Module IN3031 / INM378 Digital Signal Processing and Audio Programming

Tillman Weyde t.e.weyde@city.ac.uk

FMOD Custom DSP Programming

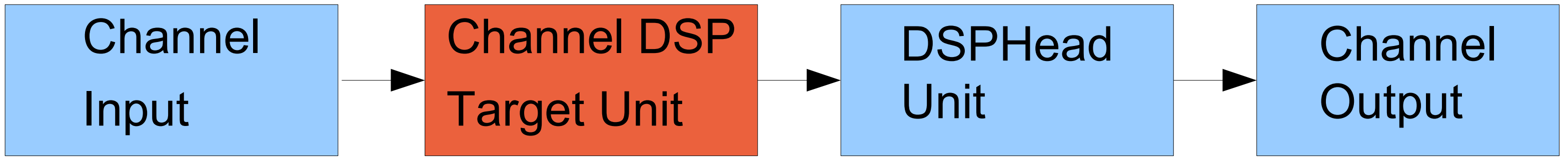
Custom FMOD DSPs

- DSP inserts
  - For whole system (all channels):

    ```
    system->addDSP()
    ```

  - For specific channel

    ```
    channel->addDSP()
    ```



Creating a custom DSP

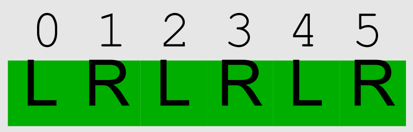
```
// Create a DSP descripton
FMOD_DSP_DESCRIPTION dspdesc;
memset(&dspdesc, 0, sizeof(dspdesc));

strncpy_s(dspdesc.name, "My first DSP unit",
    sizeof(dspdesc.name));
dspdesc.numinputbuffers = 1;
dspdesc.numoutputbuffers = 1;
dspdesc.read = DSPCallback;


// Create your new DSP object
result = system->createDSP(&dspdesc, &dsp);
FmodErrorCheck(result);
```

Custom DSP callback

```
FMOD_RESULT F_CALLBACK DSPCallback(FMOD_DSP_STATE *dsp_state,
    float *inbuffer, float *outbuffer,
    unsigned int length, int inchannels, int *outchannels)
{
    for (unsigned int samp = 0; samp < length; samp++)
    {
        for (int chan = 0; chan < *outchannels; chan++)
        {
            outbuffer[(samp * *outchannels) + chan] =
                inbuffer[(samp * inchannels) + chan];
        }
    }
};
```

Example:

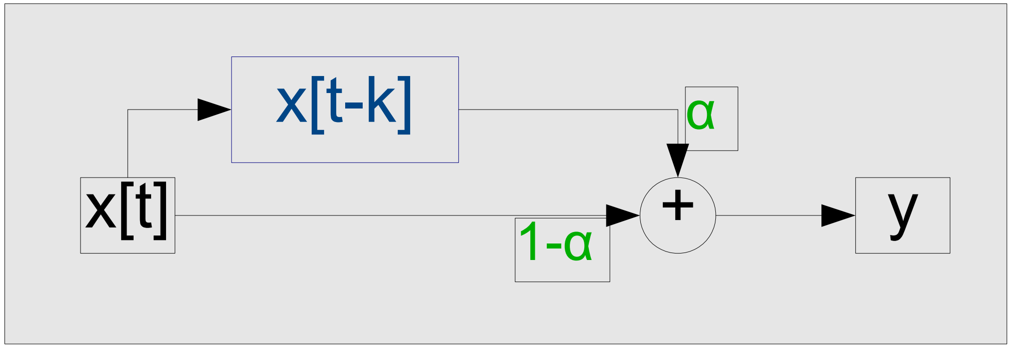
2 channel `inbuffer`



Creating a Delay effect

- Echo adds a **delayed** signal to the original input
- Both delayed and original signal are scaled to stay in range, with $0 \le \alpha \le 1$

