Augmented Reality: Using GART

This article explains how to run the sample code provided by GART (Geo Augmented RealityToolkit) in Windows Phone. It will also let you know how to customize their sample code.

Introduction

Augmented Reality means placing some virtual object(s) on to the real view of the world. We can place any image, attach video or audio on a real world view to modify/duplicate the actual view of the world. To know more about Augmented Reality, you may go through the Wikipedia.

To implement Augmented Reality, couple of APIs are available for Windows Phone. Some of them are GARTand SLARToolkit. In this article, we will make use of GART APIs.

Note: It is not possible to test the sample code provided by GART in the emulator, you need a WP device for testing.

Downloading & Running the sample code provided by GART

- Go to this page and download the Samples.
- Unzip the *Samples file* into your workspace and import the project named "**SimpleAR**"
- Build the project and try to run it on device to check what it does.

Summarizing the functionalities provided by *SimpleAR project*

The sample code *SimpleAR project* illutrates the following features,

- Capturing the camera view
- Showing the current location using Bing Maps
- Showing a virtual circle embedding a cone which reflects the direction in which the device is moving
- Adding/Deleting labels (3 labels maximum in the default sample code, but you may add more) from the view
- Rotating the view

The virtual circle embedding a cone can be shown directly on the view provided by the camera or on maps. This circle can be made visible/invisible as well depending on the requirements.

Note: To understand more about this project, please go through the documentation here.

Understanding the Design Page

The resources part of the design page of the SimpleAR project would be like:

```
<phone:PhoneApplicationPage.Resources>
       <maps:ApplicationIdCredentialsProvider x:Key="BingCredentials" ApplicationId="AsoYwOquow1D9inkw-YSN1iYJ7xPqmsKHvKPUZP_7B_sww
   </phone:PhoneApplicationPage.Resources>
```

This sample app makes heavy use of the Bing Maps to show the locations on the map. For this, a key and an AppId has been allocated above.

```
    <phone:PhoneApplicationPage.ApplicationBar>
  <shell:ApplicationBar>
   <shell:ApplicationBarIconButton x:Name="RotateButton" IconUri="/icons/appbar.rotate.png" IsEnabled="True" Text="rotate" Click="Ro
   <shell:ApplicationBarIconButton x:Name="HeadingButton" IconUri="/icons/appbar.heading.png" IsEnabled="True" Text="Heading" Click=
   <shell:ApplicationBarIconButton x:Name="MapButton" IconUri="/icons/appbar.map.png" IsEnabled="True" Text="map" Click="MapButton_C
   <shell:ApplicationBarIconButton x:Name="ThreeDButton" IconUri="/icons/appbar.3D.png" IsEnabled="True" Text="3D" Click="ThreeDButt
            <shell:ApplicationBar.MenuItems>
                <shell:ApplicationBarMenuItem x:Name="AddLocationsMenu" Text="add locations" Click="AddLocationsMenu_Click" />
                <shell:ApplicationBarMenuItem x:Name="ClearLocationsMenu" Text="clear locations" Click="ClearLocationsMenu_Click" />
            </shell:ApplicationBar.MenuItems>
        </shell:ApplicationBar>
</phone:PhoneApplicationPage.ApplicationBar>
```

Using the above code, four ApplicationBarIconButton and two ApplicationBarMenuItems are placed. Each of the icon button or menu item is associated with its event handlers which will carry out their corresponding tasks (as mentioned above).

```
    <Grid x:Name="LayoutRoot">
  <ARControls:ARDisplay x:Name="ARDisplay" d:LayoutOverrides="Width">
   <ARControls:VideoPreview x:Name="VideoPreview" />
   <ARControls:OverheadMap x:Name="OverheadMap" CredentialsProvider="{StaticResource BingCredentials}" />
                        <ARControls:WorldView x:Name="WorldView" />
   <ARControls:HeadingIndicator x:Name="HeadingIndicator" HorizontalAlignment="Center" VerticalAlignment="Center" />
  </ARControls:ARDisplay>
 </Grid>
```


(C) Copyright Nokia 2013. All rights reserved.

In the above snippet, ARControls:ARDisplay is the main control in which four views have been placed. These four views will act as the children of the parent control - ARDisplay. The views used are:

- VideoPreview - to show the camera view
- OverheadMap - to show the Bing Maps
- WorldView - to place the virtual things or we can say to actually implemented the Augmented Reality in our app
- HeadingIndicator - to show you the headings (if any)

The Bing credentials are provided with the OverheadMap view to use the Bing Maps.

Code Behind

We will not go in detail to understand the code behind file here not only because it's already mentioned in the documentation section on the GART site but the main focus of this article is to understand "How to customize the sample code provided by GART".

