ArcGIS SDK para QT

QML bases

Archivo principal de QML

❖Main.qml

```
ApplicationWindow {
    visible: true
    width: 80
    height: 60
```

El componente “ApplicationWindow” define las dimensiones de la ventana de nuestra app.

```
    QMLForm {
        anchors.fill: parent
    }
}
```

Nos redirecciona al archivo QMLForm.qml

```
16
17  ApplicationWindow {
18      visible: true
19      width: 80
20      height: 60
21
22
23      QMLForm {
24          anchors.fill: parent
25      }
26  }
27
```

Archivo principal de C++

- Main.cpp

```
// Register the map view for QML
  qmlRegisterType<MapQuickView>("Esri.QML", 1, 0,
"MapView");

  // Register the QML (QQuickItem) for QML
  qmlRegisterType<QML>("Esri.QML", 1, 0, "QML");

  // Initialize application view
  QQmlApplicationEngine engine;

  // Add the import Path

engine.addImportPath(QDir(QCoreApplication::application
DirPath()).filePath("qml"));

  // Set the source
  engine.load(QUrl("qrc:/qml/main.qml"));

  return app.exec();
```

Con la librería QQmlApplicationEngine creamos una variable (en este caso llamada engine) en la que instanciamos el motor que va a cargar el archivo main.qml

```
#include "QML.h"

#include "ArcGISRuntimeEnvironment.h"
#include "MapQuickView.h"

#include <QDir>
#include <QGuiApplication>
#include <QQmlApplicationEngine>
```

Archivo ...Form.qml

- QMLForm.qml

```
Item {

  // Create MapQuickView here, and create its Map etc. in
C++ code
  MapView {
    id: view
    anchors.fill: parent
    // set focus to enable keyboard navigation
    focus: true

  }

  // Declare the C++ instance which creates the map etc.
and supply the view
  QML {
    id: model
    mapView: view



  }
}
```

Establecemos las vistas del mapa

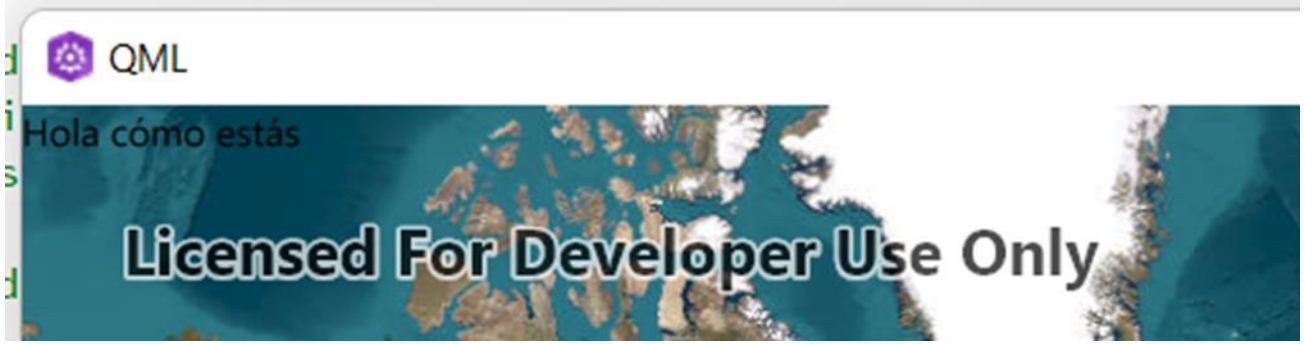
Establecemos las vistas del mapa
con las instancias de C++ creadas

Archivo ...Form.qml

- QMLForm.qml
- Crear saludo con el componente Text

```
MapView {
    id: view
    anchors.fill: parent
    // set focus to enable keyboard navigation
    focus: true

    Text {
      id: saludo
      text: qsTr("Hola cómo estás")
    }

  }
```

Se ha de mostrar en la vista de nuestro mapa



Archivo ...Form.qml

- QMLForm.qml
- Crear saludo con el componente Text

```
MapView {
    id: view
    anchors.fill: parent
    // set focus to enable keyboard navigation
    focus: true

    Text {
        Text
        F1
        ____________________________________
        Specifies how to add formatted text to a scene.
    }

}
```

