

Summary of Lesson 17

Functions , introduced in this lesson

```
dsp_initclass();
// init class as a dsp object

void dsp_add(t_perfroutine p, long argc, ...);
// add perform function to dsp chain
```

ex1.c

```
#include "ext.h"
#include "z_dsp.h"

// sig~

typedef struct {
    t_pxobject b_ob;
    int value;
    void *out;
} t_ex1;

void ex1_int(t_ex1 *x, long input);
void ex1_dsp(t_ex1 *x, t_signal **sp, short *count);
t_int *ex1_perform(t_int *w);
void *ex1_new(t_symbol *s, short argc, t_atom *argv);
void ex1_free(t_ex1 *x);

void *ex1_class;

void main()
{
    setup((t_messlist **)&ex1_class, (method)ex1_new, 0L, (short)sizeof(t_ex1),
    0L, 0L);
    addmess((method)ex1_dsp, "dsp", A_CANT,0);
    addint((method)ex1_int);
    dsp_initclass();
}

void ex1_dsp(t_ex1 *x, t_signal **sp, short *count)
{
    dsp_add(ex1_perform, 3,x,sp[1]->s_vec,sp[0]->s_n);
}

void ex1_int(t_ex1 *x, long input)
{
    x->value = input;
```

```

}

t_int *ex1_perform(t_int *w)
{
    float *out1;
    t_ex1 *x;
    long vs,i;

    x = (t_ex1*)(w[1]);
    out1 = (float*)(w[2]);
    vs = w[3];

    for(i=0;i<vs;i++)
    {
        out1[i] = x->value;
    }

    return w+4;
}

void *ex1_new(t_symbol *s, short argc, t_atom *argv)
{
    t_ex1 *x;
    x = (t_ex1*)newobject(ex1_class);
    dsp_setup((t_pxobject*)x, 1);
    outlet_new((t_pxobject *)x, "signal");
    return x;
}

```

ex2.c

```

#include "ext.h"
#include "z_dsp.h"

// noise~

typedef struct {
    t_pxobject b_ob;
    void *out;
} t_ex2;

void ex2_dsp(t_ex2 *x, t_signal **sp, short *count);

void *ex2_new(t_symbol *s, short argc, t_atom *argv);
void ex2_free(t_ex2 *x);
t_int *ex2_perform(t_int *w);

void *ex2_class;

void main()

```

```

{
    setup((t_messlist **)&ex2_class, (method)ex2_new, 0L, (short)sizeof(t_ex2),
0L, 0L);
    address((method) ex2_dsp, "dsp", A_CANT,0);
    dsp_initclass();
}

void ex2_dsp(t_ex2 *x, t_signal **sp, short *count)
{
    dsp_add(ex2_perform, 2 ,sp[1]->s_vec,sp[0]->s_n);
}

t_int *ex2_perform(t_int *w)
{
    float *out;
    int vs;
    out = (float *) (w[1]);
    vs = w[2];

    for(;vs;vs--)
    {
        out[vs] = (float)(rand() % 10000 - 5000) / 5000.0;
    }
    return w+3;
}

void *ex2_new(t_symbol *s, short argc, t_atom *argv)
{
    t_ex2 *x;
    x = (t_ex2*)newobject(ex2_class);
    dsp_setup((t_pxobject*)x, 1);
    outlet_new((t_pxobject *)x, "signal");

    return x;
}

```

ex3.c

```

#include "ext.h"
#include "z_dsp.h"

// bad tuned phasor~ fixed to 441Hz
// 44100 / 441 = 100 (samples to reach to 1)

typedef struct {
    t_pxobject b_ob;
    int current;
    void *out;
} t_ex3;

```

```

void ex3_dsp(t_ex3 *x, t_signal **sp, short *count);

void *ex3_new(t_symbol *s, short argc, t_atom *argv);
t_int *ex3_perform(t_int *w);

void *ex3_class;

void main()
{
    setup((t_messlist **)&ex3_class, (method)ex3_new, 0L, (short)sizeof(t_ex3),
    0L, 0L);
    address((method) ex3_dsp, "dsp", A_CANT,0);
    dsp_initclass();
}

void ex3_dsp(t_ex3 *x, t_signal **sp, short *count)
{
    dsp_add(ex3_perform, 3 ,x,sp[1]->s_vec,sp[0]->s_n);
}

t_int *ex3_perform(t_int *w)
{
    float *out;
    int i,vs;
    t_ex3 *x = (t_ex3*)(w[1]);
    out = (float *)(w[2]);
    vs = w[3];

    for(i = 0;i <vs;i++)
    {
        out[i] = x->current * 0.01; // (step) 1/ 100
        x->current++;
        if(x->current > 99 )
        {
            x->current = 0;
        }
    }

    return w+4;
}

void *ex3_new(t_symbol *s, short argc, t_atom *argv)
{
    t_ex3 *x;
    x = (t_ex3*)newobject(ex3_class);
    dsp_setup((t_pxobject*)x, 1);
    outlet_new((t_pxobject *)x, "signal");
    x->current = 0;
    return x;
}

```
