Lecture 22

Password: shrimpFriedRice

- GUI programming
- GUI application template
  - Canvas
  - Main Menu
  - View-Control Dialog
- Adding menu items
- Binding call-back functions
- Modal Dialog
  - Message Dialog
  - Taking a number input
    - Blocking and non-blocking modal dialog
    - Re-entrancy problem
    - Dialog closure as an event
- Using a file-dialog

GUI programming

- It is more of the event-driven programming.
- You need to add ‘widgets’ and connect call-back functions.

- Widget
  - A unit of GUI.
  - Menu, dialog, button, list-box, text-box, etc.

GUI Application Template

- GUI library: FsGuiLib, FsGui3d
- A GUI+OpenGL toolkit for MacOSX, Linux, Windows, and iOS.
- Draws all widgets with OpenGL.
- Theoretically portable to any platforms that supports OpenGL and C++11.
- It is small and does not require massive DLLs.
- Everything is contained in the source files under public directory. No external dependencies.

- Open source with BSD license.

GUI Application template

- That was about time when I was to look into Qt, when Nokia ditched it. (Lucky me! I didn't waste my precious time for learning Tcl/Tk.)
- I had enough of it.
- I decided to write my own. At least I can use it as long as I and C++ and OpenGL are alive
- I started from something like I showed in the previous lecture, and added features.

GUI Application Template

Using FsGui3D template:

1. Copy the template by:
   svn export ../public/src/fsgui3d/template new_project_name
2. Change TARGET_NAME in the copied CMakeLists.txt
3. Add the project directory to the top-level CMakeLists.txt

- Then, compile and run the program.

FsGUI3D template

By default, the template includes main menu, canvas, and view-control dialog.

- Main Menu

  Provides a conventional pull-down menu.

- Canvas

  The area for drawing 2D and 3D graphics. By default, the user can rotate the view by Shift+Left button, move by Shift+Right button, Zoom/Unzoom by Shift+Left+Right buttons.

- View-Control Dialog

  Provides a basic view-control.

Adding menu items

- Let’s add menu items.
- You need to do four things:
  - Add a menu item.
  - Write event handler prototype in the application class.
  - Implement the event handler.
  - Connect the menu item and the event handler.s

Adding menu items

- Main menu is constructed in a function called MakeMainMenu in fsgui3dapp.cpp
- MakeMainMenu function is called from the constructor of FsGui3DMainCanvas.
- The sequence will be:
  1. The application starts.
  2. Low-level toolkit creates the window and OpenGL context.
  3. First call to FsGui3DMainCanvas::GetMainCanvas(), where FsGui3DMainCanvas is created.
     1. Constructor of FsGui3DMainCanvas is called.
        - The constructor calls MakeMainMenu
     2. Low-level toolkit calls FsGui3DMainCanvas::Initialize
  4. Event-loop in the low-level toolkit starts.

Adding menu items

- Main menu is a member variable of FsGui3DMainCanvas

  FsGuiPopUpMenu *mainMenu;

- Find MakeMainMenu function in fsgui3dapp.cpp, and

```
void FsGui3DMainCanvas::MakeMainMenu(void)
{
    mainMenu=new FsGuiPopUpMenu;
    mainMenu->Initialize();
    mainMenu->SetIsPullDownMenu(YSTRUE);

    FsGuiPopUpMenuItem *fileMenu=mainMenu->AddTextItem(0,FSKEY_F,L"File");
    FsGuiPopUpMenu *fileSubMenu=fileMenu->GetSubMenu();

    fileSubMenu->AddTextItem(0,FSKEY_X,L"Exit")->BindCallBack(&THISCLASS::File_Exit,this);


    auto viewSubMenu=mainMenu->AddTextItem(0,FSKEY_V,L"View")->GetSubMenu();
    viewSubMenu->AddTextItem(0,FSKEY_NULL,L"Red Background");
    viewSubMenu->AddTextItem(0,FSKEY_NULL,L"Green Background");
    viewSubMenu->AddTextItem(0,FSKEY_NULL,L"Blue Background");
    viewSubMenu->AddTextItem(0,FSKEY_NULL,L"White Background");


    SetMainMenu(mainMenu);
}
```

Parameters to AddTextItem function

- The first parameter is an integer number. Optionally, you can assign an identifier to each menu.
- Now you can directly bind a call-back function to the menu item with std::function, and the identifier is rarely used.
- The second parameter is a key code for menu short cut.

