43PIR3

$4/3PIR^3$ indica il Volume di una Semisfera. Il lavoro svolto è atto al creare una interfaccia di controllo per quanto riguarda la *cupola* dell'*Aula Bianchini* del *Conservatorio S. Cecilia* di Roma, facilmente espandibile ad altri sistemi (E.g. *Ambisonic*), attraverso la reinterpretazione di alcuni messagi.

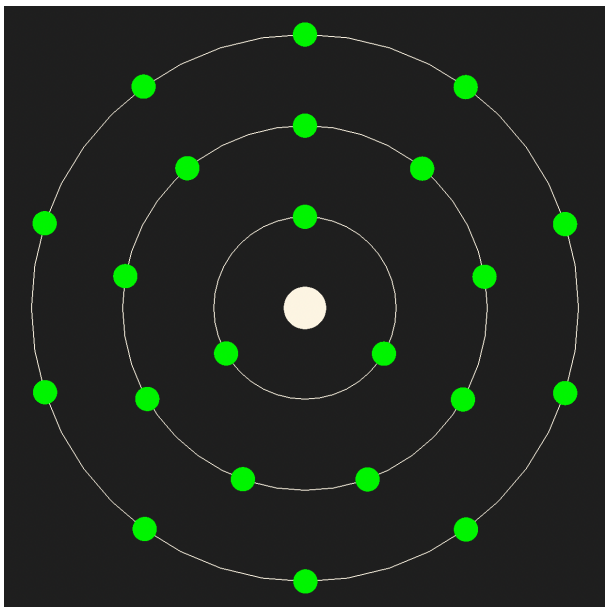
Interfaccia

Per l'interfaccia è stato utilizzata la libreria di ***P5Js***, una libreria *Javascript* per il *"creative coding"*, realizzazione di lavori grafici e algoritmi interattivi *web-based*.

La realizzazione consiste nella creazione dei 3 anelli di altoparlanti:

- 1^o, anello sul piano orizzontale, di 10 altoparlanti;
- 2^o, anello di mezzo tra l'orizziontale e il verticale, 9 altoparlanti;
- 3^o, anello sul piano verticale, 3 altoparlanti.

E sono così disposti:



> il grafico è una rappresentazione planare della semisfera dall'alto.
>
> I *cerchi verdi* sono i **led** atti a monitorare lo stato del guadagno audio, dove il bianco $x = 0$ e il rosso se $x > 0.97$.

Attraverso la libreria di *P5Js*, attraverso il puntamento *multi-touch*, con `touches[0]`, ovvero il primo valore della lista contenente gli input *touch* in ordine di apparizione. Conseguentemente all'interazione, la funzione `calc()` calcola la posizione polare del

puntamento (in *gradi* e *distanza goniometrica* dal cerchio verticale) e li invia tramite OSC ad una patch di *Max MSP* che controlla gli altoparlanti.

```
function calc() {

// calcola posizione polare quando percepisce 1 input

        if (touches.length > 0) {
        mx = clip(touches[0].x, W*siz, W-W*siz);
        my = clip(touches[0].y, H*siz, H-H*siz);

        let mxx = map(mx, W-W*siz, W*siz, -1, 1);
        let myy = map(my, H-H*siz, H*siz, -1, 1);

// calcolo della posizione in radianti e gradi

        rad = atan2(myy, mxx);
        deg = 360-((rad/PI-1.5)%2)*-180;

// relalizzazione di incremento e decremento infinito

        if (ddeg < -180) {cc+=1;}
        if (ddeg > 180) {cc-=1;}
        deg_old = deg;

// calcolo del valore di azimuth

        val = clip(sqrt(mxx*mxx + myy*myy), 1, 1/3);

// calcolo per la rappresentazione grafica

        let vx = (val)*cos(rad);
        let vy = (val)*sin(rad);

        sxx = map(vx*siz, -1, 1, W-W*siz, W*siz);
        syy = map(vy*siz, -1, 1, W-W*siz, W*siz);
        sx = map(vx, -0.9, 0.9, 0, W);
        sy = map(vy, -0.9, 0.9, 0, W);

// invio dei messaggi tramite OSC

        sendMSG('/distance', 144*((1-val)/(1/1.5)));
        sendMSG('/deg', deg+360*cc);
```

