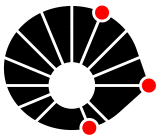
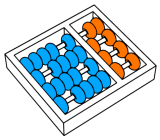




Estudos sobre o InstInt e o Socioenativismo: buscando soluções de áudio para a instalação

João Guilherme Alves Santos
M. Cecília C. Baranauskas
Emanuel Felipe Duarte
Yusseli Lizeth Méndez Mendoza

Relatório Técnico - IC-PFG-23-68
Projeto Final de Graduação
2023 - Dezembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Estudos sobre o InstInt e o Socioenativismo: buscando soluções de áudio para a instalação

João Guilherme Alves Santos[1], M. Cecília C. Baranauskas[1], Emanuel Felipe Duarte[1], Yusseli Lizeth Méndez Mendoza[1]

[1] Instituto de Computação Universidade Estadual de Campinas (UNICAMP), Caixa Postal 6176
13083-970 Campinas-SP, Brasil

j199624@dac.unicamp.br

mccb@unicamp.br

contato@emanuelfelipe.net

yusseli.mendez.m@gmail.com

Resumo. *Este Documento descreve o Projeto Final de Graduação do aluno João Guilherme Alves Santos, aluno de Engenharia de Computação modalidade AA de 2018, ra 199624. O projeto Final de Graduação do aluno foi orientado pela Prof.ª Drª Maria Cecília Calani Baranauskas, professora Titular da Universidade Estadual de Campinas (UNICAMP), afiliada como colaboradora no Instituto de Computação e co-orientado pelo Prof.º Drº Emanuel Felipe Duarte, Doutor em Ciência da Computação pela Universidade Estadual de Campinas e Yusseli Lizeth Méndez Mendoza, mestre em Ciência da Computação pela Universidade Estadual de Campinas.*

O Projeto foi executado no segundo semestre de 2023, através da disciplina MC030 - Projeto Final de Graduação.

Palavras-Chave: InstInt; Sistemas Socioenativos, Computação Ubíqua e Pervasiva.

1. Introdução

O projeto do InstInt é uma instalação socienativa que faz parte de um projeto maior intitulado "Sistemas Socioenativos: Investigando Novas Dimensões no Design da Interação Mediada por Tecnologias de Informação e Comunicação", apoiado pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), processo (#2015/16528-0).

O Projeto de Sistemas Socioenativos, título no qual nos referimos ao projeto, busca criar soluções de design de sistemas socioenativos[1] e sua experimentação em cenários reais com o objetivo de estudar o fenômeno da interação em cenários construídos com tecnologia ubíqua e pervasiva a partir de uma abordagem enativista[4].

O InstInt é uma instalação que permite que crianças e adultos criem uma composição sonora ao tocar 5 fitas luminosas que acionam sons de determinados instrumentos musicais como bateria, flauta, órgão, clarinete e baixo, enquanto a instalação se movimenta[1]. Ele passou por diversas fases do desenvolvimento durante o projeto inicial, como ideação, prototipação, construção em tamanho real, experimentação aberta ao público e outros. O projeto abre possibilidades de estudo sobre o fenômeno socioenativo em cenários de experimentação com pessoas e sobre o efeito da tecnologia de um sistema desse tipo, como ela envolve o social, o emocional e o físico.

Atividade prática realizada no InstInt [1]

Dado esse cenário inicial, alguns espaços da pesquisa podem ser explorados e atuar na continuidade do projeto foi desejo inicial. O plano de estudos do projeto final de graduação se pauta na análise de conceitos de design que se relacionam ao tema e desenvolvimento na instalação socioenativa InstInt, explorando aspectos sonoros da atração com desenvolvimento de novas ferramentas, discussão e planejamento de mudanças arquitetônicas do sistema referentes ao som.

2. Metodologia

Durante o planejamento do projeto, o plano estipulado para o aluno referente ao método de estudo e abordagem do tema era baseado em separar o projeto em três partes principais: Estudo de aspectos da instalação, Conceitos de Design e Desenvolvimento de novas funções.

Porém, dado o aprofundamento do contato do aluno com a instalação e uma melhor contextualização do cenário, a divisão foi simplificada para Estudo de Conceitos pertinentes, Análise da Instalação do InstInt e Discussões e Propostas de novos aspectos e melhorias de código, com foco nos aspectos sonoros da estrutura.

Com isso, os dois primeiros tópicos estão condensados na terceira seção do documento, Estudo de conceitos e Análise da Instalação, onde é ilustrado o processo de primeiro contato com conceitos fundamentais nos quais a instalação se fundamenta e observação do cenário atual da aplicação e da arquitetura da mesma, enquanto na quarta seção, Ferramentas Analisadas, são apresentadas as ferramentas investigadas durante o estudo; na quinta seção, Discussões e Propostas, apresentamos as discussões sobre as análises obtidas e propostas para a instalação. Por fim, a última seção, Conclusão e Projetos Futuros, reflete sobre o desfecho do projeto e possíveis ações futuras.

