Temporizador simples para jogos

Este artigo apresenta um temporizador (*timer*) simples para jogos, baseado na classe `CIdle`.

A classe `CIdle` é um objeto ativo que é executado quando não existe nenhum outro com prioridade maior esperando para ser executado. Ao contrário de outras alternativas encontradas no Symbian OS, `CIdle` não usa um intervalo de tempo pré-definido para gerar eventos.

Definindo a classe

```
class CCustomTimer : public CBase
{
  public:

    // Cria uma instância da classe. MTimerListener
    // é um objeto interessado ("interessado") em tratar o evento de tempo.
    static CCustomTimer* NewL (MTimerListener & aListener);

    ~CCustomTimer ();

    // Iniciar o temporizador.
    void Start  ();

    // Cancelar o temporizador.
    void Cancel ();


  private:

    // Função que é usada internamente para notificar o interessado
    // que vai receber o evento de tempo. Requerida pelo CIdle.
    static TInt IdleCallBack (TAny* aPtr);

  private:

    // Construtor, guarda a referência para o interessado.
    CCustomTimer (MTimerListener & aListener)
    : iListener (& aListener)
    {}

    // Segunda parte do construtor de duas fases.
    void ConstructL ();

  private:


    MTimerListener * iListener;  // o interessado
    CIdle*  iIdle;
};
```

A interface `MTimerListener` é definida como a seguir:

```
class MTimerListener
{
  public:

    virtual void OnTimer () = 0;
};
```

A classe `CCustomTimer` é implementada deste modo:

```
void CCustomTimer::ConstructL ()
{
  iIdle = CIdle::NewL (CIdle::EPriorityIdle);
}

CCustomTimer::~CCustomTimer ()
{
  if (iIdle)
  {
     iIdle->Cancel ();
     delete iIdle;
  }
}

void CCustomTimer::Start ()
{
  if (!iIdle->IsActive () )
   iIdle->Start (TCallBack (IdleCallBack, this) );
}

void CCustomTimer::Cancel ()
{
  iIdle->Cancel();
}


TInt CCustomTimer::IdleCallBack (TAny* aPtr)
{
  CCustomTimer* me = ((CCustomTimer*)aPtr);
  me->iListener->OnTimer ();
  return ETrue;
}
```

Uso

(C) Copyright Nokia 2013. All rights reserved.

```
// Suponha que o jogo deve rodar a 30 quadros por segundo. Dessa forma,
// é necessário calcular o intervalo de tempo (time step) para atender
// a esse requisito. O tempo é especificado em microsegundos.
const TInt KTargetTimeStep = 1000000 / 30;

void MyGame::OnTimer ()
{
   // Tratar dados de entrada (teclado)
   ProcessInput ();

   // Nessa parte será calculado o número de time steps a serem
   // executados, para que jogo não fique atrasado.

   // Guardar o tempo atual.
   TTime time;
   time.HomeTime();

   // Calcular o tempo decorrido desde a última atualização.
   // iStartTime é membro da classe MyGame.
   TInt elapsed = I64LOW (time.MicroSecondsFrom (iStartTime).Int64());

   // Atualizar o tempo total até o momento. A variável iTotalTime
   // também é membro da classe MyGame.
   iTotalTime += elapsed;

   // Calcular quantas atualizações devem ser executadas. Observe que
   // a divisão é inteira (ou seja, 5 / 2 = 2).
   elapsed = iTotalTime / KTargetTimeStep;

   for (TInt i = 0; i < elapsed; ++i)
    Update ();

   // Subtrair o tempo correspondente aos time steps processados.
   iTotalTime -= elapsed * KTargetTimeStep;
   iStartTime = time.Int64();

   // Desenhar.
   Render ();
}
```


(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.