Visualização de Música à Distância de um Gesto

João Tiago Gomes Maria Beatriz Carmo Ana Paula Cláudio
LabMAg, Faculdade de Ciências, Universidade de Lisboa
Campo Grande 1749-016
fc37415@alunos.fc.ul.pt, {bc,apc}@di.fc.ul.pt

Sumário
Este artigo apresenta uma aplicação que tem como objetivo dar a qualquer pessoa a possibilidade de criar música partindo de vários sons, que podem ser progressivamente compostos em níveis crescentes de complexidade. Inspirada no instrumento eletrónico Theremin, a interação será feita de modo a criar ou manipular música de uma forma natural. A composição de música é acompanhada de uma visualização tridimensional, que não só serve de guia ao utilizador no processo de composição, como também permite visualizar a música que está a ser criada.

Keywords
Visualização, música, composição musical, leap motion, interação natural

1. INTRODUÇÃO

Lev Sergeevich Termen patenteou em 1928 um instrumento musical eletrónico com o nome *Theremin*, que era controlado pelo músico sem qualquer contacto físico [Theremin28]. O instrumento dispõe de duas antenas que controlam respetivamente o *pitch* (ou frequência) e a amplitude (ou volume) do som. O músico controla cada antena com uma mão, afastando ou aproximando as mãos das antenas de modo a fazer variar o som do instrumento.

A 21 de Maio de 2012 foi anunciado pela empresa Leap Motion um controlador, com o mesmo nome, constituído por duas camaras infravermelhas e três LED infravermelhos. Este dispositivo permite fazer o *tracking* dos dedos e das mãos de um utilizador de forma extremamente precisa (precisão até 1/100 de milímetro) num espaço mais ou menos hemisférico com 1 metro de alcance [Leap].

Em meados de 2013 iniciou-se a comercialização do dispositivo Leap Motion e desde então surgiram aplicações em diversos domínios, como, por exemplo, na área da medicina ou jogos interativos, que tiram partido desta nova forma de interação. Em particular na área musical foi publicado, na loja da Leap Motion (Airspace [Airspace]), um considerável número de aplicações, grande parte das quais se concentrava em tornar a Leap Motion um controlador MIDI. No entanto este trabalho, ainda que inserido na área musical, segue outro rumo.

As semelhanças entre um *Theremin* e um controlador Leap Motion são óbvias. Assim surgiu a ideia de usar o dispositivo Leap Motion não só para tocar um instrumento virtual que produza apenas um tipo de som, tal como acontece com o *Theremin*, mas também para tocar vários sons de vários tipos diferentes ao mesmo tempo, servindo assim de ferramenta de composição musical. O propósito deste trabalho é, então, sobrepor vários sons de modo a criar uma música. A composição da música e a sua complexidade ficam a cargo do utilizador.

O processo de produzir vários tipos de sons, quer seja através da voz quer seja através de um instrumento, e de juntar em tempo real esses sons de modo a que eles formem uma música não é algo novo. Vários músicos contemporâneos têm seguido esta abordagem só possível com recurso à tecnologia (hardware e software) que existe hoje. É importante também referir que esta tendência não se prende a um estilo musical. Existem hoje artistas das mais variadas áreas que utilizam este processo para produzir música, em áreas que vão *do Hip Hop* [Dub FX] à música erudita [Zoe Keating], passando pelo *Rap, Pop, Soul* [Bernhoft], *RnB* e *Jazz* [Tom Thum].

O objetivo deste trabalho foi desenvolver uma aplicação que reunisse três componentes: em primeiro lugar, a vertente musical, que possibilita ao utilizador tocar vários sons de diferentes formas e compô-los de modo a criar música fácil e interactivamente; em segundo lugar, a utilização do dispositivo *Leap Motion* como interface para a criação da música na aplicação, tomando especial atenção à forma como esta interação é feita, de modo a que seja o mais natural possível; e, finalmente, a visualização da música de forma a distinguir claramente os vários sons utilizados na composição. Estes três módulos, embora distintos, têm de funcionar em conjunto para proporcionar uma boa experiência de utilização. Isto torna-se particularmente relevante pelo facto de estarmos a trabalhar com um dipositivo de interação, que não o rato e o teclado, com o qual a maior parte dos utilizadores

não está familiarizada. Uma vez que estamos a utilizar o dispositivo Leap Motion, temos que ter em conta novos problemas e desafios que não encontramos com as interfaces tradicionais, que já são estudadas há bastante tempo.

Nesta secção foi dada uma pequena introdução aos vários temas que foram usados como inspiração para a aplicação. Na secção 2 é apresentado o trabalho relacionado com o descrito neste artigo. Na secção 3 são abordados os vários componentes que constituem a aplicação. Na secção 4 são apresentadas as principais conclusões. E finalmente na secção 5 são sugeridos melhoramentos futuros.