3. Estudo de conceitos e Análise da Instalação

Neste capítulo é ilustrado o processo de primeiro contato com conceitos fundamentais nos quais a instalação se fundamenta e observação do cenário atual da aplicação e da arquitetura da mesma.

3.0. Sistemas Socioenativos e Computação Ubíqua e Pervasiva.

O estudo e contato com o termo “socioenativo”[2] e o contato com Computação Ubíqua e Pervasiva[3] foram passos primordiais para o entendimento de sistemas que envolvem esses temas. Portanto foi necessário estudo destes temas por parte do aluno através de leituras e contato com projetos que ilustram estes conceitos através do InterHAD, grupo de pesquisa que tem estes tópicos entre as áreas de investigação da equipe.

Como definido na seção de introdução, O InstInt é um sistema socioenativo, um sistema computacional composto de artefatos físicos e sistemas digitais que possibilitam que as relações intersubjetivas (atenção conjunta, ação conjunta e coordenação) ocorram dentro de um cenário específico de interação. Nestes cenários, a interação com a tecnologia envolve todo o corpo em seus aspectos físicos, emocionais e sociais. Seria então a visão da interação do organismo com o ambiente como fundamental para a cognição, não apenas um processo mental interno, mas sim moldada pela percepção do organismo no ambiente em união com a inclusão de aspectos sociais na formação da cognição.

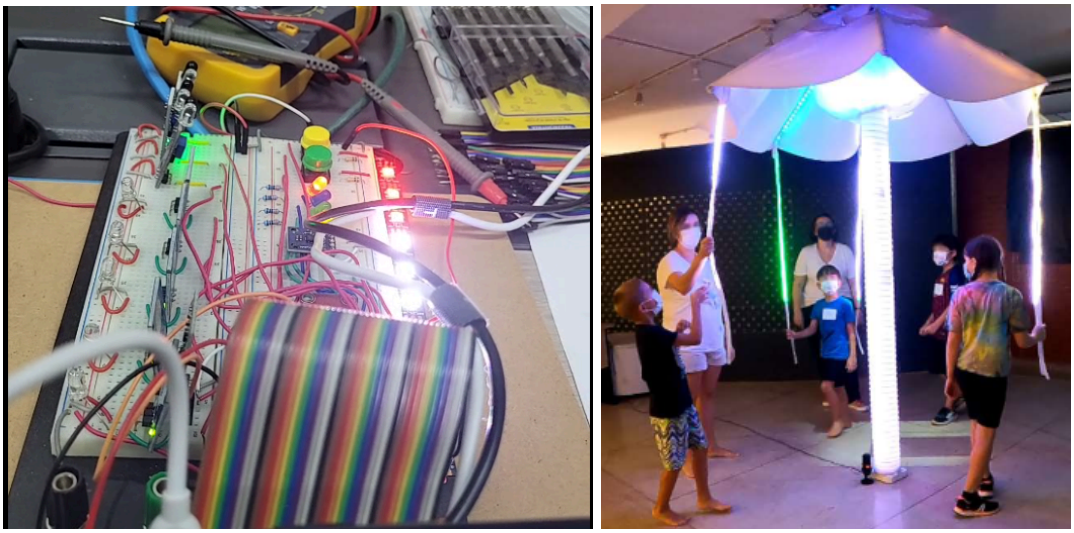
Já o conceito de Computação Ubíqua e Pervasiva, também referidos como Computação Pervasiva e Ubicomp, se pauta fundamentalmente nos conceitos propostos por Mark Weiser[3] e são relacionados pela integração da tecnologia no ambiente de maneira profunda e difusa, onde a tecnologia da computação deve ser onipresente e estar integrada de maneira sutil e transparente no cotidiano humano.

3.1. Instalação InstInt

Como mencionado na seção anterior, O projeto do InstInt é uma instalação socioenativa e um trabalho que faz parte de um projeto maior intitulado 'Sistemas Socioenativos:

Investigando Novas Dimensões no Design da Interação Mediada por Tecnologias de Informação e Comunicação.'. É uma instalação que permite que crianças e adultos criem uma composição sonora ao tocar 5 fitas luminosas que acionam sons de determinados instrumentos musicais como bateria, flauta, órgão, clarinete e baixo, enquanto a instalação se movimenta.

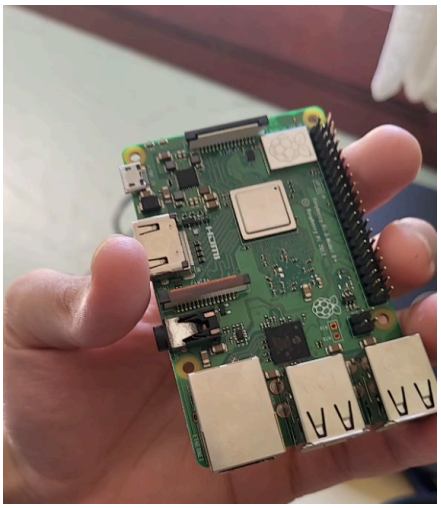
Atualmente há um protótipo e uma versão em tamanho real da instalação.



