Lecture 07

Debug

- Error in the VBO code in the last lecture (The one uploaded to the ramennoodle server is correct.)

Error

```
template <class T>
GLuint PushBufferSubData(GLuint &currentPtr,GLuint n,const T incoming[])
{
    auto returnPtr=currentPtr;
    auto bufLength=sizeof(T)*n;
    glBufferSubData(GL_ARRAY_BUFFER,currentPtr,bufLength,incoming);
    currentPtr+=bufLength;
    return currentPtr;
}
```

Correct

```
template <class T>
GLuint PushBufferSubData(GLuint &currentPtr,GLuint n,const T incoming[])
{
    auto returnPtr=currentPtr;
    auto bufLength=sizeof(T)*n;
    glBufferSubData(GL_ARRAY_BUFFER,currentPtr,bufLength,incoming);
    currentPtr+=bufLength;
    return returnPtr;
}
```

- Clean up view control with YsViewControl class
- Dealing with a binary data file
- Binary-STL reader
- Picking – Going back and forth between 2D and 3D

YsViewControl class

- Inherited from YsViewPoint and YsProjectionTransformation classes.
- YsViewPoint class provices view-point management, matrix calculation.
- YsProjectionTransformation class provides projection-matrix calculation.
- See ysviewcontrol.h

1. Copy project from simple-shadow, and rename TARGET_NAME to glsl3d_ysviewcontrol
2. In FsLazyWindowApplication class, replace:

```
double h,p,b;
YsVec3 viewTarget;
double viewDist;
```

   with:

```
Ys3DDrawingEnvironment drawEnv;
```

3. In the constructor, replace:

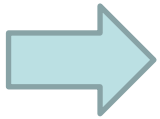
```
h=YsPi/4.0;
p=YsPi/5.0;
b=0;
viewTarget.Set(0.0,0.0,0.0);
viewDist=20.0;
```

   with:

```
drawEnv.SetProjectionMode(YsProjectionTransformation::PERSPECTIVE);
drawEnv.SetViewTarget(YsVec3::Origin());
drawEnv.SetViewDistance(20.0);
drawEnv.SetViewAttitude(YsAtt3(YsPi/4.0,YsPi/5.0,0.0));
```

4. In Interval function, replace:

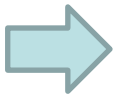
```
if(0!=lb && (mx!=prevMx || my!=prevMy))
{
    double denom=(double)YsGreater(wid,hei);
    double dx=2.0*(double)(prevMx-mx)/denom;
    double dy=2.0*(double)(prevMy-my)/denom;

    YsVec3 fv(0,0,-1),uv(0,1,0);
    fv.RotateXZ(-dx);
    fv.RotateZY(-dy);
    uv.RotateXZ(-dx);
    uv.RotateZY(-dy);

    YsMatrix3x3 view; // Camera to World
    view.RotateXZ(h);
    view.RotateZY(p);
    view.RotateXY(b);
    fv=view*fv;
    uv=view*uv;

    h=atan2(fv.x(),-fv.z());
    p=asin(YsBound(-fv.y(),-1.0,1.0));
    b=0.0; // Tentative

    YsMatrix3x3 newView;
    newView.RotateXZ(h);
    newView.RotateZY(p);
    newView.RotateXY(b); // In fact it's no rotation.
    newView.MulInverse(uv,uv);
    b=atan2(-uv.x(),uv.y());
}
```



```
if(0!=lb && (mx!=prevMx || my!=prevMy))
{
    double denom=(double)YsGreater(wid,hei);
    double dx=2.0*(double)(prevMx-mx)/denom;
    double dy=2.0*(double)(prevMy-my)/denom;
    drawEnv.RotateView(dx,dy);
}
```

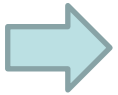
5. In Draw function, replace:

```
YsProjectionTransformation proj;
proj.SetProjectionMode(
    YsProjectionTransformation::PERSPECTIVE);
proj.SetAspectRatio((double)wid/(double)hei);
proj.SetFOVY(YsPi/4.0);
proj.SetNearFar(0.1,100.0);

GLfloat projMat[16];
proj.GetProjectionMatrix().
    GetOpenGlCompatibleMatrix(projMat);
```



```
drawEnv.SetProjectionMode(
    YsProjectionTransformation::PERSPECTIVE);
drawEnv.SetAspectRatio((double)wid/(double)hei);
drawEnv.SetFOVY(YsPi/4.0);
drawEnv.SetNearFar(0.1,100.0);

GLfloat projMat[16];
drawEnv.GetProjectionMatrix().
    GetOpenGlCompatibleMatrix(projMat);
```

```
YsMatrix4x4 view;
view.Translate(0,0,-viewDist);
view.RotateXY(-b);
view.RotateZY(-p);
view.RotateXZ(-h);
view.Translate(-viewTarget);
```



```
auto &view=drawEnv.GetViewMatrix();
```

Dealing with a binary data file

Binary file

- Just a sequence of numbers.
- Not in the human-readable format.
- Program needs to interpret.

