Collision detection using CPeriodic to control movement

Overview

This snippet demonstrates how to implement the logic for basic collision detection in Symbian C++, which is useful, for example, in simple games. In the snippet, five blocks are drawn onto the screen and sent moving. Collisions against walls and other blocks are detected. Moving is handled with `CPeriodic`.

In this snippet, the code is implemented in a container control named `CAppView`. The control is derived from `CCoeControl`.

Header file

```
const TInt KBlocks = 5;
const TInt KBlockWidth = 10;
const TInt KBlockHeight = 10;
const TInt KBlockMaxSpeed = 3;

const TInt KDelay = 10000;
const TInt KInterval = 10000;
```

```
/**
 * Static callback function for timer, called periodically.
 * @param aPtr pointer to this class
 * @return one of S60 error codes
 */
static TInt TimerCallBack(TAny* aPtr);

/**
 * Timer function called from TimerCallBack.
 * Takes care of moving the blocks.
 * @return one of S60 error codes
 */
TInt DoTimer();

// Data

TPoint iPosition[KBlocks];  // positions for blocks
TPoint iSpeed[KBlocks];     // speeds for blocks
TRgb iColor[KBlocks];       // colors for blocks

CPeriodic* iTimer;
```

Source file

```
#include <e32math.h>
```

```
void CAppView::ConstructL(const TRect& aRect)
    {
    // Create a window for this application view
    CreateWindowL();

    // Create a periodic timer to move the blocks
    iTimer = CPeriodic::NewL(EPriorityNormal);

    // Initialize the blocks
    TInt i;
    for (i = 0; i < KBlocks; i++)
        {
        iSpeed[i] = TPoint(
                Math::Random() % (2 * KBlockMaxSpeed + 1) - KBlockMaxSpeed,
                Math::Random() % (2 * KBlockMaxSpeed + 1) - KBlockMaxSpeed);
        iPosition[i] = TPoint(
                Math::Random() % aRect.Width(),
                Math::Random() % aRect.Height());
        iColor[i] = TRgb(
                Math::Random() % 256,
                Math::Random() % 256,
                Math::Random() % 256);
        }

    // Start the timer
    iTimer->Start(KDelay, KInterval, TCallBack(TimerCallBack, this));

    // Set the window size
    SetRect(aRect);

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

```
TInt CAppView::TimerCallBack(TAny* aPtr)
    {
    return ((CAppView*)aPtr)->DoTimer();
    }
```


(C) Copyright Nokia 2013. All rights reserved.

```
TInt CAppView::DoTimer()
    {
    // Get gc to draw with
    CWindowGc& gc = SystemGc();

    gc.Activate(Window());
    gc.Clear();
    gc.SetPenStyle(CGraphicsContext::ENullPen);
    gc.SetBrushStyle(CGraphicsContext::ESolidBrush);

    // Handle each block in turn
    for (TInt i = 0; i < KBlocks; i++)
        {
        gc.SetBrushColor(iColor[i]);
        TPoint& position = iPosition[i];
        TPoint& speed = iSpeed[i];

        // Move blocks
        position = position + speed;

        TRect rect = Rect();

        // Detect block colliding against walls
        if ((position.iX + KBlockWidth >= rect.iBr.iX) ||
            (position.iX <= rect.iTl.iX))
            {
            speed.iX = -speed.iX;
            position = position + speed;
            }

        if ((position.iY + KBlockHeight >= rect.iBr.iY) ||
            (position.iY <= rect.iTl.iY))
            {
            speed.iY = -speed.iY;
            position = position + speed;
            }

        // Detect blocks colliding against each other
        TRect boxRect(position, TSize(KBlockWidth, KBlockHeight));
        for (TInt j = 0; j < KBlocks; j++)
            {
            TRect boxRect2(iPosition[j], TSize(KBlockWidth, KBlockHeight));
            if ((i != j) && (boxRect.Intersects(boxRect2)))
                {
                speed.iY = -speed.iY;
                speed.iX = -speed.iX;
                }
            }

        // Draw current block
        gc.DrawRect(boxRect);
        }
    gc.Deactivate();

    return KErrNone;
    }
```

Postconditions

Five blocks are bouncing on the screen. **COPY-PASTE** the desired platforms into the wiki text and save. All relevant version categories should be applied.
Windows Phone: [[Category:Windows Phone]]
[[Category:Windows Phone 7.5]]
[[Category:Windows Phone 8]]

Series 40: [[Category:Series 40]]
[[Category:Series 40 1st Edition]] [[Category:Series 40 2nd Edition]]
[[Category:Series 40 3rd Edition (initial release)]] [[Category:Series 40 3rd Edition FP1]] [[Category:Series 40 3rd Edition FP2]]
[[Category:Series 40 5th Edition (initial release)]] [[Category:Series 40 5th Edition FP1]]
[[Category:Series 40 6th Edition (initial release)]] [[Category:Series 40 6th Edition FP1]] [[Category:Series 40 Developer Platform 1.0]] [[Category:Series 40 Developer Platform 1.1]] [[Category:Series 40 Developer Platform 2.0]]

Symbian: [[Category:Symbian]]
[[Category:S60 1st Edition]] [[Category:S60 2nd Edition (initial release)]] [[Category:S60 2nd Edition FP1]] [[Category:S60 2nd Edition FP2]]
[[Category:S60 2nd Edition FP3]]
[[Category:S60 3rd Edition (initial release)]] [[Category:S60 3rd Edition FP1]] [[Category:S60 3rd Edition FP2]]
[[Category:S60 5th Edition]]
[[Category:Symbian^3]] [[Category:Symbian Anna]] [[Category:Nokia Belle]]

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.