Protótipo e instalação em tamanho real do InstInt[1]

O protótipo, localizado no Laboratório do InterHAD no Instituto de Computação da Unicamp (Universidade Estadual de Campinas), é composto por diferentes sensores de aproximação e toque, controlados por um dispositivo Raspberry Pi 4; é composto por dispositivos de emissão de sons e luz, botões e sensores para simular as ações de toque e aproximação semelhantemente a ações na instalação em tamanho real, que está localizada no Museu da Unicamp (Universidade Estadual de Campinas).

Durante toda a etapa de produção do projeto final de graduação, o protótipo se mostrou útil para o desenvolvimento e teste do cenário atual e de novas funcionalidades no sistema. Além disso, foi disponibilizado para o aluno um dispositivo Raspberry Pi 3 para desenvolvimento em um ambiente semelhante ao presente no Protótipo.



Raspberry Pi 3 disponibilizada para o projeto [1]

3.2. Literatura e Estado Atual do Sistema

O relatório técnico de título “InstInt: Design e Desenvolvimento de uma instalação Socioenativa.”[1] foi o ponto de partida para a contextualização acerca de todos os detalhes sobre a instalação. Nele está presente o processo do design e desenvolvimento da instalação, contendo o processo de construção em escala pequena e escala real e uma descrição detalhada sobre funcionalidades e dispositivos presentes na atração.

O trabalho relata o papel habilitador da instalação em relação ao fenômeno socioenativo e cita alguns possíveis caminhos futuros da plataforma, na qual o termo motivador para o projeto foi a exploração computacional acerca dos efeitos sonoros do sistema.

3.3. **Configurações e softwares da instalação**

Os arquivos para configuração, testes e código fonte da instalação são mantidos em repositório Git[6], nele há o procedimento para configurar a instalação e garantir o funcionamento dos aparelhos em um dispositivo Raspberry Pi.

O código principal, escrito em linguagem de programação Python[7], realiza a integração dos dispositivos como um todo, sejam dispositivos sonoros, luminosos ou outros. O script segue como princípio o diagrama de estados[?[para funcionamento da instalação.

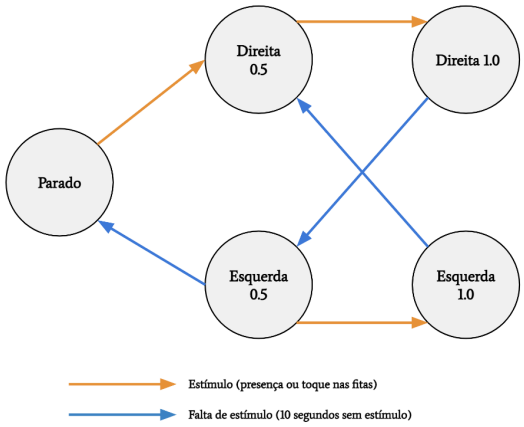


Diagrama de estados da aplicação [6]

Em relação aos efeitos sonoros da aplicação, ponto principal de observação desta pesquisa, a aplicação é responsável por reproduzir áudios no formato .wav (WAVEform audio format), que são carregadas através da biblioteca Pydub[8] para manipulação do áudio. Cada música do InstInt é composta na totalidade em 6 faixas de áudio, onde cada faixa é disparada no momento em que é detectado um toque em uma das fitas sensoriais do sistema, sendo reproduzidas através do player de áudio padrão do sistema no console.

Atualmente a implementação do código faz com que a instalação suporte apenas entradas de áudio renderizadas que possam ser reproduzidas como arquivos .wav, onde é realizada a reprodução do áudio através de uma chamada do sistema pelo player padrão coletado pela biblioteca Pydub, onde é reproduzida uma faixa de música assim que um evento de toque no sensor é registrado.

4. Ferramentas Analisadas

Como metas iniciais do projeto, foram traçados os objetivos de adicionar uma nova música na instalação que fosse relacionada a uma temática semelhante a músicas tocadas em carrosséis, e estudar ferramentas que possibilitam uma personalização do som em si, tornando acessível novas *features* na instalação.

Com esses objetivos em mente, foram estudadas ferramentas e aplicações para auxiliar em ambos os casos, além de outras ferramentas que pudessem auxiliar em outros processos refletindo em ideias futuras para a aplicação. As principais ferramentas estudadas durante esse período são apresentadas na sequência.