Binary dump

- Often used for inspecting a binary file.
- Prints 256 numbers per chunk, 16 numbers per line.
- Numbers shown in hexa-decimal number.

```
00000000l 64 86 06 00 7e 57 9e 56 3f 05 00 00 24 00 00 00
00000010l 00 00 00 00 2e 64 72 65 63 74 76 65 00 00 00 00
00000020l 00 00 00 00 2f 00 00 00 04 01 00 00 00 00 00 00
00000030l 00 00 00 00 00 00 00 00 00 0a 10 00 2e 64 65 62
00000040l 75 67 24 53 00 00 00 00 00 00 00 00 bc 00 00 00
00000050l 33 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000060l 40 00 10 42 2e 72 64 61 74 61 00 00 00 00 00 00
00000070l 00 00 00 00 54 00 00 00 ef 01 00 00 00 00 00 00
00000080l 00 00 00 00 00 00 00 00 40 00 40 40 2e 74 65 78
00000090l 74 24 6d 6e 00 00 00 00 00 00 00 00 c8 01 00 00
000000a0l 43 02 00 00 0b 04 00 00 00 00 00 00 13 00 00 00
000000b0l 20 00 50 60 2e 78 64 61 74 61 00 00 00 00 00 00
000000c0l 00 00 00 00 18 00 00 00 c9 04 00 00 e1 04 00 00
000000d0l 00 00 00 00 01 00 00 00 40 00 30 40 2e 70 64 61
000000e0l 74 61 00 00 00 00 00 00 00 00 00 00 18 00 00 00
000000f0l eb 04 00 00 03 05 00 00 00 00 00 00 06 00 00 00
```

Binary dump program.

```
#include <stdio.h>


void PrintDump(const char fn[])
{
    FILE *fp=fopen(fn,"rb");
    if(nullptr!=fp)
    {
        long long int offset=0,readSize;
        unsigned char buf[256];
        while(0<(readSize=fread(buf,1,256,fp)))
        {
            for(int i=0; i<256; i+=16)
            {
                printf("%08llxl",offset+i);
                for(int j=0; j<16; ++j)
                {
                    if(readSize<=i+j)
                    {
                        break;
                    }
                    printf(" %02x",buf[i+j]);
                }
                printf("\n");
            }
            printf("\n");
            offset+=256;
        }
        fclose(fp);
    }
    else
    {
        printf("Cannot open file.\n");
    }
}
```

```
int main(int argc,char *argv[])
{
    if(2==argc)
    {
        PrintDump(argv[1]);
    }
    else
    {
        printf("Usage: dump.exe <filename>\n");
        return 0;
    }
}
```

Hexa-decimal number

- 1 digit represents from 0-15.

0 => 0	8 => 8
1 => 1	9 => 9
2 => 2	A => 10
3 => 3	B => 11
4 => 4	C => 12
5 => 5	D => 13
6 => 6	E => 14
7 => 7	F => 15

- In C++, you can write a hexadecimal number by adding 0x in front.
- Example:

Hexa-Decimal		Decimal		
0x0A	=>	$0*16^1 + 10*16^0$	=	10
0x80	=>	$8*16^1 + 0*16^0$	=	128
0xA0	=>	$10*16^1 + 0*16^0$	=	160
0xFF	=>	$15*16^1 + 15*16^0$	=	255
0x100	=>	$1*16^2 + 0*16^1 + 0*16^0$	=	256