Binding call-back functions

- With these additional lines, you see View menu and some menu items.
- Now you can define behaviors for these menus.
- You need to add core-data structures and call-back functions.
- Core-data structure

  In this case, core-data structure is the background color.

  ```
  GLfloat bgColor[4];
  ```

- Call-back functions

  ```
  void View_SetRedBackground(FsGuiPopUpMenuItem *);
  void View_SetBlueBackground(FsGuiPopUpMenuItem *);
  void View_SetGreenBackground(FsGuiPopUpMenuItem *);
  void View_SetWhiteBackground(FsGuiPopUpMenuItem *);
  ```

Call-back functions

```
void FsGui3DMainCanvas::View_SetRedBackground(FsGuiPopUpMenuItem *)
{
   bgColor[0]=1;
   bgColor[1]=0;
   bgColor[2]=0;
   bgColor[3]=1;
}
void FsGui3DMainCanvas::View_SetBlueBackground(FsGuiPopUpMenuItem *)
{
   bgColor[0]=0;
   bgColor[1]=1;
   bgColor[2]=0;
   bgColor[3]=1;
}
void FsGui3DMainCanvas::View_SetGreenBackground(FsGuiPopUpMenuItem *)
{
   bgColor[0]=0;
   bgColor[1]=0;
   bgColor[2]=1;
   bgColor[3]=1;
}
void FsGui3DMainCanvas::View_SetWhiteBackground(FsGuiPopUpMenuItem *)
{
   bgColor[0]=1;
   bgColor[1]=1;
   bgColor[2]=1;
   bgColor[3]=1;
}
```

Change in the Draw function

- Also add the following line before glClear in Draw function:

```
glClearColor(bgColor[0],bgColor[1],bgColor[2],bgColor[3]);
```

Binding call-back functions

- Ready to bind the call-back function.
- A call-back function can be bound by BindCallBack member function defined in fsguipopupmenu.h

```
template <typename objType>
void BindCallBack(void (objType::*func)(FsGuiPopUpMenuItem *),objType *obj)
{
    stdCallBack=std::bind(func,obj,this);
}
```

- Since it is a template function, it can bind any member function of a class *objType* that takes a pointer to FsGuiPopUpMenuItem *. (Advantage of C++11 std::function)

Binding call-back functions

- In MakeMainMenu function,

```
auto viewSubMenu=mainMenu->AddTextItem(0,FSKEY_V,L"View")->GetSubMenu();
viewSubMenu->AddTextItem(0,FSKEY_NULL,L"Red Background")->
    BindCallBack(&THISCLASS::View_SetRedBackground,this);
viewSubMenu->AddTextItem(0,FSKEY_NULL,L"Green Background")->
    BindCallBack(&THISCLASS::View_SetGreenBackground,this);
viewSubMenu->AddTextItem(0,FSKEY_NULL,L"Blue Background")->
    BindCallBack(&THISCLASS::View_SetBlueBackground,this);
viewSubMenu->AddTextItem(0,FSKEY_NULL,L"White Background")->
    BindCallBack(&THISCLASS::View_SetWhiteBackground,this);
```

Message Dialog

- Message dialog is used for showing notifications, errors, and various messages to the user.
- Typically a modal dialog.

- Modal and Modeless dialog:
  - Modal dialog

    While the modal dialog is open, all events are sent to the top-most modal dialog. Other widgets won't receive key strokes, mouse events, etc.
  - Modeless dialog

    Events are sent to all widgets on the canvas. The user can click on the buttons and menus while modeless dialogs are open. Default view-control dialog is a modeless dialog.

Message box test

- Let's add a new menu item in File menu, and open a modal message dialog from there.

```
void FsGui3DMainCanvas::File_Message(FsGuiPopUpMenuItem *)
{
    auto dlg=FsGuiDialog::CreateSelfDestructiveDialog<FsGuiMessageBoxDialog>();
    dlg->Make(L"Test Message",L"This is a test message",L"OK",nullptr);
    AttachModalDialog(dlg);
}
```

- In this case, the message dialog is no longer needed after the user clicks on the OK button.
- By creating it as a self-destructive dialog, it deletes itself when it is no longer needed.

