Leal, Andreia Neto Cardoso

Som e cor: estudo teórico e implementação prática : “Play Music Through Color”

Metadados

Data de Publicação 2020

Resumo Este relatório visa descrever e apresentar a pesquisa e trabalho prático realizados no âmbito das disciplinas de Projeto Individual I e Projeto Individual II da Licenciatura em Música – Variante de Música Electrónica e Produção Musical da Escola Superior de Artes Aplicadas do Instituto Politécnico de Castelo Branco. A pesquisa trata da relação entre o som e a cor aplicados numa pequena aplicação demonstrativa. O trabalho prático consiste numa aplicação interativa que permite ao utilizad...

Editor IPCB. ESART

Tipo report

Revisão de Pares Não

Coleções ESART - Música - Variante de Música Eletrónica e Produção Musical

Instituto Politécnico de Castelo Branco
Escola Superior de Artes Aplicadas

Licenciatura em Música – Variante de Música Electrónica e Produção Musical

Som e Cor - Estudo Teórico e Implementação Prática

"Play Music Through Color"

Andreia Neto Cardoso Leal

Orientador
Prof. Rui Dias

Junho de 2014

Índice

AGRADECIMENTOS **4**
RESUMO **5**
ABSTRACT **5**
INTRODUÇÃO **6**
OBJECTIVOS **6**
CAPÍTULO 1 – O SOM E A COR **7**
CAPÍTULO 2 – SINESTESIA **11**
2.1 – SINESTESIA COMO CONDIÇÃO NEUROLÓGICA 11
2.2 – SINESTESIA COMO CONDIÇÃO ESTÉTICA 12
CAPÍTULO 3 – PROPOSTAS DE RELAÇÃO COR/SOM - ESCALAS COLORIDAS **13**
3.1 – ISAAC NEWTON 13
3.2 – LOUIS BERTRAND CASTEL 14
3.3 – GEORGE FIELD 15
3.4 – D. D. JAMESON 15
3.5 – THEODOR SEEMANN 17
3.6 – ALEXANDER WALLACE RIMINGTON 17
3.7 – BAINBRIDGE BISHOP 18
3.8 – H. VON HELMOLTZ 19
3.9 – ALEXANDER SCRIABIN 20
3.10 – RIMSKY-KORSAKOV 21
3.11 – IRA JEAN BELMONT 21
3.12 – OLIVER MESSIAEN 22
3.13 – STEVE ZIEVERINK 23
CAPÍTULO 4 – FASE DE DEMONSTRAÇÃO PRÁTICA DO ESTUDO **25**
4.1 – APLICAÇÃO DEMONSTRATIVA 25
4.2 – ELABORAÇÃO 25
CAPÍTULO 5 – FASE DE DEFINIÇÃO DO PROJETO **29**
5.1 – INSPIRAÇÃO 29
CAPÍTULO 6 – INTERATIVIDADE **33**
6.1 – CONCEITO 33
6.1.1 – NÍVEIS DE INTERAÇÃO 33
6.2 – SISTEMAS INTERATIVOS 34
6.2.1 – SISTEMAS INTERATIVOS EM MÚSICA 35
6.3 – MODELOS 35
CAPÍTULO 7 – ESTADO DE ARTE **36**
7.1 – COLORCHORD SOUND LIGHTNING 36
7.2 – PERFECT PITCH 37
7.3 – REACTABLE 38
7.4 – BUCEPHALUS 39
CAPÍTULO 8 – FASE DE CONCEPÇÃO **40**

8.1 – APLICAÇÃO INTERATIVA 40
8.2 – ELABORAÇÃO 40
8.2.1 - INTERFACE 40
8.2.1.1 - PROTÓTIPO 40
8.2.1.2 - DEFINIÇÃO DO OBJECTO 41
8.2.1.3 – PROGRAMAÇÃO EM PROCESSING 42
8.2.2 - COMUNICAÇÃO OSC 50
8.2.3 - PROGRAMAÇÃO EM MAXMSP 53
8.2.4 - FM8 61
8.3 - PROBLEMAS 62
CONCLUSÃO 62
REFERÊNCIAS 64
BIBLIOGRAFIA 67

Agradecimentos

A vida, o trabalho, as ações são influenciadas pelas pessoas que nos rodeiam. Há pessoas que nos marcam, que nos definem, que nos movem, que nos dão força, que acreditam em nós, que nos fazem querer ser melhores de dia para dia. Essas são aquelas a quem eu agradeço do fundo do coração.

Em primeiro lugar gostaria de agradecer à minha família. À minha mãe pelo apoio incondicional, por me fazer acreditar e por me relembrar sempre que o que eu quero é o que devo fazer. Ao meu irmão por me motivar e acreditar em mim. Ao meu pai por me incentivar a trabalhar e por me lembrar sempre que o "amor de família" é o melhor amor que um ser humano pode ter.

Em segundo lugar, um obrigada especial aos Meninos do Castelo, Carlos Ferreira, Joana Teixeira, Kevin Pardal, Simão Pereira e Diana Figueiredo, por serem aqueles amigos que me fizeram ficar e acreditar que tudo isto era possível.

Não poderia deixar de agradecer ao João Neves e ao João Borges, meus companheiros, pelos amigos incansáveis que são e pelas discussões que tanto me ajudaram na elaboração deste projeto. Ao António João Couto por ser o meu "irmão mais velho" e por me puxar sempre para cima.

À Inês Piedade, por ser a minha *beta-tester* e por me aturar as frustrações intermináveis.

Ao meu Orientador de projeto, prof, Rui Dias, por me fazer ter iniciativa e me "obrigar" a pensar por mim.

Ao Nuno João Casteleira, o meu maior obrigada, por todos os brainstorming, pelos ensinamentos, pelas explicações, pela ajuda, pelas horas que gastou a aturar os "porque é que isto não funciona?". Sem ele seria muito mais complicado realizar este projeto.

A lista poderia estender-se por aqui fora mas vou ficar apenas por aqueles que não foram referidos em cima mas que estiveram comigo e me ajudaram, de qualquer forma, na conclusão desta etapa:

Sérgio Lavrador | Carolina Gonçalves | Marlene Bento | Cristiano Lopes | Cláudia Silva | Natacha Pereira | Patrícia Laureano | Filipe Pereira | Luís Dias | Cátia Vasconcelos | Tiago Cruz | Hélder Silveira | Carlos Sá | Luís Marques | Ângela Santos | Maria Augusta | Inês Duarte

Resumo

Este relatório visa descrever e apresentar a pesquisa e trabalho prático realizados no âmbito das disciplinas de Projeto Individual I e Projeto Individual II da Licenciatura em Música – Variante de Música Electrónica e Produção Musical da Escola Superior de Artes Aplicadas do Instituto Politécnico de Castelo Branco.

A pesquisa trata da relação entre o som e a cor aplicados numa pequena aplicação demonstrativa. O trabalho prático consiste numa aplicação interativa que permite ao utilizador tocar música através de um interface gráfico intuitivo.

Abstract

This report aims to describe and present the research and practical work carried out in the context of the disciplines of Individual Project I and Individual Project II of Bachelor of Music - variant of Electronic Music and Musical Production of Escola Superior de Artes Aplicadas of Instituto Politécnico de Castelo Branco.

The research is about the relation between sound and color applied in a small demo application. The practical work is an interactive application that allows the user to play music through an intuitive graphic interface.

Introdução

Este projeto tem como base uma pesquisa aprofundada acerca da relação entre o som e a cor, e visa a elaboração de uma aplicação interativa onde o utilizador pode tocar música através de um interface composto por círculos e quadrados coloridos que representam as notas e os efeitos sonoros, respectivamente.

Desde cedo começou a haver especulações acerca da ligação da música com a cor. Com o avançar dos anos muitas propostas foram apresentadas, muitas teorias fundamentadas, muitas teorias falhadas. Mas há um leque de propostas que estão fundamentadas e foram utilizadas em trabalhos e obras de personalidades importantes.

Na sequência do desenvolvimento do estudo sobre a relação som-cor, decidiu.se criar uma aplicação interativa em que fossem aplicados os conhecimentos adquiridos.

Objectivos

O projeto passou por várias fases de evolução, algumas delas no primeiro semestre (Fases 1, 2, 3 e 4), outras no segundo semestre (Fases 5, 6 e 7). Cada uma dessas fases tinha como objectivo o alcance de uma etapa, sendo estas:

Fase 1 - Adquirir objecto de estudo (pesquisa).

Fase 2 – Análise e estudo da pesquisa.

Fase 3 – Elaboração do relatório relativo ao estudo feito.

Fase 4 – Aplicação dos conhecimentos adquiridos numa aplicação demonstrativa.

Fase 5 – Definição do Conceito da componente prática

Fase 6 – Recolha, Análise e Diagnóstico.

Fase 7 – Concepção da Aplicação Interativa.

Capítulo 1 – O Som e a Cor

Desde cedo se começaram a desenvolver teorias acerca da existência de uma ligação entre o som e a cor.

Rapidamente se confirmou a existência da mesma mas, apesar de todos os estudos realizados ao longo dos anos, as opiniões tendem a divergir no que toca à natureza da relação entre estes dois elementos.

O objectivo será encontrar uma forma mais correta cientificamente da conversão cor-som.

Alguns musicoterapeutas defendem que a primeira nota da escala, sendo Dó, equivale à primeira nota do arco-íris, vermelho, e assim sucessivamente, ou seja:

Dó = vermelho;

Ré = laranja;

Mi = amarelo;

Fá = verde;

Sol = azul claro;

Lá = azul escuro (anil);

Si = violeta.

(Edu Silva in

http://audiolist.quasar.eng.br/forum/kb2cd0.php?mode=article&k=53)

Esta teoria faz sentido por ser uma ligação com lógica, pois atribui as cores do arco íris às notas, numa sequência ascendente de ambas, mas é facilmente corrompida quando confrontada com o científico.

Por outro lado, alguns esotéricos defendem que a ligação entre o som e a cor é distribuída da seguinte forma:

Dó = vermelho ou rosa prateado;

Ré = vermelho ou laranja;

Mi = amarelo creme;

Fá = verde ou preto;

Sol = azul ou índigo com púrpura;

Lá = índigo ou amarelo;

Si = branco ou violeta.

(Edu Silva http://audiolist.quasar.eng.br/forum/kb2cd0.php?mode=article&k=53)

Segundos eles, estas são as cores primordiais de Deus, o que corrobora automaticamente a defesa desta teoria perante a comunidade científica.

Outra forma de abordar esta relação foi fazendo a multiplicação da frequência do som, por exemplo, Lá 440Hz por 10^{12} para encontrar a frequência da cor.

Ou seja,

Sol - 392 x 10^{12} seria infravermelho,

Lá - 440 x 10^{12} seria vermelho,

Sol - 784 10^{12} seria ultravioleta.

(Edu Silva http://audiolist.quasar.eng.br/forum/kb2cd0.php?mode=article&k=53)

Esta teoria peca por razões estritamente musicais. Quando queremos transpor uma nota não o fazemos desta maneira mas sim multiplicando a frequência por 2 sucessivamente, subindo nas oitavas. Se adotássemos esta teoria então as notas de 44Hz, 440Hz, 4400Hz seriam todas Lá, o que não é verdade, ou seja, retira todo o carácter científico à teoria.

No meio de tanta teoria sem aprovação científica, pensou-se então usar um elemento em comum na composição da cor e do som para obter uma relação coesa. Assim sendo, a maioria defende que a cor está diretamente ligada ao som através da frequência.

A luz, apesar de normalmente ser calculada com comprimento de onda, também possuí frequência. As frequências da luz visível, enquadram-se exatamente entre as frequências da luz infravermelha e a frequência dos raios ultravioleta.

Se multiplicarmos a frequência de uma nota por 2, iremos obter a frequência da mesma nota uma oitava acima, ou seja, a cada oitava temos o dobra da frequência.

Para conseguirmos relacionar as frequências audíveis com as frequências da cor precisamos de dobrar continuamente a frequência da nota até chegarmos à gama das frequências da luz visível.

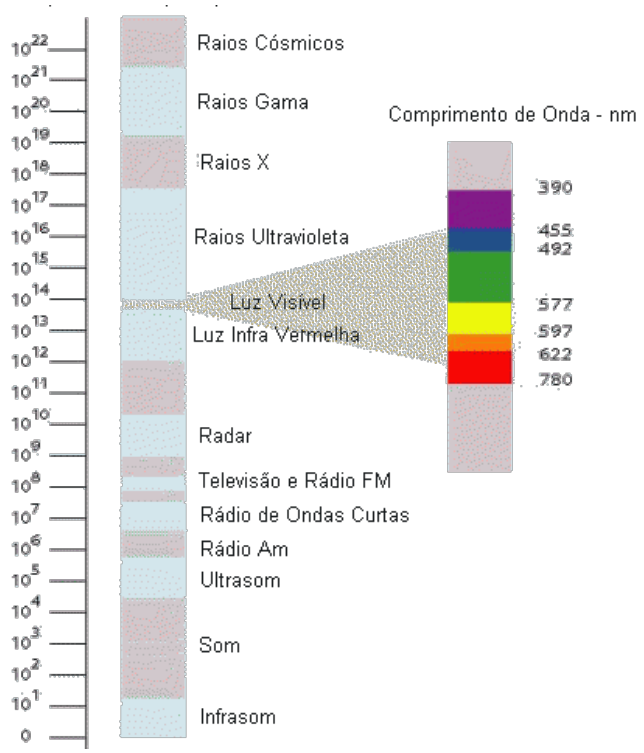


Imagem 1 - Espectro de frequências. A partir da imagem conseguimos perceber onde se situa a gama de frequências da luz visível, quais são os seus valores e o quanto se distanciam da gama de frequências audíveis.

A radiação visível é delimitada aproximadamente por $384x10^{12}$ (vermelho) e $769x10^{12}$(violeta). As fontes de informação divergem ligeiramente nos valores, isto deve-se aos métodos de medição e também às próprias características do olho humano.

Cor	Comprimento de onda (nm)	Frequência (10^{12} Hz)
vermelho	780 - 622	384 - 482
laranja	622 - 597	482 - 503
amarelo	597 - 577	503 - 520
verde	577 - 492	520 - 610
azul	492 - 455	610 - 659
violeta	455 - 390	659 - 769

Imagem 2 - frequências e comprimentos de onda para as cores, no vazio.

Pegando nos valores de 780nm (equivalente ao vermelho) e 500nm (equivalente ao amarelo), e recorrendo à fórmula $f = {}^{c}/_{l}$ sendo f = frequência, c = velocidade de propagação, l = comprimento de onda, obtemos a frequência equivalente ao comprimento de onda. Passamos os valores do comprimento de onda para mm em vez de nm (0,00078 e 0,0005). Para a velocidade de propagação recorremos ao valor da velocidade da luz em mm/s, sendo este $3x10^{11}$ mm/s, ou seja, 300.000.000.000 mm/s.

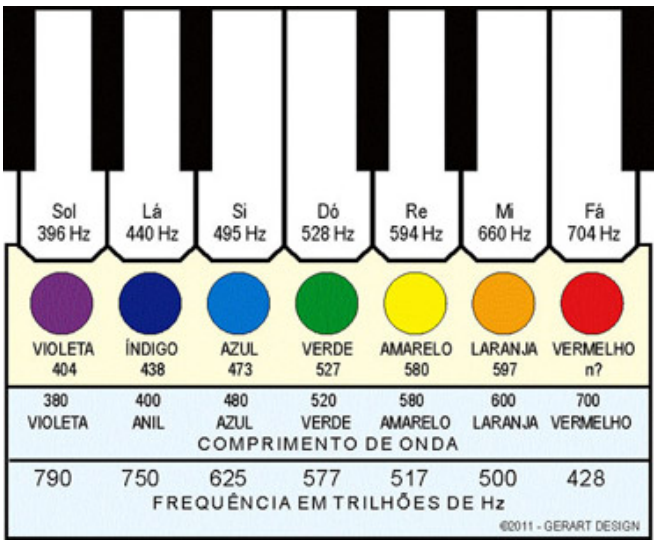


Imagem 3 - Relação: cor – nota, comprimento de onda - frequência.

Assim sendo,

$f = \frac{300.000.000.000}{l}$

$f = \frac{300.000.000.000}{0,00078}$

f = 384Hz aprox.

Agora, transpondo a frequência da cor até uma oitava conveniente musicalmente para trabalhar, obtemos a nota correspondente à cor, ou pelo menos uma aproximação.

A passagem de uma cor para a outra é feita em gradiente por isso não é possível obter uma frequência exata para cada cor, como se pode confirmar na Imagem 2 acima. Da mesma maneira, não podemos afirmar que um lá equivale exatamente a 440Hz quando é sabido que muitas orquestras e músicos afinam o lá a 441Hz ou 439Hz. Por esta razão temos cores que abrangem mais do que uma nota, o que muda é a sua tonalidade.

Esta é a forma que me pareceu mais correta de atribuir cores aos sons mas mesmo assim não podemos afirmar que é realmente "científica" pois continua a ser vaga e longe daquilo a que poderemos um dia chamar ciência exata.

Capítulo 2 – Sinestesia

Sinestesia aplicada à literatura

Figura de estilo que designa a fusão de planos sensoriais diferentes. A qualidade de um sentido atribuído a outro.

"E um **doce vento**, que se erguera, punha nas folhas alagadas e lustrosas um **frémito alegre e doce**." (Eça de Quiroz)

"Sempre havia, ao amanhecer, uma **cor estridente** no horizonte." (Giuliano Fratin)

Sinestesia aplicada à ciência

Série de fenómenos provocados por uma condição neurológica que "confunde" ou "une" duas qualidades sensoriais distintas.

2.1 – Sinestesia como condição neurológica

O ouvido é um órgão complexo capaz de descodificar frequências entre os 16Hz e os 20000Hz.

Divide-se em 3 partes:

- Ouvido Externo: funciona como protetor e capta o som.
- Ouvido Médio: converte a pressão sonora em impulsos mecânicos.
- Ouvido Interno: converte os impulsos mecânicos em impulsos eléctricos.

Os impulsos eléctricos são processados pelo cérebro.

O olho equipara-se em grau de complexidade ao ouvido. Captam a luz transformando-a em impulsos elétricos para que o cérebro descodifique toda a imagem.

Os nervos que conduzem os impulsos óticos ao cérebro são relativamente próximos daqueles que transportam a informação auditiva. Em certos casos, pode ocorrer um cruzamento entre eles como se houvesse uma troca de informação entre uns e outros. Isso faria com que "ouvíssemos" cores ou "víssemos" sons. A esse fenómeno damos o nome de sinestesia.

Muitos foram os compositores que se suspeita que terão tido esta condição. É o caso de Scriabin, Messiaen, Sibelius, Cesar Franck, Debussy, Tchaikowsky, Listz, entre outros.

2.2 – Sinestesia como condição estética

Muitos são afectados pela sinestesia através de uma malformação, outros preferem adota-la com recurso à imaginação e ao gosto próprio.

Para "ouvirmos" cores podemos recorrer à nossa criatividade. Nada nos impede de dizer "para mim o vermelho soa a um acorde maior" ou "o castanho soa a uma nota grave, talvez um Sol2".

O nosso cérebro é um instrumento controlado por nós daí termos todas aquelas teorias que relacionam a cor e o som baseando-se em todo o tipo de matérias. Podemos até criar a nossa própria paleta de cores e relacioná-las com o som que quisermos.

Os sinestésicos preferem associar cores às tonalidades, intervalos, acordes e timbres mas, tal como se poderá ilustrar no capítulo seguinte, não há nenhum padrão definido, cada um "vê" o som que quer.

Capítulo 3 – Propostas de Relação Cor/Som - Escalas Coloridas

Como já foi referido anteriormente, a relação cor – música foi analisada ao longo dos anos por músicos, esotéricos, musicoterapeutas, artistas e apenas curiosos. Entre muitas correspondências possíveis entre as características da cor – hue (cor pura, matiz), saturation (intensidade da cor), value (valor da luminosidade) – e as do som – pitch (nota), amplitude e tone color (timbre) – as propostas mais relevantes utilizam a relação pitch – hue. Muitas foram as prospostas ao longo dos tempo mas estas são, possivelmente as mais relevantes.