Binary STL file

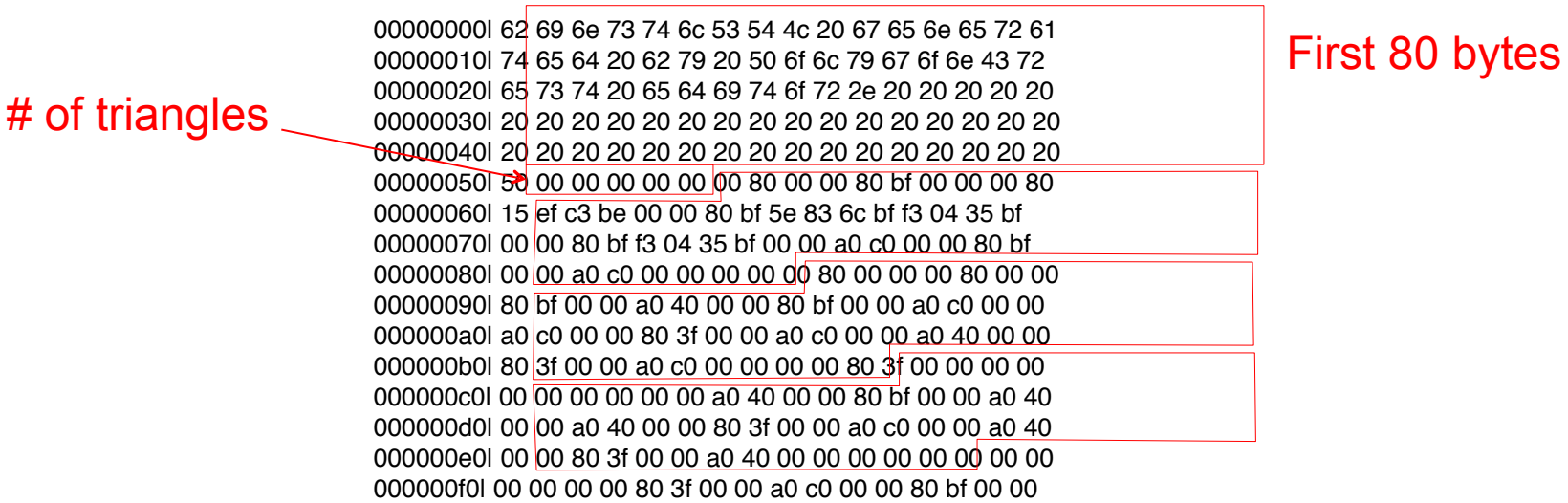
80 bytes Comment in ASCII characters

4 bytes Number of triangles

50 bytes Normal, Three (x,y,z) coords, and volume id.

50 bytes Normal, Three (x,y,z) coords, and volume id.

:



Reading a binary data file

- Open a file with “rb” mode.
- Use fread function to read binary data.
  fread(buf,unit,count,fp);
  buf unsigned char pointer to the data buffer.
  unit Unit size. Usually 1.
  count Number of units to read.
  fp File pointer.
  It reads unit*count bytes from the file fp.

Interpreting a binary data file.

- Everything is in an array of unsigned char.
- Converting four bytes of unsigned chars to an unsigned integer:

  (Input is const unsigned char dat[4], output is unsigned int value.)

```
unsigned char b0=dat[0];
unsigned char b1=dat[1];
unsigned char b2=dat[2];
unsigned char b3=dat[3];
unsigned int value=b0+b1*0x100+b2*0x10000+b3*0x1000000;
```

- Converting four bytes of unsigned chars (in IEEE standard floating-point format) to a float:

  (Input is const unsigned char dat[4], output is float value.)

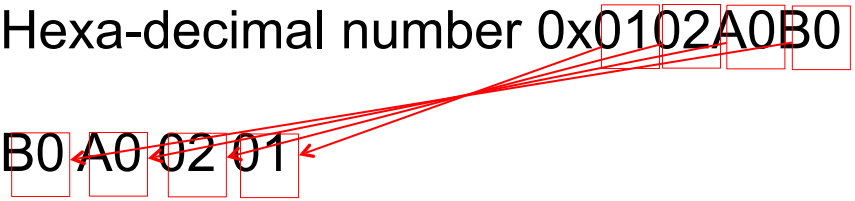
```
const float *fPtr=(const float *)dat;
float value=*fPtr;
```

What if the world is not an Intel world?

- The byte-order problem. (Little endian vs. Big endian.)
- Little endian: Least-Significant Byte appears first.

  Hexa-decimal number 0x0102A0B0 will be stored as:

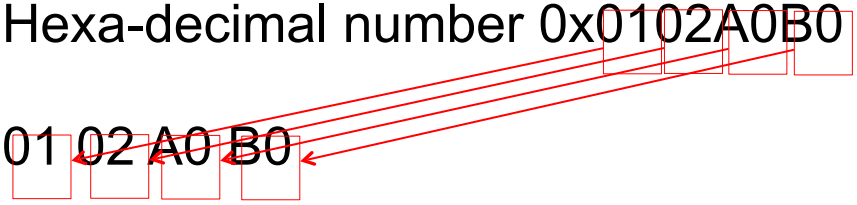
  B0 A0 02 01



- Big endian: Most-Significant Byte appears first.

  Hexa-decimal number 0x0102A0B0 will be stored as:

  01 02 A0 B0



- To write a general-purpose binary-reading code, you need to be aware of the endian-ness of the file and the CPU.

- Historically, Intel CPUs have been using little endian, and Motorola CPUs have been using big endian.
- ARM uses bi-endian, which can switch the endian-ness. But, probably the program may not know until run time.

- Hopefully the CPU uses same endian-ness for all data types.

Question:

- How can you identify the endian-ness of the CPU in C++?

```
bool CPUisLittleEndian(void)
{
    ?
}
```

Endian-ness of a Binary STL

- Known to be Little-Endian. Compatible with Intel CPU.
- This integer-conversion works regardless of the CPU.

