

[habr.com](https://habr.com)

# Как создать эвристический алгоритм онлайн-мастеринга и получить предупреждение от RIAA

Сергей Гришаков

36–51 minutes

---

Добрый день, меня зовут Сергей. В своей статье я бы хотел осветить тему аудио мастеринга, а именно: автоматизированного онлайн-мастеринга музыки.

Я расскажу о своём пути от продюсера психоделического транс до мейнтейнера самой популярной open source библиотеки автоматизированного референсного мастеринга на Python, получившей предупреждение от американской ассоциации звукозаписывающих компаний RIAA ([прусф](#)).

Простыми словами, [аудио мастеринг](#) - это финальная стадия производства музыкальной композиции. Для электронной танцевальной музыки я выделяю эти основные этапы мастеринга:

- Задать определённую окраску звучания (эквализация, сатурация)
- Придать глубины и пространства (работа со стереобазой)
- Сделать композицию как можно громче, не внося искажения (левелинг, компрессия, лимитинг)

Разработчикам должна быть понятна подобная аналогия: аудио

мастеринг - это функция, превращающая один массив байт (WAV файл) в другой. Длительность (длина массива) и общее содержание файла сохраняются, меняются только некоторые нюансы в звучании.



Фото pvproductions

С этой рутинной сталкиваются абсолютно все композиторы и артисты, желающие выпускать свою музыку на цифровых площадках (Spotify, Apple Music) или каких-либо носителях (CD, винил). В большинстве случаев эти задачи решаются с помощью [VST плагинов](#) таких известных производителей, как [iZotope](#), [Waves Audio](#), [FabFilter](#) и других. Это [довольно крупная сфера разработки ПО](#), работники которой умело наживаются на бедных музыкантах.

В наше время существует множество онлайн-сервисов, пытающихся автоматизировать процесс аудио мастеринга тем или иным способом. Я обязательно приведу их список ниже.

У рядового музыканта теперь нет необходимости покупать и изучать сложные инструменты для ручного рутинного аудио мастеринга: зайти на сайт, загрузи свой микс, подожди пару минут и скачай результат.


---



Арт хорошего друга

В этой статье пойдёт речь об опыте создания подобного сервиса, а также его переезде в open source в виде Python (PyPI) библиотеки и контейнеризированного веб-приложения для Docker. Устраивайтесь поудобнее, текста будет очень много.

## Обо мне

 Можете смело проматывать до следующего раздела, если вам неинтересна история прокрастинирующего студента, остальные - велком.

Во мне всегда боролись два начала: техническое и творческое. Мама-учитель видела меня будущим успешным физиком или программистом. Папа-актёр считал, что моя судьба - стать великим мировым музыкантом.

Этот внутриличностный конфликт начался еще в школе, в 2004 году, когда один мой знакомый с локального IRC-сервера показал мне свои поделки в [Cubase](#). В то время я был всецело поглощён музыкой таких исполнителей [психоделического транса](#), как Infected Mushroom, Astral Projection, Vibe Tribe. Посетив пару тематических мероприятий, я понял, что хочу так же. Хочу стоять



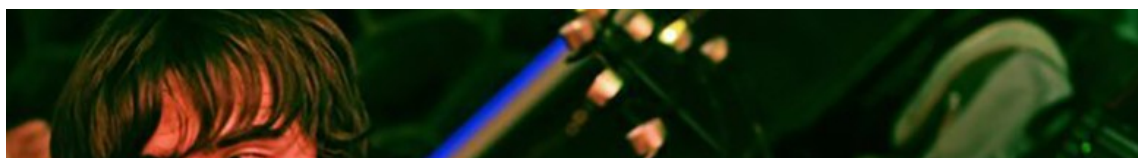
на сцене и играть свою музыку, видеть восторженные глаза слушателей, испытывать это непередаваемое чувство.



Сказочная, непередаваемая атмосфера псайтранс пати

Cubase мне дался не сразу, все-таки в отличие от таких программ, как Photoshop, методом "научного тыка" было довольно сложно добиться чего-то вменяемого. Я начал покорение этой вершины: вместо подготовки к ЕГЭ я читал книги по Cubase, вместо репетиторов по физике и математике, я уделял всё внимание изучению тематических форумов и общению с музыкантами и диджеями в ICQ и MSN Messenger.

Со временем начало получаться: свой собственный full-on psytrance, ура! Подготовив материал на половину альбома и выпустив несколько EP, я сумел пробиться на сцену - связи в ICQ сделали своё дело. Мое первое настоящее выступление: лес, оупен-эйр, волшебная атмосфера и я, играющий свою музыку, а следом - второе, третье... Конечно же, это совпадало с выпускными экзаменами, что не могло не сказаться на успеваемости.



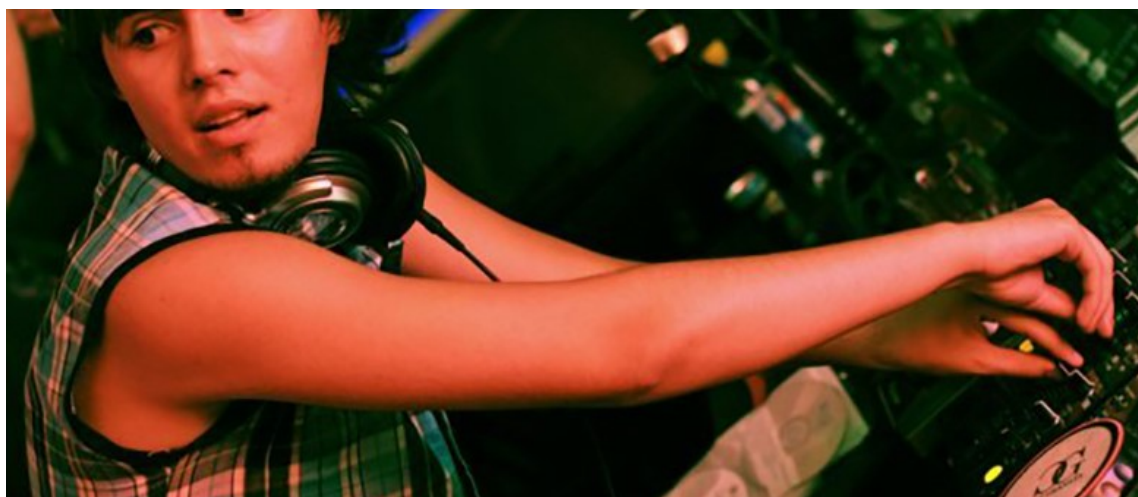


Фото из клуба Cvet Granata г. Москва (2010)

Олимпиада "Шаг в будущее" - завалена, прошли все мои одноклассники, кроме меня. На поступление в Бауманку не хватило баллов. Результат: Московский институт электроники и математики. В тот момент я начинал понимать, что усидеть на двух стульях вряд ли получится.

Дальше начался первый курс, мои выступления продолжались, количество материала росло. В 2009 году мне удалось договориться с австралийским лейблом Sundance Records о выпуске своего первого альбома. Настоящий альбом на CD тиражом в 1000 копий. [Выпуск состоялся в 2010 году](#), мне даже прислали 50 дисков, это стало моей личной победой.



