Inverse Fourier Transform

$$f(t) = \sum_{v=-\infty}^{\infty} F(v)e^{2\pi ivt}$$

$$f(t) = \int_{v=-\infty}^{\infty} F(v)e^{2\pi ivt}dv$$

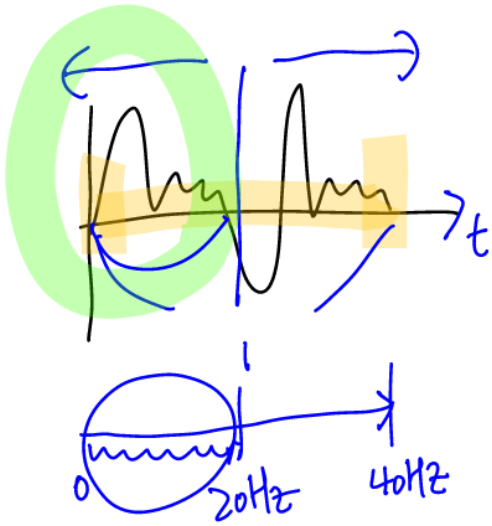
Forward Fourier Transform

$$F(v) = \sum_{t=-\infty}^{\infty} f(t)e^{-2\pi ivt}$$

$$F(v) = \int_{t=-\infty}^{\infty} f(t)e^{-2\pi ivt}dt$$

Hint

$$\frac{f(t)}{e^{2\pi ivt}}$$



Discrete Fourier Transform

$$F(k) = \sum_{n=0}^{N-1} f(n)e^{-2\pi i k\frac{n}{N}}$$

Amblitude

$$\frac{|F(k)|}{N} = \frac{\sqrt{Re(F(k))^2 + Im(F(k))^2}}{N}$$

Inverse Transform

$$f(n) = \frac{1}{N}\sum_{k=0}^{N-1} F(k)e^{2\pi ik\frac{n}{N}}$$

conjugate

Implementation

```
using ComplexArray = std::valarray<std::complex<double> >;
```

Preparing Signal

```
        std::complex<double> test[BIN_SIZE];
        // fill test signal data
        double x = 0;
        double y;
        for(int i = 0; i < BIN_SIZE; ++i)
        {
            test[i] = std::complex<double>( SignalFunction( x )
);
            x += SAMPLING_STEP;
        }
```

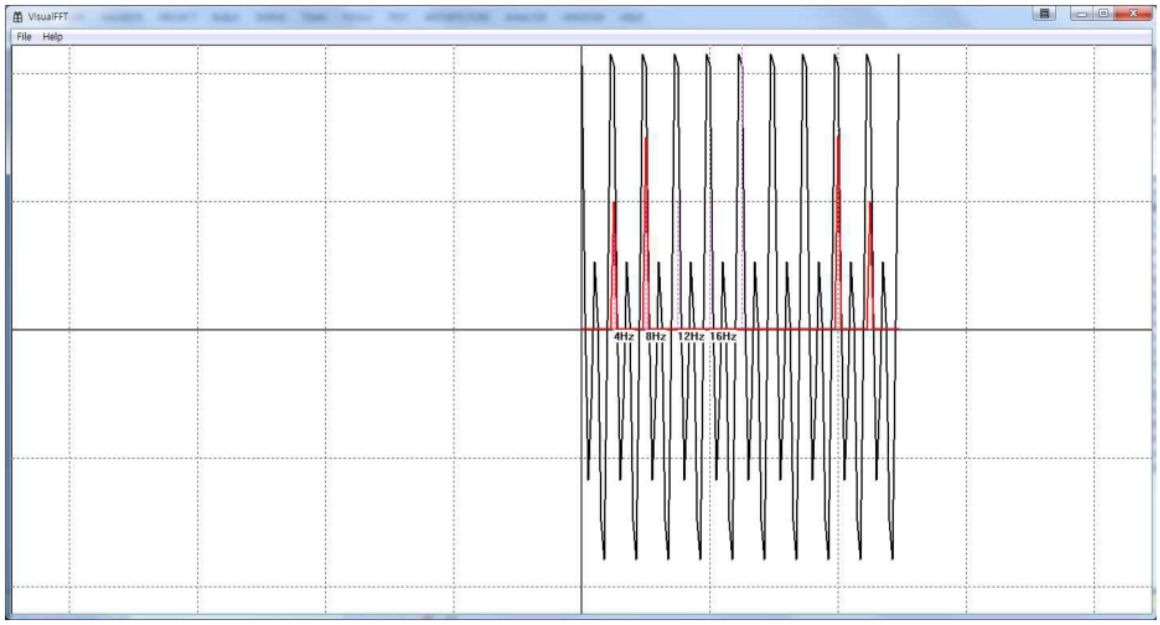
Forward Fourier Transform

```
void dft(ComplexArray& x)
{
    const size_t N = x.size();

    ComplexArray xresult(N);

    for (size_t k = 0; k < N; ++k)
    {
        xresult[k] = 0;
        for (size_t n = 0; n < N; ++n)
        {
```

```
                std::complex<double> e = std::polar(1.0, -2 * M_PI
* k * (double)n / (double)N);
                xresult[k] += x[n] * e;
            }
        }
        x = xresult;
}
```

Interpretation of the result

```
                // display Hz for every 10 steps
                if(i % 10 == 0)
                {
                    KVectorUtil::DrawLine( g_hdc, KVector2(
(double)x, (double)1 ), KVector2( (double)x, (double)0 ), 1,
PS_DOT, RGB( 255, 0, 255 ) );
                    KVector2 screenPos =
KVectorUtil::GetScreenPoint( KVector2( x, 0.0 ) );
                    char buffer[80];
                    double ratio = (double)i / double(BIN_SIZE);
                    sprintf_s( buffer, "%gHz", ratio /
SAMPLING_STEP );
                    ::TextOutA( hdc, (int)screenPos.x,
(int)screenPos.y, buffer, strlen( buffer ) );
                }
```



Fast Fourier Transform

@