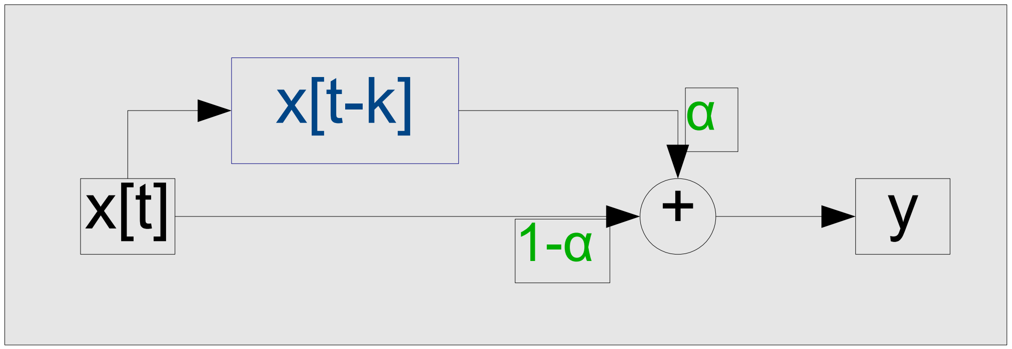
- **Signal flow diagram** of the effect:



Accessing x[t-k]

- Buffer the input signal for a time at least equal to the sample delay time k
- Then access x[t-k] from the buffer

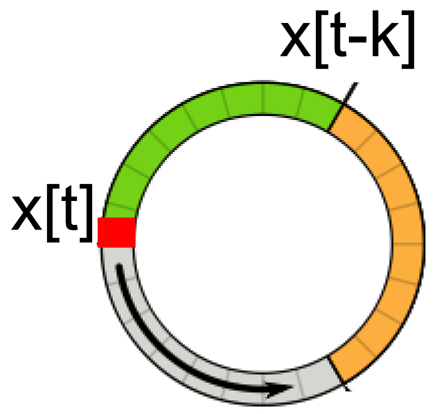
0 <= α <=1



Circular Buffer

- Goal: continuous buffering of incoming data in linear array
- Address the buffer:

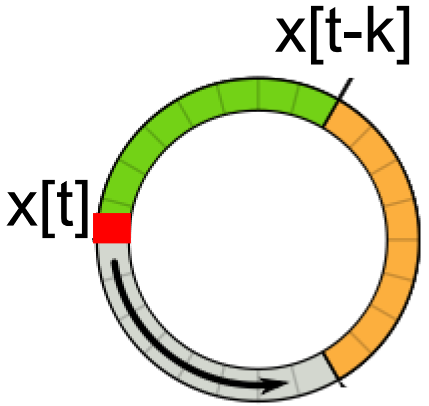
```
pos % buffer_length
```

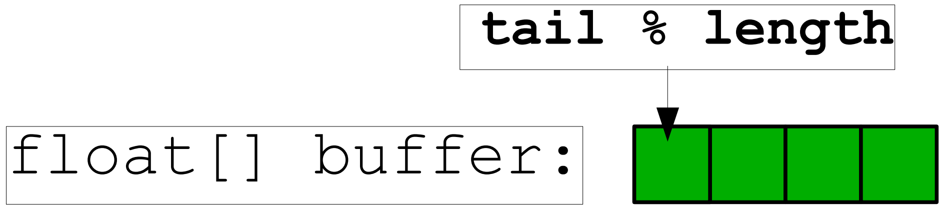


Circular Buffer

- Minimise buffer maintenance costs
- Address the buffer:

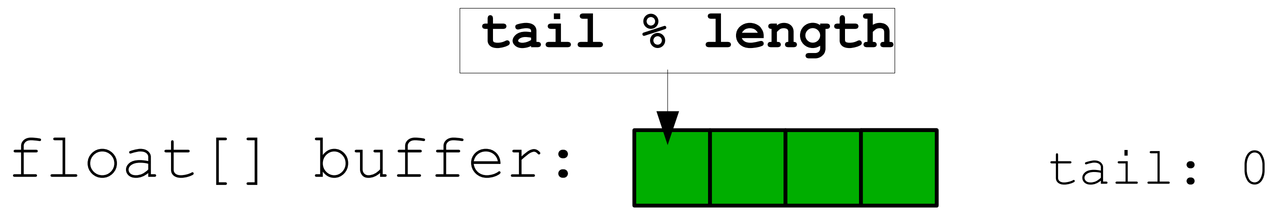
  `pos % buffer.length`
- Use index `tail` to manage the buffer
- `tail % length` points to write position