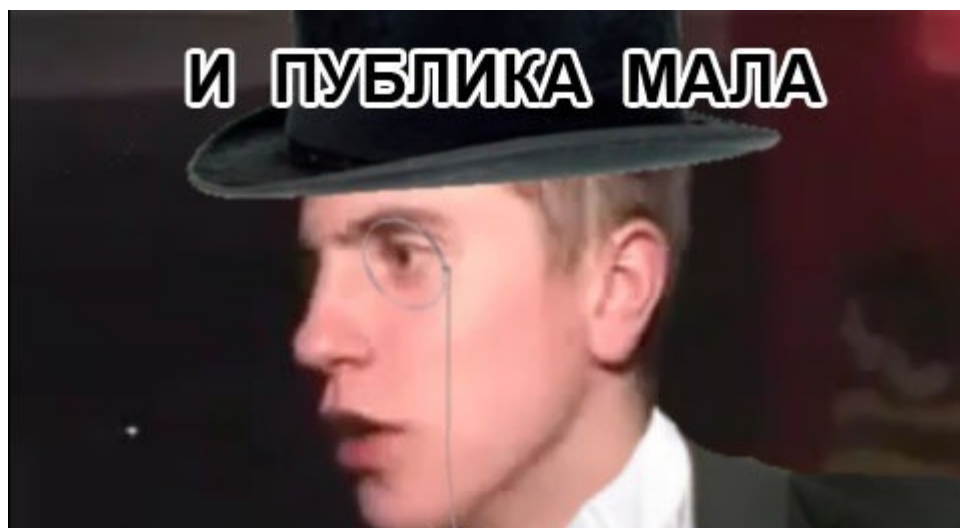


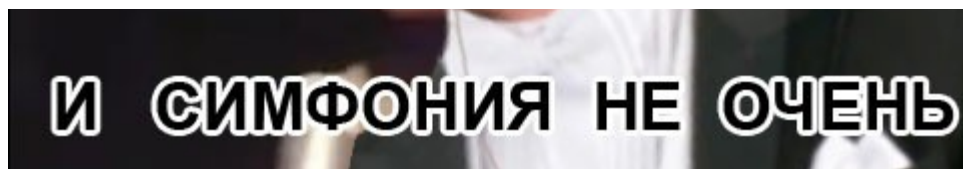
## Explicit – Inspiration Feeling (2010)

Со временем я стал сомневаться в своём занятии:

- Субкультура психоделического транса была очень малочисленна, в ней при желании было бы не так сложно добиться "потолка", но это бы не гарантировало стабильный заработок
- Я стал ощущать давление от местной атмосферы, восторженные глаза слушателей превратились для меня в глаза торчков под экстази, у меня стало появляться чувство, что я направляю свои ресурсы не в то русло
- [Психоделический транс](#) в большинстве случаев - это набор случайных звуков под монотонный бас, но моя музыка была не такой. Я старался добавлять запоминающиеся мотивы, мелодии, вокал и оригинальные идеи, и я чувствовал, что это пустая трата времени, потому что рядовому трансеру это было не нужно

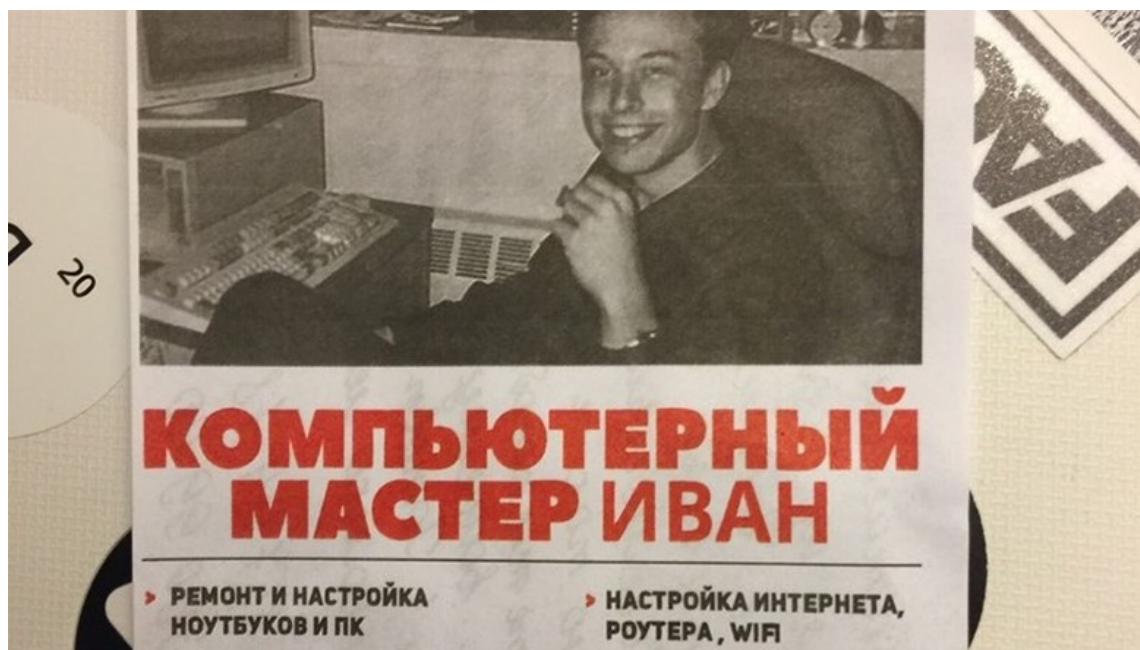
По последней причине, кстати, множество моих коллег-музыкантов со временем стали переходить в более перспективные и "благодарные" музыкальные стили, но мне это было не по душе. Во мне жила надежда, что я смогу сделать самый оригинальный и мелодичный психоделический транс на свете. И эта надежда таяла с каждым днём.





После череды различных не совсем верных решений я впал в стагнацию. Выступлений было всё меньше, и мне всё меньше хотелось принимать в них участие. Я придумывал различные отмазки для организаторов. Я знаю, что был не прав по отношению к этим людям, но я ничего не мог с собой поделать. Институт я тоже бросил, потому что я не видел особого смысла в том, как мои однокурсники собирали "пожертвования преподам", чтобы получать зачёты. Я хотел быть честен с собой.

В итоге к 2011 я остался и без учёбы, и практически без музыки. Зато появилась работа. Наверняка, вы видели объявления на подъездах: "Компьютерный мастер на дому"? Я стал им. Эникей по вызову - я считал, что это максимум того, что я достоин, после того, как обманул надежды родителей.



Дёшево и сердито

Проработав несколько лет, я захотел большей стабильности и устроился в [Московский кредитный банк](#). Ночной дежурный специалист технической поддержки. Наш отдел занимался

поддержкой систем банка в нерабочее время, общаться приходилось только с сотрудниками банка, не с клиентами. Смены, в основном, длились от 16 часов до суток. С коллективом и начальником очень повезло (привет, Сергей Владимирович). В первый год я узнал там довольно много нового, начиная с PL/SQL, продуктов VMware и заканчивая бездной различного банковского ПО.

И, хотя такой формат работы был мне по душе, моё яркое прошлое не давало мне покоя. В сердце теплилась надежда о следующем большом музыкальном проекте. Я не мог смириться с тем, что это всё, на что я способен.



### Пристанище бывшего трансового артиста

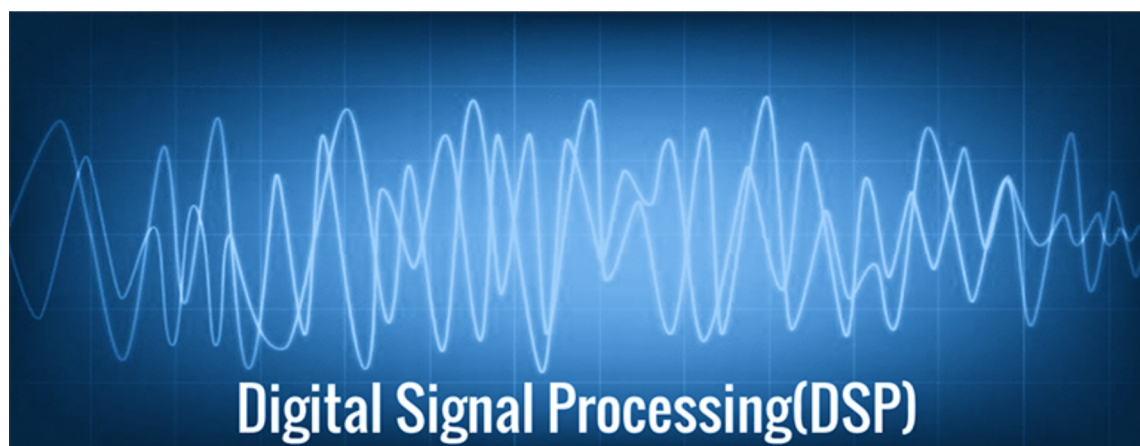
Да, у меня была возможность отдыха во время отсутствия рабочих задач. Этот грязненький, измазанный в тонере (и не только), матрас стал моим пристанищем на следующие пять лет. Этаким символом того, к чему я пришёл. "Погонишься за двумя зайцами" в действии. Но я благодарен этим бессонным ночам, которые провёл на нём. Они позволили мне переосмыслить свою жизнь.



