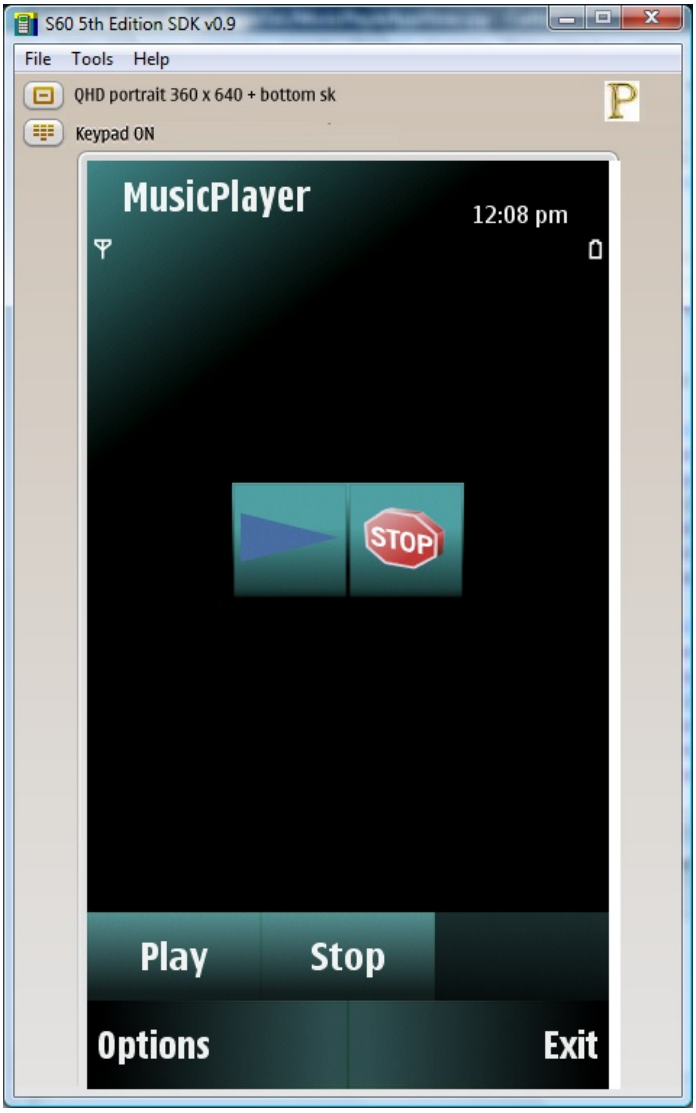
Creating custom Music Player with S60 Touch UI APIs

Overview

In this article, we will learn how to use different S60 Touch UI APIs to create custom Music Player. You can extend it to add more functionality. Here, Generic Button API, Toolbar API, Stylus Popup List API and LongTapDetector API have been used in this article to give an idea of how one can implement such rich set of APIs to create promising applications.

Which APIs are used and why

- Generic Button API: To create buttons for Playing and Stoping the music track.
- Toolbar API: To create toolbar and provides the same functionality.
- Stylus Popup List API: To give floating stylus Popup Menu which does the same functionality to play and stop the music track.
- LongTapDetector API: To handle long tap events on Buttons and to illustrate its use.
- MdaAudioPlayerUtility API: To play and stop the music track.



Create Music Player Utility core classes

AudioPlayer.h

```
#ifndef AUDIOPLAYER_H_
#define AUDIOPLAYER_H_
```


(C) Copyright Nokia 2013. All rights reserved.

```
#include <e32std.h>
#include <MdaAudioSamplePlayer.h>
class CAudioPlayerUtility : public CBase, public MMdaAudioPlayerCallback
{
 public:
  static CAudioPlayerUtility* NewL(const TDesC& aFileName);
  static CAudioPlayerUtility* NewLC(const TDesC& aFileName);
  ~CAudioPlayerUtility();
 private:
  CAudioPlayerUtility();
  void ConstructL(const TDesC& aFileName);
 public:
  void Play();
  void Stop();
 public: // from MMdaAudioToneObserver
  void MapcInitComplete(TInt aError, const TTimeIntervalMicroSeconds& aDuration);
  void MapcPlayComplete(TInt aError);
 private:
  CMdaAudioPlayerUtility* iPlayUtility;
};

#endif /* AUDIOPLAYER_H_ */
```

