インタラクション

インタラクティヴ・リスニング

個々の音響群が有している広大なパラメータ空間を探索するために、マウスによるインタラクションを活用する。コンピュータのデスクトップ上のマウスの位置によってパラメータを変化させ、音の変化をインタラクティブに操作する。マウスの座標は水平垂直の2次元なので、同時に2つのパラメータを探索することができる。インタラクティブな方法を用いることで、パラメータの変化と音の聴こえ方の関係を発見的(heuristic)に探求することができる。

マウスモニター

インタラクティブ・リスニング中にマウスの座標を表示することができると便利である。マウス座標は、デスクトップの左上が(0, 0)、右下が(1, 1)である。

```
// マウスモニターのサーバー定義
(
SynthDef(\mousexToClient, { |trig_id = 0|
    var mousex, mousey, trig;
    mousex = MouseX.kr(0, 1, \linear, 0);
    mousey = MouseY.kr(0, 1, \linear, 0);
    trig = HPZ1.kr(mousex).abs;
    trig = HPZ1.kr(mousey).abs;
    SendTrig.kr(trig, 0, mousex);
    SendTrig.kr(trig, 1, mousey);
}).send(s);
)

a.free;
a = Synth(\mousexToClient);

// マウス座標をウィンドウに表示する
(
var win, mouse, x, y, text, resp;
win = SCWindow("mouse", Rect(100, 200, 350, 40)).front;
win.view.background_(Color.white);
win.view.decorator = FlowLayout(win.view.bounds);
mouse = SCStaticText(win, 350@20).font_(Font("Monaco", 12)).stringColor_(Color.black);

resp = OSCresponderNode(s.addr, '/tr', { |time, resp, msg|
    if( msg[2] == 0,{ x = msg[3] }, { y = msg[3] });
    text = "[ x =" + x.asString + ", y =" + y.asString + "]";
    { mouse.string_(text); }.defer;
}).add;
win.onClose_({resp.remove});
)
```

パラメータの値にマウス座標をマッピングする

```
// 周波数、振幅のプロキシの定義
~freq.kr(1);
~amp.kr(1);

// 点列を鳴らす
~out = { Impulse.ar(~freq.kr, 0, ~amp.kr).dup };

// マウスのX座標を周波数に線形マッピングする
~freq = { MouseX.kr(0.1, 22050.0, 0) };

// マウスのY座標を振幅に線形マッピングする
~amp = { MouseY.kr(1.0, 0.0, 0) };

// 線を鳴らす
~out = { SinOsc.ar(~freq.kr, 0, ~amp.kr).dup };

// 継続時間とトリガー用プロキシの定義
~sus.kr(1);
~trig.kr(1);

// 線の継続時間を設定する
```

```
~out = { SinOsc.ar(~freq.kr, 0, EnvGen.ar(Env.linen(0, ~sus.kr, 0, ~amp.kr, 0), ~trig.kr)).dup };

// トリガー間隔は5秒
~trig = { Impulse.kr(0.2) };

// 振幅は0.5;
~amp = 0.5;

// マウスのY座標を継続時間に線形マッピングする
~sus = { MouseY.kr(0.001, 1, 0) };

// 面の継続時間を設定する
~out = { WhiteNoise.ar(EnvGen.ar(Env.linen(0, ~sus.kr, 0, ~amp.kr, 0), ~trig.kr)).dup };

// マウスのX座標を振幅に線形マッピングする
~amp = { MouseX.kr(0.0, 1.0, 0) };
```

絶対知覚と相対知覚

人間の知覚には、絶対的な部分と相対的な部分がある。絶対的な知覚とは、絶対音感のように対象のパラメータの値そのもの（絶対音感の場合は音の高さ）が問題となる場合であり、相対的な知覚とは、2つの対象を比較対照したときのパラメータの比が問題となる場合である。

音の高さ、つまり音程そのものは周波数という絶対的なパラメータである。しかしオクターブや協和/不協和という音程関係は、周波数の比によって決まる。パラメータの比が問題になるということは、パラメータのスケールがリニアではなく、対数スケールで表現されるということでもある。人間の可聴周波数の範囲はおおよそ20[Hz]～20000[Hz]であるが、音程が周波数の差ではなく比で定義されるため、その中の目盛りは20（20^1）、40（20^2）、80（20^3）、160　（20^4）、320（20^5）…というように対数的に刻まれている。音程だけでなく、音量（音の大きさ）についても、人間の聴覚は対数スケールである。音の振幅が2倍になったからといって、聴覚上の音量が2倍になるわけではない。実際、感覚レベルの音量の単位であるデシベル（dB）は対数スケールになっている。音のエネルギー(音の強さ)が倍になるとデシベル値が約3[dB]、4倍になると約6[dB]増加し、半分になると約3[dB]低下する。

