Appendix A

Appendix: Source code of implemented Software

A.1 main.cpp

```
#include <osgUtil/Optimizer>
#include <osgDB/ReadFile>
#include <osgViewer/CompositeViewer>
#include <osg/StateSet>
#include <osg/Notify>
#include <osgGA/NodeTrackerManipulator>
#include <osgGA/TrackballManipulator>
#include <osgGA/StateSetManipulator>
#include "movement.h"
#include "rotateHT.h"
#include <gtk/gtk.h>
#include <ViewerGtk.h>

// This function reads the coordinates files and store them into Move
extern void readcoordinates (Movement* Move, int nUAVs);

// Function that creates the two GTK viewers with their buttons
extern osgViewer::ViewerGtk* Createview (osg::ref_ptr<osg::Group>
&rootNode, osgGA::NodeTrackerManipulator* tm, int i);

// Tne next function is responsible for constructing all elements of the scene
extern bool setupScene(osg::ref_ptr<osg::Group> &rootNode,
                osg::ref_ptr<osg::Group> &Helicopter, Movement* Move, int i );

using namespace std;

int main( int argc, char **argv )
{
 int nUAVs = 0;
 // Next integer allows us to know which view we are creating.
 int i = 0;

 // Is found if the number of UAVs is valid
 while ((nUAVs < 1)||(nUAVs > 10))
 {
    cout<<"Please enter the number of UAVs in the scene between 1 and 10"<<endl;
    cin >> nUAVs;
```

```
 }

 Movement *Move;
// Dinamic array of variables type Move
 Move = new Movement[10];

// Runs the yarp server to receive the Head-Tracking data
    yarp::os::Network::init();
    yarp::os::Port yport;
    yport.open("/app3D/headpos-in");
    yport.setReader(headposProcessor);
    yarp::os::Network::connect("/head-tracking/headpos","/app3D/headpos-in");

// Call to the function that reads the coordinates files and store them
// into the array Move
  readcoordinates(Move, nUAVs);

  // Initiation of GTK
  gtk_init( &argc, &argv );
  gtk_gl_init( &argc, &argv );

   // pointers to the root node and the UAV node
   osg::ref_ptr<osg::Group> rootNode;
   osg::ref_ptr<osg::Group> Plane;

   // build the scene with the terrain and UAVs
   if (!setupScene(rootNode, Plane, Move, nUAVs))
   {
      std::cout<< "problem setting up scene" << std::endl;
      return -1;
   }

   // We locate the superior camera in the scene
   osg::PositionAttitudeTransform * Superiorcamera =
      new osg::PositionAttitudeTransform();
   Superiorcamera->setPosition( osg::Vec3(0.0,0.0,5000000.0) );
   Superiorcamera->setAttitude(
      osg::Quat( osg::DegreesToRadians(90.0), osg::Vec3(1,0,0) ));

   // tm2 will be the nodetracker manipulator asociated to the superior
   // camera
   osgGA::NodeTrackerManipulator* tm2 = new osgGA::NodeTrackerManipulator;
    tm2->setTrackerMode( osgGA::NodeTrackerManipulator::NODE_CENTER_AND_ROTATION );
    tm2->setRotationMode( osgGA::NodeTrackerManipulator::ELEVATION_AZIM);
    tm2->setTrackNode(Superiorcamera);

   // We define a camera to follow the protagonist UAV
   osg::ref_ptr<osg::PositionAttitudeTransform> followerOffset =
                              new osg::PositionAttitudeTransform();
   followerOffset->setPosition( osg::Vec3(0.0,-7000.0,-1000.0) );
   followerOffset->setAttitude(osg::Quat(osg::DegreesToRadians(0.0),
         osg::Vec3(1,0,0),osg::DegreesToRadians(0.0),osg::Vec3(0,1,0),
                     osg::DegreesToRadians(180.0), osg::Vec3(0,0,1)));

   // With the next command we associate the follower Offset to the UAV
   Plane.get()->addChild(followerOffset);

   // Now we are going to create a new Node, HTmode which will be a son
   // of the followerOffset and will allow us to rotate when the
   // Head-Tracking mode will be activated.
   osg::ref_ptr<osg::MatrixTransform> HTmode =
         new osg::MatrixTransform;
```

```
    followerOffset->addChild( HTmode );

    // Callback in rotateHt.h that update the rotation of the camera
    // when we activate the Head-tracking mode and while we are
    // receiving data form the Head_tracking system.
    HTmode->setUpdateCallback( new RotateHT );

    // tm will be the nodetracker manipulator asociated to the UAV
    // protagonist
    osgGA::NodeTrackerManipulator* tm = new osgGA::NodeTrackerManipulator;
    tm->setTrackerMode( osgGA::NodeTrackerManipulator::NODE_CENTER_AND_ROTATION );
    tm->setRotationMode( osgGA::NodeTrackerManipulator::ELEVATION_AZIM);
    tm->setTrackNode(HTmode);

   // The viewer with its two views is created
   osgViewer::CompositeViewer viewer;
   osg::DisplaySettings::instance()->setNumMultiSamples( 4 );
   viewer.addView(Createview(rootNode, tm2, i) );
   i = 1;
   viewer.addView(Createview(rootNode, tm, i) );

    // Launch the gtk loop
    gtk_main();

    // Free memory
    free (Move);
}
```

A.2 Createscene.cpp

```
/*This .cpp has the aim to build all the scene. The terrain, the
texture of the terrain and add all elements necessaries to it, like
the UAVs, the line to know the height of the UAVs, and elements like
the control tower or the hangars.*/

#include <osgUtil/Optimizer>
#include <osgDB/ReadFile>
#include<osgText/Text>
#include <osg/Projection>
#include <osg/AutoTransform>
#include <osg/Geometry>
#include <osg/StateSet>
#include <osg/Notify>
#include <osg/Material>
#include <osg/Geode>
#include <osg/BlendFunc>
#include <osg/Depth>
#include <osg/MatrixTransform>
#include <osg/PositionAttitudeTransform>
#include <iostream>
#include "movement.h"
#include <vector>

// We need to reference to this functions to create the terrain and the
// trees located in terrain.cpp

extern osg::Node *makeTerrain( int index ); extern
osg::Node*makeTrees( void );

//This function create a associated vertical line for each UAV in the scene,
// to know better the height of each UAV.
osg::Node* Createline()
{
    //We need a PAT node to put the line under UAV
    osg::PositionAttitudeTransform* Helicopteraux =
                new osg::PositionAttitudeTransform();
    osg::Group* group = new osg::Group;
    osg::Vec3Array* vertices = new osg::Vec3Array;

    // we define both ends of the line
    osg::Vec3 pos = osg::Vec3(0.0,0.0,0.0);
    vertices->push_back(pos);
    pos = osg::Vec3(0.0,0.0,-1000000.0);
    vertices->push_back(pos);
    osg::Vec4Array* colors = new osg::Vec4Array;
    //Color of the lines
    colors->push_back(osg::Vec4(0.0f,1.0f,0.0f,1.0f));

    osg::Geometry* geom = new osg::Geometry;
    geom->setVertexArray(vertices);
    geom->setColorArray(colors);
    geom->setColorBinding(osg::Geometry::BIND_OVERALL);
    geom->addPrimitiveSet(new osg::DrawArrays(GL_LINE_STRIP,0,vertices->size()));

    osg::Geode* geode = new osg::Geode;
    geode->addDrawable(geom);

    group->addChild(geode);
```

```
    // line will start a little under the UAV
    Helicopteraux->setPosition( osg::Vec3(0.0,-10000.0,500.0) );
    Helicopteraux->addChild(group);

    return Helicopteraux;
}

// The next function is responsible for show the name of each UAV
osg::Node* createAutoScale(const osg::Vec3& position, float
characterSize, const std::string& message, float minScale=0.0, float
maxScale=FLT_MAX) {
    std::string timesFont("fonts/arial.ttf");

    osgText::Text* text = new osgText::Text;
    text->setCharacterSize(characterSize);
    text->setText(message);
    text->setFont(timesFont);
    text->setAlignment(osgText::Text::CENTER_CENTER);

    osg::Geode* geode = new osg::Geode;
    geode->addDrawable(text);
    geode->getOrCreateStateSet()->setMode(GL_LIGHTING,
                            osg::StateAttribute::OFF);

    osg::AutoTransform* at = new osg::AutoTransform;
    at->addChild(geode);

    at->setAutoRotateMode(osg::AutoTransform::ROTATE_TO_SCREEN);
    at->setAutoScaleToScreen(true);
    at->setMinimumScale(minScale);
    at->setMaximumScale(maxScale);
    at->setPosition(position);

    return at;
}

// This is an auxiliar function to create the terrains around the main
// terrain where is placed the airport.
osg::PositionAttitudeTransform* Createterrainaux(int i, int j)
{
    osg::PositionAttitudeTransform* terrainex =
            new osg::PositionAttitudeTransform();
    if ((i != 0)||(j != 0))
    {
        terrainex->addChild(makeTerrain(1));
        terrainex->setPosition( osg::Vec3(i*10.8,j*11.398,0.0) );
    }
    return terrainex;
}

// With this function, we create the trees present in the scene
osg::PositionAttitudeTransform* Createtreesaux(int i, int j) {
    osg::PositionAttitudeTransform* treesex =
            new osg::PositionAttitudeTransform();
    treesex->addChild(makeTrees());
    if ((i != 0)||(j != 0))
    {
        // We move the trees to locate in the different terrains.
        treesex->setPosition( osg::Vec3(i*10.8,j*11.398,0.0) );
    }
    return treesex;
}
```

```
// This function is responsible to create the terrain, add the trees and
// add the texture to it.
osg::Group* createModel() {
    osg::Group* group = new osg::Group;
    int i;
    int j;

    for (i = -2; i < 3; i++)
    {
        for (j = -2; j < 3; j++)
        {
            group->addChild(Createtreesaux(i,j));
            group->addChild(Createterrainaux(i,j));
        }
    }
    group->addChild(makeTerrain(0));
    return group;
}

// This function is responsible to create each the UAV and to associate
// the line and the name to each of the UAV.
osg::ref_ptr<osg::PositionAttitudeTransform>
CreateUAV(osg::ref_ptr<osg::Group> &Helicopter, Movement* x, int index)
{
    // We define a table with the names for the UAVs
   char UAVnumber[10][10]={
    {"UAV_1"},
    {"UAV_2"},
    {"UAV_3"},
    {"UAV_4"},
    {"UAV_5"},
    {"UAV_6"},
    {"UAV_7"},
    {"UAV_8"},
    {"UAV_9"},
    {"UAV_10"},
};

   osg::ref_ptr<osg::PositionAttitudeTransform> Helicoptermove =
                              new osg::PositionAttitudeTransform();
   Helicoptermove->setDataVariance( osg::Object::DYNAMIC );
   osg::PositionAttitudeTransform* Helicopterform =
                 new osg::PositionAttitudeTransform();
   Helicopterform->addChild(Helicopter.get());

   // Next two lines have the responsability to add to the UAV its line
   // of height and its name respectively
   Helicopterform->addChild(Createline());
   Helicopterform->addChild(createAutoScale(osg::Vec3(0.0,-10000.0,15000.0),
                                                    15.0, UAVnumber[index]));
   Helicoptermove->addChild(Helicopterform);

   // Here we call to the function that simulates the movement of each of the UAVs
   Helicoptermove->setUpdateCallback( x );

 return (Helicoptermove);
}

// In this function we receive rootNode, the UAV model, the array of
// variable type Move and the number of the UAVs in the scene. With
// these data setupScene build the hole scene with all the necessaries
```

```
// elements.
bool setupScene(osg::ref_ptr<osg::Group> &rootNode,
                osg::ref_ptr<osg::Group> &Helicopter, Movement* Move, int i )

// Setup a scene, return (via reference arguments) handles
// to the root node.
// Function returns false if it can't load
//  models or arguments are not NULL, true otherwise.
{
   // We define all elements nodes in the scene

   int index = 0; //Auxiliar variable to create all necessary UAvs
   rootNode = new osg::Group();
   osg::ref_ptr<osg::Group> Aux;
   // Is defined a PAT containing the terrain and it will be a son
   // of the root Node.
   osg::PositionAttitudeTransform * Terrainform =
                new osg::PositionAttitudeTransform();
   rootNode->addChild(Terrainform);

   Aux=createModel();
   Terrainform->addChild(Aux.get());
   Terrainform->setScale(osg::Vec3(200000.0,200000.0,200000.0));

    // Now, we load the model of the control tower and then we locate
    // it into the scene
    osg::ref_ptr<osg::Group> Controltower = (osg::Group*)
      osgDB::readNodeFile("./3Dmodels/Buildings/modelotorre/models/Controltower.3DS");
   if( ! Controltower)
   {
      std::cout << "no Control tower" << std::endl;
      return false;
   }

   osg::PositionAttitudeTransform* Controltowerform =
                    new osg::PositionAttitudeTransform();
   Controltowerform->addChild(Controltower.get());
   Controltowerform->setPosition( osg::Vec3(-80000.0,335000.0,388500.0) );
   Controltowerform->setAttitude( osg::Quat(osg::DegreesToRadians(270.0),
            osg::Vec3(1,0,0), osg::DegreesToRadians(0.0), osg::Vec3(0,1,0),
                              osg::DegreesToRadians(77.0), osg::Vec3(0,0,1) ));
   rootNode->addChild(Controltowerform);

   //We load the 3D model of the hangar

   osg::ref_ptr<osg::Group> Hangar = (osg::Group*)
      osgDB::readNodeFile("./3Dmodels/Buildings/HANGAR/HANGAR/HANGAR_L.3DS");
   if( ! Controltower)
   {
      std::cout << "no Hangar" << std::endl;
      return false;
   }

   //Locating the first hangar
   osg::PositionAttitudeTransform* Hangarform = new osg::PositionAttitudeTransform();
   Hangarform->addChild(Hangar.get());
   Hangarform->setPosition( osg::Vec3(-90000.0,-325000.0,357500.0) );
   Hangarform->setAttitude( osg::Quat(osg::DegreesToRadians(0.0), osg::Vec3(1,0,0),
                                      osg::DegreesToRadians(0.0), osg::Vec3(0,1,0),
                                   osg::DegreesToRadians(50.0), osg::Vec3(0,0,1) ));
   rootNode->addChild(Hangarform);
```

```
    //Locating the second hangar

    osg::PositionAttitudeTransform* Hangar2form = new osg::PositionAttitudeTransform();
    Hangar2form->addChild(Hangar.get());
    Hangar2form->setPosition( osg::Vec3(260000.0,-130000.0,357500.0) );
    Hangar2form->setAttitude( osg::Quat(osg::DegreesToRadians(0.0), osg::Vec3(1,0,0),
                                        osg::DegreesToRadians(0.0), osg::Vec3(0,1,0),
                                      osg::DegreesToRadians(-130.0), osg::Vec3(0,0,1) ));
    rootNode->addChild(Hangar2form);

    // We call to the function which creates the UAVs
    Helicopter = (osg::Group*)
       osgDB::readNodeFile("./3Dmodels/UAVs/Helicopters/500D/500D/500D_L.3DS");
    if( ! Helicopter)
    {
       std::cout << "no Helicopter" << std::endl;
       return false;
    }
    while (index < i)
    {
     rootNode->addChild(CreateUAV(Helicopter, &Move[index], index));
     index++;
    }

    //Helicopter near the hangar
    osg::ref_ptr<osg::Group> Helicopter1 = (osg::Group*)
       osgDB::readNodeFile("./3Dmodels/UAVs/Helicopters/ABJETR2/ABJETR2/ABJETR2_.3DS");
    if( ! Helicopter1)
    {
       std::cout << "no Plane" << std::endl;
       return false;
    }

    //We locate the helicopter in the scene

    osg::PositionAttitudeTransform* Helicopter1form =
                  new osg::PositionAttitudeTransform();
    Helicopter1form->addChild(Helicopter1.get());
    Helicopter1form->setPosition( osg::Vec3(245000.0,-165000.0,352100.0) );
    Helicopter1form->setAttitude( osg::Quat(osg::DegreesToRadians(0.0),
          osg::Vec3(1,0,0), osg::DegreesToRadians(0.0), osg::Vec3(0,1,0),
                         osg::DegreesToRadians(-40.0), osg::Vec3(0,0,1) ));
    rootNode->addChild(Helicopter1form);

    return true;
}
```