2. TRABALHO RELACIONADO

2.1 Processo Musical

Foi analisado o trabalho de vários artistas musicais (*DubFX, Zöe Keating, Tom Thum e Jarle Bernhoft*) que fazem música sozinhos recorrendo a vários instrumentos e sons para tentar perceber o processo utilizado por eles. Há aspectos comuns a todos estes artistas neste processo, variando em alguns detalhes, nomeadamente no nível de complexidade da música criada e no nível de preparação prévia necessária. O processo consiste no seguinte: em primeiro lugar, o artista dispõe sempre de equipamento (hardware) e/ou tecnologia (software) que lhe permite gravar várias *tracks* (em português, faixas, mas é importante não confundir com o termo faixas que é usado coloquialmente para nos referirmos a uma música de um álbum). O artista começa por gravar numa *track* um som, quer seja ele vindo da sua própria voz quer seja de um instrumento. Após a gravação, esse som começa a ser reproduzido imediatamente em *loop*. De seguida é gravado outro som noutra *track*, enquanto o anterior está a ser reproduzido. Este processo vai-se repetindo, de forma a que as várias *tracks* se vão sobrepondo para formar uma música cada vez mais complexa. A música produzida com este processo pode ser bastante complexa, com várias *tracks* sobrepostas e a reproduzirem vários sons, ou algo simples com poucas *tracks*: cabe ao artista decidir.

2.2 Visualização de Música em Tempo Real

Foram analisadas aplicações que têm a capacidade de apresentar uma visualização em tempo real de uma música que está a ser tocada. Verificou-se que em alguns casos a visualização é específica para uma dada música e é necessário alterar o código fonte para a substituir por outra. No entanto, a visualização dessa música é feita em tempo real.

As aplicações analisadas fazem uso da informação acerca das frequências do som num dado momento para construírem a respetiva visualização. O exemplo mais comum disto é um simples equalizador de barras que se encontra em inúmeros aparelhos de música.

Foram analisadas 4 aplicações com visualizações em 3D (*Loop Waveform Visualizer [Loop]*, *Music Colour Particles* [Color Particles], *Cube Visualizer [Cube Visualizer]e A dive in Music* [Dive]). Três delas possuem apenas uma visualização disponível e são, por isso, mais simples. Estas aplicações mais básicas fazem uso de visualizações que são já bastante conhecidas, tal como uma simples onda de som (figura 1), e transformam-na numa versão 3D.

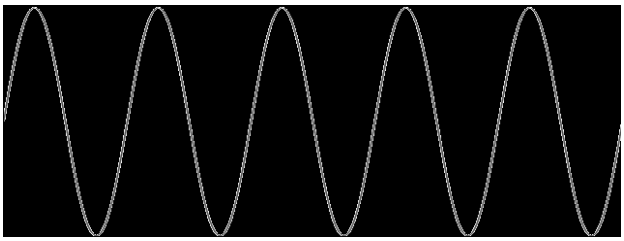

Figura 1 - Onda de som

Por outro lado, apesar de serem mais simples, estas aplicações permitem que se tenha uma clara noção da relação entre o som e a respetiva representação gráfica. Quando passamos para aplicações mais elaboradas visualmente, esta relação tende a perder-se. Caso disso é a aplicação *A dive in Music* que tem mais de 20 visualizações bastante complexas e que envolvem emissores de partículas. O resultado final que se obtém é mais agradável esteticamente mas perde-se a relação que existe entre som e imagem.

2.3 Visualização de Música com Pré-processamento

A vantagem clara da visualização de música com pré-processamento em relação às visualizações feitas em tempo real é que o resultado gráfico pode ser muito mais rico e pode usar mais poder de processamento para gerar a imagem final, não só relativamente à visualização, como à extração da informação da música que servirá de *input* para a visualização. Assim, e comparando, por exemplo, a visualização anteriormente referida *A dive in Music* com *Ljósið* [Ljósið], é notória a diferença na qualidade das partículas usadas e no efeito visual que esta visualização causa.

Note-se que ambas as visualizações recorrem a partículas; no entanto, em *Ljósið* é possível distinguir claramente a que som pertence cada visualização. O mesmo não acontece no exemplo anterior. Uma explicação possível para que isto aconteça prende-se com a natureza pré-processada da visualização e o tempo e esforço que pode ter sido despendido a fazer uma separação mais cuidada dos sons individuais de cada instrumento (piano e violino), em vez de usar simplesmente o som como um todo, tal como é feito nas aplicações em tempo real. Outro exemplo de visualização de música com pré-processamento foi desenvolvido por Christopher Garcia [Garcia11]. Este autor, a partir da análise da música, gera um ficheiro de texto que serve de *input* à geração da visualização.

2.4 Visualização e criação de música simultaneamente

Nesta secção são abordadas algumas aplicações que além de gerarem uma visualização da música, tal como as anteriores, têm também a capacidade de permitir ao

utilizador criar música. Isto muda o propósito da aplicação, se compararmos com as anteriormente referidas, pois agora o foco é na criação da música e não simplesmente em assistir a uma visualização. A interatividade desempenha, portanto, um papel central.

