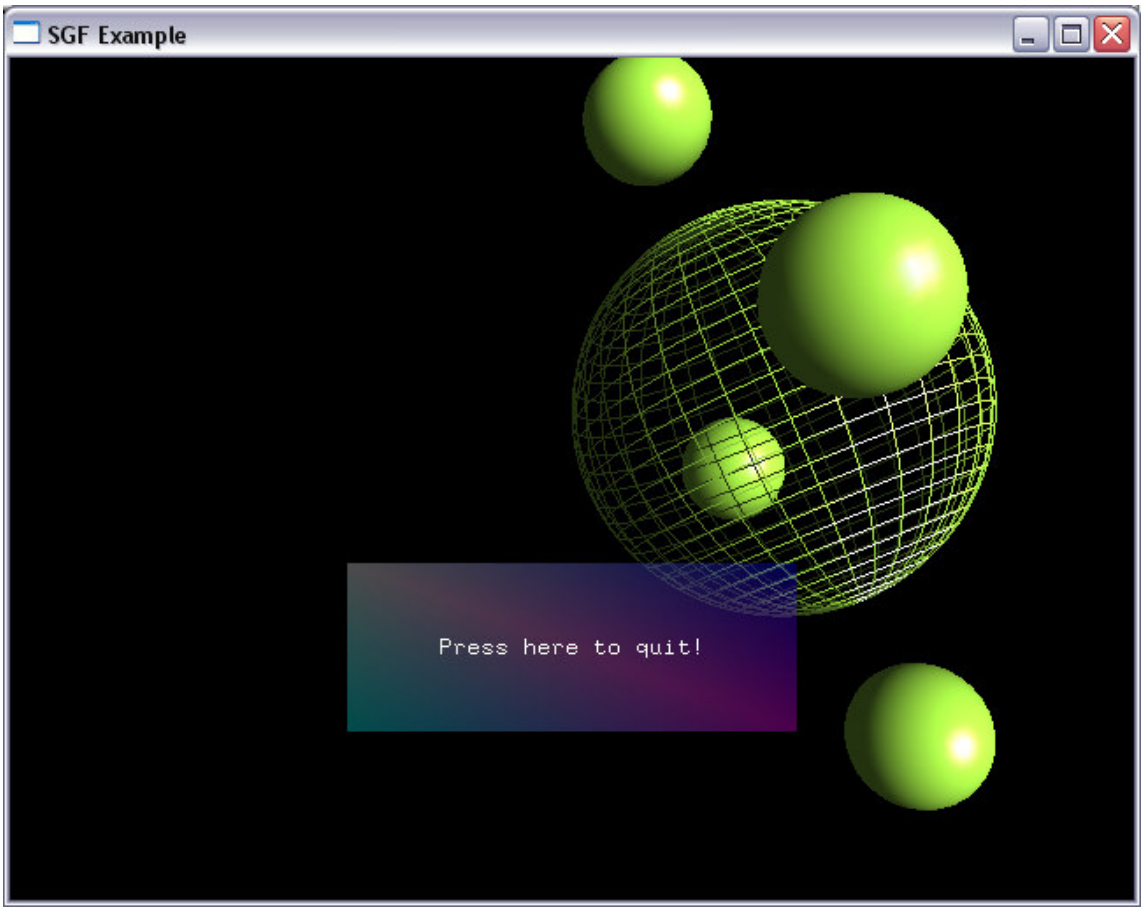
Tutorial

Denna del innehåller en tutorial där vi använder SGF och freeglut(Free OpenGL Utility Toolkit) för att göra ett enkelt GUI i OpenGL. Freeglut finns att hämta på http://freeglut.sourceforge.net/



Figur 15. Skärmbild från exempelprogrammet SGFExample.