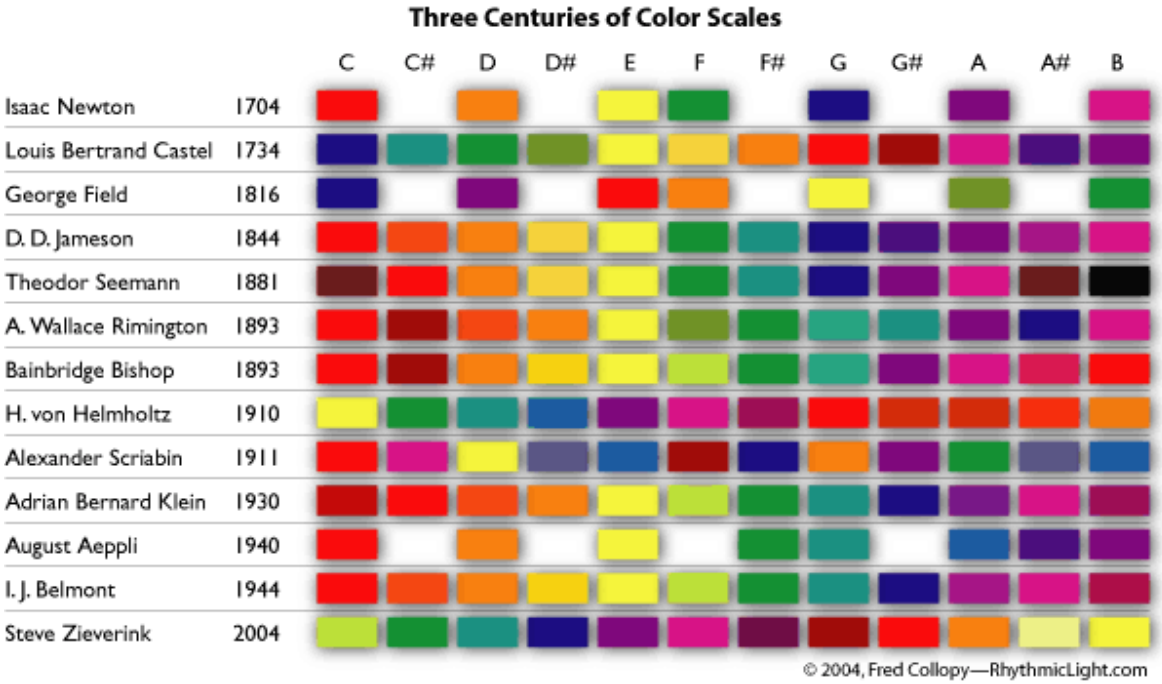


Imagem 4 - Tabela de escalas cor - nota.

3.1 – Isaac Newton

Newton foi o primeiro a entender o arco-íris refractando a luz branca com um prisma, obtendo assim os seguintes componentes: vermelho, laranja, amarelo, verde, azul e violeta. A partir daí percebeu-se que a luz é que criava as cores.

Newton dizia que "a cor deveria ser dividida entre os seus principais graus, Vermelho, Laranja, Amarelo, Verde, Azul, Indigo e Violeta, tal como as notas divididas em tons".

A escala de Newton seria então Vermelho - Dó, Laranja - Ré, Amarelo – Mi, Verde - Fá, Azul - Sol, Indigo - Lá e Violeta - Si. (Gerstner, p.167)

.

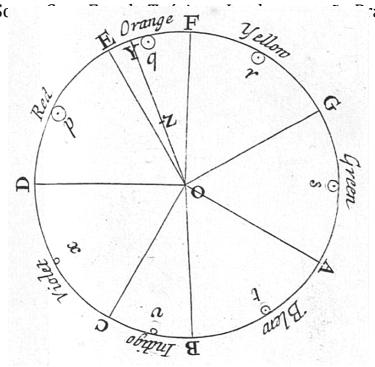


Imagem 5 - Círculo de 7 notas e 7 cores

3.2 – Louis Bertrand Castel

Louis Bertrand Castel foi um conhecido matemático francês. Em 1725, fez a sua própria ilustração dos sons para aplicar ao *Ocular Harpsichord* ou *clavecin oculaire*, um instrumento musical com 60 pequenas janelas, cada uma com um vidro colorido diferente e uma pequena cortina que levantava cada vez que se pressionava a tecla correspondente, fazendo com que a luz referente à nota se mostrasse. O compositor alemão Telemann deslocou-se de propósito a França para ver este instrumento, ficou tão cativado que compôs uma série de peças para *Ocular Harpsichord* e escreveu um livro sobre o mesmo.

Imagem 6 – Caricatura de Castel e Ocular Organ by Charles Germain de Sint Aubin

Aquando a construção do *Ocular Organ,* Castel desenvolveu um jogo de cores, não para os oito tons mas sim para as 12 notas da escala dodecafónica.

Assim sendo, propôs, totalmente a seu gosto, que a sua escala de cores fosse:

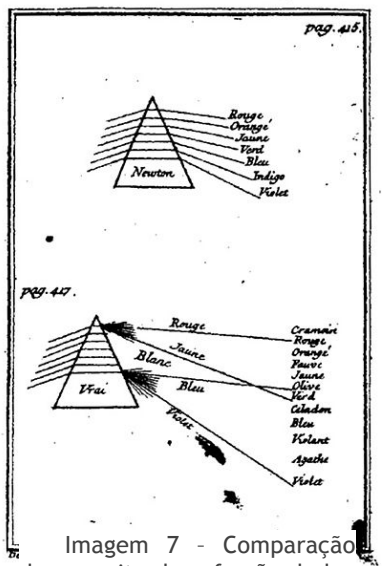


Imagem 7 – Comparação do conceito de refração da luz de Newton com Castel.

Azul - Dó;	Vermelho – Sol;
Azul esverdeado – Dó#;	Carmesim – Sol#;
Verde – Ré;	Violeta – Lá;
Verde *olive* – Ré#;	*Agate* – Lá#;
Amarelo – Mi;	Índigo – Si;
Amarelo torrado – Fá;	
Laranja – Fá#;	(Peacock, p.400)

A oitava acima seria um tom mais clara e a oitava abaixo um tom mais escura.

Em 1740, escreveu a obra *Optique des Couleurs* e publicou uma crítica à teoria do prisma de Isaac

Newton. Bertrand Castel defendia que Newton só tinha analisado um caso e que na realidade a luz branca difundida pelo prisma depende da distância a que se encontra do prisma.

Castel morreu em 1757. Nesse mesmo ano, um assistente inglês "deu vida" ao projeto mais ambicioso de Castel, um *Ocular Hapsichord* com 12 oitavas, ou seja, 144 teclas que, quando pressionadas, abrem as persianas individuais que estão sobre aberturas quadradas contendo painéis coloridos.

3.3 – George Field

Field criticou o padrão de cores de Castel e propôs uma união num "universal system of Analogical Philosophy".

Segundo Field, as cores primárias têm uma harmonia e melodia significante e representam a Trindade, equivalem assim às notas Dó, Mi e Sol, ou seja, formam o acorde de Dó Maior. As secundárias foram distribuídas pelo resto das notas "estranhas" ao acorde, sendo as graves mais escuras e as agudas mais claras. Ao longo do seu estudo percebeu mais tríades e defende que tudo gira à volta das mesmas, por exemplo, arco, ângulo e linha, as vogais O, A e I.

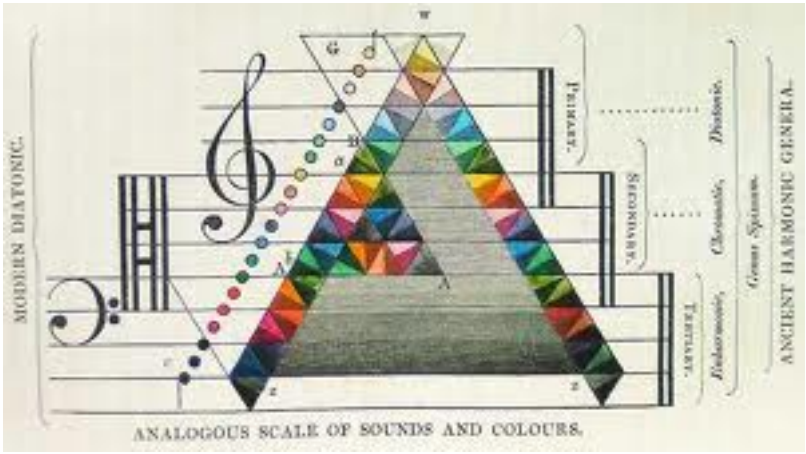


Imagem 8 – Universal System of Analogical Philosophy.

Owen Jones recorreu às ideias de Field quando estava em processo criativo de design do Palácio de Cristal, em 1850.

Assim seria a escala de George Field:

Azul - Dó;

Roxo – Ré;

Vermelho – Mi;

Laranja – Fá;

Amarelo – Sol;

Verde amarelado – Lá;

Verde – Si;

(Klein, p.69)

3.4 – D. D. Jameson

Jameson publicou *Colour-Music*, em 1844, uma descrição de um sistema de notação que consistia na associação das notas às cores, a oitava à altura e a duração à largura, a acentuação ou intensidade à instensidade da cor, as pausas ao espaço em branco.

"A notação colorida possui a vantagem de fazer que com que toda, ou uma parte considerável da música que representa, fique presente na mente através do olho." *[1]

A escala colorida segundo Jameson seria então:

Vermelho - Dó;

Vermelho alaranjado– Dó#;

Laranja – Ré;

Laranja amarelado – Ré#;

Amarelo – Mi;

Verde – Fá;

Verde azulado – Fá#;

Azul – Sol;

Azul arroxeado– Sol#;

Roxo – Lá;

Roxo –Violeta – Lá#;

Violeta – Si;

(Jameson, p.12)

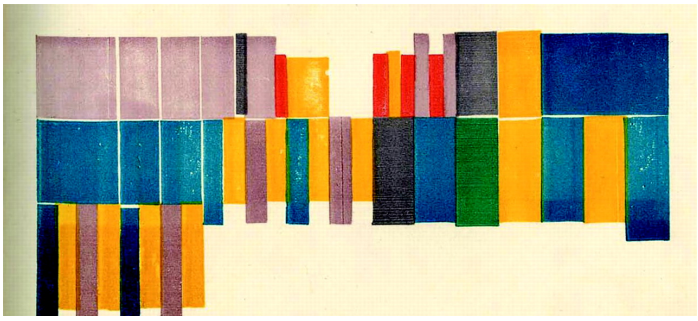


Imagem 9 – Di Tanti Palpiti segundo a notação de Jameson

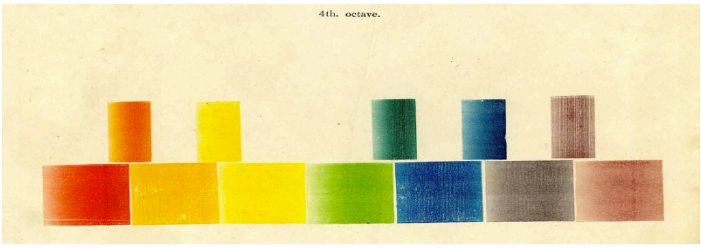


Imagem 10 – Escala colorida da 4ªoitava.

[1] Jameson, D. D., *Colour-Music*, London, Smith, Elder and Co., 65, Cornhill, 1844 EM ANEXO (anexo 6) traduzido do original em Inglês.

3.5 – Theodor Seemann

Não há muito a dizer sobre Theodor Seemann, o pouco que se sabe é relacionado com o seu trabalho *The Laws of Colour Harmony.* Nesse estudo associou as cores com as notas da seguinte forma:

Carmim – Dó;

Scarlet – Dó#;

Laranja – Ré;

Laranja amarelado – Ré#;

Amarelo – Mi;

Verde – Fá;

Verde azulado – Fá#;

Azul – Sol;

Indigo– Sol#;

Violeta – Lá;

Castanho – Lá#;

Preto – Si;

(Klein, p.86)

3.6 – Alexander Wallace Rimington

Rimington era pintor, começou a estudar a analogia entre som e cor inspirado pelas técnicas do Impressionismo Francês. Na altura não havia registos acerca do que Castel havia feito por isso, Rimington achava que ao criar o seu *Colour-Organ*, estava a inventar uma nova arte. Em 1895, na sua dissertação *A New Art: "Colour-Music"* , comparou o seu instrumento a *"uma palete de cores pela qual poderíamos pintar com efeito instantâneo na tela combinando as cores num só acorde ou compor matizes na sua superfície".* *[2]

Patenteou o seu *Colour-Organ* em 1895 e descreveu-o detalhadamente no seu livro *Colour-Music: The Art of Mobile Colour* (1911). O instrumento consumia o equivalente a 1300 velas. A luz das mesmas passava através de vidros coloridos que rodavam de forma a obter uma variedade de intensidade das diferentes cores. Tal como o *clavecin oculaire* de Castel, este também não produzia som mas já apresentava possibilidades de ajuste de certos valores de cor, luminosidade e saturação.

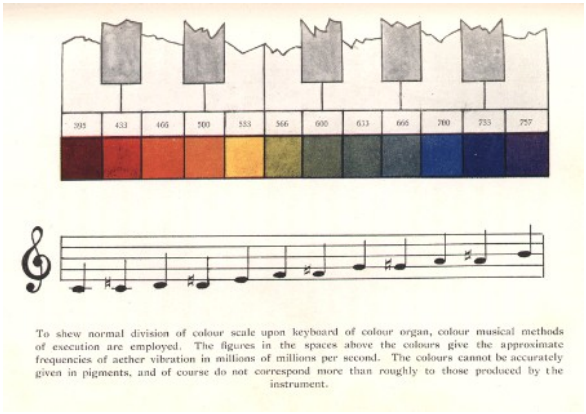


Imagem 11 – Página do livro Colour-Music: The Art of Mobile Colour (1911).

[2] *traduzido do original em Inglês.

Assim a escala proposta por A. Wallace Rimington foi:

Vermelho – Dó;	Verde – Fá#;
Carmesim – Dó#;	Verde azulado – Sol;
Laranja-Carmesim – Ré;	Azul esverdeado – Sol#;
Laranja – Ré#;	Indigo – Lá;
Amarelo – Mi;	Azul escuro – Lá#;
Amarelo esverdeado – Fá;	Violeta – Si;

(Peacock, p.402)

3.7 – Bainbridge Bishop

Bishop foi contemporâneo de Rimington e, tal como ele, altamente influenciado pelo Impressionismo Francês. Na altura em que Bishop publicou o seu esquema de cores associado à escala musical, já tinha construído pelo menos três *color organs*, capazes de reproduzir cor e som ao mesmo tempo ou separadamente. O problema dos *organs* era que a luz provinha de velas, que muitas vezes incendiavam, por essa razão a maior parte dos *organs* contruídos até essa altura foram destruídos em incêndios.

"Eu fiz um número de instrumentos experimentais, remodelando-os e alterando-os para realizar a ideia de melhor maneira e obter um melhor efeito. (...) Acordes eram difundidos propriamente, o baixo espalhava-se por todo o espaço (...) e por toda a mobília, bonitos e harmoniosos efeitos em combinação com a música. " *[3]

Bishop, Bainbridge, *A Souvenir of The Colour Organ, with some suggestions in regard to thesoulof the rainbow and the harmony of Light with marginal notes and illuminations,* New Russia, Essex County, N.Y. 1893 Copyright, 1893.

A escala proposta por Bishop é idêntica à de Rimington:

Vermelho – Dó;	Verde – Fá#;
Scarlet – Dó#;	Verde azulado ou *aquamarine*– Sol;
Laranja – Ré;	Azul – Sol#;
Dourado ou amarelo torrado – Ré#;	Indigo ou Violeta-azul – Lá;
Amarelo ou Verde-dourado – Mi;	Violeta – Lá#;
Amarelo esverdeado– Fá;	Violeta-vermelho – Si;

(Bishop, p.11)

[3] *traduzido do original em Inglês.

[4] *traduzido do original em Inglês.

3.8 – H. Von Helmoltz

Helmholtz é reconhecido como uma das personalidades mais importantes da área científica do sec.XIX. Escreveu *Manual of Physiological Optics*, onde faz um estudo da cor ao pormenor defendendo que as três cores principais são: Vermelho, Verde e Azul-violeta.

"Dourado ocupa apenas uma posição entre o amarelo e o laranja, tal como o indigo entre o azul e o violeta; o mesmo acontece no que respeita ao amarelo esverdeado ou azul esverdeado. De verdade, não há divisões reais entre as cores do espectro. Essas divisões são mais ou menos caprichosas e claramente um resultado de um mero amor de chamar as coisas pelo nome. Na opinião do autor, portanto, esta comparação entre música e cor deve ser posta de parte."[4]

Mesmo assim, há registo de uma proposta de escala de cor de Helmholtz:

Amarelo – Dó;

Verde – Dó#;

Verde azulado – Ré;

Cayan-blue – Ré#;

Indigo azul – Mi;

Violeta – Fá;

Final de vermelho – Fá#;

Vermelho – Sol;

Vermelho – Sol#;

Vermelho – Lá;

Vermelho-laranja – Lá#;

Laranja – Si;

(Helmholtz, p.22)

[4] Helmhotz, H. Von, *Treatise on Physiological Optics*, Vol II, 117, 1910.

3.9 – Alexander Scriabin

Scriabin foi um compositor e pianista russo. Mantinha uma excelente relação com Sergei Rachmaninoff e Rimsky Korsakov. Rachmaninoff em *Recollections* gravou uma conversa entre os três sobre a associação que Scriabin fazia entre a cor e som. Nessa mesma conversa, Rachmaninoff fica admirado por Korsakov apoiar esta ideia de se interligar os dois elementos, mas as opiniões de Korsakov e Scriabin começam a divergir quando chegou a altura de atribuir uma cor a cada nota.

Nas obras *Prometeu* (1910) e *Poema do Êxtase* (1908), usava jogos de cores recorrendo à relação entre a música e cores, de maneira a criar misticismo à audiência. O segundo possuía uma parte para uma máquina conhecida como *clavier à lumiéres* que era, nada mais nada menos, que um *color organ* desenhado especificamente para a performance deste poema. Nem todas as performances incluíam este instrumento, mas a de N.Y. em 1915 teve direito às suas cores projectadas na tela.

Scriabin descreveu um teclado sinestésico pelo qual se guiava para compor as suas obras.

C	Db	D	Eb	E	F	F#	G	Ab	A	Bb	B
Red	Violet	Yellow	Steel	Pale Blue	Dark Red	Bright Blue	Orange	Purple	Green	Steel	Pale Blue

Imagem 12 - Teclado Sinestésico de Scriabin.

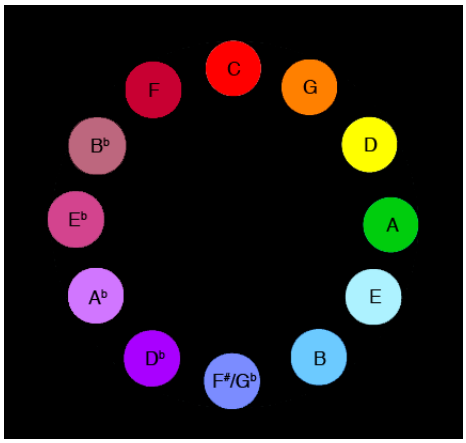


Imagem 13 - Circulo de quintas espectral.

Também distribuiu as cores pelo ciclo de quintas.

Muitos falam da possibilidade de Scriabin sofrer de sinestesia.

Ele fez as suas composições mais sobre uma visão cor-tonalidade do que propriamente cor-nota. Fez a distinção entre tonalidades "espirituais", "materiais" e "terrestres". Ele dava qualidades às cores, por exemplo, o azul e violeta sendo as "cores da razão".

A escala de Scrabin era bastante subjectiva:

Vermelho – Dó;

Violeta – Dó#;

Amarelo – Ré;

Aço com um brilho de metal – Ré#;

Azul pérola do brilho do luar – Mi;

Vermelho escuro – Fá;

Azul brilhante – Fá#;

Laranja rosado – Sol;

Roxo – Sol#;

Verde – Lá;

Aço com um brilho de metal – Lá#;

Azul pérola do brilho do luar – Si;

(Jones, p.104)

3.10 – Rimsky-Korsakov

Não podemos falar de Scriabin e deixar Korsakov de parte. Korsakov atribuiu cores às tonalidades de uma forma intuitiva e espontânea:

Dó Maior – Branco;

Sol Maior – Castanho dourado, luz;

Ré Maior – Luz do dia, amarelado, real;

Lá Maior – Rosa claro;

Mi Maior – Azul safira brilhante;

Si Maior – Sombrio, azul escuro com brilho de aço;

Fá# Maior – Cinza-verde;

Réb Maior – Escura e quente;

Láb Maior – Cinza-violeta;

Mib Maior – Escuro e sombrio, cinza azulado;

Sib Maior – Escuro;

Fá Maior – Verde claro;

Imagem 14 – Círculo de quintas de Korsakov.

3.11 – Ira Jean Belmont

Belmont era um pintor nova iorquino conhecido pelas suas obras de *Color-music*.

Não foi encontrada muita informação sobre este artista, mas sabe-se que o ideal dele era passar as músicas de grandes compositores para quadros.

Imagem 15 – Reprodução de Greensleeves por Belmont.

Imagem 16 – Reprodução de Variações Enigma de Elgar por Belmont

A escala de cor que ele utilizava era:

Vermelho – Dó;
Vermelho alaranjado – Dó#;
Laranja – Ré;
Amarelo Torrado – Ré#;
Amarelo– Mi;
Amarelo esverdeado – Fá;
Verde – Fá#;
Azul-verde – Sol;
Azul – Sol#;
Azul-violeta – Lá;
Violeta– Lá#;
Vermelho-violeta – Si;

(Belmont, p.226)

3.12 – Oliver Messiaen

Oliver Messiaen nasceu em França em 1908, foi compositor, organista e ornitólogo.