## Цифровая обработка сигналов

В одну из бессонных ночных смен мне стало интересно, как создаются музыкальные программы, а именно [VST плагины](#). Как сделать свой VST эффект, VST инструмент? Моя логика была проста: если бы я смог сделать свой личный уникальный VST плагин, я бы смог воспользоваться им в своей музыке, и она стала бы еще более уникальной и оригинальной. *Некий "secret tool"*, который я бы сделал сам для себя. Я стал верить, что если сделаю что-то подобное, то смогу грандиозно вернуться в музыку и покорить все вершины.

После непродолжительных поисков я нашел мнение одного из специалистов: проектируешь [DSP \(ЦОС\) алгоритмы](#) в MATLAB и далее разрабатываешь чистовой вариант на C++ и VST SDK. И я загорелся. Мне захотелось придумать такой DSP алгоритм, который бы сделал мою музыку самой уникальной. Может, это был бы какой-то неведомый эффект, либо синтезатор с уникальным видом осциллятора, которого еще не существовало. Я пока не знал ответа, но был уверен, что придумаю что-то, если досконально изучу цифровую обработку сигналов.



Арт mikeroyal

Изначально я считал, что ЦОС невозможно изучить самостоятельно. Я даже вёл переписку с [Алексеем Лукиным из iZotope](#) в надежде узнать, как постичь данную науку, куда

поступить и так далее. На 2015 год картина складывалась примерно следующая:

- Почти все российские факультеты или кафедры хоть как-то связанные с ЦОС были "про электронику: ПЛИС и тому подобное", не "про алгоритмы для обработки аудио"
- Могло повезти с наставником на кафедре, но маловероятно
- Возможно, за рубежом ситуация лучше
- Вероятно, продуктивнее всего было бы просто изучать существующие открытые исходники VST плагинов

Так как я хотел фундаментально постичь DSP, я не рассматривал последний вариант на данном этапе. Также я решил не рисковать поступать на факультет электротехники в любой из местных университетов, потому что предполагал, что это не то, что мне нужно. Методом исключения остается...



Да, лёжа на том несчастном, всеми забытом матрасе, я решил попробовать поехать учиться [цифровой обработке сигналов](#) в Германии. Я консультировался со специалистами по переезду, искал варианты. Выяснилось, что университетов, обучающих этой специальности, довольно много. И хотя нигде не было указано, что я смогу создавать аудио софт, используя эти знания, у представленных программ обучения была бОльшая связь с Computer Science нежели с электротехникой.

► Список рассматриваемых университетов

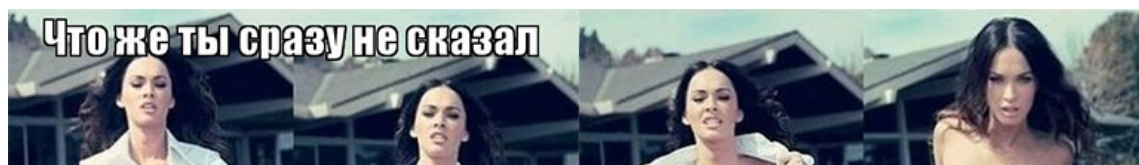
План был такой: курсы немецкого, сертификат B1, штудиенколлег (студенческий колледж), далее сам университет: 6-7 лет - и я стану спецом по ЦОС к 2022 году! Меня поддержали родители, друзья и коллеги. Но...



Можно искать множество отговорок, но если быть честным, то я побоялся таких кардинальных изменений. У меня не было знакомых, кто провернул бы похожую авантюру, и я спасовал. Как можно было обменивать тёплые часы с будущей женой за Dota 2 в Мытищах на учёбу в неизвестной далекой стране, где у тебя никого нет? Я сделал свой выбор.

Но, несмотря на это, идея обучиться ЦОС никуда не пропала, и я нашел другой подход: самообучение. Я стал скупать все доступные книги по [цифровой обработке сигналов](#). Также параллельно с этим я изучал MATLAB. Если нужно было бы сократить всю статью до трёх слов, я бы выбрал:

- [Свёртка](#)
- [Преобразование Фурье](#)







И хотя содержание прочитанных мною книг значительно отличалось, все они были сосредоточены на этих двух понятиях. Я понимал, что эти знания можно было бы использовать в проектировании [фильтров](#), [эквайзеров](#), [свёрточных ревербераторов](#). Но как, используя эти знания, можно было бы сделать свой секретный [VST плагин](#), который бы позволил мне создавать свою поистине уникальную музыку? Ответа пока не было (спойлер: его так и нет).

► Советую эти книги по DSP

Получив *некоторые* знания, я решил, что мне нужна практика. Я стал искать возможности проявить себя в сфере [ЦОС](#). Я пытался устроиться в [GPU.AUDIO](#) / [BRAINGINES](#) на позицию ЦОС алгоритмиста. Но их задачи по распараллеливанию DSP алгоритмов под видеокарты были крайне нетривиальны для меня на тот момент. Хотя я не уверен, что даже сейчас бы справился с чем-то подобным.

В тот период времени я частенько "зависал" с Александром Александровым, основавшим [Eternal Engine EMI](#). Будучи моим коллегой по сцене, он также является истинным электронщиком с образованием в отличие от меня. Обмен идеями и опытом с этим гениальным человеком значительно повлияли на моё будущее развитие. Через несколько лет он успешно запустит в производство свой [Apparatus - первый российский дуофонический синтезатор на основе вакуумных и газонаполненных радиоламп](#).

Также мне удалось поработать тестером плагинов [Polyverse Music](#). Общение с [Эрезом Айзенем из Infected Mushroom](#) стало

для меня неоценимым опытом. 5 лет назад признание моей работы такими людьми, как он, было бы верхом несбыточных мечтаний. Сидя в хрущёвке в Мытищах, я периодически отправлял команде Эреза видео с различными найденными багами в их разрабатываемой VST-шке, и на основе моих наблюдений эти легендарные люди вносили изменения в свой будущий продукт... Это чувство было сложно описать словами.

## Сэм

Июнь 2016 года. Было жаркое лето, все говорили о революционном [приложении Prisma](#). Для тех, кто не в теме: это было одно из первых приложений "нейронного переноса стиля". Как бы выглядела ваша фотография, нарисованная в стиле Ван Гога? Это был хит.



### Принцип работы Prisma

В то время я часто переписывался в ВК с Сэмом. Да, это самый настоящий русскоговорящий Сэм. Он был моим коллегой по сцене, и в 2016 году он активно занимался написанием [Tech House](#), [Techno](#) и подобной музыки. У Сэма все было на мази, кроме мастеринга.

В то время я частенько подрабатывал [мастеринг-инженером](#) "только для друзей", и Сэм иногда пользовался моими услугами. Чтобы понять, как должна звучать отмастеренная версия его трека, я частенько просил его присылать мне "референс", то есть

готовый трек какого-нибудь популярного артиста, этаким трек эталонного качества по мнению Сэма. И я старался мастерить треки Сэма так, чтобы они звучали как "референс". И хотя мои действия немного отличались от трека к треку, в них отслеживались повторяемые действия:

1. Найти самые громкие места в обоих треках
2. Подровнять трек Сэма по мощности ([RMS](#)) к "референсу"
3. Подогнать [АЧХ](#) трека Сэма к "референсу" (мне нравилось пользоваться [iZotope Ozone Match EQ](#) для этого)
4. Поколдовать со стереобазой
5. Незаметно убрать все "пики" brickwall лимитером (я использовал [Voxengo Elephant](#) или [FabFilter Pro-L](#))

Имея небольшой школьный и институтский опыт программирования, я стал задаваться вопросом: а возможно ли автоматизировать этот процесс? Аналогия с Prisma глубоко засела в моей голове.

