DOI 10.37882/2223–2966.2021.03.32

КОРРЕКТИРОВКА АУДИОСИГНАЛА ПРИ МОНТАЖЕ АУДИОЗАПИСЕЙ В ПРОГРАММНОЙ СРЕДЕ AUDACITY®, ИСПОЛЬЗУЯ МУЛЬТИФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ ЯЗЫКА ПРОГРАММИРОВАНИЯ NYQUIST

AUDIO SIGNAL ADJUSTMENT WHEN EDITING AUDIO RECORDINGS IN THE AUDACITY® SOFTWARE ENVIRONMENT, USING NYQUIST PROGRAMMING LANGUAGE MULTIFUNCTIONALITIES

V. Taran

Summary. The scientific article analyses engineering (applied) practices of computer sound editing based on the Audacity® software package. Key element of the analysis are the processes of adjusting smooth interval transitions of spectrum-frequency amplitude of audio signal from the state of low frequency potential is increased and vice versa when computer editing of audio recordings. The main tool for adjusting the audio signal is Nyquist programming language. Subject area of the article addresses the problem of improving the quality of processing fragments of audio material subject to various types of smooth direct polar fading and crossfading (including both polar and crossfading modulation of audio signal) based on the in-program capabilities of the Audacity® audio editor, including its terminal control via NyquistPrompt. The article summarizes and reveals the enormous potential of accurate computer editing based on freely distributed software and improves the existing engineering approaches to the processes of fading and crossfading. Strategies for using the Nyquist language programming code as a special ultra precise tool organically following-up the traditional interface-oriented program options are justified. An important role of the processes dealing with control and management during the routine operations in the field of applied audio engineering that contribute to the improvement of general engineering approaches that ensure timely solution of tasks is assigned in the article too.

Keywords: computer processing of audio material, spectrum-frequency amplitude, fading, crossfading, Audacity®, Nyquist, NyquistPrompt.

Таран Василий Васильевич
К. культурологии, Лаборатория компьютерного дизайна и прикладной информатики «SPLASH»; ФГБУН «Всероссийский институт научной и технической информации РАН»
allscience@lenta.ru

Аннотация. Научная статья посвящена анализу инженерных (прикладных) практик компьютерного редактирования звука на основе программного комплекса Audacity®. Ключевым звеном анализа являются процессы корректировки плавных интервальных переходов спектро-частотной амплитуды аудиосигнала из состояний заниженного частотного потенциала — к повышенному и наоборот в условиях компьютерного монтажа аудиозаписей. В качестве основного инструмента корректировки аудиосигнала выступает язык программирования Nyquist. В своей предметной области статья затрагивает проблематику повышения качества обработки фрагментов аудиоматериала, подлежащих различным типам плавных прямых полярных и перекрёстных затуханий (в том числе при полярной модуляции и перекрёстной модуляции аудиосигнала) на основе внутрипрограммных возможностей аудиоредактора Audacity®, включая его терминальное управление посредством NyquistPrompt. Структура статьи обобщает, систематизирует и раскрывает колоссальный потенциал возможностей точного компьютерного редактирования на основе свободно распространяемого программного обеспечения, усовершенствуя ныне существующие инженерные подходы к процессам фейдинга и кроссфейдинга. Обоснованы стратегии применения программного кода языка Nyquist, как специального ультра точного инструмента, органично дополняющего традиционные интерфейсно-ориентированные опции программы. Немаловажная роль в статье отводится процессам, контролирующим общий ход выполнения рутинных операций в области прикладной аудиоинженерии, способствующих усовершенствованию инженерных практик, обеспечивающих своевременное решение поставленных задач.

Ключевые слова: компьютерная обработка аудиоматериала, спектро-частотная амплитуда, фейдинг, кроссфейдинг, Audacity®, Nyquist, NyquistPrompt.

Современные компьютерные программы обработки аудиоматериала имеют внушительный арсенал средств, помогающих звукооператору (звукорежиссёру или ассистенту по аудиомонтажу) воплощать творческие замыслы в практической плоскости. За последние несколько десятилетий компьютерные науки и аудиоинформатика смогли существенно улучшить прикладные технологии обработки аудиоданных, акцентировав внимание на объектно-ориентированных программных и аппаратно-программных компьютерных решениях.

В последнее десятилетие объектно-ориентированные программы имеют особый успех в профессиональных и дилетантских кругах любителей компьютерной техники. И это вполне объяснимо, потому как их интерфейс в большей степени позволяет выполнять различные операции без привязки к языкам программирования[1], которые могут быть разного уровня сложности и иметь отличные друг от друга технические характеристики. Заниженный потенциал терминального управления программой и перевод её на интерфейсно-зависимую основу открывает особые возможности для специалистов, чья область деятельности не связана напрямую с компьютерными науками и информационно-коммуникационными технологиями.

Кроме того, мозг человека устроен таким образом, что наглядные чётко выстроенные схемы, либо определённые предписания, правила и инструкции позволяют человеку лучше воспринимать информацию и последовательно её обрабатывать. Вместе с тем хочется отметить, что любая компьютерная программа это, прежде всего алгоритм, который впоследствии должен быть грамотно и желательно быстро интерпретирован вычислительной машиной, что, в свою очередь, также способствует качеству обработки информации при проектировании и просчёте аудиоматериала [1].

С точки зрения человека (компьютерного пользователя, оператора звука) быстрый и понятный для восприятия отклик программы позволит лучше ориентироваться в происходящих технических процессах и соответственно эффективнее управлять ими. За визуальное отображение технической информации в архитектуре любой программы отвечает её интерфейс. Чем выше его качество (краткость и информативность выдаваемых пользователю программы сообщений, правильное расположение управляющих элементов программы, симметрично и геометрически правильно подобраны отображаемые управляющие модули и элементы) тем быстрее протекают процессы принятия решений человеком — оператором. Иными словами, усовершенствуется качество *технической коммуникации* между оператором и вычислительной машиной — компьютером[2]. Именно поэтому современный вектор развития компьютерных наук и информационно-коммуникационных технологий (включающих в себя телекоммуникационные услуги) направлен на объектно-ориентированные компьютерные системы.

Но, к сожалению, объектно-ориентированные компьютерные редакторы, располагающие довольно развитым программным интерфейсом, не всегда могут разрешать узконаправленные проблемы. Как правило, эти проблемы связаны с креативной стороной вопроса. В области обработки звука здесь речь идёт, в первую очередь, о дизайне, редизайне, аудиосинтезе, очистке аудиоматериала (полном или частичном восстановлении акустического спектра) и нестандартном микшировании аудиофрагментов [2,3].

Поскольку мы подразумеваем современные компьютерные программы по обработке аудиоматериала, то их интерфейс в большей степени ориентирован на выполнение стандартных (рутинных) операций (нормирование амплитуды звука аудиодорожек, исправление отклонения стерео и моно дорожек от центральной вводной фазы[3], наложение различных специальных эффектов, стандартные процедуры коррекции аудиоформы и т.д.). Такие интерфейсно-ориентированные опции очень полезны, если оператор монтажа выполняет стандартные (утилитарные) действия по обработке аудиосигнала. И немного иначе дела будут обстоять в ситуации, когда оператор захочет расширить собственные креативные возможности за счёт скрытого потенциала программы.

---

[1] Прим.автора. Концепции «интерфейсно-ориентированности» и «объектно-ориентированности» во многом схожи, различие состоит лишь в том, что объектно-ориентированные программные среды, способны манипулировать языками программирования различной степени сложности, используя интерфейс программы как командную оболочку, тем самым позволяя пользователю — программисту составлять программы по принципу конструктора, подключая уже готовые библиотеки и прочие технические составляющие.

[2] Прим. автора. В настоящее время преимущественно компьютером, в силу многоаспектности выполнения поставленных перед ним задач. Современные бытовые компьютеры (персональные компьютеры) и рабочие станции (станции digital-проектирования) способны решать огромное количество сложных и утилитарных задач, выполняя вычисления единовременно (в некоторых случаях могут использоваться параллельные и мультипараллельные вычисления), данное обстоятельство существенно отличает компьютеры от вычислительных машин, которые были популярны и широко использовались в середине 50-х годов двадцатого века. Отличие также состоит и в аппаратной архитектуре вычислительной машины (ограничение по мощности вычисления, либо чересчур мощная оснастка для вычислений), которая, как правило, также заточена под выполнение конкретных операций и ориентирована на специализированную операционную систему. Компьютеры (в силу своей многозадачности) способны эффективно решать задачи прикладного характера в области аудиоинженерии и обработки аудиоданных повышенной сложности.

[3] Прим. автора. Отклонение аудиоданных от центральной вводной фазы является наиболее распространённой погрешностью при записи аудиоматериала, имеющего различный тип спектро-акустических свойств. Причины данной погрешности ввода аудиоданных в компьютерную звукозаписывающую программу могут быть разные: задержка аудиосигнала на входе (вызванная неправильной аппаратной коммутацией оборудования или программной маршрутизацией аудиосигнала — цепочка драйверов ⟶ операционная система ⟷ аудиоредактор), неправильная настройка микрофонов (особенно тех, которые имеют встроенную функцию фонокоррекции и промежуточного затухания входящего аудиосигнала для устранения шипящих и фонящих пауз между произнесёнными на микрофон словами или куплетами) и т.д. Кроме того, причины могут крыться и в неправильной программной конвертации аудиоматериала, часто приводящей к смещению аудиоданных от центральной фазы, проблема проявляется при конвертации аудиоматериала из одного оригинального аудиоформата (к примеру, WAVE) в другой (MP3, OGG).

Рис. 1. Логотипы программы Audacity® олицетворяющие её эволюционное развитие.
В середине ныне действующий логотип.

Рис. 2. Логотип обучающей продукции Audacity® для онлайн и оффлайн каталогов
(Логотип построен автором, с оригинального макета находящегося по адресу: https://www.microsoft.com/ru-ru/p/audacity-user-guide/9pjzwlx9zrxj?cid=msft_web_appsforwindows_collection&activetab=pivot: overviewtab)

Конечно, на рынке программной продукции сегодня представлен обширный пул программ специального назначения (включая редакторы по обработке звука). И в зависимости от лицензионных регламентов, нормами которых регулируется их распространение, имеется возможность выбора подходящего для подобных операций продукта. Одним из таких ярких продуктов, получившим широкое распространение в сети Интернет и зарекомендовавшим себя среди специалистов, чья рабочая деятельность связана с компьютерной обработкой аудиоданных, является программный комплекс Audacity®[1].

Программный комплекс Audacity® — это специализированное кроссплатформенное свободно распространяемое программное обеспечение, призванное решать задачи начального, среднего и высшего уровней в области компьютерного монтажа аудиоматериала и обработки аудиосигнала нелинейным путём.

Audacity® имеет развитую электронную образовательную оболочку, позволяющую пользователям на интуитивно понятном уровне управлять процессами обработки звука. Именно поэтому в Audacity® имеется огромное количество внешних гиперсвязей, реализуемых через вкладки «Краткая справка» и «Подробная справка», позволяющих оперативно находить информацию, необходимую пользователям программы.

Audacity® — это как раз тот самый случай, когда аудиоредактор имеет расширенный программный интерфейс и обладает достаточным арсеналом встроенных модулей спектрочастотной и сонограммно-амплитудной обработки аудиосигнала, которые можно свободно применять при проведении стандартных и нестандартных процедур, направленных на улучшение качества аудиоматериала, а также удобно совмещать собственную интерфейсно-ориентированную систему с внешними вспомогательными средствами обработки аудио — NyquistIDE.

Несомненно, приведённая выше характеристика рассматриваемого нами программно-технического средства способствует и повышению точности корректировки обрабатываемого аудиосигнала при всех видах монтажа различных фрагментов аудиоматери-

[1] Audacity® — официальный сайт, компьютерного аудиоредактора: www.audacityteam.org.

ала. Речь идет, прежде всего, о расширении[1] технических средств (модулей) обработки звука за счёт дополнительной программной базы, выполненной как отдельное приложение (NyquistIDE), так и имеющее интеграционные связи непосредственно с редактором обработки аудиоматериала (Audacity®/Nyquist Prompt).