Still if we try to summarize, the main points would be:

- Started the AR service in OnNavigatedTo()
- Will stop the service in OnNavigatedFrom()
- The four ApplicationBarIconButtons will do their corresponding tasks.
- On pressing "Add Location" menuItem, we are generating three instances of *ARItem* type providing them the *Content* value and *GeoLocation* value. These three instances are added one by one into the ARDisplay control via

```
ARDisplay.ARItems.Add(item);
```

The ARItem class originally is defined as:

```
    public class ARItem : ObservableObject
     {
         public ARItem();

         public object Content { get; set; }
         public GeoCoordinate GeoLocation { get; set; }
         public Vector3 RelativeLocation { get; set; }
         public Action<ItemCalculationSettings, ARItem> WorldCalculation { get; set; }
         public Vector3 WorldLocation { get; set; }
     }
```

- On pressing "Clear Location" menuItem, the already added ARItem instances are cleared using

```
ARDisplay.ARItems.Clear();
```

It is suggested to understand the code till this point so that you can try running the app on device as we are going to customize this example in the next section.

Customizing the Sample Code

Now we will customize the sample code by showing some texts and an image at some location. The image will be a static image embedded in the project. The text strings would be: "Place Name Here", "Description Here" and "Rating Here". To show them on some location we have used the current location of the device itself.

Changes in Design Page

We will add two more menuItems named ShowCustomizedMenu and ClearCustoizedMenu via: Inside Application Bar:

```
<shell:ApplicationBarMenuItem x:Name="ShowCustomizedMenu" Text="show customization effect" Click="ShowCustomizedMenu_Click" />
<shell:ApplicationBarMenuItem x:Name="ClearCustoizedMenu" Text="clear customization effect" Click="ClearCustoizedMenu_Click" />
```

Inside ARDisplay control:

```
<ARControls:WorldView x:Name="WorldView"  ItemTemplate="{StaticResource CityItem}"
```

Above, we are attaching a style template like:

```
      <phone:PhoneApplicationPage.Resources>
        <maps:ApplicationIdCredentialsProvider x:Key="BingCredentials" ApplicationId="AsoYwOquow1D9inkw-YSN1iYJ7xPqmsKHvKPUZP_7B_sww

        <DataTemplate x:Key="CityItem">
            <Border BorderBrush="Black" BorderThickness="4" CornerRadius="8" Background="#FF003847" Width="320">
                <StackPanel Margin="4">
                    <TextBlock x:Name="NameBlock" TextWrapping="NoWrap" Text="{Binding Content}" FontSize="38" VerticalAlignment="Ce
                    <TextBlock x:Name="DescriptionBlock" TextWrapping="Wrap" Text="{Binding Description}" FontSize="24" VerticalAlig
                    <TextBlock x:Name="RatingBlock" TextWrapping="Wrap" Text="{Binding Rating}" FontSize="22" VerticalAlignment="Cen
                   <Image x:Name="Img" Source="{Binding Image}" Width="150" Height="150"/>
                </StackPanel>
            </Border>
        </DataTemplate>

    </phone:PhoneApplicationPage.Resources>
```

Changes in Code Behind File

(C) Copyright Nokia 2013. All rights reserved.

- Declare an ObservableCollection<ARItem> type object like:

```
private ObservableCollection<ARItem> locationsCollection;
```

- Initializing it in the constructor like:

```
locationsCollection = new ObservableCollection<ARItem>();
```

- We will declare new event handlers like:

```
        private void ShowCustomizedMenu_Click(object sender, EventArgs e)
        {
            GetData();
        }
        private void ClearCustoizedMenu_Click(object sender, EventArgs e)
        {
            ARDisplay.ARItems.Clear();
        }
        private void GetData()
        {
         Uri uri = new Uri("/SamplePic.png", UriKind.Relative);
            BitmapImage imgSource = new BitmapImage(uri);

            // Start with the current location
            GeoCoordinate current = ARDisplay.Location;

            locationsCollection.Add(new Places()
              {
                  GeoLocation = new GeoCoordinate(current.Latitude,current.Longitude),
                  Content = "Place Name Here",
                  Description = "Description Here",
                  Rating = "Rating Here",
                  Image =  imgSource
              });
            ARDisplay.ARItems = locationsCollection;
        }
```

Above, we have used a new class named *Places* which we will derive from ARItem like shown below:

```
namespace SimpleAR
{
    public class Places: ARItem
    {
    private string description;
    private string rating;
    private BitmapImage image;

    public string Description
    {
        get
        {
            return description;
        }

        set
        {
           if (description != value)
            {
               description = value;
               NotifyPropertyChanged(() => Description);
            }
        }
    }

    public string Rating
    {
        get
        {
            return rating;
        }

        set
        {
            if (rating != value)
            {
                rating = value;
                NotifyPropertyChanged(() => Rating);
            }
        }
    }

 public BitmapImage Image
    {
        get
        {
            return image;
        }

        set
        {
            if (image != value)
            {
                image = value;
                NotifyPropertyChanged(() => Image);
            }
        }
    }
   }

}
```

The above declared class *Places* is derived from ARItem, also we have added two new string items into the class named *Description* and *Rating*. While constructing the instance of Places class, if either of the two new member variables values will be changed then only they will be reflected.

Running and Testing the customized sample code

(C) Copyright Nokia 2013. All rights reserved.

Now build the project and run it. Try to press the "show customization effect" menu item and you should be able to see the effects.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.