How to show GPS Device Activity Status

Introduction

Displaying a GPS activity status to the user while creating a Symbian C++ application that makes use of Location Based Services is often ignored by many of us. Showing GPS activity status on the screen minimizes the confusion that generally comes with the GPS technology itself such as hot/cold boot up time for the GPS device, unclear line of sight and notifying user if the GPS device is out mobile phone's bluetooth reach when an external GPS module is used. Last indicating if the GPS device has been shutdown. This article will demonstrate how to show GPS device activity status on user's mobile screen.

Process

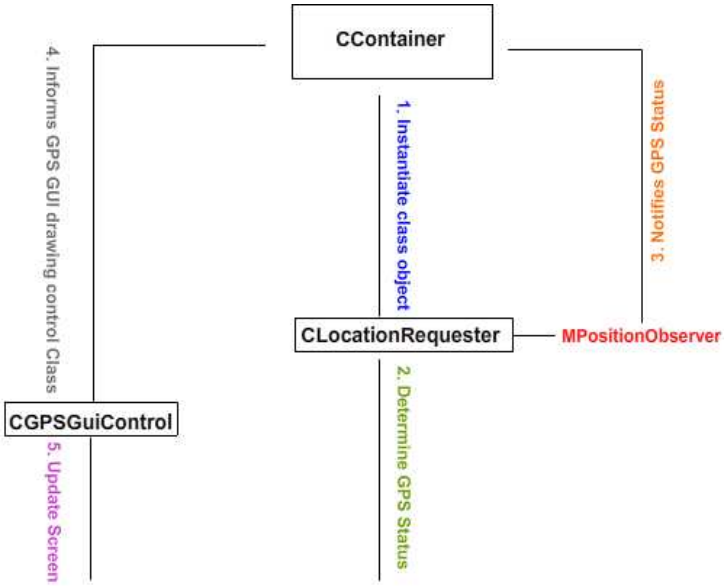
Following are the GPS activity status we need to indicate on the screen.

- GPS device is out of reach or it is shutdown
- Connection with the GPS device has been established however it is in a look up state hence searching for satellites.
- GPS is successfully fetching coordinates from the satellites.

As it is seen from the points that this requires use of active objects so that status change is immediately reflected on the stage.

Top Level Class Diagram

Following is the Class Diagram used in achieving the results



GPS status indicators

--::-- GPS device is out of reach or it is shutdown

--::-- GPS Lookup state

--::-- GPS is active and obtaining position data

Code

- CGPSMonitorPositionRequestor Class -

```
// ---------------------------------------------------------------------------
// RunL() implements CActive RunL()
// Handles an active object’s request completion event.
// ---------------------------------------------------------------------------
//
```