AudioPlayer.cpp

```
#include "AudioPlayer.h"
#include <MdaAudioTonePlayer.h>
#include <eikmenup.h>
CAudioPlayerUtility* CAudioPlayerUtility::NewL(const TDesC& aFileName)
{
 CAudioPlayerUtility* self = NewLC(aFileName);
 CleanupStack::Pop(self);
 return self;
}
CAudioPlayerUtility* CAudioPlayerUtility::NewLC(const TDesC& aFileName)
{
 CAudioPlayerUtility* self = new (ELeave) CAudioPlayerUtility();
 CleanupStack::PushL(self);
 self->ConstructL(aFileName);
 return self;
}
CAudioPlayerUtility::~CAudioPlayerUtility()
{
 if(iPlayUtility)
 {
  iPlayUtility->Stop();
  iPlayUtility->Close();
 }
 delete iPlayUtility;
}
CAudioPlayerUtility::CAudioPlayerUtility()
{
}
void CAudioPlayerUtility::ConstructL(const TDesC& aFileName)
{
 iPlayUtility = CMdaAudioPlayerUtility::NewFilePlayerL(aFileName, *this);
}
void CAudioPlayerUtility::Play()
{
 iPlayUtility->Play();
}
void CAudioPlayerUtility::Pause()
{
 iPlayUtility->Pause();
}
void CAudioPlayerUtility::Stop()
{
 iPlayUtility->Stop();
}
void CAudioPlayerUtility::MapcPlayComplete(TInt /*aError*/)
{
}
void CAudioPlayerUtility::MapcInitComplete(TInt aError, const TTimeIntervalMicroSeconds& /*aDuration*/)
{
 if(aError == KErrNone)
 {
  iPlayUtility->SetVolume(iPlayUtility->MaxVolume());
 }
}
```

Resource for Buttons

```
...
#include <avkon.mbg>
//Button resource
RESOURCE AVKON_BUTTON r_first_button
    {
    flags  = KAknButtonTextInsideFrame;
    states =
        {
        AVKON_BUTTON_STATE
            {
            helptxt = "Play";
            bmpfile = AVKON_BITMAP_FILE;
            bmpid =  EMbmAvkonQgn_indi_navi_arrow_right;
            bmpmask =  EMbmAvkonQgn_indi_navi_arrow_right_mask;
            },
        AVKON_BUTTON_STATE
            {
            flags = KAknButtonStateHasLatchedFrame;
            bmpfile = AVKON_BITMAP_FILE;
            bmpid =  EMbmAvkonQgn_indi_navi_arrow_right;
            bmpmask =  EMbmAvkonQgn_indi_navi_arrow_right_mask;
            }
        };
    }
RESOURCE AVKON_BUTTON r_second_button
```


(C) Copyright Nokia 2013. All rights reserved.

```
    {
    flags  = KAknButtonTextInsideFrame;
    states =
        {
        AVKON_BUTTON_STATE
            {
            bmpfile = AVKON_BITMAP_FILE;
            bmpid = EMbmAvkonQgn_note_error;
            bmpmask = EMbmAvkonQgn_note_error_mask;
            helptxt = "Stop";
            }
        };
    }
```

Resource for Toolbar

```
// Add following resource definition for toolbar declared in EIK_APP_INFO
RESOURCE EIK_APP_INFO_EXT r_toolbar_ext
 {
  popup_toolbar = r_test_toolbar;
 }

RESOURCE AVKON_TOOLBAR r_test_toolbar
{
 flags = KAknToolbarFixed;
 items =
 {
  TBAR_CTRL
  {
  type = EAknCtButton;
  id = ECommand1; // In this example, we are using same set of commands as of used in AppUi:HandleCommandL
  control = AVKON_BUTTON
   {
   flags = KAknButtonTextInsideFrame;
   states =
    {
    AVKON_BUTTON_STATE
     {
     txt = "Play";
     helptxt = "Play me";
     }
    };
   };
  },
  TBAR_CTRL
  {
  type = EAknCtButton;
  id = ECommand2; // In this example, we are using same set of commands as of used in AppUi:HandleCommandL
  control = AVKON_BUTTON
   {
   flags = KAknButtonTextInsideFrame;
   states =
    {
    AVKON_BUTTON_STATE
     {
     txt = "Stop";
     helptxt = "Stop me";
     }
    };
   };
  }
 };
}
```