Taking a number input from a modal dialog

- Blocking and non-blocking modal dialog
- Re-entrancy problem
- Dialog closure as an event.

Is AttachModal dialog blocking?

- Let's add a printf after AttachModal dialog, and see if it is blocking or non-blocking.

Caution about the modal-ness

- Some toolkits provides with a blocking message box.
- Example: MessageBox function in Win32 API.
- The function will not return until the user clicks on the OK or Cancel button.

- Modern toolkits like Cocoa opens a modal dialog, but the function returns immediately without waiting for the user clicking on the OK or Cancel button.
- In this case, the dialog blocks events sent to other widgets, but your program is not blocking.

Blocking modal dialog and re-entrancy problem

- Non-blocking modal dialog has less probability of *re-entrancy* problem.
- Re-entrancy: When you run an event-loop in a call-back function, like Interval to wait for an event, the same call-back function may be called from the event loop. It is called a re-entrancy.
- Unless you write your call-back function safe for the re-entrancy, you may see an unintended result.
- The source of the re-entrancy is typically an event-loop running inside a call-back function. Therefore it can be avoided by not blocking inside an event call-back function.

Non-blocking modal dialog and close-modal call back

- You avoid re-entrancy by not blocking in the modal dialog, but what if you need to do something when the user clicks on the OK or Cancel button?
- Modal-dialog closure needs to be considered an event.
- Add one more function for receiving a dialog-closure event when the user clicks on OK or Cancel button.

Input-number dialog

- Let's add a new menu item and open an input-number dialog from there.

```
void FsGui3DMainCanvas::File_Input_Number_Test(FsGuiPopUpMenuItem *)
{
   auto dlg=FsGuiDialog::CreateSelfDestructiveDialog <FsGuiInputNumberDialog>();
   dlg->Make(0.0,4,L"Test number input",L"Test number input",L"Number",L"OK",L"Cancel");
   dlg->BindCloseModalCallBack(
      &FsGui3DMainCanvas::File_Input_Number_Test_CloseDialogCallBack,this);
   AttachModalDialog(dlg);
}

void FsGui3DMainCanvas::File_Input_Number_Test_CloseDialogCallBack(
      FsGuiDialog *dlg,int returnCode)
{
   auto numDlg=dynamic_cast<FsGuiInputNumberDialog *>(dlg);
   if(nullptr!=numDlg && returnCode==(int)YSOK)
   {
      auto dlg=FsGuiDialog::CreateSelfDestructiveDialog<FsGuiMessageBoxDialog>();

      YsString str;
      str.Printf("%.4lf",numDlg->GetNumber());
      YsWString msg;
      msg.SetUTF8String(str);

      dlg->Make(msg,msg,L"OK",nullptr);
      AttachModalDialog(dlg);
   }
}
```

File-Dialog

- Unfortunately not supported in iOS. iOS is a closed environment. Apple is making sure it is very difficult for anyone outside Apple to write something useful.
- Same idea as the input-number dialog.
- It uses the system-standard file dialog in Windows and MacOSX. It uses its own in Linux.

```
#include <stdio.h>

#include <ysport.h>

#include <fsguifiledialog.h>
#include "fsgui3dapp.h"

void FsGui3DMainCanvas::File_Open(FsGuiPopUpMenuItem *)
{
    printf("%s %d\n",__FUNCTION__,__LINE__);

    auto fdlg=FsGuiDialog::CreateSelfDestructiveDialog <FsGuiFileDialog>();
    fdlg->Initialize();
    fdlg->mode=FsGuiFileDialog::MODE_SAVE;
    fdlg->multiSelect=YSFALSE;
    fdlg->title.Set(L"Open a mesh");
    fdlg->fileExtensionArray.Append(L".stl");
    fdlg->fileExtensionArray.Append(L".srf");
    fdlg->fileExtensionArray.Append(L".sff");
    fdlg->fileExtensionArray.Append(L".obj");
    fdlg->defaultFileName=L"./*.stl";
    fdlg->SetCloseModalCallBack(NULL);
    fdlg->BindCloseModalCallBack(&THISCLASS::File_Open_OnCloseDialog,this);
    this->AttachModalDialog(fdlg);
}
```

```
void FsGui3DMainCanvas::File_Open_OnCloseDialog(FsGuiDialog *dlg,int returnCode)
{
    auto fdlg=dynamic_cast <FsGuiFileDialog *>(dlg);
    if(nullptr!=fdlg && (int)YSOK==returnCode)
    {
        auto fn=fdlg->selectedFileArray[0];

        YsString sysEnc;
        YsUnicodeToSystemEncoding(sysEnc,fn);

        printf("%s\n",(const char *)sysEnc);
    }
}
```

