

# OpenSSL & OpenGL

*Max Kleiner*

[http://max.kleiner.com/download/openssl\\_opengl.pdf](http://max.kleiner.com/download/openssl_opengl.pdf)



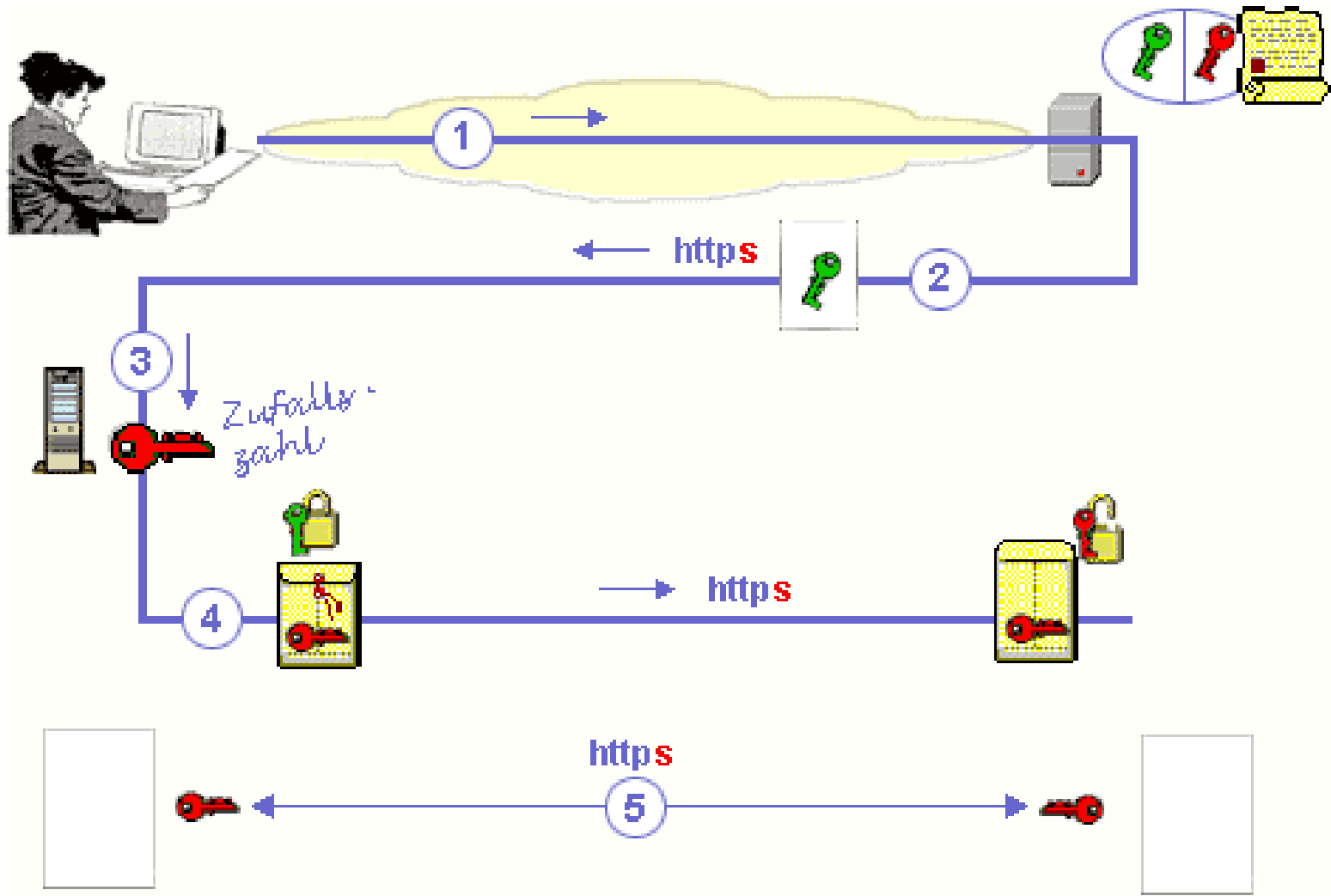
# OpenSSL and OpenGL

- <http://www.openssl.org>
- free library providing cryptographic functions
  - it's not the only one, alternatives: Crypto++ and Cryptlib of Peter Guttman
- the important feature is the complete implementation of the protocols/handshaking of SSLv2, SSLv3 and TLSv1
  
- Der Computer arbeitet deshalb so schnell, weil er nicht denkt.
- Achtung: Lesen kann ihre Dummheit gefährden
- Lang ist der Weg durch Lehren, kurz und wirksam durch Beispiele