Escreveu *Traité de rythme, de couleur, et d'ornithologie* (*Tratado de ritmo, cor e música de pássaros*), no qual descreve acordes como "rochas azul-violetas, salpicadas com pequenos cubos cinzentos, cobalto azul, azul *Prussian* profundo, destacado por um pouco de violeta-roxo, dourado, vermelho, *ruby*, e estrelas de malva, pretas e brancas". *[5]

Imagem 17 – *Simultaneous Contrast: Sun and Moon*, de Robert Delaunay, Paris, 1912.

Messiaen idolatrava Robert Delaunay, um pintor de arte abstrata conhecido pelo contraste simultâneo que usa nos seus quadros.

Isto de contraste simultâneo é um fenómeno que ocorre quando a visão procura encontrar equilíbrio entre as cores. O olho é sensibilizado por uma cor e automaticamente vai procurar uma complementar dessa cor para que esses tons se anulem e voltem ao equilíbrio inicial, atingindo o que chamam de "harmonia cromática".

Messian associava as cores a agregados ou configurações melódicas e a luminosidade à oitava. Quanto mais agudo mais claro. (Cristina Henriques, 2013)

[5] *traduzido do original em Inglês.

"Messiaen defende que a sua sinestesia é de natureza intelectual, não se devendo a nenhuma condição física, mas sim a uma capacidade interior de associar a audição à visão, de gerar uma audição-colorida, que se instaura tanto quando ouve música como quando a lê. "

***Cristina Henriques** / 13.Maio.2003*

Retirado do site da Fundação Calouste Gulbenkian

3.13 – Steve Zieverink

Natural de Chicago, Zieverink é um artista e músico da atualidade. Em 2000, fundou a comunidade virtual de artistas onde colaboram e organizam projetos, a *Unit 2 art collective*.

Zieverink defende que a música, a matemática e o xadrez melhoram a funcionalidade natural do cérebro humano por exigir uma concentração extra nas associações e estratégias obrigando-nos a encontrar padrões de resolução de problemas. Nas suas obras, cria uma ligação entre estas 3 áreas.

Este artista fez também uma proposta de interligação do som com a cor, obtendo então esta escala:

Imagem 18 – Escala de Zieverink.

Verde-amarelo – Dó;
Verde – Dó#;
Verde azulado– Ré;
Azul – Ré#;
Indigo – Mi;
Violeta – Fá;
Ultra Violeta – Fá#;
Infra-vermelho – Sol;
Vermelho – Sol#;
Laranja – Lá;
Amarelo-branco – Lá#;
Amarelo – Si;

(Cincinnati Contemporary Art Center)

Uma aluna de mestrado em Gráfico, Marca e Identidade no London College of Comunication, Natasha Stone, fez um estudo sobre as diferenças entre algumas escalas de cor, num trabalho em que desenvolve os seus conhecimentos acerca da consolidação do som e visão e em que desenvolve uma notação através da cor.

No seu trabalho transformou as notas escritas por Haendel em *La Rejouissance* de 1700 e a música *Purple Haze* do músico Jimi Hendrix em quadrados coloridos segundo a escala de Newton. Depois, comparou a música de Jimi Hendrix escrita sob o olhar de Newton, com a mesma música escrita sob o olhar de Zieverink.

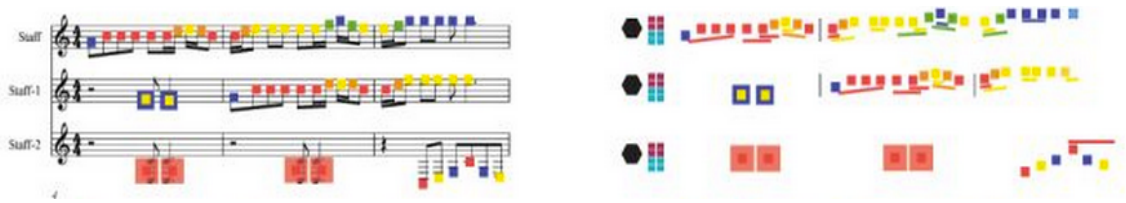


Imagem 18 – Lado esquerdo, atribuição das cores nas notas da partitura e do lado direito o mesmo excerto mas agora com a notação criada pela aluna.

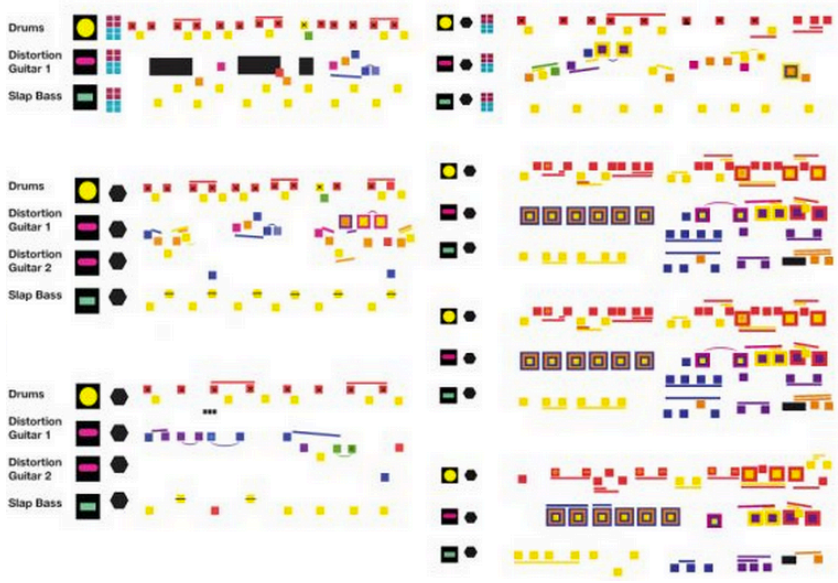


Imagem 19 – Música *Purple Haze* de Jimi Hendrix pelos olhos de Newton.

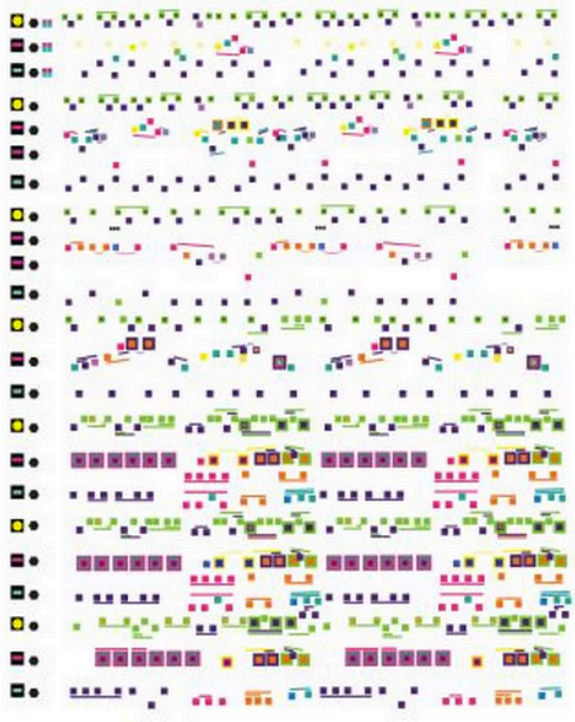


Imagem 20 – Música *Purple Haze* de Jimi Hendrix pelos olhos de Zieverink.

Capítulo 4 – Fase de Demonstração Prática do Estudo

4.1 – Aplicação demonstrativa

A aplicação desenvolvida no decorrer deste estudo pretende mostrar as várias propostas de relação cor – nota feitas ao longo dos tempos. Tende a ser uma aplicação mais virada para o lado demonstrativo mas pretende-se desenvolvê-la de maneira a fazer uma abordagem mais pedagógica e interativa.

Esta primeira versão permite ao utilizador escolher o autor da escala cor-nota que pretende utilizar. Após escolher os "olhos" pelos quais quer ver, o utilizador dispõe de um *piano roll* que, ao acionar a nota com o rato ou com as teclas do computador, devolve um feedback visual da nota que está a ser tocada ao "pintar" a tecla com a cor correspondente a essa mesma nota. A aplicação dispõe também de reproduções em MIDI de melodias predefinidas.

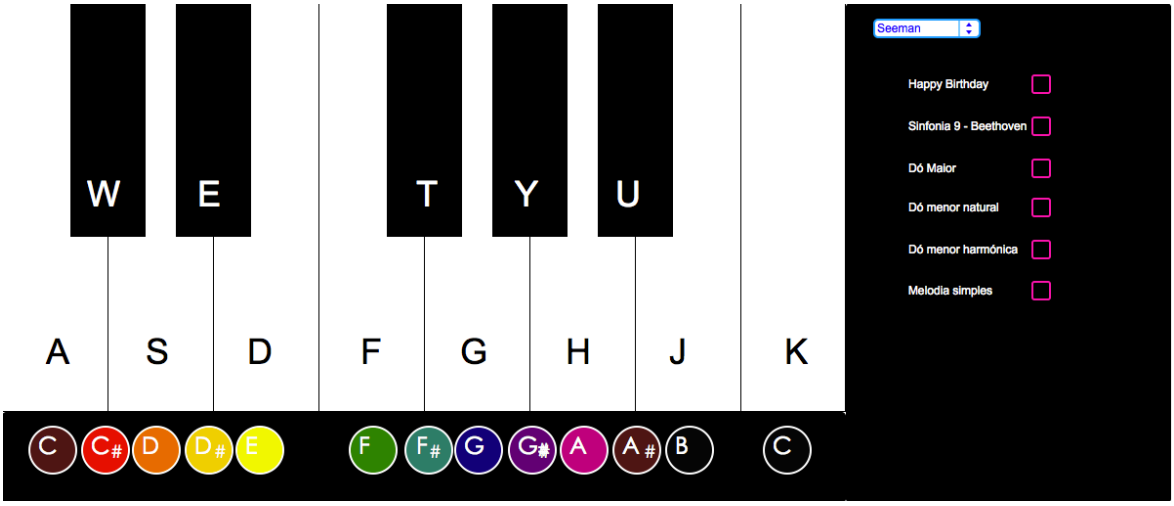


Imagem 33 - Print Screen da aplicação.

4.2 – Elaboração

A aplicação foi programada em Max/MSP recorrendo a conhecimentos adquiridos durante o curso e pesquisa. Os dados apresentados na aplicação foram todos encontrados durante a pesquisa para este projeto e estão apresentados no Capítulo 3.

A aplicação começou por ser realizada através de "if's" mas rapidamente apresentou problemas de funcionalidade e usabilidade. Depois de expor o problema ao orientador, chegou-se a conclusão que a melhor maneira de fazer o pretendido era usando o objecto "coll".

Assim, foi criado um "coll" para cada proposta de escala, nomeando os mesmo com o nome do artista em si.

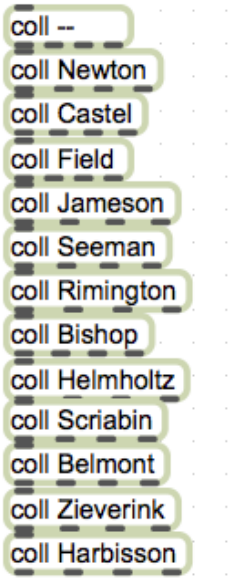


Imagem 35 – Print Screen dos coll utilizados.

Como podemos verificar na imagem, os artistas pelos "olhos dos quais podemos ver" são: Isaac Newton, Louis Bertrand Castel, George Field, D. D. Jameson, Theodor Seeman, A. Wallace Rimington, Bainbridge Bisop, H. Von Helmholtz, Alexander Scriabin, I. J. Belmont, Steve Zieverink e finalmente, Neil Harbisson. Há também a possibilidade de ver como resulta usando a transposição de frequência cor-som.

Com o objecto "Colorpicker" e selecionando a cor pretendida, consegui obter os valores da cor em valores de 0. a 1.. Anteriormente pensei utilizar valores RGB mas a compatibilidade é baixa com o Max, o que originou um défice na amostragem das cores, daí a alternativa serem os valores de 0 a 1.

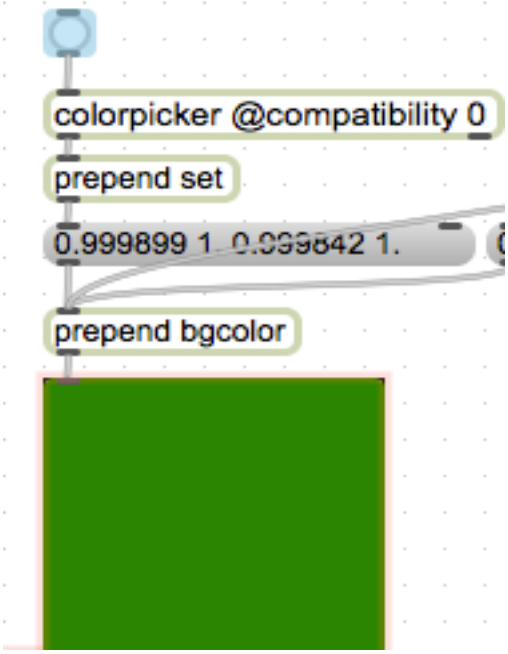


Imagem 35 – Print Screen da aplicação – "color picker".

Seguidamente defini os valores de cada cor em cada “coll”.

Harbisson.txt

```
0, 0.15778 0.504405 0.965157 1.;
1, 0. 0.093663 0.953371 1.;
2, 0.437196 0.130983 0.956609 1.;
3, 0.899941 0.189875 0.967385 1.;
4, 0.909444 0.165719 0.493116 1.;
5, 0.904842 0.156616 0.097662 1.;
6, 0.939747 0.517528 0.147886 1.;
7, 0.999917 0.989269 0.24022 1.;
8, 0.616776 0.978891 0.218827 1.;
9, 0.371007 0.970816 0.214296 1.;
10, 0.368198 0.973641 0.533933 1.;
11, 0.35728 0.980987 0.989876 1.;
12, 0.15778 0.504405 0.965157 1.;
```

Imagem 36 – Print Screen da aplicação – “coll” de Harbisson.

A aplicação tem um “umenu” que contem todos os nomes dos artistas e onde podemos selecionar o artista pretendido.

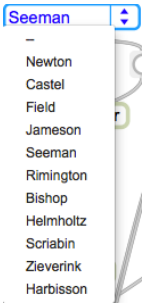


Imagem 37 – Print Screen da aplicação – “umenu”.

A ligação é feita do segundo output do “umenu" (output de mensagem) para uma caixa de mensagem, que é lida pelo “prepend refer” chamando assim as informações referentes ao “coll” com o nome que passou na mensagem, passando-as seguidamente para dois “coll”. O primeiro envia a informação da cor referente à tecla pressionada para uma mensagem lida pelo “prepend bgcolor” ligado a um “panel”. O segundo está a ser controlado por um “metro” para enviar as informações das cores referentes à lista selecionada que está dentro do “coll” para os leds que representam a escala. Inicialmente experimentei usar o uzi em vez do metro mas dava-me problemas e trocava só algumas cores. Não percebi bem o porquê de acontecer isso, por isso, passei automaticamente para outra solução.

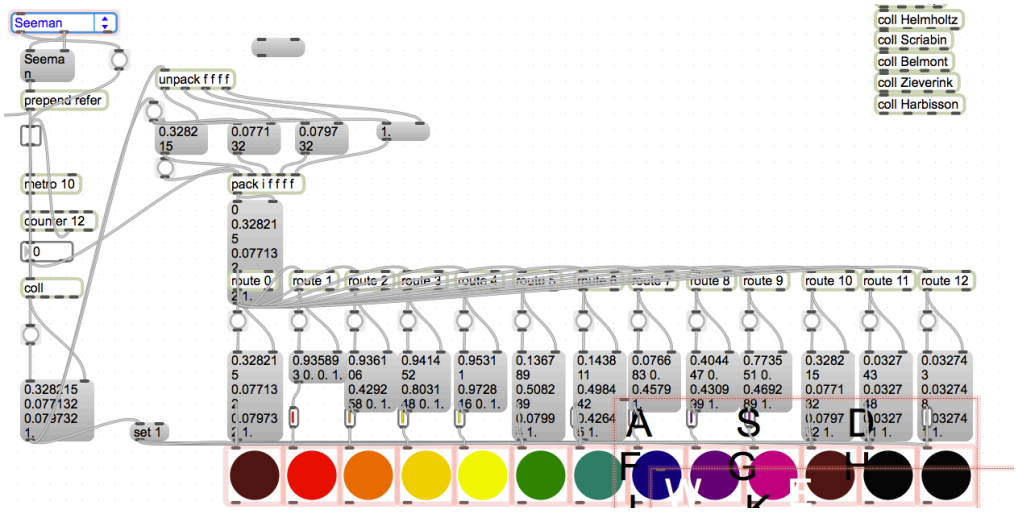


Imagem 38 – Print Screen da aplicação – segundo “coll” e atribuição das cores nos leds

A informação de cada cor é obtida através de um "pack" de todos os elementos necessários: número da posição da informação e os 4 parâmetros da cor. Obtemos assim uma mensagem com uma lista de 5 elementos em que o primeiro é o "índex". Fazemos "route" do mesmo e deixamos passar apenas a informação dos 4 parâmetros da cor que são enviados para um "oncolor" ligado a cada LED.

O "kslider" está definido com valores de 0 a 12 para facilitar a leitura e a programação da aplicação. Quando a tecla é pressionada ouvimos o som referente à mesma e a tecla acende na cor correspondente. O teclado pode ser acionado através do clique do rato sobre o "kslider" ou através das letras "A W S E D F T G Y H U J K".

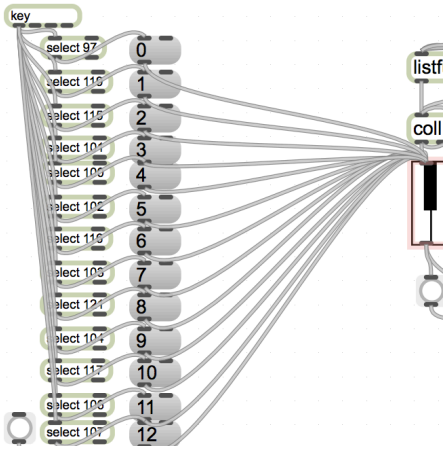


Imagem 39 – Print Screen da aplicação – atribuição das teclas do teclado do computador às notas do "kslider".

No caso de Newton e Field, as escalas apenas têm cores nas notas brancas, pois as propostas eram referente apenas aos 7 sons principais. Isso também acontece na de frequências pois não consegui obter uma informação visual que me permitisse recolher uma amostra de cor referente às notas das teclas pretas. Mas esse ponto é uma questão a melhorar.

A aplicação apresenta também a possibilidade de, em vez de ser o utilizador a tocar uma melodia, reproduzir a escala de Dó Maior, Dó menor, Dó menor harmónica e as melodias simples dos Parabéns e do Hino da Alegria.

Capítulo 5 – Fase de Definição do projeto

5.1 – Inspiração

Neil Harbisson é um artista e presidente da Fundação Cyborg. Nasceu em 1982, em Londres, mas passou a sua infância em Mataró (Catalunha, Espanha), onde estudou música, dança e drama. Aos 11 anos já compunha para piano. Harbisson nasceu com uma condição visual chamada acromatopsia, que o impede de ver cores, ou seja, só consegue ver a preto e branco. Por ironia, ingressou no Instituto Alexandre Satorras no curso de Belas Artes, onde lhe foi concedido um estatuto que lhe permitia apresentar trabalhos só a preto e branco. Em 2001, mudou-se para a Irlanda par completar os seus estudos em piano na *Waltons New School of Music* em Dublin. Em 2002, começou os seus estudos de Composição Musical no *Dartington College of Arts* em Inglaterra.

O projeto de Neil começou a ser desenvolvido em 2003. Após assistir a uma palestra sobre cibernética dada por Adam Montadon, um especialista em futuro digital e consultor de negócio criativo, propôs-lhe começar um projeto conjunto. A partir desse momento começaram a criar o Eyeborg, um dispositivo montado na cabeça que possui um sensor de frequência de cor e transpondo-as para frequências de som. Neil passou então a ouvir as cores.

Este Eyeborg ganhou, em 2004, o *Europrix Award in Content Tools and Interface Design*, em Viena. Em 2007, conheceu Peter Kese, um programador de software que desenvolveu o seu Eyeborg de maneira a perceber a saturação da cor através dos volumes e a identificar 360 tons através de microtons.

Este artista foi, sem dúvida, a maior inspiração para a realização deste projeto. Não só pelo trabalho que elaborou mas também pela força de vontade e o facto de toda a sua vida ter estado ligado à música. O interesse neste artista surgiu logo no inicio da pesquisa, aquando a visualização do vídeo da conferência feita no âmbito do programa TED (*Techonology, Entertainment, Design*) em que apresenta o seu projeto "The Sound of Colour".

Na conferência TED, Neil apresenta este projeto começando com a frase "*I've never seen color and I don't know what color looks like.*"*[6]. A sua vida passou a ser uma sinfonia colorida a toda a hora. Diz que vê as cores das pessoas, cada pessoa tem um agregado de notas, os quadros de artistas conceituados podem soar bem ou soar mal, quando se veste, fá-lo consoante os sons e não as cores. *"Today I'm wearing C Major."*[7] disse na mesma conferência, vestido com umas calças amarelas, uma camisola azul e um casaco rosa. Diz também que teve de decorar o nome das cores e associa-los às notas, mas que passado um tempo toda a informação que o seu cérebro

[6] "Eu nunca vi cor e não sei como é que a cor é."