Make it a surface mesh viewer

- Bring some functions and classes from the STL viewer (Phong shading) example.
- Then render it with YsGLSLShared3DRenderer

Make Vertex selectable

- Copy PickedVertex function from greedy_path_finding example.
- Add:

```
std::vector <YsShell::VertexHandle> selectedVertex;
std::vector <float> selVtxBuf;
void RemakeSelectedVertexBuffer(void);
```

- In LoadModel():

```
selectedVertex.clear();
selVtxBuffer.clear();
```

- Add the following 3 functions:

```
void FsGui3DMainCanvas::Select_UnselectAll(FsGuiPopUpMenuItem *)
void FsGui3DMainCanvas::Select_Vertex(FsGuiPopUpMenuItem *)
void FsGui3DMainCanvas::Select_Vertex_LButtonDown(FsGuiMouseButtonSet,YsVec2i))
```

- Add select menu.
- Draw selected vertices.

```
void FsGui3DMainCanvas::RemakeSelectedVertexBuffer(void)
{
    selVtxBuffer.clear();
    for(auto vtHd : selectedVertex)
    {
        auto pos=shl.GetVertexPosition(vtHd);
        selVtxBuffer.push_back(pos.xf());
        selVtxBuffer.push_back(pos.yf());
        selVtxBuffer.push_back(pos.zf());
    }
}
```

```
YsShell::VertexHandle FsGui3DMainCanvas::PickedVtHd(int mx,int my) const
{
    int wid,hei;
    FsGetWindowSize(wid,hei);

    double pickedZ=YsInfinity;
    YsShell::VertexHandle pickedVtHd=nullptr;
    for(auto vtHd : shl.AllVertex())
    {
        YsVec3 pos=shl.GetVertexPosition(vtHd);
        drawEnv.GetViewMatrix().Mul(pos,pos,1.0);
        drawEnv.GetProjectionMatrix().Mul(pos,pos,1.0);
        if(-1.0<=pos.z() && pos.z()<=1.0)
        {
            const double u=(pos.x()+1.0)/2.0;
            const double v=(pos.y()+1.0)/2.0;

            int x=(int)((double)wid*u);
            int y=hei-(int)((double)hei*v);
            if(mx-8<=x && x<=mx+8 && my-8<=y && y<=my+8)
            {
                if(nullptr==pickedVtHd || pos.z()<pickedZ)
                {
                    pickedVtHd=vtHd;
                    pickedZ=pos.z();
                }
            }
        }
    }

    return pickedVtHd;
}
```

```
void FsGui3DMainCanvas::Select_UnselectAll(FsGuiPopUpMenuItem *)
{
    selectedVertex.clear();
    RemakeSelectedVertexBuffer();
}

void FsGui3DMainCanvas::Select_Vertex(FsGuiPopUpMenuItem *)
{
    BindLButtonDownCallBack(&THISCLASS::Select_Vertex_LButtonDown,this);
}
YSRESULT FsGui3DMainCanvas::Select_Vertex_LButtonDown(FsGuiMouseButtonSet,YsVec2i mos)
{
    auto vtHd=PickedVtHd(mos.x(),mos.y());
    if(nullptr!=vtHd)
    {
        for(auto v : selectedVertex)
        {
            if(v==vtHd)
            {
                return YSOK;
            }
        }
        selectedVertex.push_back(vtHd);
        RemakeSelectedVertexBuffer();
        SetNeedRedraw(YSTRUE);
    }
    return YSOK;
}
```

- In Draw()

```
{
    YsGLSLPlain3DRenderer renderer;
    renderer.SetUniformPointSize(8.0f);
    glEnable(GL_PROGRAM_POINT_SIZE);
    GLfloat magenta[]={1,0,1,1};
    renderer.SetZOffset(-0.0001);
    renderer.SetUniformColor(magenta);
    renderer.DrawVtx(GL_POINTS,selVtxBuffer.size()/3,selVtxBuffer.data());
    renderer.SetZOffset(0);
    glDisable(GL_PROGRAM_POINT_SIZE);
}
```