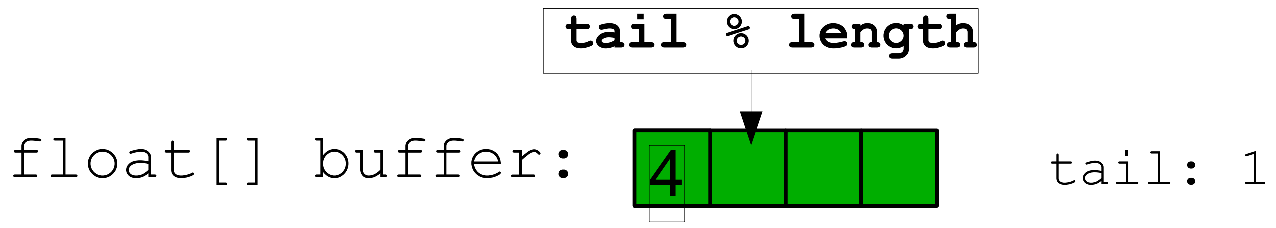
Circular Buffer

```
CircBuffer *cBuffer = new CircBuffer(4);
```



Circular Buffer

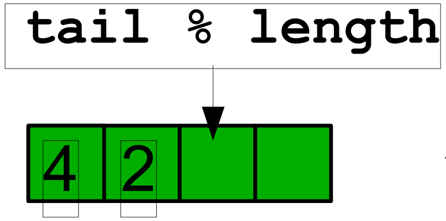
```
CircBuffer cBuffer = new CircBuffer(4);

Cbuffer->Put(4.0);
```



Circular Buffer

```
CircBuffer cBuffer = new CircBuffer(4);

Cbuffer->Put(4);

Cbuffer->Put(2);
```

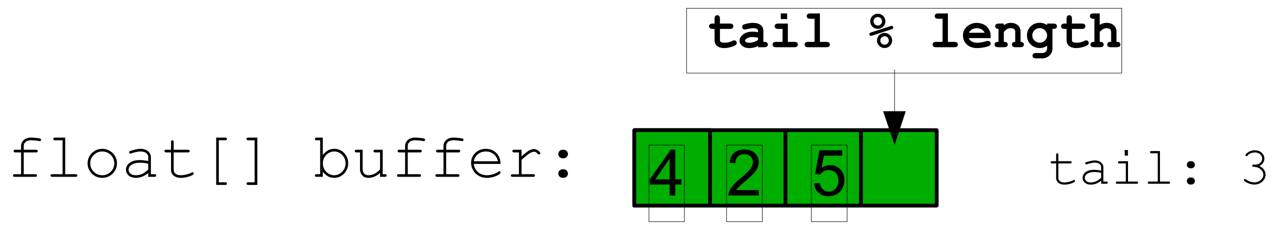
float[] buffer:



tail: 2

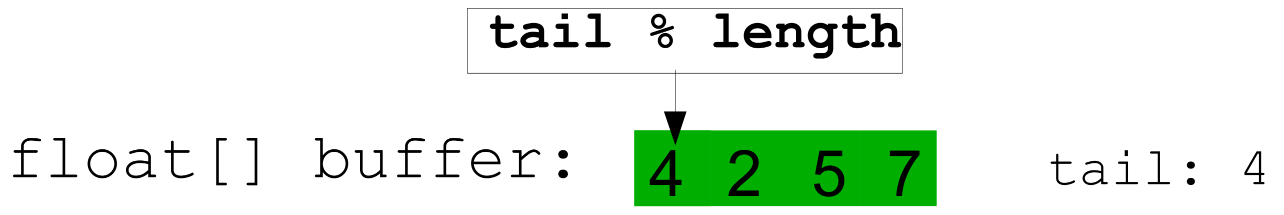
Circular Buffer

```
CircBuffer cBuffer = new CircBuffer(4);
Cbuffer->Put(4);
Cbuffer->Put(2);
Cbuffer->Put(5);
```



