2. Физическая и математическая модели 3](#_Toc327981951)

[3. Алгоритм и средства программной реализации 4](#_Toc327981952)

[4. Текст программы 5](#_Toc327981953)

[5. Презентация проекта 35](#_Toc327981954)

[6. Вывод 37](#_Toc327981955)

[7. Литература и ссылки 38](#_Toc327981956)

1. Цель проекта

Разработать программу, моделирующая стоячие волны на непрерывной струне. В качестве выходных данных мы получаем звук, образующийся вследствие её колебания.

1. Физическая и математическая модели

В данной модели рассматривается процесс возникновения поперечных стоячих волн на непрерывной струне.

– уравнение стоячей волны, где

– отклонение точки от состояния покоя

– амплитуда колебания

– длина струны

– координата точки

– частота гармонической волны, рассчитывается по формуле

, где – сила натяжения струны, а - её плотность.

*–* текущее время

Каждая точка струны будет совершать гармонические колебания, амплитуда и фаза которой будут зависеть от координаты x.

Также при колебании струна порождает звуковые волны, частота которых совпадает с частотой колебания струны.

Также в каждый момент времени для струны рассчитываются потенциальная и кинетическая энергии.

1. Алгоритм и средства программной реализации
   1. *Основные этапы разработки*
2. Программная реализация физического расчета колебаний струны.
3. Реализация пользовательского интерфейса.
4. Создание 3D модели, демонстрирующей колебания струны.
5. Добавление в программу воспроизведения звука, «издаваемого» виртуальной струной.
   1. *Среда разработки и язык программирования*

Программа написана на языке С++. Среда программирования – Microsoft Visual Studio 2010. Для отображения окон 3D моделей и реализации графиков использовалась графическая библиотека OpenGL. Интерфейс написан с использованием функциональных возможностей Win32Api.

* 1. *Описание вспомогательных библиотек*
* DevIL.dll – DevIL используется для чтения и записи изображений (текстур) из/в большое число различных графических форматов.
* freetype6.dll - используется для отображения шрифтов в окнах OpenGL.
* ftgl\_dynamic\_MT.dll – позволяет выводить шрифт в OpenGLприложениях.
* ILU.dll– предоставляет ряд возможностей по обработке изображений.
* msvcp100.dll - стандартная библиотека C++ для машинного кода.
* msvcr100.dll - библиотека среды выполнения языка С (CRT) для машинного кода.
* zlib1.dll - предоставляет функции компрессии и декомпрессии в памяти, включая проверку целостности несжатых данных.
  1. *Этапы разработки и особенности реализации*

1. Разработка физико-математической модели колебаний струны.
2. Создание интерактивного интерфейса, позволяющего изменять параметры модели, запускать и останавливать моделируемые процессы.
3. Создание и внедрение в проект 3D модели процесса. Для отрисовки объектов использовались стандартные графические примитивы. На элементы модели были нанесены соответствующие им текстуры.
4. Добавление графика зависимости энергии струны от времени, отладка параллельного отображения главного окна и окна графика.
5. Составление справки к разработанной программе, содержащей всю необходимую информацию для эффективной работы пользователя. Создание заставки и внесение информация о разработчиках. Дополнение программного кода подробными комментарии.

На протяжении всех этапов разработки проводилась работа по оптимизации с целью ускорить работу программы и сделать программный код более удобным для работы. За все время разработки были произведены сотни тестов, при проведении которых важную роль сыграла среда разработки, предоставившая все необходимые средства отладки программы.

1. Текст программы

**<font.h>**

# ifndef \_FONT\_H

# define \_FONT\_H

# include "main.h"

# ifdef \_MAIN\_H

# include <wchar.h>

# include "ftgl\FTFont.h"

# include "ftgl\FTGLBitmapFont.h"

# include "ftgl\FTGLOutlineFont.h"

class CFont

{

public:

CFont ( char \*ttf, int FSize, int FDepth );

~ CFont ( );

FTFont \*Font;

void Print ( float x, float y, const wchar\_t \*text );

};

# endif

# endif

**<font.cpp>**

# include "stdafx.h"

# include "font.h"

# ifdef \_FONT\_H

CFont :: CFont ( char \*ttf, int FSize, int FDepth )

{

this -> Font = new FTGLBitmapFont ( ttf );

if ( !Font -> FaceSize ( FSize ) )

{

MessageBox ( NULL, L"Can't set font FaceSize", L"Error", MB\_OK );

exit ( 1 );

}

Font -> Depth ( (float)FDepth );

Font -> CharMap ( ft\_encoding\_unicode );

}

void CFont :: Print ( float x, float y, const wchar\_t \*text )

{

glPushMatrix ( );

glLoadIdentity ( );

glDisable ( GL\_TEXTURE\_2D );

glTranslatef ( x, y, -1 );

glRasterPos2f ( -1, 0.5 );

Font -> Render ( text );

glEnable ( GL\_TEXTURE\_2D );

glPopMatrix ( );

}

# endif

**<camera.h>**

# ifndef \_CAMERA\_H

# define \_CAMERA\_H

# include "main.h"

#ifdef \_MAIN\_H

class CVector3

{

public:

float x, y, z;

CVector3 ( ) { }

CVector3 ( float X, float Y, float Z )

{

x = X;

y = Y;

z = Z;

}

CVector3 operator + ( CVector3 vVector )

{

return CVector3 ( vVector.x + x, vVector.y + y, vVector.z + z );

}

CVector3 operator - ( CVector3 vVector )

{

return CVector3 ( x - vVector.x, y - vVector.y, z - vVector.z );

}

CVector3 operator \* ( float num )

{

return CVector3 ( x \* num, y \* num, z \* num );

}

CVector3 operator / ( float num )

{

return CVector3 ( x / num, y / num, z / num );

}

};

class CCamera

{

private:

CVector3 m\_vPosition;

CVector3 m\_vView;

CVector3 m\_vUpVector;

CVector3 m\_vStrafe;

public:

CCamera ( );

CVector3 Position ( ) { return m\_vPosition; }

CVector3 View ( ) { return m\_vView; }

CVector3 UpVector ( ) { return m\_vUpVector; }

CVector3 Strafe ( ) { return m\_vStrafe; }

void PositionCamera ( float PositionX, float PositionY, float PositionZ,

float ViewX, float ViewY, float ViewZ,

float upVectorX, float upVectorY, float upVectorZ );

void StrafeCamera ( float speed );

void StrafeCameray ( float speed );

void MoveCamera ( float speed );

void RotateView ( float angle, float x, float y, float z );

void CheckForMovement ( GLfloat kSpeed );

void Update ( GLfloat kSpeed );

void Look ( );

void SetViewByMouse ( );

void RotateAroundPoint ( CVector3 vCenter, float angle, float x, float y, float z );

};

# endif

# endif

**<camera.cpp>**

# include "stdafx.h"

# include "camera.h"

# ifdef \_CAMERA\_H

float g\_FrameInterval = 0.0f;

void GetFrameTime ( )

{

static float frameTime = 0.0f;

float currentTime = timeGetTime ( ) \* 0.001f;

g\_FrameInterval = currentTime - frameTime;

frameTime = currentTime;

}

CVector3 Cross ( CVector3 vVector1, CVector3 vVector2 )

{

CVector3 vNormal;

vNormal.x = ( ( vVector1.y \* vVector2.z ) - ( vVector1.z \* vVector2.y ) );

vNormal.y = ( ( vVector1.z \* vVector2.x ) - ( vVector1.x \* vVector2.z ) );

vNormal.z = ( ( vVector1.x \* vVector2.y ) - ( vVector1.y \* vVector2.x ) );

return vNormal;

}

float Magnitude ( CVector3 vNormal )

{

return ( float ) sqrt ( ( vNormal.x \* vNormal.x) + ( vNormal.y \* vNormal.y ) + ( vNormal.z \* vNormal.z ) );

}

CVector3 Normalize ( CVector3 vVector )

{

float magnitude = Magnitude ( vVector );

vVector = vVector / magnitude;

return vVector;

}

void CCamera :: SetViewByMouse ( )

{

POINT mousePos;

int middleX = SCREEN\_WIDTH >> 1; *// Вычисляем половину ширины*

int middleY = SCREEN\_HEIGHT >> 1; *// И половину высоты экрана*

float angleY= 0.0f;

float angleZ = 0.0f;

static float currentRotX = 0.0f;

GetCursorPos ( &mousePos );

if ( ( mousePos.x == middleX) && ( mousePos.y == middleY ) )

return;

SetCursorPos ( middleX, middleY );

angleY = ( float ) ( ( middleX - mousePos.x ) ) / 1000.0f;

angleZ = ( float ) ( ( middleY - mousePos.y ) ) / 1000.0f;

static float lastRotX = 0.0f;

lastRotX = currentRotX;

if ( currentRotX > 1.0f )

{

currentRotX = 1.0f;

if ( lastRotX != 1.0f )

{

CVector3 vAxis = Cross ( m\_vView - m\_vPosition, m\_vUpVector );

vAxis = Normalize ( vAxis );

RotateView ( 1.0f - lastRotX, vAxis.x, vAxis.y, vAxis.z );

}

}

else

{

if ( currentRotX < -1.0f )

{

currentRotX = -1.0f;

if ( lastRotX != -1.0f )

{

CVector3 vAxis = Cross ( m\_vView - m\_vPosition, m\_vUpVector );

vAxis = Normalize ( vAxis );

RotateView ( -1.0f - lastRotX, vAxis.x, vAxis.y, vAxis.z );

}

}

else

{

CVector3 vAxis = Cross ( m\_vView - m\_vPosition, m\_vUpVector );

vAxis = Normalize ( vAxis );

RotateView ( angleZ, vAxis.x, vAxis.y, vAxis.z );

}

}

RotateView ( angleY, 0, 1, 0 );

}

void CCamera :: StrafeCamera ( float speed )

{

m\_vPosition.x += m\_vStrafe.x \* speed;

m\_vPosition.z += m\_vStrafe.z \* speed;

m\_vView.x += m\_vStrafe.x \* speed;

m\_vView.z += m\_vStrafe.z \* speed;

}

void CCamera :: CheckForMovement ( GLfloat kSpeed)

{

float speed = kSpeed \* g\_FrameInterval;

if ( ( GetKeyState ( VK\_UP ) & 0x80 ) || ( GetKeyState ( 'W' ) & 0x80 ) )

MoveCamera ( speed );

if ( ( GetKeyState ( VK\_DOWN ) & 0x80 ) || ( GetKeyState ( 'S' ) & 0x80 ) )

MoveCamera ( - speed );

if ( ( GetKeyState ( VK\_LEFT ) & 0x80 ) || ( GetKeyState ( 'A' ) & 0x80 ) )

StrafeCamera (-speed);

if ( ( GetKeyState ( VK\_RIGHT ) & 0x80 ) || ( GetKeyState ( 'D' ) & 0x80 ) )

StrafeCamera (speed);

}

void CCamera :: Update ( GLfloat kSpeed )

{

CVector3 vCross = Cross ( m\_vView - m\_vPosition, m\_vUpVector );

m\_vStrafe = Normalize ( vCross );

SetViewByMouse ( );

CheckForMovement ( kSpeed );

GetFrameTime ( );

}

void CCamera :: Look ( )

{

gluLookAt ( m\_vPosition.x, m\_vPosition.y, m\_vPosition.z,

m\_vView.x, m\_vView.y, m\_vView.z,

m\_vUpVector.x, m\_vUpVector.y, m\_vUpVector.z);

}

CCamera :: CCamera ( )

{

CVector3 vZero = CVector3 ( 0.0, 0.0, 0.0 );

CVector3 vView = CVector3 ( 0.0, 1.0, 0.5 );

CVector3 vUp = CVector3 ( 0.0, 0.0, 1.0 );

m\_vPosition = vZero;

m\_vView = vView;

m\_vUpVector = vUp;

}

GLvoid CCamera :: PositionCamera ( float PositionX, float PositionY, float PositionZ,

float ViewX, float ViewY, float ViewZ,

float upVectorX, float upVectorY, float upVectorZ )

{

CVector3 vPosition = CVector3 ( PositionX, PositionY, PositionZ );

CVector3 vView = CVector3 ( ViewX, ViewY, ViewZ );

CVector3 vUpVector = CVector3 ( upVectorX, upVectorY, upVectorZ );

m\_vPosition = vPosition;

m\_vView = vView;

m\_vUpVector = vUpVector;

}

void CCamera :: RotateView ( float angle, float x, float y, float z )

{

CVector3 vNewView;

CVector3 vView;

vView.x = m\_vView.x - m\_vPosition.x;

vView.y = m\_vView.y - m\_vPosition.y;

vView.z = m\_vView.z - m\_vPosition.z;

float cosTheta = ( float ) cos ( angle );

float sinTheta = ( float ) sin ( angle );

vNewView.x = ( cosTheta + ( 1 - cosTheta ) \* x \* x ) \* vView.x;

vNewView.x += ( ( 1 - cosTheta ) \* x \* y - z \* sinTheta ) \* vView.y;

vNewView.x += ( ( 1 - cosTheta ) \* x \* z + y \* sinTheta ) \* vView.z;

vNewView.y = ( ( 1 - cosTheta ) \* x \* y + z \* sinTheta ) \* vView.x;

vNewView.y += ( cosTheta + ( 1 - cosTheta ) \* y \* y ) \* vView.y;

vNewView.y += ( ( 1 - cosTheta ) \* y \* z - x \* sinTheta ) \* vView.z;

vNewView.z = ( ( 1 - cosTheta ) \* x \* z - y \* sinTheta ) \* vView.x;

vNewView.z += ( ( 1 - cosTheta ) \* y \* z + x \* sinTheta ) \* vView.y;

vNewView.z += ( cosTheta + ( 1 - cosTheta ) \* z \* z ) \* vView.z;

m\_vView.x = m\_vPosition.x + vNewView.x;

m\_vView.y = m\_vPosition.y + vNewView.y;

m\_vView.z = m\_vPosition.z + vNewView.z;

}

void CCamera :: MoveCamera ( float speed )

{

CVector3 vVector = m\_vView - m\_vPosition;

vVector = Normalize ( vVector );

m\_vPosition.x += vVector.x \* speed;

m\_vPosition.y += vVector.y \* speed;

m\_vPosition.z += vVector.z \* speed;

m\_vView.x += vVector.x \* speed;

m\_vView.y += vVector.y \* speed;

m\_vView.z += vVector.z \* speed;

}

# endif

**<glelements\_and\_texture.h>**

# ifndef \_GLELEMENTS\_AND\_TEXTURE\_H

# define \_GLELEMENTS\_AND\_TEXTURE\_H

# include "main.h"

# ifdef \_MAIN\_H

# include "devil\include\IL\il.h"

# include "devil\include\IL\ilu.h"

struct TextureImage

{

GLubyte \*imageData; *// Данные текстуры*

GLuint bpp; *// Байт на пиксел*

GLuint width,height; *// Высота и ширина*

GLuint texID; *// ID текстуры*

};

class CTexture

{

public:

TextureImage textures [ 41 ];

void InitTextures ( CTexture \*Texture );

CTexture ( )

{

ilInit ( );

iluInit ( );

}

~CTexture ( ) { }

void LoadTexture ( ILenum FileType, char \*filename, TextureImage \*texture );

void FreeTexture ( TextureImage \*texture ) { }

};

class GLelements

{

public:

void CreateTimeNumElement ( GLfloat HEIGHT, GLfloat RADIUS, int R, int G, int B );

void CreateOctagon ( float X, float Y, float Z, GLfloat LENGHT, GLfloat RADIUS );

void CreateHolderStrings ( CTexture \*Texture, float X, float Y, float Z );

void CreateBigStend ( CTexture \*Texture, int NOT );

void CreateQuadNoteLine ( float X, float Y, float Z, int R, int G, int B );

void CreateTimeNum\_0 ( float X, float Y, float Z, GLfloat HEIGHT, GLfloat RADIUS, GLfloat size, int R, int G, int B );

void CreateTimeNum\_1 ( float X, float Y, float Z, GLfloat HEIGHT, GLfloat RADIUS, GLfloat size, int R, int G, int B );

void CreateTimeNum\_2 ( float X, float Y, float Z, GLfloat HEIGHT, GLfloat RADIUS, GLfloat size, int R, int G, int B );

void CreateTimeNum\_3 ( float X, float Y, float Z, GLfloat HEIGHT, GLfloat RADIUS, GLfloat size, int R, int G, int B );

void CreateTimeNum\_4 ( float X, float Y, float Z, GLfloat HEIGHT, GLfloat RADIUS, GLfloat size, int R, int G, int B );

void CreateTimeNum\_5 ( float X, float Y, float Z, GLfloat HEIGHT, GLfloat RADIUS, GLfloat size, int R, int G, int B );

void CreateTimeNum\_6 ( float X, float Y, float Z, GLfloat HEIGHT, GLfloat RADIUS, GLfloat size, int R, int G, int B );

void CreateTimeNum\_7 ( float X, float Y, float Z, GLfloat HEIGHT, GLfloat RADIUS, GLfloat size, int R, int G, int B );

void CreateTimeNum\_8 ( float X, float Y, float Z, GLfloat HEIGHT, GLfloat RADIUS, GLfloat size, int R, int G, int B );

void CreateTimeNum\_9 ( float X, float Y, float Z, GLfloat HEIGHT, GLfloat RADIUS, GLfloat size, int R, int G, int B );

};

# endif

# endif

**<glelements\_and\_texture.cpp>**

# include "stdafx.h"

# include "glelements\_and\_texture.h"

# ifdef \_GLELEMENTS\_AND\_TEXTURE\_H

void CTexture :: InitTextures ( CTexture \*Texture )

{

Texture -> LoadTexture ( IL\_BMP, "01.bmp", &textures [ 1 ] );

Texture -> LoadTexture ( IL\_BMP, "02.bmp", &textures [ 2 ] );

Texture -> LoadTexture ( IL\_BMP, "03.bmp", &textures [ 3 ] );

Texture -> LoadTexture ( IL\_BMP, "04.bmp", &textures [ 4 ] );

Texture -> LoadTexture ( IL\_BMP, "05.bmp", &textures [ 5 ] );

Texture -> LoadTexture ( IL\_BMP, "06.bmp", &textures [ 6 ] );

Texture -> LoadTexture ( IL\_BMP, "07.bmp", &textures [ 7 ] );

Texture -> LoadTexture ( IL\_BMP, "08.bmp", &textures [ 8 ] );

Texture -> LoadTexture ( IL\_BMP, "09.bmp", &textures [ 9 ] );

Texture -> LoadTexture ( IL\_BMP, "10.bmp", &textures [ 10 ] );

Texture -> LoadTexture ( IL\_BMP, "11.bmp", &textures [ 11 ] );

Texture -> LoadTexture ( IL\_BMP, "12.bmp", &textures [ 12 ] );

Texture -> LoadTexture ( IL\_BMP, "13.bmp", &textures [ 13 ] );

Texture -> LoadTexture ( IL\_BMP, "14.bmp", &textures [ 14 ] );

Texture -> LoadTexture ( IL\_BMP, "15.bmp", &textures [ 15 ] );

Texture -> LoadTexture ( IL\_BMP, "16.bmp", &textures [ 16 ] );

Texture -> LoadTexture ( IL\_BMP, "17.bmp", &textures [ 17 ] );

Texture -> LoadTexture ( IL\_BMP, "18.bmp", &textures [ 18 ] );

Texture -> LoadTexture ( IL\_BMP, "19.bmp", &textures [ 19 ] );

Texture -> LoadTexture ( IL\_BMP, "20.bmp", &textures [ 20 ] );

Texture -> LoadTexture ( IL\_BMP, "21.bmp", &textures [ 21 ] );

Texture -> LoadTexture ( IL\_BMP, "22.bmp", &textures [ 22 ] );

Texture -> LoadTexture ( IL\_BMP, "23.bmp", &textures [ 23 ] );

Texture -> LoadTexture ( IL\_BMP, "24.bmp", &textures [ 24 ] );

Texture -> LoadTexture ( IL\_BMP, "251.bmp", &textures [ 25 ] );

Texture -> LoadTexture ( IL\_BMP, "2521.bmp", &textures [ 26 ] );

Texture -> LoadTexture ( IL\_BMP, "2522.bmp", &textures [ 27 ] );

Texture -> LoadTexture ( IL\_BMP, "2523.bmp", &textures [ 28 ] );

Texture -> LoadTexture ( IL\_BMP, "2524.bmp", &textures [ 29 ] );

Texture -> LoadTexture ( IL\_BMP, "2525.bmp", &textures [ 30 ] );

Texture -> LoadTexture ( IL\_BMP, "2526.bmp", &textures [ 31 ] );

Texture -> LoadTexture ( IL\_BMP, "2527.bmp", &textures [ 32 ] );

Texture -> LoadTexture ( IL\_BMP, "2528.bmp", &textures [ 33 ] );

Texture -> LoadTexture ( IL\_BMP, "2529.bmp", &textures [ 34 ] );

Texture -> LoadTexture ( IL\_BMP, "25210.bmp", &textures [ 35 ] );

Texture -> LoadTexture ( IL\_BMP, "36.bmp", &textures [ 36 ] );

Texture -> LoadTexture ( IL\_BMP, "37.bmp", &textures [ 37 ] );

Texture -> LoadTexture ( IL\_BMP, "38.bmp", &textures [ 38 ] );

