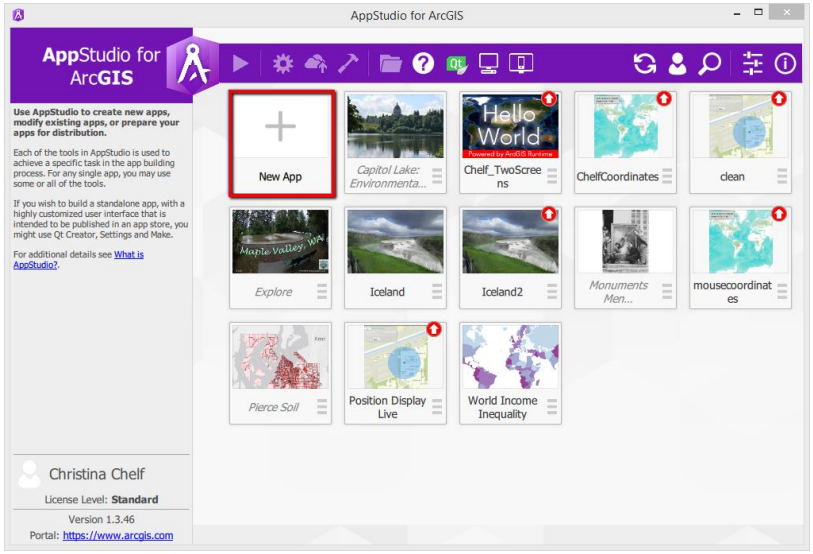
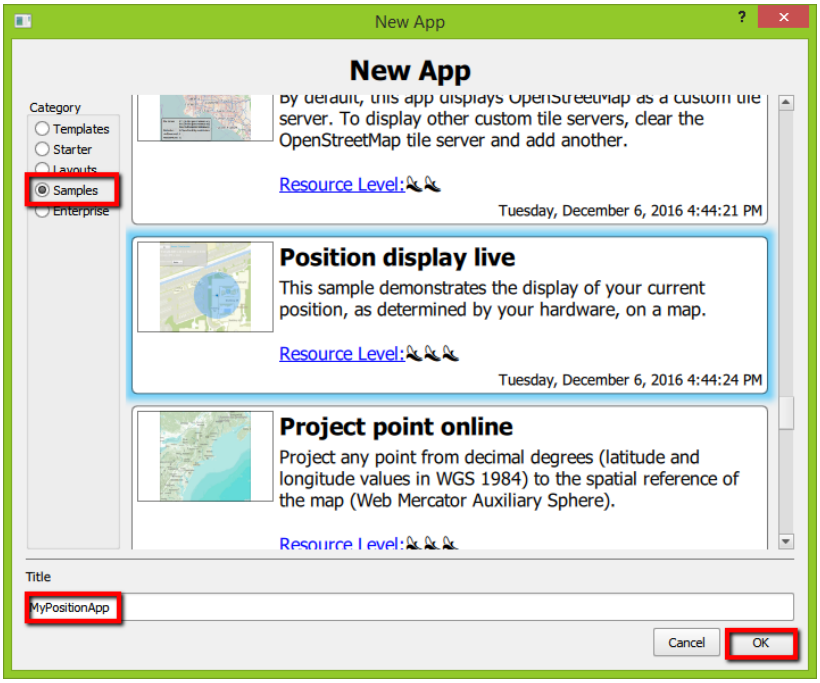
This tutorial will walk you through how to create a Live Position Display app with mouse hover coordinates added in. This app could be useful to people wanting to know their location while on a walk or run. For adding other functionality to your app I recommend checking out this site for reference https://doc.arcgis.com/en/appstudio/api-guide/apiguide.htm