  (Input is const unsigned char dat[4], output is unsigned int value.)

```
unsigned char b0=dat[0];
unsigned char b1=dat[1];
unsigned char b2=dat[2];
unsigned char b3=dat[3];
unsigned int value=b0+b1*0x100+b2*0x10000+b3*0x1000000;
```

- If CPU's endian-ness is big-endian, floating-point conversion needs to be:

```
if(true==CPUisLittleEndian())
{
    auto *fPtr=(const float *)dat;
    value=*fPtr;
}
else
{
    auto *valuePtr=(unsigned char *)(&value);
    valuePtr[0]=dat[3];
    valuePtr[1]=dat[2];
    valuePtr[2]=dat[1];
    valuePtr[3]=dat[0];
}
```

Copy byes to where the value is stored in the reverse order.

Reading and displaying a binary STL

1. Copy a project from ysviewcontrol. Rename TARGET_NAME to glsl3d_binary_stl
2. Copy vertex_buffer_object.h and add to CMakeLists.txt
3. Add member variables:

```
std::vector <float> vtx,nom;
YsVec3 min,max;
```

4. Add binary_stl.h and binary_stl.cpp (also add in CMakeLists.txt), binary_stl.h should be included from main.cpp

binary_stl.h

```
#ifndef BINARY_STL_IS_INCLUDED
#define BINARY_STL_IS_INCLUDED

#include "vector"
void LoadBinaryStl(std::vector <float> &vtx,std::vector <float> &nom,const char fn[]);

#endif
```

binary_stl.cpp

```
#include "binary_stl.h"

bool CPUisLittleEndian(void)
{
   unsigned int one=1;
   auto *dat=(const unsigned char *)&one;
   if(1==dat[0])
   {
      return true;
   }
   return false;
}
int BinaryToInt(const unsigned char dw[4])
{
   int b0=(int)dw[0];
   int b1=(int)dw[1];
   int b2=(int)dw[2];
   int b3=(int)dw[3];
   return b0+b1*0x100+b2*0x10000+b3*0x1000000;
}
float BinaryToFloat(const unsigned char dw[4])
{
   if(true==CPUisLittleEndian())
   {
      const float *fPtr=(const float *)dw;
      return *fPtr;
   }
   else
   {
      float value;
      auto *valuePtr=(unsigned char *)&value;
      valuePtr[0]=dw[3];
      valuePtr[1]=dw[2];
      valuePtr[2]=dw[1];
      valuePtr[3]=dw[0];
      return value;
   }
}
```

Incoming bytes are little-endian. This is universal conversion to an unsigned integer.

If the endian-ness of the CPU matches the endian-ness of the STL, the bytes are already ordered. No conversion necessary.

If the endian-ness of the CPU is reverse of the endian-ness of the STL, the bytes are reversed. First reverse the bytes and then convert.

binary_stl.cpp

```
void AddBinaryStlTriangle(std::vector <float> &vtx,std::vector <float> &nom,const unsigned char buf[50])
{
    float nx=BinaryToFloat(buf),ny=BinaryToFloat(buf+4),nz=BinaryToFloat(buf+8);
    nom.push_back(nx);
    nom.push_back(ny);
    nom.push_back(nz);
    nom.push_back(nx);
    nom.push_back(ny);
    nom.push_back(nz);
    nom.push_back(nx);
    nom.push_back(ny);
    nom.push_back(nz);

    vtx.push_back(BinaryToFloat(buf+12));
    vtx.push_back(BinaryToFloat(buf+16));
    vtx.push_back(BinaryToFloat(buf+20));
    vtx.push_back(BinaryToFloat(buf+24));
    vtx.push_back(BinaryToFloat(buf+28));
    vtx.push_back(BinaryToFloat(buf+32));
    vtx.push_back(BinaryToFloat(buf+36));
    vtx.push_back(BinaryToFloat(buf+40));
    vtx.push_back(BinaryToFloat(buf+44));

    // buf[48] and buf[49] are volume identifier, which is usually not used.
}
```

binary_stl.cpp

```
void LoadBinaryStl(std::vector <float> &vtx,std::vector <float> &nom,const char fn[])
{
   FILE *fp=fopen(fn,"rb");
   if(nullptr!=fp)
   {
      unsigned char title[80];
      fread(title,1,80,fp); // Skip title

      unsigned char dw[4];
      fread(dw,4,1,fp);  // Read 4 bytes
      auto nTri=BinaryToInt(dw);
      printf("%d triangles\n",nTri);

      int nTriActual=0;
      vtx.clear();
      nom.clear();
      for(int i=0; i<nTri; ++i)
      {
         unsigned char buf[50];  // 50 bytes per triangle
         if(50==fread(buf,1,50,fp))
         {
            AddBinaryStlTriangle(vtx,nom,buf);
            ++nTriActual;
         }
         else
         {
            break;
         }
      }
      printf("Actually read %d\n",nTriActual);

      fclose(fp);
   }
}
```