A.3 Createviews.cpp

```
/*This .cpp contains functions to create the views in the viewer and
to create the gtk buttons necessaries to control the global
variables to activate the Head-Tracking and loitering_mode.*/

#include <osgGA/NodeTrackerManipulator>
#include <gtk/gtk.h>
#include <ViewerGtk.h>

extern bool manuallyPlaceCamera; int width=800, height=600; int
loitering_mode = 0;

// Netx text will be shown at the bottom of the viewer
const char* HELP_TEXT =
    "<b><i>Right-click</i></b> while holding <b><i>CTRL</i></b>
    or</u> <b><i>SHIFT</i></b> to pop up a fake Gtk menu.\n"
    "\n"
    "Use the standard OSG TrackballManipulator keys and mouse actions to\n"
    "rotate the camera.\n"
    "\n"
    "<i>Note:\n"
    "<small>\tIf the viewer is stopped, user actions won't be processed until
    the viewer is restarted.\n"
    "\tIn this example events are only processed when the viewer is
    running.</small></i>\n"
;

// This is also global so the callback function can set the
// state of the toggle button
GtkWidget* fps_button;

// Callback to handle mouse button events
// This is what will look for right-clicks and pop up the menu
gboolean on_button_event(GtkWidget* widget, GdkEventButton* event,
gpointer data);

//CallBack that runs the Head-Tracking Mode
void HTM_clicked(GtkWidget* widget, gpointer data);

// Callback that runs the loitering mode
void loitering_mode_button_clicked(GtkWidget* widget, gpointer
data);

// Callback that starts/stops the viewer
void on_fps_toggled(GtkWidget* widget, gpointer data);

// The next function receive the rootNode and a NodeTrackerManipulator
// and return a GTK viewer

osgViewer::ViewerGtk* Createview (osg::ref_ptr<osg::Group>&rootNode,
                                  osgGA::NodeTrackerManipulator* tm, int i)
{
  osgViewer::ViewerGtk* view = new osgViewer::ViewerGtk();
  view->set_fps(60);

  view->setSceneData(rootNode);
  view->setCameraManipulator( tm );

  // Now we'll setup the viewer in a gtk window using the
  // setup_viewer_in_gtk_window() convenience method
```

```
// The return type is a Gtk widget that we can use anywhere we want
osgViewer::GraphicsWindowGtk* gw = view->setup_viewer_in_gtk_window(width,height);

// All Gtk apps need a toplevel window
//
// We'll also set the title, add our widget created by the viewer
// and make sure it's visible
GtkWidget* window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
if (i == 0)
{
  gtk_window_set_title(GTK_WINDOW(window), "Aerial view");
}
else
{
  gtk_window_set_title(GTK_WINDOW(window), "UAV view");
}

// Almost all of this is just Gtk code... creating the various
// layouts and other UI widgets
GtkWidget* main_vbox = gtk_vbox_new(FALSE, 3);
GtkWidget* bottom_hbox = gtk_hbox_new(FALSE, 3);
GtkWidget* button_vbox = gtk_vbox_new(TRUE, 3);
GtkWidget* help_text = gtk_label_new(HELP_TEXT);
GtkWidget* HTM_button = gtk_button_new_with_label("Head Tracking Mode");
GtkWidget* loitering_mode_button = gtk_button_new_with_label("Loitering mode");
GtkWidget* close_button = gtk_button_new_with_label("Close");
fps_button = gtk_toggle_button_new();

gtk_label_set_use_markup(GTK_LABEL(help_text), TRUE);

gtk_container_add(GTK_CONTAINER(window), main_vbox);
gtk_container_add(GTK_CONTAINER(main_vbox), gw->gtk_widget());
gtk_container_add(GTK_CONTAINER(main_vbox), bottom_hbox);
gtk_container_add(GTK_CONTAINER(bottom_hbox), button_vbox);
gtk_container_add(GTK_CONTAINER(bottom_hbox), help_text);
gtk_container_add(GTK_CONTAINER(button_vbox), HTM_button);
gtk_container_add(GTK_CONTAINER(button_vbox), loitering_mode_button);
gtk_container_add(GTK_CONTAINER(button_vbox), fps_button);
gtk_container_add(GTK_CONTAINER(button_vbox), close_button);
gtk_widget_show_all(window);

// Even though we won't do anything except popup the menu on button release
// we still need to handle the case where the button is pressed with the
// shift or control key. Otherwise the button press will be passed on to
// the gtk handler.
g_signal_connect( G_OBJECT(gw->gtk_widget()),
                  "button_press_event",
                  G_CALLBACK(on_button_event),
                  NULL
                );
g_signal_connect( G_OBJECT(gw->gtk_widget()),
                  "button_release_event",
                  G_CALLBACK(on_button_event),
                  NULL
                );

// Let's connect the other Gtk buttons to their callbacks
g_signal_connect( G_OBJECT(HTM_button),
                  "clicked",
                  G_CALLBACK(HTM_clicked),
                  view
                );
```

```
  g_signal_connect( G_OBJECT(loitering_mode_button),
                    "clicked",
                    G_CALLBACK(loitering_mode_button_clicked),
                    view
                  );
  g_signal_connect( G_OBJECT(fps_button),
                    "toggled",
                    G_CALLBACK(on_fps_toggled),
                    view
                  );
  g_signal_connect(G_OBJECT(close_button),"clicked",G_CALLBACK(gtk_main_quit),0);

  // And, just before we kick things off, let's start the viewer running
  // by setting the fps button to active, allowing the connected callback
  // to start things off.
  gtk_toggle_button_set_active(GTK_TOGGLE_BUTTON(fps_button), TRUE);
  g_signal_connect(G_OBJECT(window),"delete_event",G_CALLBACK(gtk_main_quit),0);

return view;
}

// This function is called every time there is a button press or button
// release event that occurs in the OSG graphics window
// Even though it is only the button release that we're concerned with,
// we still handle the button press event since anything that we don't
// handle is passed along to OSG. We don't want button press events
// to be passed into OSG when the user is holding down the shift
// key or the control key, so we stop that by returning true
// when either key is held down and the right mouse button is clicked.
gboolean on_button_event(GtkWidget* widget, GdkEventButton* event,
gpointer data) {
  if ( event->button == 3 and ( event->state & GDK_SHIFT_MASK or
                                event->state & GDK_CONTROL_MASK ) )
  {
    // We get here if it's either a button press or release, but we'll
    // only pop up the menu on a release
    if ( event->type == GDK_BUTTON_RELEASE )
    {
      GtkWidget* menu = gtk_menu_new();
      gtk_menu_append(menu,gtk_menu_item_new_with_label("Option"));
      gtk_menu_append(menu,gtk_menu_item_new_with_label("Another Option"));
      gtk_menu_append(menu,gtk_menu_item_new_with_label("Still More Options"));
      gtk_widget_show_all(menu);
      gtk_menu_popup(GTK_MENU(menu),NULL,NULL,NULL,NULL,event->button,event->time);
    }

    // Whether it's press or release, if it's a right mouse event with
    // control or shift held down we return true, stopping it from
    // going on to OSG
    //
    // If we wanted to handled it _and_ let it pass into OSG we
    // could simply return false here.
    return true;
  }

  return false;
}

//CallBack that runs the Head-Tracking Mode when the button Head-Tracking
// is pressed
void HTM_clicked(GtkWidget* widget, gpointer data)
{
```

```
    if (! manuallyPlaceCamera)
        manuallyPlaceCamera = true;
    else
        manuallyPlaceCamera = false;
}

// Callback that runs/stop the loitering mode when the button
// loitering_mode is pressed void
loitering_mode_button_clicked(GtkWidget* widget, gpointer data)
{
    if (loitering_mode == 0)
     {
        loitering_mode = 1;
    }
    else
    {
        loitering_mode = 0;
    }
}

// Callback that starts/stops the viewer when the button is pressed
void on_fps_toggled(GtkWidget* widget, gpointer data)
{
  osgViewer::ViewerGtk* viewer = (osgViewer::ViewerGtk*)data;
  if ( gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(widget)))
  {
    if ( viewer ) viewer->run();
    gtk_button_set_label(GTK_BUTTON(widget), "Click to stop");
  }
  else
  {
    if ( viewer ) viewer->stop();
    gtk_button_set_label(GTK_BUTTON(widget), "Click to run at 60fps");
  }
}
```