Texture -> LoadTexture ( IL\_BMP, "HOLDER01.bmp", &textures [ 39 ] );

Texture -> LoadTexture ( IL\_BMP, "HOLDER02.bmp", &textures [ 40 ] );

}

void DrawQ ( CTexture \*Texture, GLfloat \*TCBlock, int TEXT\_ID, GLfloat \*CVBlock )

{

static GLushort QVBlock [ ] = { 0, 1, 2, 3 }; *//Номера вершин*

glBindTexture ( GL\_TEXTURE\_2D, Texture -> textures [ TEXT\_ID ].texID );

glPushName ( TEXT\_ID );

glTexCoordPointer ( 2, GL\_FLOAT, 0, TCBlock );

glVertexPointer ( 3, GL\_FLOAT, 0, CVBlock );

glDrawElements ( GL\_QUADS, 4, GL\_UNSIGNED\_SHORT, QVBlock );

glPopName ( );

}

void CTexture :: LoadTexture ( ILenum FileType, char \*filename, TextureImage \*texture )

{

ilLoad ( FileType, ( LPCWSTR ) filename );

int err=ilGetError ( );

if ( err != IL\_NO\_ERROR )

{

const wchar\_t\* strError = iluErrorString(err);

MessageBox(NULL, (LPCWSTR)strError, L"Ошибка при загрузке!", MB\_OK);

exit(1);

}

texture->width = ilGetInteger(IL\_IMAGE\_WIDTH); *// Ширина*

texture->height = ilGetInteger(IL\_IMAGE\_HEIGHT); *// Высота*

texture->bpp = ilGetInteger(IL\_IMAGE\_BYTES\_PER\_PIXEL); *// Байт на пиксель*

*// Загружаем данные в нашу текстуру*

texture->imageData = ilGetData();

ilEnable(IL\_CONV\_PAL);

*// Тип данных изображения*

unsigned int type = ilGetInteger(IL\_IMAGE\_FORMAT);

*// Генерируем текстуру*

glGenTextures(1, &texture->texID);

*// Привязываем данные текстуры к ID*

glBindTexture(GL\_TEXTURE\_2D,texture->texID);

*// биндим мип-мапы*

gluBuild2DMipmaps(GL\_TEXTURE\_2D, texture->bpp, texture->width,

texture->height, type, GL\_UNSIGNED\_BYTE, texture->imageData);

*// Устанавливаем качество текстур*

glTexParameteri ( GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR );

glTexParameteri ( GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR );

}

void GLelements :: CreateTimeNumElement ( GLfloat HEIGHT, GLfloat RADIUS, int R, int G, int B )

{

# pragma region Координаты вершин 6-угольника

GLfloat CVertex3TineElem [ ] =

{

0.0f, -HEIGHT, 0.0f,

RADIUS, -HEIGHT / 1.35f, 0.0f,

RADIUS, HEIGHT / 1.35f, 0.0f,

0.0f, HEIGHT, 0.0f,

-RADIUS, HEIGHT / 1.35f, 0.0f,

-RADIUS, -HEIGHT / 1.35f, 0.0f

};

# pragma endregion

# pragma region Номера вершин 6-угольника

static GLushort QVertex3 [ ] =

{

0, 1, 2, 3, 4, 5

};

# pragma endregion

# pragma region Отрисовка OpenGL

glPushMatrix();

glEnable ( GL\_LINE );

*//glEnable ( GL\_LINE\_SMOOTH );*

glEnableClientState ( GL\_VERTEX\_ARRAY );

glLineWidth ( 1.0f );

glColor3ub ( R, G, B );

glVertexPointer ( 3, GL\_FLOAT, 0, CVertex3TineElem );

glDrawElements ( GL\_LINE\_STRIP, 7, GL\_UNSIGNED\_SHORT, QVertex3 );

glColor3ub ( 255, 255, 255 );

*//glPopName ( );*

glLineWidth ( 1.0f );

glDisableClientState ( GL\_VERTEX\_ARRAY );

*//glDisable ( GL\_LINE\_SMOOTH );*

glPopMatrix ( );

# pragma endregion

}

void GLelements :: CreateOctagon ( float X, float Y, float Z, GLfloat LENGHT, GLfloat RADIUS )

{

glTranslatef ( X, Y, Z );

# pragma region Координаты вершин 8-угольника

GLfloat CVertex3String01 [ ] =

{

0.0f, 0.0f, 0.0f, *//1//*

0.0f, -RADIUS, RADIUS,

LENGHT, -RADIUS, RADIUS,

LENGHT, 0.0f, 0.0f

};

GLfloat CVertex3String02 [ ] =

{

0.0f, -RADIUS, RADIUS, *//2//*

0.0f, -RADIUS, RADIUS \* 2,

LENGHT, -RADIUS, RADIUS \* 2,

LENGHT, -RADIUS, RADIUS

};

GLfloat CVertex3String03 [ ] =

{

0.0f, -RADIUS, RADIUS \* 2, *//3//*

0.0f, 0.0f, RADIUS \* 3,

LENGHT, 0.0f, RADIUS \* 3,

LENGHT, -RADIUS, RADIUS \* 2

};

GLfloat CVertex3String04 [ ] =

{

0.0f, 0.0f, RADIUS \* 3, *//4//*

0.0f, RADIUS, RADIUS \* 3,

LENGHT, RADIUS, RADIUS \* 3,

LENGHT, 0.0f, RADIUS \* 3

};

GLfloat CVertex3String05 [ ] =

{

0.0f, RADIUS, RADIUS \* 3, *//5//*

0.0f, RADIUS \* 2, RADIUS \* 2,

LENGHT, RADIUS \* 2, RADIUS \* 2,

LENGHT, RADIUS, RADIUS \* 3

};

GLfloat CVertex3String06 [ ] =

{

0.0f, RADIUS \* 2, RADIUS \* 2, *//6//*

0.0f, RADIUS \* 2, RADIUS,

LENGHT, RADIUS \* 2, RADIUS,

LENGHT, RADIUS \* 2, RADIUS \* 2

};

GLfloat CVertex3String07 [ ] =

{

0.0f, RADIUS \* 2, RADIUS, *//7//*

0.0f, RADIUS, 0.0f,

LENGHT, RADIUS, 0.0f,

LENGHT, RADIUS \* 2, RADIUS

};

GLfloat CVertex3String08 [ ] =

{

0.0f, RADIUS, 0.0f, *//8//*

0.0f, 0.0f, 0.0f,

LENGHT, 0.0f, 0.0f,

LENGHT, RADIUS, 0.0f

};

# pragma endregion

# pragma region Номера вершин 8-угольника

static GLushort QVertex3 [ ] =

{

0, 1, 2, 3

};

# pragma endregion

# pragma region Отрисовка OpenGL

glPushMatrix();

glEnableClientState ( GL\_VERTEX\_ARRAY );

glEnable ( GL\_SMOOTH );

glColor3ub ( 50, 50, 50 );

glVertexPointer ( 3, GL\_FLOAT, 0, CVertex3String01 );

glDrawElements ( GL\_QUADS, 4, GL\_UNSIGNED\_SHORT, QVertex3 );

glColor3ub ( 150, 150, 150 );

glVertexPointer ( 3, GL\_FLOAT, 0, CVertex3String02 );

glDrawElements ( GL\_QUADS, 4, GL\_UNSIGNED\_SHORT, QVertex3 );

glVertexPointer ( 3, GL\_FLOAT, 0, CVertex3String03 );

glDrawElements ( GL\_QUADS, 4, GL\_UNSIGNED\_SHORT, QVertex3 );

glVertexPointer ( 3, GL\_FLOAT, 0, CVertex3String04 );

glDrawElements ( GL\_QUADS, 4, GL\_UNSIGNED\_SHORT, QVertex3 );

glColor3ub ( 50, 50, 50 );

glVertexPointer ( 3, GL\_FLOAT, 0, CVertex3String05 );

glDrawElements ( GL\_QUADS, 4, GL\_UNSIGNED\_SHORT, QVertex3 );

glColor3ub ( 150, 150, 150 );

glVertexPointer ( 3, GL\_FLOAT, 0, CVertex3String06 );

glDrawElements ( GL\_QUADS, 4, GL\_UNSIGNED\_SHORT, QVertex3 );

glVertexPointer ( 3, GL\_FLOAT, 0, CVertex3String07 );

glDrawElements ( GL\_QUADS, 4, GL\_UNSIGNED\_SHORT, QVertex3 );

glColor3ub ( 50, 50, 50 );

glVertexPointer ( 3, GL\_FLOAT, 0, CVertex3String08 );

glDrawElements ( GL\_QUADS, 4, GL\_UNSIGNED\_SHORT, QVertex3 );

glColor3ub ( 255, 255, 255 );

glDisable ( GL\_SMOOTH );

glDisableClientState ( GL\_VERTEX\_ARRAY );

glPopMatrix();

# pragma endregion

glTranslatef ( -X, -Y, -Z );

}

void GLelements :: CreateHolderStrings ( CTexture \*Texture, float X, float Y, float Z )

{

glTranslatef ( X, Y, Z );

# pragma region Координаты вершин стоек

GLfloat CVBlock01 [ ] =

{

0.0f, -1.49999f, 0.55f,

0.0f, -1.49999f, 0.0f,

0.0f, 1.99999f, 0.0f,

0.0f, 1.99999f, 0.55f

};

GLfloat CVBlock02 [ ] =

{

0.2f, -1.49999f, 0.55f,

0.2f, -1.49999f, 0.0f,

0.2f, 1.99999f, 0.0f,

0.2f, 1.99999f, 0.55f

};

GLfloat CVBlock03 [ ] =

{

0.2f, -1.49999f, 0.55f,

0.2f, -1.49999f, 0.0f,

0.0f, -1.49999f, 0.0f,

0.0f, -1.49999f, 0.55f

};

GLfloat CVBlock04 [ ] =

{

0.2f, -1.49999f, 0.0f,

0.2f, 1.99999f, 0.0f,

0.0f, 1.99999f, 0.0f,

0.0f, -1.49999f, 0.0f

};

GLfloat CVBlock05 [ ] =

{

0.2f, 1.99999f, 0.1f,

0.2f, 1.99999f, 0.0f,

0.0f, 1.99999f, 0.0f,

0.0f, 1.99999f, 0.1f

};

GLfloat CVBlock06 [ ] =

{

0.2f, -1.49999f, 0.55f,

0.2f, 1.99999f, 0.55f,

0.0f, 1.99999f, 0.55f,

0.0f, -1.49999f, 0.55f

};

# pragma endregion

# pragma region Верщины текстурных координат

static GLfloat TCBlock01 [ ] =

{

0, 1, 1, 1, 1, 0, 0, 0

};

static GLfloat TCBlock02 [ ] =

{

1, 1, 1, 0, 0, 0, 0, 1

};

# pragma endregion

# pragma region Отрисовка OpenGL

glPushMatrix();

glEnableClientState ( GL\_VERTEX\_ARRAY );

glEnableClientState ( GL\_TEXTURE\_COORD\_ARRAY );

glEnable ( GL\_SMOOTH );

DrawQ ( Texture, TCBlock01, 40, CVBlock01 ); DrawQ ( Texture, TCBlock01, 40, CVBlock02 );

DrawQ ( Texture, TCBlock02, 39, CVBlock03 ); DrawQ ( Texture, TCBlock02, 40, CVBlock04 );

DrawQ ( Texture, TCBlock02, 39, CVBlock05 ); DrawQ ( Texture, TCBlock02, 40, CVBlock06 );

glDisable ( GL\_SMOOTH );

glDisableClientState ( GL\_TEXTURE\_COORD\_ARRAY );

glDisableClientState ( GL\_VERTEX\_ARRAY );

glPopMatrix();

# pragma endregion

glTranslatef ( -X, -Y, -Z );

}

void GLelements :: CreateBigStend ( CTexture \*Texture, int NOT )

{

# pragma region Координаты вершин

GLfloat CVBlock01 [ ] =

{

0.5f, -1.5f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

19.5f, -1.5f, 0.6f, *// |\\_\_\_\_\_\_\_\_\_\_\_\_/|*

19.5f, -1.5f, 0.0f, *// || ||*

0.5f, -1.5f, 0.0f *// |/\_\_\_\_\_\_\_\_\_\_\_\_\|*

};

GLfloat CVBlock02 [ ] =

{

19.5f, -1.5f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

19.5f, 2.0f, 0.6f, *// |\\_\_\_\_\_\_\_\_\_\_\_\_/|*

19.5f, 2.0f, 0.0f, *// ||\_\_\_\_\_\_\_\_\_\_\_\_ |*

19.5f, -1.5f, 0.0f *// |/\_\_\_\_\_\_\_\_\_\_\_\_\|*

};

GLfloat CVBlock03 [ ] =

{

19.5f, 2.0f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

0.5f, 2.0f, 0.6f, *// |\ /|*

0.5f, 2.0f, 0.0f, *// ||\_\_\_\_\_\_\_\_\_\_\_\_||*

19.5f, 2.0f, 0.0f *// |/\_\_\_\_\_\_\_\_\_\_\_\_\|*

};

GLfloat CVBlock04 [ ] =

{

0.5f, 2.0f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

0.5f, -1.5f, 0.6f, *// |\\_\_\_\_\_\_\_\_\_\_\_\_/|*

0.5f, -1.5f, 0.0f, *// | \_\_\_\_\_\_\_\_\_\_\_\_||*

0.5f, 2.0f, 0.0f *// |/\_\_\_\_\_\_\_\_\_\_\_\_\|*

};

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

GLfloat CVBlock05 [ ] =

{

0.0f, -4.5f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

25.0f, -4.5f, 0.6f, *// |\\_\_\_\_\_\_\_\_\_\_\_\_/|*

25.0f, -4.5f, 0.0f, *// ||\_\_\_\_\_\_\_\_\_\_\_\_||*

0.0f, -4.5f, 0.0f *// |/ \|*

};

GLfloat CVBlock06 [ ] =

{

25.0f, -4.5f, 0.0f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

25.0f, -4.5f, 0.6f, *// |\\_\_\_\_\_\_\_\_\_\_\_\_/*

25.0f, 2.5f, 0.6f, *// ||\_\_\_\_\_\_\_\_\_\_\_\_|*

25.0f, 2.5f, 0.0f *// |/\_\_\_\_\_\_\_\_\_\_\_\_\*

};

GLfloat CVBlock07 [ ] =

{

25, 2.5, 0.6f, *// |\\_\_\_\_\_\_\_\_\_\_\_\_/|*

0, 2.5, 0.6f, *// ||\_\_\_\_\_\_\_\_\_\_\_\_||*

0, 2.5, 0.0f, *// |/\_\_\_\_\_\_\_\_\_\_\_\_\|*

25, 2.5, 0.0f

};

GLfloat CVBlock08 [ ] =

{

0.0f, -4.5f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

0.0f, -4.5f, 0.0f, *// \\_\_\_\_\_\_\_\_\_\_\_\_/|*

0.0f, 2.5f, 0.0f, *// |\_\_\_\_\_\_\_\_\_\_\_\_||*

0.0f, 2.5f, 0.6f *// /\_\_\_\_\_\_\_\_\_\_\_\_\|*

};

*//--------------------Передняя панель--------------------------//*

GLfloat CVBlock09 [ ] =

{

0.0f, -2.5f, 0.6f,

20.0f, -2.5f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_*

20.0f, -1.5f, 0.6f, *// /\_\_\_\_\_\_\_\_\_\_\_\_\*

0.0f, -1.5f, 0.6f

};

GLfloat CVBlock10 [ ] =

{

20.0f, -1.5f, 0.6f, *// /|*

20.0f, 2.0f, 0.6f, *// ||*

19.5f, 2.0f, 0.6f, *// \|*

19.5f, -1.5f, 0.6f

};

GLfloat CVBlock11 [ ] =

{

20.0f, 2.5f, 0.6f,

0.0f, 2.5f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

0.0f, 2.0f, 0.6f, *// \\_\_\_\_\_\_\_\_\_\_\_\_/*

20.0f, 2.0f, 0.6f

};

GLfloat CVBlock12 [ ] =

{

0.0f, 2.0f, 0.6f, *// |\*

0.0f, -1.5f, 0.6f, *// ||*

0.5f, -1.5f, 0.6f, *// |/*

0.5f, 2.0f, 0.6f

};

*//----------------------Задняя панель--------------------------//*

GLfloat CVBlock13 [ ] =

{

0.0f, -2.5f, 0.0f,

20.0f, -2.5f, 0.0f, *// \_\_\_\_\_\_\_\_\_\_\_\_*

20.0f, -1.5f, 0.0f, *// /\_\_\_\_\_\_\_\_\_\_\_\_\*

0.0f, -1.5f, 0.0f

};

GLfloat CVBlock14 [ ] =

{

20.0f, -1.5f, 0.0f, *// /|*

20.0f, 2.0f, 0.0f, *// ||*

19.5f, 2.0f, 0.0f, *// \|*

19.5f, -1.5f, 0.0f

};

GLfloat CVBlock15 [ ] =

{

20.0f, 2.5f, 0.0f,

0.0f, 2.5f, 0.0f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

0.0f, 2.0f, 0.0f, *// \\_\_\_\_\_\_\_\_\_\_\_\_/*

20.0f, 2.0f, 0.0f

};

GLfloat CVBlock16 [ ] =

{

0.0f, 2.0f, 0.0f, *// |\*

0.0f, -1.5f, 0.0f, *// ||*

0.5f, -1.5f, 0.0f, *// |/*

0.5f, 2.0f, 0.0f

};

*//-------------Нижняя панель (передняя сторона)-------------//*

GLfloat CVBlock17 [ ] =

{

0.25f, -4.5f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

19.75f, -4.5f, 0.6f, *// | |\_\_\_\_\_\_\_\_\_\_\_\_| |*

19.75f, -4.25f, 0.6f, *// | |\_\_\_\_\_\_\_\_\_\_\_\_| |*

0.25f, -4.25f, 0.6f *// |\_| |\_|*

};

GLfloat CVBlock18 [ ] =

{

20.0f, -4.5f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

20.0f, -2.5f, 0.6f, *// | |\_\_\_\_\_\_\_\_\_\_\_\_|*

19.75f, -2.5f, 0.6f, *// | |\_\_\_\_\_\_\_\_\_\_\_\_|*

19.75f, -4.5f, 0.6f *// |\_|\_\_\_\_\_\_\_\_\_\_\_\_|*

};

GLfloat CVBlock19 [ ] =

{

19.75f, -2.5f, 0.6f, *// \_ \_*

0.25f, -2.5f, 0.6f, *// | |\_\_\_\_\_\_\_\_\_\_\_\_| |*

0.25f, -2.75f, 0.6f, *// | |\_\_\_\_\_\_\_\_\_\_\_\_| |*

19.75f, -2.75f, 0.6f *// |\_|\_\_\_\_\_\_\_\_\_\_\_\_|\_|*

};

GLfloat CVBlock20 [ ] =

{

0.0f, -2.5f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

0.0f, -4.5f, 0.6f, *// |\_\_\_\_\_\_\_\_\_\_\_\_| |*

0.25f, -4.5f, 0.6f, *// |\_\_\_\_\_\_\_\_\_\_\_\_| |*

0.25f, -2.5f, 0.6f *// |\_\_\_\_\_\_\_\_\_\_\_\_|\_|*

};

GLfloat CVBlock21 [ ] =

{

0.25f, -4.25f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

19.75f, -4.25f, 0.6f, *// | \_\_\_\_\_\_\_\_\_\_\_\_ |*

19.75f, -4.25f, 0.35f, *// || ||*

0.25f, -4.25f, 0.35f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

GLfloat CVBlock22 [ ] =

{

19.75f, -4.25f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

19.75f, -2.75f, 0.6f, *// | \_\_\_\_\_\_\_\_\_\_\_\_ |*

19.75f, -2.75f, 0.35f, *// ||\_\_\_\_\_\_\_\_\_\_\_\_ |*

19.75f, -4.25f, 0.35f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

GLfloat CVBlock23 [ ] =

{

19.75f, -2.75f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

0.25f, -2.75f, 0.6f, *// | |*

0.25f, -2.75f, 0.35f, *// ||\_\_\_\_\_\_\_\_\_\_\_\_||*

19.75f, -2.75f, 0.35f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

GLfloat CVBlock24 [ ] =

{

0.25f, -2.75f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

0.25f, -4.25f, 0.6f, *// | \_\_\_\_\_\_\_\_\_\_\_\_ |*

0.25f, -4.25f, 0.35f, *// | \_\_\_\_\_\_\_\_\_\_\_\_||*

0.25f, -2.75f, 0.35f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

*//------------Нижняя панель (задняя маленькая №1)-------------//*

GLfloat CVBlock25\_1 [ ] =

{

0.25f, -3.75f, 0.35f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

19.75f, -3.75f, 0.35f, *// | |*

19.75f, -2.75f, 0.35f, *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

0.25f, -2.75f, 0.35f

};

*//------------Нижняя панель (задняя маленькая №1)-------------//*

GLfloat CVBlock25\_2 [ ] =

{

0.25f, -4.25f, 0.35f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

19.75f, -4.25f, 0.35f, *// | |*

19.75f, -3.75f, 0.35f, *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

0.25f, -3.75f, 0.35f

};