NyquistIDE это интегрируемая среда разработки, позволяющая разрабатывать новые мини аудиопрограммы, командные сценарии (в виде макросов), утилиты и аудиоприложения, которые могут быть загружены как отдельные компоненты, так и как компоненты, имеющие связи с внутрипрограммной архитектурой Audacity® [4].

Nyquist Prompt это встроенный модуль, интерпретирующий последовательность действий программного кода языка Nyquist[2]. Фактически Nyquist Prompt выполняет функции программного менеджера языка программирования Nyquist внутри программы Audacity®. Сам же язык программирования Nyquist в основном базируется на диалекте языка LISP[3]—XLISP, что существенно расширяет композиционную базу декларативного программирования на данном языке.

Язык программирования Nyquist успешно зарекомендовал себя как язык для синтеза аудиоматериала, поскольку имеет широкие возможности для проведения лабораторных исследований звука (написания отдельных модулей для программы Audacity® в рамках Nyquist-плагинов) и генерации новых сигналов в отдельной среде NyquistIDE. Также Nyquist это и инженерный язык, позволяющий контролировать процессы обработки аудиосигнала с ювелирной точностью внутри программной среды Audacity® посредством Nyquist Prompt.

С авторской точки зрения, язык программирования Nyquist является одним из передовых узкоспециализированных средств, способный решать достаточно сложные задачи, это утверждение делает предпочтительным Nyquist и для аудиоинженерии в целом, открывая возможности не только для внутрипрограммной прикладной обработки аудиоматериала в Audacity®, но и совершенствуя механизмы потоковых аудиотрансляций посредством управления аудиоформатами [5,6].

В рассматриваемом нами языке действительно скрыт огромный мультифункционал многолетних инженерных практик обработки звука, вобравших в себя лучшее из них и воплотившийся в современный узкоспециализированный технический язык, способный достаточно оперативно решать поставленные задачи. Такое положение дел значительно расширяет традиционный интерфейсно-ориентированный функционал программы за счёт интеграции программных сценариев и микропрограмм (модулей обработки звука) непосредственно в сам аудиоредактор. К примеру, программный код, имеющий описание подключаемого модуля составленный программистом на языке Nyquist, имеет возможность функционировать как отдельное приложение в NyquistIDE, так и как зависимая часть программы (внутренний подключаемый модуль) в среде аудиоредактора Audacity® через посредничество встроенного проводника выполнения команд Nyquist Prompt.

Важной задачей в области полноценного компьютерного аудиомонтажа является обеспечение всеобъемлющего контроля над процессами повышения и понижения многочастотной амплитуды аудиосигнала на определённом отрезке времени[4], что характерно для студийных процедур обработки аудиосигнала, проводимых с целью создания плавных переходов между аудиодорожками, либо при дизайн-проектировании сложной аудиоформы — для уточнения затухания отрезка времени аудиоматериала (обычно это последняя часть аудиокомпозиции либо начало трека для плавного введения нелинейного аудиопотока).

В Audacity®, как и в любом подобном профессиональном аудиоредакторе, имеются специальные интерфейсно-ориентированные модули управления затуханием аудиоматериала[5] Adjustable fade, Crossfade

[1] Прим. автора. Программа также хорошо взаимодействует с интегративными средами VST и LADSPA/LV2.

[2] Прим. автора. Важным моментом является то, что язык программирования Nyquist встроен в программу Audacity® и его широкие инженерные возможности можно реализовывать внутри самой программы, предварительно отладив алгоритм действий в NyquistIDE. Отладка программного кода потенциального модуля обработки аудиоданных в среде NyquistIDE позволит избежать ошибок обработки данных в Nyquist Prompt (уже как приглашение языка программирования Nyquist внутри Audacity®).

[3] Прим. автора. Далее в статье автор упоминает название демонстрируемого языка программирования с большой буквы и дальнейшим строчным написанием — Lisp. Это объясняется историческими причинами его развития. Первоначально он именовался как LISP, но более свежие его версии и дополнения идут в каталогах с названием «Lisp». Поэтому в статье автор руководствуется преемственностью и по историческим причинам в начале упоминает его как «LISP», а далее в более современном варианте — «Lisp».

[4] Прим. автора. То есть корректировка затуханий и нарастаний начальных и конечных фрагментов аудиосигнала.

[5] Прим. автора. Основным отличием Audacity® от других профессиональных программных решений в данной области является возможность внутренней программной корректировки вводимых значений и тонкой подстройки исполнения алгоритмом просчёта применяемого к звуковому наполнению эффекта. Такой контроль устанавливается, прежде всего, за счёт внутренней интегрированной системы управления модулями Nyquist Prompt. Ввод команд может происходить, не отрываясь от объектно-ориентированной формы модуля (плагина), тем самым можно устанавливать необходимые зависимости и корректировать вводимые значения, подкрепляя визуализацию исполняемых оператором действий, регистрацией

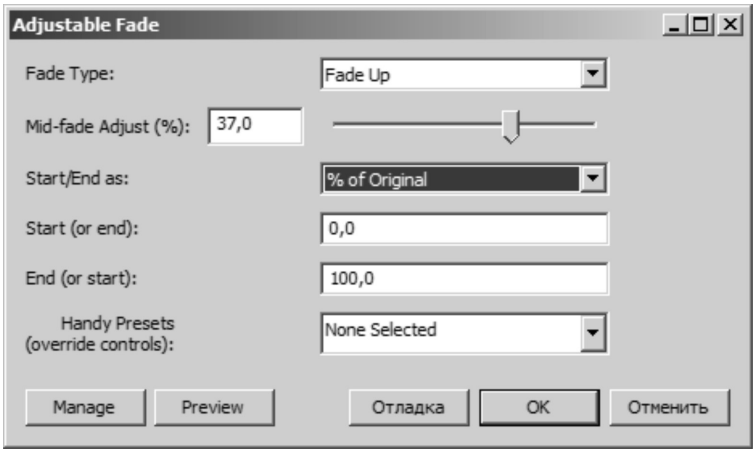


Рис. 3. Интерфейс модуля обработки аудиоданных Adjustable fade.

Рис. 4 Опция интерфейса модуля обработки аудиоданных Adjustable fade «Start/End as», показывает процент от оригинала и мощность в Децибелах.

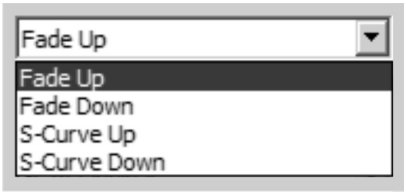


Рис. 5. Опция интерфейса модуля обработки аудиоданных Adjustable fade «Fade Type», меню демонстрирует список предусмотренных алгоритмом типов затуханий.

tracks, Crossfade Clips и Studio Fade Out, Fade In, Fade Out. Мы проведём технический анализ функциональных возможностей программного кода, написанного на языке программирования Nyquist с целью понимания процессов обработки аудиосигнала программным путём. Изменение программного кода данных модулей[1], с авторской точки зрения, позволит эффективнее разрешать узкопоставленные задачи в области прикладной аудиоинженерии.

Adjustable fade (регулируемое затухание) — модуль отвечающий за регулируемое затухание аудиосигнала внутри программной среды Audacity®, модуль берёт на себя функции штатного плагина по обеспечению снижения либо увеличения частотной амплитуды аудиосигнала с возможностью регулирования степени нарастания и спада выделенных интервальных значений сонограммы.

Опционально модуль поддерживает два типа затухания Fade Up и Fade Down, которые имеют возможность формирования амплитуды от низкого уровня

событий на языке программирования Nyquist. Конечно, данная функция в большинстве своём доступна профессионалам, поскольку необходимы знания Lisp, некоторых его диалектов (XLISP) и других Lisp-ориентированных языков. Помимо этого неплохо было бы ещё ориентироваться в синтаксисе языков SAL и Nyquist, но усвоенные по данным языкам знания (в сочетании с интерфейсно-ориентированным аудиоредактором Audacity®) откроют огромные возможности по редактированию аудиоматериала, и позволят раскрыть все креативные замыслы компьютерных аудиоинженеров, инженеров сведения, и аудиомонтажа.

[1] Прим. автора. Разумеется, в соответствии с лицензией на их использование. Большинство подобных модулей распространяется по общедоступной лицензии GNU LGPL, v.2.0. С информацией о Лицензии на русском языке можно ознакомиться в статье: Компьютерная очистка аудиоматериала штатными средствами программы Audacity® (программно-ориентированный подход) / В.В. Таран // Современная наука: актуальные проблемы теории и практики (Серия естественные и технические науки) /// Информатика, вычислительная техника и управление.— 2020.— № 9.— С. 123–127 | разд. Приложение | [ISSN2223–2966]. (DOI 10.37882/2223–2966.2020.09.37).

Рис. 6. Опция интерфейса модуля обработки аудиоданных Adjustable fade «Mid-Fade Adjust (%)», бегунок устанавливает уровень в процентах.

Рис. 7. Опция интерфейса модуля обработки аудиоданных Adjustable fade «Mid-Fade Adjust (%)», бегунок устанавливает уровень в процентах, демонстрируемый уровень — 0.

к более высокому. Start/End as — данный раскрывающийся список позволяет выбирать между процентами (%) или децибелами (дБ) в качестве единиц измерения, которые ниже будут использоваться в полях параметров Start и End.

Последние два текстовых поля должны содержать по одному номеру в каждом. Они предопределяют начальное и конечное усиление (величину усиления или степень затухания). Неважно, вводятся ли они с начальным усилением в первом текстовом поле и конечным усилением — в последнем, либо наоборот, поскольку направление затухания определяется положением «Fade Type» которое санкционирует выбор модели затухания.

Fade Type — фактически устанавливает режим процентного соотношения (нарастания/спада) от исходной амплитуды либо предоставляет возможность регулировки громкости в Децибелах (% of Original, dB Gain).

Fade Up — призван обеспечить линейное (простое) плавное усиление от установленного значения с низким коэффициентом усиления до установленного значения с высоким коэффициентом усиления.

Fade Down — наоборот призван обеспечить линейное (простое) плавное затухание[1] кривой от установленного значения с высоким коэффициентом усиления до установки с более низким коэффициентом усиления.

Помимо опций Fade Up и Fade Down, модуль располагает и подопциями S-Curve Up и S-Curve Down. S-Curve Up — формирует двойную кривую, которая изгибается в разные стороны. Данная подопция повышает уровень громкости потом прогрессивно (более круто) уровень будет повышаться к середине затухания, прежде чем постепенно уравниваться. Работа со средними точками (в режиме S-Curve) довольно хорошо описана в руководстве пользователя Audacity® [7,8]. Некоторые профильные вопросы, касающиеся, прежде всего, микширования звука, также хорошо изложены в книге (русск. назв. Книга об Audacity®: запись, редактирование, сведение и мастеринг (со свободным аудиоредактором) [9]. S-Curve Down — наоборот первоначально снижает уровень громкости, затем снижение продолжается (более круто) к середине затухания, прежде чем постепенно начнёт выравниваться. Опция Mid-fade Adjust (регулировка среднего затухания) позволяет контролировать и вести расчёт изменения формы затухания в зависимости от её выбора в раскрывающемся меню Fade Type.

Регулятор управления имеет диапазон от –100,0 до 100,0%. По умолчанию движок находится в положении 0%, установка регулятора управления на уровне больше нуля приводит к поднятию центра затухания, в то время как при отрицательных значениях вектор движения центра затухания будет направлен вниз. При использовании «Fade Type: = Fade Up» (значение нуль по умолчанию приведёт к линейному затуханию аудиосигнала).

Далее рассмотрим, некоторые опциональные характеристики, соответствующие функциям меню:

1. Значения, превышающие ноль, сначала приведут к быстрому росту затухания, прежде чем выровняются до более высокого уровня усиления.
2. Значения меньше нуля, сначала приведут к медленному росту затухания, а затем постепенно к более быстрому его росту.

[1] Прим. автора. Линейные затухания — эти основные затухания их применяют к выбранному звуку таким образом, что амплитуда выбора переходит от абсолютной тишины к исходной амплитуде (Нарастание) или от исходной амплитуды к абсолютной тишине (Затухание). Форма затухания линейна, поэтому она выглядит как прямая линия от начала до конца (при просмотре в режиме Waveform View Mode по умолчанию). Таким образом, скорость нарастания или затухания является постоянной на протяжении всей длины аудиоматериала, и она также полностью зависит от длины отметки, выбранной для затухания.