[7] "Hoje estou a vestir Dó Maior."

passou a uma percepção, seguidamente a um sentimento, e a dada altura começou a ter cores preferidas.

"When I started to dreams in color, I felt the software and my brain had United. And that's when I started to feel like a cyborg, It had became a part of my body, an extension of my senses"[8]

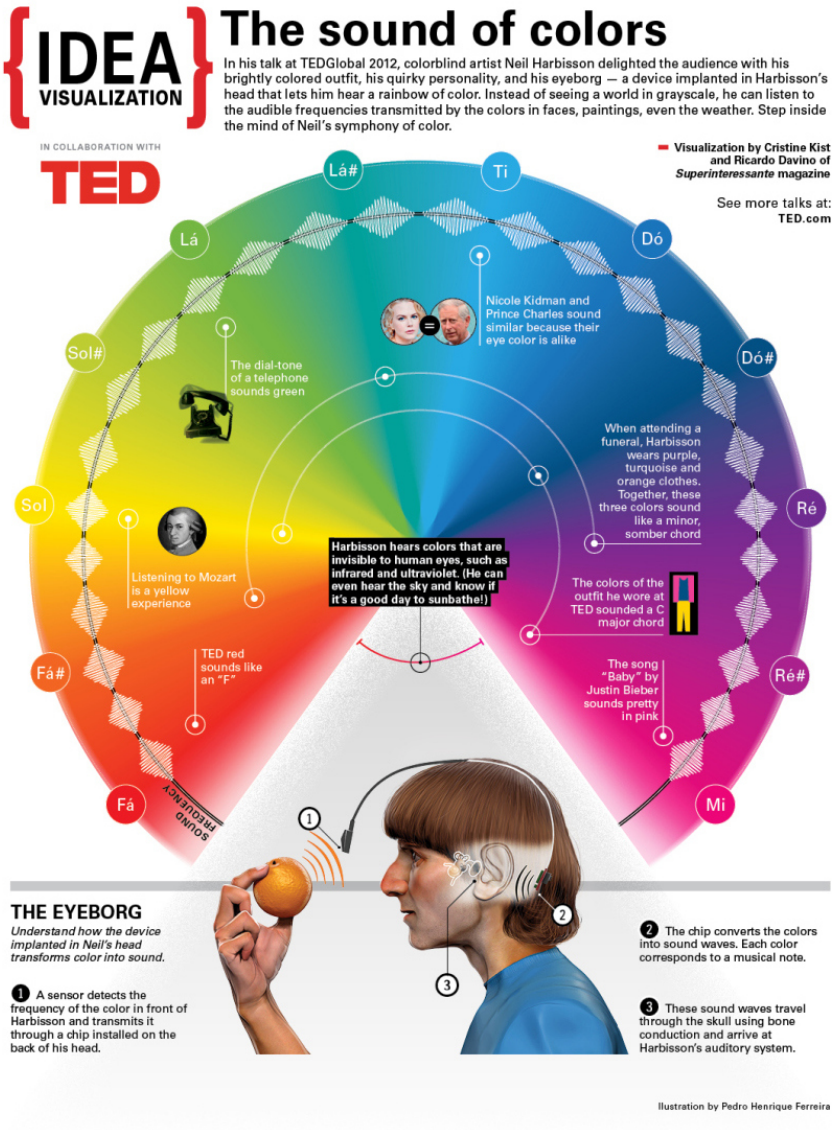


Imagem 21 – *Neil Harbisson* - Ilustração de Pedro Henrique Ferreira.

Neil diz então que sofre de Sonocromatismo ou Sonocromatopsia, pois a acromatopsia significa não ser capaz de perceber as cores, mas ele consegue faze-lo através dos sons, mas mesmo assim não as consegue ver, não sabe como são, sabe como soam.

[8] "Quando comecei a sonhar a cores, eu senti que o software e o meu cérebro se tinham unido. E foi quando eu comecei a sentir-me um cyborg, (o software) passou a ser parte do meu corpo, uma extensão dos meus sentimentos."

Usou então a escala musical sonocromática logo na primeira versão do seu eyeborg, em 2004.

A escala pura sonocromática é exclusivamente uma transposição das frequências da luz para frequências do som.

PURE SONOCHROMATIC SCALE		
(invisible)	Ultraviolet	Over 717.591 Hz
	Violet	607.542 Hz
	Blue	573.891 Hz
	Cyan	551.154 Hz
	Green	478.394 Hz
	Yellow	462.023 Hz
	Orange	440.195 Hz
	Red	363.797 Hz
(invisible)	Infrared	Below 363.797 Hz

SONOCHROMATIC MUSIC SCALE (basic 12/360)		
	Rose	E
	Magenta	D#
	Violet	D
	Blue	C#
	Azure	C
	Cyan	B
	Spring	A#
	Green	A
	Chartreuse	G#
	Yellow	G
	Orange	F#
	Red	F

Imagem 22 - À esquerda a escala pura sonocromática e à direita a escala musical sonocromática.

Para além disto tem realizado trabalhos desde então. As suas obras estão entre as 10 performances mais chocantes de todos os tempos. O seu *Concerto para Piano nº1* tem como partitura um quadro com um piano *Steinway* de cauda com diferentes cores. Durante a performance ele usava o olho cibernético para saber as frequências das cores.

Na primeira performance voz-cor a cantora tinha um computador e um microfone e cantava os microtons que Harbisson usava enquanto este criava pinturas em tempo real no palco.

Imagem 23 - Harbisson a fazer um *Sound Portrait* a James Cameron.

Em relação à pintura, Neil tem feito sessões de *Sound Portraits*, ou seja, retratos através do que o olho cibernético detecta. Desde 2005,, já fez retratos de Prince Charles, Antoni Tápies, Tracey Emin, Leonardo DiCaprio, Peter Brook, Al Gore, García Bernal, Marina Abravomic, Woody Allen, entre muitos outros. Estes retratos foram todos feitos em pessoa, ele afirma que não é possível obter um retrato fiel olhando para uma imagem, pois as cores não são as mesmas.

The Human Colour Wheel é um gradiente de cores que Harbisson detecta na pele das pessoas. Para ele não há pessoas negras ou brancas, para ele a pele humana vai do laranja-vermelho até ao laranja amarelo.

Atribuiu também cores às grandes cidades que visitou. Bratislava é amarela e turquesa, Andorra verde escura e *Fuschia* (rosa-choque), o Mónaco azul e rosa-salmão. A este trabalho de recolha ele deu o nome de *Capital Colors of Europe.*

Imagem 24 - Por ordem: Londres, Lisboa e Madrid.

Finalmente, as suas partituras de cor, em que ele ouvia obras de grandes compositores e as transformava em cores.

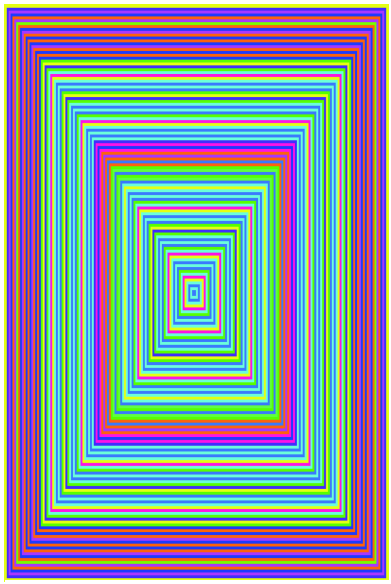


Imagem 25 – *The Rite of Spring* de Stravinsky transposta por Neil Harbisson.

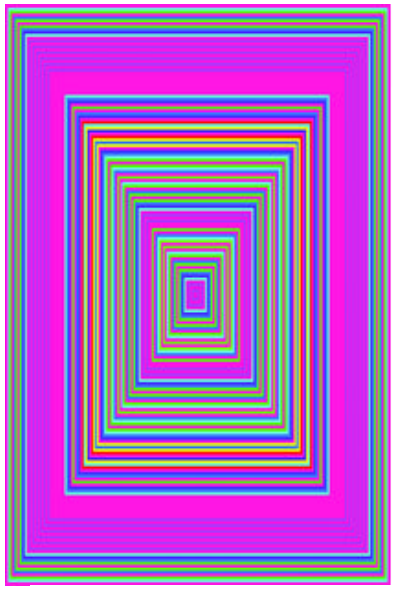


Imagem 26 – Fur Elise de Beethoven transposta por Neil Harbisson.

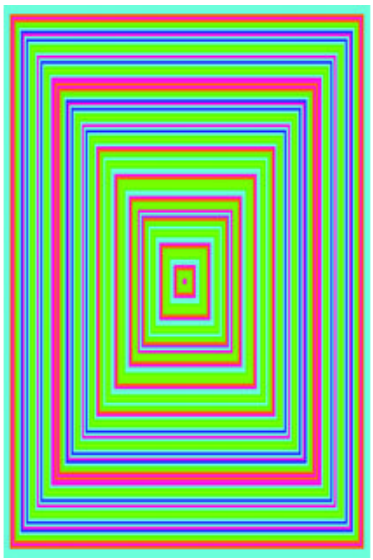


Imagem 28 – *Spring* de Vivaldi transposta por Neil Harbisson.

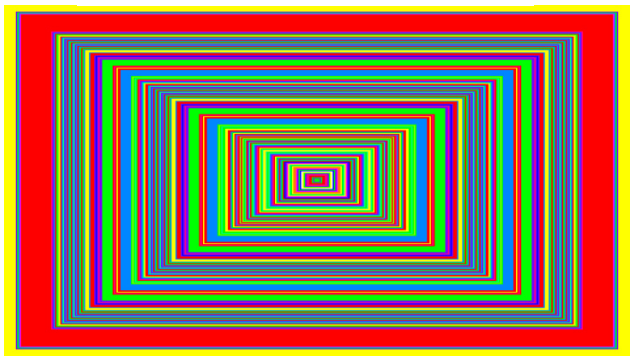


Imagem 27 – Queen of The Night de Mozart transposta por Neil Harbisson.

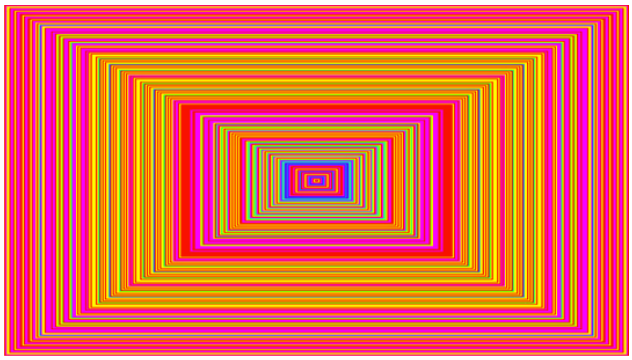


Imagem 29 – Baby de Justin Bieber transposta por Neil Harbisson.

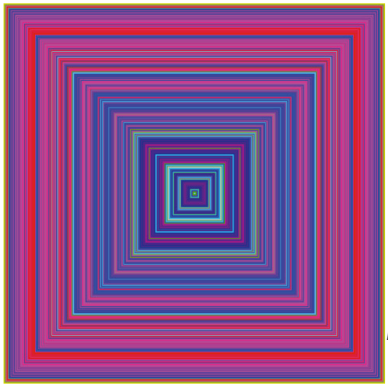


Imagem 30 – *I Have a Dream* do discurso de Martin Luther King transposta por Neil Harbisson.

Capítulo 6 – Interatividade

Interatividade não é um termo fácil de encontrar em dicionários. (RICHARDS, R., 2006). No Dicionário de Inglês de Oxford encontra-se as definições:

- Atividade que envolve interação.
- Propriedade de ser interativo.

6.1 – Conceito

Interatividade do ponto de vista sociológico

"A relação entre duas ou mais pessoas que, em determinada situação, adaptam seus comportamentos e ações uns aos outros"[9]

Interatividade do ponto de vista tecnológico

Tecnologicamente, interatividade remete para um diálogo que ocorre entre um ser humano (ou qualquer outra criatura viva) e um programa de computador. (ROUSSE, Margaret). Aquilo a que chamam "Human-Computer Interface" (HCI), traduz toda a linguagem interativa entre humano e máquina. Todo o tipo de *input* usado numa máquina, seja ele interno ou externo (teclado, rato, mesa gráfica, touchscreen, microfone, etc.). O uso destes *inputs* comanda ações visuais ou de áudio para o *output.* Essas ações formam sequências de ações que formam o que chamamos de interação. Damarin (1982), identificou uma série de opções de interação: ver, procurar/encontrar, fazer, usar, construir e criar.

Segundo Dickinson (1995), interatividade não se pode cingir apenas a um clique, tem de ser uma relação mais pessoal. Tudo se resume ao conceito e não à ação usada para interagir. Um grande conceito que funcione interativamente é o que se deve procurar atingir, seja a ação simples ou complexa.

6.1.1 – Níveis de Interação

A interação distingue-se em 3 níveis: Reativo, Coativo e Proativo. (Rhodes & Azbell,1985).

[9] JENSEN, J. F. *Interactivity: Tracing a new concept in media and communication studies,* vol. 19. Nordicom Review. 1998. Pp. 185-204

O nível reativo trata de uma relação responsiva, ou seja, em que ações diretas foram previamente programadas para desencadear certas respostas (ação – ação).

O nível coativo remete para uma relação que se processa de um forma mais generativa. A ação desencadeia um processo, ou seja, um clique ativa uma sequência de eventos e não um só evento (ação – processo).

O nível proativo controla tanto a estrutura como o conteúdo (ação – composição).

Schwier & Misanchuk (1993) fez uma complementação detalhada da interatividade baseada em 3 dimensões: Níveis (reativo, proactive, mutual), Funções (confirmação, estimulação, navegação, investigação, elaboração) e Transações (teclado, *touch screen*, rato, voz). Os *"níveis de interação são baseados na qualidade instrucional da interação"*.[10]

6.2 – Sistemas Interativos

Atualmente vivemos uma fase de explosão no que toca ao desenvolvimento da tecnologia. A evolução acontece cada vez mais rápido e é cada vez maior a oferta na área. A tecnologia apoderou-se completamente da sociedade. Já é quase impensável viver sem telemóvel ou computador. O mundo passou a estar "ligado" através da tecnologia.

Mas o que poderemos chamar de Sistema Interativo?

O termo Sistema Interativo resume todo o tipo de objecto, físico ou virtual, que aceita input por ordem de um utilizador.

A natureza desses sistemas, aplicados à área musical, pode ser de 3 tipos (QUEIROZ, Marcelo [et.Al.] Projeto MOB!LE):

1. Computacional
2. Electrónica
3. Conceitual

A primeira consiste numa abordagem baseada em sistemas e algoritmos implementados na forma de software ou no desenvolvimento de aplicações baseados em programas de programação já existentes (MaxMSP, por exemplo). (QUEIROZ, Marcelo [et.Al.] Projeto MOB!LE)

A segunda, trata essencialmente de uma vertente mais física, abordando o trabalho em hardware específico para projetos musicais interativos. (QUEIROZ, Marcelo [et.Al.] Projeto MOB!LE)

[10] Schwier, R.A. & Misanchuk, E. (1993). Interactive Multimedia Instruction. Englewood CLiffs, NJ: Educational Technology Publications.

A última dedica-se essencialmente à lógica. Remete para a criação de sistemas lógicos ou algoritmos que, normalmente são implementados na forma de programas computacionais ou usados como auxiliares de composição e performance. (QUEIROZ, Marcelo [et.Al.] Projeto MOB!LE)

A definição de Interatividade que mais se adequa ao que observamos hoje nos ditos "Sistemas Interativos" é a de Gimenes (2008), "Sistemas interativos musicais são sistemas computacionais que, através da troca de informações musicais, têm a capacidade de perceber o ambiente, analisar e tomar ações de modo a alterar os estados vizinhos, bem como alterar os seus próprios estados internos"

6.2.1 – Sistemas Interativos em Música

A Interatividade em música consiste em fazer música recorrendo a atividades paralelas. A Música por interatividade caracteriza-se por se desenvolver conforme ações que não dependem do compositor mas sim do ambiente em redor, tomando assim um caminho aberto e mais ligado à performance. Podemos até dizer que é feita em tempo-real, apesar de ter um trabalho de programação, criatividade e composição vasto.

Sistemas Interativos estão relacionados diretamente com a Computação Musical. Quando falamos de Interatividade Musical devemos ter em conta 3 aspectos:

- A troca de informações musicais.
- Capacidades cognitivas.
- Capacidade de influenciar e receber influência.

6.3 – Modelos

Pode diferenciar-se dois modelos de Sistemas Interativos (ZAPAROLLI, Mariana). Esses são:

- Instrumento
- Instrumentista

Os Sistemas de Interatividade que se enquadram no modelo de Instrumento, está associado, tanto à construção de um instrumento musical, como aos gestos executados por um instrumentista. Estes são analisados pelo computador, fazendo com que ative o método de resposta do sistema. Assemelha-se a um solo.

Os Sistemas que seguem um modelo Instrumentista pretendem "desenhar" um instrumentista artificial. Conforme ações paralelas, o sistema dispara a resposta, desempenhando um papel de dueto com o factor (sendo este humano ou não) que fez disparar a resposta.

Capítulo 7 – Estado de Arte

Muitos trabalhos e projetos foram desenvolvidos nesta área que envolve a música com a cor, luz e interatividade. Como se pode perceber nas páginas anteriores têm-se vindo a discutir este assunto já há muito tempo. Quando envolvemos tecnologia aparecem programas, instrumentos, concertos de luzes a relacionar estas duas vertentes, audição e visão de forma interativa. A variedade de oferta existente é muita, alguma fiável, outras não tão fiáveis.

7.1 – ColorChord Sound Lightning

O projeto de Charles Lohr iniciou-se após a compra de uma guitarra transparente e implementação de um dispositivo led USB na mesma. Após ter começado a testar a guitarra surgiu a hipótese de criar um algoritmo que convertesse som em cor.

Criou então o que chama de *ColorChord* , uma tecnologia que seleciona cores através do som. Utiliza a escala cromática como recurso para a análise do som, ao contrário da maior parte do software visual para música que recorre normalmente à detecção de beat.

O conceito usado no *ColorChord* é o seguinte: a primeira cor primária (Amarelo) é atribuída à nota da tonalidade em que a música que vai ser tocada está.

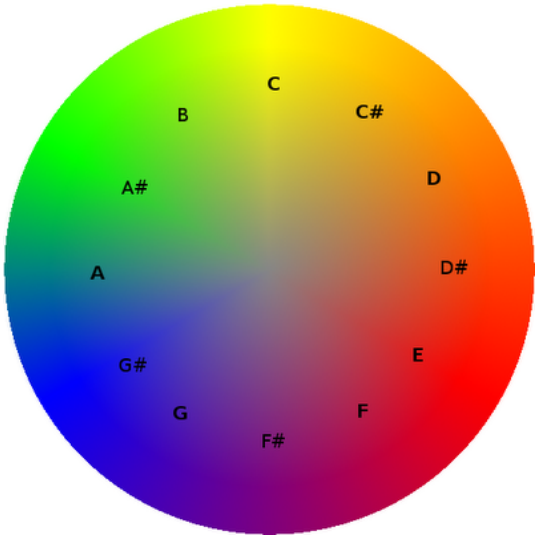


Imagem 31 – Roda de cor e nota sendo a música na tonalidade de Dó.

Usando a tonalidade de Dó Maior o primeiro acorde [Dó, Mi, Sol] será qualquer coisa como [Amarelo, Vermelho, Roxo azulado].

Podemos ver esta guitarra LED em ação nos anexos visuais 1, 2, 3 e 5. No anexo visual 4 observamos este projeto aplicado, exclusivamente, à voz.

O software é bastante complexo e necessita de condições com muito pouca latência senão não funciona corretamente. O programa processa o som que entra por um microfone, identificando o pitch e transforma-o em cor em tempo real. O projeto é caseiro e não está patenteado.

7.2 – Perfect Pitch

Perfect Pitch é um programa comercializado de treinamento de ouvido. O software associa cores aos sons. O seu criador, David Lucas Burge, estudou música desde pequeno, e sempre ouviu a típica frase "Nunca vais ser tão bom como [x]! Ele tem ouvido absoluto.

Ouvido absoluto é a capacidade que algumas pessoas têm de identificar e reproduzir com a voz as notas, os acordes, as tonalidades sem margem de erros.

David sempre se sentiu intrigado com esta capacidade que não conseguia possuir, mas um dia surgiu a ideia de ouvir naturalmente enquanto o seu cérebro começava a associar cores aos tons. A dada altura tinha adquirido ouvido absoluto através do que chama de *Color Hearing*, ou seja "ouvir" as cores.

A partir desse momento começou a afirmar que o ouvido absoluto nem sempre nascia connosco, era possível desenvolve-lo.

Em 1930, lançou o seu método de treinos *Perfect Pitch*, nos anos seguintes difundiu o seu trabalho através de uma *tour* de seminários nos quais explicava o seu conceito de *ear training* e dava lições para desenvolver o tal ouvido absoluto.