A.4 GraphicsWindowsGtk.cpp

```
#include "GraphicsWindowGtk.h"
#include <iostream>
#include <gtk/gtkdrawingarea.h>

namespace osgViewer
{
  GraphicsWindowGtk::GraphicsWindowGtk(int width, int height, int mode):
      m_gtk_widget(NULL),
      m_gdk_gl_config(NULL)
  {
    // OSG housekeeping
    _traits = new GraphicsContext::Traits();
    _traits->x = 0;
    _traits->y = 0;
    _traits->width = width;
    _traits->height = height;

    this->setState( new osg::State );
    this->getState()->setGraphicsContext(this);

    if (_traits.valid() && _traits->sharedContext)
    {
      getState()->setContextID(_traits->sharedContext->getState()->getContextID());
      incrementContextIDUsageCount( getState()->getContextID() );
    }
    else
    {
      getState()->setContextID( osg::GraphicsContext::createNewContextID() );
    }

    // GTK housekeeping
    m_gtk_widget = gtk_drawing_area_new();
    if ( m_gtk_widget == NULL ) throw(-1);

    gtk_widget_add_events( m_gtk_widget,
                           GDK_POINTER_MOTION_MASK |
                           GDK_BUTTON_MOTION_MASK |
                           GDK_BUTTON1_MOTION_MASK |
                           GDK_BUTTON2_MOTION_MASK |
                           GDK_BUTTON3_MOTION_MASK |
                           GDK_BUTTON_PRESS_MASK |
                           GDK_BUTTON_RELEASE_MASK |
                           GDK_KEY_PRESS_MASK |
                           GDK_KEY_RELEASE_MASK |
                           GDK_PROXIMITY_IN_MASK |
                           GDK_PROXIMITY_OUT_MASK
                         );

    m_gdk_gl_config = gdk_gl_config_new_by_mode( (GdkGLConfigMode) mode );
    if ( m_gdk_gl_config == NULL ) throw(-1);

    if ( ! gtk_widget_set_gl_capability( m_gtk_widget, m_gdk_gl_config,
                           NULL, TRUE, GDK_GL_RGBA_TYPE ) ) throw(-1);

    this->getEventQueue()->setCurrentEventState(
        osgGA::GUIEventAdapter::getAccumulatedEventState().get());

    g_signal_connect(G_OBJECT(m_gtk_widget), "configure-event",
            G_CALLBACK(GraphicsWindowGtk::on_configure_event), this );
```

```
  g_signal_connect(G_OBJECT(m_gtk_widget), "button-press-event",
                   G_CALLBACK(GraphicsWindowGtk::on_button_event), this );

  g_signal_connect(G_OBJECT(m_gtk_widget), "button-release-event",
                   G_CALLBACK(GraphicsWindowGtk::on_button_event), this );

  g_signal_connect(G_OBJECT(m_gtk_widget), "key-press-event",
                   G_CALLBACK(GraphicsWindowGtk::on_key_event), this );

  g_signal_connect(G_OBJECT(m_gtk_widget), "key-release-event",
                   G_CALLBACK(GraphicsWindowGtk::on_key_event), this );

  g_signal_connect(G_OBJECT(m_gtk_widget), "motion-notify-event",
      G_CALLBACK(GraphicsWindowGtk::on_motion_notify_event), this );

  g_signal_connect(G_OBJECT(m_gtk_widget), "proximity-in-event",
           G_CALLBACK(GraphicsWindowGtk::on_proximity_event), this );

  g_signal_connect(G_OBJECT(m_gtk_widget), "proximity-out-event",
           G_CALLBACK(GraphicsWindowGtk::on_proximity_event), this );

  g_signal_connect(G_OBJECT(m_gtk_widget), "scroll-event",
                   G_CALLBACK(GraphicsWindowGtk::on_scroll_event), this );

  g_object_set(m_gtk_widget, "can-focus", true, NULL);

  gtk_widget_set_size_request( m_gtk_widget, width, height );
}

GraphicsWindowGtk::~ GraphicsWindowGtk()
{
}

bool GraphicsWindowGtk::isSameKindAs(const osg::Object* object) const
{
  return dynamic_cast<const GraphicsWindowGtk*>(object)!=0;
}

const char* GraphicsWindowGtk::libraryName() const
{
  return "osgGtk";
}

const char* GraphicsWindowGtk::className() const
{
  return "GraphicsWindowGtk";
}

GtkWidget * GraphicsWindowGtk::gtk_widget()
{
  return m_gtk_widget;
}

bool GraphicsWindowGtk::setWindowRectangleImplementation(int x, int y,
                                                         int width, int height)
{
  if ( not GTK_WIDGET_REALIZED(m_gtk_widget) ) return false;

  GtkRequisition req;
  req.width = width;
  req.height = height;
```

```
  gtk_widget_size_request(m_gtk_widget, &req);

  // from osgViewer::GraphicsWindowX11
  // add usleep here to give window manager a chance to handle the request, if
  // we don't add this sleep then any X11 calls right afterwards can produce
  // X11 errors.
  usleep(100000);

  return true;
}

bool GraphicsWindowGtk::setWindowDecorationImplementation(bool flag)
{
  GdkWindow* root = gtk_widget_get_root_window(m_gtk_widget);

  if ( root == NULL ) return false;

  if ( flag )
    gdk_window_set_decorations( root, GDK_DECOR_ALL );
  else
    gdk_window_set_decorations( root, GDK_DECOR_BORDER );

  return true;
}

void GraphicsWindowGtk::setWindowName(const std::string & name)
{
  if ( not GTK_WIDGET_REALIZED(m_gtk_widget) ) return;

  GdkWindow* root = gtk_widget_get_root_window(m_gtk_widget);
  if ( root == NULL ) return;
  gdk_window_set_title(root, name.c_str());
}

void GraphicsWindowGtk::grabFocus()
{
  gtk_widget_grab_focus( m_gtk_widget );
}

void GraphicsWindowGtk::grabFocusIfPointerInWindow()
{
  gint x, y;

  gtk_widget_get_pointer( m_gtk_widget, &x, &y );

  gint left   = m_gtk_widget->allocation.x,
       right  = m_gtk_widget->allocation.x + m_gtk_widget->allocation.width,
       top    = m_gtk_widget->allocation.y,
       bottom = m_gtk_widget->allocation.y + m_gtk_widget->allocation.height;

  if ( x >= left && x <= right && y >= top && y <= bottom )
    gtk_widget_grab_focus( m_gtk_widget );
}

bool GraphicsWindowGtk::valid() const
{
  return true;
}

void GraphicsWindowGtk::setCursor(MouseCursor mouseCursor)
{
  GdkCursor* cursor;
```

```
    switch(mouseCursor)
    {
      case InheritCursor:      cursor = NULL;
break;
      case NoCursor:           cursor = NULL;
break;
      case RightArrowCursor:   cursor = gdk_cursor_new(GDK_SB_RIGHT_ARROW);
break;
      case LeftArrowCursor:    cursor = gdk_cursor_new(GDK_SB_LEFT_ARROW);
break;
      case InfoCursor:         cursor = gdk_cursor_new(GDK_GUMBY);
break;
      case DestroyCursor:      cursor = gdk_cursor_new(GDK_CROSS_REVERSE);
break;
      case HelpCursor:         cursor = gdk_cursor_new(GDK_QUESTION_ARROW);
break;
      case CycleCursor:        cursor = gdk_cursor_new(GDK_EXCHANGE);
break;
      case SprayCursor:        cursor = gdk_cursor_new(GDK_SPRAYCAN);
break;
      case WaitCursor:         cursor = gdk_cursor_new(GDK_WATCH);
break;
      case TextCursor:         cursor = gdk_cursor_new(GDK_XTERM);
break;
      case CrosshairCursor:    cursor = gdk_cursor_new(GDK_TCROSS);
break;
      case UpDownCursor:       cursor = gdk_cursor_new(GDK_SB_V_DOUBLE_ARROW);
break;
      case LeftRightCursor:    cursor = gdk_cursor_new(GDK_SB_H_DOUBLE_ARROW);
break;
      case TopSideCursor:      cursor = gdk_cursor_new(GDK_TOP_SIDE);
break;
      case BottomSideCursor:   cursor = gdk_cursor_new(GDK_BOTTOM_SIDE);
break;
      case LeftSideCursor:     cursor = gdk_cursor_new(GDK_LEFT_SIDE);
break;
      case RightSideCursor:    cursor = gdk_cursor_new(GDK_RIGHT_SIDE);
break;
      case TopLeftCorner:      cursor = gdk_cursor_new(GDK_TOP_LEFT_CORNER);
break;
      case TopRightCorner:     cursor = gdk_cursor_new(GDK_TOP_RIGHT_CORNER);
break;
      case BottomRightCorner:  cursor = gdk_cursor_new(GDK_BOTTOM_RIGHT_CORNER);
      break;
      case BottomLeftCorner:   cursor = gdk_cursor_new(GDK_BOTTOM_LEFT_CORNER);
break;
    }

    GdkWindow* root = gtk_widget_get_root_window( m_gtk_widget );
    if ( not root ) return;

    gdk_window_set_cursor(root, cursor);
    gdk_cursor_destroy(cursor);
  }

  bool GraphicsWindowGtk::realizeImplementation()
  {
    if ( GTK_WIDGET_REALIZED(m_gtk_widget) )
    {
      osg::notify(osg::NOTICE)<<''GraphicsWindowGtk::realizeImplementation()
                                                  Already realized''<<std::endl;
```

```
      return true;
    }

    gtk_widget_realize(m_gtk_widget);

    return GTK_WIDGET_REALIZED(m_gtk_widget);
  }

  bool GraphicsWindowGtk::isRealizedImplementation() const
  {
    return GTK_WIDGET_REALIZED(m_gtk_widget);
  }

  bool GraphicsWindowGtk::makeCurrentImplementation()
  {
    if ( not GTK_WIDGET_REALIZED(m_gtk_widget) )
    {
      osg::notify(osg::NOTICE)<<''Warning: GraphicsWindowGtk not realized,
                                          cannot do makeCurrent.''<<std::endl;
      return false;
    }

    GdkGLDrawable* drawable = gtk_widget_get_gl_drawable(m_gtk_widget);
    GdkGLContext* context = gtk_widget_get_gl_context(m_gtk_widget);

  if ( not drawable or not context )
    {
      if ( not context )
        osg::notify(osg::NOTICE)<<"Warning: GraphicsWindowGtk doesn't have
        a valid GL context, cannot do makeCurrent."<<std::endl;
      if ( not drawable )
        osg::notify(osg::NOTICE)<<"Warning: GraphicsWindowGtk doesn't have
        a valid GL drawable, cannot do makeCurrent."<<std::endl;
      return false;
    }

    if ( not gdk_gl_drawable_make_current(drawable, context) )
    {
      osg::notify(osg::NOTICE)<<"Warning: GraphicsWindowGtk couldn't call
      GL drawable's make_current(), cannot do makeCurrent."<<std::endl;
      return false;
    }


    return true;
  }

  bool GraphicsWindowGtk::releaseContextImplementation()
  {
    return true;
  }

  void GraphicsWindowGtk::closeImplementation()
  {

  }

  void GraphicsWindowGtk::swapBuffersImplementation()
  {
    this->swap_buffers();
  }
```

```
bool GraphicsWindowGtk::swap_buffers()
{
  GdkGLDrawable* drawable = gtk_widget_get_gl_drawable(m_gtk_widget);

  if (not GTK_WIDGET_REALIZED(m_gtk_widget) or not drawable) return false;

  if ( gdk_gl_drawable_is_double_buffered(drawable) )
    gdk_gl_drawable_swap_buffers(drawable);
  else
    glFlush();

  return true;
}

bool GraphicsWindowGtk::gl_begin()
{
  GdkGLDrawable* drawable = gtk_widget_get_gl_drawable(m_gtk_widget);
  GdkGLContext* context = gtk_widget_get_gl_context(m_gtk_widget);

  if ( not GTK_WIDGET_REALIZED(m_gtk_widget) or not drawable or not context )
    return false;

  return gdk_gl_drawable_gl_begin( drawable, context );
}

bool GraphicsWindowGtk::gl_end()
{
  GdkGLDrawable* drawable = gtk_widget_get_gl_drawable(m_gtk_widget);

  if ( not GTK_WIDGET_REALIZED(m_gtk_widget) or not drawable ) return false;

  gdk_gl_drawable_gl_end( drawable );

  return true;
}

gboolean GraphicsWindowGtk::on_button_event(GtkWidget * widget,
                      GdkEventButton * event, gpointer user_data)
{
  static bool emitting = false;
  bool handled;

  if (emitting) return false;

  emitting = true;
  handled = gtk_widget_event(widget, (GdkEvent*)event);
  emitting = false;

  if ( not handled and GTK_WIDGET_REALIZED(widget) )
  {
    GraphicsWindowGtk* gw = static_cast<GraphicsWindowGtk*>(user_data);
    double x = event->x;
    double y = event->y;
    gw->transform_mouse_x_y(x, y);

    switch (event->type)
    {
      case GDK_BUTTON_PRESS:
        gw->_eventQueue->mouseButtonPress( x, y, event->button );
        break;
      case GDK_2BUTTON_PRESS:
        gw->_eventQueue->mouseDoubleButtonPress( x, y, event->button );
```

```
          break;
        case GDK_BUTTON_RELEASE:
          gw->_eventQueue->mouseButtonRelease( x, y, event->button );
          break;
        case GDK_3BUTTON_PRESS: /* 3-button presses aren't handled in OSG */
        default:
          return false;
      }
    }

    // We always return true to indicate that we handled the signal
    // Otherwise the signal will emit a second time
    return TRUE;
  }

  gboolean GraphicsWindowGtk::on_configure_event(GtkWidget * widget,
                        GdkEventConfigure * event, gpointer user_data)
  {
    GraphicsWindowGtk* gw = static_cast<GraphicsWindowGtk*>(user_data);
//      gw->resized(event->x, event->y, event->width, event->height);
    gw->_eventQueue->windowResize(event->x, event->y, event->width, event->height);
    return false;
  }

  gboolean GraphicsWindowGtk::on_key_event(GtkWidget * widget,
                        GdkEventKey * event, gpointer user_data)
  {
    GraphicsWindowGtk* gw = static_cast<GraphicsWindowGtk*>(user_data);
    switch (event->type)
    {
      case GDK_KEY_PRESS_MASK:
        gw->_eventQueue->keyPress( event->keyval);
        break;
      case GDK_KEY_RELEASE_MASK:
        gw->_eventQueue->keyRelease( event->keyval);
        break;
      default:
        return false;
    }
    return true;
  }

  gboolean GraphicsWindowGtk::on_proximity_event(GtkWidget * widget,
                        GdkEventProximity * event, gpointer user_data)
  {
    GraphicsWindowGtk* gw = static_cast<GraphicsWindowGtk*>(user_data);

    osgGA::GUIEventAdapter::TabletPointerType tab_type;

    switch (event->device->source)
    {
      case GDK_SOURCE_PEN:
        tab_type = osgGA::GUIEventAdapter::PEN;
        break;
      case GDK_SOURCE_ERASER:
        tab_type = osgGA::GUIEventAdapter::ERASER;
        break;
      case GDK_SOURCE_CURSOR:
        tab_type = osgGA::GUIEventAdapter::PUCK;
        break;
      default:
        tab_type = osgGA::GUIEventAdapter::UNKNOWN;
```

```
      break;
  }

  switch ( event->type )
  {
    case GDK_PROXIMITY_IN:
      gw->_eventQueue->penProximity(tab_type, true);
      break;
    case GDK_PROXIMITY_OUT:
      gw->_eventQueue->penProximity(tab_type, false);
      break;
    default:
      break;
  }
  return false;
}

gboolean GraphicsWindowGtk::on_motion_notify_event(GtkWidget * widget,
                                GdkEventMotion * event, gpointer user_data)
{
  GraphicsWindowGtk* gw = static_cast<GraphicsWindowGtk*>(user_data);
  gw->_eventQueue->mouseMotion(event->x, event->y);
  return true;
}

gboolean GraphicsWindowGtk::on_scroll_event(GtkWidget * widget,
                            GdkEventScroll * event, gpointer user_data)
{
  GraphicsWindowGtk* gw = static_cast<GraphicsWindowGtk*>(user_data);

  if ( event->device->num_axes >= 2 )
  {
    gw->_eventQueue->mouseScroll(osgGA::GUIEventAdapter::SCROLL_2D);
    gw->_eventQueue->mouseScroll2D(event->x, event->y);
  }
  else
  {
    osgGA::GUIEventAdapter::ScrollingMotion dir;
    switch(event->direction)
    {
      case GDK_SCROLL_UP:
        dir = osgGA::GUIEventAdapter::SCROLL_UP;
        break;
      case GDK_SCROLL_DOWN:
        dir = osgGA::GUIEventAdapter::SCROLL_DOWN;
        break;
      case GDK_SCROLL_LEFT:
        dir = osgGA::GUIEventAdapter::SCROLL_LEFT;
        break;
      case GDK_SCROLL_RIGHT:
        dir = osgGA::GUIEventAdapter::SCROLL_RIGHT;
        break;
    }
    gw->_eventQueue->mouseScroll(dir);
  }
  return false;
}

void GraphicsWindowGtk::transform_mouse_x_y(double & x, double & y)
{
  // From osgViewer::GraphicsWindowX11::transformMouseXY
  if (this->getEventQueue()->getUseFixedMouseInputRange())
```

```
    {
      osgGA::GUIEventAdapter* eventState = this->getEventQueue()->getCurrentEventState();
      x = eventState->getXmin() +
      (eventState->getXmax()-eventState->getXmin())*x/double(_traits->width);
      y = eventState->getYmin() +
      (eventState->getYmax()-eventState->getYmin())*y/double(_traits->height);
    }
  }
}
```

