```
//
// Programmer:    Craig Stuart Sapp <craig@ccrma.stanford.edu>
// Creation Date: Mon Dec 18 20:31:02 PST 2006
// Last Modified: Wed Jan  3 06:09:32 PST 2007
// Filename:      MzSpectralFlux.h
// URL:           http://sv.mazurka.org.uk/include/MzSpectralFlux.h
// Documentation: http://sv.mazurka.org.uk/MzSpectralFlux
// Syntax:        ANSI99 C++; vamp 0.9 plugin
//
// Description:   Calculate changes in spectral energy for onset detection.
//

#ifndef _MZSPECTRALFLUX_H_INCLUDED
#define _MZSPECTRALFLUX_H_INCLUDED

#include "MazurkaPlugin.h"  // Mazurka plugin interface for Sonic Visualiser
#include "MazurkaTransformer.h"
#include "MazurkaWindower.h"

#include <vector>

class MzSpectralFlux : public MazurkaPlugin {

   public:

   // plugin interface functions:

                    MzSpectralFlux          (float samplerate);
      virtual      ~MzSpectralFlux          ();

      // required polymorphic functions inherited from PluginBase:
      std::string   getName                 (void) const;
      std::string   getMaker                (void) const;
      std::string   getCopyright            (void) const;
      std::string   getDescription          (void) const;
      int           getPluginVersion        (void) const;

      // optional parameter interface functions
      ParameterList getParameterDescriptors (void) const;

      // required polymorphic functions inherited from Plugin:
      InputDomain   getInputDomain          (void) const;
      OutputList    getOutputDescriptors    (void) const;
      bool          initialise              (size_t channels,
                                             size_t stepsize,
                                             size_t blocksize);
      FeatureSet    process                 (float **inputbufs,
                                             Vamp::RealTime timestamp);
      FeatureSet    getRemainingFeatures    (void);
      void          reset                   (void);

      // optional polymorphic functions from Plugin:
      size_t        getPreferredStepSize    (void) const;
      size_t        getPreferredBlockSize   (void) const;
      // size_t     getMinChannelCount      (void) const { return 1; }
      // size_t     getMaxChannelCount      (void) const { return 1; }

   // non-interface functions and variables:

      static void   generateMidiNoteList    (std::vector<std::string>& alist,
                                             int minval = 0, int maxval = 127);
      static void   makeFreqMap             (std::vector<int>& mapping,
                                             int fftsize, float srate);
      static void   createMidiSpectrum      (std::vector<double>& midispectrum,
                                             std::vector<double>& magspec,
                                             double srate);
      static void   createWorkingSpectrum   (std::vector<double>& magpectrum,
                                             MazurkaTransformer& transformer,
                                             double srate, int spectrum_type,
                                             double smooth);
      static int    calculateSpectrumSize   (int spectrumType, int blocksize,
                                             double srate);
      static int    calculateMidiSpectrumSize(int transformsize, double srate);
      static double getMean                 (std::vector<double>& sequence,
                                             int mmin = -1, int mmax = -1);
      static double getStandardDeviation    (std::vector<double>& sequence,
                                             double mean);
      static int    localmaximum            (std::vector<double>& data,
                                             int target, int minimum,
                                             int maximum);
      static void   findOnsets              (std::vector<Vamp::RealTime>&
                                             onset_times,
                                             std::vector<double>&
                                             onset_levels,
                                             std::vector<double>&
                                             mean_function,
                                             std::vector<double>&
                                             threshold_function,
                                             std::vector<double>&
                                             scaled_function,
                                             std::vector<Vamp::RealTime>&
                                             functiontimes,
                                             double delta, double alpha);
      static double getSpectralFlux         (std::vector<double>&
                                             spectral_derivative,
                                             int fluxtype, double pnormorder);
      static void   smoothSpectrum          (std::vector<double>& sequence,
                                             double gain);
   private:

      int    mz_slope;         // how to calculate the harmonicness of a pitch
      int    mz_stype;         // how to calculate the harmonicness of a pitch
      double mz_pnorm;         // for calculating norm of spectral difference
      double mz_delta;         // local mean threshold for peak identification
      double mz_alpha;         // feedback gain for peak threshold function
      double mz_smooth;        // feedback gain for spectral smoothing

      std::vector<double>         mz_rawfunction; // store SF function for later
      std::vector<Vamp::RealTime> mz_rawtimes;    // times of raw SF function

      MazurkaTransformer   mz_transformer;  // interface FFTW Fourier transforms
      MazurkaWindower      mz_windower;     // interface for windowsing signals
      std::vector<double>  lastframe;       // store the last frame of spectrum

      // input parameters:
      //
      //    "method";          -- variant of spectral flux
      //    "windowsamples";   -- number of samples in audio window
      //    "stepsamples";     -- number of samples between window starts

};


#endif // _MZSPECTRALFLUX_H_INCLUDED
```