Em 30 anos, 97% das pessoas que adquiriram o método de Burge, desenvolveram ouvido absoluto.

Agora com a versão digital tornou-se muito mais fácil adquirir e utilizar o equipamento. O equipamento que inclui, 24 aulas dadas pelo David Burge em 8 CD's com um livro de mão, está disponível para encomenda no site do mesmo por 199$ sem desconto.

Imagem 32 – Publicidade ao Método de Burge.

7.3 – Reactable

A *Reactable* é um instrumento musical electrónico em que a sua suprefície é sensível ao toque e desenvolve reações conforme o mesmo.

Foi desenvolvida por Sergi Jordá, Marcos Alonso, Martin Kaltenbrunner and Günter Geiger juntamente com o grupo de Tecnologias da Música da Universitat Pompeu Fabra em Barcelona, Espanha.

O instrumento é uma mesa redonda colodada numa sala escura com objectos físicos que que se posicionam no tampo da mesa. Estes objectos (cubos, placas) representam os diferentes módulos de um sintetizador analógico, como por exemplo, VCO, LFO e VCF.

O display é a mesa em si. O cubo é colocado sobre a mesa e aparecem no display vários simbolos que representam formas de onda, ciclos, etc etc.

A sua estrutura é composta por uma câmara de vídeo por baixo da mesa juntamente com um projetor. Ambos estão conectados a um computador e estão programados em Pure Data e SuperCollider. Os objectos são lidos pela câmara e processados.

Este intrumento já foi desenvolvido para iPad, iPhone e iPod Touch e encontra-se disponível em duas versões, Reactable Live e Reactable experience.

Imagem 33 – Reactable.

Imagem 34 – Reactable.

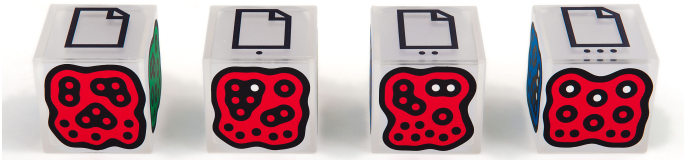


Imagem 35 – Blocos.

7.4 – Bucephalus

Bucephalus é uma aplicação para iOS desenvolvida pela empresa *Refined Stochastic Technology*, baseada na física do som. A versão 2.0 apresenta um sintetizador granular sob o mesmo design utilizando uma biblioteca e samples personalizados. Possuí também efeitos sonoros como por exemplo 3 reverbs, 3 filtros, distorção, compressão e saturação. (PAPERT, Jonathan)

Características:

- 5 tamanhos de bolas controlados globalmente;
- Precisão;
- Adicionar até 20 bolas para serem reproduzidas em simultâneo;
- Samples de Alta qualidade de *resonance*;
- Rigidez e Ressonância variáveis;
- Elasticidade e gravidade variáveis;
- Possibilidade de gravar os estados;
- Audiobus – conectar a outros dispositivos áudio em tempo real;
- Capacidade de alterar a afinação das notas em mais de 10 oitavas;

Foi desenvolvido para iPad e iPhone. (SYNTHEAD, 2013 | PAPERT, Jonathan, 2013)

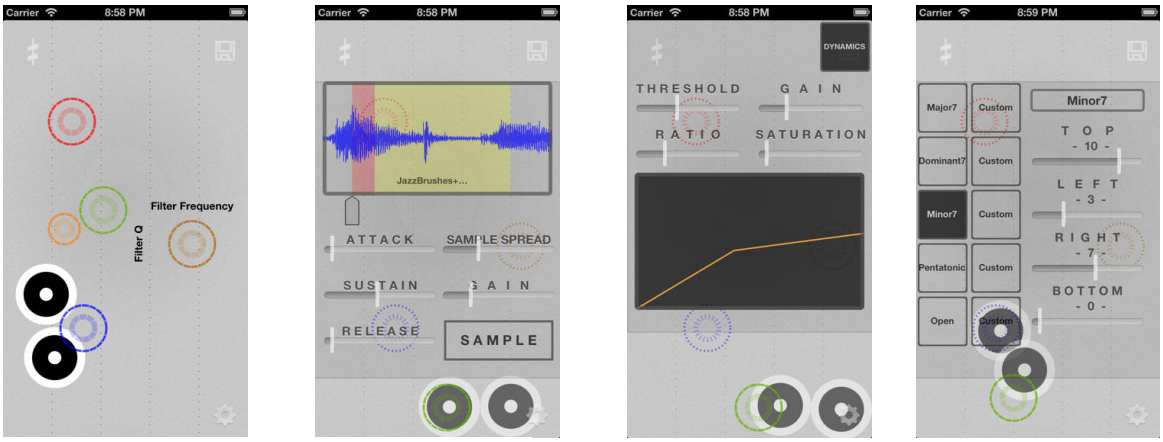


Imagem 36 - *Bucephalus* para iPhone.

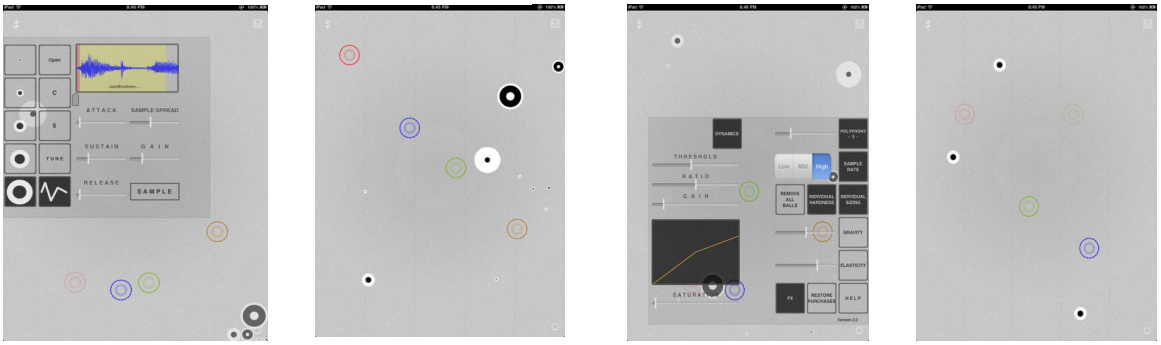


Imagem 37 – *Bucephalus* para iPad.

Capítulo 8 – Fase de Concepção

8.1 – Aplicação Interativa

A aplicação interativa desenvolvida no decorrer do 2º semestre tende a ser uma aplicação mais virada para o lado artístico criando uma experiência performativa e/ou lúdica ao utilizador. Possui um interface intuitivo que funciona como um instrumento virtual, controlando parâmetros de um sintetizador virtual.

8.2 – Elaboração

A elaboração desta aplicação percorreu várias etapas. Começando pela elaboração da interface e só depois o desenvolvimento musical. Para desenvolver esta aplicação foi necessária uma pesquisa dos recursos a utilizar.

Primeiramente a ideia foi exposta ao Orientador de projeto, prof. Rui Dias, que me indicou que a melhor forma de fazer a interface seria em Processing e o processamento de som em MaxMSP, explicando também que bastava depois fazer a comunicação entre um programa e outro através da biblioteca oscP5.

8.2.1 - Interface

O desenvolvimento da interface passou por várias fases:

- Protótipo;
- Definição do objecto;
- Programação em Processing;

8.2.1.1 - Protótipo

Quando a ideia de fazer esta aplicação surgiu, rapidamente foi criado um protótipo visual do que seria a aplicação em funcionamento.

Este protótipo recorria ao sistema de cores de Zieverink aplicado nas bolas representativas da nota alinhadas à direita por cima de um esquema de um teclado de piano. As bolas alinhadas na parte superior da tela representavam os

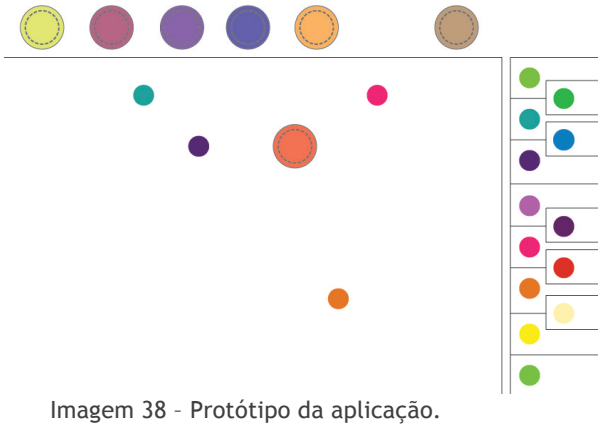


Imagem 38 – Protótipo da aplicação.

efeitos sonoros, como por exemplo, reverb, delay, equalizações.

As bolas correspondentes às notas, ao serem movidas, gerariam o som da nota respectiva. Quando a ideia foi apresentada ainda não estava definida a forma como seria essa reprodução da nota, havia várias propostas:

- A tela poderia representar uma *timeline* que seria reproduzida da esquerda para a direita em loop lendo a informação que lá poderia estar.
- A bola ser depositada dentro da tela e mover-se aleatoriamente, reproduzindo a nota correspondente quando tocasse as paredes da tela.
- A bola ser colocada na tela e ter um metro que a reproduzia, por exemplo, a nota ser reproduzida nos tempos fortes de um quaternário.

Quanto aos efeitos, pensou-se em serem aplicados individualmente a cada nota ou em todas as que estivessem dentro da tela a serem reproduzidas.

8.2.1.2 - Definição do objecto

A fase de definição do objecto não foi mais do que um estudo sobre o que iria constar na interface.

Pegou-se assim no protótipo concecionado e confirmou-se o que já estava determinado: as bolas alinhadas num teclado posicionado do lado direito da tela iriam representar as notas. Já as bolas acima da tela, seriam substituídas por quadrados que seriam os efeitos. Os quadrados devidamente identificados com uma tira de texto por cima dos mesmos. Só precisaríamos então de construir a tela.

Inicialmente, era suposto a aplicação corresponder aos objectivos propostos e só depois avançar para os objectivos bónus.

Com o desenvolver do projeto foi sendo possível aumentar o número de características da aplicação. Foram então introduzidos, um menu em que o utilizador pode escolher o instrumento que quer ouvir e outro em que pode optar pela proposta de cor que mais lhe agrada. Tornando, assim, a aplicação mais interativa.

Foi também adicionado um botão de reset que reiniciaria todos os valores.

Decidiu-se então que as bolas e os quadrados funcionariam através de *drag and drop* e poderiam ser eliminados individualmente apenas com um duplo clique sobre si mesmos.

Numa primeira versão, apenas poderíamos ter uma bola de cada nota, agora podemos ter mais que uma, ou seja, se antes só podíamos ter um dó, agora é-nos possível ter vários dós.

As linhas que definem a tela definem o espaço em que obtemos a resposta interativa. Se as bolas forem posicionadas fora da tela, são automaticamente eliminadas. Os efeitos deixam de estar ativos.

8.2.1.3 – Programação em Processing

Processing é uma linguagem de programação, um ambiente de desenvolvimento e uma comunidade *online*. Desde 2001, Processing tem promovido instrução na área do programa aplicado às artes visuais e tecnológicas. Hoje em dia, é usado por estudantes, artistas, designers, insvestigadores e amadores para instrução, criação de protótipos, ambientes gráficos e desenvolvimento de software. (PROCESSING.ORG)

Este programa foi utilizado para programar todo a interface da aplicação.

O processo de construção da interface interativo começou pelos grafismos e só depois foi desenvolvida a função dos mesmos, isso impediu que se percebesse certas coisas logo desde inicio. O primeiro esboço do código criava as bolas individualmente. O segundo já construía as bolas recorrendo a uma classe para que fosse possível arrastar as bolas individualmente sem que estas interferissem umas com as outras.

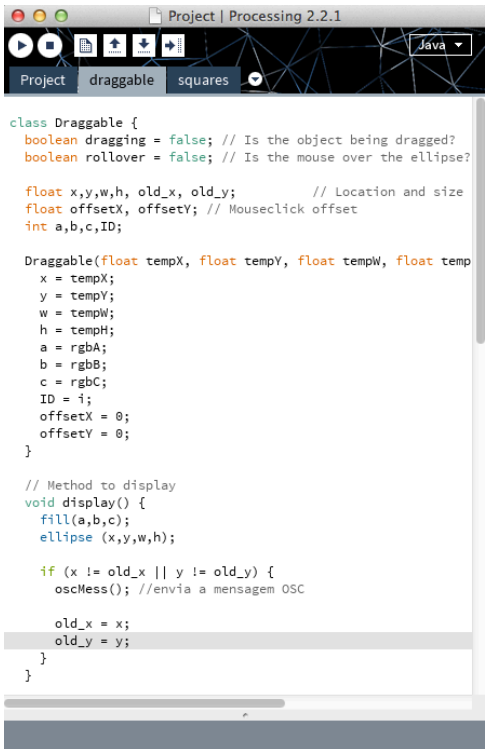


Imagem 39 – Print screen do separador da class Draggable.

Como podemos observar na imagem à esquerda, a *class* Draggable é criada num separador novo. O método *display* cria a bola conforme os dados recebidos. Esses dados são enviados quando para a *class* quando, no separador *main* (Project), esta é chamada.

No *main* temos um *array* bolas que cria 13 Draggable's. Essas 13 equivalem ao número de bolas que queremos criar.

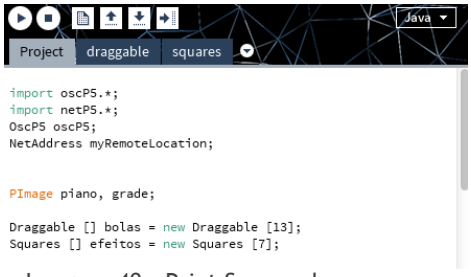
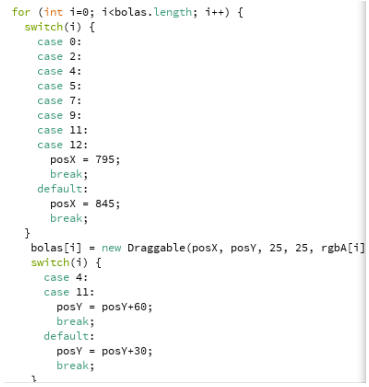


Imagem 40 – Print Screen do array.

O ciclo *for* percorre o *array* bolas criando as mesmas nas posições definidas no código. Neste caso temos um switch para que as bolas fiquem colocadas nas posições corretas, definindo primeiro o x e depois o y.



Todo este processo acontece igualmente com a *class* squares, que cria os quadrados dos efeitos.

Começou-se com este código pois era suficiente ter uma bola para cada nota, com o passar do tempo, o desenvolvimento do projeto tornou-se premissa, e tentou-se arranjar uma forma de dar possibilidade ao utilizador de ter mais bolas de uma só nota.

Imagem 41 – Print Screen do ciclo *for*.

O projeto foi-se desenvolvendo a passo e passo. Passou por várias fases e o código inicial acabou por ser praticamente esquecido, dando origem a um código e abordagem diferente.

A interface estabeleceu-se da seguinte forma:

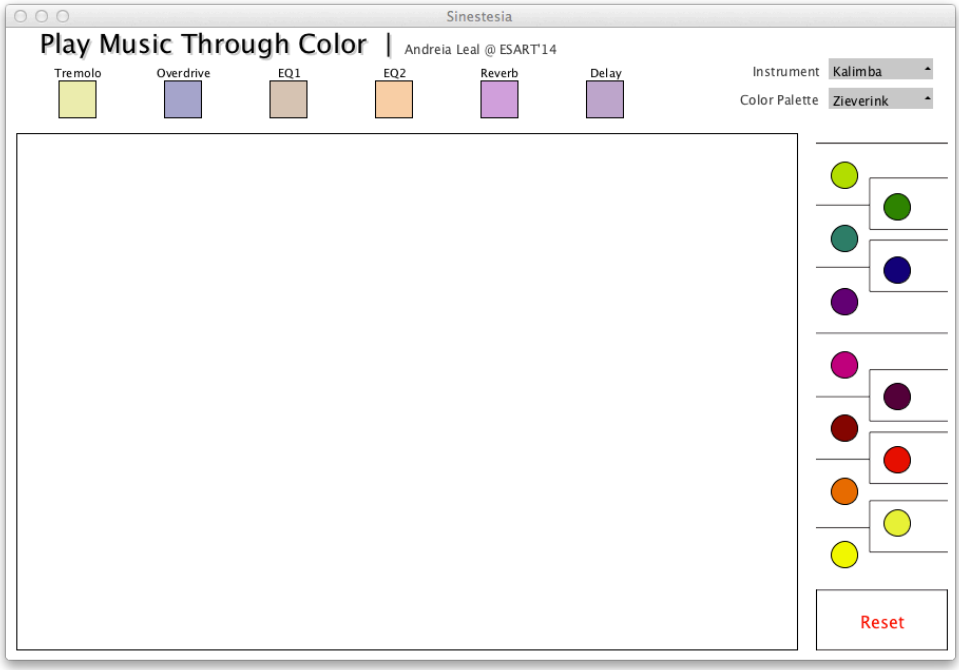


Imagem 42 – Interface Final da Aplicação.

Do lado direito, podemos observar um teclado sinestésico disposto na vertical. As bolas coloridas funcionam como botões que, quando clicados, criam uma bola nova.

Foi criada uma *class* circularButton que constrói um objecto, adicionado à *array* bolasBotoes, com x, y, base (w), altura (h), as 3 propriedades RGB (a, b, c) e o ID que corresponde à nota.

Sinestesia | circularButton | circularDraggable | squaredDraggable

```
class circularButton {
  int x, y, w, h; // Location and size
  int a, b, c, ID; // Color (a,b,c) and ID

    circularButton(int tempX, int tempY, int tempW, int tempH, int rgbA, int rgbB, int rgbC, int tempID) {
    x = tempX;
    y = tempY;
    w = tempW;
    h = tempH;
    a = rgbA;
    b = rgbB;
    c = rgbC;
    ID = tempID;
  }
```

Imagem 43 – Construtor dos botões. (Tab circularButton)

Para funcionar como botão são chamadas as funções *mousePressed()* e *mouseReleased()* presentes no *main* (tab Sinestesia). Se verificar a condição de ser um ponto no círculo (teste lógico[11] na função *isMouseOverCircularButton*) adiciona um objecto (bola) da *class* circularDraggable com as mesmas propriedades (x, y, w, h, a, b, c, ID) e com o valor do incrementador *bolasTotal.* Os objectos da *class* circularDraggable são adicionados à *arrayList*[12].

```
void isMouseOverCircularButton(int mx, int my) {
  float disX = x - mouseX;
  float disY = y - mouseY;
  if (sqrt(sq(disX) + sq(disY)) < w/2) {
    bolasItems.add(new circularDraggable(x, y, 25, 25, a, b, c, ID, bolasTotal));
  }
 println (bolasTotal);
}
```

Imagem 44 - Função *isMouseOverCircularButton*. (Tab circularButton)

```
circularDraggable(float tempX, float tempY, float tempW, float tempH,
int rgbA, int rgbB, int rgbC, int i, int j) {
  x = tempX;
  y = tempY;
  w = tempW;
  h = tempH;
  a = rgbA;
  b = rgbB;
  c = rgbC;
  IDnota = i;
  IDarray = j;
  offsetX = 0;
  offsetY = 0;
  bolasTotal ++;
}
```

Imagem 45 - Construtor do objecto bola. (Tab circularDraggable)

De seguida, a posição das bolas é verificada ao ser chamada na função *mouseReleased().* Se a posição estiver fora dos limites da tela de ação, as bolas terão de ser removidas.

```
//se as bolas forem colocadas fora da tela, são removidas.
void checkPosition() {
  if (x < 10 || x > 750) {
    for (int i = bolasItems.size ()-1; i >= 0; i--) {
      circularDraggable  testeDrag = bolasItems.get(i);
      if ( testeDrag.getIDarray() == IDarray) {
        bolasItems.remove(i);
      }
    }
  }
  if (y < 100 || y > 590) {
    for (int i = bolasItems.size ()-1; i >= 0; i--) {
      circularDraggable  testeDrag = bolasItems.get(i);
      if ( testeDrag.getIDarray() == IDarray) {
        bolasItems.remove(i);
      }
    }
  }
}
```

Imagem 46 - Verificador de posição do sítio onde a bola é deixada. (Tab circularDraggable)

```
  //se os quadrados forem colocadas fora da tela, assumem posiçao original.
void checkPosition() {
  if (x < 10 || x > 750) {
    for (int i=0; i<quadradosItems.length; i++) {
      squaredDraggable  testeDrag = quadradosItems[i];
      if ( testeDrag.getIDefeito() == IDefeito) {
        originalPos();
      }
    }
  }
  if (y < 100 || y > 590) {
     for (int i=0; i<quadradosItems.length; i++) {
      squaredDraggable  testeDrag = quadradosItems[i];
      if ( testeDrag.getIDefeito() == IDefeito) {
        originalPos();
      }
    }
  }
}
```

Imagem 47 - Verificador de posição do sítio onde o quadrado é deixado. (Tab squaredDraggable)

[11] Condição *if.*

[12] *ArrayList* é uma *array* de tamanho variável, com ou sem tamanho inicial.