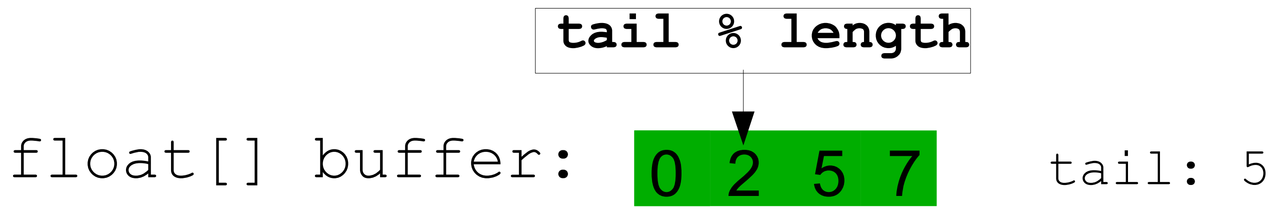
Circular Buffer

```
CircBuffer cBuffer = new CircBuffer(4);
Cbuffer->Put(4);
Cbuffer->Put(2);
Cbuffer->Put(5);
Cbuffer->Put(7);
```



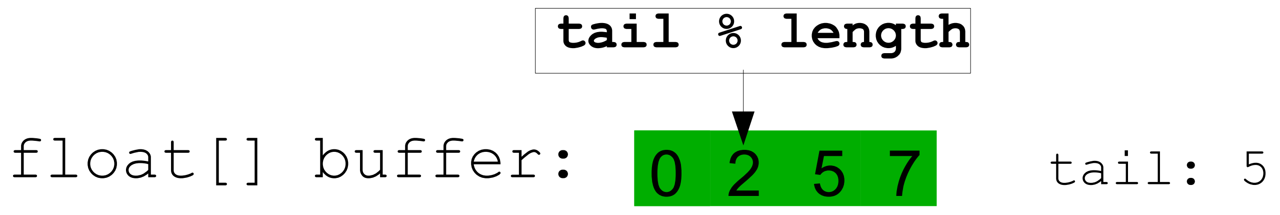
Circular Buffer

```
CircBuffer cBuffer = new CircBuffer(4);
Cbuffer->Put(4);
Cbuffer->Put(2);
Cbuffer->Put(5);
Cbuffer->Put(7);
Cbuffer->Put(0);
```



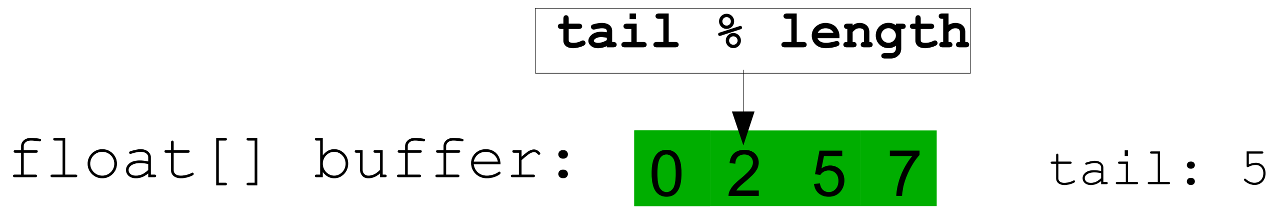
Circular Buffer

```
cBuffer->AtPosition(2);
  // returns: 5
```



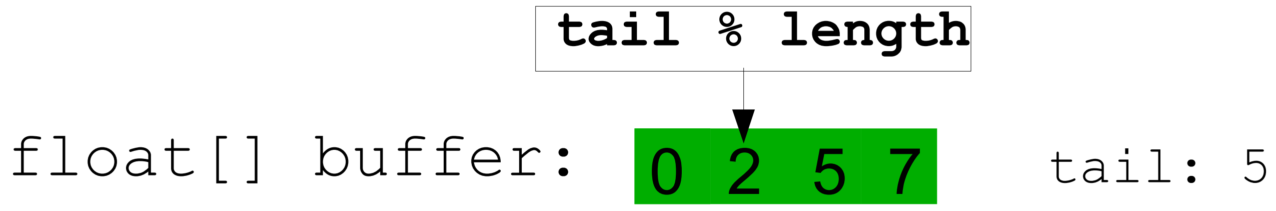
Circular Buffer

```
cBuffer->AtPosition(2);
  // returns: 5
cBuffer->AtPosition(4);
  // returns: 0
```