Resource for Stylus Popup Menu

```
//For stylus
RESOURCE STYLUS_POPUP_MENU r_stylus_popup_menu
{
 items =
 {
  STYLUS_POPUP_MENU_ITEM
  {
   txt = "Play";
   command = ECommand1;
  },
  STYLUS_POPUP_MENU_ITEM
  {
   txt = "Stop";
   command = ECommand2;
  }
 };
}
```

Implementing Music Player with S60 Touch UI APIs in the code

- We will inherit our class from `MCoeControlObserver` interface and implement its pure virtual method `HandleControlEventL()` to receive and handle events for Buttons.
- We will inherit our class from `MAknToolbarObserver` interface and implement its pure virtual method `OfferToolbarEventL()` to receive and handle events for Toolbar.
- We will inherit our class from `MAknLongTapDetectorCallBack` interface and implement its pure virtual method `HandleLongTapEventL()` to receive and handle events for LongTap Detection.
- We will inherit our class from `MEikMenuObserver` interface and implement its pure virtual method `ProcessCommandL()` to receive and handle events for Stylus Popup Menu commands.

(C) Copyright Nokia 2013. All rights reserved.

- Declare objects of `CAknButton` and use it to set observer for this class and handle events on Play and Stop buttons.
- Declare objects of `CAknLongTapDetector` and use it to set observer for this class and handle events on Long tap detection.
- Declare objects of `CAknStylusPopUpMenu` for Stylus Popup Menu.
- Declare objects of `CAudioPlayerUtility` for Audio Player Utility.

MusicPlayerAppView.h

```
...
//For Toolbar API
#include <AknToolbar.h>
#include <akntoolbarobserver.h>
//For Longtap detection
#include <aknlongtapdetector.h>
//For Stylus
#include <aknstyluspopupmenu.h>
#include <EIKMOBS.H>
//For Generic Button
#include <aknbutton.h>
//For Audio Player Utility
#include "AudioPlayer.h"
....
// CLASS DECLARATION
class CMusicPlayerAppView : public CCoeControl, public MCoeControlObserver, public MAknToolbarObserver,
                            public MAknLongTapDetectorCallBack, public MEikMenuObserver
 {
 ....
  virtual void HandleControlEventL(CCoeControl* aControl,TCoeEvent aEventType);
  TInt CountComponentControls() const;
  CCoeControl* ComponentControl(TInt aIndex) const;
  void CreateButtonUsingResourceL();

  //From MAknToolbarObserver
  void OfferToolbarEventL(TInt aCommand);

  virtual void HandleLongTapEventL(const TPoint& aPenEventLocation, const TPoint& aPenEventScreenLocation);

  //For stylus
  void ProcessCommandL(TInt aCommandId);
  void SetEmphasis(CCoeControl* /*aMenuControl*/,TBool /*aEmphasis*/)
  {
  }
 private:
  //For Buttons
  CAknButton* iPlayButton;
  CAknButton* iStopButton;

  //For Audio Player
  CAudioPlayerUtility* iMyAudioPlayer;

  //For Long Tap Detection
  CAknLongTapDetector* iLongTapDetector;

  //For Stylus Popup Menu
  CAknStylusPopUpMenu* iStylusPopupMenu;
 };
```

MusicPlayerAppView.cpp

- Call to create Buttons from `ConstructL()` as shown below.
- Defining LongTapDetection object
- Defining Audio Player Utility object
- Setting observer for Toolbar observer

```
// ----------------------------------------------------------------------------
// CMusicPlayerAppView::ConstructL()
// Symbian 2nd phase constructor can leave.
// ----------------------------------------------------------------------------
//
void CMusicPlayerAppView::ConstructL(const TRect& aRect)
 {
 // Create a window for this application view
 CreateWindowL();

 iLongTapDetector = CAknLongTapDetector::NewL(this);

 // Set the windows size
 SetRect(aRect);

 CreateButtonUsingResourceL();

 iMyAudioPlayer = CAudioPlayerUtility::NewL(KMusicFile);

 //Set this to toolbar observer - We will receive events in OfferToolbarEventL
 if(STATIC_CAST(CMusicPlayerAppUi*, CEikonEnv::Static()->EikAppUi())->CurrentFixedToolbar())
 {
  STATIC_CAST(CMusicPlayerAppUi*, CEikonEnv::Static()->EikAppUi())->CurrentFixedToolbar()->SetToolbarObserver(this);
 }

 // Activate the window, which makes it ready to be drawn
 ActivateL();
 }
```