(C) Copyright Nokia 2013. All rights reserved.

```
void CGPSMonitorPositionRequestor::RunL()
    {
    TBuf<KPositionMaxModuleName> buffer;

    switch ( iStatus.Int() )
        {
        // The fix is valid
         case KErrNone:
        // Pre process the position information
        // The fix has only partially valid information.
        // It is guaranteed to only have a valid timestamp
        case KPositionPartialUpdate:

        {
             PositionUpdatedL();
             break;
        }

        // The data class used was not supported
        case KErrArgument:
            {
            // Set info base to position info
            iPosInfoBase = &iPositionInfo;

             // Request next position
            iPositioner.NotifyPositionUpdate( *iPosInfoBase, iStatus );

            // Set this object active
            SetActive();

            //Update GPS State on Screen
     iPositionListener.UpdateGPSStatusOnScreenL(0); //EGPSLocationNotAvailable

            break;
            }
        // The position data could not be delivered
        case KPositionQualityLoss:
            {
            if(iMaxTries < 5) // Maximum Tries
        {
              if ( iGettingFirstValidPosition )
                {
                //Change the data class type
                iPosInfoBase = &iSatelliteInfo;
                }
               // Request next position
              iPositioner.NotifyPositionUpdate( *iPosInfoBase, iStatus );

       // Set this object active
           SetActive();

       SetEngineState(EConnectingToLbs);
              iMaxTries++; // Update Max Tries Counter
      }
      else //After 5 Retries
      {
  // Send event to position listener
             HBufC* textBuf = CEikonEnv::Static()->AllocReadResourceAsDes16LC(R_POSITION_QUALITY_ERROR_TEXT);
             iPositionListener.ShowGPSErrorL(*textBuf);
             CleanupStack::PopAndDestroy(textBuf);


  // Close the positioner
      iPositioner.Close();
      // Close the session to the position server
      iPosServer.Close();

  //Update GPS State on Screen
  iPositionListener.UpdateGPSStatusOnScreenL(0); //EGPSDisconnected
  SetEngineState(ELbsIdle);
  //Releasing connection
         iGpsOnStatus = EFalse;
  // Send event to position listener
         TBuf<KGpsErrBuffer> text;
  //If Retry Option is Enabled in the Settings Screen run the timer
  //to start searching for a GPS Device again.
  if(iGPSReconnectionStatus)
    {
       text.Format(KGpsErrDisconnect,(iRetryAfter/KMinute));                            //Sleep and Retry after RetryTime Set by the
     iPositionListener.ShowGPSErrorL(KGpsErrStopped);
     iPositionListener.ShowGPSErrorL(text);
    }
    else //Disconnect GPS Device
    {
            text.Copy(KGpsErrStopped);
              //Change GPS Engine Status
        //so that changes reflect in the menu
               iPositionListener.ShowGPSErrorL(text);
          iGPSEngine->DeactivateGpsL();
           }
      }
             break;
           }
        // Access is denied
        case KErrAccessDenied:
            {
            // Send error to position listener
            buffer.Format(KLbsErrAccess, iStatus.Int());
            iPositionListener.ShowGPSErrorL(buffer);

            //Update GPS State on Screen
        iPositionListener.UpdateGPSStatusOnScreenL(0); //EGPSDisconnected
            break;
            }
        // Request timed out
        case KErrTimedOut:
            {
         // Send error to position listener
         //HBufC* text = CEikonEnv::Static()->AllocReadResourceAsDes16LC(R_POSITION_TIMEOUT_TEXT);
         //iPositionListener.ShowGPSErrorL(*text);
         //CleanupStack::PopAndDestroy(text);

         if ( iGettingFirstValidPosition )
         {
             //Change the data class type
             iPosInfoBase = &iSatelliteInfo;
         }

         // Request position
         iPositioner.NotifyPositionUpdate( *iPosInfoBase, iStatus );
```


(C) Copyright Nokia 2013. All rights reserved.

```
        // Set this object active
        SetActive();

        //Update GPS State on Screen
iPositionListener.UpdateGPSStatusOnScreenL(0); //EGPSDisconnected
            break;
            }
        // The request was canceled
        case KErrCancel:
            {
                // Send error to position listener
                iPositionListener.ShowGPSErrorL( KLbsErrCanceled );

    //Update GPS State on Screen
iPositionListener.UpdateGPSStatusOnScreenL(0); //EGPSDisconnected
            break;
            }
        // There is no last known position
        case KErrUnknown:
            {
            // Send event to position listener
            HBufC* text = CEikonEnv::Static()->AllocReadResourceAsDes16LC(R_POSITION_UNKNOWN_TEXT);
            iPositionListener.MessageL(*text);
            CleanupStack::PopAndDestroy(text);

            if ( iGettingFirstValidPosition )
                {
                //Change the data class type
                iPosInfoBase = &iSatelliteInfo;

                //Mark that we are not requesting NotifyPositionUpdate
                iGettingFirstValidPosition = EFalse;
                }

            // Request next position
            iPositioner.NotifyPositionUpdate( *iPosInfoBase, iStatus );
            // Set this object active
            SetActive();
            //Update GPS State on Screen
    iPositionListener.UpdateGPSStatusOnScreenL(1); //EGPSLookupState
            break;
            }
        // Unrecoverable errors.
        default:
            {
            // Send error to position listener
            buffer.Format(KLbsErrLocRequest, iStatus.Int());
            iPositionListener.ShowGPSErrorL( buffer );

    //Update GPS State on Screen
iPositionListener.UpdateGPSStatusOnScreenL(0); //EGPSDisconnected
            break;
            }
        }

        //We are not going to query the last known position anymore.
        if ( iGettingFirstValidPosition )
            {
            //Mark that we are not requesting NotifyPositionUpdate
            iGettingFirstValidPosition = EFalse;
            }
    }
```

```
// ---------------------------------------------------------------------------
// CGPSMonitorPositionRequestor::PositionUpdatedL
// Pre process the position information
// ---------------------------------------------------------------------------
//
void CGPSMonitorPositionRequestor::PositionUpdatedL()
    {
    //If we are getting the first valid position
    if ( iGettingFirstValidPosition )
        {
        // Obtain GPS TimeToFix before
        // Sending position information to the registered listener

        //Change the data class type
        iPosInfoBase = &iSatelliteInfo;

        // Request next position
        iPositioner.NotifyPositionUpdate( *iPosInfoBase, iStatus );

        // Set this object active
        SetActive();

        }
    else
        {
        // Check if the id of the used PSY is 0
        if ( 0 == iUsedPsy.iUid)
            {
            // Set the id of the currently used PSY
            iUsedPsy = iPosInfoBase->ModuleId();
            }
        // Check if the position module has changed
        else if ( iPosInfoBase->ModuleId() != iUsedPsy )
            {
            // Set the id of the currently used PSY
            iUsedPsy = iPosInfoBase->ModuleId();

            //Position module info of new module
            TPositionModuleInfo moduleInfo;

            // Get module info
            iPosServer.GetModuleInfoById(iUsedPsy,moduleInfo);

            // Get classes supported
            TInt32 moduleInfoFamily = moduleInfo.ClassesSupported(EPositionInfoFamily);

            iPosInfoBase = &iSatelliteInfo;

            // Check if the new module supports
            // TPositionSatelliteInfo class
```