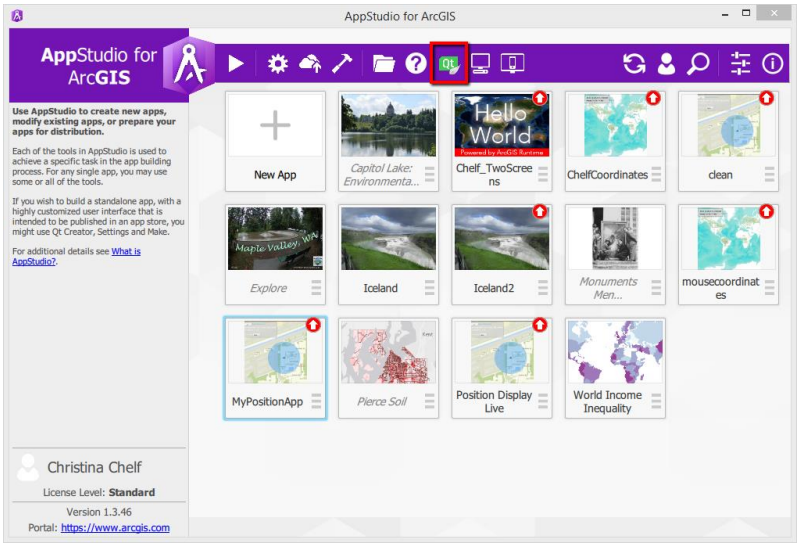
First open AppStudio for ArcGIS on your desktop. If you need to download it go to this site: http://doc.arcgis.com/en/appstudio/download/ and this site for instructions https://doc.arcgis.com/en/appstudio/create-apps/installappstudio.htm. Once you are in the AppStudio you will see an option to add a new app. Select this location.



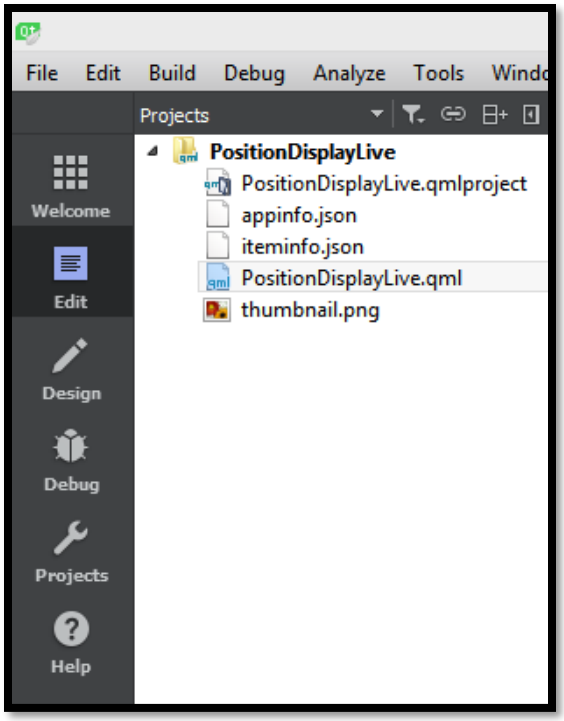
On the right hand side there are options to select a “Template”, a “starter”, “layouts”, “samples” or “Enterprise”. Select the Sample circle. Scroll down to the “Position display live” example. Type a name for your app and press “OK”.



To view the app, double click on the app or select the play button next to the AppStudio for ArcGIS logo. When you are ready to move to editing the app code yourself click on the green QT button on the top tool bar.



Once you are in the QT Creator click on the “PositionDisplayLive.qml” as seen here. This will show the code that controls the app.



Now that you are in the code for the app scroll down to the section within the “Map” tag. This is where we will add the mouse coordinates code and where you can change the basemap. Which base map you choose is up to you, I prefer an aerial view so in my example I called the World imagery service basemap and changed the opacity to 0.8. This just help the background not be overwhelming. For other basemap options see this site. https://services.arcgisonline.com/ArcGIS/rest/services/

```
        ArcGISTiledMapServiceLayer {
            url:
"http://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServe
r"
            opacity:0.8
    }
```

Next scroll down below the entire Position Display function. Here is where we are going to copy our first chunk of code. This is the code that allows the map to see and display the mouse coordinates. This should be pasted just below the “onModeChanged” function and entirely outside the “positionDisplay” container code.

```
        // call mouse coordinates by hovering
        MouseArea {
            id: mousecoordinates
            anchors.fill: parent
            hoverEnabled: true

            onPositionChanged: {
                var mapPoint = mainMap.toMapGeometry(mapToItem(mainMap,
mouseX, mouseY));
                coordsText2.text = mapPoint.toDecimalDegrees(2);
            }
    }
```

Scrolling down there will be some “Rectangle” and “Button” functions. These control the “Show Position buttons and the corresponding coordinate box that appears when the button is pushed. Move down to the “Column” container. Here there are a few improvements to the sample app. Within the first “Row” container there is a button function. I recommend changing the text that appears on the button from a single “X” to “Close” or “Exit Position” but this is personal preference.

```
        Row {
            spacing: 50

            Button {
                id: closeButton
                text: "Close"

                onClicked: {
                    positionSource.active = false;
                    compass.active = false;
                }
            }
```