# OpenSSL in DWS

- <http://sourceforge.net/projects/delphiwebstart>
- DelphiWebStart (DWS) is an Application Loader with TCP Sockets based on a SmallClient which is first spread over the Web, VPN or Intranet. Then a user can download data (exes, maps, files etc.) from a easy list and start it. DWS 1.8 supports OpenSSL.
- From DelphiWebStart to DataWebSecure (demo)
- the important feature is the complete implementation of the protocols SSLv2,SSLv3, TLSv1 and a non-blocking IO abstraction (BIO)

# This is OpenSSL



# Step by Step

## **Vorbereiten der CA**

1. Generieren eines Schlüsselpaares für die CA
2. Verteilen des CA-Zertifikates auf alle Browser

## **Vorbereiten des Webservers**

3. generieren eines Schlüsselpaares für den Webserver
4. Zertifizierung des Webservers nach Prüfung durch die CA

## **Unsymmetrischer Sitzungsaufbau**

1. Aufbau der Verbindung <https://www.ar.admin.ch> auf Port 443
2. Übertragen des Webserver-Zertifikats zum Browser
3. Prüfen der Signatur des Zertifikats anhand des von der CA hinterlegten Schlüssels, bei Erfolg ist die Identität des Webservers festgestellt
4. Generieren eines temporären Sitzungsschlüssels
5. Senden des Schlüssels in einer nur für den Webserver lesbaren Art
6. Entschlüsseln des Sitzungsschlüssels

## **Symmetrischer SSL-Tunnel**

7. Symmetrische Ver- und Entschlüsselung beim Client
8. Symmetrische Ver- und Entschlüsselung beim Server

# Algorithms implemented

- Block ciphers: DES, 3DES, DESX, CAST, RC2, RC5, IDEA (patent), Blowfish, AES
- stream cipher: RC4, RC5 (patent)
- digests: MD2, MD4, MD5, SHA-1, RIPEMD 160, MDC2 (for smartcards, IBM patent)
- asymmetric cryptosyst.: RSA, DSA, DH
- MAC: HMAC

# Standards implemented

- PKCS 1 (full), PKCS 7 (almost complete for the types actually used: Data, Signed and Enveloped), PKCS 8 (full), PKCS10 (full) and PKCS 12
- X509v3, CSRs, CRLs
- ASN.1 with DER encoding (not complete)
- PEM based ASCII-binary encoding
- SSLv3 and TLSv1 (practically identical)

# Command Shell implemented

- OpenSSL includes a command line utility that can be used to perform a variety of cryptographic functions like generating your machine certificate in [CERT].

First you need a RootCA (selfsigned)

// we generate the private key of the CA:

1. openssl genrsa -des3 -out CA\_pvk.pem 1024

// we sign the private to make a certificate of CA

2. openssl -new -x509 -days 365 -key CA\_pvk.pem -out CA\_cert.pem

// we need the host private key

3. openssl genrsa -des3 -out host\_pvk.pem 1024

// we sign the host private from the CA (machine certificate)

4. openssl req -new key host\_pvk.pem -out host\_csr.pem

5. openssl ca -out host\_cert.pem -in host\_csr.pem -cert CA\_cert.pem -keyfile CA\_pvk.pem

in this way we get:

[CERT]

ROOTCERT=cert\CA\_cert.pem

SCERT=cert\host\_cert.pem

RSAKEY=cert\host\_pvk.pem

# STANDARD COMMANDS (1)

- `asn1parse`
  - parse an ASN.1 sequence
- `ca`
  - Certificate Authority (CA) management
- `ciphers`
  - cipher suite description
- `crl`
  - Certificate Revocation List (CRL) management
- `crl2pkcs7`
  - CRL to PKCS#7 conversion

# STANDARD COMMANDS (2)

- dgst
  - message digest calculation
- dh
  - Diffie-Hellman parameter management.  
Obsoleted by dhparam
- dsa
  - DSA data management
- dsaparam
  - DSA parameter generation
- enc
  - encoding with ciphers

# STANDARD COMMANDS (3)

- genrsa
  - generation of RSA parameters
- ocsp
  - Online Certificate Status Protocol utility
- passwd
  - generation of hashed passwords
- pkcs12
  - PKCS#12 data management
- pkcs7
  - PKCS#7 data management

# STANDARD COMMANDS (4)

- rand
  - generate pseudo-random bytes
- req
  - X.509 Certificate Signing Request (CSR) management
- rsa
  - RSA data management
- rsautl
  - RSA utility for signing, verification, encryption, and decryption