Si damos click al cuadro de dialogo (es decir el icono en forma de bombillo) podemos modificar los parámetros de este componente

Archivo ...Form.qml

- QMLForm.qml
- Crear saludo con el componente Text

Hemos cambiado el estilo del texto

```
Text {
        id: saludo
        color: "#0edfe6"
        text: qsTr("Hola cómo estás")
        font.bold: true
        styleColor: "#fd4141"
        verticalAlignment: Text.AlignVCenter
        horizontalAlignment: Text.AlignHCenter
        style: Text.Sunken
        font.pointSize: 20
    }
```

Archivo ...Form.qml

- QMLForm.qml
- Crear saludo con el componente Text

Si lo anclamos al ancho de la ventana principal, el texto se centrará en la mitad del mapa

```
Text {
        id: saludo
        color: "#0edfe6"
        text: qsTr("Hola cómo estás")
        font.bold: true
        styleColor: "#fd4141"
        verticalAlignment: Text.AlignVCenter
        horizontalAlignment: Text.AlignHCenter
        style: Text.Sunken
        font.pointSize: 20
        anchors.fill: parent
        //font.pixelSize: window.width/100*4
    }
```

```
MouseArea{
   anchors.fill: parent
   onClicked:{
      saludo.cambiarTexto()
   }
}

Text {
   id: saludo
   color: "#0edfe6"
   text: qsTr("Hola cómo estás")
   font.bold: true
   styleColor: "#fd4141"
   verticalAlignment: Text.AlignVCenter
   horizontalAlignment: Text.AlignHCenter
   style: Text.Sunken
   font.pointSize: 20
   anchors.fill: parent

   function cambiarTexto(){
   text = "El texto acaba de cambiar"}
}
```

Empleamos el componente MouseArea y el disparados onClicked para ejecutar nuestra función una vez demos click en el mapa.

Las funciones las llamamos a partir del identificador que hemos creado y del nombre que le asignamos

Creamos una función para cambiar el texto. Está función está escrita en js

Archivo ...Form.qml

- QMLForm.qml
- Crear un rectangulo

Características del rectangulo

```
Rectangle{
        id: rectangulo
        //anchors. centerIn: parent
        color: "#da0bf5"
        border.color: "black"
        width: 70
        height: 80
        x: 10
        y: 50
    }
```