(C) Copyright Nokia 2013. All rights reserved.

```
            if ( EPositionSatelliteInfoClass & moduleInfoFamily )
                {
                // Set info base to satellite info
                iPosInfoBase = &iSatelliteInfo;
                }
            // The position module must support atleast
            // TPositionInfo class
            else
                {
                // Set info base to position info
                iPosInfoBase = &iPositionInfo;
                }
            }
            //A minimum of 3 satellites are needed to assure that acquired positioning data
            //is valid
            if(iSatelliteInfo.NumSatellitesUsed()<3)
         {
             //Update GPS State on Screen
        iPositionListener.UpdateGPSStatusOnScreenL(1); //EGPSLookupState                        iSatelliteDown=ETrue;
         }
         else
         {
           if(iGpsOnStatus == EFalse)
           {
            iGpsOnStatus = ETrue;
            iPositionListener.ShowGPSErrorL(KGpsErrStarted);
           }
           iSatelliteDown=EFalse;
           //Update GPS State on Screen
    iPositionListener.UpdateGPSStatusOnScreenL(2); //EGPSLocationAvailable
         }
      iSatelliteInfo.GetCourse(iCourse);
      iHorizontalAccuracy = iSatelliteInfo.HorizontalDoP();
      iVerticalAccuracy = iSatelliteInfo.VerticalDoP();
      iSpeedAccuracy      = iCourse.SpeedAccuracy();
      // Send position information to registered listener
        iPositionListener.PositionUpdatedL(*iPosInfoBase,iSatelliteDown);
            // Request next position
            iPositioner.NotifyPositionUpdate( *iPosInfoBase, iStatus );

                // Set this object active
            SetActive();
        }
    }
```

- CGPSGuiControl Class - used for drawing GPS status images on the screen

```
// ---------------------------------------------------------------------------
//  void CGPSGuiControl::SetGPSState(TInt aState)
//
//  Screen GPS Display changes based on the GPS State
// ---------------------------------------------------------------------------
//
void CGPSGuiControl::SetGPSState(TInt aState)
{
  iState = aState;
  if(iState == 0)
  {
     iGPSStatusBitmap  = iGPSInActiveBmp; //CFbsBitmap* = CFbsBitmap*
  }
  else if(iState ==1)
  {
   iGPSStatusBitmap  = iGPSLookupBmp;
  }
  else if(iState == 2)
  {
   iGPSStatusBitmap  = iGPSActiveBmp;
  }
     DrawDeferred();
}
```

```
// ---------------------------------------------------------------------------
//  void CGPSGuiControl::DrawL()
//
//  Draw container contents.
// ---------------------------------------------------------------------------
//
void CGPSGuiControl::Draw( const TRect& aRect ) const
 {
 CWindowGc& gc = SystemGc();
   gc.Clear( aRect );

   if(iGPSStatusBitmap!=NULL)
 {   gc.BitBltMasked(TPoint(iGPSStatusx,iGPSStatusy),iGPSStatusBitmap,iGPSStatusSize,iGPSStatusMaskBitmap,EFalse);
 }
```

Download

Following are few files attached that were used in the process. Since these modules are part of a bigger module. Few lines of code are not necessary for accomplishing this task and shall be truncated.

Download : media:GPSActivityStatus.zip

Author

--raheal_akh 13:20, 23 May 2008 (EEST) RAHEAL AKHTAR

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.