# STANDARD COMMANDS (5)

- **smime**
  - S/MIME mail processing
- **speed**
  - algorithm speed measurement
- **verify**
  - X.509 certificate verification
- **version**
  - OpenSSL version information
- **x509**
  - X.509 certificate data management

# Documentation

- Situation is slowly improving
  - best source: <http://www.openssl.org/docs/>, updated man page (sometimes too updated)
  - for the SSL topic a book in depth is available from Eric Escorla (<http://www.rtfm.com/>) (addison wesley 2001)
    - <http://www.rtfm.com/openssl-examples/>
    - <http://www2.linuxjournal.com/cgi-bin/frames.pl/index.html>
- good support with mailing list:  
<http://www.openssl.org/support/>
- the code of the various demo applications !!!
- the file openssl.txt in the directory doc

# OpenSSL with Indy Version

- current version 0.9.8h (28.5.2008) and Indy10  
(Coming Soon: [Indy 10 for FreePascal and the Lazarus IDE](#))
- Now fits OpenSSL 0.9.8g (Stable) with Indy 9 and 10 (rename IdSSLOpenSSLHeaders9.pas or IdSSLOpenSSLHeaders10.pas to IdSSLOpenSSLHeaders.pas depending on your Indy Version). (<http://www.indyproject.org/>)
- Due changes to the Object Hierarchy you can choice between Intercept or IOHandler:

# Difference Interceptor or Handler

- `TIdTCPConnection.IOHandler` or `TIdTCPConnection.Intercept` properties.
- `Intercept` is used to perform operations that can include logging send and receive operations, or provide Secure Socket Layer (SSL) support for the connection.
- `IOHandler` is used in methods that perform low-level read or write operations like `ReadFromStack` and `WriteBuffer`. `IOHandler` is also used in methods that access the connection status like `CheckForDisconnect`, `Connected`, `DisconnectSocket`, and `Disconnect`.
- Each `IOHandler` can override additional higher level methods to provide optimizations!

# Some remarks

- The SSL capabilities of Indy 10 are now completely pluggable. Prior to Indy 10, the SSL support was pluggable at the TCP level, however protocols such as HTTP which used SSL for HTTPS were fixed to use Indy's default SSL implementation of OpenSSL.
- Indy 10 continues to include support for OpenSSL, so the SSL capabilities of Indy are completely pluggable at the core and protocol level for other implementations or frameworks.
- Verify a signature isn't included, you must specify and implement it on the callback function `onVerifyPeer()` or use the command line function!
- Intelicom.si published their recommendation regarding how to compile OpenSSL with the Indy modification

# Remarks of properties (SSLOptions)

- Set VerifyDepth to 2 means that we accept the server certificate up to 2 levels of Certificate Chain (RootCA --> CA --> ServerCert)
- Set property Method to sslvSSLv23 means the ssl protocol will negotiate the proper mode automatically.
- If [sslvrfPeer] on Server is true and we use OnVerifyPeer(), think about cross certification, means the server will request a client certificate too!

# A closer look to indy9/indy10

```
{$IFDEF INDY10}
  IOHandler:=
    TIdSSLIOHandlerSocketOpenSSL.Create(SoapClient.HTTPWebNode.Http
    pClient);
{$ELSE}
  IOHandler:=
    TIdSSLIOHandlerSocket.Create(SoapClient.HTTPWebNode.HttpClient);
{$ENDIF}
  with {$IFDEF INDY10}
    TIdSSLIOHandlerSocketOpenSSL
  {$ELSE}
    TIdSSLIOHandlerSocket
  {$ENDIF} (IOHandler), SSLOptions do
  begin
```

# OpenGL Overview

- General OpenGL Introduction
- Rendering Primitives
- Rendering Modes
- Lighting
- Texture Mapping
- Imaging