Alternative way of reading normal vectors and vertex positions (if you know that the CPU's endianness is same as the STL's)

```
int nTriActual=0;
vtx.clear();
nom.clear();
for(int i=0; i<nTri; ++i)
{
    float buf[12];
    if(48==fread(buf,1,48,fp))
    {
        nom.push_back(buf[0]); nom.push_back(buf[1]); nom.push_back(buf[2]);
        nom.push_back(buf[0]); nom.push_back(buf[1]); nom.push_back(buf[2]);
        nom.push_back(buf[0]); nom.push_back(buf[1]); nom.push_back(buf[2]);

        vtx.push_back(buf[ 3]); vtx.push_back(buf[ 4]); vtx.push_back(buf[ 5]);
        vtx.push_back(buf[ 6]); vtx.push_back(buf[ 7]); vtx.push_back(buf[ 8]);
        vtx.push_back(buf[ 9]); vtx.push_back(buf[10]); vtx.push_back(buf[11]);

        unsigned char skip[2];
        fread(skip,1,2,fp);

        ++nTriActual;
    }
}
```

Reading and displaying a binary STL

5. In FsLazyWindowApplication add the following function:

```
void FsLazyWindowApplication::LoadBinaryStl(const char fn[])
{
    ::LoadBinaryStl(vtx,nom,fn);
    CacheBoundingBox();
}
```

5. Modify BeforeEverything function as:

```
void FsLazyWindowApplication::BeforeEverything(int argc,char *argv[])
{
   if(2<=argc)
   {
      LoadBinaryStl(argv[1]);
   }
}
```

6. Delete the following (and erase where these are used):

```
   YsAtt3 cubeAtt;
   YsVec3 cubePos;

   std::vector <GLfloat> cubeVtx;
   void MakeCube(double d);
   void DrawPlainCube(YsGLSLShaded3DRenderer &renderer) const;
   void DrawPlainCube(YsGLSLPlain3DRenderer &renderer) const;
```

7. Add CacheBoundingBox function and call after LoadBinaryStl.

```
void FsLazyWindowApplication::CacheBoundingBox(void)
{
    auto nVtx=vtx.size()/3;
    // Cache bounding box
    if(0<nVtx)
    {
        float minx,miny,minz,maxx,maxy,maxz;
        minx=vtx[0];
        miny=vtx[1];
        minz=vtx[2];
        maxx=vtx[0];
        maxy=vtx[1];
        maxz=vtx[2];
        for(decltype(nVtx) i=0; i<nVtx; ++i)
        {
            YsMakeSmaller(minx,vtx[i*3]);
            YsMakeSmaller(miny,vtx[i*3+1]);
            YsMakeSmaller(minz,vtx[i*3+2]);
            YsMakeGreater(maxx,vtx[i*3]);
            YsMakeGreater(maxy,vtx[i*3+1]);
            YsMakeGreater(maxz,vtx[i*3+2]);
        }
        min.Set(minx,miny,minz);
        max.Set(maxx,maxy,maxz);
    }
    else
    {
        min=YsVec3::Origin();
        max=YsVec3::Origin();
    }
}
```

Just caching min and max of x,y,z coordinates.

8. In Draw function add:

```
drawEnv.SetViewTarget((min+max)/2.0);
drawEnv.SetViewDistance((max-min).GetLength());
```

Delete:

```
modeling.Translate(cubePos);
modeling.RotateXZ(cubeAtt.h());
modeling.RotateZY(cubeAtt.p());
modeling.RotateXY(cubeAtt.b());
```

Replace:

```
DrawPlainCube(renderer);
```

```
GLfloat color[4]={0,0,1,1};
renderer.SetUniformColor(color);
renderer.DrawVtxNom(
    GL_TRIANGLES,vtx.size()/3,vtx.data(),nom.data());
```

```
DrawPlainCube(renderer);
```

```
renderer.DrawVtx(GL_TRIANGLES,vtx.size()/3,vtx.data());
```

Identifying ASCII or Binary

- When you have a binary STL file, it is nearly impossible to check if it is really a binary STL file, or something else, or a corrupted binary STL file.
- But, if you know that the file is an STL file, you can reliably identify if it is an ASCII STL or Binary STL.
- An ASCII STL file includes some keywords, “solid”, “facet”, “loop”, and “vertex”.
- Read first 1000 bytes of the file, and check if these keywords are included.
- It is no more than a guess, but all you can do is to guess.

```
bool IsAsciiStl(const char fn[])
{
    FILE *fp=fopen(fn,"rb");
    if(nullptr!=fp)
    {
        char buf[1024];
        fread(buf,1,1024,fp);
        fclose(fp);

        bool solid,facet,loop,vertex;
        solid=false;
        facet=false;
        loop=false;
        vertex=false;
        for(i=0; i<1018; i++)
        {
            if(strncmp(buf+i,"solid",5)==0)
            {
                solid=true;
            }
            else if(strncmp(buf+i,"facet",5)==0)
            {
                facet=true;
            }
            else if(strncmp(buf+i,"loop",4)==0)
            {
                loop=true;
            }
            else if(strncmp(buf+i,"vertex",6)==0)
            {
                vertex=true;
            }
        }

        if(true==solid && true==facet && true==loop && true==vertex)
        {
            return true;
        }
    }
    return false;
}
```

