Capture Key sequence in non-gui application

This code will capture key sequence not key combination from keys 0-9,*,# in the background. Sometimes the user may need to capture a key sequence in the background and on capture he will take some action.

CAPABILITY SwEvent

KeyCapturer.h

```
#ifndef _KEYCAPTURER_
#define _KEYCAPTURER_

#include <apgwgnam.h>  //CApaWindowGroupName



/// Key capture callback.
class MKeyCallBack
{
public:

 /**
  *
  */
 virtual void KeysMap(const TDesC& keys) = 0;
};



///
class CKeyCapturer : public CActive
{
public:
 // enum for keys.
 enum TKeyCapture
  {
   EKeyCaptureNum0 = 48,
   EKeyCaptureNum1,
   EKeyCaptureNum2,
   EKeyCaptureNum3,
   EKeyCaptureNum4,
   EKeyCaptureNum5,
   EKeyCaptureNum6,
   EKeyCaptureNum7,
   EKeyCaptureNum8,
   EKeyCaptureNum9,
   EKeyCaptureHash = 127,
   EKeyCaptureAsterisk = 133
  };

public:

 /**
  * @param aCallBack The referece to key callback.
```


(C) Copyright Nokia 2014. All rights reserved.

```
   * @return CKeyCapturer class reference.
   */
  static CKeyCapturer* NewL(MKeyCallBack& aCallBack);

  /**
   * @param aCallBack The referece to key callback.
   * @return CKeyCapturer class reference.
   */
  static CKeyCapturer* NewLC(MKeyCallBack& aCallBack);

  /**
   *  @param aKeys Keys to map.
   *  @param aInterval Callback interval for key mapping.
   */
  void SetKeyCapture(const TDesC& aKeys, TInt& aInterval);

  /**
   *
   */
  virtual ~CKeyCapturer();

  /**
   * @desc Start listening.
   */
  void Listen();

 private:

  /**
   * @param aCallBack The referece to key callback.
   */
   CKeyCapturer(MKeyCallBack& aCallBack);

  /**
   *
   */
  void ConstructL();

  /**
   * @des Call by frame work.
   */
  void RunL();

  /**
   *
   */
  void DoCancel();

  /**
   * @desc It will call periodicall by timer after some interval.
   * @param aAny Pointer to TAny.
   */
  static TInt PeriodicTimerCallBack(TAny* aAny);

  /**
   * @desc It cheks key press event. If map sucessfully it will give call back.
   */
```


(C) Copyright Nokia 2014. All rights reserved.

```
 void CheckKeyPress();


private:
 MKeyCallBack& iCallBack;  // Key call back.
 RWsSession iWsSession; // Window session.
 RWindowGroup iWg;    // Windows group.
 TBuf<10>   iKeysToMap; // Store keys to map.
 TBuf<10>   iKeysPress; // Store Key press.
 CPeriodic*    iPeriodicTimer; // Periodic timer.
 TInt    iPeriodicInterval; // Periodic timer interval.
};

#endif // _KEYCAPTURER_
```

KeyCapturer.cpp

```
#include "KeyCapturer.h"
#include <apgtask.h>  // link against apgrfx.lib
#include <w32std.h>  // link against ws32.lib
#include <COEDEF.H>

const TInt KInterval(2*1000000);


///
CKeyCapturer* CKeyCapturer::NewL(MKeyCallBack& aCallBack)
{
 CKeyCapturer* self = CKeyCapturer::NewLC(aCallBack);
 CleanupStack::Pop(self);
 return self;
}



///
CKeyCapturer* CKeyCapturer::NewLC(MKeyCallBack& aCallBack)
{
 CKeyCapturer* self = new (ELeave) CKeyCapturer(aCallBack);
 CleanupStack::PushL(self);
 self->ConstructL();
 return self;
}


///
CKeyCapturer::CKeyCapturer(MKeyCallBack& aCallBack)
:CActive(EPriorityStandard),
iCallBack(aCallBack)
{
 iKeysPress = KNullDesC;
 iKeysToMap = KNullDesC;
 iPeriodicInterval = KInterval;
 iPeriodicTimer = NULL;
}
```


(C) Copyright Nokia 2014. All rights reserved.

```
///
CKeyCapturer::~CKeyCapturer()
{
 // cancel active shedular.
 Cancel();
 //Cancel CaptureKey
 iWg.CancelCaptureKey(EKeyCaptureNum0);
 iWg.CancelCaptureKey(EKeyCaptureNum1);
 iWg.CancelCaptureKey(EKeyCaptureNum2);
 iWg.CancelCaptureKey(EKeyCaptureNum3);
 iWg.CancelCaptureKey(EKeyCaptureNum4);
 iWg.CancelCaptureKey(EKeyCaptureNum5);
 iWg.CancelCaptureKey(EKeyCaptureNum6);
 iWg.CancelCaptureKey(EKeyCaptureNum7);
 iWg.CancelCaptureKey(EKeyCaptureNum8);
 iWg.CancelCaptureKey(EKeyCaptureNum9);
 iWg.CancelCaptureKey(EKeyCaptureAsterisk);
 iWg.CancelCaptureKey(EKeyCaptureHash);
 // close window group.
 iWg.Close();
 // Close window session.
 iWsSession.Close();
 // cancel periodic timer.
 if(iPeriodicTimer){
    iPeriodicTimer->Cancel();
    delete iPeriodicTimer;
    iPeriodicTimer = NULL;
 }
}



/// set key to map and interval.
void CKeyCapturer::SetKeyCapture(const TDesC& aKeys, TInt& aInterval)
{
 // set keys to map and interval.
 iKeysToMap = aKeys;
 iPeriodicInterval = aInterval;
}



///
void CKeyCapturer::ConstructL()
{
 CActiveScheduler::Add(this);
 // create session.
 User::LeaveIfError(iWsSession.Connect());
 // create window group.
 iWg = RWindowGroup(iWsSession);
 User::LeaveIfError(iWg.Construct((TUint32)&iWg, EFalse));
 // Enable app to recive event.
 iWg.SetOrdinalPosition(-1, ECoeWinPriorityNormal+2 );
```