*//-------------Нижняя панель (задняя большая)--------------//*

GLfloat CVBlock26 [ ] =

{

0.0f, -4.5f, 0.0f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

20.0f, -4.5f, 0.0f, *// | |*

20.0f, -2.5f, 0.0f, *// | |*

0.0f, -2.5f, 0.0f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

*//------------Правая панель (передняя сторона)-------------//*

GLfloat CVBlock27 [ ] =

{

20.5f, 2.0f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 1 верхняя*

24.5f, 2.0f, 0.6f, *// | |*

24.5f, 2.5f, 0.6f, *// | |*

20.5f, 2.5f, 0.6f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

GLfloat CVBlock28 [ ] =

{

20.5f, 0.5f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2 верхняя*

24.5f, 0.5f, 0.6f, *// | |*

24.5f, 1.0f, 0.6f, *// | |*

20.5f, 1.0f, 0.6f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

GLfloat CVBlock29 [ ] =

{

20.5f, -1.0f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 3 верхняя*

24.5f, -1.0f, 0.6f, *// | |*

24.5f, -0.5f, 0.6f, *// | |*

20.5f, -0.5f, 0.6f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

GLfloat CVBlock30 [ ] =

{

20.5f, -2.5f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 4 верхняя*

24.5f, -2.5f, 0.6f, *// | |*

24.5f, -2.0f, 0.6f, *// | |*

20.5f, -2.0f, 0.6f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

GLfloat CVBlock54 [ ] =

{

20.5f, -4.5f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 5 верхняя*

24.5f, -4.5f, 0.6f, *// | |*

24.5f, -3.5f, 0.6f, *// | |*

20.5f, -3.5f, 0.6f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

GLfloat CVBlock31 [ ] =

{

20.0f, -4.5f, 0.6f, *// \_\_\_ левая*

20.5f, -4.5f, 0.6f, *// | |*

20.5f, 2.5f, 0.6f, *// | |*

20.0f, 2.5f, 0.6f *// |\_\_\_|*

};

GLfloat CVBlock32 [ ] =

{

24.5f, -4.5f, 0.6f, *// \_\_\_ правая*

25.0f, -4.5f, 0.6f, *// | |*

25.0f, 2.5f, 0.6f, *// | |*

24.5f, 2.5f, 0.6f *// |\_\_\_|*

};

*//----------Правая панель (внутренний контур №1)-----------//*

GLfloat CVBlock33 [ ] =

{

20.5f, 1.0f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

24.5f, 1.0f, 0.6f, *// | \_\_\_\_\_\_\_\_\_\_\_\_ |*

24.5f, 1.0f, 0.55f, *// || ||*

20.5f, 1.0f, 0.55f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

GLfloat CVBlock34 [ ] =

{

24.5f, 1.0f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

24.5f, 2.0f, 0.6f, *// | \_\_\_\_\_\_\_\_\_\_\_\_ |*

24.5f, 2.0f, 0.55f, *// ||\_\_\_\_\_\_\_\_\_\_\_\_ |*

24.5f, 1.0f, 0.55f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

GLfloat CVBlock35 [ ] =

{

24.5f, 2.0f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

20.5f, 2.0f, 0.6f, *// | |*

20.5f, 2.0f, 0.55f, *// ||\_\_\_\_\_\_\_\_\_\_\_\_||*

24.5f, 2.0f, 0.55f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

GLfloat CVBlock36 [ ] =

{

20.5f, 2.0f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

20.5f, 1.0f, 0.6f, *// | \_\_\_\_\_\_\_\_\_\_\_\_ |*

20.5f, 1.0f, 0.55f, *// | \_\_\_\_\_\_\_\_\_\_\_\_||*

20.5f, 2.0f, 0.55f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

*//----------Правая панель (внутренний контур №2)-----------//*

GLfloat CVBlock37 [ ] =

{

20.5f, -0.5f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

24.5f, -0.5f, 0.6f, *// | \_\_\_\_\_\_\_\_\_\_\_\_ |*

24.5f, -0.5f, 0.55f, *// || ||*

20.5f, -0.5f, 0.55f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

GLfloat CVBlock38 [ ] =

{

24.5f, -0.5f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

24.5f, 0.5f, 0.6f, *// | \_\_\_\_\_\_\_\_\_\_\_\_ |*

24.5f, 0.5f, 0.55f, *// ||\_\_\_\_\_\_\_\_\_\_\_\_ |*

24.5f, -0.5f, 0.55f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

GLfloat CVBlock39 [ ] =

{

24.5f, 0.5f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

20.5f, 0.5f, 0.6f, *// | |*

20.5f, 0.5f, 0.55f, *// ||\_\_\_\_\_\_\_\_\_\_\_\_||*

24.5f, 0.5f, 0.55f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

GLfloat CVBlock40 [ ] =

{

20.5f, 0.5f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

20.5f, -0.5f, 0.6f, *// | \_\_\_\_\_\_\_\_\_\_\_\_ |*

20.5f, -0.5f, 0.55f, *// | \_\_\_\_\_\_\_\_\_\_\_\_||*

20.5f, 0.5f, 0.55f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

*//----------Правая панель (внутренний контур №3)-----------//*

GLfloat CVBlock41 [ ] =

{

20.5f, -2.0f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

24.5f, -2.0f, 0.6f, *// | \_\_\_\_\_\_\_\_\_\_\_\_ |*

24.5f, -2.0f, 0.55f, *// || ||*

20.5f, -2.0f, 0.55f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

GLfloat CVBlock42 [ ] =

{

24.5f, -2.0f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

24.5f, -1.0f, 0.6f, *// | \_\_\_\_\_\_\_\_\_\_\_\_ |*

24.5f, -1.0f, 0.55f, *// ||\_\_\_\_\_\_\_\_\_\_\_\_ |*

24.5f, -2.0f, 0.55f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

GLfloat CVBlock43 [ ] =

{

24.5f, -1.0f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

20.5f, -1.0f, 0.6f, *// | |*

20.5f, -1.0f, 0.55f, *// ||\_\_\_\_\_\_\_\_\_\_\_\_||*

24.5f, -1.0f, 0.55f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

GLfloat CVBlock44 [ ] =

{

20.5f, -1.0f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

20.5f, -2.0f, 0.6f, *// | \_\_\_\_\_\_\_\_\_\_\_\_ |*

20.5f, -2.0f, 0.55f, *// | \_\_\_\_\_\_\_\_\_\_\_\_||*

20.5f, -1.0f, 0.55f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

*//----------Правая панель (внутренний контур №4)-----------//*

GLfloat CVBlock49 [ ] =

{

20.5f, -3.5f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

24.5f, -3.5f, 0.6f, *// | \_\_\_\_\_\_\_\_\_\_\_\_ |*

24.5f, -3.5f, 0.55f, *// || ||*

20.5f, -3.5f, 0.55f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

GLfloat CVBlock50 [ ] =

{

24.5f, -3.5f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

24.5f, -2.0f, 0.6f, *// | \_\_\_\_\_\_\_\_\_\_\_\_ |*

24.5f, -2.0f, 0.55f, *// ||\_\_\_\_\_\_\_\_\_\_\_\_ |*

24.5f, -3.5f, 0.55f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

GLfloat CVBlock51 [ ] =

{

24.5f, -2.0f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

20.5f, -2.0f, 0.6f, *// | |*

20.5f, -2.0f, 0.55f, *// ||\_\_\_\_\_\_\_\_\_\_\_\_||*

24.5f, -2.0f, 0.55f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

GLfloat CVBlock52 [ ] =

{

20.5f, -2.0f, 0.6f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

20.5f, -3.5f, 0.6f, *// | \_\_\_\_\_\_\_\_\_\_\_\_ |*

20.5f, -3.5f, 0.55f, *// | \_\_\_\_\_\_\_\_\_\_\_\_||*

20.5f, -2.0f, 0.55f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

*//----------Правая панель (задняя маленькая №1)-----------//*

GLfloat CVBlock45 [ ] =

{

20.5f, 1.0f, 0.55f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

24.5f, 1.0f, 0.55f, *// | |*

24.5f, 2.0f, 0.55f, *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

20.5f, 2.0f, 0.55f

};

*//----------Правая панель (задняя маленькая №2)-----------//*

GLfloat CVBlock46 [ ] =

{

20.5f, -0.5f, 0.55f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

24.5f, -0.5f, 0.55f, *// | |*

24.5f, 0.5f, 0.55f, *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

20.5f, 0.5f, 0.55f

};

*//----------Правая панель (задняя маленькая №3)-----------//*

GLfloat CVBlock47 [ ] =

{

20.5f, -2.0f, 0.55f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

24.5f, -2.0f, 0.55f, *// | |*

24.5f, -1.0f, 0.55f, *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

20.5f, -1.0f, 0.55f

};

*//----------Правая панель (задняя маленькая №4)-----------//*

GLfloat CVBlock53 [ ] =

{

20.5f, -3.5f, 0.55f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

24.5f, -3.5f, 0.55f, *// | |*

24.5f, -2.0f, 0.55f, *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

20.5f, -2.0f, 0.55f

};

*//-------------Правая панель (задняя большая)--------------//*

GLfloat CVBlock48 [ ] =

{

20.0f, -4.5f, 0.0f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

25.0f, -4.5f, 0.0f, *// | |*

25.0f, 2.5f, 0.0f, *// | |*

20.0f, 2.5f, 0.0f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

*//-------------BG--------------//*

GLfloat CVBlock55 [ ] =

{

-50.0f, -50.0f, -0.0f, *// \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

50.0f, -50.0f, -0.0f, *// | |*

50.0f, 50.0f, -0.0f, *// | |*

-50.0f, 50.0f, -0.0f *// |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|*

};

# pragma endregion

# pragma region Верщины текстурных координат

static GLfloat TCBlock01 [ ] =

{

0, 1, 1, 1, 1, 0, 0, 0

};

static GLfloat TCBlock02 [ ] =

{

0, 0, 1, 0, 1, 1, 0, 1

};

static GLfloat TCBlock03 [ ] =

{

1, 1, 1, 0, 0, 0, 0, 1

};

static GLfloat TCBlock04 [ ] =

{

1, 1, 0, 1, 0, 0, 1, 0

};

static GLfloat TCBlock05 [ ] =

{

0, 0, 0, 1, 1, 1, 1, 0

};

static GLfloat TCBlock06 [ ] =

{

1, 0, 0, 0, 0, 1, 1, 1

};

# pragma endregion

# pragma region Отрисовка OpenGL

glPushMatrix();

glEnableClientState ( GL\_VERTEX\_ARRAY );

glEnableClientState ( GL\_TEXTURE\_COORD\_ARRAY );

DrawQ ( Texture, TCBlock02, 1, CVBlock01 ); DrawQ ( Texture, TCBlock03, 2, CVBlock02 );

DrawQ ( Texture, TCBlock01, 3, CVBlock03 ); DrawQ ( Texture, TCBlock03, 4, CVBlock04 );

DrawQ ( Texture, TCBlock02, 7, CVBlock05 ); DrawQ ( Texture, TCBlock02, 8, CVBlock06 );

DrawQ ( Texture, TCBlock02, 7, CVBlock07 ); DrawQ ( Texture, TCBlock06, 8, CVBlock08 );

DrawQ ( Texture, TCBlock02, 9, CVBlock09 ); DrawQ ( Texture, TCBlock05, 10, CVBlock10 );

DrawQ ( Texture, TCBlock04, 11, CVBlock11 ); DrawQ ( Texture, TCBlock05, 12, CVBlock12 );

DrawQ ( Texture, TCBlock06, 9, CVBlock13 ); DrawQ ( Texture, TCBlock05, 10, CVBlock14 );

DrawQ ( Texture, TCBlock04, 11, CVBlock15 ); DrawQ ( Texture, TCBlock03, 12, CVBlock16 );

DrawQ ( Texture, TCBlock02, 14, CVBlock17 ); DrawQ ( Texture, TCBlock05, 15, CVBlock18 );

DrawQ ( Texture, TCBlock04, 16, CVBlock19 ); DrawQ ( Texture, TCBlock03, 17, CVBlock20 );

DrawQ ( Texture, TCBlock02, 1, CVBlock21 ); DrawQ ( Texture, TCBlock03, 2, CVBlock22 );

DrawQ ( Texture, TCBlock01, 3, CVBlock23 ); DrawQ ( Texture, TCBlock01, 4, CVBlock24 );

DrawQ ( Texture, TCBlock02, 25, CVBlock25\_1 ); DrawQ ( Texture, TCBlock02, NOT, CVBlock25\_2 );

DrawQ ( Texture, TCBlock06, 18, CVBlock26 );

DrawQ ( Texture, TCBlock02, 19, CVBlock27 ); DrawQ ( Texture, TCBlock02, 20, CVBlock28 );

DrawQ ( Texture, TCBlock02, 21, CVBlock29 ); DrawQ ( Texture, TCBlock02, 22, CVBlock30 );

DrawQ ( Texture, TCBlock02, 23, CVBlock31 ); DrawQ ( Texture, TCBlock02, 24, CVBlock32 );

DrawQ ( Texture, TCBlock02, 1, CVBlock33 ); DrawQ ( Texture, TCBlock03, 2, CVBlock34 );

DrawQ ( Texture, TCBlock01, 3, CVBlock35 ); DrawQ ( Texture, TCBlock01, 4, CVBlock36 );

DrawQ ( Texture, TCBlock02, 1, CVBlock37 ); DrawQ ( Texture, TCBlock03, 2, CVBlock38 );

DrawQ ( Texture, TCBlock01, 3, CVBlock39 ); DrawQ ( Texture, TCBlock01, 4, CVBlock40 );

DrawQ ( Texture, TCBlock02, 1, CVBlock41 ); DrawQ ( Texture, TCBlock03, 2, CVBlock42 );

DrawQ ( Texture, TCBlock01, 3, CVBlock43 ); DrawQ ( Texture, TCBlock01, 4, CVBlock44 );

DrawQ ( Texture, TCBlock01, 36, CVBlock45 ); DrawQ ( Texture, TCBlock01, 36, CVBlock46 );

DrawQ ( Texture, TCBlock01, 36, CVBlock47 ); DrawQ ( Texture, TCBlock06, 37, CVBlock48 );

DrawQ ( Texture, TCBlock02, 1, CVBlock49 ); DrawQ ( Texture, TCBlock03, 2, CVBlock50 );

DrawQ ( Texture, TCBlock01, 3, CVBlock51 ); DrawQ ( Texture, TCBlock01, 4, CVBlock52 );

DrawQ ( Texture, TCBlock01, 36, CVBlock53 ); DrawQ ( Texture, TCBlock02, 38, CVBlock54 );

glDisableClientState ( GL\_TEXTURE\_COORD\_ARRAY );

glDisableClientState ( GL\_VERTEX\_ARRAY );

glPopMatrix();

# pragma endregion

}

void GLelements :: CreateQuadNoteLine ( float X, float Y, float Z, int R, int G, int B )

{

glTranslatef ( X, Y, Z );

# pragma region Координаты вершин 4-угольника

GLfloat CVertex3OktElem01 [ ] =

{

0.0f, -4.25f, 0.38f,

0.0f, -2.75f, 0.38f,

0.025f, -2.75f, 0.38f,

0.025f, -4.25f, 0.38f

};

GLfloat CVertex3OktElem02 [ ] =

{

0.025f, -4.25f, 0.38f,

0.025f, -4.25f, 0.35f,

0.025f, -2.75f, 0.35f,

0.025f, -2.75f, 0.38f

};

GLfloat CVertex3OktElem03 [ ] =

{

0.0f, -4.25f, 0.35f,

0.0f, -4.25f, 0.38f,

0.0f, -2.75f, 0.38f,

0.0f, -2.75f, 0.35f

};

# pragma endregion

# pragma region Номера вершин 4-угольника

static GLushort QVertex3 [ ] =

{

0, 1, 2, 3

};

# pragma endregion

# pragma region Отрисовка OpenGL

glPushMatrix();

glEnable ( GL\_LINE );

glEnable ( GL\_LINE\_SMOOTH );

glEnableClientState ( GL\_VERTEX\_ARRAY );

glLineWidth ( 1.0f );

glColor3ub ( R, G, B );

glVertexPointer ( 3, GL\_FLOAT, 0, CVertex3OktElem01 );

glDrawElements ( GL\_QUADS, 4, GL\_UNSIGNED\_SHORT, QVertex3 );

glVertexPointer ( 3, GL\_FLOAT, 0, CVertex3OktElem02 );

glDrawElements ( GL\_QUADS, 4, GL\_UNSIGNED\_SHORT, QVertex3 );

glVertexPointer ( 3, GL\_FLOAT, 0, CVertex3OktElem03 );

glDrawElements ( GL\_QUADS, 4, GL\_UNSIGNED\_SHORT, QVertex3 );

glColor3ub ( 255, 255, 255 );

glPopName ( );

glLineWidth ( 1.0f );

glDisableClientState ( GL\_VERTEX\_ARRAY );

glDisable ( GL\_LINE\_SMOOTH );

glPopMatrix ( );

# pragma endregion

glTranslatef ( -X, -Y, -Z );

}

void GLelements :: CreateTimeNum\_0 ( float X, float Y, float Z, GLfloat HEIGHT, GLfloat RADIUS, GLfloat size, int R, int G, int B )

{

glTranslatef ( X, Y, Z );

glTranslatef ( 0.0f, 0.8f \* size, 0.0f );

glRotatef ( 90.0f, 0.0f, 0.0f, 1.0f ); *//*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_|*

glRotatef ( -90.0f, 0.0f, 0.0f, 1.0f ); *// |\_|*

glTranslatef ( 0.0f, -0.8f \* size, 0.0f );

glTranslatef ( 0.0f, -0.8f \* size, 0.0f ); *// \_*

glRotatef ( 90.0f, 0.0f, 0.0f, 1.0f ); *// |\_|*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// | |*

glRotatef ( -90.0f, 0.0f, 0.0f, 1.0f );

glTranslatef ( 0.0f, 0.8f \* size, 0.0f ); *//*

glTranslatef ( -0.4f \* size, 0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// \_|*

glTranslatef ( 0.4f \* size, -0.4f \* size, 0.0f ); *// |\_|*

glTranslatef ( 0.4f \* size, 0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_*

glTranslatef ( -0.4f \* size, -0.4f \* size, 0.0f ); *// |\_|*

glTranslatef ( -0.4f \* size, -0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_|*

glTranslatef ( 0.4f \* size, 0.4f \* size, 0.0f ); *// \_|*

glTranslatef ( 0.4f \* size, -0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_|*

glTranslatef ( -0.4f \* size, 0.4f \* size, 0.0f ); *// |\_*

glTranslatef ( -X,- Y, -Z );

}

void GLelements :: CreateTimeNum\_1 ( float X, float Y, float Z, GLfloat HEIGHT, GLfloat RADIUS, GLfloat size, int R, int G, int B )

{

glTranslatef ( X, Y, Z );

glTranslatef ( 0.4f \* size, 0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_*

glTranslatef ( -0.4f \* size, -0.4f \* size, 0.0f ); *// |\_|*

glTranslatef ( 0.4f \* size, -0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_|*

glTranslatef ( -0.4f \* size, 0.4f \* size, 0.0f ); *// |\_*

glTranslatef ( -X,- Y, -Z );

}

void GLelements :: CreateTimeNum\_2 ( float X, float Y, float Z, GLfloat HEIGHT, GLfloat RADIUS, GLfloat size, int R, int G, int B )

{

glTranslatef ( X, Y, Z );

glTranslatef ( 0.0f, 0.8f \* size, 0.0f ); *//*

glRotatef ( 90.0f, 0.0f, 0.0f, 1.0f ); *//*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_|*

glRotatef ( -90.0f, 0.0f, 0.0f, 1.0f ); *// |\_|*

glTranslatef ( 0.0f, -0.8f \* size, 0.0f );

glRotatef ( 90.0f, 0.0f, 0.0f, 1.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// | |*

glRotatef ( -90.0f, 0.0f, 0.0f, 1.0f ); *// |\_|*

glTranslatef ( 0.0f, -0.8f \* size, 0.0f ); *// \_*

glRotatef ( 90.0f, 0.0f, 0.0f, 1.0f ); *// |\_|*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// | |*

glRotatef ( -90.0f, 0.0f, 0.0f, 1.0f );

glTranslatef ( 0.0f, 0.8f \* size, 0.0f );

glTranslatef ( 0.4f \* size, 0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_*

glTranslatef ( -0.4f \* size, -0.4f \* size, 0.0f ); *// |\_|*

glTranslatef ( -0.4f \* size, -0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_|*

glTranslatef ( 0.4f \* size, 0.4f \* size, 0.0f ); *// \_|*

glTranslatef ( -X,- Y, -Z );

}

void GLelements :: CreateTimeNum\_3 ( float X, float Y, float Z, GLfloat HEIGHT, GLfloat RADIUS, GLfloat size, int R, int G, int B )