A.5 GraphicsWindowsGtk.h

```
#ifndef OSGVIEWER_GRAPHICSWINDOWGTK_H
#define OSGVIEWER_GRAPHICSWINDOWGTK_H
#include <gtk/gtkwidget.h>
#include <gtk/gtkgl.h>
#include <osgViewer/GraphicsWindow>

namespace osgViewer
{
  /**
   * This class extends osgViewer::GraphicsWindow to provide an OpenGL
   * rending context for osgViewer::Viewer inside a gtk_gl_drawing_area
   * @todo add a constructor that takes a traits argument and configures
   * GL mode from traits
   */
  class GraphicsWindowGtk: public osgViewer::GraphicsWindow
  {
    public:

      /** Constructs the gtk graphics window of the specified width and height */
      GraphicsWindowGtk(int width, int height,
      int mode=GDK_GL_MODE_RGB|GDK_GL_MODE_DEPTH|GDK_GL_MODE_DOUBLE);

      virtual ~GraphicsWindowGtk();

      virtual bool isSameKindAs(const osg::Object* object) const;

      virtual const char* libraryName() const;

      virtual const char* className() const;

      /** Returns the gtk_gl_drawing_area that the scenegraph will be rendered
      into */
      GtkWidget* gtk_widget();

      /** Implicitly converts the graphics window to a GtkWidget* */
      operator GtkWidget*() { return m_gtk_widget; }

      /** Overrides osgViewer::GraphicsWindow with Gtk specific behavior */
      virtual void grabFocus();

      /** Overrides osgViewer::GraphicsWindow with Gtk specific behavior */
      virtual void grabFocusIfPointerInWindow();

      /** Overrides osgViewer::GraphicsWindow with Gtk specific behavior */
      virtual bool valid() const;

      /** Overrides osgViewer::GraphicsWindow with Gtk specific behavior */
      virtual bool setWindowRectangleImplementation(int x, int y,
                                                    int width, int height);

      /** Overrides osgViewer::GraphicsWindow with Gtk specific behavior */
      virtual void setCursor(MouseCursor mouseCursor);

      /** Overrides osgViewer::GraphicsWindow with Gtk specific behavior */
      virtual bool setWindowDecorationImplementation(bool flag);

      /** Overrides osgViewer::GraphicsWindow with Gtk specific behavior */
      virtual void setWindowName(const std::string& name);
```

```
        /** Overrides osgViewer::GraphicsWindow with Gtk specific behavior */
        virtual bool realizeImplementation();

        /** Overrides osgViewer::GraphicsWindow with Gtk specific behavior */
        virtual bool isRealizedImplementation() const;

        /** Overrides osgViewer::GraphicsWindow with Gtk specific behavior */
        virtual bool makeCurrentImplementation();

        /** Overrides osgViewer::GraphicsWindow with Gtk specific behavior */
        virtual bool releaseContextImplementation();

        /** Overrides osgViewer::GraphicsWindow with Gtk specific behavior */
        virtual void closeImplementation();

        /** Overrides osgViewer::GraphicsWindow with Gtk specific behavior */
        virtual void swapBuffersImplementation();

        /**
         * Convenience method to swap the OpenGL buffers if double-buffered
         * or call glFlush() if single-buffered
         */
        bool swap_buffers();

        /** Convenience method to implement glBegin() on this window's OpenGL drawable */
        bool gl_begin();

        /** Convenience method to implement glEnd() on this window's OpenGL drawable */
        bool gl_end();

    protected:
      GtkWidget* m_gtk_widget;
      GdkGLConfig* m_gdk_gl_config;

      static gboolean on_button_event(GtkWidget* widget,
                    GdkEventButton* event,gpointer user_data);

      static gboolean on_configure_event(GtkWidget* widget,
                GdkEventConfigure* event, gpointer user_data);

      static gboolean on_key_event(GtkWidget* widget,
                      GdkEventKey* event, gpointer user_data);

      static gboolean on_motion_notify_event(GtkWidget* widget,
                      GdkEventMotion* event, gpointer user_data);

      static gboolean on_proximity_event(GtkWidget* widget,
                GdkEventProximity* event, gpointer user_data);

      static gboolean on_scroll_event(GtkWidget* widget,
                    GdkEventScroll* event, gpointer user_data);

      void transform_mouse_x_y( double& x, double& y );
  };
} #endif
```

A.6 hat.cpp

```
\*This functions allow us to locate the trees in the scene

#ifdef _MSC_VER
#include <Windows.h>
#pragma warning( disable :4244)
#endif
#include <osg/GL>
#include <osg/Math>
#include <stdio.h>
#include "terrain_coords.h"
#include "hat.h"

static int inited = 0;

static float dbcenter[3]; static float dbradius;

static void getDatabaseCenterRadius( float dbcenter[3], float
*dbradius )
{
    int i;
    double n=0.0;
    double center[3] = { 0.0f, 0.0f, 0.0f };
    float  cnt;

    cnt = 39 * 38;
    for( i = 0; i < cnt; i++ )
    {
        center[0] += (double)vertex[i][0];
        center[1] += (double)vertex[i][1];
        center[2] += (double)vertex[i][2];

        n = n + 1.0;
    }

    center[0] /= n;
    center[1] /= n;
    center[2] /= n;

    float r = 0.0;

    //    for( i = 0; i < sizeof( vertex ) / (sizeof( float[3] )); i++ )
    for( i = 0; i < cnt; i++ )
    {
        double d = sqrt(
        (((double)vertex[i][0]-center[0])*((double)vertex[i][0]-center[0]))+
        (((double)vertex[i][1]-center[1])*((double)vertex[i][1]-center[1]))+
        (((double)vertex[i][2]-center[2])* (double)vertex[i][2]-center[2])));

        if( d > (double)r ) r = (float)d;

    }

    *dbradius = r;
    dbcenter[0] = (float)center[0];
    dbcenter[1] = (float)center[1];
    dbcenter[2] = (float)center[2];

    int index = 19 * 39 + 19;
    dbcenter[0] = vertex[index][0] - 0.15;
```

```
    dbcenter[1] = vertex[index][1];
    dbcenter[2] = vertex[index][2] + 0.35;
}


static void init( void ) {
    getDatabaseCenterRadius( dbcenter, &dbradius );
    inited = 1;
}


static void getNormal(float *v1,float *v2,float *v3,float *n)
{
    float V1[4], V2[4];
    float   f;
    int i;

    /* Two vectors v2->v1 and v2->v3 */

    for( i = 0; i < 3; i++ )
    {
        V1[i] = v1[i] - v2[i];
        V2[i] = v3[i] - v2[i];
    }

    /* Cross product between V1 and V2 */

    n[0] = (V1[1] * V2[2]) - (V1[2] * V2[1]);
    n[1] = -((V1[0] * V2[2]) - ( V1[2] * V2[0] ));
    n[2] = (V1[0] * V2[1] ) - (V1[1] * V2[0] );

    /* Normalize */

    f = sqrtf((n[0] * n[0]) + (n[1] * n[1]) + (n[2] * n[2]));
    n[0] /= f;
    n[1] /= f;
    n[2] /= f;
}


float Hat( float x, float y, float z )
{
    int m, n;
    int i, j;
    float tri[3][3];
    float norm[3];
    float d, pz;

    if( inited == 0 ) init();

    // m = columns
    // n = rows
    m = (sizeof( vertex ) /(sizeof( float[3])))/39;
    n = 39;

    i = 0;
    while( i < ((m-1)*39) && x > (vertex[i+n][0] - dbcenter[0]) )
        i += n;

    j = 0;

    while( j < n-1 && y > (vertex[i+j+1][1] - dbcenter[1]) )
```

```
        j++;

    tri[0][0] = vertex[i+0+j+0][0] - dbcenter[0];
    tri[0][1] = vertex[i+0+j+0][1] - dbcenter[1];
    //tri[0][2] = vertex[i+0+j+0][2] - dbcenter[2];
    tri[0][2] = vertex[i+0+j+0][2];

    tri[1][0] = vertex[i+n+j+0][0] - dbcenter[0];
    tri[1][1] = vertex[i+n+j+0][1] - dbcenter[1];
    //tri[1][2] = vertex[i+n+j+0][2] - dbcenter[2];
    tri[1][2] = vertex[i+n+j+0][2];

    tri[2][0] = vertex[i+0+j+1][0] - dbcenter[0];
    tri[2][1] = vertex[i+0+j+1][1] - dbcenter[1];
    //tri[2][2] = vertex[i+0+j+1][2] - dbcenter[2];
    tri[2][2] = vertex[i+0+j+1][2];

    getNormal( tri[0], tri[1], tri[2], norm );

    d = (tri[0][0] * norm[0]) +
        (tri[0][1] * norm[1]) +
        (tri[0][2] * norm[2]);

    d *= -1;
    pz = (-(norm[0] * x) - (norm[1] * y) - d)/norm[2];

    return z - pz;
}
```