Foram analisadas aplicações com visualizações tanto em 2D como em 3D mas todas eram relativamente parecidas (*ToneMatrix* [ToneMatrix], *Beat Petite* [BeatPetite] *e ToneCraft* [Tone Craft]). Nestas aplicações é possível tocar vários sons em diferentes momentos, sendo que o metrónomo que marca o tempo é um elemento bastante presente e notório pois é ele que dá estrutura à música. Em cada uma das aplicações é possível tocar um total de 16 notas até que o tempo volte ao início (encontrando-se o tempo num *loop* infinito). Em três das aplicações analisadas os elementos gráficos estão dispostos de acordo com uma grelha regular bidimensional. Nessas grelhas o eixo dos *xx* corresponde ao tempo e o eixo dos *yy* corresponde ou a um som diferente ou, tratando-se do mesmo som, um *pitch* diferente desse som, o que corresponde na prática a uma nota diferente que é tocada.

Existem neste conjunto de aplicações dois pontos aos quais vale a pena prestar alguma atenção. O primeiro é o facto de existir um tempo subjacente e que regula o momento a que as notas são tocadas, um metrónomo.

Este metrónomo tem a particularidade de, quando acabam de tocar as 16 notas possíveis, voltar ao início e começar a tocar de novo as mesmas notas. Isto produz um efeito interessante que, à partida, pode passar despercebido. O facto de existir uma repetição constante faz com que quaisquer que sejam as notas tocadas, a música que daí resulta soa agradavelmente. O segundo ponto prende-se em particular com a aplicação *ToneMatrix*, que, para além de fazer uso do metrónomo e da repetição, tal como foi referido anteriormente, junta uma outra funcionalidade, o ligeiro atraso entre cada batida do metrónomo. A repetição e o atraso tornam a sonoridade da música semelhante a um andamento de jazz.

3. TRABALHO DESENVOLVIDO

O trabalho desenvolvido envolveu três componentes principais: a) áudio, que trata da reprodução e gravação dos vários sons, da combinação dos sons gravados e da marcação do tempo da forma mais exata possível; b) interação do utilizador, com o recurso ao controlador Leap Motion, para a composição de música e a interface gráfica que auxilia a construção e a composição da música; c) visualização da música criada.

O trabalho foi desenvolvido com recurso à linguagem *javascript* e para o browser *Google Chrome* que é o único browser neste momento que suporta todas as funcionalidades de HTML5 necessárias para a aplicação funcionar. A aplicação resultante ficará disponível numa página web, que, uma vez *online*, permitirá que qualquer pessoa possa aceder à aplicação de forma rápida e fácil, não sendo preciso fazer qualquer *download* adicional. Esta foi uma das decisões que mais pesou na escolha das tecnologias a usar para desenvolver este projeto. Deste modo, é apenas preciso possuir o dispositivo Leap Motion e aceder a um endereço *web* para usufruir da aplicação.

3.1 Audio

A componente de áudio foi a que se demonstrou mais desafiante. Nesta componente podemos encontrar duas vertentes. Por um lado desenvolver um metrónomo para a aplicação ser capaz de marcar o tempo de forma exata. Por outro lado, disponibilizar vários sons, organizar as várias *tracks* que compõem a aplicação e lidar com o processo de reprodução e gravação de *tracks*.

Tendo em conta a tecnologia utilizada, foi difícil conseguir acomodar estes requisitos e ao mesmo tempo ter em conta aspetos de desempenho, isto é evitar que os atrasos no som sejam percetíveis aos utilizadores.

3.1.1 Metrónomo, marcação de tempo e desempenho

Ao começar o módulo de áudio foi claro que a marcação do tempo e a necessidade de construir um metrónomo que fosse capaz de o contar de forma precisa e igualmente espaçada seria crucial. Assim, e nunca esquecendo que toda a aplicação é em tempo real, pensou-se desde o início nas implicações em termos de desempenho. Desde logo pelo facto de a aplicação correr num *browser*, fica limitada em termos de precisão do seu relógio. O valor de atraso do relógio pode ser significativo quando se trata de som. Por isso, foram usadas várias técnicas para tentar mitigar os efeitos do atraso do relógio do *Google Chrome*, *browser* para o qual a aplicação foi desenvolvida.

A forma como o metrónomo funciona na aplicação é a seguinte: com base no BPM (*beats per minute*) e no número de notas que é possível dar num dado compasso, é definido um número de milissegundos de intervalo entre cada "batida" da música; cada vez que passa esse intervalo de tempo, o metrónomo "acorda" e sabe que necessita de efetuar uma série de ações; essas ações em *javascript* traduz-se na execução de uma função *callback*. Esta função é responsável por: a) começar a gravar uma *track*, caso tenha sido efetuada essa ação pelo utilizador; b) terminar de gravar uma *track*; c) actualizar o contador que é responsável por saber quando parar de gravar uma *track*; d) reproduzir som se o utilizador quer tocar o som de determinada *track*; e) reproduzir os sons das *tracks* base; f) caso haja alguma *track* com algum som já gravado e se for o momento de reproduzir esse som, de facto reproduzi-lo g) por último, acertar os valores da contagem das notas que servem para marcar o tempo (estes valores são apresentados na interface gráfica e serão explicados numa secção posterior).

Como já foi referido anteriormente, existe um atraso de alguns milissegundos no relógio das aplicações que usam o *browser*. Isto faz com que o intervalo de tempo que passa entre cada "acordar" do metrónomo não seja igual. O efeito cumulativo de todos os atrasos causa, naturalmente, problemas acrescidos em termos de desempenho.