{

glTranslatef ( X, Y, Z );

glTranslatef ( 0.0f, 0.8f \* size, 0.0f );

glRotatef ( 90.0f, 0.0f, 0.0f, 1.0f ); *//*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_|*

glRotatef ( -90.0f, 0.0f, 0.0f, 1.0f ); *// |\_|*

glTranslatef ( 0.0f, -0.8f \* size, 0.0f );

glRotatef ( 90.0f, 0.0f, 0.0f, 1.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// | |*

glRotatef ( -90.0f, 0.0f, 0.0f, 1.0f ); *// |\_|*

glTranslatef ( 0.0f, -0.8f \* size, 0.0f ); *// \_*

glRotatef ( 90.0f, 0.0f, 0.0f, 1.0f ); *// |\_|*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// | |*

glRotatef ( -90.0f, 0.0f, 0.0f, 1.0f );

glTranslatef ( 0.0f, 0.8f \* size, 0.0f );

glTranslatef ( 0.4f \* size, 0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_*

glTranslatef ( -0.4f \* size, -0.4f \* size, 0.0f ); *// |\_|*

glTranslatef ( 0.4f \* size, -0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_|*

glTranslatef ( -0.4f \* size, 0.4f \* size, 0.0f ); *// |\_*

glTranslatef ( -X,- Y, -Z );

}

void GLelements :: CreateTimeNum\_4 ( float X, float Y, float Z, GLfloat HEIGHT, GLfloat RADIUS, GLfloat size, int R, int G, int B )

{

glTranslatef ( X, Y, Z );

glRotatef ( 90.0f, 0.0f, 0.0f, 1.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// | |*

glRotatef ( -90.0f, 0.0f, 0.0f, 1.0f ); *// |\_|*

glTranslatef ( -0.4f \* size, 0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// \_|*

glTranslatef ( 0.4f \* size, -0.4f \* size, 0.0f ); *// |\_|*

glTranslatef ( 0.4f \* size, 0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_*

glTranslatef ( -0.4f \* size, -0.4f \* size, 0.0f ); *// |\_|*

glTranslatef ( 0.4f \* size, -0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_|*

glTranslatef ( -0.4f \* size, 0.4f \* size, 0.0f ); *// |\_*

glTranslatef ( -X,- Y, -Z );

}

void GLelements :: CreateTimeNum\_5 ( float X, float Y, float Z, GLfloat HEIGHT, GLfloat RADIUS, GLfloat size, int R, int G, int B )

{

glTranslatef ( X, Y, Z );

glTranslatef ( 0.0f, 0.8f \* size, 0.0f );

glRotatef ( 90.0f, 0.0f, 0.0f, 1.0f );

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_|*

glRotatef ( -90.0f, 0.0f, 0.0f, 1.0f ); *// |\_|*

glTranslatef ( 0.0f, -0.8f \* size, 0.0f );

glRotatef ( 90.0f, 0.0f, 0.0f, 1.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// | |*

glRotatef ( -90.0f, 0.0f, 0.0f, 1.0f ); *// |\_|*

glTranslatef ( 0.0f, -0.8f \* size, 0.0f ); *// \_*

glRotatef ( 90.0f, 0.0f, 0.0f, 1.0f ); *// |\_|*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// | |*

glRotatef ( -90.0f, 0.0f, 0.0f, 1.0f );

glTranslatef ( 0.0f, 0.8f \* size, 0.0f );

glTranslatef ( -0.4f \* size, 0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// \_|*

glTranslatef ( 0.4f \* size, -0.4f \* size, 0.0f ); *// |\_|*

glTranslatef ( 0.4f \* size, -0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_|*

glTranslatef ( -0.4f \* size, 0.4f \* size, 0.0f ); *// |\_*

glTranslatef ( -X,- Y, -Z );

}

void GLelements :: CreateTimeNum\_6 ( float X, float Y, float Z, GLfloat HEIGHT, GLfloat RADIUS, GLfloat size, int R, int G, int B )

{

glTranslatef ( X, Y, Z );

glTranslatef ( 0.0f, 0.8f \* size, 0.0f );

glRotatef ( 90.0f, 0.0f, 0.0f, 1.0f );

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_|*

glRotatef ( -90.0f, 0.0f, 0.0f, 1.0f ); *// |\_|*

glTranslatef ( 0.0f, -0.8f \* size, 0.0f );

glRotatef ( 90.0f, 0.0f, 0.0f, 1.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// | |*

glRotatef ( -90.0f, 0.0f, 0.0f, 1.0f ); *// |\_|*

glTranslatef ( 0.0f, -0.8f \* size, 0.0f ); *// \_*

glRotatef ( 90.0f, 0.0f, 0.0f, 1.0f ); *// |\_|*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// | |*

glRotatef ( -90.0f, 0.0f, 0.0f, 1.0f );

glTranslatef ( 0.0f, 0.8f \* size, 0.0f );

glTranslatef ( -0.4f \* size, 0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// \_|*

glTranslatef ( 0.4f \* size, -0.4f \* size, 0.0f ); *// |\_|*

glTranslatef ( -0.4f \* size, -0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_|*

glTranslatef ( 0.4f \* size, 0.4f \* size, 0.0f ); *// \_|*

glTranslatef ( 0.4f \* size, -0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_|*

glTranslatef ( -0.4f \* size, 0.4f \* size, 0.0f ); *// |\_*

glTranslatef ( -X,- Y, -Z );

}

void GLelements :: CreateTimeNum\_7 ( float X, float Y, float Z, GLfloat HEIGHT, GLfloat RADIUS, GLfloat size, int R, int G, int B )

{

glTranslatef ( X, Y, Z );

glTranslatef ( 0.0f, 0.8f \* size, 0.0f );

glRotatef ( 90.0f, 0.0f, 0.0f, 1.0f );

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_|*

glRotatef ( -90.0f, 0.0f, 0.0f, 1.0f ); *// |\_|*

glTranslatef ( 0.0f, -0.8f \* size, 0.0f );

glTranslatef ( 0.4f \* size, 0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_*

glTranslatef ( -0.4f \* size, -0.4f \* size, 0.0f ); *// |\_|*

glTranslatef ( 0.4f \* size, -0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_|*

glTranslatef ( -0.4f \* size, 0.4f \* size, 0.0f ); *// |\_*

glTranslatef ( -X,- Y, -Z );

}

void GLelements :: CreateTimeNum\_8 ( float X, float Y, float Z, GLfloat HEIGHT, GLfloat RADIUS, GLfloat size, int R, int G, int B )

{

glTranslatef ( X, Y, Z );

glTranslatef ( 0.0f, 0.8f \* size, 0.0f );

glRotatef ( 90.0f, 0.0f, 0.0f, 1.0f );

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_|*

glRotatef ( -90.0f, 0.0f, 0.0f, 1.0f ); *// |\_|*

glTranslatef ( 0.0f, -0.8f \* size, 0.0f );

glRotatef ( 90.0f, 0.0f, 0.0f, 1.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// | |*

glRotatef ( -90.0f, 0.0f, 0.0f, 1.0f ); *// |\_|*

glTranslatef ( 0.0f, -0.8f \* size, 0.0f ); *// \_*

glRotatef ( 90.0f, 0.0f, 0.0f, 1.0f ); *// |\_|*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// | |*

glRotatef ( -90.0f, 0.0f, 0.0f, 1.0f );

glTranslatef ( 0.0f, 0.8f \* size, 0.0f );

glTranslatef ( -0.4f \* size, 0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// \_|*

glTranslatef ( 0.4f \* size, -0.4f \* size, 0.0f ); *// |\_|*

glTranslatef ( 0.4f \* size, 0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_*

glTranslatef ( -0.4f \* size, -0.4f \* size, 0.0f ); *// |\_|*

glTranslatef ( -0.4f \* size, -0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_|*

glTranslatef ( 0.4f \* size, 0.4f \* size, 0.0f ); *// \_|*

glTranslatef ( 0.4f \* size, -0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_|*

glTranslatef ( -0.4f \* size, 0.4f \* size, 0.0f ); *// |\_*

glTranslatef ( -X,- Y, -Z );

}

void GLelements :: CreateTimeNum\_9 ( float X, float Y, float Z, GLfloat HEIGHT, GLfloat RADIUS, GLfloat size, int R, int G, int B )

{

glTranslatef ( X, Y, Z );

glTranslatef ( 0.0f, 0.8f \* size, 0.0f );

glRotatef ( 90.0f, 0.0f, 0.0f, 1.0f );

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_|*

glRotatef ( -90.0f, 0.0f, 0.0f, 1.0f ); *// |\_|*

glTranslatef ( 0.0f, -0.8f \* size, 0.0f );

glRotatef ( 90.0f, 0.0f, 0.0f, 1.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// | |*

glRotatef ( -90.0f, 0.0f, 0.0f, 1.0f ); *// |\_|*

glTranslatef ( 0.0f, -0.8f \* size, 0.0f ); *// \_*

glRotatef ( 90.0f, 0.0f, 0.0f, 1.0f ); *// |\_|*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// | |*

glRotatef ( -90.0f, 0.0f, 0.0f, 1.0f );

glTranslatef ( 0.0f, 0.8f \* size, 0.0f );

glTranslatef ( -0.4f \* size, 0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// \_|*

glTranslatef ( 0.4f \* size, -0.4f \* size, 0.0f ); *// |\_|*

glTranslatef ( 0.4f \* size, 0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_*

glTranslatef ( -0.4f \* size, -0.4f \* size, 0.0f ); *// |\_|*

glTranslatef ( 0.4f \* size, -0.4f \* size, 0.0f ); *// \_*

CreateTimeNumElement ( HEIGHT \* size, RADIUS \* size, R, G, B ); *// |\_|*

glTranslatef ( -0.4f \* size, 0.4f \* size, 0.0f ); *// |\_*

glTranslatef ( -X,- Y, -Z );

}

# endif

**<sound.h>**

# ifndef \_SOUND\_H

# define \_SOUND\_H

# include "main.h"

# ifdef \_MAIN\_H

# define M\_PI 3.14159265358979323846

class Sound

{

private:

int SoundCreate;

LPDIRECTSOUNDBUFFER DsBuffer;

int BitsPerSample;

int SamplesPerSec;

int frequency;

short \*bufferBytes;

DWORD lockedSize;

WAVEFORMATEX wfx;

DSBUFFERDESC dsbd;

LPDIRECTSOUND DS;

float pos;

float r;

float value;

DWORD q;

int flagi;

public:

Sound ( )

{

SoundCreate = 0;

DsBuffer = NULL;

BitsPerSample = 16;

SamplesPerSec = 96000;

frequency = 440;

DS = NULL;

flagi = 1;

}

int get\_frequency ( )

{

return frequency;

}

void set\_frequency ( int set )

{

int ch = 0;

ch = fmod ( (float) set, 2 );

if ( ch == 1 )

frequency = set + 1;

else

frequency = set;

}

int get\_flagi ( ) { return flagi; }

void set\_flagi ( int f ) { flagi = f; }

int InitAudio ( LPDIRECTSOUNDBUFFER \*buffer, int secs, int samplerate, int bitspersample );

void FillBuffer ( LPDIRECTSOUNDBUFFER buffer, float freq, int size, int samplerate );

void SoundString ( bool SoundFlag );

};

# endif

# endif

**<sound.cpp>**

# include "stdafx.h"

# include "sound.h"

# ifdef \_SOUND\_H

int Sound :: InitAudio ( LPDIRECTSOUNDBUFFER \*buffer, int secs, int samplerate, int bitspersample )

{

ZeroMemory(&wfx, sizeof(WAVEFORMATEX));

wfx.wFormatTag = WAVE\_FORMAT\_PCM;

wfx.nChannels = 1;

wfx.nSamplesPerSec = samplerate;

wfx.wBitsPerSample = bitspersample;

wfx.nBlockAlign = (WORD)(wfx.wBitsPerSample \* wfx.nChannels / 8);

wfx.nAvgBytesPerSec = wfx.nSamplesPerSec \* wfx.nBlockAlign;

ZeroMemory(&dsbd, sizeof(DSBUFFERDESC));

dsbd.dwSize = sizeof(DSBUFFERDESC);

dsbd.dwFlags = DSBCAPS\_GLOBALFOCUS;

dsbd.dwBufferBytes = samplerate \* secs;

dsbd.lpwfxFormat = &wfx;

DirectSoundCreate(NULL, &DS, NULL);

DS->SetCooperativeLevel(GetForegroundWindow(), DSSCL\_PRIORITY);

DS->CreateSoundBuffer(&dsbd, buffer, NULL);

*//Возвращаем размер буфера*

return dsbd.dwBufferBytes;

}

void Sound :: FillBuffer ( LPDIRECTSOUNDBUFFER buffer, float freq, int size, int samplerate )

{

*//Запираем буфер*

buffer->Lock(0, size, (void \*\*)(&bufferBytes), &lockedSize, NULL, NULL, 0L);

for ( q = 0; q < lockedSize / 2; q++)

{

*//Определяем цикл, в котором находимся*

pos = freq / (float)samplerate \* (float)q;

*//Берём остаток и переводим в радианы*

r = (pos - floor(pos)) \* 2 \* M\_PI;

value = sin(r);

bufferBytes[q] = value \* 32767 / 8;

}

*//Отпираем буфер*

buffer->Unlock(bufferBytes, lockedSize, NULL, 0);

}

void Sound :: SoundString ( bool SoundFlag )

{

if ( flagi == 1 )

{

SoundCreate = InitAudio ( &DsBuffer, 1, SamplesPerSec, BitsPerSample);

set\_flagi ( 2 );

}

FillBuffer ( DsBuffer, frequency, SoundCreate, SamplesPerSec );

*//440 Гц - "Ля" первой октавы*

if ( SoundFlag )

DsBuffer -> Play ( 0, 0, DSBPLAY\_LOOPING );

if ( !SoundFlag )

{

DsBuffer -> Stop ( );

*//DsBuffer -> Release ( );*

*//DS -> Release ( );*

}

}

# endif

**<main.h>**

# ifndef \_MAIN\_H

# define \_MAIN\_H

*// Хидеры, необходимые для работы программы*

# include <windows.h>

# include <stdio.h>

# include <string.h>

# include <math.h>

# include <time.h>

# include <Mmsystem.h>

# include <WinUser.h>

# include "GL\GL.H"

# include "GL\GLU.H"

# include "GL\glut.h"

# include "GL\GLAux.h"

# include <dsound.h>

# include "StringPhysics.h"

# include "Control.h"

# pragma comment ( lib, "winmm.lib" )

*// Звуковая библиотека*

# pragma comment ( lib, "dsound.lib" )

*// Обьявим глобальные переменные, ширину, высоту и глубину цвета экрана*

# define SCREEN\_WIDTH 800

# define SCREEN\_HEIGHT 600

# define SCREEN\_DEPTH 16

*// Глобальные параметры окна; будут доступны из других файлов:*

extern HWND g\_hWnd;

extern RECT g\_rRect;

extern HDC g\_hDC;

extern HGLRC g\_hRC;

extern HINSTANCE g\_hInstance;

*// Функция - главный цикл программы*

WPARAM MainLoop();

*// Функция, устанавливающая формат пиксела*

bool bSetupPixelFormat(HDC hdc);

*// Прототип функции, устанавливающей размеры окна OpenGL*

void SizeOpenGLScreen(int width, int height);

*// Функция, инициализирующая OpenGL*

void InitializeOpenGL(int width, int height);

*// Общая инициализация*

void InitOpenGL(HWND hWnd);

*// Инициализация звука*

int InitAudio(LPDIRECTSOUNDBUFFER \*buffer, int secs, int samplerate, int bitspersample);

void FillBuffer(LPDIRECTSOUNDBUFFER buffer, float freq, int size, int samplerate, int bitsPerSample);

*// Функция, которая собственно рисует сцену*

void RenderScene();

*// Де-инициализация*

void DeInit();

# endif

**<stdafx.h>**

*// stdafx.h: включаемый файл для стандартных системных включаемых файлов*

*// или включаемых файлов для конкретного проекта, которые часто используются, но*

*// не часто изменяются*

*//*

#pragma once

#include "targetver.h"

#define WIN32\_LEAN\_AND\_MEAN

*// Файлы заголовков Windows:*

#include <windows.h>

*// Файлы заголовков C RunTime*

#include <stdlib.h>

#include <malloc.h>

#include <memory.h>

#include <tchar.h>

*//Мои заголовки*

#include <commctrl.h>

#include <iostream>

#include <typeinfo.h>

#include <locale.h>

#include <time.h>

#include <omp.h>

**<Control.h>**

# ifndef \_CONTROL\_H

# define \_CONTROL\_H

# include <Windows.h>

# define ButtonStop\_ID 1

# define ButtonStandart\_ID 3

# define L\_TB\_ID 4

# define L\_Ed\_ID 5

# define L\_UpB\_ID 6

# define L\_DownB\_ID 7

# define p\_TB\_ID 12

# define p\_Ed\_ID 13

# define p\_UpB\_ID 14

# define p\_DownB\_ID 15

# define Fn\_TB\_ID 16

# define Fn\_Ed\_ID 17

# define Fn\_UpB\_ID 18

# define Fn\_DownB\_ID 19

# define Ao\_TB\_ID 20

# define Ao\_Ed\_ID 21

# define Ao\_UpB\_ID 22

# define Ao\_DownB\_ID 23

# define vol\_CB\_ID 24

# define time\_TB\_ID 25

# define OGL\_ID 40

typedef class tagCONTROL

{

private:

char FirstEditsIndex[5];

*// панель настроек высотой 120-130*

*// дескрипторы контролов и OGL окна*

HWND ButtonStop, ButtonStandart;

HWND OGL;

HWND L\_TB, L\_St,L\_St1, L\_Ed, L\_UpB, L\_DownB;

HWND p\_TB, p\_St,p\_St1, p\_Ed, p\_UpB, p\_DownB;

HWND Fn\_TB, Fn\_St, Fn\_Ed, Fn\_UpB, Fn\_DownB;

HWND Ao\_TB, Ao\_St, Ao\_St1, Ao\_Ed, Ao\_UpB, Ao\_DownB;

HWND vol\_St, vol\_CB;

HWND time\_TB, time\_St,time\_St2;

*//главные дескрипторы*

HWND \_hWnd;

HINSTANCE \_hInst;

*//вспомогательные параметры*

int maxX, maxY, FmaxX, FmaxY;

HDC hdc;

HFONT hFont;

HPEN MainPen, OldPen;

HBRUSH MainBrush, OldBrush;

int par;

wchar\_t lpch[20];

*//установка значений по умолчанию*

void Standart();

*//установка шрифтов*

inline void set\_Font( \_\_in LPCWSTR, \_\_in int);

inline void Font\_to\_Ctrl\_Buttons();

inline void Font\_to\_Ctrl\_StringParam();

inline void Font\_to\_TimeTB();

inline void Font\_to\_volTB();

*//загрузка дескрипторов*

inline void set\_hWnd( \_\_in HWND);

inline void set\_hInstance( \_\_in HINSTANCE);

*//создание панелей и окон*

inline void CreateButtons();

inline void CreateOGLWindow( \_\_in WNDPROC);

inline void CreateStringParamOption();

inline void CreateTimeSpeedPanel();

inline void CreateVolumeButton();

public:

*//конструктор*

tagCONTROL( \_\_in HWND, \_\_in HINSTANCE);

*//создание панели*

void CreateInterfacePanel( \_\_in WNDPROC WndProc);

*//команда реакции*

LRESULT CALLBACK ReactionAllButtonsAndEdits(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);

LRESULT CALLBACK ReactionAllTrackBars(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);

*//команда, вызываемая при изменении размеров окна*

void MoveAllWnd();

*//get-set parameter*

HWND GetOGLHWND();

int \_message\_value;

int \_message\_typename;

}CONTROL,\*LPCONTROL;

# endif

**<Control.cpp>**

# include "stdafx.h"

# include "Control.h"

# include "resource.h"

# ifdef \_CONTROL\_H

tagCONTROL::tagCONTROL( \_\_in HWND hWnd, \_\_in HINSTANCE hInstance)

{

InitCommonControls();

set\_hInstance(hInstance);

set\_hWnd(hWnd);

RECT rt;

GetClientRect (\_hWnd, &rt);

maxX = rt.right, maxY = rt.bottom;

FmaxX = maxX; FmaxY = maxY;

ZeroMemory(FirstEditsIndex,sizeof(bool)\*5);

}

void tagCONTROL::set\_hWnd( \_\_in HWND hWnd)

{

\_hWnd = hWnd;

}

void tagCONTROL::set\_hInstance( \_\_in HINSTANCE hInst)

{

\_hInst = hInst;

}

void tagCONTROL::set\_Font( \_\_in LPCWSTR font\_name, \_\_in int Height)

{

hFont = CreateFont( Height, 0, 0, 0, 500,

0, 0, 0, 1, 0,

CLIP\_DEFAULT\_PRECIS, PROOF\_QUALITY,

FIXED\_PITCH, font\_name);

}

void tagCONTROL::Font\_to\_Ctrl\_Buttons()

{

set\_Font(L"MS Sans Serif",14);

SendMessage( ButtonStop, WM\_SETFONT, (WPARAM) hFont, 0 );

SendMessage( ButtonStandart, WM\_SETFONT, (WPARAM) hFont, 0 );

}

void tagCONTROL::Font\_to\_Ctrl\_StringParam()

{

set\_Font(L"Arial",16);

SendMessage( L\_Ed, WM\_SETFONT, (WPARAM) hFont, 0 );

SendMessage( p\_Ed, WM\_SETFONT, (WPARAM) hFont, 0 );

SendMessage( Fn\_Ed, WM\_SETFONT, (WPARAM) hFont, 0 );

SendMessage( Ao\_Ed, WM\_SETFONT, (WPARAM) hFont, 0 );

set\_Font(L"Arial",14);

SendMessage( L\_St, WM\_SETFONT, (WPARAM) hFont, 0 );