A.7 hat.h

```
#ifndef __HAT_H
#define __HAT_H extern float Hat( float x,float y,float z );
#endif
```

A.8 movement.h

```
/* This .h file is responsible to the movement of the UAV in the
scene, with the CallBacks we get to move the UAVs in the scene*/

#include <osg/NodeCallback>
#include <osg/MatrixTransform>
#include <iostream>
#include <vector>
#include <math.h>
#include <osg/PositionAttitudeTransform>

using namespace std;

// We define a struct that contains the three coordinates and if the
// UAV is the protagonist

struct Coordinates
{
    double cx;
    double cy;
    double cz;
    int protagonist;

    Coordinates(void)
    {
        cx = 0;
        cy = 0;
        cz = 0;
        protagonist = 0;
    }

    Coordinates(const Coordinates &c)
    {
        cx = c.cx;
        cy = c.cy;
        cz = c.cz;
        protagonist = c.protagonist;
    }

    Coordinates &operator=(const Coordinates &c)
    {
        cx = c.cx;
        cy = c.cy;
        cz = c.cz;
        protagonist = c.protagonist;

        return *this;
    }
};

//Global variable that allows us to stop the UAV protagonist
extern int loitering_mode;

/* Now we are going to define a new class that will allow us to make
the movement of the UAVs using the callbacks. This class is a
NodeCallBack and OSG calls the operator()() method in your derived
class during each update traversal allowing your application to
modify the Node. Or what is the same when the viewer is updated the
method operator() is called.*/
```

```
    class Movement : public osg::NodeCallback
{
public:

    vector<Coordinates> UAV;
    int path;
    double counter;
    int c;
    int initiation;
    int endrot;
    double aux;
    int biggerpi;
    int smallerpi;
    double speed;
    double condition;

    virtual void operator()( osg::Node* node,
            osg::NodeVisitor* nv )
    {
        // Define the dynamic PAT that will allow us to move the UAVs
        osg::PositionAttitudeTransform* mtLeft =
                dynamic_cast<osg::PositionAttitudeTransform*>( node );

        // We check if the UAV is in loitering mode and if the UAV is
        // the protagonist, in that case, the UAV will be stopped until
        // the loitering-mode button will be pressed again.
        if ((loitering_mode == 0)||(UAV[0].protagonist == 0))
        {
            // It is necessary a initiation to guide the UAV succesfully
            if (initiation == 0)
            {
            _r=-atan2(UAV[path+1].cx-UAV[path].cx,UAV[path+1].cy-UAV[path].cy);
            distance = sqrt(
            (UAV[path+1].cx-UAV[path].cx)*(UAV[path+1].cx-UAV[path].cx)+
            (UAV[path+1].cy-UAV[path].cy)*(UAV[path+1].cy-UAV[path].cy)+
            (UAV[path+1].cz-UAV[path].cz)*(UAV[path+1].cy-UAV[path].cy));
            initiation=1;
            // We give a value to the speed
            speed = 400.0;
            }

            // We check that the distance traveled for the UAV is minor
            // than the necessary to reach to the next coordinate, if that
            // is the case the program will not enter in the "if" and will
            // execute the "else". With the "else", the UAV will continue
            // to advance  to the next point. When the UAV reaches the
            // next coordinate,the program will enter in the next "if"
            // and will do the necessary rotation to guide the UAV to
            // the next point.
            if (distance <= (speed*counter))
            {
                // This condition ensures that UAV keeps rotating until the
                // UAV reaches the desired orientation.
                if (endrot == 0)
                {
                    // It must be remembered that in the function
                    // Readcoordinates we added again the first coordinate,
                    // at the end of the UAV element. So, when UAV reaches
                    // the coordinates UAV.size -2, the UAV will be in
                    // the home position.
                    if (path == (int) UAV.size() -2)
                    {
```

```
            aux = -atan2(UAV[1].cx-UAV[0].cx,UAV[1].cy-UAV[0].cy);
        }
        else
        {
        aux = -atan2(
        UAV[path+2].cx-UAV[path+1].cx,UAV[path+2].cy-UAV[path+1].cy);
        }

        // The next commands are necessary to define the sense
        // of rotation and how much we need to rotate the UAV
        // until reach the desired rotation.
        if (((_r >= aux)||(biggerpi==1))&&(smallerpi==0))
        {
            if ((_r-osg::PI > aux)||(biggerpi == 1))
                {
                    biggerpi = 1;
                    _r += 0.005;
                    if (_r >= aux + 2*osg::PI)
                    {
                        _r = _r-2*osg::PI;
                        biggerpi = 0;
                    }

                }
            else
                {
                    _r += -0.005;
                }
            if ((_r <= aux)&&(biggerpi==0))
            {
                _r = aux;
                endrot = 1;
            }

        }
        else
        {
            if ((_r +osg::PI <= aux)||(smallerpi ==1))
            {
                smallerpi = 1;
                _r += -0.005;
                if (_r <= aux-2*osg::PI)
                {
                    _r = _r+2*osg::PI;
                    smallerpi = 0;
                }
            }
            else
            {
            _r += 0.005;
            }
            if ((_r >= aux)&&(smallerpi == 0))
            {
                _r = aux;
                endrot = 1;
            }

        }
    }
    else
    {
        // Once ended the rotation is necessary to define the
```

```
                    // new path, for safety we store the current position
                    // and equate to zero all variables.
                    UAV[path+1].cx=UAV[path].cx+_x;
                    UAV[path+1].cy=UAV[path].cy+_y;
                    UAV[path+1].cz=UAV[path].cz+_z;
                    path++;
                    counter = 0;
                    endrot = 0;
                    _x = 0;
                    _y = 0;
                    _z = 0;

                    // Is checked if th UAV is in the home position. In
                    // fact, if the UAV is in the home position, is
                    // necessary to equate to zero the variable "path".
                    if (path == (int) UAV.size() -1)
                    {
                        path = 0;
                    }
                    distance = sqrt(
                    (UAV[path+1].cx-UAV[path].cx)*(UAV[path+1].cx-UAV[path].cx)+
                    (UAV[path+1].cy-UAV[path].cy)*(UAV[path+1].cy-UAV[path].cy)+
                    (UAV[path+1].cz-UAV[path].cz)*(UAV[path+1].cy-UAV[path].cy));
                }
            }

            // While the UAV does not reach the following coordinate,
            //the variables x, y and z will be increased.
            else
            {
                _x += ((UAV[path+1].cx-UAV[path].cx)*speed)/distance;
                _y += ((UAV[path+1].cy-UAV[path].cy)*speed)/distance;
                _z += ((UAV[path+1].cz-UAV[path].cz)*speed)/distance;
                counter++;
            }

        // Now, we apply the necessary translation and rotation
        mtLeft->setPosition(osg::Vec3(UAV[path].cx+_x,UAV[path].cy+_y,
                                                        UAV[path].cz+_z));
        mtLeft->setAttitude( osg::Quat(_r,osg::Vec3(0.0,0.0,1.0)));

        // Continue traversing so that OSG can process
        // any other nodes with callbacks.
        traverse( node, nv );
        }
}

protected:
    double _x;
    double _y;
    double _z;
    double _h;
    double _r;
    double distance;
};
```

A.9 Readcoordinates.cpp

```
/* A very simple function that receive an array of variables type
Move and store, into them, the coordinates of each UAV loaded from
the coordinates files located in the same folder that the program.
Also, we receive the number of UAVs present in the scene with the
aim to know how many coordinates files we have to load.*/

#include "movement.h"
#include <iostream>

void readcoordinates (Movement* Move, int nUAVs)
{
  char line[30];
  FILE* file;
  // We define a table with the names of coordinates files located
  // in the same folder that the program.
  char name [10][20] = {
    {"Coordinates.txt"},
    {"Coordinates1.txt"},
    {"Coordinates2.txt"},
    {"Coordinates3.txt"},
    {"Coordinates4.txt"},
    {"Coordinates5.txt"},
    {"Coordinates6.txt"},
    {"Coordinates7.txt"},
    {"Coordinates8.txt"},
    {"Coordinates9.txt"},
};
  // This index allows us to travel the table
  int index = 0;
  // We define a variable of type Coordinates to store coordinates values
  Coordinates data;
  // Store data equal to the number of UAVs on the scene
  while (index < nUAVs)
  {
    file = fopen(name[index], "r" );
     if (file != NULL);
     {
        // We read line to line from coordinates file.
        while(fgets(line, sizeof line, file)!=NULL)
        {
            // Here we store the coordinates into data
            sscanf(line, "%lf %lf %lf", &data.cx, &data.cy, &data.cz);
            // And now, we use the properties of vectors to add at the end
            // the new coordinates
            Move[index].UAV.push_back(data);
        }
        // Like we start from the first coordinate I decided to add that
        // coordinate again at the end of the table to follow in a more
        // easy way the loop
        Move[index].UAV.push_back(Move[index].UAV[0]);
        if (index == 0)
        {
            // We define the protagonist.
            Move[index].UAV[0].protagonist = 1;
        }
     }
     index++;
  }
}
```

A.10 rotateHT.h

```
// This .h file contains the CallBack that allow us to move the camera of
// the UAV depending of Head-Tracking datas.

#include <osg/NodeCallback>
#include <osg/MatrixTransform>
#include <iostream>
#include <math.h>
#include "./head_tracking/cyarpheadpos.h"
#include "./head_tracking/cyarpheadposprocessor.h"


using namespace std;

// We define a global variable that will allow us to acces to the
// Head-Tracking system.
bool manuallyPlaceCamera = false;

// We define a variable type CyarpHeadPosProcessor that will receive
// data from the Head-Tracking thanks to yarp server.
CYarpHeadPosProcessor headposProcessor;

class RotateHT : public osg::NodeCallback { public:
    // This method will be called in each traverse update.
    virtual void operator()( osg::Node* node,
            osg::NodeVisitor* nv )
    {
        double roll, pitch, yaw, x, y, z;

        // With the next condition we will know if the Head-Tracking
        // Button was pressed and we are receiving data from
        // Head-Tracking system.
        if (manuallyPlaceCamera && headposProcessor.popHeadPos(roll, pitch,
                                                               yaw, x, y, z) == 0)
        {
        osg::MatrixTransform* HTmode =
        dynamic_cast<osg::MatrixTransform*>( node );
        osg::Matrix mR, mT;
        // Translation Matrix will be 0, because we want only the rotation
        mT.makeTranslate( 0., 0., 0. );
        mR.makeRotate(
                osg::DegreesToRadians(0.0), osg::Vec3(0,1,0), // roll, deactivated
                osg::DegreesToRadians(-pitch), osg::Vec3(1,0,0) , // pitch
                osg::DegreesToRadians(1.5*(-yaw)-33), osg::Vec3(0,0,1) ); // yaw

        // now define the transformation matrix
        HTmode->setMatrix( mR * mT );

        // Continue traversing so that OSG can process
        //   any other nodes with callbacks.
        traverse( node, nv );
        }
    }
};
```