Archivo ...Form.qml

- QMLForm.qml
- Crear un rectangulo

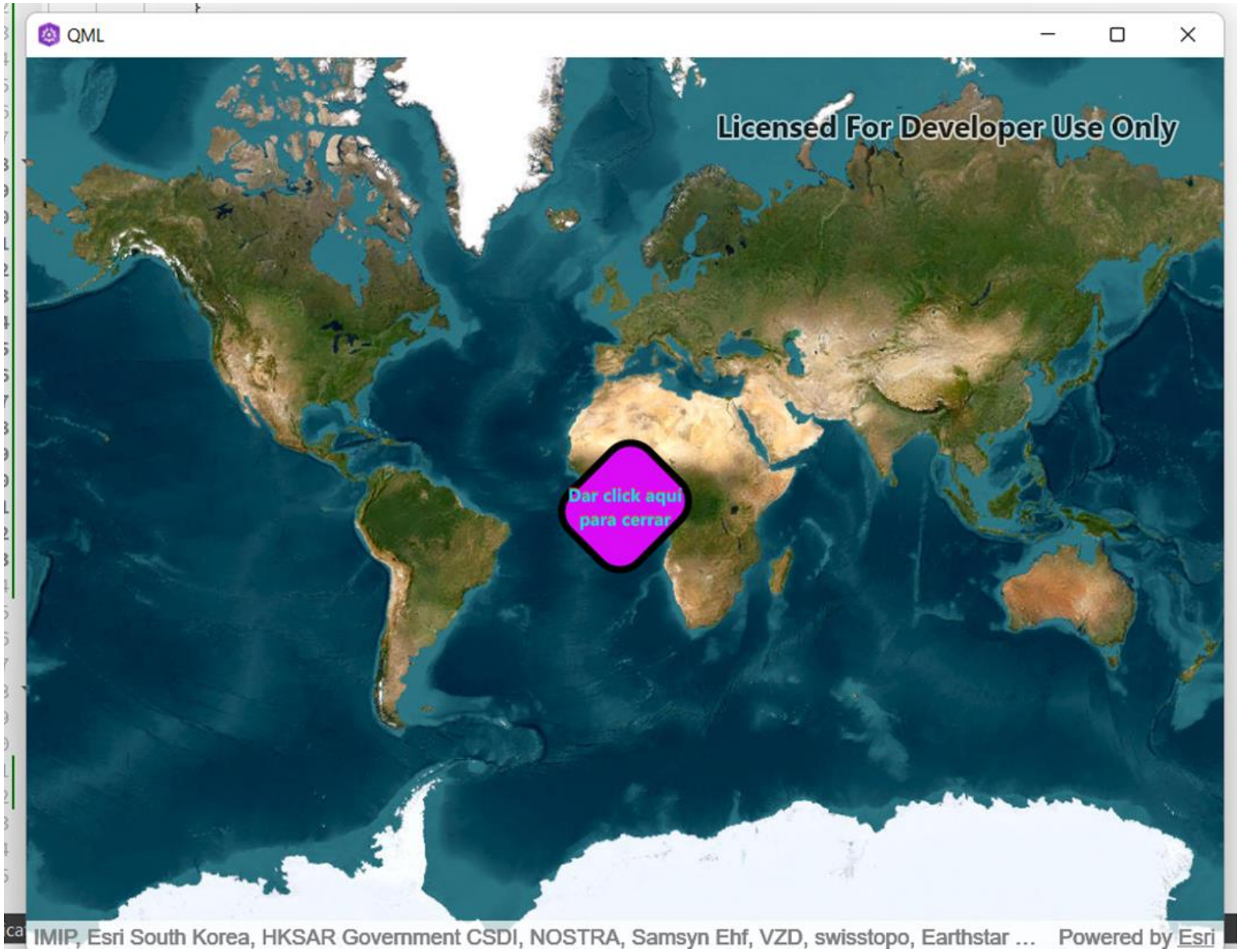
```
Rectangle{
        id: rectangulo
        anchors.centerIn: parent
        color: "#da0bf5"
        border.color: "black"
        width: 70
        height: 80
        //x: 10
        //y: 50
        radius: 20
        border.width: 5
    }
```

Archivo ...Form.qml

- QMLForm.qml
- Integrar el componente MouseArea al rectangulo

```
Rectangle{
        id: rectangulo
        anchors.centerIn: parent
        color: "#da0bf5"
        border.color: "black"
        width: 70
        height: 80
        //x: 10
        //y: 50
        radius: 20
        border.width: 5
        rotation: 45

        MouseArea{
           anchors.fill: parent
           onClicked: Qt.quit()
        }
    }
```