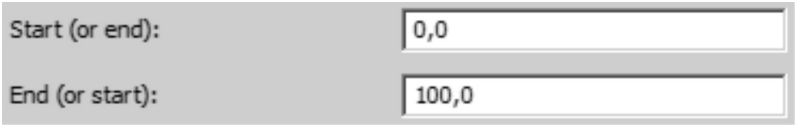


Рис. 8. Опция интерфейса модуля обработки аудиоданных Adjustable fade, меню установки значений «Start (or end)/End (or start)».

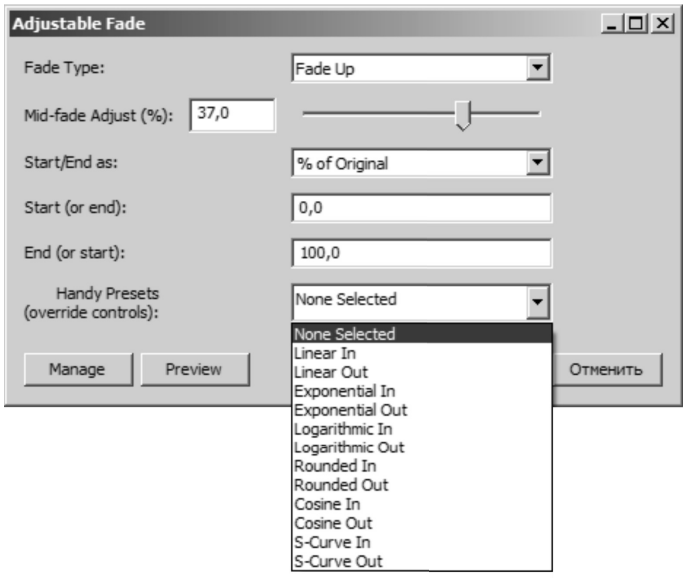


Рис. 9. Опция интерфейса модуля обработки аудиоданных Adjustable fade Handy Presets (override controls), демонстрирует раскрывающийся список различных геометрий затухания аудиосигнала.

3. Чем дальше от нулевого уровня установка, тем более искривлённым будет затухание.
4. Среднее усиление затухания никогда не будет меньше начала затухания или больше конца затухания.

Когда мы используем «Fade Type: = Fade Down» (значение нуль, по умолчанию, приведёт к линейному затуханию аудиосигнала). Далее рассмотрим некоторые *опциональные характеристики,* соответствующие функциям меню:

1. Значения, превышающие ноль, заставят затухание снижаться сначала постепенно, а затем все более резко по мере приближения к конечному фазовому уровню.
2. Значения, меньше нуля, заставят затухание сначала быстро снижаться, а затем постепенно выравниваться.
3. Чем дальше от нулевого уровня установка, тем более искривленным будет затухание.
4. Усиление среднего затухания никогда не будет больше начала затухания или меньше конца затухания.
5. При использовании «Fade Type: = S-Curve Up или S-Curve Down» на полпути через затухание усиление будет точно на полпути между начальной точкой и конечной точкой.
6. При значениях больше нуля затухание сохранит свой характер «двойной» кривой, но будет немного выше в средней точке.
7. При значениях ниже нуля затухание сохранит свой характер «двойной» кривой, но будет немного ниже в средней точке.

Поля Start (or end) — End (or start) используются для контроля значений конечного коэффициента.

Start (or end) — позволяют устанавливать начальный либо конечный коэффициент усиления. При использовании процентных единиц значение по умолчанию 0 — (тишина), означает начало затухания от тишины[1].

[1] Прим. автора. Что бы детально изучить опции штатного модуля, предназначенного для выполнения операций по видам затухания аудиосигнала, можно сгенерировать «белый шум» и применять данные опции последовательно, на фонограмме будет видно, какая из опций является подходящей для обработки звука.

Таблица 1. Ручные предустановки и краткая расшифровка их действий.

№	Предустановки	Инструкции
1	None Selected	Настройка вручную/ Функциональны: Тип затухания, Средняя регулировка затухания, настройки Начало — Конец.
2	Linear In	То же самое, что и эффект Fade In.
3	Linear Out	То же самое, что и эффект Fade Out.
4	Exponential In	Уровень растёт экспоненциально, также как и при использовании Envelope Tool.
5	Exponential Out	Уровень экспоненциально снижается, также как и при использовании Envelope Tool.
6	Logarithmic In	«Выпуклая» кривая, растущая умеренно круто, а затем всё менее круто от тишины.
7	Logarithmic	«Выпуклая» кривая, сначала снижающаяся умеренно, а затем более быстро.
8	Rounded In	«Выпуклая» кривая, которая от тишины сначала растёт резко, а затем менее резко.
9	Rounded Out	«Выпуклая» кривая, которая сначала постепенно опускается от начального уровня, затем постепенно всё более резко переходит к довольно крутому концу.
10	Cosine In	«Выпуклая» кривая, растущая умеренно круто от тишины, а затем постепенно выравнивается, чтобы создать плавный переход к исходному уровню.
11	Cosine Out	«Выпуклая» кривая с плавным переходом от исходного уровня, а затем все более круто снижающаяся до тишины.
12	S-Curve In	«Двойная» кривая с мягким переходом от тишины, затем растущая всё более резко, после чего выравнивается, чтобы обеспечить плавный переход к исходному уровню.
13	S-Curve Out	«Двойная» кривая, создающая плавный переход от исходного уровня, постепенно снижающаяся всё быстрее, затем выравнивается, чтобы обеспечить плавный переход к тишине. Аналогично эффекту Studio Fade Out.

End (or start) — позволяет установить конечный (или начальный) коэффициент усиления. Когда мы используем процентные единицы значение по умолчанию 100 (исходный уровень или единичное усиление) даёт возможность окончательного затухания в полном его объёме. Модуль поддерживает изменяемые настройки в ручном режиме (Handy Presets (override controls)) и позволяет пользователю (оператору по компьютерному монтажу звука) выбирать фиксированные предустановленные формы кривых[1]: линейную, экспоненциальную, логарифмическую, закругленную, косинусную и S-образную.

Все предустановки[2] контролируют рост от тишины до исходного уровня, либо затухание от исходного уровня до тишины. Следует обратить внимание, что в отличие от эффектов Fade In[3] и Fade Out[4] в верхней части меню Эффектов, этот эффект будет рассматривать пустое пространство в выбранной области как тишину,

[1] Прим. автора. Выбор предустановленных кривых приведёт к переопределению всех других диалоговых окон настройки параметров для этого эффекта. Чтобы сделать другие параметры работоспособными, вам нужно будет установить режим «None Selected» из этого раскрывающегося списка.

[2] Прим. автора. Настройки предполагают, что Start и End задаются как «% of Original». Обратите внимание, что порядок настроек «Start» и «End» не имеет значения.

[3] Прим. автора. Fade In — нарастание часто применяется в течение очень короткого выбора звука (менее секунды). Вы можете получить более «музыкальный» результат, применяя линейное нарастание в три раза чаще к одному и тому же звуковому выбору. Это приблизительно соответствует экспоненциальной форме нарастания.

[4] Прим. автора. Fade Out — затухание часто применяется к более длительному выбору, чем нарастание, обычно к выбору (длиной до десяти секунд).

Таблица 2. Предполагаемые акустические эффекты, достигаемые в результате использования модуля обработки аудиосигнала Adjustable fade по типу затухания.

Желаемый Эффект	Тип Затухания	Регулировка Среднего Затухания(%)	Начало (или Конец)	Конец (или Начало)	Область применения
Линейное нарастание от тишины до исходного уровня	Fade Up	0	0	100	Сведение, мастеринг, реставрация аудиоматериала
Линейное затухание от исходного уровня до тишины	Fade Down	0	0	100	Сведение, мастеринг, реставрация аудиоматериала
Линейное затухание от исходного уровня до половины объема	Fade Down	0	50	100	Сведение, мастеринг, реставрация аудиоматериала
Экспоненциальное затухание аналогичное использованию Envelope Tool	Fade Down	Менее 0	0	100	Сведение, мастеринг, реставрация аудиоматериала
Затухание для перекрестного затухания равной мощности	Fade Down	+50	0	100	Сведение, мастеринг, реставрация аудиоматериала
Нарастание для перекрестного затухания равной мощности	Fade Up	+50	0	100	Сведение, реставрация аудиоматериала
Аналогия эффекту «Studio Fade Out»	S-Curve Down	0	0	100	Сведение, мастеринг, реставрация аудиоматериала
Плавное изменение от половины исходного объёма к объёму вдвое превосходящему исходный	S-Curve Up	0	50	200	Сведение, мастеринг, реставрация аудиоматериала

поэтому важно выбрать именно ту область, к которой вы хотите применить эффект (таблица 1).

Для наглядности примеров использования предустановленных настроек приведём список желаемых эффектов при использовании данного модуля

После приведённых примеров обратимся к интерфейсу модуля, конкретно к его клавишам управления.

На данном этапе мы постарались внимательно рассмотреть весь интерфейсно-ориентированный функционал модуля Adjustable fade (регулируемое затухание), и теперь самое время перейти к программно-ориентированному функционалу. Программно-ориентированный функционал программы Audacity® основывается как на сочетании технических средств внутренней архитектуры (Nyquist Prompt), так и на внешних модулях спутниках, помогающих вести разработку и гибкую корректировку[1] тех же модулей для *внутренней архитектуры* (Nyquist IDE/jNyqIDE.jar.). Данный аспект применительно к обработке аудиоматериала повышенной сложности значительно расширяет функционал штатных объектно-ориентированных модулей, позволяя решать задачи нестандартным путём. В нашем случае такой нестандартный путь лежит между задачами по оптимизации спектро-частотной амплитуды аудиосигнала и решением прикладных задач в области горизонтального и вертикального сведения аудиофрагментов. Рассмотрим несколько нестандартных обстоятельств (в области компьютерной аудиоинженерии), провоцирующих применение языка программирования Nyquist для решения крайне узконаправленных задач.

- Плавное снижение громкости спектро-частотной амплитуды при посекторном понижении низких частот[2].

[1] Прим. автора. Корректировка модуля должна производиться строго с соблюдением лицензионных соглашений языков программирования Nyquist и XLISP, перераспределение исходного кода Audacity® — с соблюдением лицензионного соглашения по Audacity®.

[2] Прим. автора. Под плавным снижением громкости спектро-частотной амплитуды с посекторным понижением низких частот подразумевается ступенчатое выделение разделов (зарезервированного) конечного фрагмента аудиокомпозиции с целью выделения средних и высоких частот для обеспечения эффекта рассеивающей модуляции*.

* Эффект рассеивающей (спадающей модуляции) — процесс сложения затухающих средних и верхних частот (при уменьшении низких частот), эмитирующий уплотнение сходящего аудиосигнала для выделения (подчёркивания) фрагмента вокала, инструмента либо аранжировки. Эффект применяется в звукорежиссуре (при мастеринге) — для повышения

Таблица 3. Универсальные управляющие клавиши, используемые для контроля над процессами обработки аудиоматериала различными модулями программы Audacity®. Правый столбец таблицы кратко описывает их действия.

№	Клавиша управления	Инструкция
	Manage/ Управление	Показывает раскрывающееся меню, позволяющее управлять предустановками для инструмента и видеть некоторые детали об инструменте. Дополнительные сведения см. в разделе «Управление предустановками».
	Preview/ Предварительный просмотр	Позволяет предзагрузить предварительно обработанный модулем аудиоматериал. Предварительный просмотр проводит короткую предварительную демонстрацию того, как будет звучать звук, если эффект будет применен с текущими настройками, без внесения фактических изменений в звук. Продолжительность предварительного просмотра определяется вашими настройками в меню Edit > Preferences > Playback, установка по умолчанию — 6 секунд.
	Debug/Отладка	Отладчик применяет эффект к выбранному звуку с текущими настройками эффекта, но в отличие от «OK» эффект работает в режиме отладки. Это, в первую очередь, полезно при написании или редактировании плагинов Nyquist. Дополнительно к обычному поведению плагина открывается «окно отладки» для отображения сообщений об ошибках. Обычно окно отладки бывает пустым.
	OK	Применяет эффект к выбранному звуку с текущими настройками эффекта.
	Cancel/Отмена	Прерывает эффект и оставляет звук без изменений.