Infine attraverso una seconda interazione contemporanea a quella di movimento della sorgente, è possibile variare l'ammontare della *separazione orizzontale* o *verticale* degli altoparlanti (cfr. *spanner*).

```
// reset dei flag e storage degli ultimi valori per aumento relativo

        if (touches.length < 2) {flag = 0; old_sep.x = sep.x; old_sep.y =
sep.y;}

// quando viene percepito un secondo input

        if (touches.length > 1) {

        // calcolo dei valori relativi (delta)

                if (flag==0) {
                        dsep.x = clip(touches[1].x, W*siz, W-W*siz)-
old_sep.x;
                        dsep.y = clip(touches[1].y, H*siz, H-H*siz)-
old_sep.y;
                        flag = 1;
                }
                let dxx = clip(touches[1].x, W*siz, W-W*siz);
                let dyy = touches[1].y;

        // calcolo separazione orizzontale (sep.x) e verticale (sep.y)

                sep.x = clip((dxx-dsep.x)/dsep.x + old_sep.x, 1, 0.125);
                sep.y = clip((dyy-dsep.y)/dsep.y + old_sep.y, 1, 0.125);

        // invio dei messaggi tramite OSC

                sendMSG('/hsep', pow(1-sep.x, 3.5)*100);
                sendMSG('/vsep', pow(1-sep.y, 3.5)*100*3*0.1);
                }
        }
}
```

Il resto dell'interfaccia consiste in un raggio che indica la posizione di puntamento `(sxx, syy)`, un sistema di feedback dei *led*, un *arco* ed un ulteriore segmento che indicano il valore di *separazione*.

```
function draw() {

        background(12);
        calc();
        stroke(255*0.99, 255*0.95, 225*0.9);
        strokeWeight(2);
        noFill();

// i 3 livelli circolari
```

```
        circle(W/2, H/2, H*pow(siz,3.15));
        circle(W/2, H/2, H*pow(siz,3.15)*(2/3));
        circle(W/2, H/2, H*pow(siz,3.15)/3);
        strokeWeight(W*0.005);

// il raggio

        line(W/2, H/2, sxx, syy);
        stroke(50, 120, 255);

// la separazione verticale

        line(sxx, syy, sxx+(W/2-sxx)*sep.y, syy+(H/2-syy)*sep.y);
        var sx2 = map(sep.y*(1-val)*4, 0, 1, sxx, sx);
        var sy2 = map(sep.y*(1-val)*4, 0, 1, syy, sy);
        line(sxx, syy, sx2, sy2);

// la separazione orizzontale

        arc(W/2, H/2, pow(siz,3.15)*val*W, pow(siz,3.15)*val*H, rad-
(sep.x*PI), rad+(sep.x*PI))
        fill(50, 120,255);
        noStroke();

// cerchio centrale e cerchio alla fine del raggio

        circle(W/2, H/2, W*0.03)
        circle(sxx, syy, W*0.03);

// for loop per disegnare gli altoparlanti sui diversi anelli

        for (sp=3; sp>0; sp--) {
                var colo = [];
                if (sp==3) {
                        spk=10;
                        colo = leds1;
                }
                if (sp==2) {
                        spk=9;
                        colo = leds2;
                }
                if (sp==1) {
                        spk=3;
                        colo = leds3;
                }
                for (i=0; i<spk; i++) {
                        var r = 255*(0.996-pow(colo[i], 0.5)*0.996);
                        var g = 255*0.964;
                        var b = 255*(0.85-pow(colo[i], 0.5)*0.85);
                        if (colo[i] > 0.97) fill(255,0,0);
                        else fill(r, g, b);
```

```
                        let px = map(sin(i/spk*3.14*2)*(siz*sp/3), -1, 1,
W*siz, W-W*siz,);
                        let py = map(cos((spk-i)/spk*3.14*2)*(siz*sp/3),
-1, 1, H*siz, H-H*siz);
                        circle(px, py, W*0.025);
                }
        }
}
```

WebSocket

Per permettere la comunicazione da server accessibile da network locale (quindi anche da dispositivo diverso dal computer con l'istanza di *Max MSP*), è stato utilizzato *NodeJs* e due sue librerie, *osc.js* e *ws*, per creare il collegamento su indirzzo IP definito `192.168.1.121` su porta `:9000`, ed infine una funzione per inviare i messaggi di stato *led*.

```
const OSC = require('osc-js')
const maxApi = require('max-api');

// avvia un web socket su ws://192.168.1.121:9000
const config = { udpClient: { host:"192.168.1.121", port: 9000 }, wsServer:
{host:"192.168.1.121", port: 9000} }
const osc = new OSC({ plugin: new OSC.BridgePlugin(config) })
osc.open()

// riceve in input messaggi 'send'
maxApi.addHandler(
        "send", (...args)=>{
                var message = new OSC.Message(args[0], args[1]);
                osc.send(message);
        }
);
```