4.0. Sonic Pi e FoxDot

O SonicPi[9] é um ambiente de programação musical escrito em linguagem Ruby, projetado para tornar a criação de música acessível a programadores e entusiastas da música eletrônica. A biblioteca é utilizada para permitir que os usuários criem composições musicais através de código. O software oferece uma abordagem interativa e em tempo real, possibilitando que os músicos programem padrões, ritmos e melodias de forma dinâmica. Dada sua construção e linguagens distintas da utilizada no InstInt e não ter suporte de carregamento no formato .wav, há dificuldade elevada na integração com a instalação e adaptação da biblioteca com a proposta atual do projeto.

Já o Foxdot[10] é uma biblioteca de programação musical construída sobre a linguagem de programação Python que fornece uma abstração do SuperCollider[11], biblioteca consolidada para criação de ambientes de programação com funções parecidas, também destinada a criar música algorítmica em tempo real. Essa tem uma maior facilidade em relação a integração, visto que a linguagem de programação é a mesma, porém também sofre dos problemas que envolvem o formato do áudio e não receberá mais atualizações de código no futuro, diferentemente de outras opções observadas durante o estudo.

4.1. Music21 e o Arvo

O Music21[12] é uma biblioteca de código aberto escrita em Python, projetada especificamente para análise musical e manipulação de partituras. Desenvolvida por pesquisadores do MIT, essa ferramenta é amplamente utilizada por músicos, pesquisadores e programadores interessados em explorar a estrutura e os elementos musicais de composições.

O Music21 oferece uma gama abrangente de funcionalidades, desde a leitura e escrita de partituras até análises avançadas, como extração de acordes, detecção de intervalos e reconhecimento de padrões. Por ser integrada com Python pode proporcionar uma abordagem adequada para a criação de novas funções na instalação, desde que o formato do áudio fornecido seja condizente com os formatos suportados pela biblioteca.

Além disso, a biblioteca é suportada por uma comunidade ativa, o que significa uma maior variedade de recursos, materiais de estudo e exemplos práticos e até mesmo criação de novas funções.

Já o Arvo[13] é uma biblioteca de composição musical procedural desenvolvida por Dr. Georges Dimitrov, professor de composição na Concordia University. Essa biblioteca é baseada no framework music21, apresentado anteriormente, e é um bom exemplo de como a biblioteca anterior pode ser habilitadora para o estudo de composições e personalização de músicas, um dos focos do estudo do projeto final de graduação.

4.2. Mido

O Mido[14], abreviação para MIDI Objects, é uma biblioteca colaborativa em Python para trabalhar com portas, mensagens e arquivos MIDI 1.0. Assim como o Music21, porém mais focada no formato de arquivo midi, a ferramenta permite a criação, manipulação e análise de arquivos MIDI de forma eficiente. Além disso, o Mido permite que os desenvolvedores possam facilmente gerar sequências musicais, processar eventos MIDI, e interagir com dispositivos MIDI em tempo real.

A Mido oferece vantagens ao simplificar a complexidade natural do formato MIDI, tornando mais fácil criar e editar dados musicais. Adicionalmente, a biblioteca suporta funcionalidades avançadas, como a personalização de tipos de mensagens MIDI e a interação com hardware musical, como instrumentos com saídas digitais no formato, o que a torna uma opção valiosa para projetos que demandam a manipulação de dados MIDI em ambientes de programação Python.

4.3. Rtmidi

O Rtmidi[15] por si só é um conjunto de classes em C++ (RtMidiIn, RtMidiOut e classes específicas da API) que oferece uma common API (Interface de Programação de Aplicações) para entrada e saída MIDI em tempo real em sistemas Linux (ALSA e JACK), Macintosh OS X (CoreMIDI e JACK), Windows (Multimedia Library e UWP), Web MIDI, iOS e Android. Essa biblioteca simplifica significativamente o processo de interação com hardware e software MIDI em computadores.

Pensando em sua integração com o InstInt, há uma biblioteca que serve para possibilitar conectar o Rtmidi com Python, a python-rtmidi[16]. Ela permite a entrada e transmissão de mensagens MIDI de forma facilitada, possibilitando integrar facilmente funcionalidades MIDI em projetos Python e explorar sua capacidade de personalização de arquivos de áudio, possibilitando, por exemplo, a criação de aplicativos musicais interativos.

4.4. FluidSynth e Pygame

As duas ferramentas apresentadas nessa seção foram bibliotecas que foram estudadas para reprodução de arquivos MIDI. O motivo pelo qual esse tipo de ferramenta foi analisada é pelo fato de que arquivos MIDI e WAV têm naturezas muito distintas, assunto debatido na seção 5.2. Tanto o Fluidsynth[17] quanto o Pygame[18] são capazes de ler, manipular e reproduzir arquivos MIDI.