Para um melhor funcionamento na interação do utilizador com a interface, foi implementado o método *isMouseOverCircularDraggableDoubleClick()* que, após o teste lógico da posição do rato pressionado duas vezes (duplo clique), remove a bola da *arrayList*. Esta operação é realizada iterando a *arrayList* de forma retrogradada. Quando o ID for igual ao da bola pressionada, a mesma é eliminada.

```
// Apagar bola || Duplo Clique
void isMouseOverCircularDraggableDoubleClick(int mx, int my) {
  float disX = x - mouseX;
  float disY = y - mouseY;
  if (sqrt(sq(disX) + sq(disY)) < w/2) {
    for (int i = bolasItems.size ()-1; i >= 0; i--) {
      circularDraggable  testeDrag = bolasItems.get(i);
      if ( testeDrag.getIDarray() == IDarray) {
        bolasItems.remove(i);
        oscRem();
      }//if
    }//for
  }//if
}//isMouseOverCircularDraggableDoubleClick
```

Imagem 48 - Função *isMouseOverCircularDraggableDoubleClick()*. (Tab circularDraggable)

Na interface, os objectos que controlam os efeitos (quadrados) são criados segundo a *class* squaredDraggable e adicionados à *array* quadradosItems (declarada no *main*). Para a definição das cores foi o modo RGBA, sendo A o valor de alpha (opacidade). O construtor tem esta aparência:

```
squaredDraggable(float tempX, float tempY, float tempW, float tempH,
int rgbA, int rgbB, int rgbC, int rgbAlpha, int i) {
  x = tempX;
  orX = tempX;
  y = tempY;
  orY = tempY;
  w = tempW;
  h = tempH;
  a = rgbA;
  b = rgbB;
  c = rgbC;
  alpha = rgbAlpha;
  IDefeito = i;
  offsetX = 0;
  offsetY = 0;
}
```

Imagem 49 - Construtor dos quadrados. (Tab squaredDraggable)

Os métodos de verificação da posição são os mesmos aplicados às *classes* referidas anteriormente, alterando o teste lógico para corresponder a quadrados. Neste caso, a interação de duplo clique retorna o objecto à posição inicial.

```
//Método para retornar o rectângulo à posição inicial || Duplo clique
void isMouseOverSquaredDraggableDoubleClick(int mx, int my) {
  if (mx > x && mx < x + w && my > y && my < y + h) {
    originalPos();
  }
}
void originalPos() {
  dragging = false;
  x = orX;
  y = orY;
}
```

Imagem 50 - Função *isMouseOverSquaredDraggableDoubleClick*. Função *originalPos()*. (Tab squaredDraggable)

As ações interativas ficam concluídas com o processo de arrastar os objectos. Às *classes* circularDraggable e squaredDraggable são aplicadas as propriedades de arrastamento com os seguintes métodos:

```
// O rato está sobre a bola?
void isMouseOverCircularDraggable(int mx, int my) {
  float disX = x - mouseX;
  float disY = y - mouseY;
  if (sqrt(sq(disX) + sq(disY)) < 50/2 ) {
    dragging = true;
    // Se sim, manter o controlo da localização relativa
    offsetX = x-mx;
    offsetY = y-my;
  }
}
// Stop dragging
void stopDragging() {
  dragging = false;
}
// Arrastar bola
void drag(int mx, int my) {
  if (dragging) {
    x = mx + offsetX;
    y = my + offsetY;
  }
}
```

Imagem 51 - Métodos de arrastamento da bola. (Tab circularDraggable)

```
//Condição: o rato está em cima do quadrado?
void isMouseOverSquaredDraggable(int mx, int my) {
  if (mx > x && mx < x + w && my > y && my < y + h) {
    dragging = true;
    //Se sim, manter o controlo da localização relati
    offsetX = x-mx;
    offsetY = y-my;
  }
}
// Stop dragging
void stopDragging() {
  dragging = false;
}
// Arrastar
void drag(int mx, int my) {
  if (dragging) {
    x = mx + offsetX;
    y = my + offsetY;
  }
}
```

Imagem 52 - Métodos de arrastamento do quadrado. (Tab squaredDraggable)

O primeiro método faz o teste lógico da posição do rato em relação à bola. O *dragging* é iniciado como *boolean true* pois o objecto é instanciado com o rato premido. O *stopDragging()* é chamado no *mouseReleased()* (tab Sinestesia). O método *drag()* rastreia o trajecto do rato (enquanto pressionado) e aplica-o à bola.

Quando surgiu a ideia de dar possibilidade ao utilizador de optar pela teoria cor-nota que quisesse, avançou-se para uma fase de pesquisa de recursos para criação de menus em Processing.

Encontrou-se forma de o fazer recorrendo à biblioteca criada por Andreas Schlegel, controlP5. Esta biblioteca é um GUI (Graphical User Interface) para Processing que permite a criação de controladores básicos como Sliders, Botões, Toggles, Knobs, Caixas de Texto, Menus, entre outros. (SCHLEGEL, Andreas)

A biblioteca é importada para o projeto no *main* (tab Sinestesia) da seguinte forma:

```
import controlP5.*;
ControlP5 P5Controller;
DropdownList Propostas, Synth;
```

Imagem 53 - Importação da biblioteca controlP5 e declaração *DropdownList*. (Tab Sinestesia)

Como podemos observar na imagem à esquerda, são criadas duas *DropdownList*[13] (Propostas e Synth).

[13] Menus.

```
P5Controller = new ControlP5(this);
P5Controller.setControlFont(pfont, 12);

Propostas = P5Controller.addDropdownList("Propos").setPosition(780, 78);
Propostas.addItem("Newton", 1);
Propostas.addItem("Castel", 2);
Propostas.addItem("Field", 3);
Propostas.addItem("Jameson", 4);
Propostas.addItem("Seeman", 5);
Propostas.addItem("Rimington", 6);
Propostas.addItem("Bishop", 7);
Propostas.addItem("Helmholtz", 8);
Propostas.addItem("Scriabin", 9);
Propostas.addItem("Zieverink", 10);
Propostas.addItem("Harbisson", 11);
Propostas.setValue(10);
Propostas.enableCollapse();
Propostas.setItemHeight(20);
Propostas.setBarHeight(20);
Propostas.setBackgroundColor(color(150));
Propostas.setColorBackground(color(200));
Propostas.setColorForeground(color(150));
Propostas.setColorLabel(0);
Propostas.setHeight(500);
Propostas.toUpperCase(false);
Propostas.captionLabel().style().marginTop = 4;
Propostas.actAsPulldownMenu(true);
```

Imagem 54 - Definição dos Items do Menu e estilização do mesmo. (Tab Sinestesia)

Para criar o menu foram definidos os items que o iriam compor. O restante código estiliza o menu.

Para desencadear uma reação sempre que o valor do menu for alterado, foi chamado o *listener* incluído na biblioteca controlP5 (*controlEvent)*.

```
void controlEvent(ControlEvent theEvent) {

  if (theEvent.isGroup() && theEvent.name().equals("Propos")) {
    proposal = (int)theEvent.group().value();
    handleEvent();
  }
  if (theEvent.isGroup() && theEvent.name().equals("Sound")) {
    synth = (int)theEvent.group().value();
    handleEventSynth();
  }
}
```

Imagem 55 - *ControlEvent* dos dois menus. (Tab Sinestesia)

Este invoca a função *handleEvent()* que altera as cores das bolas e dos botões definidas na tab cores.

Caso o utilizador selecione a teoria correspondente a Newton ou Field (*proposal 1* e *3*), cujas propostas não incluem as notas cromáticas, as bolas referentes às mesmas terão de ser lidadas. Os botões correspondentes a essas notas ficam brancos e a função *isMouseOverCircularButton()* não é chamada. Os objectos da *class* circularDraggable são removidos (a *arrayList* é iterada de forma retrogradada e caso a nota de origem seja cromática a bola é removida).

```
void handleEvent() {
  for (int i=0; i<12; i++) {
    bolasBotoes[i].reFill(proposal);
  }
  for (int i=0; i<bolasItems.size (); i++) {
    bolasItems.get(i).reFill(proposal);

  }

  /*condição: se for a proposta 1 ou 3, ou seja,
   newton ou field, as bolas cromáticas posicionadas na tela
   são removidas */
  if (proposal == 1 || proposal == 3) {
    for (int i = bolasItems.size ()-1; i >= 0; i--) {
      circularDraggable  testeDrag = bolasItems.get(i);
      if (testeDrag.getIDnota() == 1 || testeDrag.getIDnota() == 3 || testeDrag.getIDnota() == 6
      || testeDrag.getIDnota() == 8 || testeDrag.getIDnota() == 10) {
        int tempIDA = testeDrag.getIDarray();
        bolasItems.remove(i);
        oscRem(tempIDA);
      }//if
    }//for
  }
 }
```

Imagem 56 - Função *handleEvent*. Condição das propostas 1 e 3. (Tab Sinestesia)

```
  void reFill(int proposal) {
    switch(proposal){
      case 1:
      a=newtonA[ID];
      b=newtonB[ID];
      c=newtonC[ID];
      break;
      case 2:
      a=castelA[ID];
      //(...) extrato
      c=scriabinC[ID];
      break;
      case 11:
      a=harbissonA[ID];
      b=harbissonB[ID];
      c=harbissonC[ID];
      break;
      default:
      a=zieverinkA[ID];
      b=zieverinkB[ID];
      c=zieverinkC[ID];
      break;
    }
 }
```

Imagem 57 - Função *reFill*. Altera as cores conforme a proposta escolhida. (Tab circularDraggable)

No caso do menu relativo aos instrumentos, o método que reage à mudança de valor envia uma mensagem OSC[14] para o MaxMSP com o valor do Item escolhido pelo utilizador.

```
void handleEventSynth() {
  returnValues();
  oscMessSynth();

}
```

Imagem 58 - Método *handleEventSynth()*. Envio do item.

[14] Definição e explicação das mensagens OSC no Capítulo 8.2.2 (Capítulo seguinte).

Após um *beta-test*, foi apurada a necessidade de uma solução para repor os valores de origem. Desta forma, foi criado um botão "RESET" por baixo do teclado. O teste lógico verifica a posição do rato em relação ao botão. Se este se demonstrar verdadeiro:

- As *dropdownList* retornam aos valores predefinidos;
- A *arrayList* é percorrida inversamente e todos os objectos são removidos;
- É chamada a função *originalPos()* que retorna os quadrados presentes na tela à posição original;
- O incrementador *bolasTotal* é zerado (retoma valor inicial, 0);
- A mensagem OSC de Reset é enviada para o MaxMSP.

```
if (mouseX > 768 && mouseX < 892 && mouseY > 533 && mouseY < 590) {
  Propostas.setValue(10); //volta às cores iniciais
  Synth.setValue(13);
  for (int i = bolasItems.size ()-1; i >= 0; i--) {
    bolasItems.remove(i); //elimina as bolas a partir da ultima a ser criada
  }
  for (int i=0; i<quadradosItems.length; i++) {
    quadradosItems[i].originalPos(); //quadrados vão para a posição inicial
  }//for
  oscReset();
  bolasTotal = 0;
 }//if
}
```

Imagem 59 – Teste lógico do botão "RESET". (Tab Sinestesia)

```
void originalPos() {
  dragging = false;
  x = orX;
  y = orY;
}
```

Imagem 60 – Função *originalPos()*. (Tab squaredDraggable)

Com outro *beta-test* percebeu-se que, quando havia certos efeitos posicionados na tela de ação e o instrumento era alterado na *dropDownList*, o programa deixava de responder. Para solucionar esse problema determinou-se que, cada vez que o utilizador alterasse o instrumento, o programa faria um *return* dos valores dos quadrados. Criou-se assim a função *returnValues()* onde os quadrados regressavam à posição original.

```
void returnValues() {
  for (int i=0; i<quadradosItems.length; i++) {
    quadradosItems[i].originalPos(); //quadrados vão para a posição
  }//for
}
```

Imagem 61 – Função *returnValues()*. (Tab Sinestesia)

8.2.2 - Comunicação OSC

Para fazer a comunicação entre a interface (Processing) e o conteúdo (MaxMSP), foi necessário utilizar uma biblioteca, o oscP5.

OscP5 foi criada por Andreas Schlegel, entre 2004 e 2012. Trata-se de uma biblioteca para o Processing, que o permite comunicar com outro hardware ou software usando o protocolo OSC (Open Sound Control). (SCHLEGEL, Andreas)

O protocolo OSC (Open Sound Control) foi desenvolvido na UC Berkeley e é utilizado para comunicação entre computadores, sintetizadores, e outros dispositivos.

Nesta aplicação, esta biblioteca foi utilizada para o envio de mensagens para o patch de MaxMSP onde a mesma era descodificada de forma a enviar o que era necessário para podermos ter um feedback auditivo.

Assim, enquanto o utilizador arrasta bolas e quadrados na interface, esses valores estão a ser passados através de mensagens OSC, chegando ao patch onde essas informações estão a ser convertidas para um feedback auditivo.

```
// Method to display
void display() {
  fill(a, b, c);
  ellipse (x, y, w, h);
  if (x != old_x || y != old_y) {
    oscMess(); //envia a mensagem OSC
    old_x = x;
    old_y = y;
  }
}
```

Imagem 62 - Envio da mensagem *oscMess()*. (Tab circularDraggable)

Tanto na *class* circularDraggable como na *class* squaresDraggable, a mensagem é passada como vemos na imagem à esquerda. A função *oscMess()* é chamada quando uma bola ou um quadrado muda de posição, ou seja, como podemos observar, temos uma condição que confirma se o *x* é diferente do *old_x,* se for, chama a função *oscMess()* e o *old_x* assume o valor de x (o mesmo acontece com o *y*).

A função chamada (*oscMess()*) é declarada em cada *class* como podemos observar nas imagens que se seguem. E a mensagem enviada para o MaxMSP contém:

- "/nota" ou "/efeito"
- IDnota, ou seja, se é dó, ré, mi, etc.
- IDarray, ou seja, número da bola.
- X, posição em x.
- Y, posição em y.

```
void oscMess() {
  OscMessage notaMess = new OscMessage("/nota");
  notaMess.add(IDnota);
  notaMess.add(IDarray);
  notaMess.add(x);
  notaMess.add(y);
  pos.send(notaMess, myRemoteLocation);
}
```

Imagem 63 - Função *oscMess()* para notas. (Tab circularDraggable)

```
//mensagem para o MaxMSP
 void oscMess() {
  OscMessage notaMess = new OscMessage("/efeito");
  notaMess.add(IDefeito);
  notaMess.add(0);
  notaMess.add(x);
  notaMess.add(y);
  pos.send(notaMess, myRemoteLocation);
 }
}
```

Imagem 64 - Função *oscMess()* para os quadrados. (Tab squaredDraggable)

Para além desta mensagem, temos outra para quando uma bola é removida.

A função *oscRem()* é chamada sempre que fazemos um *remove* de uma bola, através de um duplo clique, e envia para o MaxMSP uma mensagem que contém:

- "/remover";
- IDnota, ou seja, se é dó, ré, mi, etc.;
- IDarray, ou seja, número da bola eliminada;
- 0, pois assim a bola pára de reproduzir;
- 0, pois assim a bola pára de reproduzir.

```
  void oscRem() {
    OscMessage notaRem = new OscMessage("/remover");
    notaRem.add(IDnota);
    notaRem.add(IDarray);
    notaRem.add(0);
    notaRem.add(0);
    remove.send(notaRem, myRemoteLocation);
  }
}
```

Imagem 65 - Função *oscRem()*. (Tab circularDraggable)

No caso da proposta de cor selecionada pelo utilizador não conter notas cromáticas, a mensagem enviada contém:

- "/remover";
- 0, valor indefinido, não precisa de ser enviado mas é enviado para simplificar a descodificação da mensagem em MaxMSP;
- i, instância da nota que é eliminada;
- 0, pois assim a bola pára de reproduzir;
- 0, pois assim a bola pára de reproduzir.

```
void oscRem(int i) {
  OscMessage notaRem = new OscMessage("/remover");
  notaRem.add(0);
  notaRem.add(i);
  notaRem.add(0);
  notaRem.add(0);
  remove.send(notaRem, myRemoteLocation);
}
```

Imagem 66 - Função *oscRem()*. (Tab Sinestesia)

Não há necessidade de recorrer a este método para eliminar quadrados, pois, voltando a referir, não são eliminados mas sim repostos na posição original. Quando há uma alteração da posição, a mensagem *oscMess()* entra em ação. Assim sendo, ao regressar um quadrado à posição incial, passa para o MaxMSP a mensagem que contém as novas posições, parando este de funcionar pois está fora da tela de ação.

À necessidade de criar um botão de reset, aliou-se a necessidade de fazer com que a mensagem de que o reset aconteceu fosse passada para o MaxMSP.

Assim sendo, foi criada a função *oscReset()* que envia para o MaxMSP uma mensagem com os seguintes parâmetros:

- "/remover"
- 0, valor indefinido, não precisa de ser enviado mas é enviado para simplificar a descodificação da mensagem em MaxMSP;
- -1, valor definido, no MaxMSP este segundo valor, referente à instância, é passado por uma soma (+1) antes de ser conduzido para o poly. Sendo -1, vai tornar-se 0 após ser somado. A instância 0 do poly, envia as mensagens para todas as vozes;
- 0, para que as vozes sejam paradas;
- 0, para que as vozes sejam paradas.

Quando surgiu a possibilidade de implementar um menu que desse possibilidade ao utilizador de escolher o instrumento, teve de ser criada uma mensagem que enviasse o número referente ao Item escolhido para o MaxMSP para este descodificar

e activar o instrumento no vst. Foi então criada a função *oscMessSynth()* que envia os seguintes parâmetros:

- "/synth"
- número do Item selecionado pelo utilizador.

```
void oscMessSynth() {
  OscMessage notaMessSynth = new OscMessage("/synth");
  notaMessSynth.add(synth);
  syn.send(notaMessSynth, myRemoteLocation);
}
```

Imagem 65 - Função *oscMessSynth()*.

Para estas funções funcionarem, têm de ser declaradas no *main* do projeto em Processing.

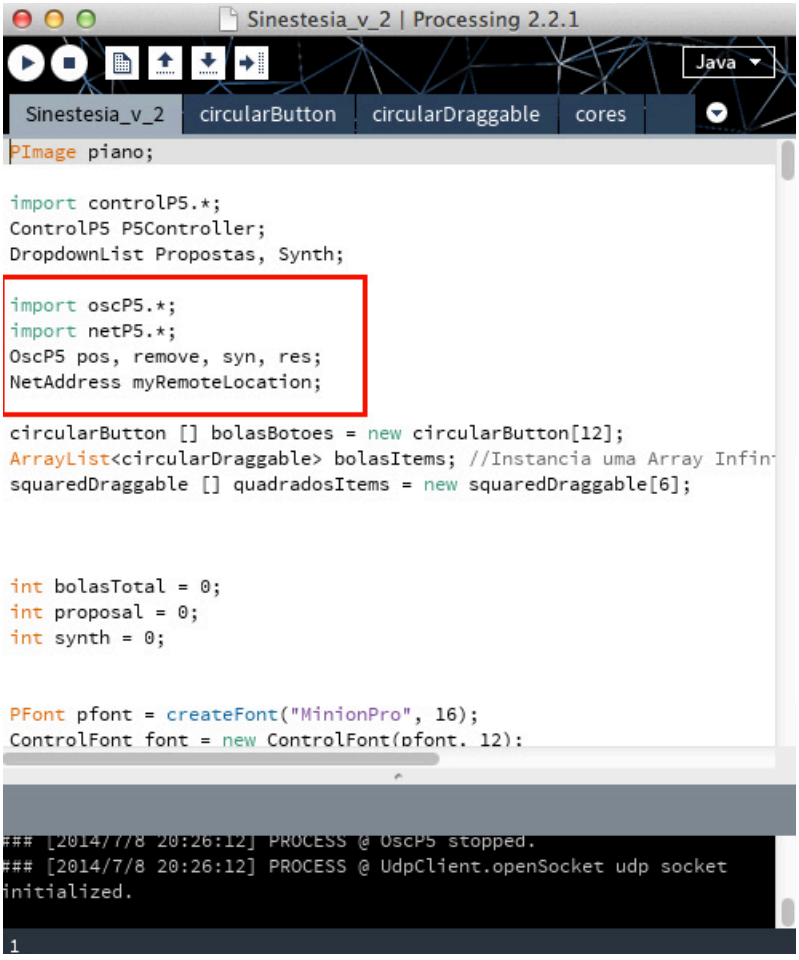


Imagem 67 - Importação da biblioteca oscP5.

À esquerda temos uma imagem que nos mostra como se fez a importação da biblioteca, declaração de *pos* e *remove* e declaração do nome do endereço pela qual serão enviadas as mensagens.

Em baixo, no *setup* do *main* a inicialização de *pos* e *remove* e da localização para envio da mensagem, o endereço.

```
void setup() {
  size(900, 600);
  smooth();

  pos = new OscP5(this, 7400);
  remove = new OscP5(this, 7400);
  syn = new OscP5(this, 7400);
  res = new OscP5(this, 7400);
  myRemoteLocation = new NetAddress("127.0.0.1", 7400);
```

Imagem 68 - Inicialização.