Изначально я попытался пойти по простому пути: я искал консольные инструменты для пакетной обработки аудио файлов. Я нашел два любопытных проекта:

- [SoX - консольная утилита для пакетной обработки аудио](#)
- [MrsWatson - консольный хост для VST плагинов](#)

Используя данные утилиты, я пробовал написать скрипт, который бы выполнял мастеринговую рутину за меня. Но в итоге у меня не получилось автоматизировать указанную выше последовательность действий, используя эти инструменты. И я не был уверен, что это вообще возможно. Так я пришёл к выводу, что здесь не обойтись только пакетными скриптами.

Я знал MATLAB и цифровую обработку сигналов. Почему было



бы не попробовать написать подобную утилиту с нуля? Но перед созданием прототипа, я решил изучить рынок.

Анализ рынка автоматизированного аудио мастеринга на 2016 год оказался следующим: скачиваемые утилиты отсутствовали полностью, но было несколько онлайн-сервисов, предлагающих подобные услуги на платной основе:

- [LANDR](#)
- [eMastered](#)
- [MajorDecibel](#)
- [MasteringBOX](#)
- [CloudBounce](#)

И хотя некоторые из них выглядели престижно, все эти сервисы объединяло следующее: они позволяли загружать только свою композицию и самостоятельно решали, как должен звучать финальный результат. Для меня это был полный nonsense, протестировав мастеринг своих композиций на этих сервисах, я получил неудовлетворительные результаты. Я искренне хотел, чтобы мой мастер звучал как [Vibe Tribe](#), а не как решал *какой-то там* [LANDR](#).

**Итог:** на тот момент не было ничего, что позволяло бы осуществить подобный автоматизированный референсный мастеринг.

## Прототип

Я довольно быстро справился с задачей загрузки аудио файлов в MATLAB. Добавил [ресемплинг](#), проверку длин файлов, количество каналов и так далее.

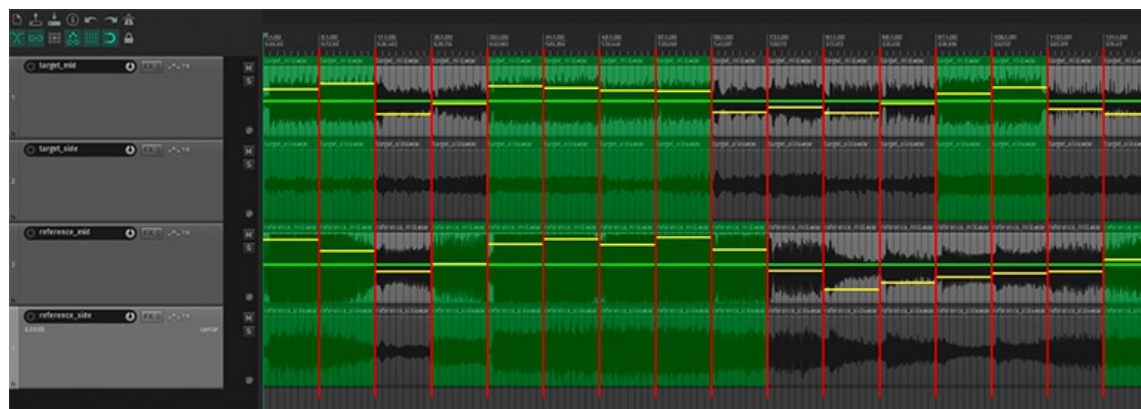
Первая сложность возникла с выравниванием аудио по

мощности:

- Как найти самые громкие места в обоих треках?
- По какому принципу разделять файл на части?

По своему опыту я знал, что в современной [электронной танцевальной музыке](#) изменения чаще всего происходят каждые 8 тактов: вводится новый инструмент или мелодия, меняется гармония, вступает новая перкуссия и так далее. Это также известно как "[музыкальная фраза](#)". Был вопрос о продолжительности этих 8 тактов в секундах (напрашивались мысли о [BPM анализаторе](#)). Также я знал, что для сравнения громкости часто используют [метрику RMS \(root mean square\)](#). Можно было бы использовать только отрезки с наибольшим RMS. После продолжительных экспериментов я пришел к...

? **Первая эвристика:** разделить аудио файл на равные отрезки по **15 секунд**, что равно 8 тактам на скорости 128 BPM. Даже если у композиции другая скорость, **возникающая погрешность незначительна**. Далее следует выбрать отрезки только со значением RMS выше среднего. **Анализ только этих частей имеет значение при сравнении.**



Найденные отрезки

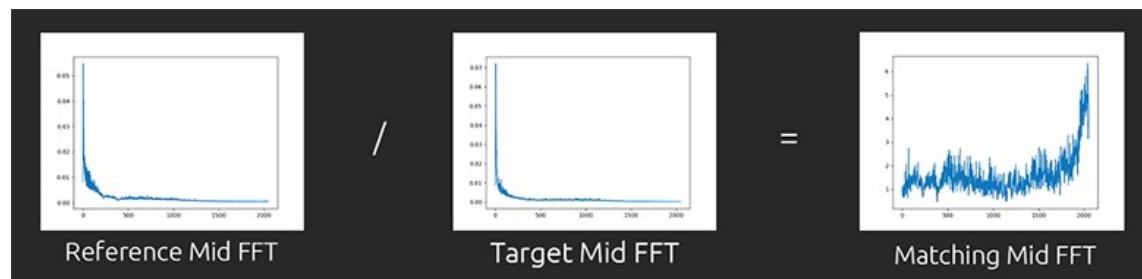
Оставалось только написать код, который бы подгонял громкость исходного трека так, чтобы значения RMS найденных отрезков совпадали для обоих файлов - простое умножение вектора на

найденный коэффициент. Итак, первая (и самая легкая) часть была готова.

Далее, подгонка [АЧХ](#): было необходимо сделать свою автоматизированную реплику [Ozone Match EQ](#) на MATLAB. Ранее, изучая цифровую обработку сигналов, я узнал, что [нерекурсивные фильтры](#) являются более точными, чем [рекурсивные](#). Они проще в проектировании, но требуют больше вычислительных ресурсов. Но, так как я разрабатывал не VST плагин, который бы работал в реальном времени, использование нерекурсивного фильтра было оправдано.

После этого последовало погружение в [преобразование Фурье](#). Отрезки, найденные в предыдущий стадии, следовало разделить на более малые: **по 4096 семплов каждый**. Это требование [алгоритма БПФ](#) - длина вектора должна быть степенью двойки. Значение в 4096 семплов довольно часто используется в подобных VST FFT анализаторах, и оно даёт хорошее частотное разрешение относительно времени. Вычисленные АЧХ следовало усреднить между собой, таким образом получались две усредненные АЧХ:

- АЧХ громких отрезков нашей композиции
- АЧХ громких отрезков эталонной композиции



Вычисление АЧХ нашего Match EQ

Так как я не хотел, чтобы фильтрация вносила [фазовые искажения](#), выбор КИХ-фильтра вновь оправдал себя.

Игнорируем фазы, используем только магнитуды - идеальный

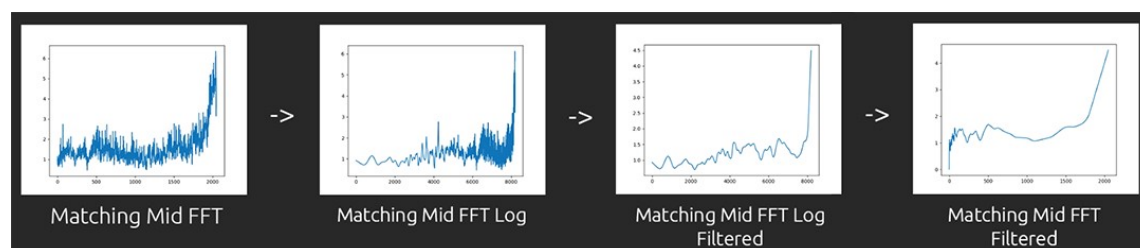


линейный фильтр для мастеринга. Оставалось только поделить одну АЧХ на другую, найти конечную [импульсную характеристику](#) нашего Match EQ фильтра обратным БПФ, перемножить её на какую-нибудь [оконную функцию](#) и произвести [свёртку](#) с исходным треком. Это классическая [цифровая обработка сигналов](#), ничего нового.