- Реверберация верхних частот при снижении громкости спектро-частотной амплитуды[1].
- Выделение из общего массива аудиоформы некоторых частот, соответствующих музыкальным инструментам или вокалу[2].

Для оперативного решения представленных нами задач и в принципе задач нестандартного типа необходимо проанализировать ход выполнения действий объектно-ориентированного модуля Adjustable Fade, с целью уточнения программного сценария изложенного на языке программирования Nyquist. Как мы уже заявляли чуть ранее, Nyquist очень много унаследовал от Lisp, поэтому для полного понимания описываемых в программном коде действий рекомендуем освежить в памяти структуру Lisp. Разработчики языка программирования Nyquist в предисловии к *официальному учебнику* рекомендуют труд Дэвида Турецкого (David Touretzky) как достаточно полное и схематичное издание о языке Lisp [10]. Но поскольку оно издано в 1990 году, мы осмелимся обратить внимание нашего читателя ещё на один хороший учебник Питера Сайбла[3] (Seibel Peter) (Общая практика по Лиспу) [11]. И поэтому самое время обратиться к программному коду данного модуля (код 1).

Рассматриваемый нами программный код разработан Стивом Далтоном[4] (Steve Daulton) и может использоваться (в практических целях) в соответствии с регламентами GNU General Public License version 2[5].

качества общего фейдинга композиции и на стадии сведения — для повышения качества перекрёстных переходов между аудиокомпозициями и фрагментами аудиокомпозиций (композиционный строй).

[1] Прим. автора. Под реверберацией верхних частот при общем снижении громкости спектро-частотной амплитуды подразумевается настраиваемый микрооткли верхних частот в конечной (подлежащей фейдингу) аудиокомпозиции. Данный эффект необходим, когда требуется создать геометрию пространства для отдельно выбранного аудиофрагмента. Эффект позволяет совершенствовать акустические практики «присутствия».

[2] Прим. автора. Под выделением из общего массива аудиоформы некоторых частот, соответствующих музыкальным инструментам или вокалу, подразумевается процесс экструзии спектро-частотных аудиофрагментов. Эффект является универсальным в практике компьютерного редактирования звука и применяется в том числе для имитации разложения парного стериосигнала на фрагментарные мультиканалы. Как правило, данный эффект создаётся в аудиоредакторах в спектральном режиме редактирования аудиокомпозиций.

[3] Прим. автора. Питер Сайбл — разработчик программного обеспечения, имеющий заслуженную репутацию в компьютерном мире, которая подтверждается солидным стажём работы. На заре развития Интернета увлекался хакерством, тестировал на устойчивость компьютерные программы. Он участвовал в процессе становления языка программирования Java в качестве одного из первых сотрудников WebLogic, которая после её приобретения BEA стала драйвером быстрого роста последней в сфере J2EE. Он также преподавал программирование на Java на курсах повышения квалификации в Калифорнийском университете в Беркли (UC Berkeley Extension). Является автором Практического общего руководства по LISP, изданного Apress.

[4] Прим. автора. Стив Далтон (Steve Daulton), в музыкальном мире известен по прозвищу «Стив — скрипка» (Steve the Fiddle) — музыкант, композитор, звукорежиссер, звукооператор, программист и энтузиаст проекта Audacity® (основные интересы — разработка и отладка плагинов Nyquist). Он также является ведущим специалистом в области обнаружения и исправления ошибок Nyquist-плагинов в операционных системах Linux, MS Windows.

[5] Прим. автора. Доработка программного кода и его изменение возможно только согласно приведённому соглашению с упоминанием первоначального авторства Nyquist-плагина.

```
(defun get-input (sig)
;; (if *previewp* sig (multichan-expand #'trim-input sig)))
  (if (get '*track* 'view)  ;NIL if preview
      sig
      (multichan-expand #'trim-input sig)))

(defun trim-input (sig)
"Trim input when previewing."
  (let ((dur (min (get-duration 1)
                  (get '*project* 'preview-duration))))
    (setf sig (extract-abs 0 dur *track*))))

;;; invalid values
(defun check-values (x y)
  (if (= units 0)  ;percentage values
    (cond
      ((or (< x 0)(< y 0))
        (throw 'err (format nil "~aPercentage values cannot be negative." err)))
      ((or (> x 1000)(> y 1000))
        (throw 'err (format nil "~aPercentage values cannot be more than 1000 %."
        err))))
    (cond   ;dB values
      ((or (> x 100)(> y 100))
        (throw 'err (format nil "~adB values cannot be more than +100 dB.~%~%~
                                 Hint: 6 dB doubles the amplitude~%~
                                 \t-6 dB halves the amplitude." err))))))
```

Код 1

```
(defun get-input (sig)
;; (if *previewp* sig (multichan-expand #'trim-input sig)))
  (if (get '*track* 'view)  ;NIL if preview
      sig
      (multichan-expand #'trim-input sig)))
```

Код 2

```
(let ((dur (min (get-duration 1)
                (get '*project* 'preview-duration))))
  (setf sig (extract-abs 0 dur *track*))))
```

Код 3

Программный код[1] обеспечивает точный контроль над всеми действиями модуля Adjustable Fade[2] и позволяет производить настройки, адаптируемые к условиям решения конкретно поставленных задач. Начальный фрагмент кода отвечает за процедуры предварительного прослушивания предобрабатываемого аудиоматериала и устанавливает длину его прослушивания в соответствии с *установленными параметрами проекта программы* и выглядит следующим образом (код 2).

[1] Прим. автора. В силу символьной ёмкости программного кода мы будем производить анализ только наиболее значимых с точки зрения автора его фрагментов, с целью демонстрации его потенциальных функциональных возможностей.

[2] Прим. автора. Adjustable Fade — модуль, регулирующий процессы затухания аудиосигнала на определённых (выделенных, обозначенных маркерами) его участках. Программная структура разработана Стивом Далтаном (Steve Daulton) в декабре 2012 года. Программная структура при необходимости может быть скорректирована и полностью либо частично изменена в соответствии с лицензионным соглашением GNU General Public License, второй версии. Полный текст лицензии в англоязычном варианте доступен по ссылке: http://www.gnu.org/licenses/old-licenses/gpl-2.0.html (дата обращения к источнику: 15.01.2021). Информация о модификациях программного кода объектно-ориентированных модулей обработки звука представлена подробно (для профессионалов в области компьютерной аудиоинженерии и аудиоинформатики, специалистов в области прикладного и системного программирования), находится по адресу: http://wiki.audacityteam.org/wiki/Nyquist_Plug-ins_Reference (дата обращения к источнику: 15.01.2021).

```
;;; select and apply fade
(defun fade (sig type curve g0 g1)
  (check-values gain0 gain1)
  (mult (get-input sig)
    (case preset
      (0  (case type                    ; Custom fade
            (0 (simple (min g0 g1) (max g0 g1) curve))
            (1 (simple (max g0 g1) (min g0 g1) curve))
            (2 (raised-cos (min g0 g1)(max g0 g1) curve))
            (T (raised-cos (max g0 g1) (min g0 g1) curve))))
      (1  (linear 0 1))                 ; Linear In
      (2  (linear 1 0))                 ; Linear Out
      (3  (log-exp-curve -60 0))        ; Exponential In
      (4  (log-exp-curve -60 1))        ; ExponentialOut
      (5  (log-exp-curve 15.311 0))     ; Logarithmic In
      (6  (log-exp-curve 15.311 1))     ; Logarithmic Out
      (7  (simple-curve 0 1 0.5))       ; Rounded In
      (8  (simple-curve 1 0 0.5))       ; Rounded Out
      (9  (cosine-curve 0 1))           ; Cosine In
      (10 (cosine-curve 1 0))           ; Cosine Out
      (11 (raised-cos 0 1 0.0))         ; S-Curve In
      (t  (raised-cos 1 0 0.0)))))      ; S-Curve Out
```

Код 3

```
(0 (simple (min g0 g1)(max g0 g1) curve))
(1 (simple (max g0 g1)(min g0 g1) curve))      →  min g0 g1  max g0 g1 | curve  0 1
(2 (raised-cos (min g0 g1)(max g0 g1) curve)) →  max g0 g1  min g0 g1          1 0
                                                        cos
(T (raised-cos (max g0 g1)(min g0 g1) curve))))
```

Код 4

Здесь defun элемент, который предопределяет функцию и получает данные о вводе сигнала. При предварительном прослушивании (*previewp*) multichan-expand устанавливает мультиканальное и межканальное (m/s) расширение с последующей обрезкой (укорачиванием по времени) входящего потока (trim-input), код 3.

Далее идёт ввод обрезанного сигнала, let — создаёт локальную привязку к продолжительности (dur), min — определяет самое маленькое число продолжительности. get '*project* 'preview-duration — устанавливает промежуточную продолжительность звучания в соответствии с общей продолжительностью звучания открытого в программе проекта.

Setf[1] — символьная функция предназначена для установки значения поля в нашем случае полевым спецификатором выступает обрабатываемый сигнал. Extract-abs — извлечение отрезка продолжительности дорожки. Далее идёт фрагмент обработки недопустимых значений. Defun check-values (x y) предопределяет функцию проверки вводимых значений от x 0 до y 0. Cond — выражение условной оценки и управляющая конструкция пара (pair), состоящая из (pred expr…), когда pred — выражение предиката, а expr — оценка (если предикат не 0). Returns — значение первого выражения, предикат которого не 0.

$$\frac{(< x\ 0)(< y\ 0)}{0,1,2,3,5\ /\ 5,3,2,1,0}$$

при просчете от 0 до 5

Допускает ошибку «формат ноль — значения не могут быть отрицательными»

[1] Прим. автора. Функции и прочий инструментал языка программирования Nyquist, а также других языков программирования, после точек автор пишет с большой буквы, руководствуясь норами русского языка. В программировании технические функции и прочий инструментал языков программирования пишется с *маленькой буквы*, за исключениями оговорённых правил по *регистрозависимости* языков.

```
;;; linear fade to the power of pow
(defun simple-curve (g0 g1 pow)
  (curve-adjust g0 g1 pow
    (linear g0 g1)))

;;; cosine fade to the power of pow
(defun cos-curve (g0 g1 pow)
  (curve-adjust g0 g1 pow
    (cosine-curve g0 g1)))

(defun curve-adjust (g0 g1 pow env)
  (scale-curve g0 g1
    (if (= pow 1)
        env
        (snd-exp
          (mult pow
            (snd-log env))))))
```

Код 5

```
;;; (1)
(defun scale-curve (g0 g1 env)
  (sum (min g0 g1)
    (mult (abs (- g0 g1)) env)))
;;; (2)
(defun cosine-curve (g0 g1)
  (let ((step (hz-to-step (/ 0.25 (get-duration 1))))
        (phase (if (> g0 g1) 90 0)))
    (osc step 1 *sine-table* phase)))
```

Код 6

```
((or (> x 1000) (> y 1000))
    0,1,2,3,5/5,3,2,1,0
```

при просчете от 0 до 5

Допускает ошибку «формат ноль — значения не могут составлять более чем 1000%»

Cond — условно оценивает значения децибел

```
((or (> x 100) (> y 100))
   0,1,2,3,5/5,3,2,1,0
```

при просчете от 0 до 5

Допускает ошибку — «значения не может составить больше чем +100 дБ». Серым оттенком в конце фрагмента кода выделены возможные допущения, приводящие к выдаче ошибок Nyquist-обработки. Вот их более подробный предполагаемый перечень:

1. Значения процента могут быть отрицательными — допускает ошибку
2. Значения процента не могут составлять более чем 1000% — допускает ошибку
3. Значения ~adB не могут составлять больше чем +100 дБ ~% ~% ~

Подсказка: 6 дБ удваивает амплитуду ~%~

Половину дБ \e–6 амплитуда — допускает ошибку

Далее идёт фрагмент кода, определяющий выбор и применение затухания (код 3)

Defun fade — предопределяет данные затухания с последующей установкой типа кривой (g0 g1). Далее следует проверка усиления (gain0 gain1), mult — позво-

```
defun scale-curve= (g_  g_ env)
(min g0 g1) / (abs (- g0 g1)) env) → 1, g... |0| ...
    sum              mult
```

Код 7

```
/defun/   cosine   -   curve   = (g0 g1)
  (if (> g0 g1) 90 0)))        ↓          → ч.шаг=0.25
       phase            1,g... |0| ...
  osc  /*sine-table*             g0 ↔ g1  → прод.=1
  step 1
```

Код 8

```
;;; linear fade in, out
(defun linear (g0 g1)
  (control-srate-abs *sound-srate*
    (if (> g0 g1)                       ; g0 = g1 does not occur here.
        (pwlv 1 1 0)                    ; fade out
        (pwlv 0 1 1))))                 ; else fade in

;;; raised cosine fades
(defun raised-cos (g0 g1 curve)
  (setq curve
    (if (> curve 0)
        (exp-scale-mid (* 2 curve))       ; mid-point -3dB @ Adjust = 50%
        (exp-scale-mid (* 1.63 curve))))  ; mid-point -12dB @ Adjust = -50%
  (setf env
    (control-srate-abs *sound-srate*      ; sound srate required for accuracy.
      (cond
        ((= g0 g1) g0)    ; amplify
        ((> g0 g1)        ; fade down
          (snd-exp
            (mult (pwlv (- 1 curve) 1 1)
              (snd-log (raised-cosin 90)))))
        (t (snd-exp                       ; fade up
             (mult (pwlv 1 1 (- 1 curve))
               (snd-log (raised-cosin -90))))))))
  (sum (min g0 g1)
    (mult (abs (- g0 g1)) env)))
```

Код 9

ляет производить смешанное умножение звуков при получении входящего аудиосигнала. Входящий аудиосигнал определяется строчкой (mult (get-input sig). И его операции с кривыми схематично показаны на код 4.