Por esta razão, o que se fez para resolver este problema foi ajustar o número de milissegundos que leva o próximo "acordar" do metrónomo, não para o seu valor esperado, mas sim para um valor que tenha em conta o atraso anterior. Por exemplo, supondo que o metrónomo deveria "acordar" no instante 100 mas que, devido ao atraso, apenas acorda no instante 104, e supondo ainda que o intervalo entre cada "acordar" do metrónomo é de 100ms, então o próximo instante seria provavelmente o 208 (104 do instante original mais os 100ms de intervalo mais o atraso que este segundo "acordar" teria, 104 + 100 + 4 = 208). Ora, este não é o efeito pretendido. O esperado seria o próximo "acordar" do metrónomo aos 200ms ou, na pior das hipóteses, no instante 204, que tem apenas o erro deste instante e não o erro acumulado do "acordar" inicial.

Para conseguir os tempos o mais regulares possível foram usadas algumas técnicas. A primeira foi ter em conta o atraso do *callback* passado e chamar o próximo com uma correção. No exemplo anterior, isto queria dizer que o segundo *callback* seria chamado não com 100ms de atraso mas sim com 96ms (100ms menos 4ms de atraso do *callback* anterior). Assim elimina-se o erro cumulativo e fica-se apenas com o erro de cada *callback*, sendo este ajustado para o próximo *callback* que é feito.

A técnica descrita acima reduz para níveis aceitáveis o erro de relógio entre as chamadas de *callback*. Mas dentro da função *callback* que é sistematicamente chamada é preciso também ter cuidados. Nesta função são feitas operações como começar a gravar, parar de gravar, tocar o som de várias *tracks*. Se estas operações não forem feitas em sequência resultam em discrepâncias no áudio posteriormente ouvido pelo utilizador. Assim, existe uma zona crítica na qual as operações têm de ser feitas consecutivamente. Para isso, é necessário que as operações não críticas sejam previamente preparadas.

Testou-se a utilização de *threads* para esta parte do áudio, mas verificou-se a sua ineficiência, pois a comunicação entre a *thread* principal e a secundária implicaria demasiado atraso, e isto seria inaceitável numa aplicação que corre em tempo real e que trata áudio.

3.1.2 Tracks

A aplicação é constituída por várias *tracks*, tocando cada uma um som específico. Esse som é obtido através de um ficheiro *.wav* carregado quando a aplicação inicia e que contém o som de um instrumento ou uma parte de um instrumento (maioritariamente sons de bateria: bombo, tarola, pratos). Existem tipos diferentes de *tracks*: as *tracks* base e as *tracks* normais.

A diferença estre as duas é o nível de controlo que o utilizador tem sobre elas: nas *tracks* base o utilizador não tem qualquer controlo, elas são tocadas automaticamente desde que a aplicação é iniciada até que é encerrada. O objetivo destas *tracks* é dar ao utilizador uma base sobre a qual trabalhar. Presentemente existem 2 *tracks* base, uma que marca o ritmo com o som de um bombo de uma bateria e uma que dá um som envolvente e contínuo. Este som contínuo é sintetizado em tempo de execução, usando geração aleatória de sons limitados a um dado intervalo de frequências.

Passando para as *tracks* normais, estas podem ser controladas pelo utilizador, concretamente através do dispositivo Leap Motion. Será explicado em mais detalhe na secção seguinte como é que é feita esta interação.

Os sons para cada uma das *tracks* foram escolhidos tendo em conta como soam isoladamente e em conjunto e – o que é mais importante – pela sua duração. Tal como já foi explicado anteriormente, existe um intervalo de tempo específico que separa as chamadas consecutivas de um *callback* e os sons, ao serem escolhidos, não podem exceder esse limite para que não se sobreponham uns aos outros.

Finalmente, o som envolvente encontrado numa das *tracks* base é o único que não é real mas sim sintetizado. Foi usado código de uma biblioteca para obter este som. A forma como o som é obtido envolve o uso de vários *Biquad Filters* [Biquad Filter] sobrepostos (por defeito são usados 20) e a utilização de um *Panner* [Panner] para mudar a posição de onde o som vem, o que permite que o som ouvido não seja monótono e pareça variar, produzindo um efeito de envolvência.

3.1.3 Gravar Som

Para gravar o som foram tentadas várias abordagens. A primeira foi gravar o som usando uma biblioteca que utilizava nós *javascript* e *threads*. O resultado não obtinha a precisão necessária para começar a gravar nem para parar num momento exato, devido à latência inerente às *threads* em *javascript* e aos nós *javascript* que são usados na *Web Audio API* [Web Audio API].

Depois de várias iterações de tentativa e erro, chegou-se finalmente a uma solução aceitável em termos de desempenho: não gravar o som que saía de determinada *track* e sim gravar o momento em que o som era tocado. Isto significa que na prática o que acontece é: quando a aplicação está em modo de gravação, sabe que está a gravar determinada *track* e sabe quando é tocado um som; assim, é gravado num *array* um valor booleano, se o valor for verdadeiro então, quando for altura de reproduzir o som posteriormente, este é reproduzido, se o valor for falso, o som não é reproduzido.

