Notes quant à l'utilisation de la librairie OSC (open sound control), OscP5, développée par Andreas Schlegel (http://www.sojamo.de/iv/index.php).

Structure de la classe OscP5

Deux des constructeurs possibles :

```
OscP5(theParent, theReceiveAtPort);
OscP5(theParent, theAddress, thePort);
```

Avec une instance de la classe OscP5, on rédige comme suit :

```
// Appel de la libraire
import oscP5.*;
import netP5.*;
// Déclaratiion d'une instance OscP5
OscP5 monOsc;
// Déclaratiion d'une instance NetAddress
NetAddress destination;
// Ouverture d'une connexion Osc sur le port de Processing (12000)
monOsc = new OscP5(this, 12000);
// On confirme une adresse de destination : adresse IP de la machine
// qui reçoit et choix du port sur cette machine.
destination = new NetAddress("127.0.0.1", 12345);
// Plus loin dans le code, après la fabrication d'un message OscMessage...
monOsc.send(monMessageEnvoi, destination);
```

Structure d'un OscMessage

OscMessage (theAddrPattern, theArguments);

Les arguments d'un message Osc (un tableau d'objets) sont d'abord précédés d'un Osc AddressPattern (un type défini par une étiquette en format String). Le reste du paquet est précisé par la méthode .add. Les données ajoutées au tableau d'objets sont soit des entiers (int), des fractions (float) ou des chaînes de caractères (String).

Par exemple :

```
// On déclare une instance de OscMessage puis on choisi un identificateur de message.
OscMessage monMessageEnvoi = new OscMessage("/osc/
midi/out/noteOn");
// On ajoute ensuite le nombre d'arguments que l'on souhaite transmettre.
monMessageEnvoi.add(0); // le canal midi
monMessageEnvoi.add(60); // La note
monMessageEnvoi.add(127); // La vélocité
monOsc.send(monMessageEnvoi, destination);
```

Le format des données acceptées sont diverses :

```
monMessageEnvoi.add(123); // add an int to the osc message
monMessageEnvoi.add(12.34); // add a float to the osc message
monMessageEnvoi.add("some text"); // add a string to the osc message
monMessageEnvoi.add(new byte[] {0x00, 0x01, 0x10, 0x79});
// add a byte blob to the osc message
monMessageEnvoi.add(new int[] {1,2,3,4});
// add an int array to the osc message
```

En fait, la documentation décrit les formats suivants :

```
blobValue ( ) // get the byte array (blob) of the osc argument.
booleanValue ( ) // get the boolean value of the osc argument.
bytesValue ( ) // get the byte array of the osc argument.
charValue ( ) // get the char value of the osc argument.
doubleValue ( ) // get the double value of the osc argument.
floatValue ( ) // get the float value of the osc argument.
intValue ( ) // get the int value of the osc argument.
longValue ( ) // get the long value of the osc argument.
stringValue ( ) // get the String value of the osc argument.
```

Envoi d'un message

On peut donc écrire directement le send comme suit :

```
monOsc = new OscP5(this, 12000);
monOsc.send("/osc/midi/out/noteOn", new Object[] {new
Integer("1"), new Float(2.0), "Chaine de caractères"}, new
NetAddress("127.0.0.1", 12345));
```

Dans cet exemple, deux choses :

- la connexion Osc dans Processing est prête à faire l'écoute de messages pouvant lui être adressés;
- on envoie un message à une application dont le port d'écoute serait le 12345. Cela pourrait être une autre application (Max/MSP, par exemple).

Le message possède maintenant plusieurs paquets.

Pour connaître le nombre de paquets et le format des messages, utiliser la méthode suivante dans un oscEvent :

```
monMessageReception.typetag();
// retourne par exemple "ifs", ce qui veut dire que le premier élément du tableau est
un entier, le second, un float et le dernier, un String.
```

Réception d'un message

Il faut créer une méthode d'écoute à l'extérieur des méthodes setup() et draw(). Cette méthode est donc active pour tout message susceptible d'être intercepté. Il nous faudra donc filtrer les messages suivant leurs propriétés.

```
void oscEvent(OscMessage monMessageReception) {
// On insère ici les tâches de filtrage des messages
}
```