Заметим, что в случаях 1) — 2) допустимой отметкой шага кривых может быть в min — 0 1, в max — 1 0. Это означает, что при использовании огибающих в Audacity® (к примеру, для смягчения тонов спада) кривая будет применяться всегда на шаг ниже, нежели при редактировании аудиоматериала только огибающими Audacity® без модуля Adjustable Fade. Далее идёт определение типа пользовательского затухания с математическими отметками шагов затуханий (типы затуханий выделены серым оттенком).

Следующий этап, который иллюстрирует фрагмент кода, расположившийся строчкой ниже, предполагает линейное (косинусообразное) затухание относительно основного усиления (код 5).

Здесь ключевыми значениями выступают параметры усиления g0 g1. Snd-exp — вычисляет экспоненциал каждой выборки звука, может также использоваться как обобщённая экспоненциальная функция

```
(if (> g0 g1)                          ; g0 = g1
        (pwlv 1 1 0)                   ; затухание
        (pwlv 0 1 1))))                ; увеличение плавного нарастания
```

Код 10

```
;;; raised cosine curve
(defun raised-cosin (phase)
  (let ((hz (hz-to-step (/ (get-duration 2)))))
    (mult 0.5
      (sum 1
        (osc hz 1 *sine-table* phase)))))

;;; log or exponential curve scaled 0 to 1
;;; x is the minimum level in dB before scaling
(defun log-exp-curve (x direction)
  (control-srate-abs *sound-srate*
    (let ((x (db-to-linear x)))
      ;; If direction=0 fade-in else fade-out
      (if (= direction 0)
        (setf env (pwev x 1 1))
        (setf env (pwev 1 1 x)))
      (mult (/ (- 1 x))     ; normalize to 0 dB
        (diff env x)))))    ; drop down to silence
```

```
control-srate-abs *sound-srate*
```

```
(if (= direction 0)     /     (x (db-to-linear x)
```

плавное нарастание / постепенное затухание (setf env (pwev x 1 1)) → (setf env (pwev 1 1 x)))

Код 11

(s-exp)[1] [12]. Snd-log — вычисляет натуральный логарифм каждой выборки звука. Фрагмент кода ниже демонстрирует масштабирование кривых к минутам. Hz-to-step — регулирует шаг частоты в соответствии с усилением фазы > g0 g1 / 90 0, а также контролирует синусоидальные колебания. *Sine-table* является глобальной переменной и предназначена для использования осцилляторами (osc) поиска по частотной таблице (код 6).

Шкала кривых (1), указывающая на минимальные и максимальные значения подразумевает использование операторов g[2] (код 7).

Точками указаны пределы уровня громкости в зависимости от шага фазы огибающей. Работа с косинусоидальными кривыми (2) происходит также с оператором g (код 8).

Пределы уровня громкости здесь также могут зависеть от огибающей, частоты шага и продолжительности периода затухания.

Фрагмент кода, расположившийся ниже, демонстрирует плавное нарастание аудиосигнала на входе с последующим его выводом (код 9).

Здесь control-srate-abs относится к функциям изменения среды Nyquist, которая отмечается другими высокоуровневыми функциями, обычно такие изменения происходят относительно текущей среды. Подробнее об изменении среды, функциях преобразования и переменных окружения можно ознакомиться в пуле инструкций по языку программирования Nyquist разных версий [13,14]. Имеются также и абсолютные версии каждой функции преобразования за исключением seq, seqrep, sim, simrep. Абсолютные версии при запуске или окончании abs не просматривают текущую среду, но устанавливают текущую переменную окружения в определённое значение. Тем самым разделы программного кода могут быть изолированы от внешних преобразований. *Sound-srate* — является частью сре-

[1] Прим. автора. Может использовать возврат многоканального звука к каждому элементу через привязку sound.

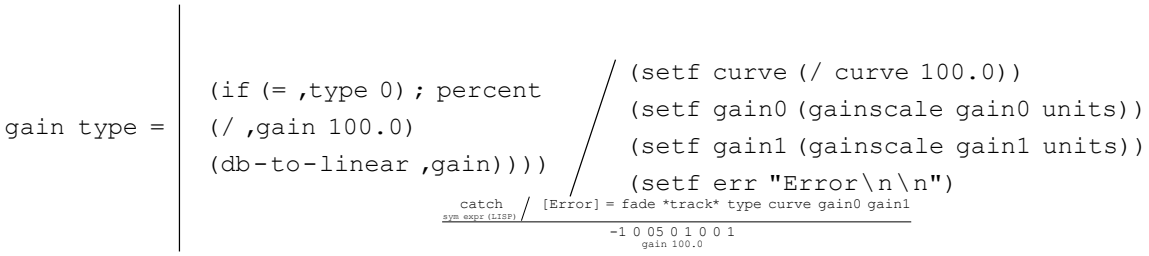
[2] Прим. автора. Уровень громкости в соответствии с приближениями, например t1, t2, t4, l1, l2, l3, [dur]. Или g…|0| …/ g0 ↔ g1.

```
;;; curve scaling for S-curve
(defun exp-scale-mid (x)
  (let ((e (exp 1.0)))
    (/ (- (exp (- 1 x)) e)
      (- 1 e))))

(defmacro gainscale (gain type)
  `(setf ,gain
      (if (= ,type 0)  ; percent
          (/ ,gain 100.0)
          (db-to-linear ,gain))))


(setf curve (/ curve 100.0))
(setf gain0 (gainscale gain0 units))
(setf gain1 (gainscale gain1 units))
(setf err "Error\n\n")


(catch 'err (fade *track* type curve gain0 gain1))
```



Код 12

ды и устанавливает уровень частоты дискретизации аудиофрагмента. Pwlv — создаёт кусочно-линейный конверт с контрольными точками (0, l_1), (t_2, l_2), который заканчивается значениями t_n, l_n. В другом случае поведение выглядит схоже с pwl[1](код 10).

Дальше идёт процедура, которая повышает косинусные затухания. Setq — устанавливает значения символа по отношению к кривой. Sum — класс суммы формирует сумму чисел, по одной от каждого из двух других образцов. Первый параметр может быть образцом, второй параметр может быть образцом и числом. Exp[2] — является арифметической функцией, производящей вычисления «средней» и контролирующей её усиления.

- При средней точке — 3дБ
  @ корректировать при 50%
- При средней точке –12 дБ
  @ корректировать при — 50%

Scale — масштабирует амплитуду звука в децибелах, стартовое время и частота дискретизации берутся от аудиоматериала. Abs —абсолютное выражение числа. Программный код, расположившийся чуть ниже, регулирует повышение кривой косинуса.

Связка defun raised-cosin определяет повышение косинуса канальной фазы. Берёт продолжительность частоты шага для оптимального смешивания и умножения многоканальных звуков. Код второго абзаца, разбираемого нами программного фрагмента, указывает на обработку приподнятой косинусообразной кривой и чуть ниже после промежуточного абзаца работа с масштабированием экспоненциальной кривой от 0 до 1, и работа с неизвестными, где x минимальный уровень в Децибелах перед масштабированием (код 11).

После преобразований (–1 x)) звук нормализуется до значений 0дБ, а (diff env x) — звук уменьшается до состояния тишины.

Следующий фрагмент программного кода рассматривает опции кривой, масштабируемой для S-curve.

[1] Прим. автора. Pwl (Piece-wise Linear) — кусочно-линейная функция, задаёт аппроксимированные значения (с использованием контрольных точек) — на единицу времени.

[2] Прим. автора. Exp имеет следующие функциональные процедуры: x-expr — устанавливает число с плавающей запятой, returns — осуществляет возврат выражения к (x) усилению.

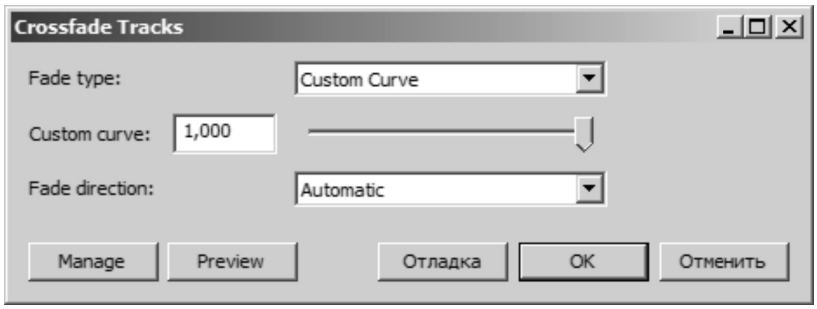


Рис. 10. Интерфейс модуля обработки аудиоданных Crossfade Tracks.

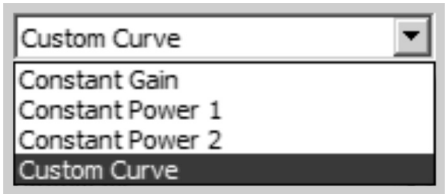


Рис. 11. Опция интерфейса модуля обработки аудиоданных Crossfade Tracks, демонстрирующая типы затуханий.

Здесь defmacro[1] — символьная функция, определяющая макрос и управляющая им. Gainscale — масштабирует громкость в соответствии с её типом [15]. Масштабирование громкости зависит от устанавливаемых процентных значений. Нижняя часть кода корректирует ошибки введённых установочных значений (код 12).

В зависимости от уровня усиления устанавливаются параметры кривых, соответствующие gain0/ gain1. Нижняя ветка, представленная на схеме, обуславливает, что catch является контрольной конструкцией, работающей в паре с «catch sym expr», которая оценивает выражения, а также фиксирует и оказывает воздействие на характер изменения процесса. Sym — идентификация структурных элементов по символу, expr — выражения, подлежащие оценке. При уровне 100.0 текстурирование от –1 до 1.

Предполагаемые ошибки, выдаваемые в процессе отладки программного кода.

- ***Error: unbound function*** (ошибка не связанной функции).
- ***If continued: try evaluating symbol again*** (дополнительная попытка оценки введённого символа).
- ***Error: unbound variable*** (несвязанная переменная).

Второй штатный модуль в Audacity®, отвечающий за плавные перекрёстные переходы — Crossfade Tracks.