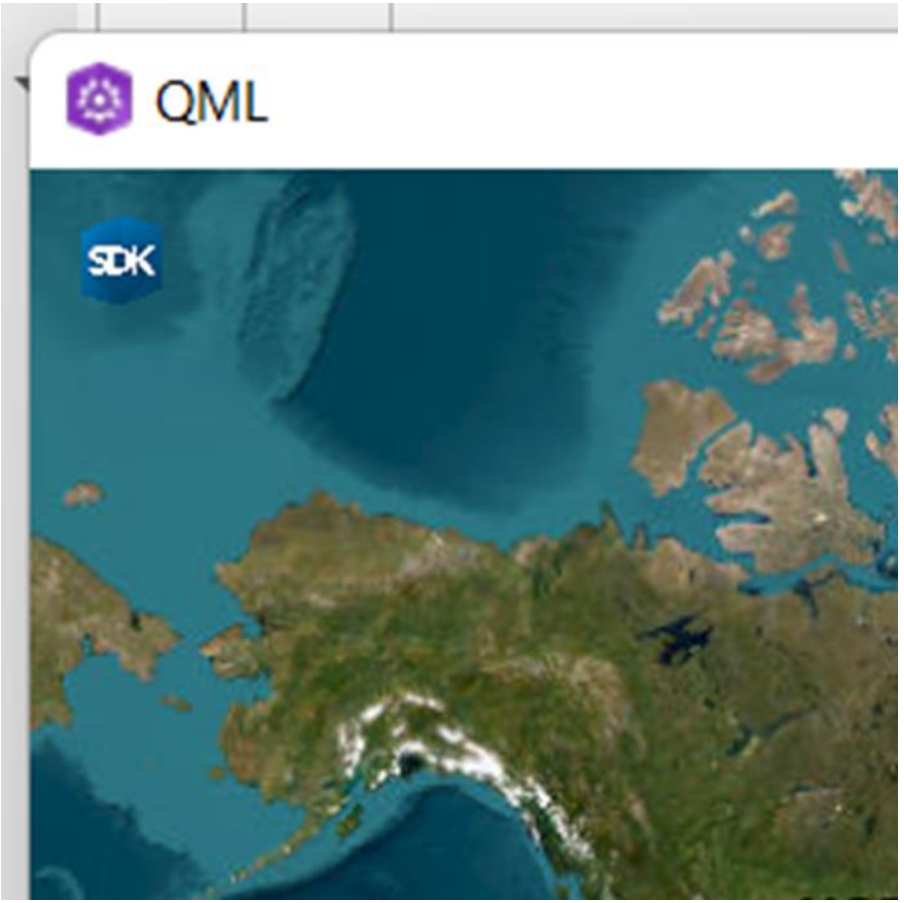
```
Text {
        id: saludo
        color: "#0edfe6"
        text: qsTr("Dar click aquí <br>para cerrar")
        font.bold: true
        styleColor: "#fd4141"
        verticalAlignment: Text.AlignVCenter
        horizontalAlignment: Text.AlignHCenter
        style: Text.Sunken
        //font.pointSize: 20
        anchors.fill: parent
        font.pixelSize: window.width/100*4

        //function cambiarTexto(){
        //text = "El texto acaba de cambiar"}
    }
```


QML
Licensed For Developer Use Only
Dar click aqui
para cerrar
IMIP, Esri South Korea, HKSAR Government CSDI, NOSTRA, Samsyn Ehf, VZD, swisstopo, Earthstar ...
Powered by Esri

Archivo ...Form.qml

- QMLForm.qml
- Componente Image

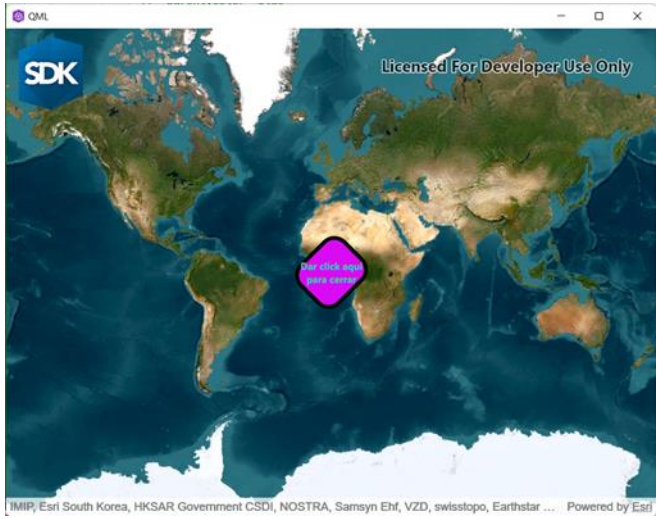
```
Image {
        id: logo
        source: "qrc:/Resources/AppIcon.png"
        width: 20
        height: 20
        x:10
        y:10
    }
```