デシベルから振幅への変換

```
// 0[dB]が最大振幅(1.0)に相当する
0.dbamp;

// 3[dB]低下すると振幅はおおよそ√0.5倍(音の強さは振幅の2乗に比例する)になる
-3.dbamp;
-3.dbamp.squared;

//  6[dB]低下すると振幅はおよそ半分(√0.25倍)になる。
-6.dbamp;
-6.dbamp.squared;

//  10[dB]低下すると振幅はちょうど√0.1倍になる(音の強さが10倍になると10[dB]増加する)。
-10.dbamp;
-10.dbamp.squared;


// 逆に振幅が半分になるとどのくらいdbが低下するか。
0.5.ampdb;

// 20[dB]低下すると振幅は10分の1になる。
-20.dbamp;
```

スケーリング

音響素材のパラメータをマウス座標のような(視覚的)空間にマッピングする場合、そのスケーリングに配慮する必要がある。パラメータとマウス座標を対数マッピングした例を示す。

```
// 点列を鳴らす
~out = { Impulse.ar(~freq.kr, 0, ~amp.kr).dup };

// マウスのX座標を周波数に対数マッピングする
~freq = { MouseX.kr(0.1, 22050.0, 1) };

// マウスのY座標を振幅に対数マッピングする
~amp = { MouseY.kr(1.0, 0.01, 1) };

// 線を鳴らす
~out = { SinOsc.ar(~freq.kr, 0, ~amp.kr).dup };
```

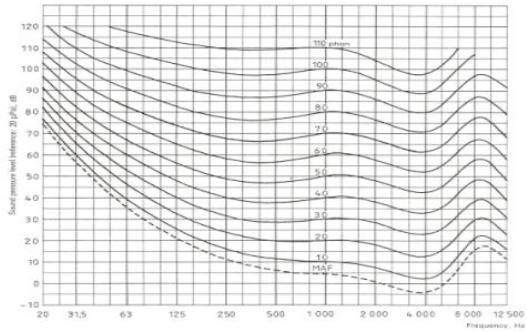
```
// 周波数のマッピングの範囲を変更する
~freq = { MouseX.kr(20.0, 22050.0, 1) };

// 線の継続時間を設定する
~out = { SinOsc.ar(~freq.kr, 0, EnvGen.ar(Env.linen(0, ~sus.kr, 0, ~amp.kr, 0), ~trig.kr)).dup };

// マウスのY座標を継続時間に対数マッピングする
~sus = { MouseY.kr(0.001, 1, 1) };

// 面の継続時間を設定する
~out = { WhiteNoise.ar(EnvGen.ar(Env.linen(0, ~sus.kr, 0, ~amp.kr, 0), ~trig.kr)).dup };

// マウスのX座標を振幅に対数マッピングする
~amp = { MouseX.kr(0.01, 1.0, 1) };
```

周波数とマウスの位置が線形マッピングされている場合、全体として高音域を中心としたパラメータの探索がやりやすい。それに対して、対数マッピングの場合は中低音域を中心とした探索がやりやすい。

マウスのX座標を周波数に、Y座標を振幅にマッピングした場合、マウスを水平に動かした(振幅が一定である)のにもかかわらず、音量が変化しているように聴こえる。これは、「リスニング」の項で述べたように、周波数によって耳の感度が異なるためである。一般的に人間の耳は3〜4000Hzのあたりで感度が非常に良く、それより高音および低音域では感度が悪くなる。特に音が弱くなる(小さくなる)と、高音に比べ低音の感度が極端に悪くなる。

(注)人間の外耳道の基本振動数がおおよそ3400Hzである。この値は耳の感度の良い周波数域に一致する。

参考：等ラウドネスレベル曲線(ISO226)



様々な周波数の音が感覚的に同じ大きさ（ラウドネス）に聞こえる音圧レベルを結んで作られる周波数特性を等ラウドネスレベル曲線という。これは聴覚の基本的な周波数感度特性を示す等感曲線であり、聴覚における最も基礎的な特性の一つである。この特性はISO226として国際規格化されている。最新の研究成果に基づいて、2003年に全面改訂された。

```
// 振幅を一定のまま、線の周波数を20[Hz]〜22050[Hz]まで対数スイープする。音量の変化をどのように感じるか?
~out = { SinOsc.ar(~freq.kr, 0, 0.5).dup };
~freq = { XLine.kr(20.0, 22050.0, 60, doneAction:2) };
```

時間とピッチ感覚

点はどこから線になるか?