In binary_stl.cpp.
Function prototype in binary_stl.h

- After adding IsAsciiSTL function:

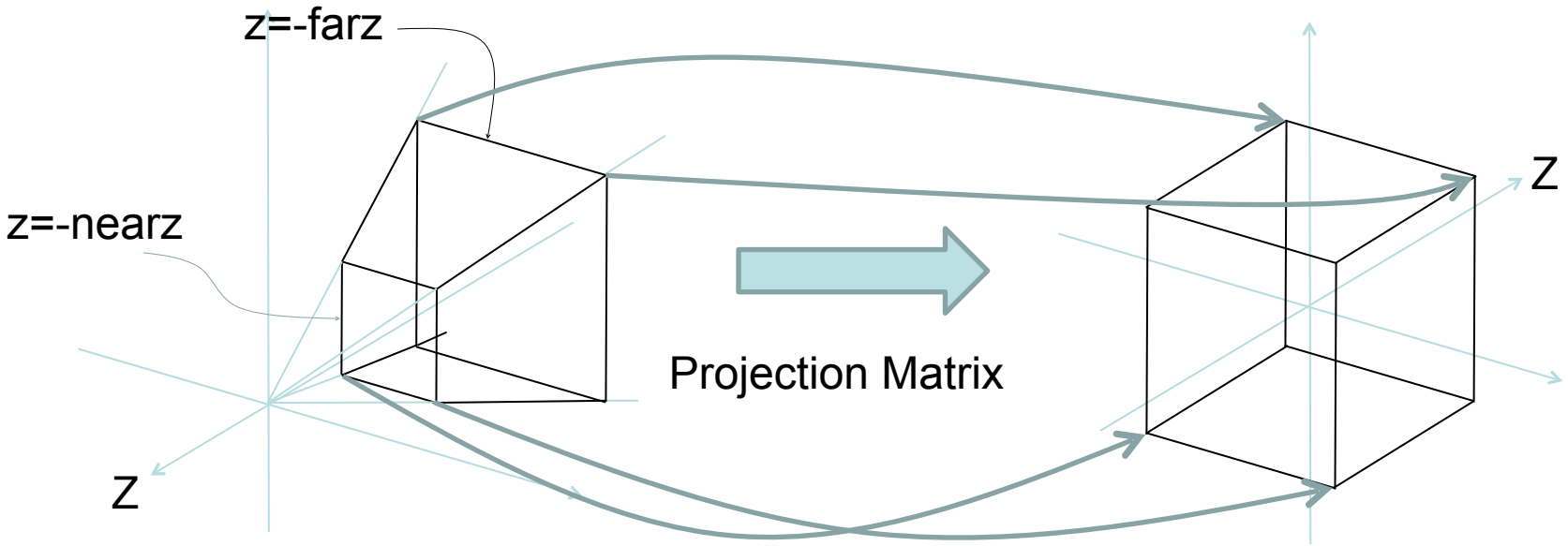
```
/* virtual */ void FsLazyWindowApplication::BeforeEverything(int argc,char *argv[])
{
    if(2<=argc)
    {
        if(true==::IsAsciiStl(argv[1]))
        {
            printf("It is not a binary stl!\n");
            return;
        }
        LoadBinaryStl(argv[1]);
    }
}
```

Picking

- Identifying which primitive is under the mouse cursor.
- Possible options:
  1. Using OpenGL's picking feature – Very poorly designed. May not be supported in the newer versions.
  2. Drawing primitives in different color, and then check the color of the pixel under the mouse cursor – The actual RGB value written to the pixel may be reduced to as low as 12 bits. The identification information may be lost.
  3. Transform mouse pointer to a 3D line by the inverse of projection and view matrix, and calculate intersection with the primitives. - The most reliable. Can easily be parallelized.