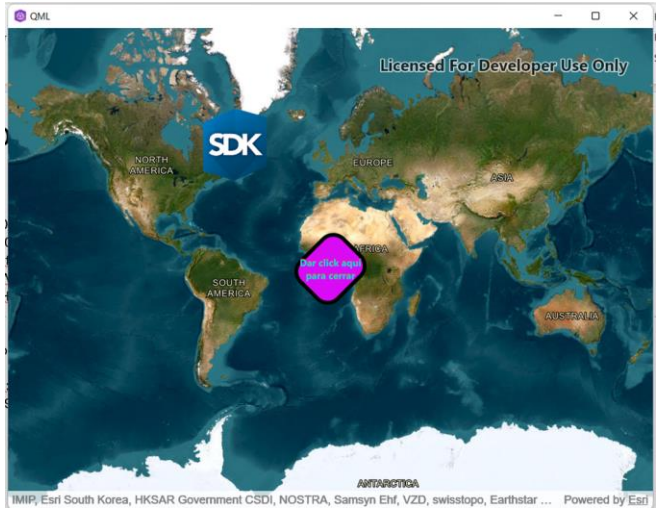


Archivo ...Form.qml

- QMLForm.qml
- Arrastrar y mover objetos con el componente Flickable

```
Flickable{
        width: 300
        height: 300
        anchors.fill: parent
        contentWidth: logo.sourceSize.width -200
        contentHeight: logo.sourceSize.height -200
       //boundsBehavior: Flickable.StopAtBounds
        //interactive: true

        Image {
          id: logo

          source: "qrc:/Resources/AppIcon.png"
          width: 90
          height: 90
          x:10
          y:10
        }
    }
```





Mostrar la hora en nuestra app

Archivo .h

```
class QML : public QObject
{
   Q_OBJECT

   Q_PROPERTY(Esri::ArcGISRuntime::MapQuickView *mapView READ
mapView WRITE setMapView NOTIFY
            mapViewChanged)

public:
   explicit QML(QObject *parent = nullptr);

   ~QML() override;
   Q_INVOKABLE QString tiempo();

signals:
   void mapViewChanged();

//public slots:
   //QString tiempo();

private:
   Esri::ArcGISRuntime::MapQuickView *mapView() const;
   void setMapView(Esri::ArcGISRuntime::MapQuickView *mapView);

   Esri::ArcGISRuntime::Map *m_map = nullptr;
   Esri::ArcGISRuntime::MapQuickView *m_mapView = nullptr;
};
```

Declaramos la función como un objeto invocable o como una ranura

Archivo main.cpp

```
mlRegisterType<MapQuickView>("Esri.QML", 1, 0,
"MapView");

   // Register the QML (QQuickItem) for QML
   qmlRegisterType<QML>("Esri.QML", 1, 0, "QML");

   // Initialize application view
   QQmlApplicationEngine engine;

   // Add the import Path

engine.addImportPath(QDir(QCoreApplication::application
DirPath()).filePath("qml"));

   engine.rootContext()->setContextProperty("hora", new
QML);
   // Set the source
   engine.load(QUrl("qrc:/qml/main.qml"));
```

Establecemos un contexto, para ello debemos emplear la librería “#include <QQmlContext>”.

En new QML hace referencia al nombre de nuestra clase

Archivo ...cpp

```
QML::QML(QObject *parent /* = nullptr */)
    : QObject(parent)
    , m_map(new Map(BasemapStyle::ArcGISImagery, this))
{}

QString QML::tiempo(){

    return QDateTime::currentDateTime().toString("hh:mm:ss");
}
```

Definimos nuestra función

Archivo ...Form.qml

```
Timer{
        running: true
        repeat: true
        interval: 1000
        onTriggered: saludo.text=hora.tiempo()

    }

    Text {
        id: saludo
        color: "#0edfe6"
        text: qsTr("Dar click aquí <br>para cerrar")
        font.bold: true
        styleColor: "#fd4141"
        verticalAlignment: Text.AlignVCenter
        horizontalAlignment: Text.AlignHCenter
        style: Text.Sunken
        //font.pointSize: 20
        //anchors.fill: parent
        font.pixelSize: ApplicationWindow.width/100*4

    }
```

Cambiar vistas

Archivo ...h

```
public:
   explicit CambiarVista(QObject *parent = nullptr);
   ~CambiarVista() override;

   Q_INVOKABLE void cambiarVistasQML(QString viewpoint);



         private:
            Esri::ArcGISRuntime::MapQuickView *mapView() const;
            void setMapView(Esri::ArcGISRuntime::MapQuickView *mapView);
            double screenRatio() const;
            Esri::ArcGISRuntime::Map *m_map = nullptr;
            Esri::ArcGISRuntime::MapQuickView *m_mapView = nullptr;
            int m_rotationValue = 0;

         };
```

Archivo main.cpp

```
#include <QQmlContext>
```

En esta línea se registran los objetos C++

```
// Register the map view for QML
qmlRegisterType<MapQuickView>("Esri.cambiarVista", 1, 0, "MapView");

// Register the CambiarVista (QQuickItem) for QML
qmlRegisterType<CambiarVista>("Esri.cambiarVista", 1, 0, "CambiarVista");
```