Модуль может быть успешно применим при горизонтальном сведении аудиотреков и имеет все шансы по применению в области компьютерного восстановления аудиоматериала. Дорожки должны располагаться так, чтобы затухающая дорожка располагалась над нарастающей дорожкой, а начало нижней дорожки располагалось перед концом верхней дорожки, чтобы создать перекрытие между ними. Выберите область перекрытия в обеих дорожках (см. *пример ниже*) и примените эффект. Модуль поддерживает следующие типы затуханий:

Constant Gain (постоянное усиление) — универсальный тип затухания (по умолчанию коррелируется с направлением затухания), используется при горизонтальном и вертикальном сведении, премастеринге (на стадии уплотнения в начале и конце композиции) и реставрации аудиоматериала. Данный тип затухания эмитирует обычный эффект Fade In / Fade Out. Затухает в верхней дорожке и нарастает в нижней дорожке с линейными затуханиями. Постоянное усиление предполагает, что до тех пор, пока исходный звук не будет

[1] Defmacro работает в паре с некоторыми подэлементами типа: sym*, fargs*, expr*, returns*.
- Sym — символ, который определяется (заключается в кавычки).
- Fargs — формализованный список параметров, список лямбды (заключается в кавычки).
- Expr — выражения составляющие тело функции (заключается в кавычки).
- Returns — операции возврата функционального символа.

Рис. 12. Опция интерфейса модуля обработки аудиоданных Crossfade Tracks «Custom curve».

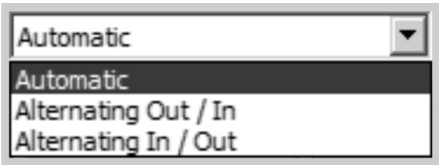


Рис. 13. Опция интерфейса модуля обработки аудиоданных Crossfade Tracks, демонстрирующая раскрывающий список видов затуханий.

обрезан, перекрестное соединение (склейка стека дорожек), по умолчанию, также не будет обрезана. В целом многократное применение перекрёстных эффектов перехода при постоянном увеличении и снижении локальной спектро-частотной амплитуды могут приводить к снижению общего объёма затухания.

Constant Power 1 (постоянная мощность затухания) противостоит процессу падения объёма, в использовании предпочтительна на стадии прямого горизонтального сведения, однако в некоторых случаях при использовании этого типа перекрёстного соединения наблюдается повышение пикового уровня в течение затухания. Поэтому необходимо внимательно следить за общим уровнем склеиваемых фрагментов аудиоматериала, желательно чтобы они имели равную амплитуду в начале и в конце сортируемого фрагмента. Для начинающих звукооператоров данный тип затухания может быть наиболее предпочтительным в силу многоаспектности его использования.

Constant Power 2 (постоянная мощность затухания, версия 2) — разновидность эффекта постоянного затухания опционально предназначена для поддержания достаточного постоянного объёма в течение всего процесса затухания. Эта вариация эффекта постоянной мощности затухания предназначена для более быстрого отклика затухания по сравнению с «Constant Power 1». При применении данной опции перекрёстное соединение может звучать более резко в первый или последний моменты затухания, поэтому если композиция имеет ярко выраженные ударные инструменты, то при данной опции они будут звучать более чётко. Алгоритм второй версии «Constant Power» базируется на предшествующих версиях Audacity®, использовавших модули «Cross Fade In» и «Cross Fade Out». Следующим типом затухания модуля «Crossfade Tracks» является «**Custom Curve**» (пользовательская кривая). В принципе предустановленные модульные алгоритмы затухания хорошо подходят для большинства задач, связанных с переходными затуханиями, но когда требуется точный контроль над применяемым эффектом, то будет полезна функция использования пользовательской кривой. Алгоритм включает регуляторы управления пользовательской кривой, которая позволяет в ручном режиме регулировать скорость пересечения дорожек. Основными элементами управления алгоритмом пользовательской кривой являются текстовое поле ввода числовых данных[1] и ручной регулятор прокрутки числовых данных от 0,000 до 1,000, позволяющий управлять «кривизной» затухания таким же образом, как и «кривой» на микшерном пульте.

При установленном нулевом значении перекрёстное соединение фактически совпадает с предустановкой Constant Gain. Высокие цифры будут полезны, если стоит задача, чтобы обе дорожки были чётко слышимы во время процедуры перекрёстной стыковки, на максимуме 1.0 верхняя дорожка будет поддерживать близкий к полному объём до расстояния близкого к концу перекрёстного соединения, а затем быстро исчезнет. В то же время нижняя дорожка будет очень быстро затухать и дойдёт почти до уровня полного объёма (по затуханию).

Общий объём, вероятно, будет существенно выше в период перекрестного соединения аудиофрагментов. При установке в промежуточное значение 0,5 затухание будет представлять кривую «постоянной мощности» и будет поддерживать достаточно устойчивый объём на протяжении всего перекрёстного соединения. Модулем Crossfade Tracks также предусмотрен режим «Fade Direction», определяющий направление затухания. Режим имеет несколько опций

[1] Прим. автора. Обратите внимание, что данная опция («Custom Curve») работает только в паре с («Custom Curve» — «Fade Type»).

```
(defun crossfade (type dir curve)
  (setf fade-out
    (case dir
      (0 (equal (guessdirection) 'OUT))   ; auto
      (1 (oddp (get '*track* 'index)))    ; fade out odd
      (T (evenp (get '*track* 'index))))) ; fade out even
  (mult *track*
    (cond
      (fade-out
        (case type
          (0 (pwlv 1 1 0))
          (1 (osc (hz-to-step (/ (get-duration 4))) 1 *sine-table* 90))
          (2 (s-sqrt (pwlv 1 1 0)))
          (T (custom curve 0))))
      (T  ; else fade in.
        ; Control envelope sample rate must match sound so that lengths
        ; match exactly, otherwise we get a click at the end of the fade.
        (setf *control-srate* *sound-srate*)
        (case type
          (0 (pwlv 0 1 1))
          (1 (osc (hz-to-step (/ (get-duration 4))) 1))
          (2 (s-sqrt (pwlv 0 1 1)))
          (T (custom curve 1)))))))
```

Код 13

Перечисленные опции позволяют гибко регулировать нарастание и спад аудиосигнала. Automatic (автоматическое затухание) — режим автоматически определяет требуется ли нарастание или затухание. В тех случаях, когда необходимо отменить операцию можно использовать сочетание клавиш (CTRL +Z). Автоматическая настройка затухания основывается на близости фрагментов выделения (аудиоклипов) к концам выбранных фрагментов (аудиоклипов). В том случае, если начало *выделения* находится ближе к началу выбранного аудиофрагмента чем конец *выделения* — к концу аудиофрагмента, то эффект создаёт постепенное нарастание. И ровно наоборот, если конец *выделения* находится в конце *выделения* или рядом с ним, а начало *выделения* не находится в начале аудиофрагмента или рядом с ним, в этом случае происходит постепенное затухание.

В некоторых редких (маловероятных) случаях, когда оба конца выделения находятся на одинаковом расстоянии от концов выбранного аудиофрагмента, эффект исчезает. Alternating Out / In (Вариант Выход/Вход). Выделенная область первой выделенной дорожки будет исчезать. Когда выбрано несколько дорожек, то выделенная область в следующей выбранной дорожке будет нарастать. Если же выбрано более двух дорожек, то направление затухания будет продолжать чередоваться как затухание / нарастание / затухание. Alternating In / Out (Вариант Вход/Выход). Выделенная область первой выделенной дорожки будет нарастать. Когда выбрано более одной дорожки, то выделенная область в следующей выбранной дорожке будет исчезать. В том случае если выбрано более двух дорожек, то направление затухания будет продолжать чередоваться нарастание / затухание / нарастание. Если взглянуть на программную архитектуру модуля, разработанную так же как и в предыдущем случае Стивом Далтоном[1] (Steve Daulton), то можно лицезреть весь арсенал программных действий с возможностями корректировки. Программно модуль может быть скорректирован с помощью следующего кода (код 13):

Код символьно демонстрирует процессы перекрёстного горизонтального перехода между обрабатываемыми аудиодорожками. Здесь pwlv — также функционирует как стандартные кусочно-линейные приближения. Символ «v» — создаёт кусочно-линейный конверт (подобно огибающей аудиосигнала) с приблизительными контрольными точками 0 → 1 → 1. Evenp является функцией предиката, определяет, является ли переменное значение целым числом. Может идти в логической цепочке evenp expr/returns. Expr — обеспечивает проверку целого числа. Oddp — функция предиката, в данном случае производит проверку це-

[1] Прим. автора. Модуль Crossfade Tracks спроектирован Стивом Далтоном в ноябре 2014-го года и регенерировался по сентябрь 2015-го года. Модуль также может быть изменён (перекомпилирован) согласно лицензии GNU General Public License, version 2.

```
                  ┌                                               ┐
                  │     (0 (equal (guessdirection) 'OUT))         │
case dir =        │     (1 (oddp (get '*track* 'index)))          │
                  │          ...1...                              │
                  │     (T (evenp (get '*track* 'index)))))       │
                  └          ...0...                              ┘
   osc_hz-to-step    _ / _      get-duration       4_1 *sine-table* 90
   управление/частота шага           ↓
                            после получения продолжительности
```

Код 14

```
              ┌(0 (pwlv 1 1 0))                                                   ┐
              │(1 (osc (hz-to-step (/ (get-duration 4))) 1 *sine-table* 90))      │
case type =   │(2 (s-sqrt (pwlv 1 1 0)))                                          │
  fade-out    │              pwlv 0 1 1 ↔ 110                                     │
              └(T (custom curve 0))))                                             ┘
```

Код 15

```
                    ┌                                                        ┐
                    │      pwlv 1 1 0                                        │
                    │(0 (pwlv 0 1 1))                                        │
case   type      =  │(1 (osc (hz-to-step (/ (get-duration 4))) 1)            │
  fade-out / fade in│(2 (s-sqrt (pwlv 0 1 1) / (2 (s-sqrt (pwlv 1 1 0)      │
                    └(T (custom curve 1)    /  (T (custom curve 0)           ┘
          (setf *control-srate* *sound-srate*)
```

Код 16

лого нечётного числа. Evenp — также функция предиката, проверяет, является ли число целым (код 14).

Далее case рассматривается как контрольная конструкция в конечном итоге, способная поддерживать плавное нарастание в процессе затухания[1], используя пользовательскую кривую (код 15).

Уровень частоты дискретизации должен соответствовать всей продолжительности трека, в противном случае мы получаем щелчок в конце затухания (код 16).

[1] Прим. автора. Плавное нарастание аудиосигнала в процессе затухания психоакустически создаёт эффект растяжки спадающей (рассеивающей) амплитуды аудиосигнала за счёт периодического уплотнения фрагментов спектра спадающей амплитуды. Эффект может подчёркивать отдельные участки сведённого аудиоматериала, к примеру немного возвышать вокал при общем затухании аудиосигнала либо акцентировать внимание на различных музыкальных инструментах.

Control-srate является элементом среды, устанавливает частоту дискретизации управляющих сигналов. По факту нет никаких различий между управляющим сигналом и аудиосигналом. Можно производить считывание управляющего сигнала *control-srate* напрямую, но использовать control-srate-abs, чтобы изменять параметры дискретизации аудиоматериала. *Sound-srate* — частота дискретизации музыкальных звуков, считывание музыкальных звуков производится напрямую, sound-srate-abs позволяет изменять параметры дискретизации музыкальных звуков.

Фрагмент процедуры постепенного затухания аудиосигнала согласно нечётному числу. Точками обозначены уровни шага (прохода), обеспечивающего крутизну спада амплитуды аудиоматериала. Формулировки, представленные вне скобок, схематично показывают взаимодействие с продолжительностью и единственной синусоидой цикла, предназначенной для исполь-

```
(defun custom (curve inout)
  ;; 'epsilon' defines the curvature of a logarithmc curve.
  ;; To avoid log 0 or /0 it must be > 0 and < 1.
  (let* ((curve (+ 0.99 (* -0.98 (min 1 (max 0 curve)))))
         ; magic number 2.7 gives approx 'constant power' curve at 50% setting.
         (epsilon (power curve 2.7)))
    (if (= inout 0)
        (setf logcurve (pwev epsilon 1 1))
        (setf logcurve (pwev 1 1 epsilon)))
    ; Scale and invert curve for 0 to unity gain.
    (sum 1
         (mult (/ -1 (- 1 epsilon))
               (diff logcurve epsilon)))))
```

Код 17

```
(defun guessdirection ()
"If the selection is closer to the start of the
audio clip, fade in, otherwise fade out."
  (let* ((start (get '*selection* 'start))
         (end (get '*selection* 'end))
         (clips (get '*track* 'clips))
         (in-dist end)
         (out-dist end))
    (if (arrayp clips)
        (setf clips (append (aref clips 0)(aref clips 1))))
    (dotimes (i (length clips))
      (setf in-dist (min in-dist (abs (- start (first (nth i clips))))))
      (setf out-dist (min out-dist (abs (- end (second (nth i clips)))))))
    (if (< in-dist out-dist) 'in 'out)))

(crossfade type direction curve)
```

Код 18

зования осциллятором поиска по таблице (osc), *sine-table*[1] — является глобальной переменной (код 17).