3.2 Interação

A interação com o utilizador é realizada através de duas componentes que se complementam: uma baseada na Leap Motion e outra na interface gráfica.

3.2.1 Interação com Leap Motion

A interação entre o utilizador e a aplicação no que respeita a composição da música e dos vários sons que se encontram nas várias *tracks* é feita através do dispositivo Leap Motion (figura 2).

Figura 2 - O dispositivo Leap Motion

Ao utilizar-se este dispositivo para fazer tocar sons e para controlar a parte musical da aplicação, foi necessário ter em conta novos aspetos de interação que são radicalmente diferentes de, por exemplo, um rato e teclado.

Na aplicação o utilizador consegue controlar sons de ritmo, ou seja, sons que não se prolongam no tempo, por exemplo, o som de um tambor de uma bateria. Um contra-exemplo disto seria o som de um violino, onde o músico pode prolongar o som de uma nota por algum tempo. Assim, visto que estes sons terminam rapidamente, escolheu-se dar ao utilizador a possibilidade de controlar dois parâmetros relativamente a eles: primeiro, o número de vezes que é tocado (tendo em conta os BPMs da aplicação), considerando sempre que o momento tocado é controlado pelo metrónomo da aplicação para que os sons "encaixem" todos no mesmo sítio; em segundo lugar, o volume do som.

Uma vez selecionada uma *track* e consequentemente um som, o utilizador pode tocar esse som. A mão esquerda controla o número de vezes que o som é tocado. Opcionalmente, a mão direita pode controlar o volume do som que está a ser tocado. Para conseguir identificar as várias posições das mãos foram definidos limites, tanto horizontais como verticais. Assim, definiu-se um plano vertical, paralelo aos eixos dos *yy* e dos *zz* colocado ao meio da imagem e que separa as zonas de ação de cada uma das mãos (figura 3). Para a mão esquerda foram definidas verticalmente quatro zonas, correspondendo a cada uma ações distintas: a mais alta impede que seja tocado qualquer som (z1); a seguinte toca o som a uma velocidade normal, e, de acordo com a batida do metrónomo (z2); abaixo desta o som toca a uma velocidade duas vezes maior que o normal (z3); e finalmente a última toca o som a uma velocidade quatro vezes superior à normal (z4).

A velocidade a que o som é tocado está diretamente relacionado com os BPMs do metrónomo e do número de notas que é possível tocar, bem como os momentos em que podem ser tocadas.

Na tabela 1 assinalam-se com "x" as células correspondentes aos momentos em que os sons podem ser reproduzidos, consoante a zona (z1, z2, z3 ou z4) em que a mão esquerda do utilizador se encontra.

A mão direita controla o volume do som tocado. Para esta mão foram igualmente definidos limites que representam diferentes ações. Assim, as zonas verticais utilizadas para a mão direita consistem no seguinte: a zona mais alta corresponde ao volume mais alto (80%), a zona mais baixa corresponde ao volume mais baixo (10%) e na zona intermédia é feita uma proporção entre as coordenadas da mão do utilizador e o volume. De notar que as zonas nos extremos, mais alta e mais baixa, ocupam cada uma 25% do espaço e a zona intermédia ocupa os restantes 50% do espaço.

Tal como foi mencionado anteriormente, o utilizador pode escolher utilizar apenas a mão esquerda (a que controla se o som é tocado ou não) e se assim for o volume do som corresponde ao último volume utilizado pela mão direita ou, caso a mão direita nunca tenha sido utilizada naquela *track*, é usado um volume por omissão.

Uma vantagem que advém disto é a possibilidade de o utilizador escolher o volume desejado para tocar determinado som com a mão direita e depois usar unicamente a esquerda para escolher os momentos exatos em que quer que o som seja tocado.

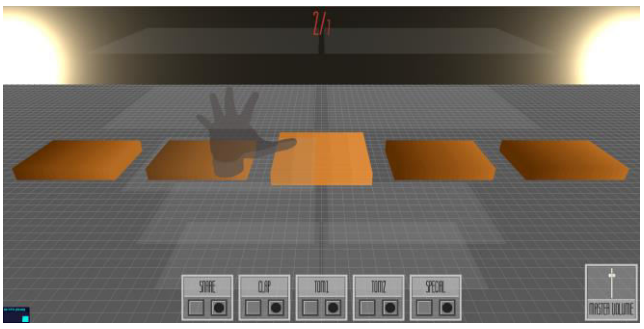

Figura 3 - Imagem da aplicação sem nenhuma *track* selecionada

z1																
z2	x				x				x				x			
z3	x		x		x		x		x		x		x		x	
z4	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Zona vs tempo	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**

Tabela 1 –Momentos em que os sons podem ser reproduzidos em cada zona de interação

Na figura 3 é possível ver a mão virtual de um utilizador que está a usar a aplicação. São visíveis também os limites discutidos anteriormente. Cinco planos dividem o lado esquerdo de modo a criar quatro zonas e quatro planos dividem o lado direito para criar três zonas. De notar, no entanto, que na imagem nenhuma *track* está selecionada, e por isso tanto os limites como a mão do utilizador encontram-se preenchidas com uma cor neutra. Assim que uma *track* é selecionada, tanto as mãos virtuais do utilizador como os limites mudam de cor para dar ao utilizador *feedback* acerca do que está a acontecer (figura 4).