Но результат оказался ужасен. Артефакты и искажения, вносимые этим фильтром, сильно портили звучание. Анализируя данную проблему, я обнаружил недостающую часть:

**сглаживание АЧХ.** Проведя не одну неделю за поиском оптимальных методик и алгоритмов сглаживания, я пришел к...

? **Вторая эвристика:** перед нахождением импульсной характеристики требуется сгладить АЧХ фильтра. Так как фильтр применяется к звуковому сигналу, **сглаживание должно проводиться в [логарифмическом масштабе](#)**. Наиболее подходящий алгоритм сглаживания в данном случае - [Locally Weighted Scatterplot Smoothing \(lowess\)](#).



## Сглаживание в логарифмическом масштабе

Некоторое время пришлось потратить на подбор коэффициентов сглаживания. Забегая вперед, скажу, что в будущей Python-версии использовались другие коэффициенты, хотя кривая оставалась примерно той же.

Тестируя результаты, выдаваемые этим фильтром, я был шокирован. По моему мнению, качество эквализации не уступало [Ozone Match EQ](#). Фильтр был настолько хорош, что мог с легкостью исправлять сильно зажатые частоты. Это стало

сердцем проекта.

Далее нужно было придумать метод автоматизированной работы со стереобазой. Я иногда использовал [Mid / Side обработку](#), поэтому решил поэкспериментировать с ней. Чтобы понять, что это такое, достаточно взглянуть на эти формулы:

$$\begin{aligned} \textit{Mid} &= \frac{\textit{Left} + \textit{Right}}{2} \\ \textit{Side} &= \frac{\textit{Left} - \textit{Right}}{2} \end{aligned}$$

В определенный момент можно переходить от работы с левым и правым каналами к работе с "*центральной*" и "*боковым*". Таким образом, можно разделять обработку, например: добавлять больше компрессии "*в центр*", больше пространства "*на средних в стороне*" и так далее. Поищите в Google, это [полезная техника](#).

И как можно было применить эту технику в данном случае?

Ответ оказался до безобразия прост.

? **Третья эвристика:** предыдущий этап подгонки АЧХ

**необходимо проводить отдельно для Mid и Side каналов.**

Таким образом, используется **два отдельных фильтра** с двумя разными импульсными характеристиками. Конечно же, обрабатываемый аудио файл должен содержать какую-либо стерео информацию - его **Side канал должен отличаться от тишины**.

У меня было чувство, что это какая-то шутка, обман. Но правда оказалось таковой - подгоняя АЧХ громких частей композиций отдельно для Mid и Side каналов, мы получаем подгонку стереобазы, как говорится, "for free".

Отлично, что дальше?

? **Четвёртая эвристика:** после подгонки АЧХ значение RMS

обрабатываемого файла *немного* меняется. Громкость необходимо подогнать повторно, **но с учётом применения**

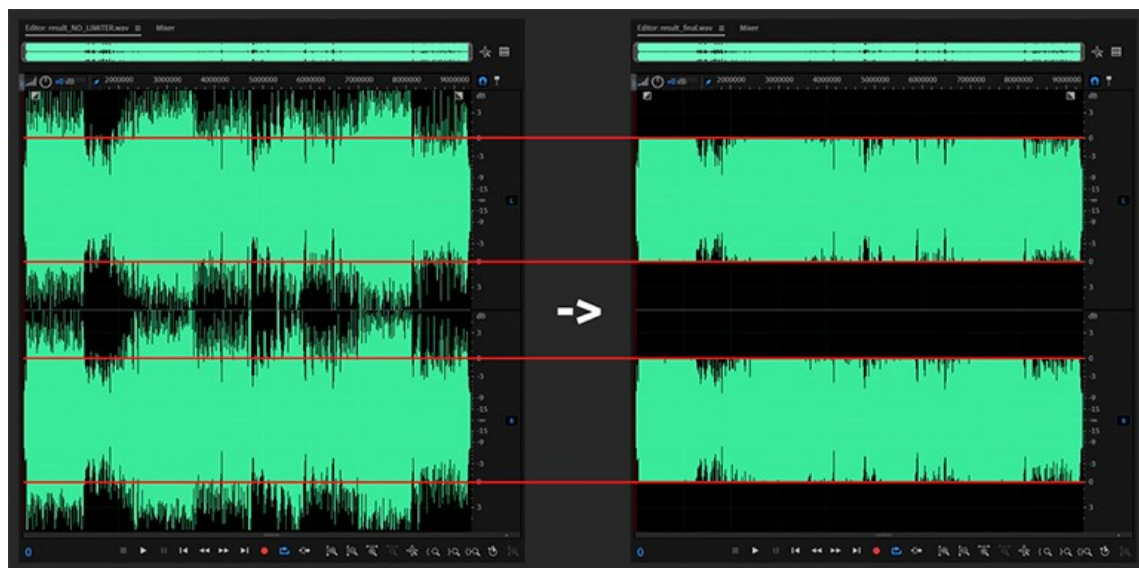
## финального лимитера.

Получается как-то так:

1. Пройтись лимитером по временной копии файла
2. Вычислить разницу в значениях RMS с "референсом"
3. Изменить громкость в оригинальном файле
4. Повторить несколько раз (3-4 раза достаточно)

В целях ускорения времени обработки, в качестве лимитера на данном этапе я решил использовать обычный хард-клиппер. Повторюсь, он применяется только к временной копии, в конечный результат никакие искажения не вносятся.

Оставалось дело за малым - добавить обработку настоящим финальным лимитером типа brickwall. Цель данного шага: избавиться от всех будущих возможных "пиков", то есть побороть превышение сигналом "красной зоны" (отметки 0дБ) до заключительного экспорта в WAV (16 или 24 бит). Данный процесс не должен вносить слышимые искажения: обработка пиков должна проводиться плавно во временной области.



Визуально лимитинг выглядит как "стрижка" звуковой волны.

Здесь стоит отметить одну особенность, про которую не все знают: процессинг аудио в формате с плавающей точкой (32-bit

float) позволяет безопасно превышать значение 0дБ. Вполне вероятно, что к данному этапу в нашем аудио наберётся много семплов, превышающих значение 0дБ, если был выбран довольно громкий "референс". Но это не страшно, так как вся обработка проводится в формате с плавающей точкой. Эта особенность подробно разбирается [в данной статье](#).

На данном этапе мастеринг-инженеры обычно используют такие плагины, как [FabFilter Pro-L](#), [Waves L2](#), [Izotope Ozone](#), [Voxengo Elephant](#) и т.д. В MATLAB к тому моменту уже [появился свой лимитер](#), но я не стал рисковать. Я посчитал, что на то время их разработка еще была сыровата для качественного аудио мастеринга.

Так как изначально я не планировал выкладывать код в открытый доступ, на этом этапе я решил воспользоваться [MrsWatson](#), о котором я рассказывал ранее. Итак, после всех описанных шагов аудио передавалось в консольный [VST хост](#), где обрабатывалось финальным проприетарным brickwall лимитером, а далее возвращалось обратно в MATLAB.

Добавив к этому перенос пиковой амплитуды и создание превьюшек, я получил первую версию утилиты, которую я в будущем назову **Matchering**.

Опыт программирования на тот момент у меня был крайне ограничен школьными и институтскими поделками на Turbo Pascal. Я не имел никаких представлений о паттернах проектирования и принципах разработки: DRY, KISS, SOLID и т.д. Поэтому качество [кода](#) оставляло желать лучшего. Но, несмотря на то, что мне было бы стыдно показать [этот код](#) кому-либо, он успешно выполнял свою задачу.