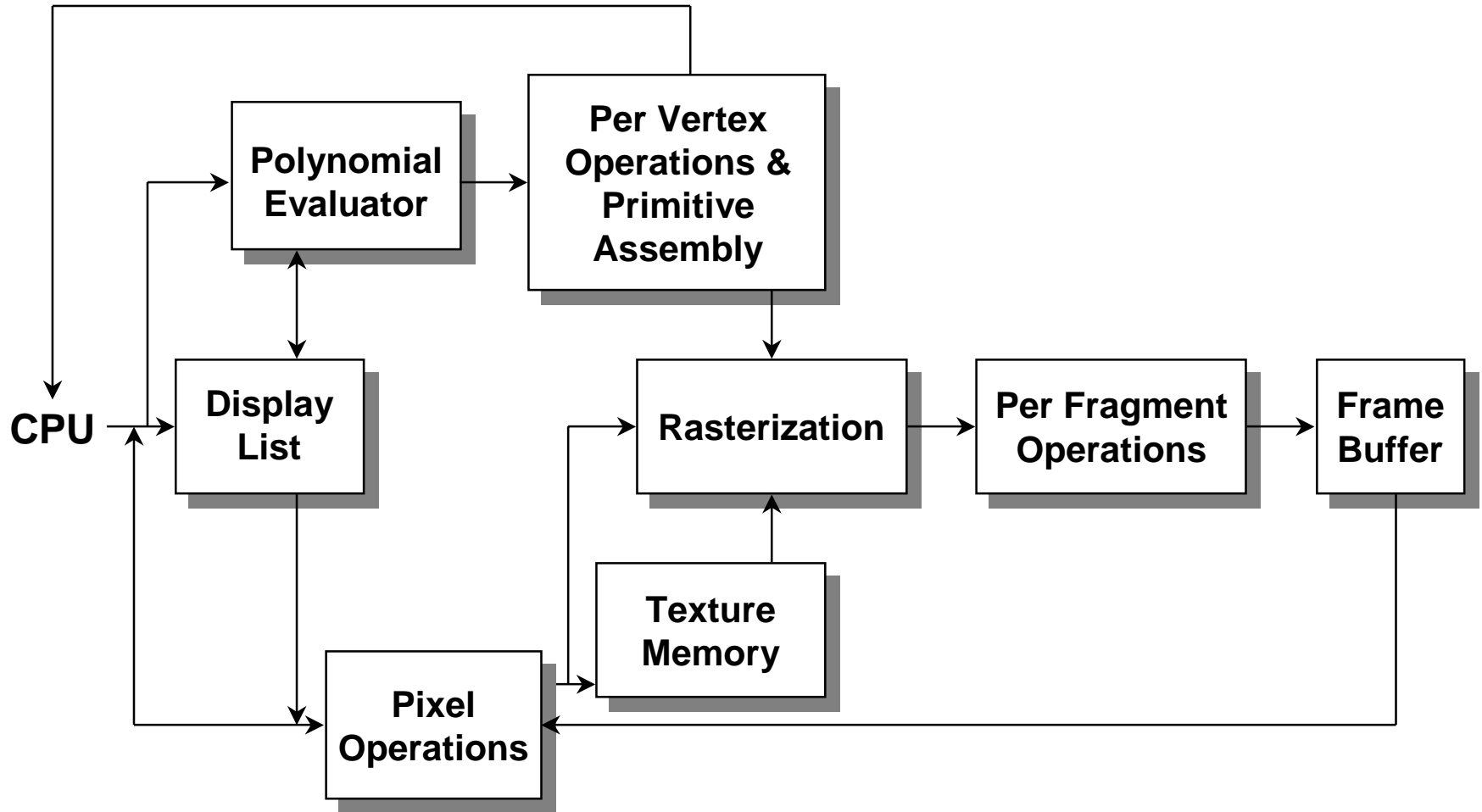
# Simple Code Schema

```
■ drawScene()  
begin  
  //set background color  
  glClearColor()  
  glClear(GL_COLOR_BUFFER_BIT)  
  //color of a polygon primitive  
  glColor3f()  
  //set the polygon  
  glBegin(GL_POLYGON)  
    glVertex2f()  
    glVertex2f()  
    .....  
  glEnd()  
  //draw the GL commands  
  glFlush()  
end
```

# What's OpenGL?

- Graphics rendering API
  - high-quality color images composed of geometric and image primitives with just 150 commands
  - window system independent (context)
  - operating system independent
  - [www.opengl.org](http://www.opengl.org)
- <http://www.softwareschule.ch/download/opengl/delphidemo.zip>
  - Open port from freepascal
  - [http://www.delphi3000.com/articles/article\\_5166](http://www.delphi3000.com/articles/article_5166)

