Kingsley’s guide to basic max object programming.

Introduction

1) start with the simplemax example
2) compile the example & test in a max patch - there should be 1 inlet and no outlets, with some messages posted to the max window when you put the object in
3) Each time you compile and test in a max patch, why not change the message it prints out to the max window when it starts to make sure you’re running the right version?

Task 1: Adding a second inlet

This is to get a second inlet on our object - the first one is created by default so we only need to go through this process if we want more than one input.

There are 4 parts to this task, as follows

1) Call a built-in max SDK function that creates the second inlet:

This needs to be done in the function that is called when the object is created, so in the “simplemax_new” function

To create the inlet we need to call the function

```
intin(x, 1);
```

This creates a second inlet to receive an integer... hence intin. The first argument, x, is a variable that is defined at the top of the simplemax_new function, and given a value when the object is created with the object_alloc function. It is basically a pointer to where the objects data is stored in the memory. The second argument is the inlet number - the first one is number 0 and is created by default, so this one is number 1.

So: put the line

```
intin(x, 1);
```

in the simplemax_new function at some point after the object_alloc function has been called.

2) Make a function to do something whenever this second inlet receives a number:

   Make a new function at the bottom of the program that will be called whenever the object receives a number in the inlet we’ve just created - this function needs two arguments, one is the pointer to the object’s data and the other is the value that comes in from the patch. Try this one first - it just prints the value to the max window:

```
void kingsley_in1(t_simplemax *x, long n)
```

```
{
	// do something with n like this for example
	object_post((t_object *)x, "i've just received the number %ld", n);
}
```

3) Declare this function at the top of the program:

Just as we do for any program we need to tell it we've made a new function at the bottom of the page so we can call it higher up without an error.

So, put this:

```
void kingsley_in1(t_simplemax *x, long n);
```

at the end of the other function declarations (simplemax_new, simplemax_free, etc.) after the headers and object struct declaration.

4) Finally, associate our new function with the inlet so it gets called whenever a value arrives in the patch:

In the int main(void) {} function we need to add this line:

```
	class_addmethod(c, (method)kingsley_in1, "in1", A_LONG, 0);
```

There's already one of these class_addmethod's in there that looks a bit like this for the "assist" stuff, so put it under there.

It tells the object what to do when it receives an integer at inlet 1, hence "in1".

Ok, that should be it. Save all, build, restart max and put in the new object.... you should get two inlets and if you send an int to the right hand one then it should print the number out in the Max window.

Task 2: Adding an outlet

This is easier than adding an inlet, as we don't need to make a function and associate it with it, we just need to create the outlet and send some stuff out of it....

So only two jobs to do here:

1) Declare the outlet in our data structure:

To do this we put the following variable declaration in the typedef struct { } thing:

```
	void*						kingsley_output;
```

Job done.

2) Make an outlet and assign it to the variable we've just made.

To do this we use a built-in function of the Max SDK called intout... this will give us an outlet that sends out integers. Other outlet types will need other functions (eg floatout for a float output... obviously).

We call this intout function just after we've created our object, so right next to where we put that intin() function in simplemax_new.

```
    x->kingsley_output = intout((t_object *)x);
```

And that's the outlet created.

Save all, build, restart max/msp and put your object in.... it should now have an outlet.

Task 3: Sending a value out of our new outlet.

Ok, now we've got an outlet, how do we use it?

Let's add another line into that function we made earlier that is called whenever our second inlet receives an integer... remember before it just printed it to the max window using:

```
    object_post((t_object *)x, "i've just received the number %ld", n);
```

Well now we have an outlet, we can call another max SDK function to send a value out, called outlet_int:

```
    outlet_int(x->kingsley_output, n);
```

x->kingsley_output is the pointer to our outlet that we made, and n is the value we want to send to it.

That's it - save all, build, restart max/msp and put your object in.... connect a number box to the right inlet and another to the outlet. Send integers into the right inlet and they should be repeated to the outlet.

Now we're getting somewhere. But the original left input still doesn't do anything.... let's sort that out.

Task 4: Doing something with the left input.

Hopefully it is starting to make sense.... to make the left input do something we have to make a function that will be called when it receives a value in the patch - just copy the one we made earlier for the right input and change the name a bit....

```
void kingsley_in0(t_simplemax *x, long n)
```

```
{
	// do something with n
	object_post((t_object *)x, "i've just received %ld in my left input", n);
	outlet_int(x->kingsley_output, n);
}
```

Now the other bits:

1) Declare the function at the start of the program:

```
void kingsley_in0(t_simplemax *x, long n);
```

2) Associate the function with the left integer intput in the int main (void) {} function:

```
	class_addmethod(c, (method)kingsley_in0, "int", A_LONG, 0);
```

Done. Save, build, restart max and test with a number box on each inlet and outlet to see if it behaves as you would expect.

Now you can hopefully start to use the inputs and outputs in a more useful way, but this should get you started.