継続時間が非常に短い線(1〜10波長分)を聴く。線が線としてピッチが認識されるのに、何波長分の継続時間を必要とするか? 周波数によって、必要とされる波長は変化するか?

```
// 線の継続時間を変化させる
~out = { SinOsc.ar(~freq.kr, 0, EnvGen.ar(Env.linen(0, ~sus.kr, 0, ~amp.kr, 0), ~trig.kr)).dup };

// トリガー間隔を1秒にする
~trig = { Impulse.kr(1) };
```

```
// マウスのX座標を周波数に対数マッピングする
~freq = { MouseX.kr(20.0, 22050.0, 1) };

// 振幅は0.5;
~amp = 0.5;

// マウスのY座標を継続時間(1〜10波長)に線形マッピングする
~sus = { MouseY.kr(1, 10, 0) / ~freq.kr };
```

継続時間を絶対時間にマッピングする。高音域のピッチの認識はどのように変化するか?

```
// マウスのY軸を継続時間(0〜0.05秒)に線形マッピングする
~sus = { MouseY.kr(0, 0.05, 0) };
```

音の長さとピッチ感覚は密接に関連している。一般に1000[Hz]以上の音の場合、音の長さとピッチ感覚の関係はおおよそ以下の通りである。

・約5[ms](1000[Hz]の音の場合5波長)以下：クリック音(ピッチが知覚できない)
・約5〜10[ms]：クリックピッチ(高低の判別は可能)
・約10[ms]以上：トーンピッチ(ピッチが知覚可能)

```
// 継続時間が1[ms]から20[ms]に線形スイープしていく4410[Hz]の短い線(しだいにピッチ感覚が明瞭になってくる)
~freq = 4410;
~sus = { Line.kr(0.001, 0.02, 60, doneAction:2) };

// 継続時間が1[ms]から20[ms]に線形スイープしていく8820[Hz]の短い線(しだいにピッチ感覚が明瞭になってくる)
~freq = 8820;
~sus = { Line.kr(0.001, 0.02, 60, doneAction:2) };
```

うなりを聴く

接近した2つの周波数の線を同時に鳴らすと、それぞれの周波数の差に相当する周期で音の強弱を聴くことができる。これが「うなり」である。うなりの周波数が低いと、2つの音はひとつの音であるかのように聴こえるが、周波数がある程度まで離れると別の2音として聴こえる。

```
// うなり用のプロキシを定義
~beat.kr(1);

// 2本の線を鳴らす
~out = { ( SinOsc.ar(~freq.kr + ~beat.kr, 0, 0.25) + SinOsc.ar(~freq.kr - ~beat.kr, 0, 0.25)).dup };

// マウスのX座標を基本周波数に対数マッピング
~freq = { MouseX.kr(20.0, 11025.0, 1) };

// マウスのY座標をうなりの周波数に線形マッピング
~beat = { MouseY.kr(0, 20, 0) };

// 2つの音を右耳と左耳に振り分ける。スピーカとヘッドホンでどのように聴こえ方が異なるか?
~out = { ( SinOsc.ar([~freq.kr + ~beat.kr, ~freq.kr - ~beat.kr], 0, 0.5)) };


// うなりの周波数を基本周波数に対する比で与える
~out = { ( SinOsc.ar(~freq.kr / ~beat.kr, 0, 0.25) + SinOsc.ar(~freq.kr * ~beat.kr, 0, 0.25)).dup };

// マウスのY座標をうなりの周波数比に対数マッピング
~beat = { MouseY.kr(1, 1.1, 1) };
```

(注)人間の音高の識別能力は約4%といわれている。

次に同じことを点列で行なう。どのように聴こえ方が変化するか?

```
// 2本の点列を鳴らす
~out = { ( Impulse.ar(~freq.kr + ~beat.kr, 0, 0.5) + Impulse.ar(~freq.kr - ~beat.kr, 0, 0.5)).dup };

// マウスのX座標を基本周波数に対数マッピング
~freq = { MouseX.kr(1.0, 11025.0, 1) };

// マウスのY座標をうなりの周波数に線形マッピング
~beat = { MouseY.kr(0, 20, 0) };

// 2つの音を右耳と左耳に振り分ける。
```

```
~out = { Impulse.ar([~freq.kr + ~beat.kr, ~freq.kr - ~beat.kr], 0, 0.8) };


// うなりの周波数を基本周波数に対する比で与える
~out = { ( Impulse.ar(~freq.kr / ~beat.kr, 0, 0.5) + Impulse.ar(~freq.kr * ~beat.kr, 0, 0.5)).dup };

// マウスのY座標をうなりの周波数比に対数マッピング
~beat = { MouseY.kr(1, 1.1, 1) };
```