- Create generic buttons Play and Stop and set the rectangle for them.
- Set "this" as an observer for buttons and eventually we will receive callbacks in `HandleControlEventL()`

```
void CMusicPlayerAppView::CreateButtonUsingResourceL()
    {
    iPlayButton = CAknButton::NewL();
    iPlayButton->ConstructFromResourceL(R_FIRST_BUTTON);
    iPlayButton->SetContainerWindowL(*this);
    iPlayButton->SetRect(TRect ( TPoint(100,130), TSize(80,80) ) );
```


(C) Copyright Nokia 2013. All rights reserved.

```
    iPlayButton->SetObserver(this);
    iPlayButton->MakeVisible(ETrue);
    iPlayButton->ActivateL();

    iStopButton = CAknButton::NewL();
    iStopButton->ConstructFromResourceL(R_SECOND_BUTTON);
    iStopButton->SetContainerWindowL(*this);
    iStopButton->SetRect(TRect ( TPoint(180,130), TSize(80,80) ) );
    iStopButton->SetObserver(this);
    iStopButton->MakeVisible(ETrue);
    iStopButton->ActivateL();
    }
```

- We will handle callback events on buttons as shown below in the `HandleControlEventL`

```
void CMusicPlayerAppView::HandleControlEventL( CCoeControl* aControl, TCoeEvent aEventType )
{
    switch ( aEventType )
    {
        case EEventStateChanged:
        {
            if(aControl == iPlayButton)
     {
                if(iPlayButton->State()->Flags()!= KAknButtonStateHasLatchedFrame)
                {
                    if(iMyAudioPlayer)
                        iMyAudioPlayer->Pause();
                }
                else
                {
                    if(iMyAudioPlayer)
                        iMyAudioPlayer->Play();
                }

                if(iStopButton->State()->Flags()==KAknButtonStateHasLatchedFrame)
                    iStopButton->SetCurrentState(KAknButtonTextInsideFrame,ETrue);
            }
            else if(aControl ==iStopButton)
            {
                if(iMyAudioPlayer)
                    iMyAudioPlayer->Stop();
                if(iPlayButton->State()->Flags()==KAknButtonStateHasLatchedFrame)
                    iPlayButton->SetCurrentState(KAknButtonTextInsideFrame,ETrue);
            }
        }
            break;

        default:
            break;
    }
}
```

- Definition for the following functions to return button controls.

```
TInt CMusicPlayerAppView::CountComponentControls() const
{
 return 2; // return number of controls inside this container
}

CCoeControl* CMusicPlayerAppView::ComponentControl(TInt aIndex) const
{
    switch ( aIndex )
    {
    case 0:
        return iPlayButton;
    case 1:
        return iStopButton;
    default:
        return NULL;
    }
}
```

Handling Long Tap Detection

Passing events to LongTap Detection from HandlePointerEventL

```
// ---------------------------------------------------------------------------
// CMusicPlayerAppView::HandlePointerEventL()
// Called by framework to handle pointer touch events.
// Note: although this method is compatible with earlier SDKs,
// it will not be called in SDKs without Touch support.
// ---------------------------------------------------------------------------
//
void CMusicPlayerAppView::HandlePointerEventL(const TPointerEvent& aPointerEvent)
{
    iLongTapDetector->PointerEventL(aPointerEvent);
    // Call base class HandlePointerEventL()
    CCoeControl::HandlePointerEventL(aPointerEvent);
}
```