O MaxMSP recebe as mensagens a partir da porta 7400.

Imagem 69 - Recepção de mensagem no MaxMSP.

8.2.3 - Programação em MaxMSP

Depois da comunicação feita, é necessário descodificar todos os parâmetros da mensagem. Esse processo é feito em MaxMSP e a partir daí todas as mensagens vão ser analisadas e convertidas para obtermos o feedback desejado.

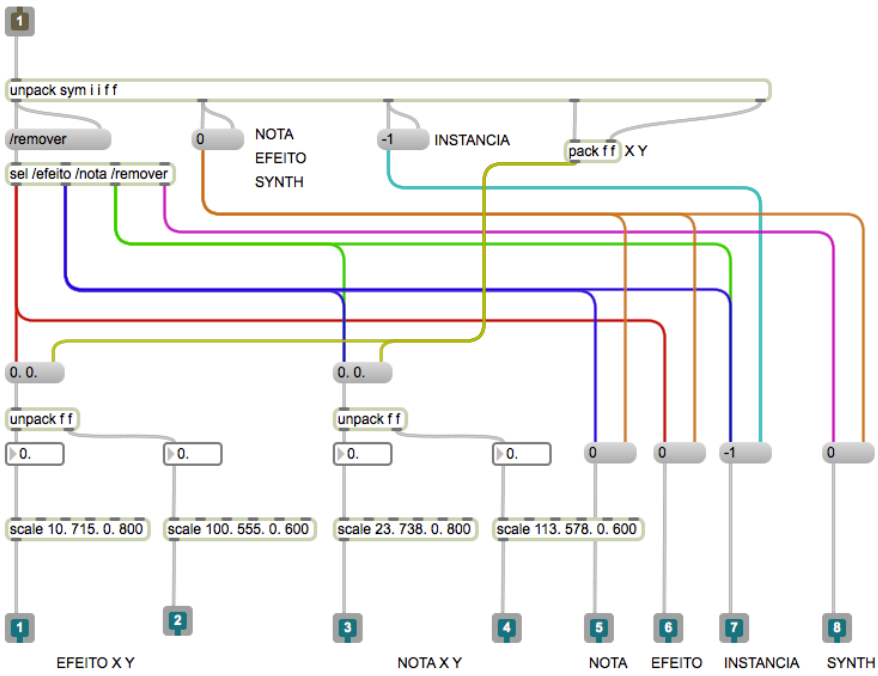


Imagem 70 – Sub-patch "p information".

Numa primeira fase, a informação chega em forma de mensagem. Essa mensagem é "partida" fazendo um unpack da mesma, obtendo assim os seguintes parâmetros:

- tipo de mensagem (efeito, nota, remove ou synth);
- número do efeito ou nota ou synth;
- número da instância (número da bola);
- x ;
- y;

O *x* e o *y* são reunidos novamente num pack para facilitar a compreensão do patch.

Sempre que recebemos valores novos, as caixas de mensagem colocadas em cima são atualizadas. A caixa que nos mostra se o objecto arrastado na interface é um quadrado ou uma bola (correspondentes a efeito ou nota, respectivamente) ou se fez *remove*, é conectada a um sel que compara a mensagem recebida com a informação que tem na caixa. Se for verdadeira envia um bang, ou seja, se for */efeito* envia bang pelo primeiro outlet, se for */nota* envia um bang pelo segundo outlet, se for */remove* envia bang pelo terceiro outlet, se for /synth envia bang pelo quarto outlet, pois não é igual a nenhuma mensagem declarada na caixa.

A mensagem com os valores x e y é recebida e, conforme /efeito ou /nota, sujeita a um unpack dando origem a dois valores *float* consequentemente transformados através de um *scale*. Este *scale* limita a tela de ação do interface redimensionando-a

para valores mais fáceis de trabalhar. Assim, para as bolas, o eixo do *x* correspondente à tela, na interface, vai do ponto 10 ao ponto 715, aqui o 10 passa a chamar-se 0 e o 715 passa a 800. O mesmo tipo de conversão é feito no eixo do *y*: o 23 passa a 0 e o 555 a 60. Para os quadrados, a tela de ação (como no código da interface estão a alinhar ao centro) tem como limites no eixo do *x* 23 e 738, e no eixo do *y* 113 e 578. Estes valores também são redimensionados para 0, 800 e 0, 600.

A mensagem que recebe o número da nota, efeito, remover ou Synth, é roteada para 3 outras caixas de mensagem que serão disparadas pelo bang disparado pelo *sel.*

A instância é referente apenas à bola por isso só é disparada quando o *sel* recebe /nota ou /remover.

Assim sendo as mensagens finais são enviadas pelos outlets e passamos novamente para o patch principal.

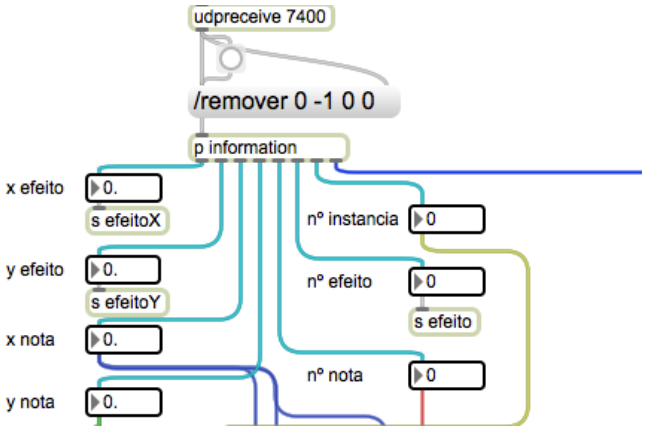


Imagem 71 - Parte do patch principal. Outlets das mensagens enviadas pelo sub-patch "p information".

Como podemos observar à esquerda, o sub-patch fica assim com 8 outlets:

- x efeito;
- y efeito;
- x nota;
- y nota;
- nº nota;
- nºefeito;
- nº instância;
- nº synth (não visível na imagem).

Segue-se então o roteamento da informação para onde é necessária.

O valor *x nota* vai ser enviado para o sub-patch "p velocity" juntamente com o valor *y nota.* Dentro deste sub-patch os valores vão ser sujeitos a uma comparação e posteriormente é feito um *scale* que determine que de 800 a 400 assume 30 100 e que de 0 a 400 assume 30 100. O mesmo processo é aplicado ao *y,* apenas muda 800 400 e 400 0 para 600 300 e 300 0. Os dois valores são comparados e se um deles for menor que 30 dispara o valor 0 para a velocity, senão dispara o valor que está a ser passado. Esse valor (0 ou outro) é enviado para o objecto poly~.

O valor de *x nota* está também a ser enviado para um scale 0 800 para 0 4, ou seja, divide a tela de ação em 5 partes. Essas 5 partes são referentes ao metro aplicado na nota. Ou seja, depois de redimensionado, se *x nota* = 0 então o metro da nota = 125.

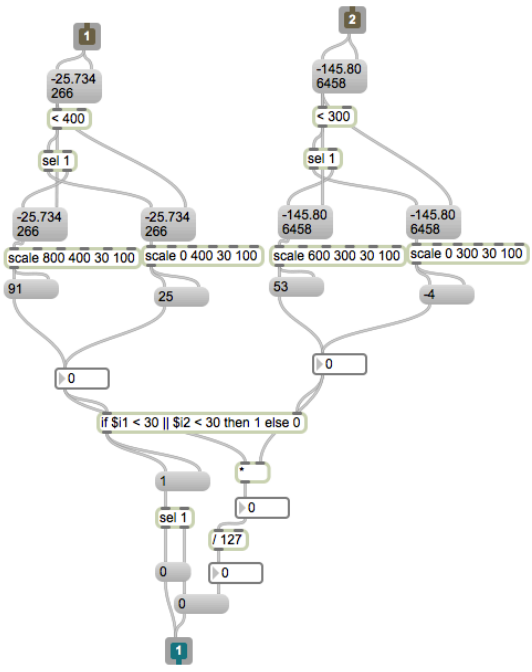


Imagem 72 - Sub-patch "p velocity".

Estes valores estão guardados no objecto "coll tempo". Quanto mais à esquerda, mais rápido é o disparo da nota. O valor correspondente à posição x é enviado para o objecto poly~.

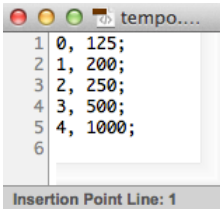


Imagem 73 - Coll tempo.

O valor *y nota* é enviado, não só para o sub-patch "p velocity"como foi referido em cima, mas também para um *scale* de 0 600 para 4 0. Este valor que obtemos após o *scale* está ligado também a um objecto *coll,* neste caso, "col transp" que determina a oitava em que a nota está. Quanto mais acima na tela, mais aguda é a nota.

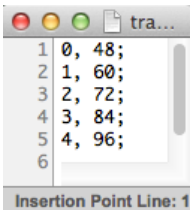
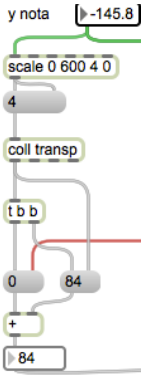


Imagem 74 - Coll transp.

Ao número nota somado este valor que obtemos do *coll* obtendo assim o nossa nota MIDI.



O valor da nota MIDI segue para o objecto poly~.

Imagem 75 - Soma do numero da nota (recebido pela linha vermelha) e o valor obtido no *coll*.

Entra então em funcionamento o objecto poly~.

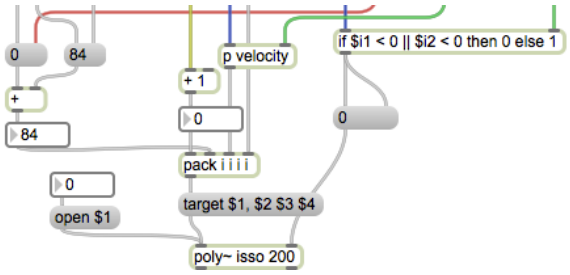


Imagem 76 - Excerto do Patch. Observa-se a mensagem enviada para o poly~.

O objecto poly~ é composto por várias instâncias onde são recebidos os mesmo tipos de valores, ou seja, é uma multiplicação de vozes.

O número da instância refere-se diretamente ao número da bola criada na interface, ou seja, cada bola vai assumir uma instância no poly~. Para isso, os valores são compactados numa só mensagem:

- Número da instância (proveniente da linha amarela na Imagem 76);
- Nota MIDI;
- Velocity;
- Metro.

Estas mensagens são enviadas para uma mensagem "target".

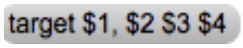


Imagem 77 – Mensagem Target.

Caso hipotético:

Sendo 4 o número da instância ($1), então vamos enviar para a instância 4: a nota MIDI 48 ($2), a velocity 100 ($3) e metro 1000 ($4).

Ou seja, target 4, 48 100 1000.

Assim permite-nos rotear as mensagens apenas para aquela voz. Cada voz é completamente independente.

No inlet da esquerda do poly~ entra essa mensagem, no da direita entra uma mensagem 0 ou 1, conforme a condição. Se *x nota* ou *y nota* for menor que 0 então manda 0, se não manda 1.

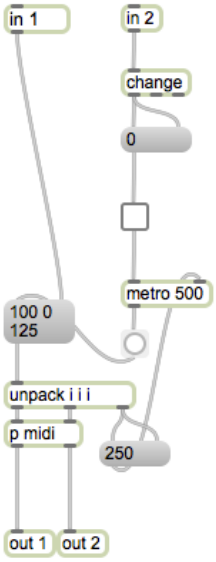


Imagem 78 – Patch do poly~.

Dentro do poly~ estas mensagens são processadas.

O inlet da esquerda passa as mensagens enviadas pelo target que são posteriormente "partidas" (unpack) dando origem à informação MIDI e ao valor do metro.

Se recebermos 1 no inlet da direita é activado um toggle que activa o metro, ou seja, é como se ligasse a voz. Se 0, o toggle desliga e a voz para de ser reproduzida (quando é removida ou posta fora da tela).

Dentro do sub-patch "p midi" é tratada a informaçãoo midi e enviada pelo primeiro outlet, e a informação para o plug-in no segundo outlet.

Imagem 79 – Sub-patch "p midi".

Os outlets do poly~ (linhas rosas na Imagem 80) seguem para outra secção do patch original. O outlet com a informação MIDI entra no objecto z.vst, o segundo outlet entra no objecto plug-in.

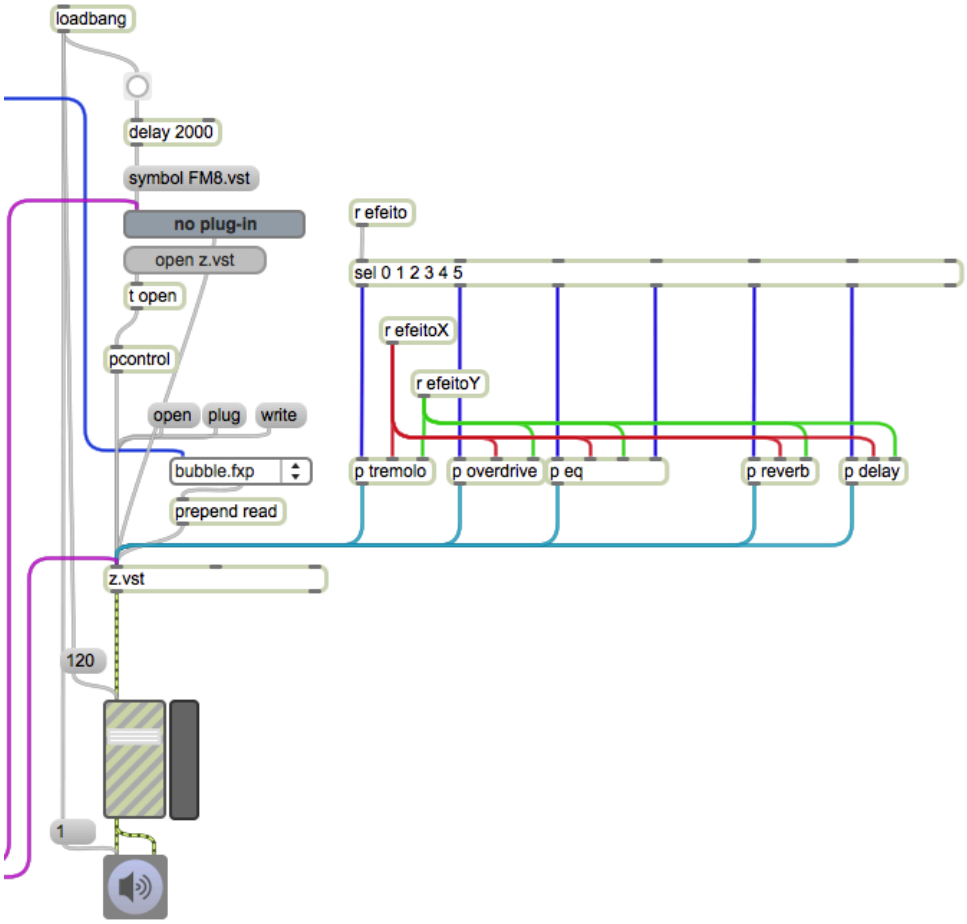


Imagem 80 – Secção do Patch Original.

Nesta secção, observamos um *loadbang*, que é enviado quando o programa é inicializado. Esse bang é enviado para abrir o plug-in "FM8", para inicializar o slide master a 120 e para ligar o ezdac~.

Após o plug-in "FM8" estar ativo, envia informação para o objecto z.vst.

Este objecto faz parte de uma biblioteca de extras para MaxMSP criada por Zachary Seldess.

O objecto faz a leitura do plug-in selecionado identificando todos os parâmetros que o compõem. Esses parâmetros são distribuídos pelo multislider e definidos com valores de 0. a 1.. Com este objecto conseguiu-se arranjar forma de controlar os parâmetros desejados através da interface da aplicação.

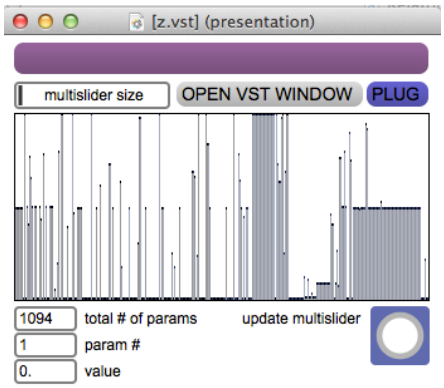


Imagem 81 – z.vst.

Para este objecto entram os seguintes parâmetros:

- Instrumento;
- Tremolo;
- Overdrive;
- Equalizador;
- Reverb;
- Delay.

O instrumento é definido através do número do item que nos foi enviado através de mensagem OSC.

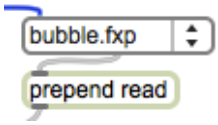


Imagem 82 – Umenu com os presets. Conforme o índex recebido (linha azul) seleciona o preset.

O Tremolo é o efeito equivalente ao primeiro quadrado. A identificação do quadrado que é arrastado na interface é essencial para sabermos que efeito será aplicado.

Para isso enviamos e recebemos o nº efeito.

Imagem 83 – Envio nº efeito.

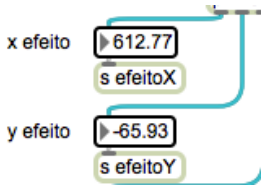


Imagem 84 – Envio x e y do efeito.

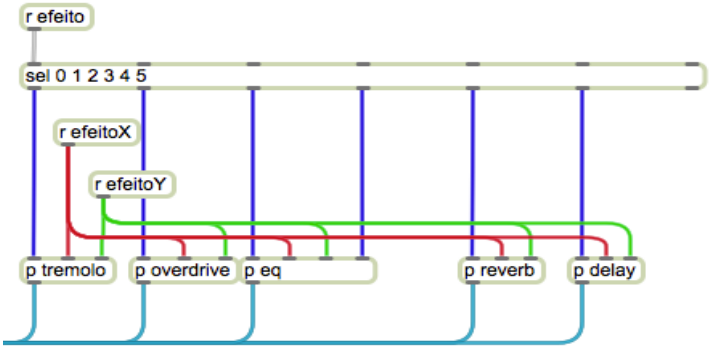


Imagem 85 – Recebe nº efeito e coordenadas.

O nº efeito é roteado por um *sel.* Se for 0, segue para o tremolo, se 1 para o overdrive, se 2 e 3 para a equalização, se 4 para o reverb e se 5 para o delay.

As cordenadas do quadrado são enviadas para todos os efeitos, pois estes são variáveis conforme a posição do quadrado em x e y mas apenas passa o que for roteado pelo *sel*.

Dentro de cada sub-patch de efeito, as mensagens das coordenadas são sujeitas a *scales* para valores entre 0. e 1. para poderem ser lidos no z.vst.

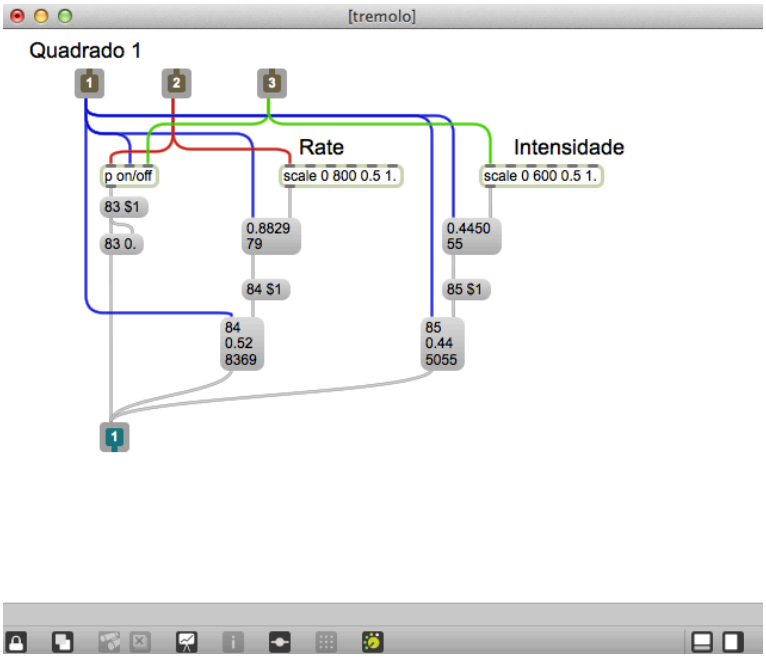


Imagem 86 – Sub-patch "p tremolo".

As linhas azuis transportam o bang que vai activar as mensagens caso aquele quadrado esteja em tela de ação. As linhas vermelhas transportam o *x*. As verdes o *y*.

O sub-patch "p on/off" processa-se exactamente da mesma forma que o sub-patch "p velocity" mas em vez de enviar 0 ou o valor da velocity, envia 0 ou 1, ou seja, depois de enviado com a mensagem "83 $1" vai assumir a posição desligado ou ligado.