Archivo ...cpp

```
#include "Point.h"
#include "Viewpoint.h"
#include "SpatialReference.h"
#include "Envelope.h"
#include <QFuture>
```

Archivo ...cpp

```
CambiarVista::~CambiarVista() {}

void CambiarVista::cambiarVistasQML(QString viewpoint){
  if (viewpoint == "Center")
  {
    Point ptEsriHeadquarters(-117.195681,34.056218, SpatialReference::wgs84());
    m_mapView->setViewpointCenterAsync(ptEsriHeadquarters);
  }
  else if (viewpoint == "Center and scale")
  {
    Point ptHawaii(-157.564, 20.677, SpatialReference::wgs84());
    m_mapView->setViewpointCenterAsync(ptHawaii, 4000000.0);
  }
  else if (viewpoint == "Geometry")
  {
    Envelope envBeijing(116.380, 39.920, 116.400, 39.940, SpatialReference::wgs84());
    m_mapView->setViewpointGeometryAsync(envBeijing);
  }
  else if (viewpoint == "Geometry and padding")
  {
    Envelope envBeijing(116.380, 39.920, 116.400, 39.940, SpatialReference::wgs84());
    m_mapView->setViewpointGeometryAsync(envBeijing, 200 * screenRatio());
  }
  else if (viewpoint == "Rotation")
  {
    m_rotationValue = (m_rotationValue + 45) % 360;
    m_mapView->setViewpointRotationAsync(m_rotationValue);
  }
  else if (viewpoint == "Scale 1:5,000,000")
  {
    m_mapView->setViewpointScaleAsync(5000000.0);
  }
  else if (viewpoint == "Scale 1:10,000,000")
  {
    m_mapView->setViewpointScaleAsync(10000000.0);
  }


}
```

Archivo ...cpp

```
void CambiarVista::setMapView(MapQuickView *mapView)
{
   if (!mapView || mapView == m_mapView) {
      return;
   }

   m_mapView = mapView;
   m_mapView->setMap(m_map);

   emit mapViewChanged();
}
double CambiarVista::screenRatio() const
{
   const double width = static_cast<double>(m_mapView-
>width());
   const double height = static_cast<double>(m_mapView-
>height());
   return height > width ? width / height : height / width;
}
```

Archivo ...Form.qml

```
Item {

  // Create MapQuickView here, and create its Map etc. in C++ code
  MapView {
    id: view
    anchors.fill: parent
    // set focus to enable keyboard navigation
    focus: true

    Component.onCompleted: {
            // Set the focus on MapView to initially enable keyboard navigation
            forceActiveFocus();
          }
  }

  ComboBox {,,,}
```

Archivo ...Form.qml

```
ComboBox {
        id: comboBoxViewpoint
        anchors {
           left: parent.left
           top: parent.top
           margins: 15
        }

        // Add a background to the ComboBox
        Rectangle {
           anchors.fill: parent
           radius: 10
           // Make the rectangle visible if a dropdown indicator exists
           // An indicator only exists if a theme is set
           visible: parent.indicator
           border.width: 1
        }
        property int bestWidth: implicitWidth
        width: bestWidth + rightPadding + leftPadding + 20
        model: [ "Center", "Center and scale", "Geometry","Geometry and padding", "Rotation","Scale 1:5,000,000",
           "Scale 1:10,000,000" ]

        onCurrentTextChanged: {
                // Call C++ invokable function to change the viewpoint
                model.cambiarVistasQML(comboBoxViewpoint.currentText)}

   }
```