No início da implementação da aplicação não existiam ainda as mãos virtuais para guiar o utilizador no ecrã. Existiam, no entanto, as mesmas zonas descritas anteriormente, delimitadas pelos mesmos limites. Constatou-se que era muito mais difícil para o utilizador colocar a mão na zona desejada, pois tinha apenas o *feedback* da visualização e o auditivo para se guiar (note-se que o *feedback* da visualização não engloba as mãos virtuais do utilizador, assunto abordado em maior detalhe na próxima secção). A partir do momento em que as mãos virtuais foram adicionadas à aplicação, tornou-se bastante mais simples a orientação do utilizador, pois o *feedback* que obtinha era relativo à localização das suas mãos no espaço virtual.

Outra consequência foi a melhoria na precisão. Antes da introdução das mãos virtuais era normal passar de uma zona para outra sem querer, obrigando o utilizador a olhar constantemente para as suas mãos reais para tentar mantê-las no mesmo sítio. Após a introdução das mãos virtuais o utilizador guia-se maioritariamente por elas e olha bastante menos para as suas mãos reais. Este facto sugeriu a possibilidade de ter ainda mais uma ou duas zonas, totalizando cinco ou seis, sem que isso se traduzisse numa perda de precisão das mãos. A hipótese, no entanto, teria de ser corroborada por testes com utilizadores efetivos.

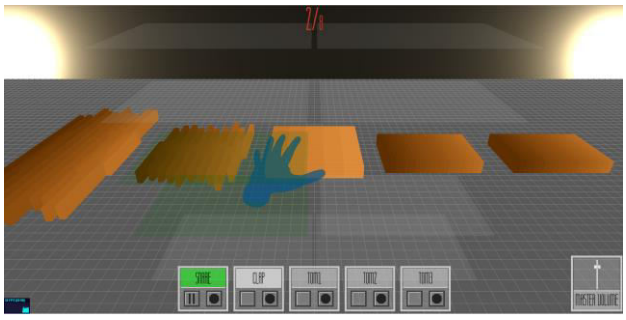

Figura 4 – Interação com uma *track* selecionada

3.2.2 Interface gráfica

A interface gráfica localiza-se na zona inferior da imagem.

À direita existe um slider que permite regular o volume de som. Ao centro estão desenhadas áreas rectangulares, cada uma delas associada a uma *track*. O utilizador, ao selecionar um dos rectângulos, inicia a reprodução do som da *track* correspondente. Em cada rectângulo existem dois botões, um para gravar e outro para fazer *pause* ou *play,* consoante a ação corrente. Ao carregar no botão de gravar a cor da metade superior do rectângulo muda para laranja, indicando que a aplicação se está a preparar para gravar (o início da gravação dá-se no início de cada compasso, o utilizador pode guiar-se pelos números do metrónomo explicados anteriormente para saber quando é que a aplicação vai começar a gravar). Quando a gravação é iniciada, a cor *da metade superior* do rectângulo muda para vermelho. Uma vez acabada a gravação, a cor muda novamente para verde para indicar que a *track* está simplesmente a reproduzir. Neste momento o utilizador pode escolher parar o som da *track* que acabou de gravar. Para isso basta carregar no botão de pausa, o som é parado imediatamente e a cor *da metade superior* do rectângulo muda para castanho para indicar a pausa.

3.3 Visualização

Um dos objectivos da aplicação era conseguir a visualização clara da música criada. Visto que cada *track* toca um som único e distinto, foi também um objetivo desde o início que se conseguisse distinguir os sons de cada *track* na visualização.

Lembrando que a aplicação está dividida em dois tipos de *tracks*, as *tacks* base, que não podem ser alteradas pelo utilizador, e as *tracks* normais que podem ser manipuladas pelo utilizador, foram criados dois tipos de visualização.

Relativamente às *tracks* base, criou-se uma visualização relativa a apenas uma delas, a *track* que contém o som do bombo. Para este efeito o plano vertical ao fundo do espaço tridimensional contém dois círculos posicionados nos cantos inferiores direito e esquerdo, tal como pode ser observado na figura 5, e que reagem em simultâneo, aumentando de tamanho quando é reproduzido o som. Devido à cor que possuem, os círculos assemelham-se a uma luz pulsante. Esta visualização é iniciada assim que o som começa a ser reproduzido.