Under tutorialens gång kommer vi definiera ett par klasser, GLBackground och GLDialog. För att underlätta skapandet av programmet används följande färdigskrivna kodskelett som definierar och sätter upp vårat fönster och funktioner:

```
#include "GL/freeglut.h"
#include "SGF.h"
#include "exampleutil.h"

/* FILE-SCOPE ÌNSTANCES */
/* FILE-SCOPE FUNCTIONS */
static void initialize() {      //This is where we create our GUI

}
static void finalize() {        //This is where we delete our GUI
}
static void render() {                      //This is where we draw our GUI
          ExampleUtil::setupFor3D();
          ExampleUtil::draw3DBackground();
          ExampleUtil::setupFor2D();
```

```
          //Draw your GUI here

     glutSwapBuffers();
}
static void motion(int x, int y) {//This is where we tell our GUI about
mouse movement
}
//This is where we tell our GUI about button-presses
static void mouse(int button, int state, int x, int y) {
}

int main(int argc, char argv[]) {
          glutInit(&argc, &argv);
          glutInitWindowSize(640, 480);
          glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE);
          glutCreateWindow("SGF Example");
          ExampleUtil::initialize(640, 480);
          initialize();
          glutReshapeFunc(ExampleUtil::resize);
          glutDisplayFunc(render);
          glutIdleFunc(render);
          glutMouseFunc(mouse);
          glutPassiveMotionFunc(motion);
          glutMotionFunc(motion);
          glutMainLoop();
          finalize();
}
```

För att simplifiera exempelprogrammet ytterliggare och fokusera på användningen av SGF används ett hjälpverktyg, ExampleUtil. Detta verktyg innehåller ett antal hjälpfunktioner för initializering och utritning:

```
namespace ExampleUtil {
          void draw3DBackground();         //Draw fancy spheres
          void drawCenteredText(float x, float y, const std::string&
text);
          void drawExampleRectangle(float x, float y, float width, float
height, float alpha);
          void initialize(int x, int y);
          void resize(int x, int y);
          int screenWidth();
          int screenHeight();
          void setupFor3D();    //Prepare for 3D
          void setupFor2D();    //Prepare for 2D
          float textWidth(const std::string& text);
          float textHeight(const std::string& text);
}
```

Med denna kodgrund visar programmet exempelsfärerna roterandes på skärmen. Vi tar och skapar och lägger till vårat gui och dess root i initialize:

```
/* FILE-SCOPE INSTANCES */
static SGFWindow* rootWindow;
static SGFSystem* gui;

/* FILE-SCOPE FUNCTIONS */
static void initialize() {
          //Create our GUI and its root
          rootWindow = new SGFWindow(SGFRectangle(SGFPosition(0.0f, 0.0f),
1.0f, 1.0f));
          gui = new SGFSystem(rootWindow);
}
```

Vi ser även till att minnet deallokeras i finalize:

```
static void finalize() {
          delete rootWindow;
          delete gui;
}
```

Eftersom vi nu har definierat root-fönstrets storlek som en rektangel vars övre vänsta hörn befinner sig i koordinat 0, 0 ,ed dess bredd och höjd satt till 1 använder vi hädanefter alltid detta koordinatsystem för komponenterna i vårat GUI. Vi kan se till att GUI't får korrekta meddelanden genom att dividera musens skärmposition med skärmens bredd resp. höjd:

```
static void motion(int x, int y) {
     SGFPosition p(x / (float) ExampleUtil::screenWidth(), y / (float)
     ExampleUtil::screenHeight());
     gui->postPosition(p);
}

static void mouse(int button, int state, int x, int y) {
     SGFPosition p(x / (float) ExampleUtil::screenWidth(), y / (float)
     ExampleUtil::screenHeight());

     if(state == GLUT_DOWN)
          gui->postButtonDown(button, p);
     else if(state == GLUT_UP)
          gui->postButtonUp(button, p);
}
```

Vi ser till att GUI't ritar ut sig genom att skicka utritningsmeddelandet i funktionen render:

```
static void render() {
          ExampleUtil::setupFor3D();
          ExampleUtil::draw3DBackground();
          ExampleUtil::setupFor2D();

          //draw your gui here
          gui->postDraw();

          glutSwapBuffers();
}
```

Vi fortsätter nu med att skriva våran första Gui-komponent, GLLabel. Dess syfte är att visa en textsträng på skärmen vid utritning. Grunden för klassens ser ut som följande:

```
class GLLabel : public SGFComponent {
          std::string mCaption;
public:
          GLLabel(const SGFPosition& position, const std::string& caption)
:
           SGFComponent(SGFRectangle(position, SGFAlignment::CENTER,
           ExampleUtil::textWidth(caption),
           ExampleUtil::textHeight(caption))),
                     mCaption(caption) {
          }
};
```