A.11 terrain.cpp

```
/* This code is based on an OpenSceneGraph example, osghangglide.
With some changes. These functions allows us to built the terrain.*/

#include <osg/Geode>
#include <osg/Geometry>
#include <osg/Texture2D>
#include <osg/TexEnv>
#include <osg/StateSet>
#include <osgDB/ReadFile>
#include "terrain_coords.h"
#include "terrain_texcoords.h"
using namespace osg;

// We take the values of vertex from terrain_coords.h and calculate the
// center and the radius of the terrain.
void getDatabaseCenterRadius( float dbcenter[3], float *dbradius )
{
    int i;
    double n=0.0;
    double center[3] = { 0.0f, 0.0f, 0.0f };
    float  cnt;

    cnt = 39 * 38;
    for( i = 0; i < cnt; i++ )
    {
        center[0] += (double)vertex[i][0];
        center[1] += (double)vertex[i][1];
        center[2] += (double)vertex[i][2];

        n = n + 1.0;
    }

    center[0] /= n;
    center[1] /= n;
    center[2] /= n;

    float r = 0.0;

    for( i = 0; i < cnt; i++ )
    {
        double d = sqrt(
        (((double)vertex[i][0]-center[0])*((double)vertex[i][0]-center[0]))+
        (((double)vertex[i][1]-center[1])*((double)vertex[i][1]-center[1]))+
        (((double)vertex[i][2]-center[2])*((double)vertex[i][2]-center[2])));

        if( d > (double)r ) r = (float)d;

    }

    *dbradius = r;
    dbcenter[0] = (float)center[0];
    dbcenter[1] = (float)center[1];
    dbcenter[2] = (float)center[2];

    int index = 19 * 39 + 19;
    dbcenter[0] = vertex[index][0] - 0.15;
    dbcenter[1] = vertex[index][1];
    dbcenter[2] = vertex[index][2] + 0.35;
```

```
}

// This function return a geode with the terrain and its texture, the
// function receive like parameter an integer that says us if we must to
// load the airport texture or the field one.
Node *makeTerrain( int index )
{
    int m, n;
    int  i, j;
    float dbcenter[3];
    float dbradius;

    getDatabaseCenterRadius( dbcenter, &dbradius );

    m = (sizeof( vertex ) /(sizeof( float[3])))/39;
    n = 39;
    int cont = 0;
    int aux = 0;

    Vec3Array& v    = *(new Vec3Array(m*n));
    Vec2Array& t    = *(new Vec2Array(m*n));
    Vec4Array& col  = *(new Vec4Array(1));

    col[0][0] = col[0][1] = col[0][2] = col[0][3] = 1.0f;

    for( i = 0; i < m * n; i++ )
    {
        v[i][0] = vertex[i][0] - dbcenter[0];
        v[i][1] = vertex[i][1] - dbcenter[1];
        v[i][2] = vertex[i][2];
        t[i][0] = texcoord[i][0] + 0.02;
        t[i][1] = texcoord[i][1];
    }

    Geometry *geom = new Geometry;

    geom->setVertexArray( &v );
    geom->setTexCoordArray( 0, &t );

    geom->setColorArray( &col );
    geom->setColorBinding( Geometry::BIND_OVERALL );

    // In this loop we build the terrain using a triangle mesh.
    for( i = 0; i < m-2; i++ )
    {
        DrawElementsUShort* elements =
        new DrawElementsUShort(PrimitiveSet::TRIANGLE_STRIP);
        elements->reserve(39*2);
        for( j = 0; j < n; j++ )
        {
            elements->push_back((i+0)*n+j);
            elements->push_back((i+1)*n+j);
        }
        geom->addPrimitiveSet(elements);
    }

    Texture2D *tex = new Texture2D;

    // Is loaded the corresponding texture
    if (index == 0)
    {
        tex->setImage(osgDB::readImageFile("./Airport.rgb"));
```

```
    }
    else
    {
        tex->setImage(osgDB::readImageFile("./Field.rgb"));
    }
    StateSet *dstate = new StateSet;
    dstate->setMode( GL_LIGHTING, StateAttribute::OFF );
    dstate->setTextureAttributeAndModes(0, tex, StateAttribute::ON );
    dstate->setTextureAttribute(0, new TexEnv );

    geom->setStateSet( dstate );

    Geode *geode = new Geode;
    geode->addDrawable( geom );

    return geode;
}
```

A.12 terrain_coords.h and terrain_texcoords.h

The source code of this two elements will be omitted because in **terrain_coords.h** we have a element called *vertex* that is a matrix with 39x38x3 elements which corresponds to the coordinates of the terrain. On the other hand **terrain_texcoords** has correspondence between each point of the terrain and its associated texture.

A.13 trees.cpp

```
#include <stdlib.h>
#include <osg/Billboard>
#include <osg/Group>
#include <osg/Geode>
#include <osg/Geometry>
#include <osg/Texture2D>
#include <osg/TexEnv>
#include <osg/BlendFunc>
#include <osg/AlphaFunc>
#include <osgDB/ReadFile>
#include "hat.h"
#ifdef _MSC_VER
#pragma warning( disable : 4244 )
#pragma warning(disable : 4305 )
#endif

using namespace osg;

#define sqr(x)  ((x)*(x))

extern void getDatabaseCenterRadius( float dbcenter[3], float
*dbradius );

static float dbcenter[3], dbradius;

static struct _tree {
    int n;
    float x, y, z;
    float w, h;
}

trees[] = {
    {   0,  -0.4769,  -0.8972,  -0.4011,   0.2000,   0.1200 },
    {   1,  -0.2543,  -0.9117,  -0.3873,   0.2000,   0.1200 },
    {   2,  -0.0424,  -0.8538,  -0.3728,   0.2000,   0.1200 },
    {   3,   0.1590,  -0.8827,  -0.3594,   0.2000,   0.1200 },
    {   4,  -0.4981,  -1.0853,  -0.4016,   0.3500,   0.1200 },
    {   5,  -0.5405,  -1.2590,  -0.4050,   0.2000,   0.1200 },
    {   6,  -0.5723,  -1.5339,  -0.4152,   0.2000,   0.1200 },
    {   7,  -0.6252,  -1.8667,  -0.4280,   0.2000,   0.1200 },
    {   8,  -0.5617,  -2.1851,  -0.4309,   0.2000,   0.1200 },
    {   9,  -0.5087,  -2.4166,  -0.4215,   0.2000,   0.1200 },
    {  10,  -0.4345,  -2.3443,  -0.4214,   0.2000,   0.1200 },
    {  11,  -3.0308,  -1.5484,  -0.4876,   0.2000,   0.1200 },
    {  12,  -3.0202,  -1.6497,  -0.4963,   0.2000,   0.1200 },
    {  13,  -2.9355,  -1.8378,  -0.4969,   0.2000,   0.1200 },
    {  14,  -0.6040,  -2.0259,  -0.4300,   0.2000,   0.1200 },
    {  15,  -0.5442,  -1.3442,  -0.4080,   0.1000,   0.1200 },
    {  16,  -0.5639,  -1.6885,  -0.4201,   0.1000,   0.1200 },
    {  17,   0.9246,   3.4835,   0.5898,   0.2500,   0.1000 },
    {  18,   0.0787,   3.8687,   0.3329,   0.2500,   0.1200 },
    {  19,   0.2885,   3.7130,   0.4047,   0.2500,   0.1200 },
    {  20,   0.2033,   3.6228,   0.3704,   0.2500,   0.1200 },
    {  21,  -0.2098,   3.9015,   0.2327,   0.2500,   0.1200 },
    {  22,  -0.3738,   3.7376,   0.1722,   0.2500,   0.1200 },
    {  23,  -0.2557,   3.6064,   0.1989,   0.2500,   0.1200 },
    {  24,   0.0590,   3.7294,   0.3210,   0.2500,   0.1200 },
    {  25,  -0.4721,   3.8851,   0.1525,   0.2500,   0.1200 },
    {  26,   0.9639,   3.2048,   0.5868,   0.1200,   0.0800 },
```

```
{   27,   0.7082,  -1.0409,  -0.3221,   0.1000,   0.1000 },
{   28,  -0.2426,  -2.3442,  -0.4150,   0.1000,   0.1380 },
{   29,  -0.1770,  -2.4179,  -0.4095,   0.1000,   0.1580 },
{   30,  -0.0852,  -2.5327,  -0.4056,   0.1000,   0.1130 },
{   31,  -0.0131,  -2.6065,  -0.4031,   0.1000,   0.1150 },
{   32,   0.0787,  -2.6638,  -0.4012,   0.1000,   0.1510 },
{   33,   0.1049,  -2.7622,  -0.3964,   0.1000,   0.1270 },
{   34,   0.1770,  -2.8687,  -0.3953,   0.1000,   0.1100 },
{   35,   0.3213,  -2.9507,  -0.3974,   0.1000,   0.1190 },
{   36,   0.4065,  -3.0163,  -0.4014,   0.1000,   0.1120 },
{   37,   0.3738,  -3.1802,  -0.4025,   0.1000,   0.1860 },
{   38,   0.5508,  -3.2048,  -0.3966,   0.1000,   0.1490 },
{   39,   0.5836,  -3.3031,  -0.3900,   0.1000,   0.1670 },
{   40,  -0.3082,  -2.7212,  -0.3933,   0.1000,   0.1840 },
{   41,  -0.1967,  -2.6474,  -0.4017,   0.1000,   0.1600 },
{   42,  -0.1180,  -2.7458,  -0.3980,   0.1000,   0.1250 },
{   43,  -0.3344,  -2.8359,  -0.3964,   0.1000,   0.1430 },
{   44,  -0.2492,  -2.8933,  -0.3838,   0.1000,   0.1890 },
{   45,  -0.1246,  -3.0491,  -0.3768,   0.1000,   0.1830 },
{   46,   0.0000,  -3.0818,  -0.3696,   0.1000,   0.1370 },
{   47,  -0.2295,  -3.0409,  -0.3706,   0.1000,   0.1660 },
{   48,  -1.3245,   2.6638,   0.0733,   0.0500,   0.0500 },
{   49,   2.2425,  -1.5491,  -0.2821,   0.2300,   0.1200 },
{   50,   0.2164,  -2.1311,  -0.4000,   0.1000,   0.0690 },
{   51,   0.2885,  -2.2130,  -0.4000,   0.1000,   0.0790 },
{   52,   0.3606,  -2.2786,  -0.4000,   0.1000,   0.0565 },
{   53,   0.4328,  -2.3442,  -0.4000,   0.1000,   0.0575 },
{   54,   0.5246,  -2.4343,  -0.4086,   0.1000,   0.0755 },
{   55,   0.6360,  -2.5245,  -0.4079,   0.1000,   0.0635 },
{   56,   0.7541,  -2.4261,  -0.4007,   0.1000,   0.0550 },
{   57,   0.7934,  -2.2786,  -0.3944,   0.1000,   0.0595 },
{   58,   1.0295,  -2.2868,  -0.3837,   0.1000,   0.0560 },
{   59,   0.8459,  -2.6474,  -0.4051,   0.1000,   0.0930 },
{   60,   1.0426,  -2.6884,  -0.4001,   0.1000,   0.0745 },
{   61,   1.1475,  -2.7458,  -0.3883,   0.1000,   0.0835 },
{   62,  -0.1967,  -1.4180,  -0.3988,   0.1000,   0.0920 },
{   63,  -0.0131,  -1.2704,  -0.3856,   0.1000,   0.0690 },
{   64,   0.2098,  -1.2049,  -0.3664,   0.1000,   0.0790 },
{   65,   0.3410,  -1.3196,  -0.3652,   0.1000,   0.0565 },
{   66,   0.5705,  -1.2704,  -0.3467,   0.1000,   0.0575 },
{   67,   0.6360,  -1.4344,  -0.3532,   0.1000,   0.0755 },
{   68,   0.9246,  -1.4180,  -0.3329,   0.1000,   0.0635 },
{   69,   1.0623,  -1.3360,  -0.3183,   0.1000,   0.0550 },
{   70,   1.2393,  -1.3934,  -0.3103,   0.1000,   0.0595 },
{   71,   1.3639,  -1.4753,  -0.3079,   0.1000,   0.0560 },
{   72,   1.4819,  -1.5983,  -0.3210,   0.1000,   0.0930 },
{   73,   1.7835,  -1.5819,  -0.3065,   0.1000,   0.0745 },
{   74,   1.9343,  -2.1065,  -0.3307,   0.1000,   0.0835 },
{   75,   2.1245,  -2.3196,  -0.3314,   0.1000,   0.0920 },
{   76,   2.2556,  -2.3032,  -0.3230,   0.1000,   0.0800 },
{   77,   2.4196,  -2.3688,  -0.3165,   0.1000,   0.0625 },
{   78,   1.7835,  -2.5327,  -0.3543,   0.1000,   0.0715 },
{   79,   1.7180,  -2.8933,  -0.3742,   0.1000,   0.0945 },
{   80,   1.9343,  -3.0409,  -0.3727,   0.1000,   0.0915 },
{   81,   2.4524,  -3.4671,  -0.3900,   0.1000,   0.0685 },
{   82,   2.4786,  -2.8851,  -0.3538,   0.1000,   0.0830 },
{   83,   2.3343,  -2.6228,  -0.3420,   0.1000,   0.0830 },
{   84,   2.8130,  -2.0737,  -0.2706,   0.1000,   0.0890 },
{   85,   2.6360,  -1.8278,  -0.2661,   0.1000,   0.0975 },
{   86,   2.3958,  -1.7130,  -0.2774,   0.2000,   0.1555 },
{   87,   2.2688,  -1.2868,  -0.2646,   0.1000,   0.0835 },
{   88,   2.4196,  -1.1147,  -0.2486,   0.1000,   0.0770 },
```

```
    {   89,   2.7802,  -2.3933,  -0.3017,   0.1000,   0.0655 },
    {   90,   3.0163,  -2.4179,  -0.2905,   0.1000,   0.0725 },
    {   91,   2.9310,  -2.2540,  -0.2798,   0.1000,   0.0910 },
    {   92,   2.6622,  -2.0983,  -0.2823,   0.1000,   0.0680 },
    {   93,   2.3147,  -1.9753,  -0.2973,   0.1000,   0.0620 },
    {   94,   2.1573,  -1.8770,  -0.3013,   0.1000,   0.0525 },
    {   95,   2.0196,  -1.7868,  -0.3044,   0.1000,   0.0970 },
    {   96,   2.7802,  -3.3031,  -0.3900,   0.1000,   0.0510 },
    {   97,   2.8589,  -3.1720,  -0.3900,   0.1000,   0.0755 },
    {   98,   3.0163,  -2.8114,  -0.3383,   0.1000,   0.0835 },
    {   99,   3.5081,  -2.4179,  -0.2558,   0.1000,   0.0770 },
    {  100,   3.5277,  -2.3196,  -0.2366,   0.1000,   0.0765 },
    {  101,   3.6654,  -2.5819,  -0.2566,   0.1000,   0.0805 },
    {  102,   3.7179,  -2.7622,  -0.2706,   0.1000,   0.0980 },
    {  103,   3.7769,  -2.4671,  -0.2339,   0.1000,   0.0640 },
    {  104,   3.3441,  -2.4671,  -0.2693,   0.1000,   0.0940 },
    { -1, 0, 0, 0, 0, 0 },
};

static Geometry *makeTree( _tree *tree, StateSet *dstate )
{
    float vv[][3] =
    {
        { -tree->w/2.0, 0.0, 0.0 },
        {  tree->w/2.0,  0.0, 0.0 },
        {  tree->w/2.0, 0.0, 2 * tree->h },
        { -tree->w/2.0, 0.0, 2 * tree->h },
    };

    Vec3Array& v = *(new Vec3Array(4));
    Vec2Array& t = *(new Vec2Array(4));
    Vec4Array& l = *(new Vec4Array(1));

    int    i;

    l[0][0] = l[0][1] = l[0][2] = l[0][3] = 1;

    for( i = 0; i < 4; i++ )
    {
        v[i][0] = vv[i][0];
        v[i][1] = vv[i][1];
        v[i][2] = vv[i][2];
    }

    t[0][0] = 0.0; t[0][1] = 0.0;
    t[1][0] = 1.0; t[1][1] = 0.0;
    t[2][0] = 1.0; t[2][1] = 1.0;
    t[3][0] = 0.0; t[3][1] = 1.0;

    Geometry *geom = new Geometry;

    geom->setVertexArray( &v );

    geom->setTexCoordArray( 0, &t );

    geom->setColorArray( &l );
    geom->setColorBinding( Geometry::BIND_OVERALL );

    geom->addPrimitiveSet( new DrawArrays(PrimitiveSet::QUADS,0,4) );

    geom->setStateSet( dstate );
```

```
    return geom;
}


static float ttx, tty;

static int ct( const void *a, const void *b )
{
    _tree *ta = (_tree *)a;
    _tree *tb = (_tree *)b;

    float da = sqrtf( sqr(ta->x - ttx) + sqr(ta->y - tty) );
    float db = sqrtf( sqr(tb->x - ttx) + sqr(tb->y - tty) );

    if( da < db )
        return -1;
    else
        return 1;
}


Node *makeTrees( void )
{
    Group *group = new Group;
    int i;

    getDatabaseCenterRadius( dbcenter, &dbradius );
    struct _tree  *t;

    Texture2D *tex = new Texture2D;
    tex->setImage(osgDB::readImageFile("Images/tree0.rgba"));

    StateSet *dstate = new StateSet;

    dstate->setTextureAttributeAndModes(0, tex, StateAttribute::ON );
    dstate->setTextureAttribute(0, new TexEnv );

    dstate->setAttributeAndModes( new BlendFunc, StateAttribute::ON );

    AlphaFunc* alphaFunc = new AlphaFunc;
    alphaFunc->setFunction(AlphaFunc::GEQUAL,0.05f);
    dstate->setAttributeAndModes( alphaFunc, StateAttribute::ON );

    dstate->setMode( GL_LIGHTING, StateAttribute::OFF );

    dstate->setRenderingHint( StateSet::TRANSPARENT_BIN );

    int tt[] = { 15, 30, 45, 58, 72, 75, 93, 96, 105, -1 };
    int *ttp = tt;

    i = 0;
    while( i < 105 )
    {
        ttx = trees[i].x;
        tty = trees[i].y;
        qsort( &trees[i], 105 - i, sizeof( _tree ), ct );

        i += *ttp;
        ttp++;
    }

    t = trees;
```

```
    i = 0;
    ttp = tt;
    while( *ttp != -1 )
    {
        Billboard *bb = new Billboard;
        //int starti = i;

        for( ; i < (*ttp); i++ )
        {
            t->x -= 0.3f;
            float  h = Hat(t->x, t->y, t->z );
            Vec3 pos( t->x, t->y, t->z-h );
            Geometry *geom = makeTree( t, dstate );
            bb->addDrawable( geom, pos );
            t++;
        }
        group->addChild( bb );
        ttp++;
    }

    return group;
}
```

