**Assignment 5:Peak Meter**

Amruta Vidwans and Milap Rane

We add a PeakMeter class to act as a PPM implementation, based on the MATLAB code provided.

The class definition is:

public:

void initPeakMeter(float fSamplingFreq, int iNumChannels);

void process(float \*\*ppfAudioData, int iNumOfFrames, float \*pfPeakValue);

void resetPeakMeterValues();

static void createInstance(CPeakMeter \*& pcPM);

static void destroyInstance(CPeakMeter \*& pcPM);

void setAlphaRT(float fRelTime);

void setAlphaAT(float fAttTime);

float getAlphaAT();

float getAlphaRT();

protected:

CPeakMeter();

virtual ~CPeakMeter();

private:

float m\_fSamplingFreq;

int m\_iNumChannels;

float m\_fAlphaRT;

float m\_fAlphaAT;

float \*m\_pfPreviousVPPM; // vppm at iHopLength

float \*m\_pfVPPM;

The initPeakMeter function initializes the PeakMeter with various member values, which include the sampling frequency, number of channels, Alpha Release time, and Alpha Attack Time, and also the past PPM Value and the current vaue. We have two set functions to set Alpha Attack Time and Alpha Release Time separately. These two also have getter functions.

We applied 6 tests for the PPM:

1. Zero input
2. DC input
3. Alpha AT change(with Impulse input)
4. Alpha Rt change(with DC input that drops to zero)
5. Sampling frequency change
6. Delayed Impulse input

All these tests were first implemented in MATLAB and then the textfiles are stored and read from in every iteration of TestPPM.cpp being called.

We also have a MeterComponent added for the GUI:

class MeterComponent: public Component

{

public:

void paint(Graphics& g);

void setValue(float val);

float getPeakValue();

void setPeakValue(float val);

MeterComponent();

~MeterComponent();

private:

float m\_fPeakVal;

float m\_fMaxPeakVal;

};

m\_fPeakVal shall be used as the PPM value and the m\_fMaxPeakVal stores the Past Peak PeakMeter value.