Archivo ...Form.qml

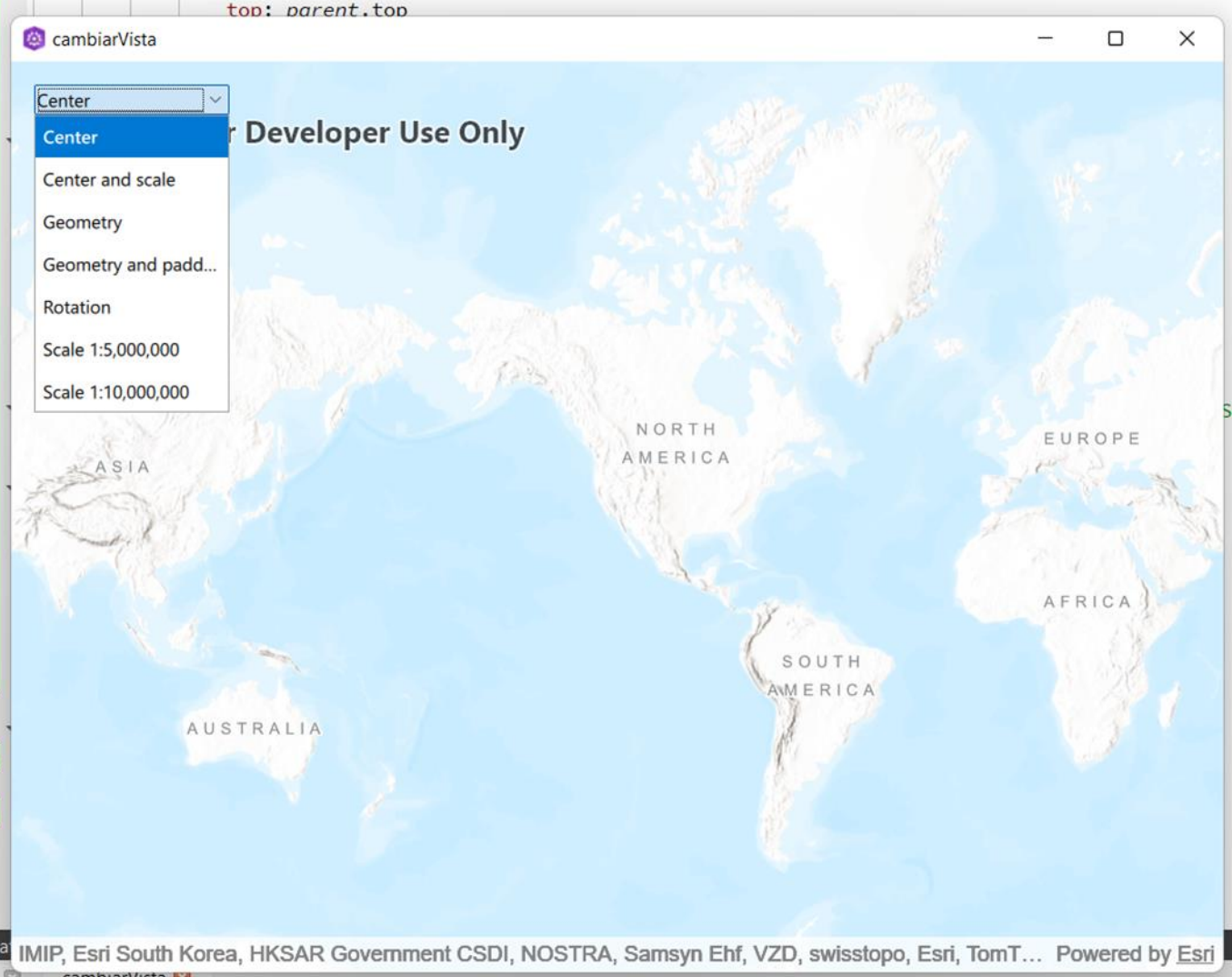
```
CambiarVista {
    id: model
    mapView: view

  }
```

Projects

- cambiarVista
  - cambiarVista.pro
  - Android
  - arcgisruntime
  - iOS
  - Mac
  - Win
  - Headers
    - CambiarVista.h
  - Sources
    - CambiarVista.cpp
    - main.cpp
  - Resources
    - qml\qml.qrc
      - /qml
        - CambiarVistaForm.qml
        - main.qml
    - Resources\Resources.qrc
- QML

CambiarVistaForm.qml | CambiarVista | CRLF | Line: 63, Col: 5 (Sel: 61)

```
36          top: parent.top
63                                                  Scale 1:5,000,000",
```



QML | cambiarVista

```
18:58:53: Starting D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Proyectos\build-cambiarVista-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\cambiarVista.exe...
QML debugging is enabled. Only use this in a safe environment.
```