## Веб-сервис



После анализа аналогичных продуктов я твёрдо принял решение, что необходимо делать именно онлайн-сервис. Конкуренты были (я указал их выше), но никто из них не предлагал референсный мастеринг. Таким образом, у меня была возможность сделать первый в своём роде онлайн-сервис автоматизированного мастеринга, позволяющий пользователю указать, как должен звучать результат, предоставив возможность загружать "референс".

Но была одна проблема: у меня отсутствовал опыт веб-разработки. Я немного знал HTML и CSS, но я понятия не имел, как сделать онлайн-сервис, который бы доставлял аудио-файлы пользователя до виртуалки с MATLAB и обратно.

На помощь пришёл мой коллега из Московского кредитного банка - Игорь Исаев. Будучи диджеем и битмейкером, он регулярно сталкивался с рутинной [аудио мастеринга](#), поэтому идея подобного онлайн-сервиса ему пришлась по душе. Также у Игоря имелся большой опыт в разработке сайтов на PHP, JavaScript и jQuery. Для меня стало большой удачей, что я работал в одном отделе вместе с таким опытным специалистом.

Оставалось дело за малым: название и домен. Недолго думая, совместив слова **Match** и **Mastering**, мы пришли к названию нашего будущего сервиса - **Matchering**. Так как я не планировал останавливаться на одном-единственном сервисе, я искал такое доменное имя, которое бы могло обобщить нашу сферу деятельности. И чудом нашел свободный домен **sound.tools!** Пять тысяч рублей в год - довольно скромная сумма за такое лаконичное название, которое впоследствии стало названием нашей "компании".





## Дизайн первой версии

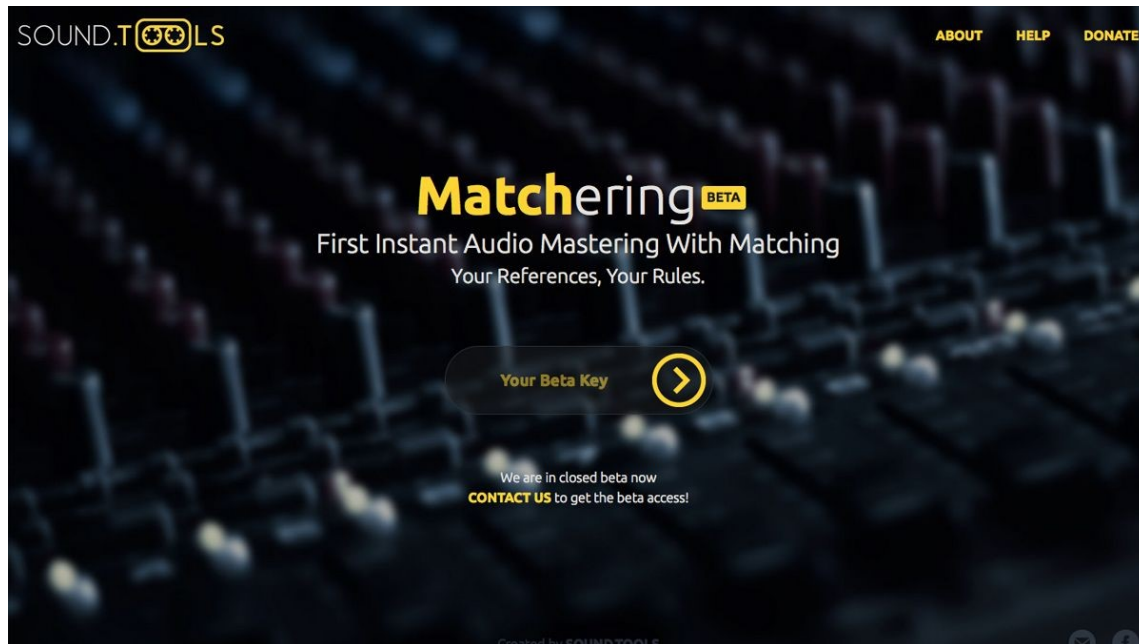
В течение нескольких бессонных месяцев мы разрабатывали наш онлайн-сервис. За это время я актуализировал свои знания о вёрстке веб-сайтов, так как очень хотел, чтобы сервис смотрелся не как "привет из 2000-х". Я регулярно консультировался по этой части с еще одним коллегой и старым другом - опытным дизайнером Артёмом Пономаренко (который, к слову, раньше рисовал мне потрясающие обложки для трансовых релизов). Его свежий взгляд позволил нам избежать множества проблем. Такими темпами мы стали готовы к запуску "беты" к началу 2017.

## От рассвета до заката

Новогодние праздники, новый 2017 год. ["Бета" нашего онлайн-сервиса была успешно запущена](#). Регистрации и личного кабинета пока не было, было решено предоставлять доступ к закрытой "бете" по ключам. Причиной этому стали наши опасения насчёт недостатка вычислительных ресурсов - боялись резкого наплыва публики. Также это стало хорошим маркетинговым ходом: мы сгенерировали несколько уникальных ключей для таких тематических сайтов, как [MusicRadar](#), [Ask.Audio](#), [Rekkerd](#), что повысило шансы публикации статей о нашем сервисе.

Я ежедневно получал дюжины писем с восторженными

отзывами и просьбами предоставить доступ. [Ближе к весне 2017 мы открыли доступ для всех.](#) Веб-сервер и 4 параллельных обработчика файлов успешно справлялись с нагрузкой.



Скриншот с закрытой "беты"

Каждый MATLAB-обработчик размещался на отдельной виртуальной машине с такими параметрами:

- 4 vCPU
- 8 GiB RAM
- 10 GiB SSD
- Windows Server 2012 R2

Виртуальная машина, использовавшаяся в качестве веб-сервера и хранилища данных, имела следующие параметры:

- 2 vCPU
- 4 GiB RAM
- 3 TiB HDD
- Ubuntu Server 16.04 LTS

Обработчики по очереди забирали файлы для каждой

мастеринг-сессии и подкладывали результаты обратно.

Информация о сессиях находилась в MySQL, обмен файлами между виртуалками осуществлялся через Samba. Так как у нас не было опыта проектирования высоконагруженных систем, у нас отсутствовали классические балансировщики нагрузки.

Шло время, и мы понимали, что необходимо развивать сервис, у нас с Игорем были планы введения платной подписки, личного кабинета и других новых фич. Но опыт и интуиция мне подсказывали, что сейчас не время. Я видел несколько проблем, без решения которых моя совесть не позволяла мне начать монетизировать наш сервис:

- Отсутствие каких либо механизмов защиты от DDoS: наш единственный Apache мог бы быть перегружен простой сотней параллельных запросов, также было бы довольно элементарно написать скрипт, который бы намертво забил очередь обработки аудио
- Невозможность использования CDN: на тот момент CloudFlare не разрешал HTTP POST загрузку файлов размером более 100MB даже на Enterprise тарифе. Хотя через несколько лет я узнал, что это ограничение можно было обойти, реализовав chunk uploader
- Нехватка места для хранения файлов мастеринг-сессий: HDD на 3 TiB регулярно заполнялся, и его приходилось чистить от старых сессий раз в месяц. Я понимал, что при должном маркетинге, его бы хватало всего на несколько дней
- Противоречивое принятие сервиса сообществом: я периодически пытался создавать посты о веб-сервисе на таких тематических ресурсах, как [KVR](#), [AudioSEX](#) и т.п. Отношение публики к подобного рода сервисам было, мягко говоря, сомнительное



- Также я не понимал, каким образом мы могли бы выполнять качественную 24/7 поддержку клиентов
- Была неясна стратегия маркетинга, на продвижение сервиса нужны были деньги, и я не хотел рисковать, учитывая предыдущие пункты

Я считал, что перед введением платных планов, мы были обязаны решить эти проблемы. Но недостаток веры и денежных ресурсов, а также отсутствие должного продакт-менеджмента сделали своё дело.