The first text function after the button is for the source of the GPS coordinates. This information could be useful for scientists and people documenting their exact location, but general users will probably not care the source of their GPS coordinates. I deleted this text

```
            Text {
                text: "Source: " + positionSource.name
                color: "#00b2ff"
                font {
                    pointSize: 12
                    italic: true
                }
    }
```

Below the close button you get to the text that appears in the location information box. I recommend changing the “locationpoint” type from degree minutes seconds to decimal degrees using the code “*locationPoint*.toDecimalDegrees(2)”.

```
        Text {
             text: locationPoint.isEmpty || !locationPoint.valid
                   ? "Invalid Coordinates"
                   : locationPoint.toDecimalDegrees(2) +
                     (positionSource.position.horizontalAccuracyValid
                      ? " ± " +
Math.round(positionSource.position.horizontalAccuracy.toString()) + "m"
                      : "")
             color: locationPoint.valid ? "white": "red"
             font {
                 bold: true
                 pointSize: 16
             }
    }
```

After you replace the degrees text there are three more text containers controlling the vertical value, the speed and a position time stamp. I delete the date/time stamp because it is not necessary.

```
        Text {
             text: "Date: " + positionSource.position.timestamp.toString()
             color: "white"
             font {
                 pointSize: 12
             }
    }
```

The next function is in a “row” container with some text and a combo box. This is the code that displays the Display Mode drop down with the options: “off”, “Autopan”, “Navigation”, and “Compass”. I copied this section and put it into its own Column so to bring it to the lower left to declutter the upper left. Again, up to the developer, but this is what I thought looked best. If you do decide to move this make sure you include the final closing “}” to close the column container from above.

```
    }//make sure to close the column with a }
//Display Mode dropdown in lower left
    Column {
        id: columnControls2
        anchors {
            left: parent.left
            bottom: parent.bottom
            margins: 20 * scaleFactor
        }

        spacing: 5
        visible: positionSource.active
        Row {
            spacing: 15

            Text {
                text: "Display mode"
                color: "white"
                font {
                    pointSize: 12
                }
            }

            ComboBox {
                id: modesCombo
                model: modesModel
                onActivated: mainMap.positionDisplay.mode = index
                width: 200
            }
        }
  }
```

The final step is to make a column so in a box in the upper right the mouse hover coordinates appear. We do this below the column you just created. Notice we are first addint a rectangle that will be the box our text appears in. Here the fill should be set to “fill: columnControls 3” (or whatever you call the column you are about to create). Also, the id for this new column must be different than the column above. The column attributes can be very similar to the attributes of the first column except the anchor should be “left: parent.left”.

```
    //////////////////////
    // MOUSE COORDINATES//
    //////////////////////

    // Gray box in upper right
        Rectangle {
            visible: MouseArea.active
            color: "lightgrey"
            radius: 5
            border.color: "black"
            opacity: 0.77
            anchors {
                fill: columnControls3
                margins: -10
            }
        }
    // Text in upper right
        Column {
            id: columnControls3
            visible: MouseArea.active
            spacing: 5
            anchors {
                right: parent.right
                top: parent.top
                margins: 20 * scaleFactor
            }

            Text {
                id: coordsText2
                color: "white"
                font {
                    pointSize: 12
                }
            }
    }
```

Leave the bottom rectangle alone since this control the map view. The final code is here:

```
/* Copyright 2015 Esri
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *    http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
software
 * distributed under the License is distributed on an "AS IS"
BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions
and
 * limitations under the License.
 *
 */

import QtGraphicalEffects 1.0
import QtPositioning 5.2
import QtQuick 2.3
import QtQuick.Controls 1.2
import QtSensors 5.0

import ArcGIS.AppFramework 1.0
import ArcGIS.AppFramework.Controls 1.0
import ArcGIS.AppFramework.Runtime 1.0

App {
    id: app
    width: 800
    height: 532

    property double scaleFactor: AppFramework.displayScaleFactor
    property Point locationPoint : Point {
        property bool valid : false
        spatialReference: SpatialReference {
            wkid: 4326
        }
    }

    ListModel {
        id: modesModel

        ListElement { text: "Off" }
        ListElement { text: "Autopan" }
        ListElement { text: "Navigation" }
        ListElement { text: "Compass" }
    }
//////////////////////
//       Map        //
//////////////////////
    Map {
        id: mainMap
        anchors.fill: parent
        wrapAroundEnabled: true
        focus: true
        rotationByPinchingEnabled: true
        magnifierOnPressAndHoldEnabled: true
        mapPanningByMagnifierEnabled: true
        zoomByPinchingEnabled: true
```