Здесь let является контрольной конструкцией, которая устанавливает цепочку последовательностей переменных числовых значений. Let допускает последовательность значений binding, привязки переменных, каждая из которых является либо:

- Символом (который размечается до нуля)
- Список содержимого, который является символом и чей CADR[2] является выражением инициализации.

Expr — выражение, подлежащее вычислению.
Returns — возврат значения последнего выражения.

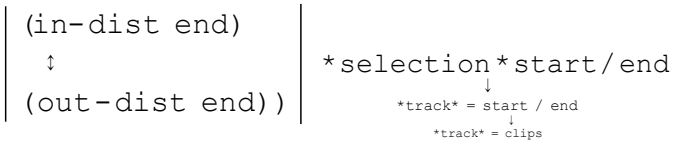
If (в режиме Lisp) оценивает выражение условно. В некоторых случаях для оценки выражений используются следующие параметры if[3] texpr, expr1 [expr2]. Texpr — тестовое выражение, expr1— выражение, которое подлежит вычислению, при условии, что texpr не равен нулю (nil). Expr2 — выражение, которое подлежит вычислению, при условии, что texpr равен нулю (nil). Параметр returns используется как значение уже выбранного выражения (код 18).

[1] Прим. автора. Сама же таблица, используемая оператором osc и другими осцилляторами *table*, первоначально рассматривается как набор к синусоиде *sine-table*.

[2] Прим. автора. CADR [произносится как «kae-der», сокращённые варианты использования в программировании CAR и CDR] — в структуре Lisp/Common Lisp можно использовать как (second), например (second expr), здесь second выступает альтернативным именем CADR. Исторические традиции Lisp-программирования довольно богаты, поскольку данный язык разрабатывался для сложно укомплектованных программных систем и компьютерных систем искусственного интеллекта, постоянно совершенствовалась его компонентная база, менялись диалекты, конструкции. И в итоге оригинальная классификация объектов была сокращена в целях упрощения компоновки кода и повышения его мобильности.

[3] Прим. автора. Обратите внимание, что синтаксис условного SAL-выражения записывается как #? Пример test, iftrue-expression, iffalse-expression, но #if может быть использован вместо #?. Любая форма может игнорировать третий аргумент, который по умолчанию равен нулю (nil), т.е. пустому списку.

```
(let* ((start (get '*selection* 'start))
       (end (get '*selection* 'end))
       (clips (get '*track* 'clips))
```



Код 19

```
(defun find-ends (T0 T1 clips)
  (let ((trk-ends ())    ;starts of clips
        (trk-starts ())) ;ends of clips
    (dolist (clip clips)
      ;; look for clip enclosing the selection.
      (when (and (>= (second clip) T1) (<= (first clip) T0))
        (psetq trk-ends   (list (/ (+ T0 T1) 2))
               trk-starts (list (/ (+ T0 T1) 2)))
        (return))
      (when (and (> (first clip) T0) (< (first clip) T1))
        (push (first clip) trk-starts))
      (when (and (> (second clip) T0) (< (second clip) T1))
        (push (second clip) trk-ends))
      (when (> (first clip) T1) (return)))
    (cond
      ((and (= (length trk-ends) 1)
            (= (length trk-starts) 1)
            (<= (car trk-ends) (car trk-starts)))
        (list (car trk-ends)(car trk-starts)))
      ((or (> (length trk-ends) 1)
           (> (length trk-starts) 1))
        (throw 'error err1))
      (T (throw 'error err2)))))
```

Код 20

Defun guessdirection () предугадывает направление. К примеру, если имеется выбор ближе к запуску, плотность амплитуды аудиофрагмента постепенно увеличивается, а в противном случае уменьшается. Сама же defun служит символьной функцией, которая предопределяет функцию (код 19).

Операторы[1] после выбранного пользователем направления пересечения аудиофрагмента определяют положение в начале фрагмента или в его конце. Либо берут весь аудиофрагмент и применяют к нему степени пересечения.

Третий модуль, обеспечивающий затухание в Audacity®, это Crossfade Clips (соединение перекрестных плавных переходов). Стыковка перекрёстных переходов призвана обеспечить простую стыковку выбранной паре аудиофрагментов на одной звуковой дорожке. Алгоритм не имеет настраиваемых параметров и использует оболочку программы для просчёта операции склеивания фрагментов. Для того чтобы воспользоваться этим эффектом, необходимо

[1] Прим. автора. Подробнее об использовании наиболее частых операторов для формирования выражений и для компиляции кода в целом можно ознакомиться в конце данной статьи в разделе приложения «Единичные и двоичные операторы для формирования выражений». Для использования математических функций, смотрите также приложение 2 — «Математические функции языка программирования Lisp/Niquist и их эквиваленты на языке программирования SAL».

```
(defun crossfade (sig out-end in-start end)
"Do the crossfade"
  (abs-env
    (control-srate-abs *sound-srate*
      (let* ((fade-out (mult sig (env out-end 0)))
             (cflen (max out-end (- end in-start))) ;crossfade length
             (finstart (max (- out-end (- end in-start)) 0))
             (fade-in (mult (extract (- end cflen) end sig)
                            (env (- cflen finstart) 1 finstart))))
        (sim fade-out fade-in)))))

(defun env (dur direction &optional (offset 0))
"Generate envelope for crossfade"
  (abs-env
    (if (< dur 0.01)            ;make it linear
        (control-srate-abs *sound-srate*
          (if (= direction 0)
              (pwlv 1 dur 0)      ;fade out
              (pwlv 0 offset 0 (+ offset dur) 1)))  ;fade in
        (if (= direction 0)     ;cosine curve
            (cos-curve dur 0)
            (seq (s-rest offset)
                 (cos-curve dur 1))))))
```

Код 21

поместить два аудиофрагмента на одну дорожку. Затем выбрать приблизительно одинаковое количество аудиоданных из конца одного фрагмента и начала другого фрагмента. После применения эффекта выбранные области будут соединены[1]. В том случае если выбранная область является непрерывным звуком без разрывов, то первая и последняя половины выбранного звука будут соединены[2]. Crossfade Clips может быть скорректирован с помощью кода приведённого ниже (код 20).

Здесь defun — символьная функция, вызывающая определение функции defun sym fargs expr…, может выполняться также для определения макроса (defmacro sym fargs expr…). В этих случаях:

1. sym — определяемый символ (в кавычках);
2. fargs — список формальных аргументов (лямбда-список, в кавычках);
3. expr — выражения, составляющие функцию (в кавычках);
4. returns — возврат символа функции.

Dolist — конструкция цикла (регулируется в режиме синтаксиса Lisp). Буквально dolist означает цикл по списку. Используется в контексте со следующими вспомогательными параметрами:

1. sym — символ для привязки к каждому элементу списка;
2. expr — выражение списка;
3. rexpr — конечное выражение (значение по умолчанию ноль nil);
4. expr — составляющая цикла (рассматривается как неявная программа).

When[3] является пользовательской функцией. Исполняется в синтаксисе SAL/Lisp. Обычное предположение if-then (если-тогда) если тест верен, то действие оценивается и возвращается. В противном случае возвращается nil. Также можно использовать if или cond для выполнения предположения (если-тогда-ещё) и более сложных условных форм (код 21).

В данном секторе программного кода, записанного в Nyquist-*файл*, происходит дальнейшая инициализация аудиосигнала посредством последовательности привязок через let* (плавное наложение дорожки) с последующим волновым спадом и выводом аудиосигнала в интерфейс программы. Abs-env является объектом

[1] Прим. автора. Обратите внимание, что аудиоклипы не должны касаться друг друга. Любое пустое пространство между клипами игнорируется.

[2] Прим. автора. Разрыв в форме сигнала, подобный большому щелчку, может быть сглажен путем применения короткого соединения поперёк сбоя.

[3] Прим. автора. when может также являться и контрольной конструкцией, с немного иным набором параметров:

1) texpr — тестовое выражение
2) expr — выражение, подлежащее вычислению, если expr не равен nil
3) returns — возврат значения последнего выражения или nil

```
 Ny  =  ...(at 10.0 (abs-env (my-beh))) →
Nyquist PL
что эквиалентно (abs-env (my-beh)) →
→ потому как   abs-env
                  вызывает
                     ↓
          среда по умолчанию
Sal = abs-env(my-beh()) @ 10 →
SAL PL
эквивалентно → abs-env(my-beh()).
```

Код 22

```
(defun cos-curve (dur direction)
"Generate cosine curve"
  (if (= direction 0) ;fade out
      (osc (hz-to-step (/ 0.25 dur)) dur *sine-table* 90)
      (osc (hz-to-step (/ 0.25 dur)) dur *sine-table* 0)))

(defun process (sig t0 t1 clips)
"Find the split positions and crossfade"
  (setf fadeclips
    (multichan-expand #'find-ends t0 t1 clips))
  (if (arrayp fadeclips)
      (prog ((fade-out-end (min (first (aref fadeclips 0))
                                (first (aref fadeclips 1))))
             (fade-in-start (max (second (aref fadeclips 0))
                                 (second (aref fadeclips 1)))))
        (return
          (multichan-expand #'crossfade sig
                                      (- fade-out-end t0)
                                      (- fade-in-start t0)
                                      (- t1 t0))))
      (crossfade sig
                 (- (first fadeclips) t0)
                 (- (second fadeclips) t0)
                 (- t1 t0))))

(if (= (length (get '*selection* 'tracks)) 1)
    (catch 'error
      (process *track*
               (get '*selection* 'start)
               (get '*selection* 'end)
               (get '*track* 'clips)))
```

Код 23

преобразования последующей конструкции и вычисляет поведение в среде по умолчанию. Играет не последнюю роль в вычислении таблиц волновых форм и различных сигналов, которые находятся во «вне» границ времени. В качестве примера рассмотрим одно из поведений (код 22).

В примере программного кода на SAL введён оператор @, регулирующий временной сдвиг, предопределяющий дальнейшее поведение. Control-srate-abs является преобразованием и оценивает поведение *control-srate*[1] с условной частотой дискретизации srate. Следующий блок программного кода демонстри-

[1] Прим. автора. Является частью среды и устанавливает уровень контроля над частотой дискретизации. Как элемент среды обеспечивает устойчивый уровень частоты дискретизации (по умолчанию) для управляющих сигналов. Управляющий сигнал эквивалентен аудиосигналу, однако в некоторых случаях он служит заглавной меткой, предопределяющей поведения.

```
(crossfade sig
        (- (first fadeclips) t0)
        (- (second fadeclips) t0)
        (- t1 t0))))
                                    (multichan-expand #'crossfade sig
                                    (- fade-out-end t0)
                                    (- fade-in-start t0)
                                    (- t1 t0))))
        (- (first fadeclips) t0)
        (- (second fadeclips) t0)
        (- t1 t0))))
                                    (multichan-expand #'crossfade sig
                                    (- fade-in-end t0)
                                    (- fade-out-start t0)
                                    (- t1 t0)))) / t0 t1
```

Код 24

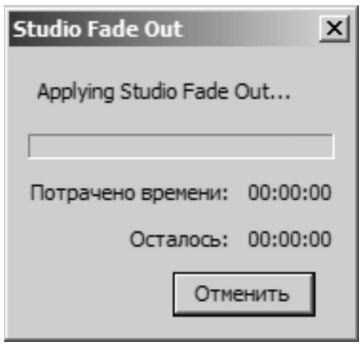


Рис. 14. Модуль обработки аудиоданных Studio Fade Out.

рует поиск и разделение позиций при кроссфейдинге[1] (код 23).

Здесь defun — также символьная функция, определяющая вычисление косинуса кривой, соответственно на Lisp зарезервированы следующие вспомогательные операторы:

1. Expr — число с плавающей запятой.
2. Returns — косинус числа (возврат косинуса числа).

Osc — является осциллятором, который возвращает звуки таблицы, изменяющиеся при балансе продолжительности и воспроизводимые с фазы в градусах (код 24).