マスキング

ある音に対する最小の可聴値が、他の音によって上昇する(聴こえ難くなる)現象をマスキングという。先程のうなりの例において、2つの音の音量バランスを変化させることで、このマスキングの効果を確認する。

```
// 2本の線を鳴らす(基音の周波数は441[Hz])
~out = { ( SinOsc.ar(441 + ~beat.kr, 0, ~amp.kr) + SinOsc.ar(441 - ~beat.kr, 0, 1 - ~amp.kr)).dup };

// マウスのX座標を2本の線のバランスに線形マッピング
~amp = { MouseX.kr(1, 0, 0) };

// マウスのY座標をうなりの周波数に線形マッピング
~beat = { MouseY.kr(0, 20, 0) };


// 2つの音を右耳と左耳に振り分ける(両耳マスキング)
~out = { SinOsc.ar([441 + ~beat.kr, 441 - ~beat.kr], 0, [~amp.kr, 1 - ~amp.kr]) };
```

一般に、マスキング量は基音(妨害音)に近づくにしたがって大きくなるが、近くなりすぎるとうなりによって逆にマスキング量が小さくなる。

▼エクササイズ：音のサーベイ(パラメータの発見的探索)

パラメトリック・リスニングのエクササイズで示したパラメータの組から任意の2組を選択し、その間をインタラクティブに補間することで、どのように音が変化するかを探索せよ。スケーリングにも留意すること。

その後、以下のような方法で音響パラメータを探求せよ。

・最初にインタラクティヴ・リスニングで、グローバルにパラメータを探索する。
・耳を研ぎ澄まして、フォーカスをあてるポイントを発見する。
・その周囲のパラメータ領域をマウス座標をリマップし、より詳細に音を探求する。
・以上の手順を繰り返し詳細化していくことで、最終的な音響パラメータの組を決定する。
・再び最初の手順に戻り、新たな領域(パラメータの組)で探索を開始する。

```
// トリガー間隔は5秒
~trig = { Impulse.kr(0.2) };

// パン用のプロキシを定義
~pan.kr(1);
```

点列

```
// 5秒間隔でくりかえし鳴る点列
(
~out = {Pan2.ar(
            Impulse.ar(~freq.kr, 0, EnvGen.ar(Env.linen(0, ~sus.kr, 0, ~amp.kr, 0), ~trig.kr)),
            ~pan.kr)};
)

// 振幅=中、周波数=中、パン=中央、持続時間=中、に設定する
~amp = 0.5;
~freq = 21;
~pan = 0;
~sus = 3;

// マウスのX座標を周波数に対数マッピング(周波数=低～高)
~freq = { MouseX.kr(1.0, 105, 1) };

// マウスのY座標を振幅に対数マッピング(振幅=中～小)
~amp = { MouseY.kr(0.5, 0.1, 1) };
```

```
// マウスのX座標をパンに線形マッピング(パン=左～右)
~freq = 21;
~pan = { MouseX.kr(-1.0, 1.0, 0) };
```

線

```
// 5秒間隔でくりかえし鳴る線
(
~out = {Pan2.ar(
            SinOsc.ar(~freq.kr, 0, EnvGen.ar(Env.linen(0, ~sus.kr, 0, ~amp.kr, 0), ~trig.kr)),
            ~pan.kr)};
)

// 振幅=中、周波数=中、パン=中央、持続時間=中、に設定する
~amp = 0.5;
~freq = 441;
~pan = 0;
~sus = 3;

// マウスのX座標を周波数に対数マッピング(周波数=低～高)
~freq = { MouseX.kr(4410.0, 11025.0, 1) };

// マウスのY座標を振幅に対数マッピング(振幅=中～極小)
~amp = { MouseY.kr(0.5, 0.01, 1) };

// マウスのX座標をパンに線形マッピング(パン=左～右)
~freq = 441;
~pan = { MouseX.kr(-1.0, 1.0, 0) };
```

面

```
// 5秒間隔でくりかえし鳴る面
(
~out = {Pan2.ar(
            WhiteNoise.ar(EnvGen.ar(Env.linen(0, ~sus.kr, 0, ~amp.kr, 0), ~trig.kr)),
            ~pan.kr)};
)

// 振幅=中、パン=中央、持続時間=中、に設定する
~amp = 0.5;
~pan = 0;
~sus = 3;

// マウスのX座標をパンに線形マッピング(パン=左～右)
~pan = { MouseX.kr(-1.0, 1.0, 0) };

// マウスのY座標を振幅に対数マッピング(振幅=中～極小)
~amp = { MouseY.kr(0.5, 0.01, 1) };

// マウスのY座標を継続時間に対数マッピング(パン=中～極々短)
~amp = 0.5;
~sus = { MouseY.kr(3.0, 0.003, 1) };
```