O FluidSynth é um sintetizador de software open-source que converte arquivos MIDI em áudio, permitindo a reprodução de instrumentos musicais sintetizados em computadores. Funciona através da utilização de SoundFonts, arquivos que definem um grupo de amostras de áudio de instrumentos reais e as utilizam para criar uma ampla variedade de sons. O FluidSynth é comumente utilizado para gerar faixas de áudio a partir de arquivos MIDI, oferecendo flexibilidade na escolha de instrumentos e ajustes de som.

Por outro lado, o Pygame é uma biblioteca de programação em Python projetada para criação de jogos e aplicações multimídia. Embora a ferramenta seja conhecida por suas funcionalidades gráficas e de interação com o usuário, ela tem destaque nesse aspecto

por possuir suporte para manipulação de arquivos MIDI. Com o Pygame, é possível carregar, reproduzir e manipular arquivos MIDI em projetos proporcionando uma integração fácil de elementos musicais em projetos interativos.

4.5.midi2audio

A biblioteca midi2audio é uma ferramenta em Python que oferece funcionalidades para converter arquivos MIDI em arquivos de áudio em diversos formatos, como WAV, usando geralmente o sintetizador FluidSynth para processar o arquivo MIDI e gerar o áudio correspondente.

O principal objetivo da biblioteca midi2audio é fornecer uma interface simples para realizar essa conversão de maneira fácil e eficiente para integrar a funcionalidade de converter áudios em seus scripts ou projetos Python para automatizar o processo de conversão de arquivos MIDI para áudio, facilitando a incorporação de trilhas sonoras ou efeitos sonoros em aplicações multimídia, jogos ou outras produções que exigem áudio a partir de dados MIDI.

5. Discussões e Propostas

Os objetivos iniciais do projeto foram adicionar uma nova música na atração que fosse relacionada a uma temática semelhante a músicas tocadas em carrosséis e estudar ferramentas que possibilitam uma personalização do som em si e tornam possível a criação de novas features na instalação.

Após o estudo de ferramentas descrito na seção anterior, foram feitas as discussões e propostas de alterações para a instalação.

5.0. Estudo de Músicas de Carrosséis

O objetivo deste tópico foi a criação de uma música nova para o InstInt com o tema de músicas para carrosséis. Durante uma longa pesquisa que envolveu uma análise extensa acerca da musicalidade desse tipo de atração, foi observado que não há uma música

específica universalmente associada a carrosséis, pois a escolha da música pode variar de acordo com o local, o evento e as preferências dos operadores.

Analisando cenários de carrosséis tradicionais famosos no mundo todo, foi observado que em alguns casos, como o caso do King Arthur Carrousel, criado em 1922 na Disneyland em Anaheim, Califórnia, os carrosséis reproduzem temas de peças clássicas de animações e contos. Já em outros casos, que se mostraram mais interessantes para a pesquisa, como o Jane's Carousel, criado em 1922 no Brooklyn, New York, são reproduzidas músicas interpretadas por um Fairground organ, órgão musical que cobre as seções de sopro e percussão de uma orquestra. Sua concepção visa gerar um volume de som alto para se destacar perante multidões e entre brinquedos e atrações, criado para acompanhar principalmente carrosséis Além desses tipos de órgãos, alguns outros instrumentos semelhantes também são utilizados em outros casos, como o Calliope.

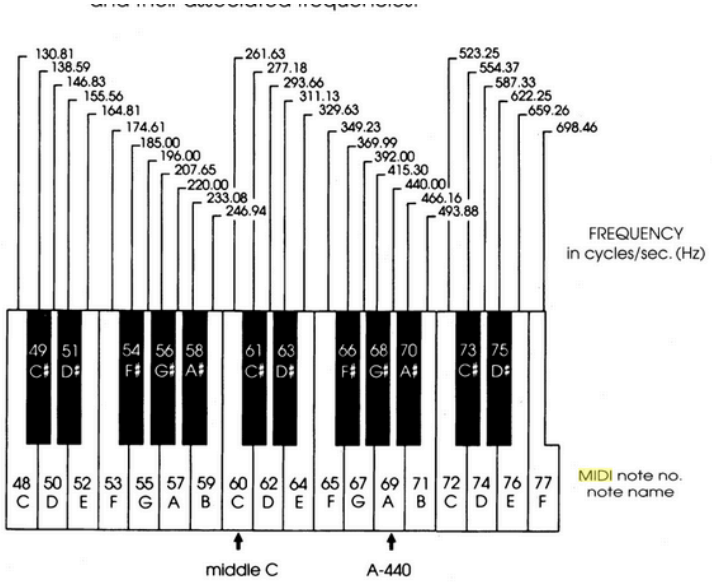
Através do estudo para encontrar um som com características semelhantes aos sons criados para carrosséis, foi possível imaginar algumas músicas que concordam com o modelo almejado, porém elas ainda precisavam ter um formato no qual o áudio pudesse ser dividido nos moldes do InstInt, em 6 faixas de música. Dado o desafio e o estudo das ferramentas durante o projeto, foi desenvolvido um código com a biblioteca Music21 para auxiliar na adição de novas músicas, descrito na próxima seção.