SendMessage( p\_St, WM\_SETFONT, (WPARAM) hFont, 0 );

SendMessage( Fn\_St, WM\_SETFONT, (WPARAM) hFont, 0 );

SendMessage( Ao\_St, WM\_SETFONT, (WPARAM) hFont, 0 );

set\_Font(L"Arial", 10);

SendMessage( L\_UpB, WM\_SETFONT, (WPARAM) hFont, 0 );

SendMessage( p\_UpB, WM\_SETFONT, (WPARAM) hFont, 0 );

SendMessage( Fn\_UpB, WM\_SETFONT, (WPARAM) hFont, 0 );

SendMessage( Ao\_UpB, WM\_SETFONT, (WPARAM) hFont, 0 );

SendMessage( L\_DownB, WM\_SETFONT, (WPARAM) hFont, 0 );

SendMessage( p\_DownB, WM\_SETFONT, (WPARAM) hFont, 0 );

SendMessage( Fn\_DownB, WM\_SETFONT, (WPARAM) hFont, 0 );

SendMessage( Ao\_DownB, WM\_SETFONT, (WPARAM) hFont, 0 );

SendMessage( L\_St1, WM\_SETFONT, (WPARAM) hFont, 0 );

SendMessage( p\_St1, WM\_SETFONT, (WPARAM) hFont, 0 );

SendMessage( Ao\_St1, WM\_SETFONT, (WPARAM) hFont, 0 );

}

void tagCONTROL::Font\_to\_TimeTB()

{

set\_Font(L"MS Sans Serif",14);

SendMessage( time\_St, WM\_SETFONT, (WPARAM) hFont, 0 );

SendMessage( time\_St2, WM\_SETFONT, (WPARAM) hFont, 0 );

}

void tagCONTROL::Font\_to\_volTB()

{

set\_Font(L"MS Sans Serif",12);

SendMessage( vol\_CB, WM\_SETFONT, (WPARAM) hFont, 0 );

SendMessage( vol\_St, WM\_SETFONT, (WPARAM) hFont, 0 );

}

void tagCONTROL::CreateButtons()

{

ButtonStop = CreateWindow (L"button", L"Остановить", WS\_CHILD|WS\_VISIBLE|WS\_CLIPSIBLINGS|BS\_PUSHBUTTON, maxX - 105, maxY - 100, 100, 40, \_hWnd,(HMENU)ButtonStop\_ID, \_hInst,NULL);

*//ButtonSetting = CreateWindow(L"button", L"Настройка", WS\_CHILD|WS\_VISIBLE|WS\_CLIPSIBLINGS|BS\_PUSHBUTTON, maxX - 105, maxY - 75, 100, 30, \_hWnd,(HMENU)ButtonSetting\_ID, \_hInst,NULL);*

ButtonStandart = CreateWindow(L"button",L"По умолчанию",WS\_CHILD|WS\_VISIBLE|WS\_CLIPSIBLINGS|BS\_PUSHBUTTON, maxX - 105, maxY - 50, 100, 40, \_hWnd,(HMENU)ButtonStandart\_ID,\_hInst,NULL);

Font\_to\_Ctrl\_Buttons();

}

void tagCONTROL::CreateOGLWindow( \_\_in WNDPROC WndProc1)

{

WNDCLASS wndclass;

memset ( &wndclass, 0, sizeof ( WNDCLASS ) );

wndclass.style = CS\_HREDRAW | CS\_VREDRAW;

wndclass.lpfnWndProc = WndProc1;

wndclass.hbrBackground = CreateSolidBrush(RGB(255,255,255));

wndclass.lpszClassName = L"OGLWNDCL";

wndclass.hCursor = LoadCursor(NULL, IDC\_ARROW);

RegisterClass ( &wndclass );

OGL = CreateWindow( L"OGLWNDCL", NULL, WS\_CHILD|WS\_CLIPSIBLINGS|WS\_VISIBLE, 0, 0,maxX, maxY - 130, \_hWnd, (HMENU) OGL\_ID, \_hInst, NULL);

}

void tagCONTROL::CreateStringParamOption()

{

L\_TB = CreateWindow(TRACKBAR\_CLASS, NULL, WS\_CHILD|WS\_VISIBLE|TBS\_HORZ|TBS\_TOOLTIPS, 20, maxY - 33, 240, 30, \_hWnd,(HMENU)L\_TB\_ID, \_hInst, NULL);

SendMessage(L\_TB, TBM\_SETRANGE, (WPARAM) TRUE, (LPARAM) MAKELONG(10, 1000)); *// min. & max. positions*

SendMessage(L\_TB, TBM\_SETPAGESIZE, 0, (LPARAM) 1); *// скорость перетаскивания бегунка*

SendMessage(L\_TB, TBM\_SETSEL,(WPARAM) FALSE,(LPARAM) MAKELONG(0, 1));

SendMessage(L\_TB, TBM\_SETPOS,(WPARAM) TRUE, (LPARAM) 1000); *// положение бегунка*

L\_St = CreateWindow(L"static",L"Длина, 10 м", WS\_CHILD|WS\_VISIBLE|SS\_CENTER, 25, maxY - 48, 75,15, \_hWnd,(HMENU)0,\_hInst,NULL);

L\_St1 = CreateWindow(L"static",L"-3", WS\_CHILD|WS\_VISIBLE|SS\_CENTER, 78, maxY - 50, 10,10, \_hWnd,(HMENU)0,\_hInst,NULL);

L\_Ed = CreateWindow(L"edit", L"1000", WS\_CHILD|WS\_VISIBLE|WS\_CLIPSIBLINGS|WS\_BORDER|ES\_LEFT|ES\_AUTOHSCROLL, 262, maxY - 33, 50, 20, \_hWnd,(HMENU)L\_Ed\_ID, \_hInst, NULL);

L\_UpB = CreateWindow(L"button",L"Ʌ", WS\_CHILD|WS\_VISIBLE|WS\_CLIPSIBLINGS|BS\_PUSHBUTTON, 314, maxY - 33, 14, 11, \_hWnd,(HMENU)L\_UpB\_ID,\_hInst,NULL);

L\_DownB = CreateWindow(L"button",L"V", WS\_CHILD|WS\_VISIBLE|WS\_CLIPSIBLINGS|BS\_PUSHBUTTON, 314, maxY - 23, 14, 11, \_hWnd,(HMENU)L\_DownB\_ID,\_hInst,NULL);

p\_TB = CreateWindow(TRACKBAR\_CLASS, NULL, WS\_CHILD|WS\_VISIBLE|TBS\_HORZ|TBS\_TOOLTIPS, 335, maxY - 33, 170, 30, \_hWnd,(HMENU)p\_TB\_ID, \_hInst, NULL);

SendMessage(p\_TB, TBM\_SETRANGE, (WPARAM) TRUE, (LPARAM) MAKELONG(1, 50));

SendMessage(p\_TB, TBM\_SETPAGESIZE, 0, (LPARAM) 1);

SendMessage(p\_TB, TBM\_SETSEL,(WPARAM) FALSE,(LPARAM) MAKELONG(0, 10));

SendMessage(p\_TB, TBM\_SETPOS,(WPARAM) TRUE, (LPARAM) 10);

p\_St = CreateWindow(L"static",L"Плотность, 10 гр/мм", WS\_CHILD|WS\_VISIBLE|SS\_CENTER, 345, maxY - 48, 113,15, \_hWnd,(HMENU)0,\_hInst,NULL);

p\_St1= CreateWindow(L"static",L"-1", WS\_CHILD|WS\_VISIBLE|SS\_CENTER, 419, maxY - 50, 10,10, \_hWnd,(HMENU)0,\_hInst,NULL);

p\_Ed = CreateWindow(L"edit", L"10", WS\_CHILD|WS\_VISIBLE|WS\_CLIPSIBLINGS|WS\_BORDER|ES\_LEFT|ES\_AUTOHSCROLL, 510, maxY - 33, 40, 20, \_hWnd,(HMENU)p\_Ed\_ID, \_hInst, NULL);

p\_UpB = CreateWindow(L"button",L"Ʌ", WS\_CHILD|WS\_VISIBLE|WS\_CLIPSIBLINGS|BS\_PUSHBUTTON, 552, maxY - 33, 14, 11, \_hWnd,(HMENU)p\_UpB\_ID,\_hInst,NULL);

p\_DownB = CreateWindow(L"button",L"V", WS\_CHILD|WS\_VISIBLE|WS\_CLIPSIBLINGS|BS\_PUSHBUTTON, 552, maxY - 23, 14, 11, \_hWnd,(HMENU)p\_DownB\_ID,\_hInst,NULL);

Fn\_TB = CreateWindow(TRACKBAR\_CLASS, NULL, WS\_CHILD|WS\_VISIBLE|TBS\_HORZ|TBS\_TOOLTIPS, 335, maxY - 78, 170, 30, \_hWnd,(HMENU)Fn\_TB\_ID, \_hInst, NULL);

SendMessage(Fn\_TB, TBM\_SETRANGE, (WPARAM) TRUE, (LPARAM) MAKELONG(1, 5000)); *// min. & max. positions*

SendMessage(Fn\_TB, TBM\_SETPAGESIZE, 0, (LPARAM) 1); *// скорость перетаскивания бегунка*

SendMessage(Fn\_TB, TBM\_SETSEL,(WPARAM) FALSE,(LPARAM) MAKELONG(0, 10));

SendMessage(Fn\_TB, TBM\_SETPOS,(WPARAM) TRUE, (LPARAM) 40); *// положение бегунка*

Fn\_St = CreateWindow(L"static",L"Натяжение, H", WS\_CHILD|WS\_VISIBLE|SS\_CENTER, 345, maxY - 94, 70,15, \_hWnd,(HMENU)0,\_hInst,NULL);

Fn\_Ed = CreateWindow(L"edit", L"40", WS\_CHILD|WS\_VISIBLE|WS\_CLIPSIBLINGS|WS\_BORDER|ES\_LEFT|ES\_AUTOHSCROLL, 510, maxY - 78, 40, 20, \_hWnd,(HMENU)Fn\_Ed\_ID, \_hInst, NULL);

Fn\_UpB = CreateWindow(L"button",L"Ʌ", WS\_CHILD|WS\_VISIBLE|WS\_CLIPSIBLINGS|BS\_PUSHBUTTON, 552, maxY - 78, 14, 11, \_hWnd,(HMENU)Fn\_UpB\_ID,\_hInst,NULL);

Fn\_DownB = CreateWindow(L"button",L"V", WS\_CHILD|WS\_VISIBLE|WS\_CLIPSIBLINGS|BS\_PUSHBUTTON, 552, maxY - 68, 14, 11, \_hWnd,(HMENU)Fn\_DownB\_ID,\_hInst,NULL);

Ao\_TB = CreateWindow(TRACKBAR\_CLASS, NULL, WS\_CHILD|WS\_VISIBLE|TBS\_HORZ|TBS\_TOOLTIPS, 20, maxY - 78, 240, 30, \_hWnd,(HMENU)Ao\_TB\_ID, \_hInst, NULL);

SendMessage(Ao\_TB, TBM\_SETRANGE, (WPARAM) TRUE, (LPARAM) MAKELONG(1, 100)); *// min. & max. positions*

SendMessage(Ao\_TB, TBM\_SETPAGESIZE, 0, (LPARAM) 1); *// скорость перетаскивания бегунка*

SendMessage(Ao\_TB, TBM\_SETSEL,(WPARAM) FALSE,(LPARAM) MAKELONG(0, 10));

SendMessage(Ao\_TB, TBM\_SETPOS,(WPARAM) TRUE, (LPARAM) 100); *// положение бегунка*

Ao\_St = CreateWindow(L"static",L"Амплитуда, 10 м", WS\_CHILD|WS\_VISIBLE|SS\_CENTER, 30, maxY - 94, 100,15, \_hWnd,(HMENU)0,\_hInst,NULL);

Ao\_St1 = CreateWindow(L"static",L"-3", WS\_CHILD|WS\_VISIBLE|SS\_CENTER, 108, maxY - 98, 10,15, \_hWnd,(HMENU)0,\_hInst,NULL);

Ao\_Ed = CreateWindow(L"edit", L"100", WS\_CHILD|WS\_VISIBLE|WS\_CLIPSIBLINGS|WS\_BORDER|ES\_LEFT|ES\_AUTOHSCROLL, 262, maxY - 78, 50, 20, \_hWnd,(HMENU)Ao\_Ed\_ID, \_hInst, NULL);

Ao\_UpB = CreateWindow(L"button",L"Ʌ", WS\_CHILD|WS\_VISIBLE|WS\_CLIPSIBLINGS|BS\_PUSHBUTTON, 314, maxY - 78, 14, 11, \_hWnd,(HMENU)Ao\_UpB\_ID,\_hInst,NULL);

Ao\_DownB = CreateWindow(L"button",L"V", WS\_CHILD|WS\_VISIBLE|WS\_CLIPSIBLINGS|BS\_PUSHBUTTON, 314, maxY - 68, 14, 11, \_hWnd,(HMENU)Ao\_DownB\_ID,\_hInst,NULL);

Font\_to\_Ctrl\_StringParam();

}

void tagCONTROL::CreateTimeSpeedPanel()

{

time\_St = CreateWindow(L"static",L"Время: %", WS\_CHILD|WS\_VISIBLE|SS\_CENTER, 587, maxY - 48, 70,15, \_hWnd,(HMENU)0,\_hInst,NULL);

time\_St2 = CreateWindow(L"static",L"1", WS\_CHILD|WS\_VISIBLE|SS\_CENTER, 627, maxY - 48, 20,15, \_hWnd,(HMENU)0,\_hInst,NULL);

time\_TB = CreateWindow(TRACKBAR\_CLASS, NULL, WS\_CHILD|WS\_VISIBLE|TBS\_HORZ|TBS\_TOOLTIPS, 577, maxY - 33, 160, 30, \_hWnd,(HMENU)time\_TB\_ID, \_hInst, NULL);

SendMessage(time\_TB, TBM\_SETRANGE, (WPARAM) TRUE, (LPARAM) MAKELONG(0, 100)); *// min. & max. positions*

SendMessage(time\_TB, TBM\_SETPAGESIZE, 0, (LPARAM) 10); *// скорость перетаскивания бегунка*

SendMessage(time\_TB, TBM\_SETSEL, (WPARAM) FALSE, (LPARAM) MAKELONG(0, 10));

SendMessage(time\_TB, TBM\_SETPOS, (WPARAM) TRUE, (LPARAM) 1); *// положение бегунка*

Font\_to\_TimeTB();

}

void tagCONTROL::CreateVolumeButton()

{

vol\_St = CreateWindow(L"static",L"Звук", WS\_CHILD|WS\_VISIBLE|SS\_CENTER, 610, maxY - 75, 30, 15, \_hWnd,(HMENU)0,\_hInst,NULL);

vol\_CB = CreateWindow(L"button",NULL, WS\_CHILD|WS\_VISIBLE|WS\_CLIPSIBLINGS|BS\_AUTOCHECKBOX, 650, maxY - 75, 15, 15,\_hWnd,(HMENU)vol\_CB\_ID,\_hInst, NULL);

Font\_to\_volTB();

}

void tagCONTROL::MoveAllWnd()

{

RECT rt;

GetClientRect (\_hWnd, &rt);

maxX = rt.right, maxY = rt.bottom;

float kx = (float)maxX / (float)FmaxX;

kx = kx\*sqrt(sqrt(kx));

MoveWindow(ButtonStop, maxX - 105, maxY - 110, 100, 40, true);

MoveWindow(ButtonStandart, maxX - 105, maxY - 55, 100, 40, true);

MoveWindow(OGL, 0, 0,maxX, maxY - 130, true);

MoveWindow(L\_TB, 20, maxY - 33, 240\*kx, 30, true);

MoveWindow(L\_St, 25, maxY - 48, 75,15, true);

MoveWindow(L\_St1, 78, maxY - 50, 10,10, true);

MoveWindow(L\_Ed, 262\*kx, maxY - 33, 50\*kx, 20, true);

MoveWindow(L\_UpB, 314\*kx, maxY - 33, 14, 11, true);

MoveWindow(L\_DownB, 314\*kx, maxY - 23, 14, 11, true);

MoveWindow(Ao\_TB, 20, maxY - 78, 240\*kx, 30, true);

MoveWindow(Ao\_St, 30, maxY - 94, 100,15, true);

MoveWindow(Ao\_St1, 108, maxY - 98, 10,15, true);

MoveWindow(Ao\_Ed, 262\*kx, maxY - 78, 50\*kx, 20, true);

MoveWindow(Ao\_UpB, 314\*kx, maxY - 78, 14, 11, true);

MoveWindow(Ao\_DownB, 314\*kx, maxY - 68, 14, 11, true);

MoveWindow(Fn\_TB, 335\*kx, maxY - 78, 170\*kx, 30,true);

MoveWindow(Fn\_St, 345\*kx, maxY - 94, 70,15, true);

MoveWindow(Fn\_Ed, 510\*kx, maxY - 78, 40\*kx, 20, true);

MoveWindow(Fn\_UpB, 552\*kx, maxY - 78, 14, 11, true);

MoveWindow(Fn\_DownB, 552\*kx, maxY - 68, 14, 11, true);

MoveWindow(p\_TB, 335\*kx, maxY - 33, 170\*kx, 30, true);

MoveWindow(p\_St, 345\*kx, maxY - 48, 113,15, true);

MoveWindow(p\_St1, 345\*kx+74, maxY - 50, 10,10, true);

MoveWindow(p\_Ed, 510\*kx, maxY - 33, 40\*kx, 20, true);

MoveWindow(p\_UpB, 552\*kx, maxY - 33, 14, 11, true);

MoveWindow(p\_DownB, 552\*kx, maxY - 23, 14, 11, true);

MoveWindow(time\_St, 587\*kx, maxY - 48, 70,15, true);

MoveWindow(time\_St2, 587\*kx+40, maxY - 48, 20,15, true);

MoveWindow(time\_TB, 577\*kx, maxY - 33, 160, 30, true);

MoveWindow(vol\_St, maxX - 234, maxY - 75, 30, 15, true);

MoveWindow(vol\_CB, maxX - 194, maxY - 75, 15, 15, true);

}

void tagCONTROL::CreateInterfacePanel( \_\_in WNDPROC WndProc1)

{

CreateButtons();

CreateOGLWindow( WndProc1);

CreateStringParamOption();

CreateTimeSpeedPanel();

CreateVolumeButton();

}

LRESULT CALLBACK tagCONTROL::ReactionAllButtonsAndEdits(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

int wmId = LOWORD(wParam);

int wmEvent = HIWORD(wParam);

switch(wmId)

{

case ButtonStop\_ID:

\_message\_typename = 1;

break;

case ButtonStandart\_ID:

this->Standart();

\_message\_typename = 2;

break;

case L\_Ed\_ID:

if(this->FirstEditsIndex[0] < 2)

{

this->FirstEditsIndex[0]++;

}

else

{

GetWindowText(L\_Ed, lpch, 10);

\_message\_typename = 3;

\_message\_value = \_wtoi(lpch);

if(\_message\_value <= 10)

{

\_message\_value = 10;

\_itow\_s(10, lpch, 10);

SetWindowText(L\_Ed, lpch);

}

else

if(\_message\_value > 1000)

{

\_message\_value = 1000;

\_itow\_s(1000, lpch, 10);

SetWindowText(L\_Ed, lpch);

}

SendMessage(L\_TB, TBM\_SETPOS,(WPARAM) TRUE,(LPARAM) \_message\_value);

}

break;

case L\_UpB\_ID:

\_message\_typename = 4;

GetWindowText(L\_Ed, lpch, 10);

par = \_wtoi(lpch) + 1;

\_itow\_s(par, lpch, 10);

SetWindowText(L\_Ed, lpch);

break;

case L\_DownB\_ID:

\_message\_typename = 5;

GetWindowText(L\_Ed, lpch, 10);

par = \_wtoi(lpch) - 1;

\_itow\_s(par, lpch, 10);

SetWindowText(L\_Ed, lpch);

break;

case p\_Ed\_ID:

if(this->FirstEditsIndex[2] < 2)

{

this->FirstEditsIndex[2]++;

}

else

{

GetWindowText(p\_Ed, lpch, 10);

\_message\_typename = 9;

\_message\_value = \_wtoi(lpch);

if(\_message\_value <= 0)

{

\_message\_value = 1;

\_itow\_s(1, lpch, 10);

SetWindowText(p\_Ed, lpch);

}

else

if(\_message\_value > 50)

{

\_message\_value = 50;

\_itow\_s(50, lpch, 10);

SetWindowText(p\_Ed, lpch);

}

SendMessage(p\_TB, TBM\_SETPOS,(WPARAM) TRUE,(LPARAM) \_message\_value);

}

break;

case p\_UpB\_ID:

\_message\_typename = 10;

GetWindowText(p\_Ed, lpch, 10);

par = \_wtoi(lpch) + 1;

\_itow\_s(par, lpch, 10);

SetWindowText(p\_Ed, lpch);

break;

case p\_DownB\_ID:

\_message\_typename = 11;

GetWindowText(p\_Ed, lpch, 10);

par = \_wtoi(lpch) - 1;

\_itow\_s(par, lpch, 10);

SetWindowText(p\_Ed, lpch);

break;

case Fn\_Ed\_ID:

if(this->FirstEditsIndex[3] < 2)

{

this->FirstEditsIndex[3]++;