On décortique le message Osc à la réception comme suit :
D'abord, identifier l'étiquette d'adressage (addressPattern), puis décomposer les arguments comme s'il s'agissait d'un tableau.

```
if ((monMessageReception.checkAddrPattern("/osc/midi/out/
noteOn") {
int canal = monMessageReception.get(0).intValue());
int note = monMessageReception.get(1).intValue());
int velocite = monMessageReception.get(2).intValue());
}
```

L'instruction conditionnelle pourrait aussi s'appuyer sur le format des messages entrant. Peu importe l'identificateur, il suffirait d'intercepter les messages comportant trois entiers consécutifs :

```
if (monMessageReception.typetag().equals("iii") {
}
```

Note : Pour être à l'écoute des messages, il nous faut fonctionner dans Processing en mode continu (avec un draw), même si rien n'a été codé dans cette méthode. Par ailleurs, je recommande d'y placer une instruction vide, comme suit :

```
void draw() {
  ;
}
```

Enfin, quand la réception d'un message Osc se fait dans un autre applet Processing, il ne faut pas oublier de déclarer une instance de la classe OscP5 :

```
OscP5 oscP5;
```

... puis de créer l'objet dans le setup() en spécifiant bien un port d'écoute unique, c'est-à-dire le numéro arbitraire de port utilisé dans le NetAddress de destination de l'applet émetteur :

```
oscP5 = new OscP5(this, 12345);
```

Usages

L'emploi de la librairie Osc ne restreint pas qu'à l'envoi de messages du type Midi. Au contraire, il faut le voir comme un protocole ouvert, permettant de lier une ou des données à une étiquette particulière, cette dernière étant par la suite exploitée comme identificateur (inputTag dans QuartzComposer par exemple).

```
OscMessage monMessage = new OscMessage("/temps");
monMessage.add(millis());
```

Les numéros de port de réception OSC par défaut

- Le port de Processing lorsque non spécifié est : 12000
- Le port de SuperCollider est le 57120
- 5500 est le numéro de port du patch OscReceiveServer (un module ajouté manuellement) de Quartz Composer dans Tiger (10.4). Quartz Composer sur Leopard et sur Snow Leopard propose un module de réception OSC dans ses patch réguliers.
- On peut aussi utiliser «Occam» (OSC-to-MIDI) pour faire circuler des messages via d'autres ports et relier Processing à des applications Midi pouvant reconnaître Occam dans ses sources Midi (Midi input). Bien entendu, le codage du message dans ces circonstances devra respecter le protocole Midi. On détermine arbitrairement le numéro de port dans Occam.

Autres applications

iPodTouch/iPhone : avec TouchOSC (http://hexler.net/touchosc)
Pour les téléphones cellulaires en général : mrmr (http://poly.share.dj/projects/#mrmr)
Flash (via XML Socket) : avec Flosc (http://www.benchun.net/flosc/)
Osculator (http://www.osculator.net/)

• Note supplémentaire provenant du site http://archive.cnmat.berkeley.edu/OpenSoundControl/Max/

UDP Versus OSC

UDP, the "User Datagram Protocol," is the Internet Protocol for sending packets (a.k.a. "datagrams") between machines without establishing a connection between the machines and without any expensive mechanism for detecting lost packets and retrying them. UDP is used for streaming video and audio and many other applications on the Internet.
Open Sound Control is an application-level protocol invented by CNMAT. OSC defines only the bit format and interpretation of those bits; you could transmit OpenSound Control messages via UDP, TCP, shared memory, compact disc, a serial port, or any other digital medium.
Because UDP can be used to transmit many kinds of data besides OSC, and because OSC can be transmitted by many kinds of networking technology besides UDP, CNMAT's implementations put the UDP part and the OSC part in separate objects. People have used the OTUDP object without the OpenSoundControl object to send and receive data in formats other than OSC. (This requires writing a max external to translate between Max data and binary data in the non-OSC format.) These uses of OTUDP would also work with udpsend/udpreceive (via the "FullPacket" message).Conversely, people have used OpenSoundControl without UDP as a sort of super pack/unpack, to be able to pass entire OSC bundles as single Max messages.

• Documentation optionnelle sur le protocole OSC :
http://www.cnmat.berkeley.edu/OpenSoundControl/

• Site des applications offrant des passerelles en OSC :
http://opensoundcontrol.org/