5.1. Script para separação de instrumentos em arquivos MIDI.

Com o desejo de adicionar uma nova música no InstInt que fosse relacionada com efeitos sonoros característicos de carrosséis somada com a dificuldade em encontrar músicas que atendessem estas características no formato correto com os instrumentos divididos em 6 faixas de áudio distintas, surgiram dois caminhos distintos: Compor uma nova música para a instalação ou encontrar maneiras de dividir uma música já existente.

Compor uma música era possível, porém o estudo das ferramentas que possibilitam manipular partituras mostrou-se um caminho promissor para esse tipo de atividade.

Sendo assim, foi criado um script[22] em linguagem Python e utilizando a biblioteca Music21 para manipular músicas em formato MIDI, com o objetivo de ser um algoritmo capaz de separar instrumentos de áudio de qualquer música em formato midi em 6 diferentes faixas de áudio, tornando-se apta para a atração.



Teclas de piano, notas, nomes, números de nota MIDI e suas frequências associadas.[19]

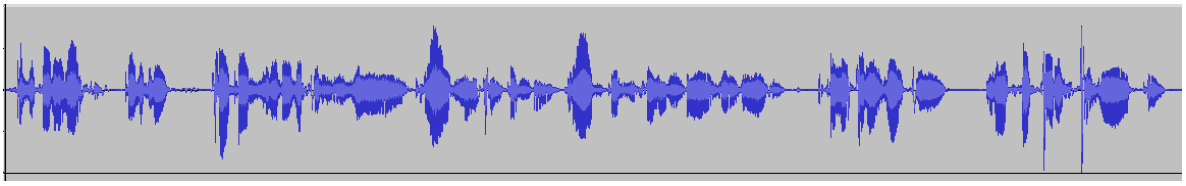
Utilizando as funções de carregamento de áudio, gravação de arquivos e de manipulação de instrumentos da biblioteca Music21 o script criado é capaz de dividir qualquer música em formato .mid em diversas faixas de áudio de cada um dos instrumentos contidos na música e, após a captura destas faixas de som de cada instrumento, uni-las em até 6 faixas distintas.

Para demonstração do funcionamento durante a pesquisa, o código foi executado em uma música em formato .mid e com temática alinhada com a desejada. O algoritmo foi capaz de detectar 7 faixas distintas de instrumentos e criou 6 arquivos .mid onde os seis primeiros foram colocados em 6 arquivos de áudio diferentes e o sétimo foi colocado juntamente ao primeiro. Por fim, como a arquitetura atual da instalação não permite reprodução direta de arquivos .mid, foi estudada a conversão de arquivos para .wav.

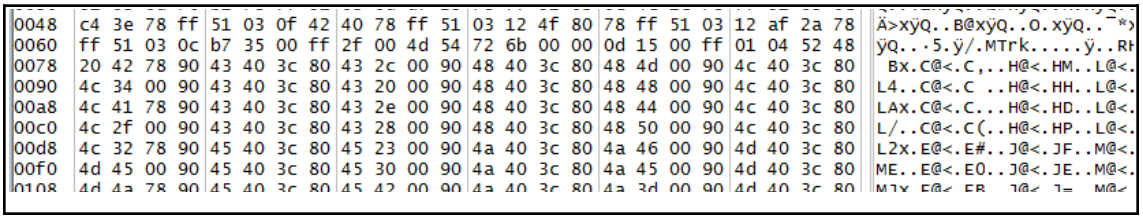
5.2. Comparação MIDI para WAV e conversor de formatos

Em um primeiro contato, é natural pensar que arquivos .mid e .wav são arquivos relativamente parecidos por se tratarem de arquivos que envolvem áudio e, portanto, a conversão de arquivos de um formato para o outro se trataria de algo trivial, porém não é. Há diversas diferenças entre os dois formatos que não só ilustram quão distintos são ambos os formatos, quanto expõem o quão ineficiente é trabalhar com ambos no mesmo domínio.

A reprodução dos dois tipos de dados em um mesmo player ou até mesmo a transformação de arquivos MIDI para WAV apresenta desafios significativos devido à natureza fundamentalmente diferente desses dois formatos. Enquanto arquivos MIDI contêm informações sobre eventos musicais, como notas, ritmo e dinâmica, mas não carregam áudio real, o formato WAV e outros formatos semelhantes de áudio armazenam amostras de áudio digital, representando diretamente as ondas sonoras.