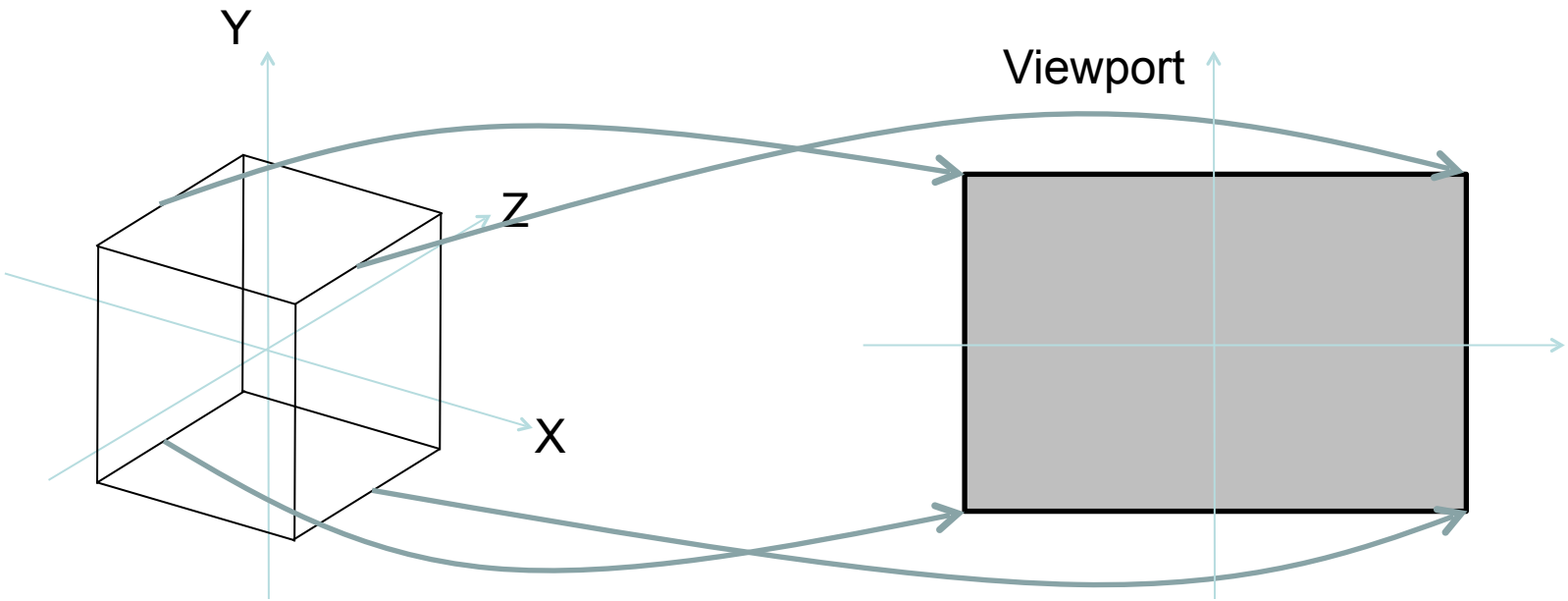
Projection transformation

- OpenGL projection transforms a view frustum into a cube (-1,-1,-1)-(1,1,1)
- Z direction will be inverted after the projection. Larger z means forward after the projection. (As shown in the thick arrow in the figure.)



Viewport transformation

- (x,y)=(-1,-1) to (1,1) in the projected coordinate are linearly mapped to the viewport, which is in general same size as the window, specified by glViewport.



Transforming a mouse coordinate to a 3D line

1. Normalize window coordinate (x_s,y_s) to (-1,-1)-(1,1) with respect to the viewport => (x_v,y_v)
2. Inverse-transform $(x_v,y_v,-1)$ and $(x_v,y_v,1)$ by the projection matrix => (x_n,y_n,z_n), (x_f,y_f,z_f)
3. Inverse-transform (x_n,y_n,z_n) and (x_f,y_f,z_f) by the model-view matrix => (x_1,y_1,z_1), (x_2,y_2,z_2)

Let's draw it on the screen.

1. Copy project from binary-stl sample, rename TARGET_NAME to glsl3d_mouse_to_line
2. Add member variables:

   ```
   YsVec3 lastClick[2];
   ```

3. Initialize lastClick[0] and lastClick[1] in the constructor as:

   ```
   lastClick[0]=YsVec3::Origin();
   lastClick[1]=YsVec3::Origin();
   ```

4. In Interval function, add:

```
if(evt==FSMOUSEEVENT_LBUTTONDOWN)
{
    double x=(double)mx/(double)wid;
    double y=(double)(hei-my)/(double)hei;
    x=x*2.0-1.0;
    y=y*2.0-1.0;

    lastClick[0].Set(x,y,-1.0);
    lastClick[1].Set(x,y, 1.0);
    for(auto &p : lastClick)
    {
        drawEnv.GetProjectionMatrix().MulInverse(p,p,1.0);
        drawEnv.GetViewMatrix().MulInverse(p,p,1.0);
    }
}
```

5. In Draw function,

```
{
    YsGLSLPlain3DRenderer renderer;  // Again, do not nest the renderer!
    renderer.SetProjection(projMat);
    renderer.SetModelView(viewMat);

    GLfloat color[8]={0,0,1,1, 1,0,0,1};
    const GLfloat vtx[6]=
    {
        lastClick[0].xf(),lastClick[0].yf(),lastClick[0].zf(),
        lastClick[1].xf(),lastClick[1].yf(),lastClick[1].zf()
    };
    renderer.DrawVtxCol(GL_LINES,2,vtx,color);
}
```

Picking

- Change color of the picked polygon.
- What needs to be done when the user clicks on the left button:
  1. Calculate a line from the mouse coordinate.
  2. Find the polygon that:
     - intersects with the line, and
     - in front of the camera, and
     - nearest to the view point.
  3. Change the color of the polygon.