A.14 Viewergtk.cpp

```
#include "ViewerGtk.h"
#include <osgGA/TrackballManipulator>

namespace osgViewer
{

  ViewerGtk::ViewerGtk():
      Viewer(),
      m_is_running(false),
      m_frame_interval(30),
      m_run_priority(G_PRIORITY_DEFAULT),
      m_frame_step_rate(USE_REFERENCE_TIME),
      m_run_to_frame_number(0),
      m_timeout_source_id(0)
  {
  }

  ViewerGtk::ViewerGtk ( osg::ArgumentParser & arguments ) :
      Viewer ( arguments ),
      m_is_running(false),
      m_frame_interval(30),
      m_run_priority(G_PRIORITY_DEFAULT),
      m_frame_step_rate(USE_REFERENCE_TIME),
      m_run_to_frame_number(0),
      m_timeout_source_id(0)
  {
  }

  ViewerGtk::ViewerGtk ( const osgViewer::Viewer & viewer, const osg::CopyOp & copyop ) :
      Viewer ( viewer, copyop ),
      m_is_running(false),
      m_frame_interval(30),
      m_run_priority(G_PRIORITY_DEFAULT),
      m_frame_step_rate(USE_REFERENCE_TIME),
      m_run_to_frame_number(0),
      m_timeout_source_id(0)
  {
  }

  ViewerGtk::~ViewerGtk()
  {
    if ( this->is_running() ) this->stop();
  }

  bool ViewerGtk::isSameKindAs(const osg::Object* object) const
  {
    return dynamic_cast<const ViewerGtk*>(object)!=0;
  }

  const char* ViewerGtk::libraryName() const
  {
    return "osgGtk";
  }

  const char* ViewerGtk::className() const
  {
    return "ViewerGtk";
  }
```

```
GraphicsWindowGtk * ViewerGtk::setup_viewer_in_gtk_window (int width,int height)
{
  GraphicsWindowGtk* gw = new GraphicsWindowGtk ( width, height );
  this->setThreadingModel ( osgViewer::Viewer::SingleThreaded );
  this->getCamera()->setViewport ( 0, 0, width, height );
  this->getCamera()->setProjectionMatrixAsPerspective ( 30.0f, static_cast<double>
                    ( width ) /static_cast<double> ( height ), 1.0f, 10000.0f );
  this->getCamera()->setGraphicsContext ( gw );

  g_signal_connect(G_OBJECT(gw->gtk_widget()), "expose-event", G_CALLBACK(
                                ViewerGtk::on_graphics_window_expose_event), this );

  return gw;
}

int ViewerGtk::run()
{
  const char* str = getenv("OSG_RUN_FRAME_COUNT");
  if (str)
    m_run_to_frame_number = atoi(str);
  else
    m_run_to_frame_number = 0;

  return this->run_implementation();
}

int ViewerGtk::run_to_frame(int frame_num)
{
  m_run_to_frame_number = frame_num;
  return this->run_implementation();
}

int ViewerGtk::run_frames(int frames)
{
  if (frames <= 0 ) return 0;
  m_run_to_frame_number = getViewerFrameStamp()->getFrameNumber() + frames;
  return this->run_implementation();
}

void ViewerGtk::stop()
{
  if ( m_timeout_source_id )
  {
    g_source_remove(m_timeout_source_id);
    m_timeout_source_id = 0;
  }

  m_is_running = false;
}

bool ViewerGtk::is_running()
{
  return m_is_running;
}

double ViewerGtk::fps()
{
  return 1000.0 / m_frame_interval;
}

void ViewerGtk::set_fps(double fps)
{
```

```
  if ( m_frame_interval == 1000.0 / fps ) return;
  m_frame_interval = 1000.0 / fps;
  if ( this->is_running() ) this->connect_timer();
}

unsigned ViewerGtk::frame_interval_ms()
{
  return m_frame_interval;
}

void ViewerGtk::set_frame_interval_ms(unsigned interval)
{
  if ( m_frame_interval == interval ) return;
  m_frame_interval = interval;
  if ( this->is_running() ) this->connect_timer();
}

int ViewerGtk::run_priority()
{
  return m_run_priority;
}

void ViewerGtk::set_run_priority(int priority)
{
  if ( m_run_priority == priority ) return;
  m_run_priority = priority;
  if ( this->is_running() ) this->connect_timer();
}

double ViewerGtk::running_frame_step_rate()
{
  return m_frame_step_rate;
}

void ViewerGtk::set_running_frame_step_rate(double frame_rate)
{
  m_frame_step_rate = frame_rate;
}

int ViewerGtk::get_run_to_frame_number()
{
  return m_run_to_frame_number;
}

gboolean ViewerGtk::on_graphics_window_expose_event(GtkWidget* widget,
                                    GdkEventExpose * event, gpointer data)
{
  ViewerGtk* viewer = (ViewerGtk*)data;

  if (viewer->_done) return false;

  if (viewer->_firstFrame)
  {
    viewer->viewerInit();

    if (! viewer->isRealized())
    {
      viewer->realize();
    }

    viewer->_firstFrame = false;
  }
```

```
    // This is the only piece that is different from ViewerBase::frame()
//      viewer->advance(simulationTime);

    viewer->eventTraversal();
    viewer->updateTraversal();
    viewer->renderingTraversals();

    return false;
  }

  bool ViewerGtk::on_run_step()
  {
    // Should we stop or run one frame?
    if ( m_run_to_frame_number and getViewerFrameStamp()->getFrameNumber() >=
                                                          m_run_to_frame_number)
    {
      this->stop();
      return 0;
    }
    else
    {
      // Step the frame
      this->frame(m_frame_step_rate);
    }

    return TRUE;
  }

  gboolean ViewerGtk::on_run_step_proxy(gpointer data)
  {
    if ( not data ) return FALSE;
    ViewerGtk* vgtk = (ViewerGtk*)data;
    return vgtk->on_run_step();
  }

  void ViewerGtk::connect_timer()
  {
    // Disconnect the old timer
    if ( m_timeout_source_id )
      g_source_remove(m_timeout_source_id);

    // Connect the new timer
    m_timeout_source_id = g_timeout_add_full( m_run_priority,
                                              m_frame_interval,
                                              on_run_step_proxy,
                                              this,
                                              NULL
                                            );
  }

  int ViewerGtk::run_implementation()
  {
    if (!getCameraManipulator() && getCamera()->getAllowEventFocus())
      setCameraManipulator(new osgGA::TrackballManipulator());

    if (!isRealized())
      realize();

    m_is_running = true;

    this->connect_timer();
```

```
    // If we didn't establish a connection to the timer, return an error
    // according to ViewerBase::run() semantics
    if ( not m_timeout_source_id ) return -1;

    return 0;
  }

}
```