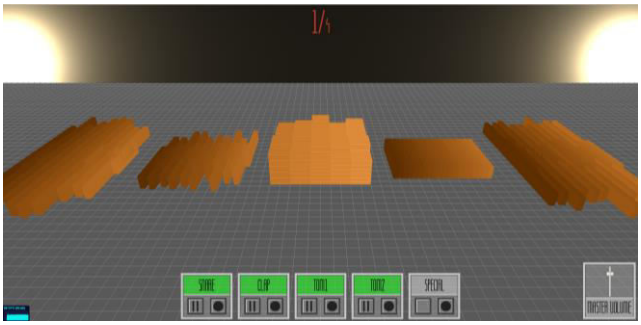

Figura 5 – Vários conjuntos de paralelepípedos a variar o seu tamanho consoante o som tocado

Relativamente às *tracks* normais, que podem ser manipuladas pelo utilizador, existem presentemente cinco, podendo no entanto ser adicionadas mais no futuro. A cada uma corresponde um dos blocos de paralelepípedos que estão dispostos horizontalmente, tal como se pode ver na figura 5.

Os blocos são animados quando é reproduzido o som da respectiva *track* e o seu movimento é função da média das frequências que compõe o som. Para facilitar a identificação da *track* que está a ser reproduzida definiram-se três tipos de animações uma para os blocos dos extremos, outra para o segundo e quarto blocos e uma terceira para o bloco do meio. Com esta distribuição de animações mantém-se a simetria da imagem.

Os blocos dos extremos são compostos por 20 paralelepípedos, com a dimensão maior paralela ao eixo dos *zz*, que efetuam uma mudança de escala segundo esse eixo. O bloco do meio é constituído por um conjunto de 5x5 paralelepípedos e efetua uma mudança de escala segundo o eixo dos *yy*. Finalmente os restantes dois conjuntos de blocos, igualmente constituídos por 20 paralelepípedos, dispostos também com a dimensão maior segundo o eixo dos *zz*, efetuam uma translação segundo este eixo de forma desencontrada, ou seja, se o primeiro paralelepípedo efectua uma translação positiva no eixo dos *zz*, o segundo efectua uma translação negativa e assim sucessivamente. Em todas as visualizações existe sempre alguma aleatoriedade no tamanho dos paralelepípedos quando reagem ao som reproduzido.

A complementar a visualização sobre as *tracks,* apresentam-se no topo da imagem e centrados dois números no formato X/X. O número da esquerda conta de um até quatro e está sincronizado com o som da batida da música, marca portanto o tempo. O segundo número conta de um até dezasseis e representa as notas que podem ser tocadas quando a mão do utilizador está na zona quatro vezes mais rápida - tabela 1.

Mostra-se também, na zona inferior esquerda da imagem, o número de *frames* por segundo (*fps*) em que a aplicação está a correr. Idealmente este valor encontra-se a 60fps. É usada para fazer testes de desempenho.

3.4 Organização dos módulos da aplicação

A figura 6 mostra a organização dos vários módulos que compõem a aplicação. O módulo *main* controla a componente de visualização e a sua ligação com os outros módulos: a componente de interação que se subdivide pela interface gráfica (modulo *Ui*) e pela interação com o dispositivo Leap Motion (módulo *Leap*); e a componente de áudio (módulos associados ao módulo *Audio*). Na componente de áudio temos os módulos relativos à simulação de um metrónomo (módulo *Metronome*), à reprodução de sons (módulo *Sounds*) e ao processo de gravação de *tracks* para composição de novos sons (módulo *Recorder*). Como já foi referido, uma das *tracks* base emite um som contínuo que serve de fundo à aplicação. Como a geração deste som é independente dos outros sons produzidos durante a execução da aplicação, há um módulo especial para este efeito (módulo *Drone*).

4. CONCLUSÕES

Desenvolveu-se uma aplicação interativa para a criação de música a partir de sons base. Esta aplicação integra a combinação de três componentes: áudio, para reprodução, gravação e combinação de sons; interação, com recurso à Leap Motion; e visualização. A aplicação baseada nas tecnologias *HTML5, CSS3* e *javascript,* tem um nível de desempenho que permite a gravação e reprodução de som de forma precisa e em tempo-real.

A aplicação foi construída de forma modular, de modo não só a tornar o código mais legível e a facilitar a manutenção como também a ser mais fácil estender o trabalho que foi desenvolvido até agora, implementando outras funcionalidades.

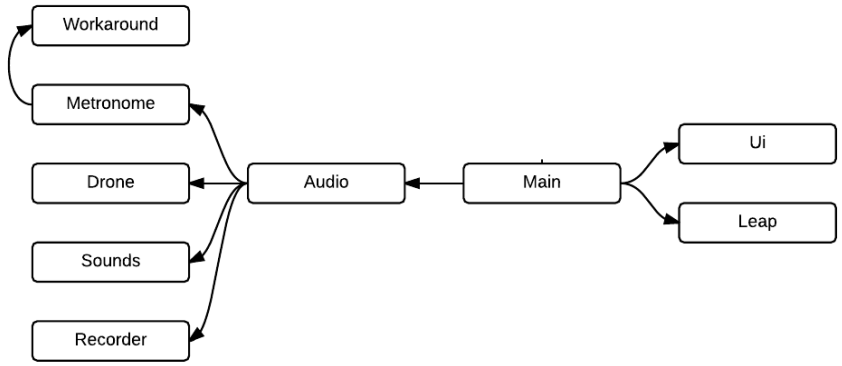

Figura 6 – Organização dos vários módulos da aplicação

O efeito final obtido com a aplicação e particularmente com a junção dos 3 módulos discutidos anteriormente (áudio, *Leap Motion* e Visualização) é algo que irá ser futuramente testado com utilizadores comuns e também com especialistas nestas áreas para obter *feedback* acerca dos vários aspetos que constituem a aplicação. Especificamente, será interessante ter *feedback* da interação do utilizador com a Leap Motion; saber se em termos de visualização esta é percetível e se é claro qual a representação visual que corresponde a cada *track* e, consequentemente, cada som. Será útil ainda saber se a divisão do espaço tridimensional em zonas que têm várias ações associadas foi uma boa escolha em termos de interação com a Leap Motion.