# OpenGL Architecture



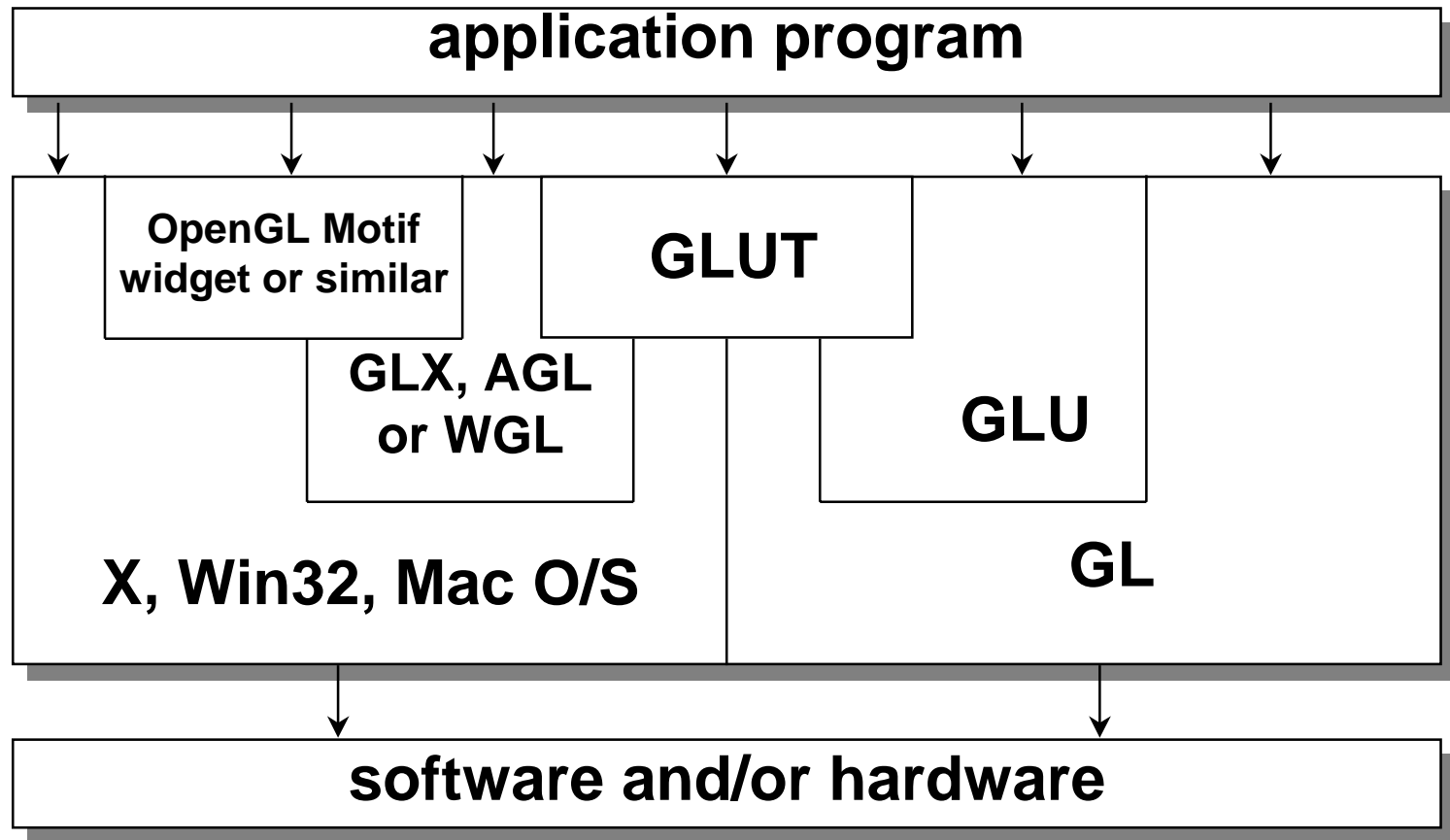
# OpenGL as a Renderer

- Geometric primitives
  - points, lines and polygons
- Image Primitives
  - images and bitmaps
  - separate pipeline for images and geometry
    - linked through texture mapping
- Rendering depends on state
  - colors, materials, light sources, etc.

# Related APIs

- AGL, GLX, WGL
  - glue between OpenGL and windowing systems
- GLU (OpenGL Utility Library)
  - part of OpenGL
  - NURBS, tessellators, quadric shapes, etc.
- GLUT (OpenGL Utility Toolkit)
  - portable windowing API
  - not officially part of OpenGL

# OpenGL and Related APIs



# Preliminaries

## ■ Headers Files

- uses
- Windows, Messages, SysUtils, Classes, Controls, Forms, Buttons, StdCtrls, GL, OpenGLContext\_d;
- GL Converted to Delphi by Tom Nuydens :  
<http://www.delphi3d.net>

## ■ Libraries

- `opengl32 = 'OpenGL32.dll';`  
`glu32 = 'GLU32.dll';`

## ■ Enumerated Types

- OpenGL defines numerous types for compatibility
  - GLfloat, GLint, GLenum, etc.