A.15 Viewergtk.h

```
#ifndef OSGVIEWERVIEWERGTK_H
#define OSGVIEWERVIEWERGTK_H

#include <osgViewer/Viewer>
#include <GraphicsWindowGtk.h>

namespace osgViewer
{
  class ViewerGtk : public Viewer
  {
    public:
      ViewerGtk();

      ViewerGtk ( osg::ArgumentParser& arguments );

      ViewerGtk ( const osgViewer::Viewer& viewer,
            const osg::CopyOp& copyop=osg::CopyOp::SHALLOW_COPY );

      // If running, stops the viewer
      virtual ~ViewerGtk();

      virtual bool isSameKindAs(const osg::Object* object) const;

      virtual const char* libraryName() const;

      virtual const char* className() const;

      // Sets up this viewer to run in a Gtk window of the specified width
      // and height
      GraphicsWindowGtk* setup_viewer_in_gtk_window(int width,int height);

      // Starts the viewer running at the currently set frame interval and
      // priority. Runs until stopped
      virtual int run();

      // Runs the viewer until the viewer's OSG frame stamp reaches @a
      // frame_num
      int run_to_frame(int frame_num);

      // Runs the viewer until the viewer's OSG frame stamp reaches the
      // current frame stamp + @a frames
      int run_frames(int frames);

      // Stops the viewer from running
      virtual void stop();

      // True if the viewer is running, false otherwise
      bool is_running();

      // Returns the frames-per-second rate based on the currently set
      // frame interval
      double fps();

      /**
       * Sets the frames per second update rate
       *
       * Since the update interval has a resolution of 1ms, the update interval
       * may not be exactly the same as the set rate.
       *
```

```
        * For example, if set_fps(60) is called the actual update rate will not
        * be 16.66667 ms (1000 ms / 60). Instead, the update rate will be the
        * floor of 1000/60 or 16ms. Thus, the actual frame rate will be 62.5 fps.
        */
       void set_fps(double fps);

       // Returns the frame interval in milliseconds that will be used when the
       // viewer is running
       unsigned frame_interval_ms();

       // Sets the frame interval in milliseconds that will be used when the
       // viewer is running
       void set_frame_interval_ms(unsigned interval);

       // Returns the run priority that will be used when the viewer is running
       int run_priority();

       // Sets the run priority that will be used when the viewer is running
       void set_run_priority( int priority );

       /**
        * Returns the frame step rate that will be used when the viewer is running.
        * This is the value that will be passed to osgViewer::Viewer::advance() and
        * controls the rate at which the viewer is advanced.
        */
       double running_frame_step_rate();

       // Sets the frame step rate that will be used when the viewer is running.
       void set_running_frame_step_rate(double frame_rate);

       // Returns the frame number that the viewer will run to
       int get_run_to_frame_number();

     protected:
       bool m_is_running;
       unsigned m_frame_interval;
       int m_run_priority;
       double m_frame_step_rate;
       int m_run_to_frame_number;
       guint m_timeout_source_id;

       static gboolean on_graphics_window_expose_event(GtkWidget* widget,
                                        GdkEventExpose* event, gpointer data);

       virtual bool on_run_step();

       void connect_timer();

       virtual int run_implementation();

     private:

       // Proxy callback point that passes the callback to the virtual on_run_step()
       static gboolean on_run_step_proxy(gpointer data);

   };
}

#endif
```

A.16 Translator.cpp

```
\newpage
/* This function is responsible to translate the coordinates of the
original terrain to ones more according to our airport. Original
coordinates was very uneven and an airport needs a flat terrain.
Also this function is responsible that the replicas of the terrain
has the same height (same coordinate z) in their ends with the aim
to get a more realistic experience.*/

#include <iostream>
#include <math.h>
#include <stdlib.h>
#include <time.h>

using namespace std;

struct Coordinates {
    float cx;
    float cy;
    float cz;

    Coordinates(void)
    {
        cx = 0;
        cy = 0;
        cz = 0;
    }

    Coordinates(const Coordinates &c)
    {
        cx = c.cx;
        cy = c.cy;
        cz = c.cz;
    }

    Coordinates &operator=(const Coordinates &c)
    {
        cx = c.cx;
        cy = c.cy;
        cz = c.cz;

        return *this;
    }
};

int main () {
    char line[100];
  FILE* file;
  FILE* file2;
  char name [20] = "prueba.txt";
  char name1 [20] = "pruebaout.txt";
  file = fopen(name, "r" );
  file2 = fopen(name1, "w");
  float aux1 = 1.75;
  int contaux = 0;
  int contcol = 0;
  int contraw = 0;
  srand ( time(NULL) );
```

```
Coordinates data;
if (file != NULL);
{
  while(fgets(line, sizeof line, file)!=NULL)
  {
      sscanf(line, "    {%f,%f,%f},", &data.cx, &data.cy, &data.cz);
      if (contraw == 39)
      {
          fprintf(file2,"\n");
          contraw = 0;
          contcol++;

      }
      else
      {
          fprintf(file2,"    {%f,%f,   %f},\n", data.cx, data.cy,aux1);
          contraw++;
          if ((contcol == 0)||(contcol >=36))
          {
              aux1 = 1.75;
          }

          else if ((contcol >= 1)&&(contcol <= 7))
          {
              if (contraw < 5)
              {
                  aux1 += 0.009*contcol;
              }
              else if ((contraw >= 5)&&(contraw < 10))
              {
                  aux1 -= 0.008*contcol;
              }
              else if ((contraw >= 10)&&(contraw <= 18))
              {
                  aux1 += 0.007*(rand() % 10 - 5);
                  if (aux1 > 2.2)
                  {
                      aux1 = 2.2;
                  }
              }
              else if ((contraw >= 18)&&(contraw <= 30))
              {
                  aux1 = 1.75;
              }
              else if ((contraw >= 31)&&(contraw <= 34))
              {
                  aux1 += 0.006*contcol;
              }
              else if ((contraw > 34)&&(contraw <37))
              {
                  aux1 -= 0.005*contcol;
              }
              else if (contraw >= 37)
              {
                  aux1 = 1.75;
              }
          }
          else if ((contcol >= 7)&&(contcol <= 14))
          {
              if (contraw < 4)
              {
```

```
            aux1 -= 0.0006*contcol;
        }
        else if ((contraw >= 5)&&(contraw <= 18))
        {
            aux1 += 0.007*(rand() % 10 - 5);
            if (aux1 > 2.2)
            {
                aux1 = 2.2;
            }
        }
        else if ((contraw >= 19)&&(contraw <= 30))
        {
            aux1 = 1.75;
        }
        else if ((contraw >= 31)&&(contraw <= 34))
        {
            aux1 -= 0.003*contcol;
        }
        else if ((contraw > 34)&&(contraw <37))
        {
            aux1 += 0.002*contcol;
        }
        else if (contraw >= 37)
        {
            aux1 = 1.75;
        }
    }
    else if ((contcol >= 14)&&(contcol <= 30))
    {
        if (contraw < 4)
        {
            aux1 -= 0.003*contcol;
        }
        else if ((contraw >= 4)&&(contraw <= 30))
        {
            aux1 = 1.75;
        }
        else if ((contraw >= 31)&&(contraw <= 34))
        {
            aux1 -= 0.003*contcol;
        }
        else if ((contraw > 34)&&(contraw <37))
        {
            aux1 += 0.002*contcol;
        }
        else if (contraw >= 37)
        {
            aux1 = 1.75;
        }
    }
    else if ((contcol >= 31)&&(contcol < 37))
    {
        if (contraw < 5)
        {
            aux1 -= 0.001*contcol;
        }
        else if ((contraw >= 5)&&(contraw < 34))
        {
            aux1 = 1.75;
        }
        else if ((contraw > 34)&&(contraw <37))
        {
```

```
                    aux1 += 0.0025*contcol;
                }
                else if (contraw >= 37)
                {
                    aux1 = 1.75;
                }
            }
        }
    }
  }
}
```

A.17 Head Tracking system source code

Moreover all code seen above there are some code files more related to Head Tracking system that were performed by Fernando and Ivan. These code files allow to my simulation tool to use the Head Tracking system in it. All the code necessary to implement the Head Tracking is in a folder named head-tracking in the same directory as the files seen before.

A.18 makefile

Finally the makefile that allow us to compile the program can be seen below:

```
CSRCS = main.cpp ViewerGtk.cpp GraphicsWindowGtk.cpp hat.cpp
terrain.cpp trees.cpp Readcoordinates.cpp Createviews.cpp
Createscene.cpp CFLAGS = -g -c -O2 -DLINUX

INCLUDE = -I. -I/usr/include/ -I/usr/include/X11/
-I/usr/local/include/GL INCOSG =
-I/usr/local/OSG/OSG2.0/OpenSceneGraph/include INCGTK =
-I/usr/include/gtk-2.0 -I/usr/lib/gtk-2.0/include
-I/usr/include/atk-1.0 -I/usr/include/cairo -I/usr/include/pango-1.0
-I/usr/include/glib-2.0 -I/usr/lib/glib-2.0/include
-I/usr/include/pixman-1 -I/usr/include/freetype2
-I/usr/include/libpng12 -I/usr/include/gtkglext-1.0
-I/usr/lib/gtkglext-1.0/include
-I/usr/local/include/osgGtk-0.1/osgGtk LDLIBS = -lm -ldl -lGL -lGLU
-lpthread -lXext -lX11 -lYARP_OS -lACE

LDFLAGS = -L. -L/usr/lib -L/usr/X11R6/lib -L/usr/local/lib LDOSG =
-L/usr/local/OSG/OSG2.0/OpenSceneGraph/lib/Linux32 -losg -losgViewer
-losgSim OSGGTK_LIBS = -Wl,--export-dynamic -L/usr/local/lib
-lgtkglext-x11-1.0 -lgdkglext-x11-1.0 -lGLU -lGL -lXmu -lXt -lSM
-lICE -lgtk-x11-2.0 -lpangox-1.0 -lX11 -lgdk-x11-2.0 -latk-1.0
-lpangoft2-1.0 -lgdk_pixbuf-2.0 -lm -lpangocairo-1.0 -lgio-2.0
-lcairo -lpango-1.0 -lfreetype -lz -lfontconfig -lgobject-2.0
-lgmodule-2.0 -lglib-2.0 -losg -losgDB -losgFX -losgGA -losgParticle
-losgSim -losgText -losgUtil -losgTerrain -losgManipulator
-losgViewer -losgWidget -losgShadow -losgAnimation -losgVolume
-lOpenThreads

COBJS=$(patsubst %.cpp,%.o,$(CSRCS))

HTOBJS = cyarpheadpos.o cyarpheadposprocessor.o

CC = g++

EXE=osex

all: $(EXE)

$(EXE): $(COBJS) $(HTOBJS)
    cc -o$(EXE) $(COBJS) $(HTOBJS) $(INCLUDE) $(INCOSG)
    $(LDFLAGS) $(LDLIBS) $(LDOSG) $(INCGTK) $(OSGGTK_LIBS)


$(COBJS): %.o : %.cpp
    $(CC) $(CFLAGS) $(INCLUDE) $(INCOSG) $(INCGTK) -o $@ $<

cyarpheadpos.o: ./head_tracking/cyarpheadpos.cpp
./head_tracking/cyarpheadpos.h
$(CC) -c ./head_tracking/cyarpheadpos.cpp $(CFLAGS) $(INCLUDE)

cyarpheadposprocessor.o:
./head_tracking/cyarpheadposprocessor.cpp
./head_tracking/cyarpheadposprocessor.h
$(CC) -c ./head_tracking/cyarpheadposprocessor.cpp $(CFLAGS) $(INCLUDE)
```