(C) Copyright Nokia 2014. All rights reserved.

```
 iWg.EnableReceiptOfFocus(EFalse);
 // window group
 CApaWindowGroupName* wn = CApaWindowGroupName::NewLC(iWsSession);
 wn->SetHidden(ETrue);
 wn->SetWindowGroupName(iWg);
 CleanupStack::PopAndDestroy();
 // set key to capture.
 User::LeaveIfError(iWg.CaptureKey(EKeyCaptureNum0, 0,0));
 User::LeaveIfError(iWg.CaptureKey(EKeyCaptureNum1, 0,0));
 User::LeaveIfError(iWg.CaptureKey(EKeyCaptureNum2, 0,0));
 User::LeaveIfError(iWg.CaptureKey(EKeyCaptureNum3, 0,0));
 User::LeaveIfError(iWg.CaptureKey(EKeyCaptureNum4, 0,0));
 User::LeaveIfError(iWg.CaptureKey(EKeyCaptureNum5, 0,0));
 User::LeaveIfError(iWg.CaptureKey(EKeyCaptureNum6, 0,0));
 User::LeaveIfError(iWg.CaptureKey(EKeyCaptureNum7, 0,0));
 User::LeaveIfError(iWg.CaptureKey(EKeyCaptureNum8, 0,0));
 User::LeaveIfError(iWg.CaptureKey(EKeyCaptureNum9, 0,0));
 User::LeaveIfError(iWg.CaptureKey(EKeyCaptureAsterisk, 0,0));
 User::LeaveIfError(iWg.CaptureKey(EKeyCaptureHash, 0,0));
 // start it.
 Listen();
 // Initialize the periodic timer.
 iPeriodicTimer = CPeriodic::NewL(CActive::EPriorityIdle);
 // start periodic timer.
 iPeriodicTimer->Start( iPeriodicInterval, iPeriodicInterval,
TCallBack(PeriodicTimerCallBack, this));
}



/// Periodic timer call back.
TInt CKeyCapturer::PeriodicTimerCallBack(TAny* aAny)
{
 // get self reference.
 CKeyCapturer* self = static_cast<CKeyCapturer*>(aAny);
 // check key press.
 self->CheckKeyPress();
 // return.
 return KErrNone;
}



/// check key press event.
void CKeyCapturer::CheckKeyPress()
{
 // check lenght.
 if (iKeysToMap.Length()<=0)
  return;
 // check key press.
 if( iKeysPress.Find(iKeysToMap) != KErrNotFound )
 {
  // set call back that key combination is map.
  iCallBack.KeysMap(iKeysToMap);
 }
 // empty it.
 iKeysPress.Copy(KNullDesC);
```


(C) Copyright Nokia 2014. All rights reserved.

```
 // return.
 return;
}



///
void CKeyCapturer::RunL()
{
 if (iStatus == KErrNone)
 {
  // get windows server event
  TWsEvent e;
  iWsSession.GetEvent(e);
  // Get event type.
  TInt type = e.Type();
  // check event type.
  switch (type)
  {
  case EEventKey:
   {
    TInt code = e.Key()->iScanCode;
    // check scan code.
    switch(code)
    {
     case EKeyCaptureNum0:
      {
       iKeysPress.Append('0');
       break;
      }
     case EKeyCaptureNum1:
      {
       iKeysPress.Append('1');
       break;
      }
     case EKeyCaptureNum2:
      {
       iKeysPress.Append('2');
       break;
      }
     case EKeyCaptureNum3:
      {
       iKeysPress.Append('3');
       break;
      }
     case EKeyCaptureNum4:
      {
       iKeysPress.Append('4');
       break;
      }
     case EKeyCaptureNum5:
      {
       iKeysPress.Append('5');
       break;
      }
     case EKeyCaptureNum6:
```


(C) Copyright Nokia 2014. All rights reserved.

```
      {
       iKeysPress.Append('6');
       break;
      }
     case EKeyCaptureNum7:
      {
       iKeysPress.Append('7');
       break;
      }
     case EKeyCaptureNum8:
      {
       iKeysPress.Append('8');
       break;
      }
     case EKeyCaptureNum9:
      {
       iKeysPress.Append('9');
       break;
      }
     case EKeyCaptureHash:
      {
       iKeysPress.Append('#');
       break;
      }

     case EKeyCaptureAsterisk:
      {
       iKeysPress.Append('*');
       break;
      }
     default:
      break;
    }
    break;
   }
  case EEventKeyUp:
   break;
  case EEventKeyDown:
   break;
  default:
   break;
  }

  // Forward key event to focus application also.
  TInt wgId = iWsSession.GetFocusWindowGroup();
  iWsSession.SendEventToWindowGroup( wgId, e );
 }
 if (iStatus != KErrCancel)
 {
  Listen();
 }
}



///
void CKeyCapturer::DoCancel()
```


(C) Copyright Nokia 2014. All rights reserved.

```
{
 iWsSession.EventReadyCancel();
}



/// start active scheduler.
void CKeyCapturer::Listen()
{
 if (!IsActive()) {
 iWsSession.EventReady(&iStatus);
 SetActive();
 }
}
```


(C) Copyright Nokia 2014. All rights reserved.