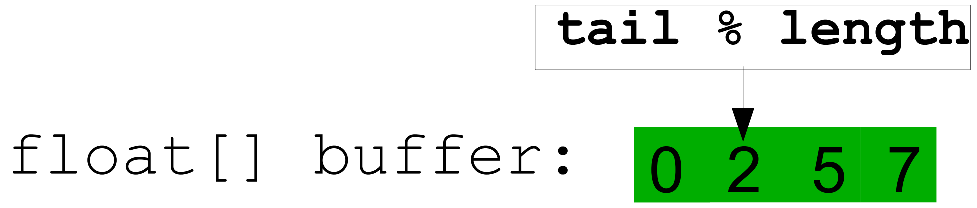
Circular Buffer

```
cBuffer->AtPosition(2);
  // returns: 5
cBuffer->AtPosition(4);
  // returns: 0
cBuffer->AtPosition(6);
```



Circular Buffer

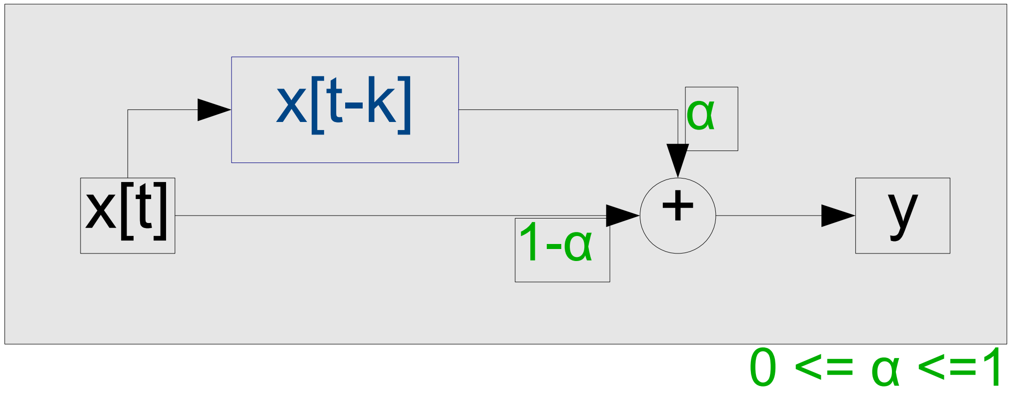
```
cBuffer->AtPosition(2);
  // returns: 5
cBuffer->AtPosition(4);
  // returns: 0
cBuffer->AtPosition(6); // throws Exception! Why?
```



Echo effect

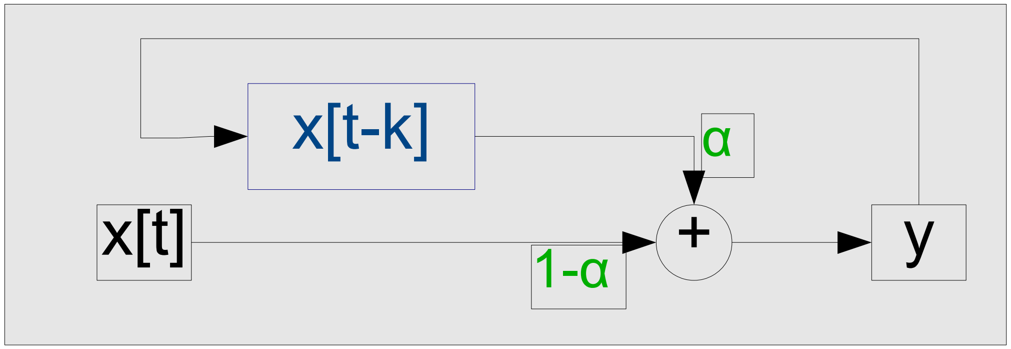
- Echo adds a **delayed** signal to the original input
- Both delayed and original signal are scaled to stay in range

- **Signal flow diagram** of the effect:



Feedback delay

- There is also a feedback version of the delay: The output is **fed back** into the buffer!
- **Simplified signal flow diagram** of the effect:



0 <= α <=1

Frequency analysis in FMOD

```
// create FFT DSP object
DSP * fft;
system->createDSPByType(FMOD_DSP_TYPE_FFT, &fft);
// define spectrum length and window
fft->setParameterInt(FMOD_DSP_FFT_WINDOWSIZE, 1024);
Fft->setParameterInt(FMOD_DSP_FFT_WINDOWTYPE,
    FMOD_DSP_FFT_WINDOW_HANNING);

// get spectrum data
FMOD_DSP_PARAMETER_FFT *fftData;
fft→getParameterData(FMOD_DSP_FFT_SPECTRUMDATA,
    (void **)fftData);
```

Reading

- FMOD Studio Low-level API tutorials on DSP architecture and usage