Intersection of a line and a polygon

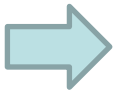
- Plane-line intersection.
- The intersecting point is on the boundary or inside of the polygon. (YsCheckInsidePolygon3)

1. Copy project from mouse-to-line, rename TARGET_NAME as glsl3d_picking
2. Add a function:

```
void FsLazyWindowApplication::MouseCoordinateTo3DLine(YsVec3 ln[2],int mx,int my) const
{
    int wid,hei;
    FsGetWindowSize(wid,hei);

    double x=(double)mx/(double)wid;
    double y=(double)(hei-my)/(double)hei;
    x=x*2.0-1.0;
    y=y*2.0-1.0;

    ln[0].Set(x,y,-1.0);
    ln[1].Set(x,y, 1.0);
    for(int i=0; i<2; ++i)
    {
        auto &p=ln[i];
        drawEnv.GetProjectionMatrix().MulInverse(p,p,1.0);
        drawEnv.GetViewMatrix().MulInverse(p,p,1.0);
    }
}
```

3. Replace:

```
if(evt==FSMOUSEEVENT_LBUTTONDOWN)
{
    double x=(double)mx/(double)wid;
    double y=(double)(hei-my)/(double)hei;
    x=x*2.0-1.0;
    y=y*2.0-1.0;

    lastClick[0].Set(x,y,-1.0);
    lastClick[1].Set(x,y, 1.0);
    for(auto &p : lastClick)
    {
        drawEnv.GetProjectionMatrix().MulInverse(p,p,1.0);
        drawEnv.GetViewMatrix().MulInverse(p,p,1.0);
    }
}
```



```
if(evt==FSMOUSEEVENT_LBUTTONDOWN)
{
    MouseCoordinateTo3DLine(lastClick,mx,my);
}
```

4. Add member variable:

```
std::vector <float> col;
```

5. In LoadBinaryStl, after calling ::LoadBinaryStl,

```
col.clear();
for(int i=0; i<vtx.size()/3; ++i)
{
    col.push_back(0);
    col.push_back(0);
    col.push_back(1);
    col.push_back(1);
}
```

6. In Draw function, replace:

```
GLfloat color[4]={0,0,1,1};
renderer.SetUniformColor(color);
renderer.DrawVtxNom(GL_TRIANGLES,vtx.size()/3,vtx.data(),nom.data());
```

```
renderer.DrawVtxNomCol(GL_TRIANGLES,vtx.size()/3,vtx.data(),nom.data(),col.data());
```

7. Add a function:

```
int FsLazyWindowApplication::PickedTriangle(int mx,int my) const
{
    YsVec3 ln[2];
    MouseCoordinateTo3DLine(ln,mx,my);

    const YsVec3 o=ln[0];
    const YsVec3 v=YsUnitVector(ln[1]-ln[0]);

    int picked=-1;
    double pickedDist=0.0;
    for(int i=0; i<vtx.size()/9; ++i)
    {
        const YsVec3 tri[3]=
        {
            YsVec3(vtx[i*9  ],vtx[i*9+1],vtx[i*9+2]),
            YsVec3(vtx[i*9+3],vtx[i*9+4],vtx[i*9+5]),
            YsVec3(vtx[i*9+6],vtx[i*9+7],vtx[i*9+8]),
        };
        YsPlane pln;
        pln.MakePlaneFromTriangle(tri[0],tri[1],tri[2]);

        YsVec3 itsc;
        if(YSOK==pln.GetIntersection(itsc,o,v))
        {
            auto side=YsCheckInsideTriangle3(itsc,tri);
            if(YSINSIDE==side || YSBOUNDARY==side)
            {
                auto dist=(itsc-o)*v; // Gives distance
                if(0.0<dist && (picked<0 || dist<pickedDist))
                {
                    picked=i;
                    pickedDist=dist;
                }
            }
        }
    }

    return picked;
}
```

8. Modify event handling for FSMOUSEEVENT_LBUTTONDOWN as:

```
if(evt==FSMOUSEEVENT_LBUTTONDOWN)
{
    MouseCoordinateTo3DLine(lastClick,mx,my);

    int triIdx=PickedTriangle(mx,my);
    if(0<=triIdx)
    {
        for(int i=0; i<3; ++i)
        {
            col[((3*triIdx)+i)*4  ]=1;
            col[((3*triIdx)+i)*4+1]=0;
            col[((3*triIdx)+i)*4+2]=0;
        }
    }
}
```