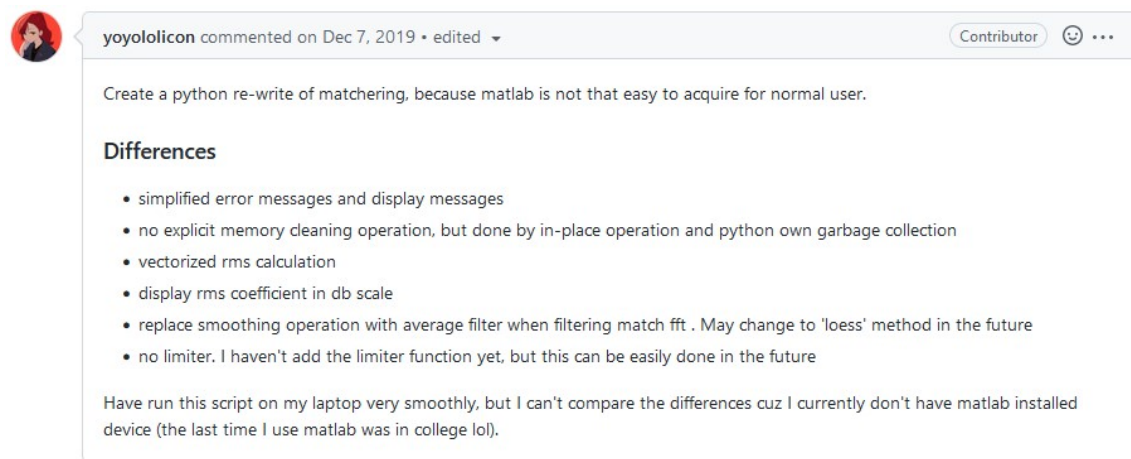
Осенью 2018 года проект был закрыт, а [исходный код MATLAB-обработчика был размещён на GitHub](#). Так как на тот момент я не первый год изучал Python, я знал, что Matchering было бы возможно переписать на Python, используя Numpy. [О чём я и указал в README.md репозитория](#), даже не надеясь на то, что подобный человек мог бы найтись. Я окончательно перевернул эту страницу моей жизни и полностью переключился на фриланс, не связанный со звуком.

## Спаситель из Тайваня

Прошло больше года, я успешно осваивался на Upwork, выполняя заказы по созданию ботов для социальных сетей на Python. Discord стал моей страстью, а разработка ботов под него давала мне отличную практику асинхронного программирования. И в этот момент случилось нечто совершенно неожиданное.

Во всеми забытый репозиторий Matchering поступил [судьбоносный пул-реквест](#) от [тайваньского разработчика Chin-Yun Yu](#). В данном PR содержался прототип *того самого* Matchering на Python. Это было очень неожиданно. Опыт фриланса на Python и недавнее прохождение [курса Эндрю Энга](#)

[по глубокому обучению](#) позволили мне свободно ориентироваться в представленном коде.



## Судьбоносный пул-реквест

Проведя тесты и сравнив результаты с оригинальным Matchering, я убедился, что это то, что нужно. Моя интуиция стала трубить о том, что мне необходимо всё отложить и плотно заняться этим проектом. Нужно было:

1. Исправить некоторые неточности, чтобы процессинг стал идентичен первой версии
2. Реализовать свой лимитер типа brickwall на Python
3. Переписать код так, чтобы он соответствовал всем современным принципам разработки
4. Сделать код модульным, чтобы Matchering стал подключаемой Python библиотекой с возможностью установки через pip
5. Придумать и реализовать что-то более-менее юзабельное для конечного пользователя, не являющегося Python-разработчиком

Самым сложным пунктом оказался второй: в экосистеме Python отсутствовала какая-либо реализация [brickwall лимитера](#). Ни у меня, ни у моего коллеги из Тайваня не было опыта разработки лимитеров. Я даже безуспешно пытался обратиться к [Алексею Ванееву из Voxengo](#) за консультацией по разработке подобных

инструментов. В итоге мы с [Chin-Yun Yu](#) пришли к собственному решению проблемы.

[VST плагины](#) в процессе обработки имеют доступ только к аудио данным, которые находятся в буфере. Но в нашем случае обработчику доступны данные всей композиции, поэтому мы пришли к такому алгоритму brickwall лимитера на [Numpy](#) и [SciPy](#):

1. Построить вектор со значениями семплов выше 0дБ, назовём это **огibaющей A**
2. Скопировать **огibaющую A**, применив к ней небольшое сглаживание фильтром Баттерворта, это будет являться симуляцией атаки лимитера или **огibaющей B**
3. Повторить предыдущий пункт, но с более сильным сглаживанием, что станет симуляцией релиза лимитера или **огibaющей C**
4. Совместить **огibaющие A, B и C** в **финальную огibaющую**, поэлементно выбрав наибольшие значения из трёх
5. Отразить **финальную огibaющую** относительно единицы и нуля
6. Поэлементно умножить аудио данные на **финальную огibaющую**

Мы даже не знали, делал ли кто-то что-нибудь подобное до нас. Но результаты обработки нашим лимитером оказались сравнимыми с результатами обработки лучшими VST лимитерами в индустрии. [Результаты тестов доступны на GitHub](#). Лимитер успешно выполнял свою задачу, внося минимум искажений.

После решения этой задачи релиз на PyPI оставался лишь делом времени. [13 января 2020 года я выпустил библиотеку Matching 2.0 для Python](#). И в то же время опубликовал

[консольную утилиту `matching-cli`](#), которая позволяла эффективно осуществлять пакетную обработку аудио файлов. Оставался последний пункт плана.

## Не забыть о музыкантах

Я искренне желал, чтобы библиотекой могли пользоваться люди, не связанные с программированием: музыканты, продюсеры и звукорежиссёры. Было несколько вариантов решения этой задачи, такие как [py2exe](#) или [pyinstaller](#), но я решил пойти по нестандартному пути и сделать контейнеризированное веб-приложение.

- У меня на руках были исходники рабочего веб-сервиса первого `Matching`, сделанного вместе с Игорем Исаевым
- Я давно хотел попрактиковаться с `Docker`, и строчка о нём в моём резюме была бы не лишней
- Также я хотел переписать PHP код Игоря на Python, используя `Django`, заработав еще одну заветную строчку в резюме

В общем, слепо следуя [CV Driven Development](#), в течение месяца я актуализировал ранее написанный Игорем фронтенд код, полностью переписав бекенд на `Django`. Воспользовавшись еще несколькими антипаттернами, я упаковал результат в `Docker`-контейнер и был готов к релизу.

Обжёгшись о критику целевой аудитории несколько лет назад, я решил в этот раз подготовить несколько материалов, доказывающих честную и качественную работу `Matching`. Заручившись помощью супруги и школьными знаниями о видеомонтаже, мы подготовили несколько видео для YouTube. Основным из которых стало это:

Как и ранее, [я отправил пресс-релизы в несколько тематических](#)



[изданий](#). Приём аудиторией оказался более тёплым, чем в первый раз. [Положительные отзывы](#), [донаты](#), [обзоры блогеров и прессы](#) - всё это показывало, что интуиция меня не подвела.