# GLUT Basics

- Application Structure
  - Configure and open window
  - Initialize OpenGL state
  - Register input callback functions
    - render
    - resize
    - input: keyboard, mouse, etc.
  - Enter event processing loop

# Sa(i)mple Program

```
■ void main( int argc, char** argv )
■ {
■     int mode = GLUT_RGB|GLUT_DOUBLE;
■     glutInitDisplayMode( mode );
■     glutCreateWindow( argv[0] );
■     init();
■     glutDisplayFunc( display );
■     glutReshapeFunc( resize );
■     glutKeyboardFunc( key );
■     glutIdleFunc( idle );
■     glutMainLoop();
■ }
```

# OpenGL Initialization

- Set up whatever state you're going to use

```
■ void init( void )  
■ {  
■     glClearColor( 0.0, 0.0, 0.0, 1.0 );  
■     glClearDepth( 1.0 );  
  
■     glEnable( GL_LIGHT0 );  
■     glEnable( GL_LIGHTING );  
■     glEnable( GL_DEPTH_TEST );  
■ }
```

# GLUT Callback Functions

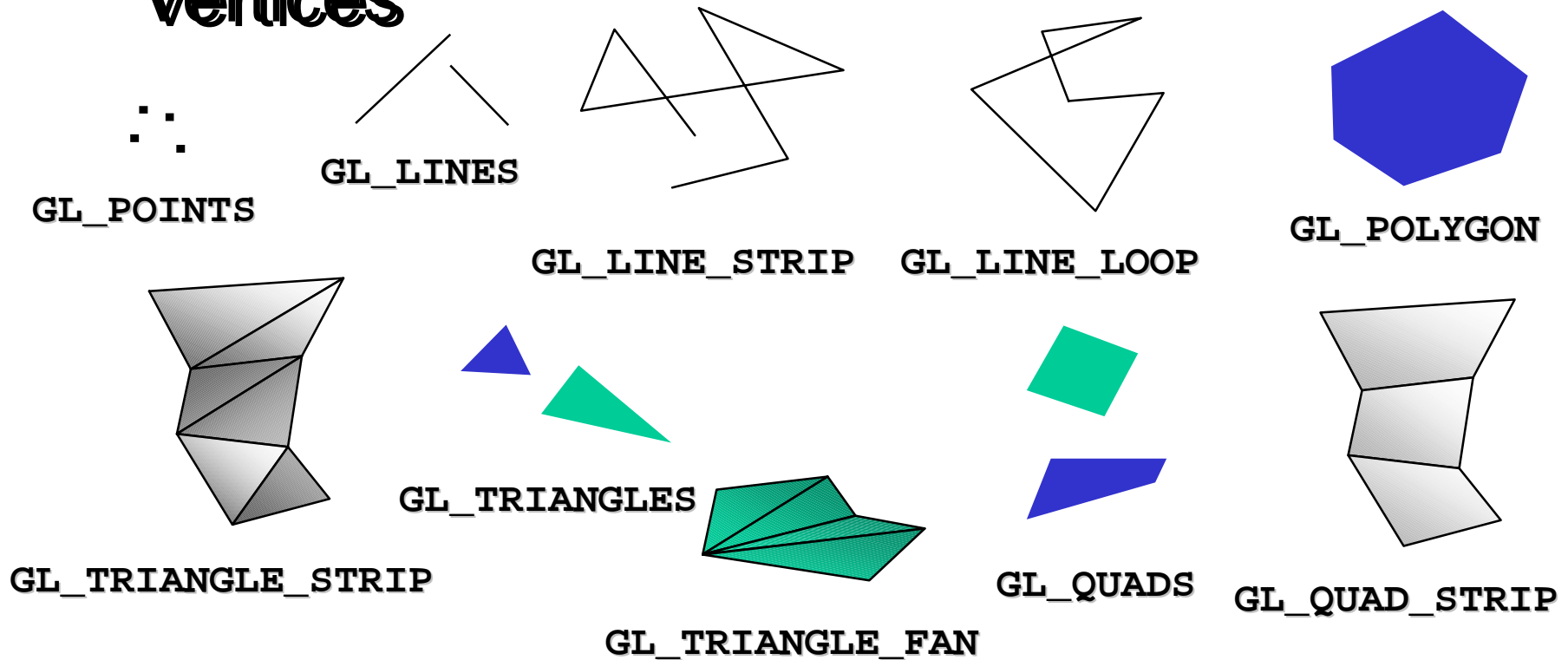
- Routine to call when something happens
  - window resize or redraw
  - user input
  - Animation
- `OnPaint:= OpenGLControl1Paint;`
- `OnResize:= OpenGLControl1Resize;`