Adding view options

- Let’s add the two view options.
  - Draw Wireframe
  - Lighting

- Add the following member variables:

  std::vector <float> wireVtx;

  FsGuiPopUpMenuItem *wireframe;

  FsGuiPopUpMenuItem *lighting;

- Change YsShellExtToVtxNom to non-static RemakeVertexAttribBuffer, and also populate wireVtx in the function.

```
void FsGui3DMainCanvas::RemakeVertexAttribArray(void)
{
    vtx.clear();    nom.clear();    col.clear();
    wireVtx.clear();
    for(auto plHd : shl.AllPolygon())
    {
        auto plVtHd=shl.GetPolygonVertex(plHd);
        if(3<=plVtHd.GetN())
        {
            auto plNom=shl.GetNormal(plHd);
            auto plCol=shl.GetColor(plHd);
            for(auto vtHd : plVtHd)
            {
                auto vtPos=shl.GetVertexPosition(vtHd);
                vtx.push_back(vtPos.xf()); vtx.push_back(vtPos.yf()); vtx.push_back(vtPos.zf());
                nom.push_back(plNom.xf()); nom.push_back(plNom.yf()); nom.push_back(plNom.zf());
                col.push_back(plCol.Rf()); col.push_back(plCol.Gf()); col.push_back(plCol.Bf());
                col.push_back(1);
            }
            for(int i=0; i<plVtHd.GetN(); ++i)
            {
                YsVec3 vtPos[2]=
                {
                    shl.GetVertexPosition(plVtHd[i]),
                    shl.GetVertexPosition(plVtHd.GetCyclic(i+1))
                };
                wireVtx.push_back(vtPos[0].xf()); wireVtx.push_back(vtPos[0].yf());
                wireVtx.push_back(vtPos[0].zf());
                wireVtx.push_back(vtPos[1].xf()); wireVtx.push_back(vtPos[1].yf());
                wireVtx.push_back(vtPos[1].zf());
            }
        }
    }
}
```

- Add two new menus in MakeMainMenu

```
wireframe=viewSubMenu->AddTextItem(0,FSKEY_NULL,L"Draw Wireframe");
wireframe->SetCheck(YSTRUE);
wireframe->BindCallBack(&THISCLASS::FlipMenuCheck,this);
lighting=viewSubMenu->AddTextItem(0,FSKEY_NULL,L"Lighting");
lighting->SetCheck(YSTRUE);
lighting->BindCallBack(&THISCLASS::FlipMenuCheck,this);
```

- Add FlipMenuCheck function

```
void FsGui3DMainCanvas::FlipMenuCheck(FsGuiPopUpMenuItem *menu)
{
    auto check=menu->GetCheck();
    YsFlip(check);
    menu->SetCheck(check);
    SetNeedRedraw(YSTRUE);
}
```

- In Draw function:

```
if(YSTRUE==lighting->GetCheck())
{
    YsGLSLShaded3DRenderer renderer;

    GLfloat lightDir[3]={0,0,1};
    renderer.SetLightDirectionInCameraCoordinate(0,lightDir);
    renderer.DrawVtxNomCol(GL_TRIANGLES,vtx.size()/3,vtx.data(),nom.data(),col.data());
}
else
{
    YsGLSLPlain3DRenderer renderer;
    renderer.DrawVtxCol(GL_TRIANGLES,vtx.size()/3,vtx.data(),col.data());
}

{
    YsGLSLPlain3DRenderer renderer;
    renderer.SetUniformPointSize(8.0f);
    glEnable(GL_PROGRAM_POINT_SIZE);
    GLfloat magenta[]={1,0,1,1};
    renderer.SetZOffset(-0.0001);
    renderer.SetUniformColor(magenta);
    renderer.DrawVtx(GL_POINTS,selVtxBuffer.size()/3,selVtxBuffer.data());
    glDisable(GL_PROGRAM_POINT_SIZE);
    if(YSTRUE==wireframe->GetCheck())
    {
        GLfloat black[]={0,0,0,1};
        renderer.SetUniformColor(black);
        renderer.DrawVtx(GL_LINES,wireVtx.size()/3,wireVtx.data());
    }
    renderer.SetZOffset(0);
}
```