}

else

{

GetWindowText(Fn\_Ed, lpch, 10);

\_message\_typename = 12;

\_message\_value = \_wtoi(lpch);

if(\_message\_value <= 0)

{

\_message\_value = 1;

\_itow\_s(1, lpch, 10);

SetWindowText(Fn\_Ed, lpch);

}

else

if(\_message\_value > 5000)

{

\_message\_value = 5000;

\_itow\_s(5000, lpch, 10);

SetWindowText(Fn\_Ed, lpch);

}

SendMessage(Fn\_TB, TBM\_SETPOS,(WPARAM) TRUE,(LPARAM) \_message\_value);

}

break;

case Fn\_UpB\_ID:

\_message\_typename = 13;

GetWindowText(Fn\_Ed, lpch, 10);

par = \_wtoi(lpch) + 1;

\_itow\_s(par, lpch, 10);

SetWindowText(Fn\_Ed, lpch);

break;

case Fn\_DownB\_ID:

\_message\_typename = 14;

GetWindowText(Fn\_Ed, lpch, 10);

par = \_wtoi(lpch) - 1;

\_itow\_s(par, lpch, 10);

SetWindowText(Fn\_Ed, lpch);

break;

case Ao\_Ed\_ID:

if(this->FirstEditsIndex[4] < 2)

{

this->FirstEditsIndex[4]++;

}

else

{

GetWindowText(Ao\_Ed, lpch, 10);

\_message\_typename = 15;

\_message\_value = \_wtoi(lpch);

if(\_message\_value <= 0)

{

\_message\_value = 1;

\_itow\_s(1, lpch, 10);

SetWindowText(Ao\_Ed, lpch);

}

else

if(\_message\_value > 100)

{

\_message\_value = 100;

\_itow\_s(100, lpch, 10);

SetWindowText(Ao\_Ed, lpch);

}

SendMessage(Ao\_TB, TBM\_SETPOS,(WPARAM) TRUE,(LPARAM) \_message\_value);

}

break;

case Ao\_UpB\_ID:

\_message\_typename = 16;

GetWindowText(Ao\_Ed, lpch, 10);

par = \_wtoi(lpch) + 1;

\_itow\_s(par, lpch, 10);

SetWindowText(Ao\_Ed, lpch);

break;

case Ao\_DownB\_ID:

\_message\_typename = 17;

GetWindowText(Ao\_Ed, lpch, 10);

par = \_wtoi(lpch) - 1;

\_itow\_s(par, lpch, 10);

SetWindowText(Ao\_Ed, lpch);

break;

case vol\_CB\_ID:

\_message\_typename = 18;

default:break;

}

return DefWindowProc(hWnd, message, wParam, lParam);

}

LRESULT CALLBACK tagCONTROL::ReactionAllTrackBars(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

if((HWND)lParam == GetDlgItem (hWnd, L\_TB\_ID))

{

par = SendMessage(L\_TB, TBM\_GETPOS, 0, 0l);

\_message\_typename = 3;

\_message\_value = par;

\_itow\_s(par, lpch, 10);

SetWindowText(L\_Ed, lpch);

}

else

if((HWND)lParam == GetDlgItem (hWnd, p\_TB\_ID))

{

par = SendMessage(p\_TB, TBM\_GETPOS, 0, 0l);

\_message\_typename = 9;

\_message\_value = par;

\_itow\_s(par, lpch, 10);

SetWindowText(p\_Ed, lpch);

}

else

if((HWND)lParam == GetDlgItem (hWnd, Fn\_TB\_ID))

{

par = SendMessage(Fn\_TB, TBM\_GETPOS, 0, 0l);

\_message\_typename = 12;

\_message\_value = par;

\_itow\_s(par, lpch, 10);

SetWindowText(Fn\_Ed, lpch);

}

else

if((HWND)lParam == GetDlgItem (hWnd, Ao\_TB\_ID))

{

par = SendMessage(Ao\_TB, TBM\_GETPOS, 0, 0l);

\_message\_typename = 15;

\_message\_value = par;

\_itow\_s(par, lpch, 10);

SetWindowText(Ao\_Ed, lpch);

}

else

if((HWND)lParam == GetDlgItem (hWnd, time\_TB\_ID))

{

\_message\_typename = 19;

\_message\_value = SendMessage(time\_TB, TBM\_GETPOS, 0, 0l);

\_itow\_s(\_message\_value, lpch, 10);

SetWindowText(time\_St2, lpch);

}

else

return DefWindowProc(hWnd, message, wParam, lParam);

return 0;

}

HWND tagCONTROL::GetOGLHWND()

{

return OGL;

}

void tagCONTROL::Standart()

{

\_itow\_s(1000,lpch,10);

SetWindowText(L\_Ed,lpch);

\_itow\_s(40,lpch,10);

SetWindowText(Fn\_Ed,lpch);

\_itow\_s(150,lpch,10);

SetWindowText(p\_Ed,lpch);

\_itow\_s(100,lpch,10);

SetWindowText(Ao\_Ed,lpch);

SendMessage(time\_TB,TBM\_SETPOS,(WPARAM)1,0l);

SetWindowText(time\_St2,L"1");

}

# endif

**<Grafics.h>**

# ifndef \_GRAFICS\_H

# define \_GRAFICS\_H

# include "main.h"

# ifdef \_MAIN\_H

# include "font.h"

# include <Windows.h>

typedef class tagGrafic

{

float MassX [32768];

float MassY1[32768];

float MassY2[32768];

int MassSize;

*//*

RECT Rect;

HDC hDC;

HGLRC hRC;

HINSTANCE hInst;

wchar\_t WindowsName[30];

void InitializeOpenGL ();

void Init();

void Render();

CFont \*Font;

wchar\_t Buffer [ 256 ];

public:

int correctX;

int correctY;

GLfloat GcorrectX;

GLfloat GcorrectY;

bool Open;

HWND hWnd;

void Push(float X, float Y\_1, float Y\_2);

void Reset();

bool GetEnd();

tagGrafic(wchar\_t \*WindowsName, HINSTANCE hInst);

void ReInit();

void Deinit();

void OpenMsg(WNDPROC WndProc);

}GRAFIC, \*LPGRAFIC;

# endif

# endif

**<Grafics.cpp>**

# include "stdafx.h"

# include "Grafics.h"

# ifdef \_GRAFICS\_H

tagGrafic::tagGrafic(wchar\_t \*WindowsName, HINSTANCE hInst)

{

wcscpy\_s(this->WindowsName, WindowsName);

this->hInst = hInst;

Open = false;

*//*

correctX = 5;

correctY = 5;

GcorrectX = 1.0f;

GcorrectY = 1.0f;

}

void tagGrafic::ReInit()

{

wglMakeCurrent ( hDC, hRC );

Render();

wglMakeCurrent ( g\_hDC, g\_hRC );

}

bool tagGrafic::GetEnd()

{

if(MassSize >= 32768)

return true;

return false;

}

void tagGrafic::Deinit()

{

if (hRC)

{

wglMakeCurrent ( NULL, NULL );

wglDeleteContext (hRC);

}

if (hDC)

ReleaseDC (hWnd, hDC);

}

void tagGrafic::Init ()

{

*//GetClientRect ( hWnd, &Rect );*

InitializeOpenGL ();*//int width, int heigth*

hRC = wglCreateContext (hDC);

}

void tagGrafic::InitializeOpenGL ()*//int width, int heigth*

{

hDC = GetDC ( hWnd );

if ( !bSetupPixelFormat ( hDC ) )

CloseWindow (hWnd);

hRC = wglCreateContext (hDC);

wglMakeCurrent ( hDC, hRC );

glEnable ( GL\_DEPTH\_TEST );

*//glViewport ( 0.0f, 0.0f, 700.0f, 440.0f );*

glMatrixMode ( GL\_PROJECTION );

glLoadIdentity ( );

gluOrtho2D ( 0.0f, 800.0f, 0.0f, 440.0f );

glDepthRange ( 0, 1 );

glMatrixMode ( GL\_MODELVIEW );

glLoadIdentity ( );

Font = new CFont ( "framd.ttf", 10, 10 );

}

void tagGrafic::OpenMsg(WNDPROC WndProc)

{

WNDCLASS wndclass;

memset ( &wndclass, 0, sizeof ( WNDCLASS ) );

wndclass.style = CS\_HREDRAW | CS\_VREDRAW;

wndclass.lpfnWndProc = WndProc;

wndclass.hIcon = LoadIcon ( hInst, MAKEINTRESOURCE(108) );

wndclass.hCursor = LoadCursor ( NULL, IDC\_ARROW );

wndclass.hbrBackground = ( HBRUSH ) ( COLOR\_WINDOW + 1 );

wndclass.lpszClassName = L"MyOpenWindow";

RegisterClass ( &wndclass );

hWnd = CreateWindow(L"MyOpenWindow", this->WindowsName, WS\_OVERLAPPED|WS\_CAPTION|WS\_SYSMENU, 200, 200, 800, 440, NULL,(HMENU)0, hInst, 0l);

if (!hWnd)return;

*//SetWindowLong(hWnd,GWL\_WNDPROC, WndProc);*

Init();

ShowWindow ( hWnd, SW\_SHOWNORMAL );

Open = true;

}

void tagGrafic::Render()

{

glClearColor ( 0.95f, 0.95f, 0.95f, 1 );

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glMatrixMode ( GL\_MODELVIEW );

glPushMatrix ( );

glLoadIdentity();

glOrtho ( 0, 800, 0, 440, 0, 1 );

#pragma region Сетка

*//сетка и цена деления*

glColor3ub ( 70, 70, 70 );

*//сетка*

for ( int i = 50; i <= 750; i += 5 \* correctX )

{

glBegin ( GL\_LINES );

glVertex2i ( i, 50 ); *//Y*

glVertex2i ( i, 400 );

glEnd ( );

}

for ( int i = 50; i <= 750; i += 5 \* correctY )

{

glBegin ( GL\_LINES );

if ( i >= 50 && i <= 400 )

{

glVertex2i ( 50, i ); *//X*

glVertex2i ( 750, i );

}

glEnd ( );

}

*//*

glBegin ( GL\_LINES );

*// линия оси OY*

glVertex2i ( 50, 50 );

glVertex2i ( 50, 410 );

*// линия оси OX*

glVertex2i ( 50, 50 );

glVertex2i ( 760, 50 );

glEnd ( );

glBegin ( GL\_TRIANGLES );

*// стрелочка оси OY*

glVertex2i ( 45, 410 );

glVertex2i ( 55, 410 );

glVertex2i ( 50, 420 );

*//*

*// стрелочка оси OX*

glVertex2i ( 760, 55 );

glVertex2i ( 760, 45 );

glVertex2i ( 770, 50 );

*//*

glEnd ( );

glColor3f ( 0.95f, 0.95f, 0.95f );

glBegin ( GL\_QUADS );

glVertex2i ( 50, 401 ); glVertex2i ( 800, 401 );

glVertex2i ( 800, 440 ); glVertex2i ( 50, 440 );

glEnd ( );

#pragma endregion

#pragma region отрисовка графика

glPushMatrix();

glColor3ub ( 200, 50, 50 );

glBegin ( GL\_QUADS );

glVertex2i ( 50, 15 ); glVertex2i ( 70, 15 );

glVertex2i ( 70, 25 ); glVertex2i ( 50, 25 );

glEnd ( );

glColor3ub ( 50, 200, 50 );

glBegin ( GL\_QUADS );

glVertex2i ( 320, 15 ); glVertex2i ( 340, 15 );

glVertex2i ( 340, 25 ); glVertex2i ( 320, 25 );

glEnd ( );

glTranslatef ( 50.0f, 50.0f, 0.0f );

glEnable ( GL\_LINE );

glLineWidth ( 2.0f );

glBegin ( GL\_LINE\_STRIP );

glColor3ub ( 200, 50, 50 );

int k = 0, p = 0;

for ( int u = 0; u < MassSize; u++ )

{

if ( MassX [ u ] \* GcorrectX <= 700 )

glVertex2f ( MassX [ u ] \* GcorrectX, MassY1 [ u ] \* GcorrectY );

k = u;

}

glEnd ( );

glBegin ( GL\_LINE\_STRIP );

glColor3ub ( 50, 200, 50 );

for ( int u = 0; u < MassSize; u++ )

{

if ( MassX [ u ] \* GcorrectX <= 700 )

glVertex2f ( MassX [ u ] \* GcorrectX, MassY2 [ u ] \* GcorrectY );

p = u;

}

glEnd ( );

glLineWidth ( 1.0f );

glDisable ( GL\_LINE );

glPopMatrix();

glPopMatrix ( );

#pragma endregion

glMatrixMode ( GL\_PROJECTION );

glPushMatrix ( );

glLoadIdentity ( );

glOrtho ( 0, 800, 0, 440, 0, 1 );

glMatrixMode ( GL\_MODELVIEW );

glLoadIdentity ( );

#pragma region text

glDisable ( GL\_DEPTH\_TEST );

glColor3f ( 0.0f, 0.0f, 0.0f );

swprintf ( Buffer, sizeof ( Buffer ), L"Время: %.3f, с", MassX [ k ] );

Font -> Font -> FaceSize ( 13 );

Font -> Print ( 640, 410, Buffer );

glEnable ( GL\_DEPTH\_TEST );

glDisable ( GL\_DEPTH\_TEST );

glColor3f ( 0.0f, 0.0f, 0.0f );

swprintf ( Buffer, sizeof ( Buffer ), L"Потенциальная энергия: %.2f, Дж", MassY1 [ k ] );

Font -> Font -> FaceSize ( 13 );

Font -> Print ( 80, 15, Buffer );

glEnable ( GL\_DEPTH\_TEST );

glDisable ( GL\_DEPTH\_TEST );

glColor3f ( 0.0f, 0.0f, 0.0f );

swprintf ( Buffer, sizeof ( Buffer ), L"Кинетическая энергия: %.2f, Дж", MassY2 [ p ] );

Font -> Font -> FaceSize ( 13 );

Font -> Print ( 350, 15, Buffer );

glEnable ( GL\_DEPTH\_TEST );

glDisable ( GL\_DEPTH\_TEST );

glColor3f ( 0.0f, 0.0f, 0.0f );

swprintf ( Buffer, sizeof ( Buffer ), L"t [с]" );

Font -> Font -> FaceSize ( 12 );

Font -> Print ( 760, 30, Buffer );

glEnable ( GL\_DEPTH\_TEST );

glDisable ( GL\_DEPTH\_TEST );

glColor3f ( 0.0f, 0.0f, 0.0f );

swprintf ( Buffer, sizeof ( Buffer ), L"E [Дж]" );

Font -> Font -> FaceSize ( 12 );

Font -> Print ( 12, 410, Buffer );

glEnable ( GL\_DEPTH\_TEST );

#pragma endregion

#pragma region Деления

*// цена деления оси OX*

for ( GLfloat i = 0.0f, iv = 50; iv <= 725; iv += 0 )

{

glDisable ( GL\_DEPTH\_TEST );

glColor3f ( 0.0f, 0.0f, 0.0f );

swprintf ( Buffer, sizeof ( Buffer ), L"%.3f", i );

Font -> Font -> FaceSize ( 10 );

Font -> Print ( iv, 35, Buffer );

glEnable ( GL\_DEPTH\_TEST );

if ( correctX == 5 )

{

GcorrectX = 710.0f \* 8.0f;

iv += 5 \* correctX \* 5;

i += 0.0041661875f;

}

if ( correctX == 4 )

{

GcorrectX = 710.0f \* 6.0f;

iv += 5 \* correctX \* 5;

i += 0.008332375f ;

}

if ( correctX == 3 )

{

GcorrectX = 710.0f \* 4.0f;

iv += ( 5 \* correctX ) \* 5;

i += 0.01666475f;

}

if ( correctX == 2 )

{

GcorrectX = 710.0f \* 2.0f;

iv += ( 5 \* correctX ) \* 5;

i += 0.0333295f;

}

if ( correctX == 1 )

{

GcorrectX = 710.0f \* 1.0f;

iv += ( 5 \* correctX ) \* 9;

i += 0.066659f;

}

}

*// цена деления оси OY*

int ic = 0;

int ivc = 0;

if ( correctY == 5 )

{

GcorrectY = 2.5f;

ic = 10;

ivc = 50 + 5 \* correctY;

}

if ( correctY == 4 )

{

GcorrectY = 0.8f;

ic = 50;

ivc = 50 + 5 \* correctY \* 2;

}

if ( correctY == 3 )

{

GcorrectY = 0.3017241379310345f;

ic = 100;

ivc = 50 + 5 \* correctY \* 2;

}

if ( correctY == 2 )

{

GcorrectY = 0.02f;

ic = 1000;

ivc = 50 + 5 \* correctY \* 2;

}

if ( correctY == 1 )

{

GcorrectY = 0.01f;

ic = 5000;

ivc = 50 + 5 \* correctY \* 10;

}

for ( int i = ic, iv = ivc; iv <= 395; iv += 0 )

{

glDisable ( GL\_DEPTH\_TEST );

glColor3f ( 0.0f, 0.0f, 0.0f );

swprintf ( Buffer, sizeof ( Buffer ), L"%d", i );

Font -> Font -> FaceSize ( 10 );

if ( i < 999 )

Font -> Print ( 30, iv, Buffer );

else

if ( i > 99999 )

Font -> Print ( 11, iv, Buffer );

else

if ( i > 9999 )

Font -> Print ( 18, iv, Buffer );

else

if ( i > 999 )

Font -> Print ( 23, iv, Buffer );

glEnable ( GL\_DEPTH\_TEST );

if ( correctY == 5 )

{

iv += 5 \* correctY;

i += 10;

}

if ( correctY == 4 )

{

iv += 5 \* correctY \* 2;

i += 50;

}

if ( correctY == 3 )

{

iv += 5 \* correctY \* 2;

i += 100;

}

if ( correctY == 2 )

{

iv += ( 5 \* correctY ) \* 2;

i += 1000;

}

if ( correctY == 1 )

{

iv += ( 5 \* correctY ) \* 10;

i += 5000;

}

}

#pragma endregion

glMatrixMode ( GL\_PROJECTION );

glPopMatrix ( );

glMatrixMode ( GL\_MODELVIEW );

glFlush();

SwapBuffers (hDC);

}

void tagGrafic::Push(float X, float Y\_1, float Y\_2)

{

this->MassX[MassSize] = X;

this->MassY1[MassSize] = Y\_1;

this->MassY2[MassSize] = Y\_2;

this->MassSize ++;

}

void tagGrafic::Reset()

{

this->MassSize = 0;

}

# endif

**<StringPhusics.h>**

#ifndef \_STRINGPHYSICS\_H

#define \_STRINGPHYSICS\_H

# include <stdio.h>

# define \_USE\_MATH\_DEFINES

# include <math.h>

# include <Windows.h>

# define COUNT 256*//количество элементов на струне.*

typedef class tagInData

{

public:

double l;*//длина струны*

double Fn;*//сила натяжения струны*

double p;*//плотность струны*

double Ao;*//первоначальное отклонение струны*

public:

tagInData( double \_l, double \_Fn, double \_p, double \_Ao) :

l(\_l), Fn(\_Fn), p(\_p), Ao(\_Ao) {};

tagInData(){};

}INDATA, \*LPINDATA;

typedef class tagStringParam : public INDATA

{

public:

float x[COUNT]; *//*

float y[COUNT]; *//текущие координаты точек*

float w;*//циклическая частота*

float v\_nyu;*//частота(Гц)*

float T;*//период*

float t;*//текущее время*

float Ek;*// кинет. энергия*

float Ep;*// потенциальная.*

bool Go;

int timespeed; *// скорость течения времени*

private:

float pw;*//Fn/ma*

float step;*//*

int i; *// индекс??*

*//доп. переменные для оптимизации*

*//*

float Ao\_2;

float sin\_wt;

float sin\_wt\_2;

float pi\_l;

float cos\_wt\_2;

float par1, par2;

*//*

void Standart()*//задание стандартного набора параметров для струны(для тестов)*

{

*//Standart Data*

l = 1000; *//mm*

Fn = 40; *//H*

p = 1; *//gramm/mm*

t = 0;

Ao = 100; *//mm*

timespeed = 1;

*//*

};

void Calc\_param()*//расчет всяких штук*

{

v\_nyu = 1000\*sqrt(Fn/p)/l;

T=1/v\_nyu;

w = M\_PI \* v\_nyu;

step = l/COUNT;

pw = Fn/(p\*l\*step\*1e-6);*//!!---*

Fill\_x();

Ao\_2 = 2 \* Ao;

pi\_l = M\_PI/l;

par1 = ( M\_PI - 1) \* M\_PI \* Fn \* Ao \* Ao /( l \* 1000) ;

par2 = ( 1 - M\_1\_PI) \* w \* w \* l \* Ao \* Ao \* 1E-9 \* p;

};

void Fill\_x() *// просто расстановка точек по длине. каждой точке ставит ее положение на струне*

{

x[COUNT - 1] = l - step;

for(int c = COUNT - 2; c >= 0; c--)

x[c] = c\*step;

};

public:

tagStringParam(){Go=true;};

void Create\_First\_String\_Pos()

{

Standart();

Calc\_param();

Fill\_x();

for(int i = COUNT - 1; i >= 0; i--)

y[i] = Ao\_2\*sin(pi\_l\*x[i]) \* sin(w\*t);

};

void WaveProcess()