# Idle Callbacks

- Use for animation and continuous update
- `Application.OnIdle:= IdleFunc;`
- ```
procedure TGLform.IdleFunc(Sender: TObject; var Done: Boolean);
```
- ```
begin
```
- ```
    openGLControl1.UpdateFrameTimeDiff;
```
- ```
    OpenGLControl1Paint(Self);
```
- ```
    Done:= false;
```
- ```
    if openGLControl1.FrameDiffTimeInMsecs > 2 then
```
- ```
        glform.HintLabel1.Caption:= 'FPS: '
```
- ```
            + inttoStr((1000 div
```
- ```
                openGLControl1.FrameDiffTimeInMsecs));
```
- ```
end;
```

# OpenGL Geometric Primitives

- All geometric primitives are specified by vertices



# Simple Example

```
■ void drawRhombus( GLfloat color[ ] )  
■ {  
    glBegin( GL_QUADS );  
    glColor3fv( color );  
    glVertex2f( 0.0, 0.0 );  
    glVertex2f( 1.0, 0.0 );  
    glVertex2f( 1.5, 1.118 );  
    glVertex2f( 0.5, 1.118 );  
    glEnd();  
■ }
```

# OpenGL Command Formats

`glVertex3fv( v )`

*Number of components*

2 - (x,y)  
3 - (x,y,z)  
4 - (x,y,z,w)

*Data Type*

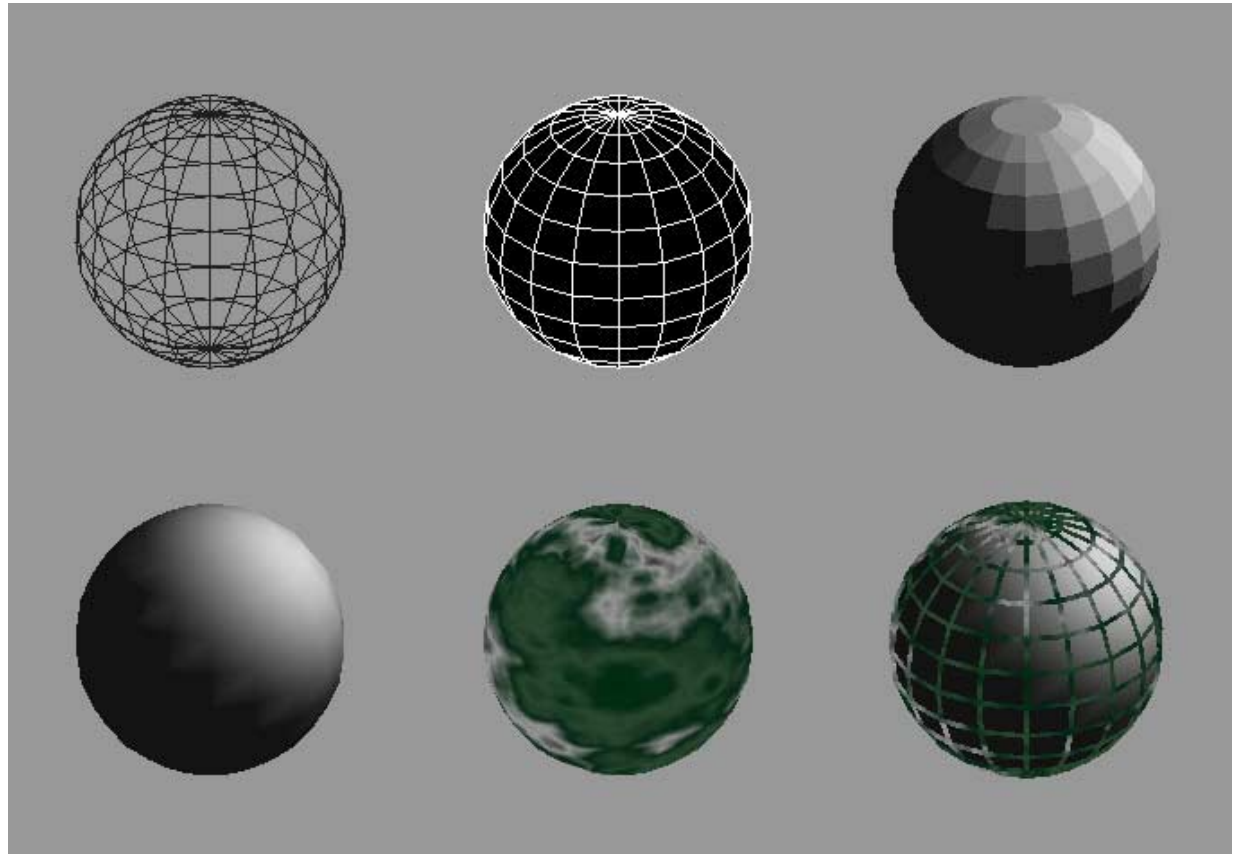
b - byte  
ub - unsigned byte  
s - short  
us - unsigned short  
i - int  
ui - unsigned int  
f - float  
d - double