GLLabels konstruktor tar en position och en sträng som inparametrar och skapar en SGFRectangle som används för att initializera dess basklass SGFComponent. Den sparar även undan textsträngen för användning vid utritning senare. För att implementera utritningen överlagrar GLLabel SGFComponents onDraw-meddelande:

```
virtual void onDraw(const SGFPosition& position) {
          const SGFPosition p = position + getBounds().getPosition();
```

```
          const float width = getBounds().getWidth();
          const float height = getBounds().getHeight();
     ExampleUtil::drawCenteredText(p.getX() + width * 0.5f, p.getY() +
 height * 0.5f, mCaption);
}
```

För att rita texten i mitten av komponenten beräknar GLLabel sin mittposition genom att först addera referenspositionen som ges av onDraw-meddelandet med sin egen position, och därefter med hälften av sin storlek.

Den andra komponenten vi kommer implementera är GLButton vars uppgift är att rita ut sin tryckyta och stänga av programmet när användaren klickar på den. För att göra knappen lite vackrare implementeras varierar den sin genomskinlighet beroende på om användaren håller musen ovanför den eller inte.

```
 class GLButton : public SGFComponent {
          float mAlpha;
public:
          GLButton(const SGFRectangle& bounds) :
                    SGFComponent(bounds),
                    mAlpha(0.3f) {
            }
            virtual void onFocus() {
                      mAlpha = 0.6f;
            }
            virtual void onUnFocus() {
                      mAlpha = 0.3f;
            }
 };
```

Genom att använda ExampleUtils funktion drawExampleRectangle kan vi enkelt implementera onDraw-meddelandet:

```
 virtual void onDraw(const SGFPosition& position) {
          const SGFPosition p = position + getBounds().getPosition();
          const float width = getBounds().getWidth();
          const float height = getBounds().getHeight();
     ExampleUtil::drawExampleRectangle(p.getX(), p.getY(), width, height,
 mAlpha);
 }
```

Slutligen implementerar vi GLButtons onButtonDown meddelande genom att be freeglut att stänga av programmet.

```
   virtual void onButtonDown(int button, const SGFPosition& position) {
             glutLeaveMainLoop();
   }
```

Genom att utöka programmet med dessa klasser kan vi nu skapa våran avslutningsdialog. Det vi behöver lägga till till programmet är markerat i fetstil:

```
 /* FILE-SCOPE INSTANCES */
 static SGFWindow* rootWindow;
 static SGFSystem* gui;
 static SGFWindow* quitDialog;
 static GLButton* quitButton;
 static GLLabel* quitLabel;

 /* FILE-SCOPE FUNCTIONS */
 static void initialize() {
          //create our gui and its root
```

```
          rootWindow = new SGFWindow(SGFRectangle(SGFPosition(0.0f, 0.0f),
1.0f, 1.0f));
          gui = new SGFSystem(rootWindow);

          //create our quit dialog
     quitDialog = new SGFWindow(SGFRectangle(SGFPosition(0.5f, 0.7f),
     SGFAlignment(0.5f, 0.5f), 0.4f, 0.2f));

          //create the button and add it to the dialog
          quitButton= new GLButton(SGFRectangle(SGFPosition(0.0f, 0.0f),
0.4f, 0.2f));
          quitDialog->addComponent(quitButton);

          //create the informative label and add it to the dialog
          quitLabel = new GLLabel(SGFPosition(0.2f, 0.1f), "Press here to
quit!");
          quitDialog->addComponent(quitLabel);

          //finally add our dialog to the gui root
          rootWindow->addComponent(quitDialog);
}

static void finalize() {
          delete quitDialog;
          delete quitButton;
          delete quitLabel;
          delete rootWindow;
          delete gui;
}
```