Без проблем, конечно, не обошлось. И хотя я выбрал Docker в надежде упростить процесс установки для рядового пользователя, в некоторых случаях получилось чуть ли не наоборот:

- Docker Desktop запускался только на Windows 10 и новее
- В феврале 2020 Docker Desktop ещё не поддерживал домашнюю версию Windows
- Несмотря на подробные инструкции по установке (в [текстовом](#) и [видео](#) вариантах), процесс запуска контейнера Matchering 2.0 оставался нетривиален
- Пользователям с отключенной аппаратной виртуализацией даже приходилось править настройки в BIOS

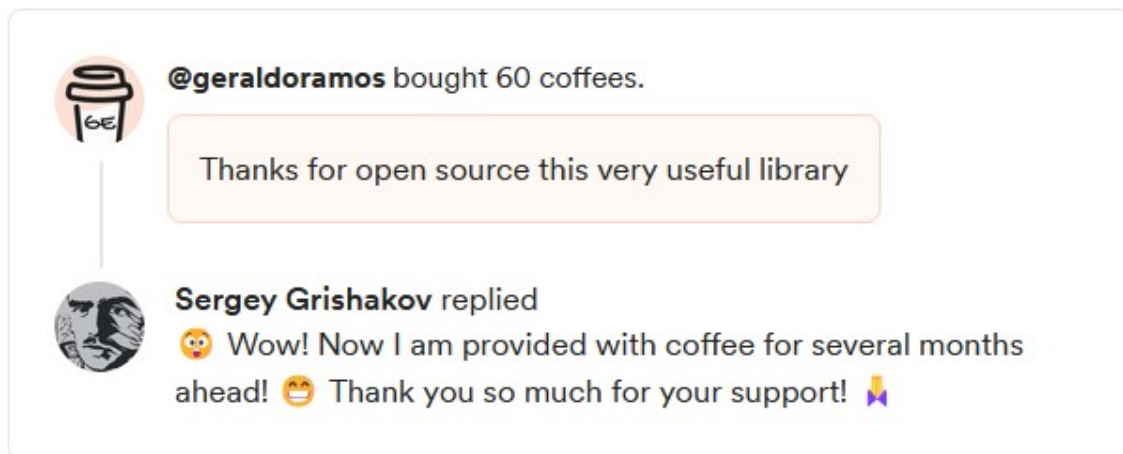
И несмотря на то, что это создавало проблемы для пользователей, я считаю, что это сыграло определенную роль в продвижении Matchering. Выбор данного решения создал своеобразный искусственный дефицит. Рядовой пользователь, привыкший к установке плагинов в пару кликов, получил целый "квест". Мотивацию не сдаваться им подогревали [статьи](#), [обзоры](#) и [положительные отзывы](#). Люди общались на форумах и искали специалистов, чтобы заполучить заветный установленный Matchering 2.0. Родился инфоповод. Честно признаюсь, что я не планировал и не ожидал подобного эффекта.





Дизайн второй версии

Так как [Matching](#), в первую очередь, можно было рассматривать как "фабрику онлайн-сервисов мастеринга", со временем стали появляться первые подобные ресурсы. Сперва я бы хотел выделить [Moises](#). Данный сервис предоставляет услуги разделения композиции на отдельные дорожки (например: ударные, басс, вокал, лид и т.д.). Также они предлагают своим клиентам услуги мгновенного референсного мастеринга. Я узнал, что под капотом их сервиса используется Matching, когда получил пожертвование от [Джеральдо Рамоса \(CEO Moises\)](#).



Это пожертвование стало наибольшим - целых 60 чашек кофе

После этого [в августе 2022 сервис Moises был проинвестирован на \\$8.65M фондами Kickstart Fund и Monashees при участии Norwest, Toba Capital, Alumni Ventures, Goodwater и Valutia.](#)

## Предупреждение RIAA

Довольно интересный казус произошел с другим сервисом,

использующим библиотеку Matchering - французским [Songmastr](#).  
[В октябре 2022 года американская ассоциация звукозаписывающих компаний RIAA передала в Торговое представительство США отчёт о сервисах, способствующих распространению пиратства или выступающих в качестве угрозы для музыкальной индустрии. Одним из таких сервисов стал Songmastr.](#) Простыми словами, претензия была такова: "если вы позволяете своим клиентам переносить АЧХ песен Бейонсе на их музыку, то вы способствуете музыкальному пиратству". Да, это та же организация, которая [ранее пыталась забанить youtube-dl](#).



Арт SAMESOUND

[Инцидент владельцу сайта в итоге удалось замять](#), для чего пришлось убрать с сайта заранее подготовленные референсные композиции известных артистов, а также изменив описание сервиса. Но осадок остался. Вполне возможно, что в будущем подобные организации, как [RIAA](#), продолжат продвигать идею, что АЧХ популярных песен должны быть защищены авторским правом.

## Что дальше

После выхода [Matching 2.0](#) прошло 2 года. Наш инструмент появляется на первой строчке поиска по запросу open source mastering. Google перестал считать "Matching" словом с опечаткой. Docker-образ был загружен более 10 тысяч раз. Референсный мастеринг "волшебным образом" появился и у LANDR, и у eMastered. Хайп давно спал, но, судя по аналитике, инструмент нашел своего потребителя.

Был ли проект успешен? Если всё мерить деньгами, то это однозначный "fail story". Учитывая все затраты, в плюс выйти так и не удалось. Но если в качестве меры выбрать "полезность для людей", то я считаю, что это можно назвать успехом. Инструмент полностью решает проблему аудио мастеринга для людей, у которых нет времени, возможности или желания использовать профессиональные VST плагины для этой цели.

Можно ли его как-то улучшить? Безусловно. Нужно ли? Хороший вопрос. Я не совсем понимаю эту моду на регулярные обновления программного обеспечения. Если молоток хорошо выполняет свою работу, нужно ли регулярно вносить изменения в его конструкцию? Я считаю, что нет, если у вас нет цели периодически подогревать инфоповод, и вам не нужно платить зарплаты сотрудникам.

Я уже более 5-ти лет не занимался написанием музыки, а готовых работ не выпускал уже более 10-ти. Я пытался несколько раз вернуться в музыку, но безрезультатно. Знание принципов разработки программного обеспечения довольно сильно изменило мой взгляд на собственный труд. Я перестал испытывать удовольствие от процесса написания музыки, видя все эти повторяющиеся рутинные действия. Да и стиль музыки, в котором я раньше работал, уже давно канул в лету.

Мечты о том самом "secret tool", ради которого я изучал ЦОС, уже



несколько лет пылятся на дальней полке. Зачем мне он, если я уже не буду заниматься музыкой? Да и зачем нужно что-то ещё, когда есть [Xfer Records Serum](#)? Увлечение музыкой заменило программирование и чтение визуальных новелл. Также я повстречал свою новую любовь - язык программирования Rust.



"Secret tool" всё же появился позже - сладкий синт-торт Virus TI, подаренный супругой на ДР

Несмотря на всё вышесказанное, у меня всё ещё остаётся несколько идей для приложений в сфере музыкальной индустрии:

- В первую очередь можно было бы переписать Matchering на Rust, оптимизировав производительность и использование памяти, а также избавившись от таких костылей, как Docker. Но я не уверен, что в экосистеме Rust найдутся все нужные “кирпичики” для построения подобного процесса обработки. Также я не уверен, что использование такого инструмента было бы удобнее, чем элементарный мастеринг в браузере на таких ресурсах, как [Songmastr](#) или [Moises](#)
- Также я задумывался о создании собственного прототипа [DAW](#) на Rust. Используя этот безопасный язык при разработке DAW,

можно было бы избавиться от всех возможных внезапных вылетов и зависаний, которые нам так известны по работе в [Cubase](#) и [FL Studio](#). Также можно было бы спроектировать более современные этапы аранжировки и сведения, сократив всю возможную рутину к минимуму. Но, опять же, я не уверен в своих навыках и наличии свободного времени для подобной колоссальной задачи. Да и ребята из [Meadowlark](#) уже, вроде бы, постепенно решают эту задачу

- Ещё у меня были идеи применения техник глубокого обучения в классических VST плагинах и не только. Это могло бы открыть новые грани звучания: нейросетевой синтез неизведанных звуков, нейросетевая обработка вокала и инструментов, нейросетевое сведение композиций. Всю “движуху” Stable Diffusion и Midjourney можно было бы попробовать применить в сфере музыки. Но это лишь мои предположения и догадки, т.к. должного опыта разработки плагинов на C++ у меня нет

На этой ноте я бы хотел поблагодарить вас за прочтение моей первой (и, надеюсь, не последней) статьи. Если вас заинтересовал [Matchering](#), вы можете ознакомиться с [его репозиторием на GitHub](#). Большое спасибо за уделённое время и с Новым 2023 годом!

Если эта публикация вас вдохновила и вы хотите поддержать автора — не стесняйтесь нажать на кнопку