{

t+=0.0002;

sin\_wt = sin(w\*t);

*//*

sin\_wt\_2 = sin\_wt \* sin\_wt;

cos\_wt\_2 = cos(w\*t); cos\_wt\_2 \*= cos\_wt\_2;

Ek = par2 \* cos\_wt\_2; Ep = par1 \* sin\_wt\_2;

*//*

for(int i = COUNT - 1; i >= 0; i--)

y[i] = Ao\_2 \* sin(pi\_l\*x[i]) \* sin\_wt;

};

void Ctrl\_Reaction(int \_value, int \_typename)

{

switch(\_typename)

{

case 0*/\*null\*/*:

return;

case 1*/\*But\_Stop\*/*:

Go = !Go;

break;

case 2*/\*But\_Std\*/*:

Standart();

break;

case 3*/\*L\*/*:

l = \_value;

Fill\_x();

break;

case 4*/\*L\_UP\*/*:

l += 1;

Fill\_x();

break;

case 5*/\*L\_DOWN\*/*:

l -= 1;

Fill\_x();

break;

case 9*/\*p\*/*:

p = (float)\_value/10;

break;

case 10*/\*p\_UP\*/*:

p += 1;

break;

case 11*/\*p\_DOWN\*/*:

p -= 1;

break;

case 12*/\*Fn\*/*:

Fn = \_value;

break;

case 13*/\*Fn\_UP\*/*:

Fn += 1;

break;

case 14*/\*Fn\_DOWN\*/*:

Fn -= 1;

break;

case 15*/\*Ao\*/*:

Ao = \_value;

break;

case 16*/\*Ao\_UP\*/*:

Ao += 1;

break;

case 17*/\*Ao\_DOWN\*/*:

Ao -= 1;

break;

case 19:

timespeed = \_value;

default:

break;

}

this->Calc\_param();

};

}STRINGPARAM, \*LPSTRINGPARAM;

#endif

**<init.cpp>**

# include "stdafx.h"

# include "main.h"

# ifdef \_MAIN\_H

bool bSetupPixelFormat(HDC hdc)

{

PIXELFORMATDESCRIPTOR pfd; *// Дескриптор формата пиксела*

int pixelformat;

pfd.nSize = sizeof(PIXELFORMATDESCRIPTOR); *// Устанавливаем размер структуры*

pfd.nVersion = 1; *// Всегда ставим = 1*

*// Передаём нужные флаги OpenGL*

pfd.dwFlags = PFD\_DRAW\_TO\_WINDOW | PFD\_SUPPORT\_OPENGL | PFD\_DOUBLEBUFFER;

pfd.dwLayerMask = PFD\_MAIN\_PLANE; *// Стандартная маска (один хрен игнорируется)*

pfd.iPixelType = PFD\_TYPE\_RGBA; *// Нам нужны RGB and Alpha типа пикселей*

pfd.cColorBits = SCREEN\_DEPTH; *// Используем наши #define для цветовой глубины*

pfd.cDepthBits = SCREEN\_DEPTH; *// Это игнорируется для RGBA, но все равно передадим*

pfd.cAccumBits = 0;

pfd.cStencilBits = 0;

*// Ф-я ищет формат пиксела, наиболее подходящий заданным требованиям, выход при неудаче*

if ( (pixelformat = ChoosePixelFormat(hdc, &pfd)) == FALSE )

{

MessageBox(NULL, L"ChoosePixelFormat failed", L"Error", MB\_OK);

return FALSE;

}

*// Устанавливаем указанный формат пиксела*

if (SetPixelFormat(hdc, pixelformat, &pfd) == FALSE)

{

MessageBox(NULL, L"SetPixelFormat failed", L"Error", MB\_OK);

return FALSE;

}

return TRUE;

}

void SizeOpenGLScreen(int width, int height)

{

if (height==0) *// Предотвратим деление на 0*

height=1;

glViewport(0,0,width,height);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45.0f,(GLfloat)width/(GLfloat)height, .5f ,1000.0f);

glMatrixMode(GL\_MODELVIEW); *// Выберем матрицу моделей*

glLoadIdentity(); *// И сбросим её*

}

void InitializeOpenGL(int width, int height)

{

g\_hDC = GetDC(g\_hWnd); *// Устанавливаем глобальный дескриптор окна*

if (!bSetupPixelFormat(g\_hDC)) *// Устанавливаем формат пиксела*

PostQuitMessage (0); *// И выходим при ошибке*

g\_hRC = wglCreateContext(g\_hDC); *// Контекст рендеринга для hdc*

wglMakeCurrent(g\_hDC, g\_hRC); *// Делаем контекст текущим*

glEnable(GL\_TEXTURE\_2D); *// Включаем текстуры*

glEnable(GL\_DEPTH\_TEST); *// И тест глубины*

*// И устанавливаем размер вьюпорта:*

SizeOpenGLScreen(width, height);

}

void DeInit()

{

if (g\_hRC)

{

wglMakeCurrent(NULL, NULL); *// Освобождает память, занятую для рендера*

wglDeleteContext(g\_hRC); *// Удаляет контекст рендеринга OpenGL*

}

if (g\_hDC)

ReleaseDC(g\_hWnd, g\_hDC); *// Убирает HDC из памяти*

UnregisterClass(L" ", g\_hInstance); *// Освобождаем класс окна*

PostQuitMessage (0); *// Выходим*

}

# endif

**<main.cpp>**

# include "stdafx.h"

# include "SS.h"

# pragma comment (lib, "opengl32.lib")

# pragma comment (lib, "glu32.lib")

# pragma comment (lib, "glaux.lib")

# pragma comment (lib, "ftgl\_dynamic\_MT.lib")

# pragma comment (lib, "DevIL.lib")

# pragma comment (lib, "ilut.lib")

# pragma comment (lib, "ilu.lib")

#pragma comment(lib,"htmlhelp.lib")

#include <htmlhelp.h>

#pragma comment(linker,"/manifestdependency:\"type='win32' name='Microsoft.Windows.Common-Controls' version='6.0.0.0' processorArchitecture='\*' publicKeyToken='6595b64144ccf1df' language='\*'\"")

#pragma comment(lib,"comctl32.lib")

# include "main.h"

# include "camera.h"

# include "glelements\_and\_texture.h"

# include "sound.h"

# include "font.h"

# include "Grafics.h"

# include <cstdlib>

# if defined \_STRINGPHYSICS\_H && defined \_CONTROL\_H

# define kSpeed 10.0f

GLelements GLE;

bool SoundFlag = false; *// treu - звук включен; false - звук выключен*

Sound SD;

CTexture \*Texture;

CCamera g\_Camera;

CFont \*Font;

*// Необходимые дескрипторы:*

HWND g\_hWnd;

RECT g\_rRect;

HDC g\_hDC;

HGLRC g\_hRC;

HINSTANCE g\_hInstance;

*//залипание курсора*

bool flag = false;

#define MAX\_LOADSTRING 100

*// Глобальные переменные:*

HINSTANCE hInst; *// текущий экземпляр*

TCHAR szTitle[MAX\_LOADSTRING]; *// Текст строки заголовка*

TCHAR szWindowClass[MAX\_LOADSTRING]; *// имя класса главного окна*

HWND hWnd;

MSG msg;

HWND hWndLogo;

LPSTRINGPARAM String;

LPCONTROL cont;

LPGRAFIC Gr;

*//////////////WinApi & MainProgram///////////////*

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

LRESULT CALLBACK WinProc(HWND, UINT, WPARAM, LPARAM);

LRESULT CALLBACK WndProcGr(HWND, UINT, WPARAM, LPARAM);

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

void DrawBackground(HWND hWndBGR, int ID\_BGR, HINSTANCE hInst)

{

HDC dc, bits;

RECT rt;

GetClientRect(hWndBGR, &rt);

int maxX = rt.right, maxY = rt.bottom;

dc = GetDC(hWndBGR);

bits = CreateCompatibleDC(dc);

HBITMAP BGR = LoadBitmap(hInst, MAKEINTRESOURCE(ID\_BGR));

SelectObject(bits, BGR);

StretchBlt(dc, 0, 0, maxX, maxY, bits, 0, 0, maxX, maxY, SRCCOPY);

DeleteDC(bits);

ReleaseDC(hWndBGR, dc);

}

int APIENTRY \_tWinMain(HINSTANCE hInstance,

HINSTANCE hPrevInstance,

LPTSTR lpCmdLine,

int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

HACCEL hAccelTable;

LoadString(hInstance, IDC\_TEST, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

#pragma region заставка

WNDCLASS wc = {0};

wc.lpszClassName = L"logo";

wc.lpfnWndProc = DefWindowProc;

wc.hCursor = LoadCursor(NULL, IDC\_ARROW);

wc.hbrBackground = (HBRUSH)GetSysColorBrush(COLOR\_3DFACE);

wc.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_SMALL));

RegisterClass(&wc);

hWndLogo = CreateWindow(L"logo", NULL, WS\_POPUPWINDOW, GetSystemMetrics(SM\_CXSCREEN) / 2.0f - 295, GetSystemMetrics(SM\_CYSCREEN) / 2.0f - 200,

590, 400, NULL, NULL, hInst, 0);

*// отображение заставки.*

ShowWindow(hWndLogo, nCmdShow);

*// замер времени*

LARGE\_INTEGER b\_start,b\_stop,freq;

QueryPerformanceFrequency(&freq);

QueryPerformanceCounter(&b\_start);

*// Отображение курсора.*

ShowCursor (true);

if (!InitInstance (hInstance, nCmdShow))return FALSE;

hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_TEST));

*// картинка*

DrawBackground( hWndLogo, IDB\_LOGO, hInst);

#pragma endregion

#pragma region загрузка программы

*//*

int i = 0;

int speed = 100;

String = new STRINGPARAM();

Gr = new GRAFIC(L"Энергия", hInst);

cont = new CONTROL(hWnd, hInstance);

cont->CreateInterfacePanel(WinProc);

*//*

InitOpenGL(cont->GetOGLHWND());

#pragma endregion

#pragma region закрытие заставки (обработка сообщений)

while(1)

{

if ( PeekMessage ( &msg, NULL, 0, 0, PM\_REMOVE ) )

{

if ( msg.message == WM\_QUIT )

break;

*// выходит по нажатию клавиши(любой).*

if ( msg.message == WM\_KEYDOWN || msg.message == WM\_MBUTTONDOWN )

{

DestroyWindow(hWndLogo);

break;

}

TranslateMessage ( &msg );

DispatchMessage ( &msg );

}

*// выходит после 10 секунд*

QueryPerformanceCounter(&b\_stop);

b\_stop.QuadPart = (b\_stop.QuadPart - b\_start.QuadPart)/(freq.QuadPart);

if(b\_stop.QuadPart == 10)

{

DestroyWindow(hWndLogo);

break;

}

}

#pragma endregion

#pragma region отображение главного окна

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

#pragma endregion

return (int) MainLoop();

}

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEX wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_TEST));

wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW);

wcex.lpszMenuName = MAKEINTRESOURCE(IDC\_TEST);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassEx(&wcex);

}

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

hInst = hInstance; *// Сохранить дескриптор экземпляра в глобальной переменной*

hWnd = CreateWindow(szWindowClass, L"Sound String", WS\_OVERLAPPEDWINDOW,

120, 50, 860, 650, NULL, NULL, hInstance, NULL);

if (!hWnd)

return FALSE;

return TRUE;

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

int wmId, wmEvent;

LONG lRet = 0;

switch (message)

{

case WM\_GETMINMAXINFO: *//Получили сообщение от Винды*

{

MINMAXINFO \*pInfo = (MINMAXINFO \*)lParam;

POINT Min = { 860, 600 };

pInfo->ptMinTrackSize = Min; *// Установили минимальный размер*

return lRet;

}

case WM\_COMMAND:

wmId = LOWORD(wParam);

wmEvent = HIWORD(wParam);

*// Разобрать выбор в меню:*

switch (wmId)

{

case IDM\_ABOUT:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

case ID\_E\_m:

Gr->OpenMsg(WndProcGr);

break;

case ID\_HELP:

HtmlHelp( hWnd, L"help.chm::/Index.htm", HH\_DISPLAY\_TOPIC, NULL);

break;

default:

lRet = cont->ReactionAllButtonsAndEdits(hWnd, message, wParam, lParam);

if(cont->\_message\_typename)

{

String->Ctrl\_Reaction(cont->\_message\_value, cont->\_message\_typename);

Gr->Reset();

String->t = 0;

if(cont->\_message\_typename == 18)

SoundFlag = !SoundFlag;

}

break;

}

break;

case WM\_SIZE:

cont->MoveAllWnd();

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

case WM\_HSCROLL:

lRet = cont->ReactionAllTrackBars(hWnd, message, wParam, lParam);

if(cont->\_message\_typename == 19)

String->Ctrl\_Reaction(cont->\_message\_value, 19);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return lRet;

}

INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)

{

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

break;

}

return (INT\_PTR)FALSE;

}

LRESULT CALLBACK WinProc(HWND hWnd,UINT uMsg, WPARAM wParam, LPARAM lParam)

{

LONG lRet = 0;

switch (uMsg)

{

case WM\_SIZE: *// Если изменён размер окна*

SizeOpenGLScreen(LOWORD(lParam),HIWORD(lParam));*// LoWord=Width, HiWord=Height*

GetClientRect(hWnd, &g\_rRect); *// получаем window rectangle*

break;

case WM\_MOUSEACTIVATE:

flag = !flag;

break;

case WM\_KEYDOWN:

if(wParam==VK\_ESCAPE) *// Если нажат ESCAPE*

PostQuitMessage(0); *// Выходим*

break;

case WM\_CLOSE: *// Если окно было закрыто*

PostQuitMessage(0); *// Выходим*

break;

default: *// Return по умолчанию*

lRet = DefWindowProc (hWnd, uMsg, wParam, lParam);

break;

}

return lRet;

}

WPARAM MainLoop()

{

String->Create\_First\_String\_Pos();

int ind = 0;

while(1)

{

if (PeekMessage(&msg, NULL, 0, 0, PM\_REMOVE))

{

if(msg.message == WM\_QUIT)

break;

TranslateMessage(&msg);

DispatchMessage(&msg);

}

else

{

if(flag)

{

g\_Camera.SetViewByMouse ( );

g\_Camera.Update ( kSpeed );

}

if(String->Go)

for(ind = 0; ind < String->timespeed; ind ++)

{

String->WaveProcess();

if(!Gr->GetEnd())

Gr->Push(String->t, String->Ep, String->Ek);

else

{

String->t = 0;

Gr->Reset();

}

}

SD.set\_frequency ( String -> v\_nyu );

SD.SoundString ( SoundFlag );

RenderScene();

if(Gr->Open)

Gr->ReInit();

}

}

DeInit ( );

Gr->Deinit();

return(msg.wParam);

}

LRESULT CALLBACK WndProcGr(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)

{

switch(msg)

{

case WM\_DESTROY:

CloseWindow(hWnd);

if(hWnd == Gr->hWnd)

Gr->Open = false;

return 0;

case WM\_KEYDOWN:

switch(wParam)

{

case VK\_NUMPAD2:

if ( Gr->correctY > 1 ) Gr->correctY--;

break;

case VK\_NUMPAD4:

if ( Gr->correctX > 1 ) Gr->correctX--;

break;

case VK\_NUMPAD6:

if ( Gr->correctX < 5 ) Gr->correctX++;

break;

case VK\_NUMPAD8:

if ( Gr->correctY < 5 ) Gr->correctY++;

break;

default:

break;

}

break;

default:

break;

}

return DefWindowProc(hWnd, msg, wParam, lParam);

};

*///////////Grafics & Sound//////////*

void DrawCorrectLenghtString ( GLfloat l\_segment, int segment, GLfloat radius )

{

int y = 0;

for ( float i = 0.0f; i < l\_segment \* segment; i+= l\_segment, y++ )

GLE.CreateOctagon ( i, String -> y [ y ] \* 0.00825f, - 0.1f, l\_segment, radius );

}

void NumViewCorrect ( int NUM, GLfloat X, GLfloat Y, GLfloat Z, int R, int G, int B )

{

switch ( NUM )

{

case 0: GLE.CreateTimeNum\_0 ( X, Y, Z, 0.38f, 0.1f, 0.4f, R, G, B ); break;

case 1: GLE.CreateTimeNum\_1 ( X, Y, Z, 0.38f, 0.1f, 0.4f, R, G, B ); break;

case 2: GLE.CreateTimeNum\_2 ( X, Y, Z, 0.38f, 0.1f, 0.4f, R, G, B ); break;

case 3: GLE.CreateTimeNum\_3 ( X, Y, Z, 0.38f, 0.1f, 0.4f, R, G, B ); break;

case 4: GLE.CreateTimeNum\_4 ( X, Y, Z, 0.38f, 0.1f, 0.4f, R, G, B ); break;

case 5: GLE.CreateTimeNum\_5 ( X, Y, Z, 0.38f, 0.1f, 0.4f, R, G, B ); break;

case 6: GLE.CreateTimeNum\_6 ( X, Y, Z, 0.38f, 0.1f, 0.4f, R, G, B ); break;

case 7: GLE.CreateTimeNum\_7 ( X, Y, Z, 0.38f, 0.1f, 0.4f, R, G, B ); break;

case 8: GLE.CreateTimeNum\_8 ( X, Y, Z, 0.38f, 0.1f, 0.4f, R, G, B ); break;

case 9: GLE.CreateTimeNum\_9 ( X, Y, Z, 0.38f, 0.1f, 0.4f, R, G, B ); break;

}

}

void HELP\_NUM\_A ( int NumA100, int NumA10, int NumA1 )

{

for ( float i = 21.15f; i <= 23.9f; i += 0.55f )

NumViewCorrect ( 0, i, 1.5f, 0.56f, 5, 69, 0 );

NumA100 = fmod ( String -> Ao, 1000 ) \* 0.01;

NumA10 = fmod ( String -> Ao, 100 ) \* 0.1;

NumA1 = fmod ( String -> Ao, 10 );

if ( NumA100 == 0 )

NumViewCorrect ( NumA100, 22.8f, 1.5f, 0.57f, 5, 69, 0 );

else

NumViewCorrect ( NumA100, 22.8f, 1.5f, 0.57f, 167, 222, 0 );

if ( NumA100 == 0 && NumA10 == 0 )

NumViewCorrect ( NumA10, 23.35f, 1.5f, 0.57f, 5, 69, 0 );

else

NumViewCorrect ( NumA10, 23.35f, 1.5f, 0.57f, 167, 222, 0 );

NumViewCorrect ( NumA1, 23.9f, 1.5f, 0.57f, 167, 222, 0 );

}

void HELP\_NUM\_L ( int NumL1000, int NumL100, int NumL10, int NumL1 )

{

for ( float i = 21.15f; i <= 23.9f; i += 0.55f )

NumViewCorrect ( 0, i, 0.0f, 0.56f, 5, 69, 0 );

NumL1000 = fmod ( String -> l, 10000 ) \* 0.001;

NumL100 = fmod ( String -> l, 1000 ) \* 0.01;

NumL10 = fmod ( String -> l, 100 ) \* 0.1;

NumL1 = fmod ( String -> l, 10 );

if ( NumL1000 == 0 )

NumViewCorrect ( NumL1000, 22.25f, 0.0f, 0.57f, 5, 69, 0 );

else

NumViewCorrect ( NumL1000, 22.25f, 0.0f, 0.57f, 167, 222, 0 );

if ( NumL1000 == 0 && NumL100 == 0 )

NumViewCorrect ( NumL100, 22.8f, 0.0f, 0.57f, 5, 69, 0 );

else

NumViewCorrect ( NumL100, 22.8f, 0.0f, 0.57f, 167, 222, 0 );

if ( NumL1000 == 0 && NumL100 == 0 && NumL10 == 0 )

NumViewCorrect ( NumL10, 23.35f, 0.0f, 0.57f, 5, 69, 0 );

else

NumViewCorrect ( NumL10, 23.35f, 0.0f, 0.57f, 167, 222, 0 );

NumViewCorrect ( NumL1, 23.9f, 0.0f, 0.57f, 167, 222, 0 );

}

void HELP\_NUM\_F ( int NumF10000, int NumF1000, int NumF100, int NumF10, int NumF1 )

{

for ( float i = 21.15f; i <= 23.9f; i += 0.55f )

NumViewCorrect ( 0, i, -1.5f, 0.56f, 5, 69, 0 );

NumF10000 = String -> v\_nyu \* 0.0001;

NumF1000 = fmod ( String -> v\_nyu, 10000 ) \* 0.001;

NumF100 = fmod ( String -> v\_nyu, 1000 ) \* 0.01;

NumF10 = fmod ( String -> v\_nyu, 100 ) \* 0.1;

NumF1 = fmod ( String -> v\_nyu, 10 );

if ( NumF10000 == 0 )

NumViewCorrect ( NumF10000, 21.7f, -1.5f, 0.57f, 5, 69, 0 );

else

NumViewCorrect ( NumF10000, 21.7f, -1.5f, 0.57f, 167, 222, 0 );

if ( NumF10000 == 0 && NumF1000 == 0 )

NumViewCorrect ( NumF1000, 22.25f, -1.5f, 0.57f, 5, 69, 0 );

else

NumViewCorrect ( NumF1000, 22.25f, -1.5f, 0.57f, 167, 222, 0 );

if ( NumF10000 == 0 && NumF1000 == 0 && NumF100 == 0 )