Formato de onda de uma gravação de áudio capturada via Audacity[20]



0048	c4	3e	78	ff	51	03	0f	42	40	78	ff	51	03	12	4f	80	78	ff	51	03	12	af	2a	78	Ä>xÿQ..B@xÿQ..O.xÿQ..¯*x
0060	ff	51	03	0c	b7	35	00	ff	2f	00	4d	54	72	6b	00	00	0d	15	00	ff	01	04	52	48	ÿQ..·5.ÿ/.MTrk.....ÿ..RH
0078	20	42	78	90	43	40	3c	80	43	2c	00	90	48	40	3c	80	48	4d	00	90	4c	40	3c	80	Bx.C@<.C,..H@<.HM..L@<.
0090	4c	34	00	90	43	40	3c	80	43	20	00	90	48	40	3c	80	48	48	00	90	4c	40	3c	80	L4..C@<.C ..H@<.HH..L@<.
00a8	4c	41	78	90	43	40	3c	80	43	2e	00	90	48	40	3c	80	48	44	00	90	4c	40	3c	80	LAx.C@<.C...H@<.HD..L@<.
00c0	4c	2f	00	90	43	40	3c	80	43	28	00	90	48	40	3c	80	48	50	00	90	4c	40	3c	80	L/..C@<.C(..H@<.HP..L@<.
00d8	4c	32	78	90	45	40	3c	80	45	23	00	90	4a	40	3c	80	4a	46	00	90	4d	40	3c	80	L2x.E@<.E#..J@<.JF..M@<.
00f0	4d	45	00	90	45	40	3c	80	45	30	00	90	4a	40	3c	80	4a	45	00	90	4d	40	3c	80	ME..E@<.E0..J@<.JE..M@<.
0108	4d	4a	78	90	45	40	3c	80	45	42	00	90	4a	40	3c	80	4a	3d	00	90	4d	40	3c	80	MJx.E@<.EB..J@<.J=..M@<.

Trecho de um arquivo de formato MIDI

O MIDI necessita de um reprodutor do som, para interpretar as notas musicais e gerar o som, dependendo do próprio player ler os comandos em MIDI, que representam instruções para instrumentos musicais virtuais ou hardware MIDI sobre como gerar som; desta forma a qualidade do instrumento e do som depende do dispositivo utilizado e do interpretador. Já no caso do WAV, o arquivo representa o som real e não requer interpretação adicional, podendo ser reproduzido diretamente sem a necessidade de instrumentos virtuais ou hardware específico.

Ou seja, de uma maneira geral é como se um arquivo MIDI contém instruções de como interpretar a música, enquanto wav e outros tipos semelhantes já contém a música renderizada em seu arquivo, e é por isso que muitas das bibliotecas observadas no projeto forneciam tantas funções para realizar alterações de notas, tempo e altura para arquivos midi e não apresentavam a mesma gama de possibilidades para arquivos do segundo tipo.

Como o tipo de arquivo lido atualmente no InstInt é arquivos .wav, e o script criado para separação de instrumentos da seção anterior cria arquivos de formato MIDI, naturalmente foram estudadas maneiras de converter áudios de .mid para .wav. Foi observado que tipo de conversão pode ser realizada facilmente tanto através de aplicações gratuitas web quanto através de scripts utilizando a biblioteca midi2audio apresentada; porém, não é um processo instantâneo e rápido, sendo improvável pensar na possibilidade de converter o áudio em tempo real durante algum evento da instalação.

5.3. Proposta de alteração de arquitetura.

Como observado durante os tópicos presentes na seção de Discussões e Propostas, há vantagens e desvantagens no modelo atual em relação ao carregamento, manipulação e reprodução de efeitos de áudio em formato WAV.

A reprodução de arquivos WAV pode ser suficiente em cenários onde é necessária uma maior fidelidade sonora, garantia de que o som será reproduzido da mesma maneira independente do player utilizado e não há muita necessidade de manipulação musical dinâmica. Já se a flexibilidade musical é uma prioridade, especialmente para aplicações onde se deseja manipular dinamicamente a música, a reprodução de arquivos MIDI é geralmente considerada mais fácil e versátil.

Desta forma, uma das conclusões do projeto final de graduação é uma proposta de alteração da arquitetura responsável pela reprodução dos sistemas sonoros no InstInt, visto que uma troca de ferramentas de carregamento e reprodução de áudios compatíveis com .wav para ferramentas compatíveis com arquivos .mid, possibilitaria

alterações no áudio que envolvem mudança de tom, instrumentos, volume, tempo e outros.

Para isso, os estudos das ferramentas analisadas no processo se mostram caminhos promissores para a realização desse novo modo de reprodução de efeitos sonoros no InstInt, sendo possível, por exemplo, criar novas músicas utilizando o script de separação de instrumentos para gerar arquivos em .mid, utilizar bibliotecas como o Music21 para ler e manipular estes áudios em tempo real e reproduzi-los utilizando Pygame ou FluidSynth.

Além dessa mudança de escopo para mudar os arquivos lidos de .wav para .mid, há também um anseio de estudar formas de utilizar as bibliotecas como mido e rtmidi, juntamente com um player como o FluidSynth, para interpretar a música midi em tempo real, possibilitando alterações na música que está tocando na medida em que o sistema recebe interações do InstInt.