*Vector*

omit "v" for scalar form  
`glVertex2f( x, y )`

# Controlling Rendering Appearance

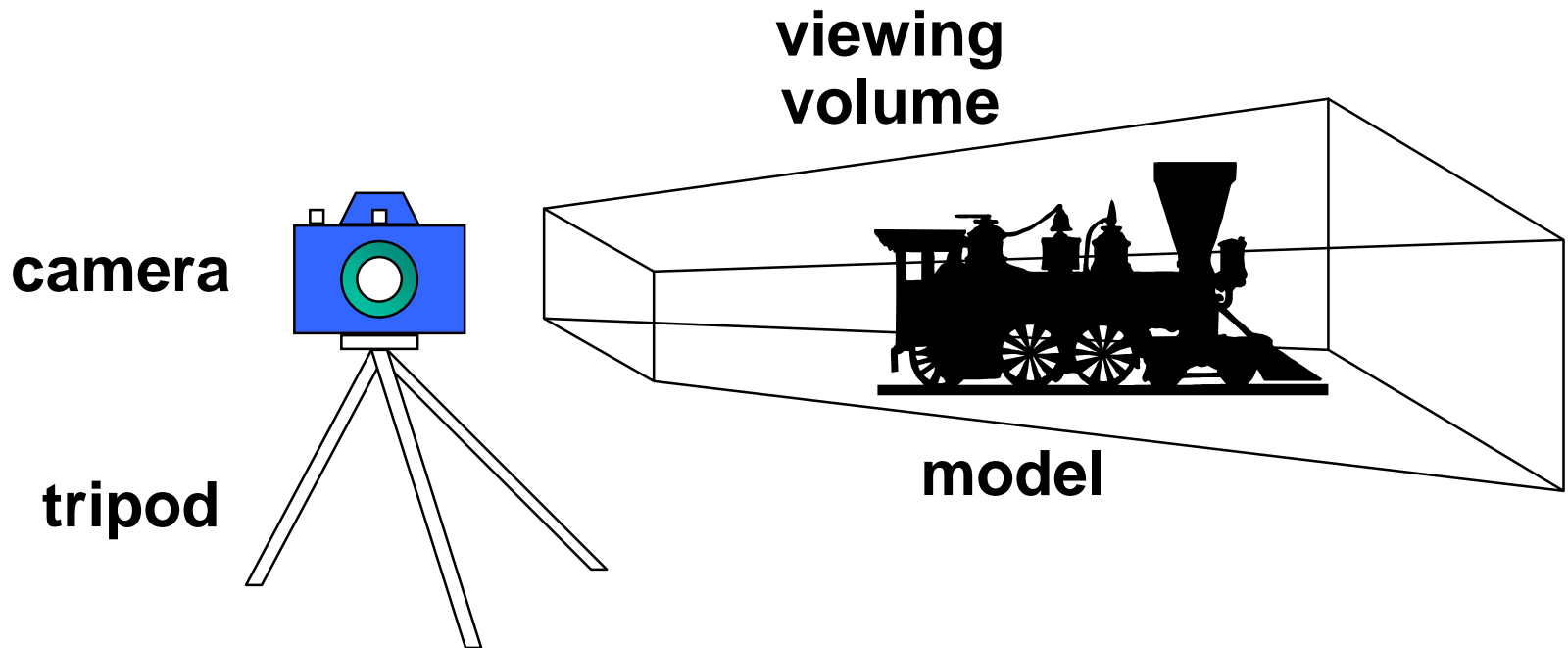
- From
- Wireframe
- to Texture
- Mapped



# Transformations in OpenGL

## Camera Analogy

- 3D is just like taking a photograph (lots of photographs!)



# Camera Analogy and Transformations

- Projection transformations
  - adjust the lens of the camera
- Viewing transformations
  - tripod—define position and orientation of the viewing volume in the world
- Modeling transformations
  - moving the model
- Viewport transformations
  - enlarge or reduce the physical photograph

---

# That's all Folks

Fazit:

Der Transformator openSSL und  
openGL lässt sich nun in jede Lok  
integrieren!