Och nu är vårat testprogram färdigt! Hela koden för SGFExample följer här under (för källkoden till exampleutil kolla se bilaga)

```
#include "GL/freeglut.h"
#include "SGF.h"
#include "exampleutil.h"

class GLButton : public SGFComponent {
          float mAlpha;
public:
          GLButton(const SGFRectangle& bounds) :
                    SGFComponent(bounds),
                    mAlpha(0.3f) {
            }
            virtual void onButtonDown(int button, const SGFPosition&
position) {
                      glutLeaveMainLoop();
            }
            virtual void onFocus() {
                      mAlpha = 0.6f;
            }
            virtual void onUnFocus() {
                      mAlpha = 0.3f;
            }
          virtual void onDraw(const SGFPosition& position) {
                    const SGFPosition p = position +
getBounds().getPosition();
                    const float width = getBounds().getWidth();
                    const float height = getBounds().getHeight();

                    ExampleUtil::drawExampleRectangle(p.getX(), p.getY(),
width, height, mAlpha);
```

```
        }
};

class GLLabel : public SGFComponent {
        std::string mCaption;
public:
        GLLabel(const SGFPosition& position, const std::string& caption)
:
          SGFComponent(SGFRectangle(position, SGFAlignment::CENTER,
ExampleUtil::textWidth(caption), ExampleUtil::textHeight(caption))),
          mCaption(caption) {
        }
        virtual void onDraw(const SGFPosition& position) {
                const SGFPosition p = position +
getBounds().getPosition();
                const float width = getBounds().getWidth();
                const float height = getBounds().getHeight();
                ExampleUtil::drawCenteredText(p.getX() + width *
0.5f, p.getY() + height * 0.5f, mCaption);
        }
};

/* FILE-SCOPE INSTANCES */
static SGFWindow* rootWindow;
static SGFSystem* gui;
static SGFWindow* quitDialog;
static GLButton* quitButton;
static GLLabel* quitLabel;

/* FILE-SCOPE FUNCTIONS */
static void initialize() {
        //Create our GUI and its root
        rootWindow = new SGFWindow(SGFRectangle(SGFPosition(0.0f, 0.0f),
1.0f, 1.0f));
        gui = new SGFSystem(rootWindow);

        //Create our quit dialog
        quitDialog = new SGFWindow(SGFRectangle(SGFPosition(0.5f, 0.7f),
SGFAlignment(0.5f, 0.5f), 0.4f, 0.2f));

        //Create the button and add it to the dialog
        quitButton= new GLButton(SGFRectangle(SGFPosition(0.0f, 0.0f),
0.4f, 0.2f));
        quitDialog->addComponent(quitButton);

        //Create the informative label and add it to the dialog
        quitLabel = new GLLabel(SGFPosition(0.2f, 0.1f), "Press here to
quit!");
        quitDialog->addComponent(quitLabel);

        //Finally, add our dialog to the GUI root
        rootWindow->addComponent(quitDialog);
}

static void finalize() {
        delete quitDialog;
        delete quitButton;
        delete quitLabel;
        delete rootWindow;
        delete gui;
}
```

```
static void render() {
          ExampleUtil::setupFor3D();
          ExampleUtil::draw3DBackground();
          ExampleUtil::setupFor2D();
          gui->postDraw();
          glutSwapBuffers();
}

static void motion(int x, int y) {
          SGFPosition p(x / (float) ExampleUtil::screenWidth(), y /
(float) ExampleUtil::screenHeight());
          gui->postPosition(p);
}

static void mouse(int button, int state, int x, int y) {
          SGFPosition p(x / (float) ExampleUtil::screenWidth(), y /
(float) ExampleUtil::screenHeight());

          if(state == GLUT_DOWN)
                    gui->postButtonDown(button, p);
          else if(state == GLUT_UP)
                    gui->postButtonUp(button, p);
}

int main(int argc, char argv[]) {
          glutInit(&argc, &argv);
          glutInitWindowSize(640, 480);
          glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE);
          glutCreateWindow("SGF Example");
          ExampleUtil::initialize(640, 480);
          initialize();
          glutReshapeFunc(ExampleUtil::resize);
          glutDisplayFunc(render);
          glutIdleFunc(render);
          glutMouseFunc(mouse);
          glutPassiveMotionFunc(motion);
          glutMotionFunc(motion);
          glutMainLoop();
          finalize();
}
```