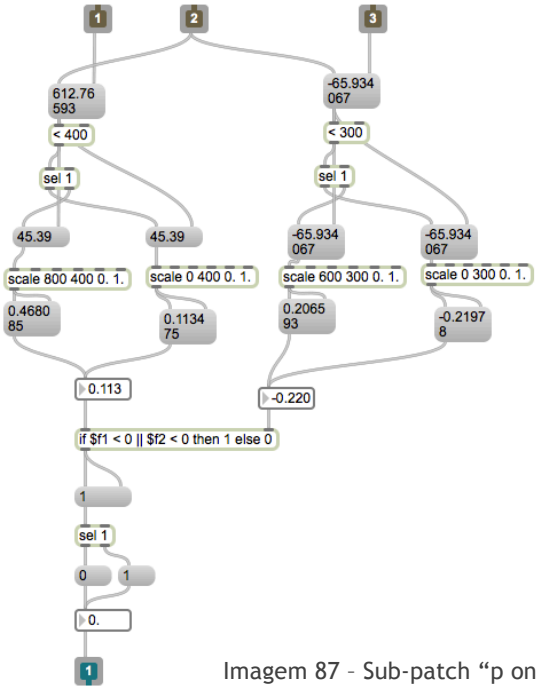


Imagem 87 – Sub-patch "p on/off".

Caso hipotético:

Se envia 1 (o quadrado encontra-se dentro da tela de ação), a mensagem enviada para o z.vst irá ser "83 0", sendo 83 o parâmetro referente ao on/off do tremolo.

84 é o parâmetro referente ao rate do tremolo e 85 o parâmetro referente à intensidade.

Fazendo este processo em todos os efeitos, conseguimos obter um resultado auditivo coerente e de uma forma muito mais intuitiva.

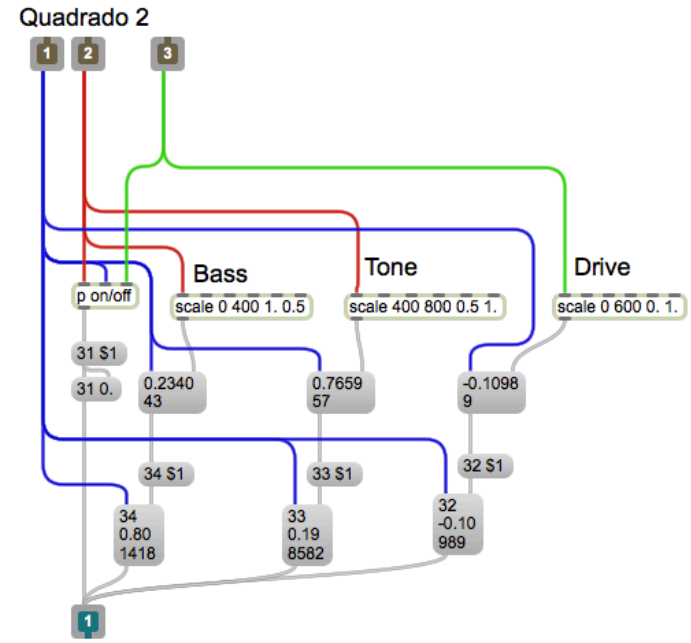


Imagem 88 – Sub-patch “p overdrive”.

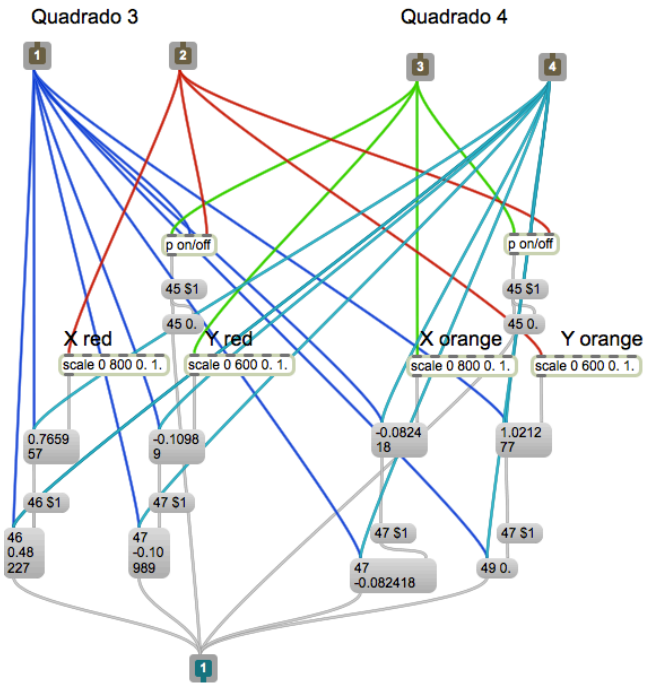


Imagem 89 – Sub-patch “p eq”.

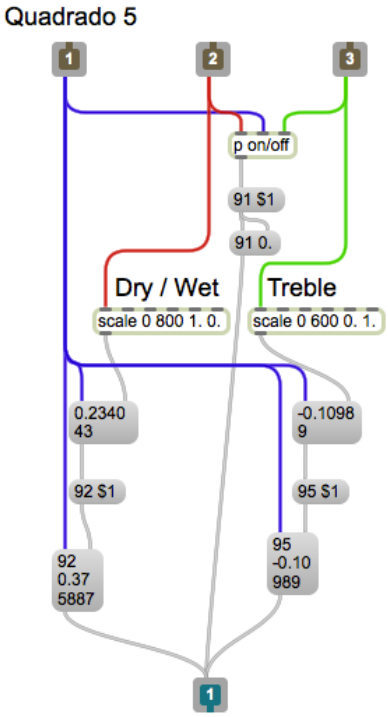


Imagem 90 – Sub-patch "p reverb".

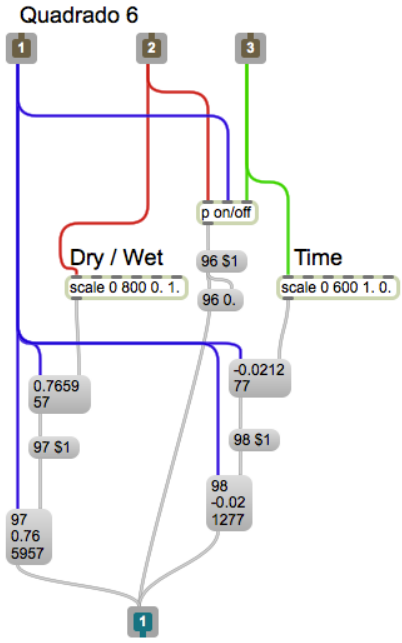


Imagem 91 – Sub-patch "p delay".

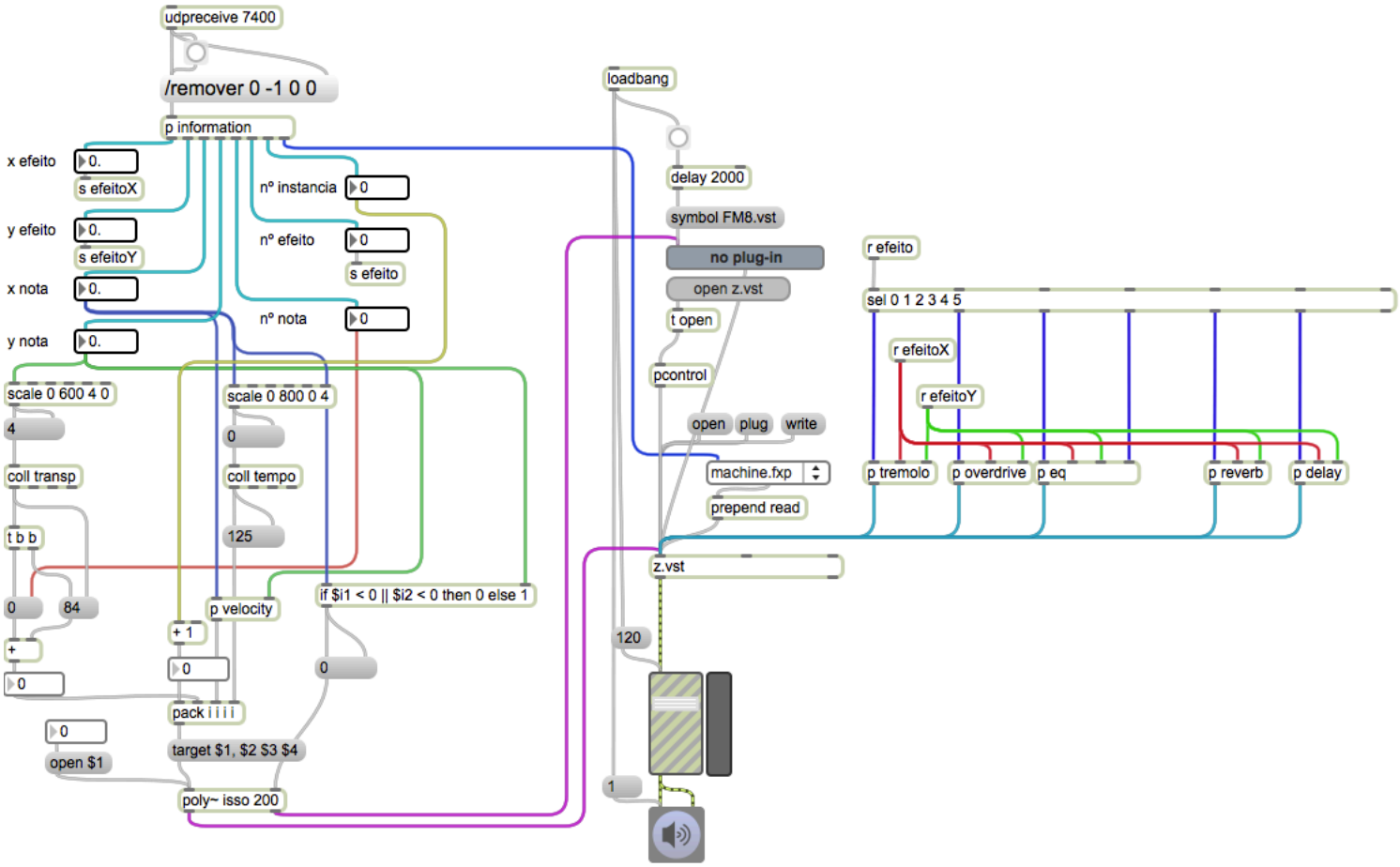


Imagem 92 – Patch completo da programação em MaxMSP.

8.2.4 - FM8

O sintetizador FM8 foi utilizado neste projeto. Por esta razão, achou-se por bem fazer uma pequena introdução a este instrumento virtual.

O FM8 é um instrumento virtual produzido pela Native Instruments. Este sintetizador trata essencialmente de síntese FM.

O conceito de síntese Fm é incrivelmente versátil mas pode ser complexo. Defendendo a abordagem do seu antecessor, Easy Edit, o FM8 traz um maior grau de simplicidade ao processo. (NATIVE INSTRUMENTS)

Possui uma biblioteca com cerca de 1200 presets sonoros. Alguns deles usados nesta aplicação.

Imagem 93 – FM8.

8.3 - Problemas

Ao longo do projeto deparei-me com algumas dificuldades, principalmente no desenvolvimento de código. Com pesquisa, dicas e explicações foi possível encontrar um caminho válido para chegar ao pretendido.

No entanto, houve um pormenor que falhou. Para a aplicação ter o impacto sonoro desejado foi necessário recorrer a um plug-in externo. Penso que, uma das coisas a desenvolver para esta aplicação no futuro será um sintetizador de raíz, completamente original, a aplicação tornar-se-ia muito mais rica e complexa.

Conclusão

Após a conclusão deste projeto faço um balanço positivo. Nunca me tinha testado desta forma a nível de programação em código e admito que foi uma experiência enriquecedora, tanto a nível académico como pessoal.

Falo a nível pessoal porque, à medida que desenvolvia este projeto, surgiram muito mais ideias para um futuro próximo. Para além disso, para concretizar este projeto, senti uma necessidade de desenvolver as minhas capacidades lógicas que ainda estavam muito "fechadas". Sinto que aprendi muito durante o tempo que dediquei a esta aplicação.

Pretendo torna-la melhor num futuro muito próximo pois, tal como disse, nenhum projeto se pode dar como concluído, há sempre possibilidade de melhorar.

Referências

Imagens

1. http://audiolist.org/forum/kb.php?mode=article&k=53

2. http://nautilus.fis.uc.pt/wwwfi/hipertextos/espectro/hiper_ espectro_vis.html

3. http://www.mundocor.com.br/cores/cor_musica.asp

4. http://rhythmiclight.com/archives/ideas/colorscales.html

5. http://www.webexhibits.org/colorart/bh.html

6. http://en.wikipedia.org/wiki/Color_organ

7. http://en.wikipedia.org/wiki/Louis_Bertrand_Castel

8. https://encrypted-tbn3.gstatic.com/images?q=tbn:ANd9GcSH5CleDcaKzJA1mkV0o8CMMqKlnCn94BgWn3j22AwUO_Xf_DX3

9. http://jdh.oxfordjournals.org/content/24/1/1/F7.large.jpg

10. Retirada de Jameson, D. D., *Colour-Music*, London, Smith, Elder and Co., 65, Cornhill, 1844.

11. http://www.lumen.nu/rekveld/wp/?page_id=185

12. http://sites.middlebury.edu/alexandra/lang/ru/scriabin/

13. http://www.classiccat.net/scriabin_a/biography.php

14. http://davewaughmusicthoughtprojects.blogspot.pt/

15. http://www.aaa.si.edu/collections/images/collection/i-j-belmont-papers-10258

16. http://www.aaa.si.edu/collections/images/collection/i-j-belmont-papers-10258

17. http://www.moma.org/collection/object.php?object_id=78302

18. http://lccpgdesign.com/2011/content/students/natasha-stone/projects/unit-1-square-project

19. http://lccpgdesign.com/2011/content/students/natasha-stone/projects/unit-1-square-project

20. http://lccpgdesign.com/2011/content/students/natasha-stone/projects/unit-1-square-project

21. http://blog.ted.com/2013/07/11/the-sound-of-color-neil-harbissons-talk-visualized/

22. http://en.wikipedia.org/wiki/Neil_Harbisson#cite_note-22

23. http://en.wikipedia.org/wiki/Neil_Harbisson#cite_note-22

24. http://lounge.obviousmag.org/coqueluche/2013/02/neil-harbisson.html

25. http://en.wikipedia.org/wiki/Neil_Harbisson#cite_note-22

26. http://venturevillage.eu/neil-harbisson-calls-for-fewer-apps-for-mobile-more-for-the-body

27. http://www.scpr.org/programs/airtalk/2013/05/13/31774/cyborg-neil-harbisson-listens-to-color/?slide=3

28. http://venturevillage.eu/neil-harbisson-calls-for-fewer-apps-for-mobile-more-for-the-body

29. http://www.scpr.org/programs/airtalk/2013/05/13/31774/cyborg-neil-harbisson-listens-to-color/?slide=3

30. http://www.scpr.org/programs/airtalk/2013/05/13/31774/cyborg-neil-harbisson-listens-to-color/?slide=3

31. http://cnlohr.blogspot.pt/2010/11/colorchord-sound-lighting.html

32. http://www.perfectpitch.com/hearforyourself.htm

33. http://www.reactable.com/

34. http://www.reactable.com/

35. http://www.reactable.com/

36. https://itunes.apple.com/pt/app/bucephalus/id597192419?mt=8

37. https://itunes.apple.com/pt/app/bucephalus/id597192419?mt=8

Bibliografia

Neil Harbisson

http://super.abril.com.br/blogs/ted/o-som-das-cores/

http://blog.ted.com/2013/07/11/the-sound-of-color-neil-harbissons-talk-visualized/

http://blog.ted.com/2012/06/27/listening-to-picasso-neil-harbisson-at-tedglobal2012/

http://en.wikipedia.org/wiki/Cyborg_Foundation

http://en.wikipedia.org/wiki/Neil_Harbisson#cite_note-22

http://venturevillage.eu/neil-harbisson-calls-for-fewer-apps-for-mobile-more-for-the-body

Visual Music

http://en.wikipedia.org/wiki/Sound_installation

http://homepage.eircom.net/~musima/visualmusic/visualmusic.htm

http://prometheus.kai.ru/ogl_e.htm

http://www.mts.net/~william5/history/hol.htm

Sinestesia

http://pt.wikipedia.org/wiki/Sinestesia

http://www.unc.edu/~parunapu/Assign3/uses.php

http://davewaughmusicthoughtprojects.blogspot.pt/

Propostas

http://pt.wikipedia.org/wiki/Olivier_Messiaen

http://www.classiccat.net/scriabin_a/biography.php

https://ericwilliambarnum.wordpress.com/tag/alexander-scriabin/

http://lccpgdesign.com/2011/content/students/natasha-stone/projects/unit-1-square-project

http://www.webexhibits.org/colorart/bh.html

http://rhythmiclight.com/archives/timeline.html

http://rhythmiclight.com/archives/ideas/colorscales.html

http://mnemotechnics.org/forums/alexander-scriabin-and-artificial-synesthesia-1016.html

Relação Cor-Som

http://audiolist.org/forum/kb.php?mode=article&k=53

http://culturadigital.br/dacio/2011/06/04/convertendo-sons-em-cores/

http://musicoterapiacampinas.blogspot.pt/2011/05/as-cores-do-som-relacoes-entre-cores-e.html

http://www.mundocor.com.br/cores/cor_musica.asp

http://www.projetofundao.ufrj.br/biologia/images/materiais/os_sons_e_as_cores_juliana_americo.pdf

http://www.virtuosoism.com/Color-Sound%20Relationships.pdf

http://www.davidrokeby.com/Culturall2/40_colour.html

http://www.simonbielman.com/2010/08/midi-keyboard-color-projetor/

http://cnlohr.blogspot.pt/2010/11/colorchord-sound-lighting.html

http://www.perfectpitch.com/minilesson.htm

http://www.mundocor.com.br/cores/cores_sons.asp

http://www.dataisnature.com/?p=443

http://www.cabinetmagazine.org/issues/22/peel.php

http://www.liveart.org/motherboard/synaesthesia/lectureimgs_280806.html

http://longstreet.typepad.com/thesciencebookstore/2008/05/color-music--fo.html

Programação

http://forum.processing.org/one/topic/controlp5-dropdown-setvalue.html

http://forum.processing.org/one/topic/simple-code-create-a-drop-down-list-with-available-com-ports.html

http://stackoverflow.com/questions/20161827/controlp5-listbox-selection-active-colour-processing

http://www.sojamo.de/libraries/controlP5/reference/controlP5/DropdownList.html

https://github.com/beattiea/TiltyIMU/blob/master/Tilty%20Software/Processing/libraries/controlP5/controlP5/examples/controllers/ControlP5dropdownList/ControlP5dropdownList.pde

http://forum.processing.org/one/topic/controlp5-and-dropdownlist-problem.html

http://forum.processing.org/one/topic/help-with-drop-down-menu.html

http://cycling74.com/forums/topic/oversampling-with-poly/

http://forum.processing.org/one/topic/controlp5-dropdown-list.html

http://openprocessing.org/sketch/5921

http://processing.org/discourse/beta/num_1235422289.html

http://forum.processing.org/one/topic/control-p5-changing-mouseover-color-font-size-font.html

http://www.openprocessing.org/sketch/8113

http://www.cycling74.com/docs/max5/refpages/msp-ref/poly~.html

http://www.sojamo.de/libraries/oscP5/

http://www.sojamo.de/

http://www.sojamo.de/libraries/oscp5/reference/

https://www.processing.org/tutorials/

https://www.processing.org/reference/

https://www.processing.org/examples/mousepress.html

https://www.processing.org/examples/

https://www.processing.org/examples/button.html

https://www.processing.org/reference/mousePressed_.html

http://stackoverflow.com/questions/481144/equation-for-testing-if-a-point-is-inside-a-circle

http://processing.org/discourse/beta/num_1275602375.html

http://www.openprocessing.org/sketch/23665

http://forum.processing.org/one/topic/game-menu.html

https://www.processing.org/reference/mouseButton.html

http://cycling74.com/tag/vst/

http://www.cycling74.com/docs/max5/refpages/msp-ref/vst~.html

https://www.processing.org/reference/switch.html

https://www.processing.org/reference/ArrayList.html

http://www.processing.org/reference/case.html

file:///Applications/Processing.app/Contents/Java/modes/java/reference/class.html

FM8

http://www.native-instruments.com/en/products/komplete/synths/fm8/

Interatividade

http://searchsoa.techtarget.com/definition/interactivity

http://www2.gsu.edu/~wwwitr/docs/interact/

http://www.slideshare.net/softpalm/princpios-bsicos-do-design-de-sistemas-interativos

http://www.eca.usp.br/mobile/portal/index.php?q=node/14

http://osbonsdeapi.wordpress.com/conceito-de-interatividade/

http://www.slideshare.net/ecoarte/arte-tecnologia-e-instalao-interativas

http://www.ime.usp.br/~leliane/MAC5701/2005-1oSem/PlanosMonografias/Mariana.pdf

http://www.reactable.com/

https://itunes.apple.com/pt/app/bucephalus/id597192419?mt=8

http://www.synthtopia.com/content/2013/02/26/bucephalus-for-ios-lets-you-experiment-with-physics-based-generative-music/