Em termos de composição de sons e da criação de música, pretende-se apurar se o processo usado para a reprodução e gravação de som é fácil de utilizar. Visto que a aplicação tem como utilizador alvo alguém que não tem nenhum tipo de treino musical, gostaríamos de saber qual a sua opinião final relativamente ao resultado da música que criou.

5. TRABALHO FUTURO

A aplicação desenvolvida recorre unicamente a sons de ritmo. No futuro será interessante tentar usar sons com notas, discretos (como um xilofone) e contínuos (como um violino). A utilização deste tipo de sons levantará novos problemas, não só em termos de áudio e do seu processamento (mais desafiante nos sons contínuos do que nos discretos) como também de interação. Como interagir com sons que têm várias notas?

Relativamente aos sons contínuos, afigura-se aconselhável o recurso a sons sintetizados em vez de sons reais, tal como é feito com a maior parte dos sons usados nesta aplicação. Deste modo será mais fácil a mudança de nota sem a perceção pelo utilizador de qualquer problema, como cortes abruptos.

Relativamente à interação com a Leap Motion, a nova versão da *driver* do Leap Motion, lançada recentemente, inclui funcionalidades que permitem simplificar algumas das opções escolhidas para a interação. É possível agora quantificar o fechamento da mão do utilizador. Com esta funcionalidade poderíamos eliminar a zona mais alta associada à mão esquerda e usar o fechamento da mão ou a sua abertura para saber se o utilizador pretende tocar ou não determinado som.

6. AGRADECIMENTOS

Agradecemos à Fundação para a Ciência e Tecnologia (FCT) e ao laboratório de I&D LabMAg o apoio financeiro dado para a realização deste trabalho, ao abrigo do projeto estratégico PEst-OE/EEI/UI0434/2014. Agradecemos também a todos os participantes nos testes.

7. REFERÊNCIAS

[Airspace] Leap Motion Store – Airspace (acedido em 28-07-2014). https://airspace.leapmotion.com/

[BeatPetite] BeatPetite (acedido em 28-07-2014) http://beatpetite.com/.

[Bernhoft] Bernhof (acedido em 28-07-2014) http://bernhoft.org/.

[Biquad Filter] The Biquad Filter Node Interface (acedido em 28-07-2014) http://webaudio.github.io/web-audio-api/#the-biquadfilternode-interface

[Color Particles] Music Colour Particles. jabtunes. (acedido em 28-07-2014) http://jabtunes.com/labs/arabesque/.

[Cube Visualizer] Cube Visualizer (acedido em 28-07-2014) http://shoffing.com/pages/projects/cubevis/.

[Dive] A dive in Music. (acedido em 28-07-2014) http://do.adive.in/music/.

[Dub FX] Dub FX (acedido em 28-07-2014) http://dubfx.net/.

[Leap] Leap Motion (acedido em 28-07-2014). https://www.leapmotion.com/

[Ljósið] Ólafur Arnalds - Ljósið (Official Music Video) (acedido em 28-07-2014) http://vimeo.com/6284199.

[Loop] Loop Waveform Visualizer. Air Tight Interactive. (acedido em 28-07-2014) http://www.airtightinteractive.com/2012/01/loop-waveform-visualizer/

[Garcia11] Garcia, Christopher Michael. Study on the Procedural Generation of Visualization from Musical Input Using Generative Art Techniques. Master Thesis Texas A&M University, 2011.

[Panner] Panner Node Interface (acedido em 28-07-2014) http://webaudio.github.io/web-audio-api/#the-pannernode-interface

[Theremin28] Ssergejewitsch, Theremin Leo. *THEREMIN SIGNALING APPARATUS. US1658953 A* USA, 14 de February de 1928. Grant.

[Tom Thum] Beatbox brilliance: Tom Thum at TEDxSydney (acedido em 28-07-2014) http://www.youtube.com/watch?v=GNZBSZD16cY&feature=youtu.be&t=8m12s

[Tone Craft] Tone Craft (acedido em 28-07-2014) http://labs.dinahmoe.com/#tonecraft.

[ToneMatrix] ToneMatrix. AudioTool (acedido em 28-07-2014) http://tonematrix.audiotool.com/.

[Web Audio API]Web Audio API (acedido em 28-07-2014)http://webaudio.github.io/web-audio-api

[Zoe Keating] Zoe Keating - 'Lost' [HD] Sound Quality, ABC Radio National. Youtube. (acedido em 28-07-2014) http://www.youtube.com/watch?v=sG9H5E2JN3s