5.4. Atualizações de Desenvolvimento

Além da proposta de mudança de arquitetura do ponto de vista da reprodução de efeitos sonoros, há algumas mudanças de desenvolvimento que foram adotadas e outras que são recomendadas para o sistema. Entre elas, as mudanças realizadas são a criação de uma branch de desenvolvimento no git do código fonte da plataforma e a criação de um requirements.txt, que é um arquivo comumente utilizado em projetos Python para especificar as dependências do projeto. Ele fornece uma lista detalhada de bibliotecas externas (pacotes) e suas versões específicas necessárias para executar o código do projeto de maneira consistente.

Já como mudanças futuras, há a possibilidade de recomendar um ambiente de desenvolvimento na documentação (virtualenv), setar um argumento “help” no add_argument do código para ilustrar melhor as opções disponíveis de modo de inicialização do script e estudar maneiras de automatizar as configurações que não envolvem Python, como download de drivers e softwares para funcionamento do algoritmo.

6. Conclusão e Projetos Futuros

O resultado do projeto foi um estudo acerca de músicas com o tema de carrosséis, onde foi possível analisar que esse tipo de atração tem como característica músicas interpretadas por instrumentos como Fairground Organ ou Calliopes e, com a utilização da biblioteca Music21, foi possível criar uma nova música para a atração separando as faixas de instrumento em um arquivo .mid através de um script em Python.

Além disso, foram analisados os formatos de música MIDI e WAV, conversões de áudios de MIDI para WAV e possíveis benefícios de reprodução de cada um deles no sistema socioenativo. Em suma, optar por arquivos MIDI habilita aos desenvolvedores a vantagens em relação a flexibilidade na manipulação e personalização das composições em tempo real, além de algumas mudanças pontuais do ponto de vista de desenvolvimento de algoritmos que foram adotados no projeto.

Pensando em atividades futuras, é observável que a separação das informações musicais da representação de áudio real permite ajustes precisos nas nuances da música, como a seleção dinâmica de instrumentos, controle de velocidade e adição ou remoção de faixas, proporcionando um nível de expressividade e criatividade superior. Em resumo, a adoção de arquivos MIDI emerge como uma abordagem estratégica para tornar o InstInt uma experiência musical ainda mais adaptável e personalizada.

Além disso, outros pontos futuros que podem ser explorados são a recomendação de ambientes de desenvolvimentos virtuais, mudança nas configurações de argumentos para maior clareza das opções de execução do sistema, e automatização de configurações de sistema necessárias para o funcionamento do projeto além das ferramentas Python.

Referências

[1] Baranauskas, M. Cecilia & Mendoza, Yusseli & Duarte, Emanuel Felipe. (2021). InstInt: Design e Desenvolvimento de uma instalação Socioenativa.

[2] Baranauskas, M. Cecilia & Mendoza, Yusseli & Duarte, Emanuel Felipe. (2021). Designing for a socioenactive experience: A case study in an educational workshop on deep time. International Journal of Child-Computer Interaction. 29. 100287. 10.1016/j.ijcci.2021.100287.

[3] Weiser, M. The Computer for the 21st. Century. Sci. Am. 265(3):94-104, 1991.

[4] Varela, F., Thompson, E., and Rosch, E. (1993). The Embodied Mind: Cognitive Science and Human Experience. MIT Press. Cognitive science: Philosophy, psychology.

[5] Grupo de Pesquisas InterHAD: https://interhad.nied.unicamp.br/

[6] Repositório InstInt https://github.com/efduarte/instint

[7] Documentação Python: https://www.python.org/

[8] Documentação Pydub: https://pypi.org/project/pydub/

[9] Documentação Sonic Pi: https://sonic-pi.net/

[10] Documentação FoxDot: https://github.com/Qirky/FoxDot

[11] Supercollider: https://supercollider.github.io/

[12] Music21: https://web.mit.edu/music21/

[13] Arvo: https://github.com/georgesdimitrov/arvo

[14] Mido: https://mido.readthedocs.io/en/stable/

[15] Rtmidi: https://www.music.mcgill.ca/~gary/rtmidi/

[16] python-rtmidi: https://pypi.org/project/python-rtmidi/

[17] FluidSynth: https://www.fluidsynth.org/

[18] pygame: https://www.pygame.org/docs/

[19] Rothstein, Joseph. MIDI: A comprehensive introduction. Vol. 7. AR Editions, Inc., 1995.

[20] Audacity: https://www.audacityteam.org/

[21] midi2audio: https://pypi.org/project/midi2audio/

[22] Santos, João Guilherme Alves (2024). “Data from the report: Estudos sobre o InstInt e o Socioenativismo: buscando soluções de áudio para a instalação” REDU - Repositório de Dados de Pesquisa da Unicamp. https://doi.org/10.25824/redu/GWZA2J