NumViewCorrect ( NumF100, 22.8f, -1.5f, 0.57f, 5, 69, 0 );

else

NumViewCorrect ( NumF100, 22.8f, -1.5f, 0.57f, 167, 222, 0 );

if ( NumF10000 == 0 && NumF1000 == 0 && NumF100 == 0 && NumF10 == 0 )

NumViewCorrect ( NumF10, 23.35f, -1.5f, 0.57f, 5, 69, 0 );

else

NumViewCorrect ( NumF10, 23.35f, -1.5f, 0.57f, 167, 222, 0 );

NumViewCorrect ( NumF1, 23.9f, -1.5f, 0.57f, 167, 222, 0 );

}

void HELP\_NUM\_O ( int NumO, int &NOT )

{

for ( float i = 21.15f; i <= 23.9f; i += 0.55f )

NumViewCorrect ( 0, i, -3.0f, 0.56f, 5, 69, 0 );

NumO = String -> v\_nyu;

if ( NumO >= 32 && NumO <= 65 )

{

NumViewCorrect ( 1, 23.9f, -3.0f, 0.57f, 167, 222, 0 );

NOT = 26;

return;

}

if ( NumO >= 65 && NumO <= 131 )

{

NumViewCorrect ( 2, 23.9f, -3.0f, 0.57f, 167, 222, 0 );

NOT = 27;

return;

}

if ( NumO >= 131 && NumO <= 261 )

{

NumViewCorrect ( 3, 23.9f, -3.0f, 0.57f, 167, 222, 0 );

NOT = 28;

return;

}

if ( NumO >= 261 && NumO <= 523 )

{

NumViewCorrect ( 4, 23.9f, -3.0f, 0.57f, 167, 222, 0 );

NOT = 29;

return;

}

if ( NumO >= 523 && NumO <= 1046 )

{

NumViewCorrect ( 5, 23.9f, -3.0f, 0.57f, 167, 222, 0 );

NOT = 30;

return;

}

if ( NumO >= 1046 && NumO <= 2093 )

{

NumViewCorrect ( 6, 23.9f, -3.0f, 0.57f, 167, 222, 0 );

NOT = 31;

return;

}

if ( NumO >= 2093 && NumO <= 4186 )

{

NumViewCorrect ( 7, 23.9f, -3.0f, 0.57f, 167, 222, 0 );

NOT = 32;

return;

}

if ( NumO >= 4186 && NumO <= 8375 )

{

NumViewCorrect ( 8, 23.9f, -3.0f, 0.57f, 167, 222, 0 );

NOT = 33;

return;

}

if ( NumO >= 8375 && NumO <= 16749 )

{

NumViewCorrect ( 9, 23.9f, -3.0f, 0.57f, 167, 222, 0 );

NOT = 34;

return;

}

if ( NumO >= 16749 && NumO <= 33497 )

{

NumViewCorrect ( 1, 23.35f, -3.0f, 0.57f, 167, 222, 0 );

NumViewCorrect ( 0, 23.9f, -3.0f, 0.57f, 167, 222, 0 );

NOT = 35;

return;

}

}

void CorrectNoteLine ( GLfloat F )

{

GLfloat Correct = 0.0f;

float norm = 16.291 / 7;

if ( F < 32 )

{

Correct = 0.0f;

GLE.CreateQuadNoteLine ( 1.894f + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

#pragma region AAAA

if(F>=32 && F<36)

{

Correct=0.581821\*(F-32);

GLE.CreateQuadNoteLine ( 1.894f + norm\*0 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=36 && F<41)

{

Correct=0.465457\*(F-36);

GLE.CreateQuadNoteLine ( 1.894f + norm\*1 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=41 && F<43)

{

Correct=1.163643\*(F-41);

GLE.CreateQuadNoteLine ( 1.894f + norm\*2 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=43 && F<49)

{

Correct=0.387881\*(F-43);

GLE.CreateQuadNoteLine ( 1.894f + norm\*3 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=49 && F<55)

{

Correct=0.387881\*(F-49);

GLE.CreateQuadNoteLine ( 1.894f + norm\*4 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=55 && F<61)

{

Correct=0.387881\*(F-55);

GLE.CreateQuadNoteLine ( 1.894f + norm\*5 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=61 && F<65)

{

Correct=0.581821\*(F-61);

GLE.CreateQuadNoteLine ( 1.894f + norm\*6 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=65 && F<73)

{

Correct=0.290911\*(F-65);

GLE.CreateQuadNoteLine ( 1.894f + norm\*0 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=73 && F<82)

{

Correct=0.258587\*(F-73);

GLE.CreateQuadNoteLine ( 1.894f + norm\*1 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=82 && F<87)

{

Correct=0.465457\*(F-82);

GLE.CreateQuadNoteLine ( 1.894f + norm\*2 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=87 && F<98)

{

Correct=0.211571\*(F-87);

GLE.CreateQuadNoteLine ( 1.894f + norm\*3 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=98 && F<110)

{

Correct=0.193940\*(F-98);

GLE.CreateQuadNoteLine ( 1.894f + norm\*4 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=110 && F<123)

{

Correct=0.179022\*(F-110);

GLE.CreateQuadNoteLine ( 1.894f + norm\*5 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=123 && F<131)

{

Correct=0.290911\*(F-123);

GLE.CreateQuadNoteLine ( 1.894f + norm\*6 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=131 && F<146)

{

Correct=0.155152\*(F-131);

GLE.CreateQuadNoteLine ( 1.894f + norm\*0 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=146 && F<164)

{

Correct=0.290911\*(F-146);

GLE.CreateQuadNoteLine ( 1.894f + norm\*1 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=164 && F<174)

{

Correct=0.232729\*(F-164);

GLE.CreateQuadNoteLine ( 1.894f + norm\*2 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=174 && F<196)

{

Correct=0.193940\*(F-174);

GLE.CreateQuadNoteLine ( 1.894f + norm\*3 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=196 && F<220)

{

Correct=0.096970\*(F-196);

GLE.CreateQuadNoteLine ( 1.894f + norm\*4 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=220 && F<246)

{

Correct=0.089511\*(F-220);

GLE.CreateQuadNoteLine ( 1.894f + norm\*5 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=246 && F<261)

{

Correct=0.155152\*(F-246);

GLE.CreateQuadNoteLine ( 1.894f + norm\*6 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=261 && F<293)

{

Correct=0.083117\*(F-261);

GLE.CreateQuadNoteLine ( 1.894f + norm\*0 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=293 && F<329)

{

Correct=0.064647\*(F-293);

GLE.CreateQuadNoteLine ( 1.894f + norm\*1 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=329 && F<349)

{

Correct=0.116364\*(F-329);

GLE.CreateQuadNoteLine ( 1.894f + norm\*2 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=349 && F<391)

{

Correct=0.055412\*(F-349);

GLE.CreateQuadNoteLine ( 1.894f + norm\*3 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=391 && F<440)

{

Correct=0.047496\*(F-391);

GLE.CreateQuadNoteLine ( 1.894f + norm\*4 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=440 && F<493)

{

Correct=0.043911\*(F-440);

GLE.CreateQuadNoteLine ( 1.894f + norm\*5 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=493 && F<523)

{

Correct=0.077576\*(F-493);

GLE.CreateQuadNoteLine ( 1.894f + norm\*6 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=523 && F<587)

{

Correct=0.043098\*(F-523);

GLE.CreateQuadNoteLine ( 1.894f + norm\*0 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=587 && F<659)

{

Correct=0.032323\*(F-587);

GLE.CreateQuadNoteLine ( 1.894f + norm\*1 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=659 && F<698)

{

Correct=0.059674\*(F-659);

GLE.CreateQuadNoteLine ( 1.894f + norm\*2 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=698 && F<783)

{

Correct=0.027380\*(F-698);

GLE.CreateQuadNoteLine ( 1.894f + norm\*3 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=783 && F<880)

{

Correct=0.023993\*(F-783);

GLE.CreateQuadNoteLine ( 1.894f + norm\*4 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=880 && F<987)

{

Correct=0.021750\*(F-880);

GLE.CreateQuadNoteLine ( 1.894f + norm\*5 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=987 && F<1046)

{

Correct=0.039446\*(F-987);

GLE.CreateQuadNoteLine ( 1.894f + norm\*6 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=1046 && F<1174)

{

Correct=0.018182\*(F-1046);

GLE.CreateQuadNoteLine ( 1.894f + norm\*0 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=1174 && F<1318)

{

Correct=0.016162\*(F-1174);

GLE.CreateQuadNoteLine ( 1.894f + norm\*1 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=1318 && F<1396)

{

Correct=0.029837\*(F-1318);

GLE.CreateQuadNoteLine ( 1.894f + norm\*2 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=1396 && F<1567)

{

Correct=0.013610\*(F-1396);

GLE.CreateQuadNoteLine ( 1.894f + norm\*3 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=1567 && F<1760)

{

Correct=0.012058\*(F-1567);

GLE.CreateQuadNoteLine ( 1.894f + norm\*4 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=1760 && F<1975)

{

Correct=0.010343\*(F-1760);

GLE.CreateQuadNoteLine ( 1.894f + norm\*5 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=1975 && F<2093)

{

Correct=0.019723\*(F-1975);

GLE.CreateQuadNoteLine ( 1.894f + norm\*6 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=2093 && F<2349)

{

Correct=0.009091\*(F-2093);

GLE.CreateQuadNoteLine ( 1.894f + norm\*0 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=2349 && F<2637)

{

Correct=0.008081\*(F-2349);

GLE.CreateQuadNoteLine ( 1.894f + norm\*1 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=2637 && F<2793)

{

Correct=0.014918\*(F-2637);

GLE.CreateQuadNoteLine ( 1.894f + norm\*2 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=2793 && F<3135)

{

Correct=0.006805\*(F-2793);

GLE.CreateQuadNoteLine ( 1.894f + norm\*3 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=3135 && F<3520)

{

Correct=0.006045\*(F-3135);

GLE.CreateQuadNoteLine ( 1.894f + norm\*4 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=3520 && F<3951)

{

Correct=0.005400\*(F-3520);

GLE.CreateQuadNoteLine ( 1.894f + norm\*5 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=3951 && F<4186)

{

Correct=0.009903\*(F-3951);

GLE.CreateQuadNoteLine ( 1.894f + norm\*6 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=4186 && F<4698)

{

Correct=0.004545\*(F-4186);

GLE.CreateQuadNoteLine ( 1.894f + norm\*0 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=4698 && F<5274)

{

Correct=0.004040\*(F-4698);

GLE.CreateQuadNoteLine ( 1.894f + norm\*1 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=5274 && F<5586)

{

Correct=0.007459\*(F-5274);

GLE.CreateQuadNoteLine ( 1.894f + norm\*2 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=5586 && F<6270)

{

Correct=0.003402\*(F-5586);

GLE.CreateQuadNoteLine ( 1.894f + norm\*3 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=6270 && F<7040)

{

Correct=0.003022\*(F-6270);

GLE.CreateQuadNoteLine ( 1.894f + norm\*4 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=7040 && F<7902)

{

Correct=0.002700\*(F-7040);

GLE.CreateQuadNoteLine ( 1.894f + norm\*5 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=7902 && F<8375)

{

Correct=0.004920\*(F-7902);

GLE.CreateQuadNoteLine ( 1.894f + norm\*6 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=8375 && F<9396)

{

Correct=0.002279\*(F-8375);

GLE.CreateQuadNoteLine ( 1.894f + norm\*0 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=9396 && F<10548)

{

Correct=0.002212\*(F-9396);

GLE.CreateQuadNoteLine ( 1.894f + norm\*1 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=10548 && F<11172)

{

Correct=0.003730\*(F-10548);

GLE.CreateQuadNoteLine ( 1.894f + norm\*2 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=11172 && F<12540)

{

Correct=0.001701\*(F-11172);

GLE.CreateQuadNoteLine ( 1.894f + norm\*3 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=12540 && F<14080)

{

Correct=0.001511\*(F-12540);

GLE.CreateQuadNoteLine ( 1.894f + norm\*4 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=14080 && F<15804)

{

Correct=0.001350\*(F-14080);

GLE.CreateQuadNoteLine ( 1.894f + norm\*5 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=15804 && F<16749)

{

Correct=0.002463\*(F-15804);

GLE.CreateQuadNoteLine ( 1.894f + norm\*6 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=16749 && F<18792)

{

Correct=0.001139\*(F-16749);

GLE.CreateQuadNoteLine ( 1.894f + norm\*0 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=18792 && F<21096)

{

Correct=0.001010\*(F-18792);

GLE.CreateQuadNoteLine ( 1.894f + norm\*1 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=21096 && F<22344)

{

Correct=0.001865\*(F-21096);

GLE.CreateQuadNoteLine ( 1.894f + norm\*2 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=22344 && F<25080)

{

Correct=0.000851\*(F-22344);

GLE.CreateQuadNoteLine ( 1.894f + norm\*3 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=25080 && F<28160)

{

Correct=0.000756\*(F-25080);

GLE.CreateQuadNoteLine ( 1.894f + norm\*4 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=28160 && F<31608)

{

Correct=0.000675\*(F-28160);

GLE.CreateQuadNoteLine ( 1.894f + norm\*5 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

if(F>=31608 && F<33497)

{

Correct=0.001232\*(F-31608);

GLE.CreateQuadNoteLine ( 1.894f + norm\*6 + Correct, 0.0f, 0.0f, 255, 0, 0 );

return;

}

#pragma endregion

}

void RenderScene ( )

{

# pragma region Инициализация переменных для числовых дисплеев

int NumA100 = 0;

int NumA10 = 0;

int NumA1 = 0;

int NumL1000 = 0;

int NumL100 = 0;

int NumL10 = 0;

int NumL1 = 0;

int NumF10000 = 0;

int NumF1000 = 0;

int NumF100 = 0;

int NumF10 = 0;

int NumF1 = 0;

int NumO = 0;

# pragma endregion

int NOT = 26;

GLfloat radius = 0.008f;

glClear ( GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT | GL\_STENCIL\_BUFFER\_BIT );

glLoadIdentity ( );

GLfloat length = String -> l \* 0.00007265625f;

glClearColor ( 0.95f, 0.95f, 0.95f, 1 );

glInitNames ( );

g\_Camera.Look ( );

HELP\_NUM\_A ( NumA100, NumA10, NumA1 );

HELP\_NUM\_L ( NumL1000, NumL100, NumL10, NumL1 );

HELP\_NUM\_F ( NumF10000, NumF1000, NumF100, NumF10, NumF1 );

HELP\_NUM\_O ( NumO, NOT );

GLE.CreateBigStend ( Texture, NOT );

GLE.CreateHolderStrings ( Texture, 0.5f, 0.0f, 0.025f );

glTranslatef ( 0.7f, 0.225f, 0.4f );

if ( String -> p > 1.3f )

DrawCorrectLenghtString ( length, 256, String -> p \* 0.007f ); *// max радиус струны 0.04*

else

DrawCorrectLenghtString ( length, 256, radius ); *// max радиус струны 0.04*

glTranslatef ( -0.7f, -0.225f, -0.4f );

glTranslatef ( length \* 256 + 0.7f, 0.0f, 0.0f );

GLE.CreateHolderStrings ( Texture, 0.0f, 0.0f, 0.025f );

glTranslatef ( - length \* 256 - 0.7f, 0.0f, 0.0f );

CorrectNoteLine ( String -> v\_nyu );

glMatrixMode ( GL\_PROJECTION );

glPushMatrix ( );

glLoadIdentity ( );

glOrtho ( 0, 800, 0, 600, 0, 1 );

glMatrixMode ( GL\_MODELVIEW );

glLoadIdentity ( );

wchar\_t Buffer [ 256 ];

glDisable ( GL\_DEPTH\_TEST );

glColor3f ( 0.0f, 0.0f, 0.0f );

swprintf ( Buffer, sizeof ( Buffer ), L"Время: %.3f, с", String->t );

Font -> Font -> FaceSize ( 20 );

Font -> Print ( 40, 45, Buffer );

glEnable ( GL\_DEPTH\_TEST );

glMatrixMode ( GL\_PROJECTION );

glPopMatrix ( );

glMatrixMode ( GL\_MODELVIEW );

glFlush();

SwapBuffers ( g\_hDC );

}

void InitOpenGL (HWND hWnd)

{

g\_hWnd = hWnd;

GetClientRect(g\_hWnd, &g\_rRect);

InitializeOpenGL(g\_rRect.right, g\_rRect.bottom);

Texture = new CTexture ( );

Texture -> InitTextures ( Texture );

Font = new CFont ( "framd.ttf", 10, 10 );

g\_Camera.PositionCamera ( 12.5f, -1.5f, 18.0f, 12.5, -1.5, 1, 0, 1, 0);

}

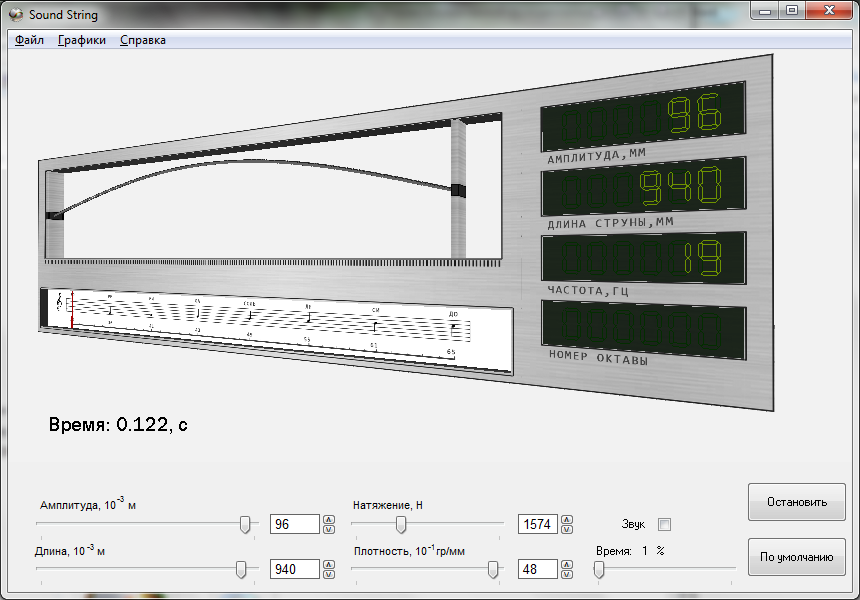
# endif

1. Презентация проекта

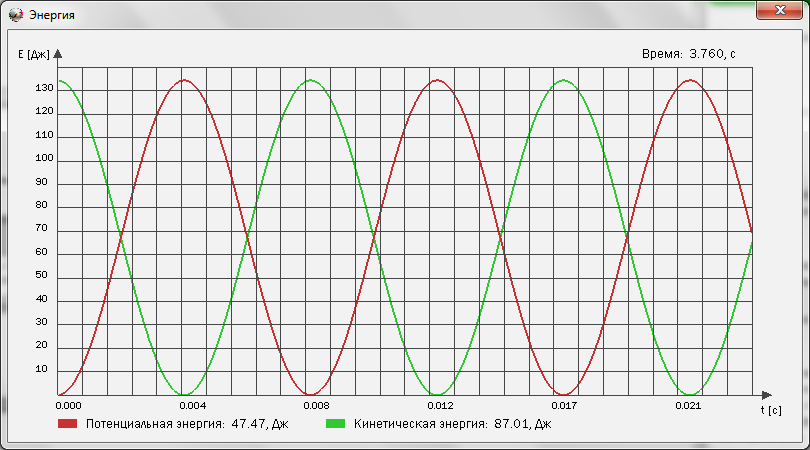
Заставка:



Пример организации главного окна интерфейса:



Пример окна графика:



1. Вывод

В ходе работы была разработана программа, моделирующая стоячие волны на непрерывной струне при различных длине, плотности и силе натяжения струны.

В процессе создания программы мы получили опыт в создании программ моделирования физики. Был приобретен опыт работы в команде. При написании программы было уделено много внимания оптимизации кода, объектно-ориентированным принципам программирования.

1. Литература и ссылки
2. [*MSDN*](http://msdn.microsoft.com/ru-ru/default.aspx)
3. [*«Оптимизация кода на С++»*](http://itw66.ru/blog/c_plus_plus/13.html)
4. [*«Уроки программирования OpenGL»*](http://masandilov.ru/)
5. [*Баяковский Ю., Игнатенко А., Фролов А.*](http://www.ict.edu.ru/ft/002387/ignatenko.pdf)[*Графическая библиотека OpenGL. Методическое пособие*](http://www.ict.edu.ru/ft/002387/ignatenko.pdf) *- М.: МГУ, 2003*
6. [*Райт-мл.* *Р.С., Липчак Б.*](http://www.ozon.ru/context/detail/id/2639015/#tab_person) *OpenGL. Суперкнига – «Вильямс», 2006*
7. *Баранов А.В., Давыдков В.В. Волны. Компьютерный лабораторный практикум по физике. – Н-к.: НГТУ, 2006*
8. *Семченок М.С., Щитов И.Н. Колебания струны. Методическое пособие. - СПб.: СПбГУКиТ, 2010*
9. *Пейн Г. Физика колебаний и волн. - М.: "Мир", 1979*
10. [*Электронное учебное пособие «Математическое моделирование систем и объектов»*](http://de.ifmo.ru/--books/0051/index.html)