Handling Long Tap Detection events

- Here we will show Stylus Popup menu on Long Tap Detection as shown below

(C) Copyright Nokia 2013. All rights reserved.

```
void CMusicPlayerAppView::HandleLongTapEventL( const TPoint& aPenEventLocation, const TPoint& aPenEventScreenLocation )
{
    if(!iStylusPopupMenu)
    {
        iStylusPopupMenu = CAknStylusPopUpMenu::NewL( this , aPenEventLocation);
        TResourceReader reader;
        iCoeEnv->CreateResourceReaderLC(reader,R_STYLUS_POPUP_MENU);
        iStylusPopupMenu->ConstructFromResourceL(reader);
        CleanupStack::PopAndDestroy();
    }
    iStylusPopupMenu->ShowMenu();
    iStylusPopupMenu->SetPosition(aPenEventLocation);
}
```

Handling commands on Stylus Popup Menu

```
void CMusicPlayerAppView::ProcessCommandL(TInt aCommand)
{
    if(aCommand == ECommand1)
    {
        if(iMyAudioPlayer)
            iMyAudioPlayer->Play();
        if(iPlayButton->State()->Flags()!=KAknButtonStateHasLatchedFrame)
            iPlayButton->SetCurrentState(KAknButtonStateHasLatchedFrame,ETrue);
        if(iStopButton->State()->Flags()==KAknButtonStateHasLatchedFrame)
            iStopButton->SetCurrentState(KAknButtonTextInsideFrame,ETrue);
    }
    else if (aCommand == ECommand2)
    {
        if(iMyAudioPlayer)
            iMyAudioPlayer->Stop();
        if(iPlayButton->State()->Flags()==KAknButtonStateHasLatchedFrame)
            iPlayButton->SetCurrentState(KAknButtonTextInsideFrame,ETrue);
    }
    else
    {
        CAknInformationNote* info = new (ELeave) CAknInformationNote;
        info->ExecuteLD(_L("Outside Stylus Popup!!!"));
    }
}
```

Destruction

- Make sure to delete objects on buttons, Long tap detection, Audio player and Stylus Popup Menu in the destructor of the class.

```
// ----------------------------------------------------------------------------
// CMusicPlayerAppView::~CMusicPlayerAppView()
// Destructor.
// ----------------------------------------------------------------------------
//
CMusicPlayerAppView::~CMusicPlayerAppView()
{
    delete iPlayButton;
    delete iStopButton;

    if(iMyAudioPlayer)
        delete iMyAudioPlayer, iMyAudioPlayer = NULL;

    if(iLongTapDetector)
        delete iLongTapDetector, iLongTapDetector = NULL;

    if(iStylusPopupMenu)
        delete iStylusPopupMenu, iStylusPopupMenu = NULL;
}
```

Useful functions

Generic Button API

- ConstructFromResourceL()
- SetCurrentState()
- SetFlags()
- Flags()

LongTapDetector API

- PointerEventL()

Stylus Popup List API

- ShowMenu()
- SetPosition()

MAknToolbarObserver

(C) Copyright Nokia 2013. All rights reserved.

- OfferToolbarEventL()

MAknLongTapDetectorCallBack

- HandleLongTapEventL()

MCoeControlObserver

- HandleControlEventL()

Keywords

Headers

- #include <AknButton.h>
- #include <akntoolbarobserver.h>
- #include <AknToolbar.h>
- #include <akntoolbarextension.h>
- #include <aknlongtapdetector.h>
- #include <aknstyluspopupmenu.h>
- #include <EIKMOBS.H>
- #include "AudioPlayer.h"

Classes

- CAknButton
- CAknButtonState
- CAknLongTapDetector
- CAknStylusPopUpMenu
- CAudioPlayerUtility
- MCoeControlObserver
- MAknToolbarObserver
- MAknLongTapDetectorCallBack

Libraries

- eikcoctl.lib
- mediaclientaudio.lib

Example Application

- My Music Player.zip

Related links

- A tour to the S60 Touch UI components
- Working with Generic Button API - S60 Touch UI
- Working with Toolbar API - S60 Touch UI
- Working with LongTapDetector API - S60 Touch UI
- Working with Stylus Pop-up Menu API - S60 Touch UI
- Working with Tactile Feedback Client API - S60 Touch UI
- Working with Adaptive Search feature - S60 Touch UI
- Working with ChoiceList API - S60 Touch UI
- Working with SingleStyleTreeList with Hierarchical Lists API - S60 Touch UI
- Working with SingleColumnStyleTreeList with Hierarchical Lists API - S60 Touch UI

Reference list

- S60 5th edition SDK help
- S60 5th Edition C++ Developer's Library v1.0

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.