Infine l'html, apre l'istanza su stesso *host*, inizializza le variabili e riceve i valori da *MaxMSP* suddivisi nei tre anelli di altoparlanti. Viene poi richiamato `sketch.js` che contiene la parte grafica e di calcolo, e un file `css` che disabilita le funzioni di *selezione*, *pinch zoom* e *scroll*.

```
<html>
<head>
<script src = "https://cdn.jsdelivr.net/npm/p5@1.4.1/lib/p5.js"></script>
<script src = "https://cdn.jsdelivr.net/npm/osc-js@2.2.0/lib/osc.min.js">
</script>
<script type = "text/javascript">
        var osc = new OSC();
        osc.open({host:"192.168.1.121", port:9000});
```

```
        var leds1 = [0,0,0,0,0,0,0,0,0,0];
        var leds2 = [0,0,0,0,0,0,0,0,0];
        var leds3 = [0,0,0];

// riceve i valori monitoring dei led e decritta i valori e indice

        osc.on('/leds1', message => {
                var id1 = (message.args-(message.args)%10);
                var id = (9-id1*0.1+1)%10;
                leds1[id] = message.args-id1;
        })

        osc.on('/leds2', message => {
                var id2 = (message.args-(message.args)%10);
                var id = (8-id2*0.1+1)%9;
                leds2[id] = message.args-id2;
        })



        osc.on('/leds3', message => {
                var id3 = (message.args-(message.args)%10);
                var id = (2-id3*0.1+1)%3;
                leds3[id] = message.args-id3;
        })
</script>
<script src="sketch.js"></script>
<link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
```

Max MSP

Infine sul software di controllo audio vengono ricevuti e inviati i messaggi e formattati da due routine di *javascript*, una che prepara le liste per il monitoring dei *led* da inviare e una che prepara i messaggi ricevuti per i vari oggetti che controllano la posizione della sorgente (*spanner10, spanner 9, spanner3 e spannervert*).

La prima:

```
autowatch = 1;
outlets = 1;
var lst1 = [];
var lst2 = [];
var lst3 = [];

// inizializzazione per interpolazione delle variabili
```

```
for (var i = 0; i < 10; i++) {
        lst1[i] = 0;
        lst2[i] = 0;
        lst3[i] = 0;
}

// loops per invio dei messaggi, l'indice del led viene indicato dalle
decine

function leds1(v) {
        var lst = arrayfromargs(arguments);
        for (i = 0; i < lst.length; i++) {
                lst[i] = (lst1[i]+lst[i])*0.5;
                outlet(0, "/leds1",i*10+lst[i]);
                lst1[i] = lst[i];
        }
}

function leds2(v) {
        var lst = arrayfromargs(arguments);
        for (i = 0; i < lst.length; i++) {
                lst[i] = (lst2[i]+lst[i])*0.5;
                outlet(0, "/leds2",i*10+lst[i]);
                lst2[i] = lst[i];
        }
}

function leds3(v) {
        var lst = arrayfromargs(arguments);
        for (i = 0; i < lst.length; i++) {
                lst[i] = (lst3[i]+lst[i])*0.5;
                outlet(0, "/leds3",i*10+lst[i]);
                lst3[i] = lst[i];
        }
}
```

> L'indice del led corrispettivo è indicato dalle decine del valore inviato, il formato della lista sarà simile a:
> `/leds1 10.5`
> dove le decine indicano il secondo *led* e le unita+decimali indicano il valore da rappresentare.

La seconda:

```
autowatch = 1;
outlets = 4;
```

```
function wsMsg() {
        var m = arrayfromargs(arguments);
        if (m[0] == "/distance") {outlet(0, "/spanner_v/Angle", m[1]);}
        if (m[0] == "/deg") {
                outlet(1, "/spanner10/Angle", m[1]);
                outlet(2, "/spanner9/Angle", m[1]);
                outlet(3, "/spanner3/Angle", m[1]);
        }

        if (m[0] == "/hsep") {
                outlet(1, "/spanner10/Separation", m[1]);
                outlet(2, "/spanner9/Separation", m[1]*9*0.1);
                outlet(3, "/spanner3/Separation", m[1]*3*0.1);
        }

        if (m[0] == "/vsep") {outlet(0, "/spanner_v/Separation", m[1]);}
```