Как мы видим перекрёстное плавное снижение и повышение уровня громкости аудиосигнала распределяется в соответствии с t^2, = t0-t1 устанавливает время возврата выборки звука. Первый фрагмент затухания 0, второй фрагмент затухания 0. Expand — устанавливает многоканальное расширение, расширяет память, добавляя сегменты. Может использоваться в паре с num, определяющей число сегментов для добавления. Оператор returns может быть использован для идентификации числа добавленных сегментов.

Четвёртый штатный модуль в Audacity®, имеющий возможность имитировать плавные затухания — «Studio Fade Out». Studio Fade Out (Студийное затухание) эффект фиксированного студийного затухания.

Предполагает плавное затухание аудиосигнала подобное тому, которое используется в конце композиций коммерческих аудиозаписей.

Алгоритм модуля эмулирует фиксированный спад аудиосигнала, предусмотренный в студийной аппаратной схемотехнике. Формирует эффект музыкального спада от первоначальной громкости до тишины, применяя двукратно изогнутое (S-образное) затухание,

[1] Прим. автора. Кроссфейдинг — перекрёстное плавное снижение амплитуды аудиоматериала для обеспечения мягких переходов между аудиофрагментами, использующееся при компьютерном монтаже аудиозаписей.

[2] Прим. автора. Осуществляет возврат первой выборки звука. Операторы Nyquist, которые добавляют дубликаты фрагментов звука (создают их копии), позволяют смещать дубликаты до половины демонстрационного периода в любой направленности для выравнивания выборки двух операндов.

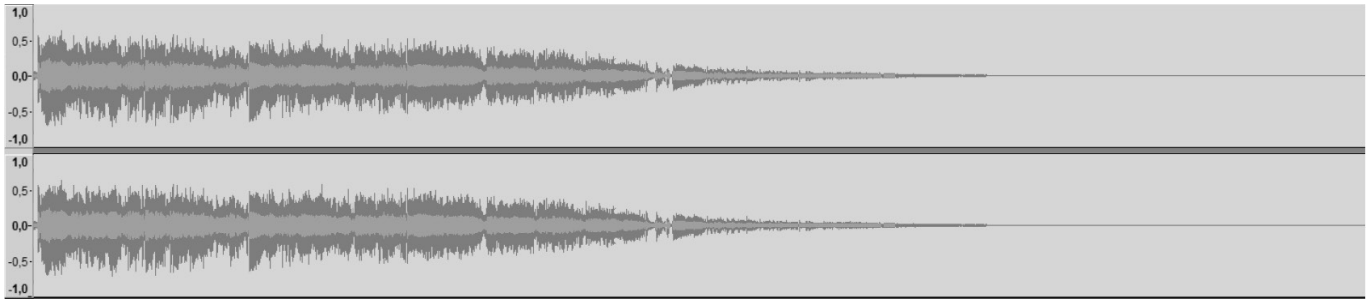


Рис. 15. Отображение плавного студийного затухания на сонограмме.

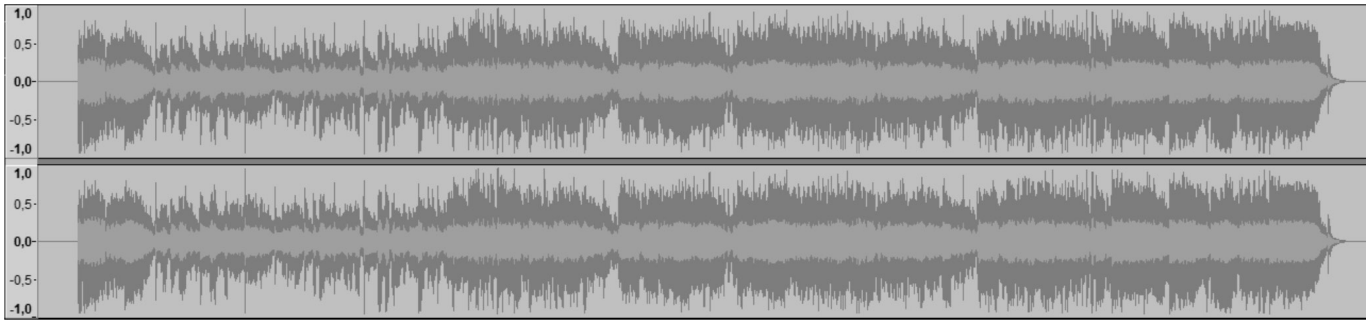


Рис. 16. Стерео-трек, визуализированный с помощью программы Audacity® с параметрами стандартной для человеческого восприятия частоты дискретизации 44 100 Гц.

и затухание более высоких частот происходит немного быстрее чем более низких частот. Такой тип спада частотной амплитуды придаёт аудиосигналу эффект «уходящего вдаль» и не звучит как механическое затухание.

Данная технология хорошо проявила себя и прочно закрепилась в студиях звукозаписи для LP и CD-треков. Алгоритм Studio Fade Out не имеет управляемого модульного интерфейса и работает в режиме внутренней обработки сигнала. Поэтому регулировка степени спада производится простым инструментом выделения аудиофрагмента (F1) внутри программы Audacity®. Для того чтобы сделать простую настройку аудиоформы любого из вышеперечисленных затуханий, повторив его с помощью RLE (Repeat Last Effect — CTRL + R).

Если повторить нарастание, звук будет оставаться тихим дольше, а затем быстрее восстанавливаться до исходного уровня. Повторение затухания приведёт к тому, что уровень сначала упадёт быстрее, а затем остановится на более низком уровне. Программный код данного акустического эффекта был разработан Стивом Далтоном (Steve Daulton) и подразумевает стандартный фиксированный вариант плавного затухания. Редакция кода может производиться в соответствии с лицензией GNU GPL v.2, и в определённых обстоятельствах (сведение, реставрация аудиоматериала) может быть очень полезным занятием. Гибкость программных сценариев Nyquist позволяет с высокой степенью точности производить нелинейное внутрипрограммное (Audacity®) редактирование аудиоматериала и адаптировать фрагменты кода под конкретное решение утилитарных задач. Алгоритм универсален и подходит для решения следующих мультифункциональных задач:

- Сведение аудиоматериала (в том числе сведение противофазы) с пошаговым снижением спектро-частотной амплитуды.
- Вертикальное и горизонтальное сведение аудиотреков.
- Регулируемые спады спектро-частотной амплитуды для кратковременного снижения внутрифазной амплитуды межфрагментарных акустических блоков.

К примеру, возьмём обычный стерео-трек с постоянной частотой дискретизации 44 100 Гц. и визуализируем его средствами сонограммы в виде (обычной) звуковой формы.

Сонограмма показывает нам, что трек имеет обычную форму с резким началом и мягким спадом амплитуды. Для того чтобы оптимизировать спад амплитуды аудиосигнала[1] в конце композиции, необходимо вос-

[1] Прим. автора. По умолчанию в Audacity® штатный модуль Studio Fade Out выполняет процедуры спада спектро-частотной амплитуды аудиосигнала автоматически. Т.е. единственным инструментом управления этим модулем в данном случае будет инструмент выделения — [I]. Инструмент выделения будет фиксировать шаг спада спектро-частотной амплитуды.

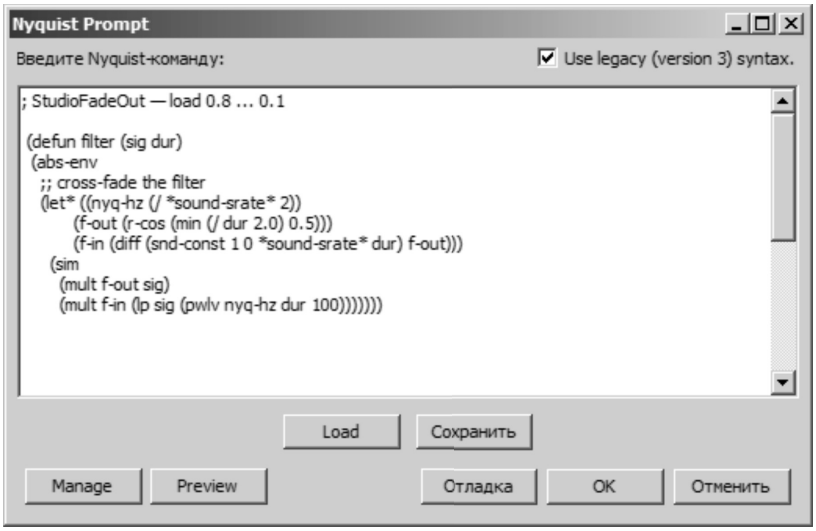


Рис. 17. Модуль NyquistPrompt позволяет вводить сценарии и команды языка программирования Niquist.

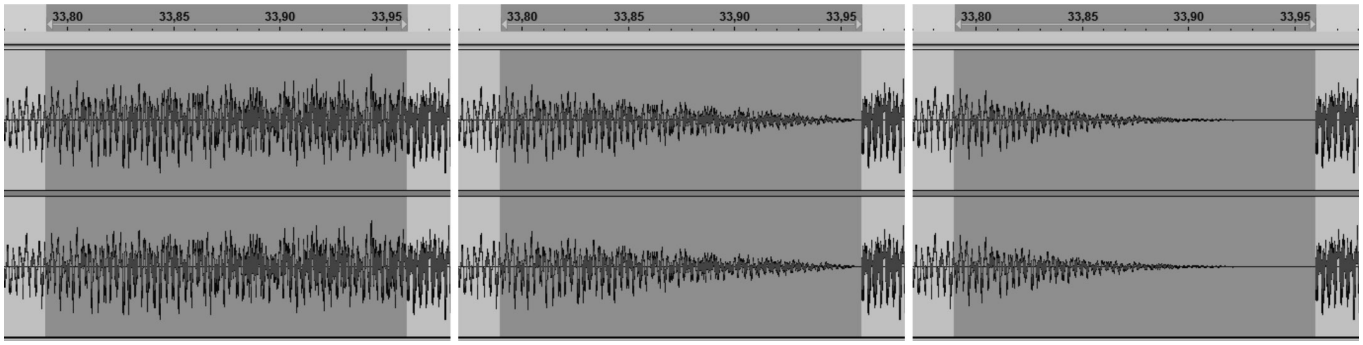


Рис. 18. Отрезки сонограммы, фрагментарно характеризующие плавные спады звука.

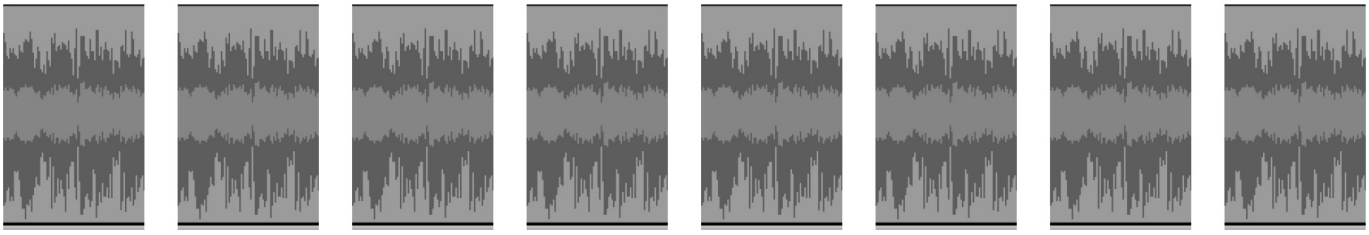


Рис. 19. Фрагментированный аудиоматериал, имеющий одинаковую структуру.

пользоваться вспомогательным *штатным модулем* ввода команд на языке Nyquist (Nyquist Prompt).

По умолчанию модуль настроен на шаговое выделение фрагмента аудиоматериала и подразумевает обращение к интерфейсной команде [I][1] — инструмент выделения.

[1] Прим. автора. Клавиша F1.

Однако в практике компьютерного аудиомонтажа бывают такие моменты, когда необходимо стыковать аудиоматериал, имеющий разную спектро-частотную составляющую, особенно если дело обстоит с аудиоматериалом смешанного содержания (музыкальный ритм, мелодия и т.д.), и в этом случае полагаться на чисто интерфейсное управление уже не представляется достаточным. Поскольку качественная работа с аудиоматериалом дело достаточно трудоёмкое, предлагаем нашим читателям рассмотреть конкретный случай.

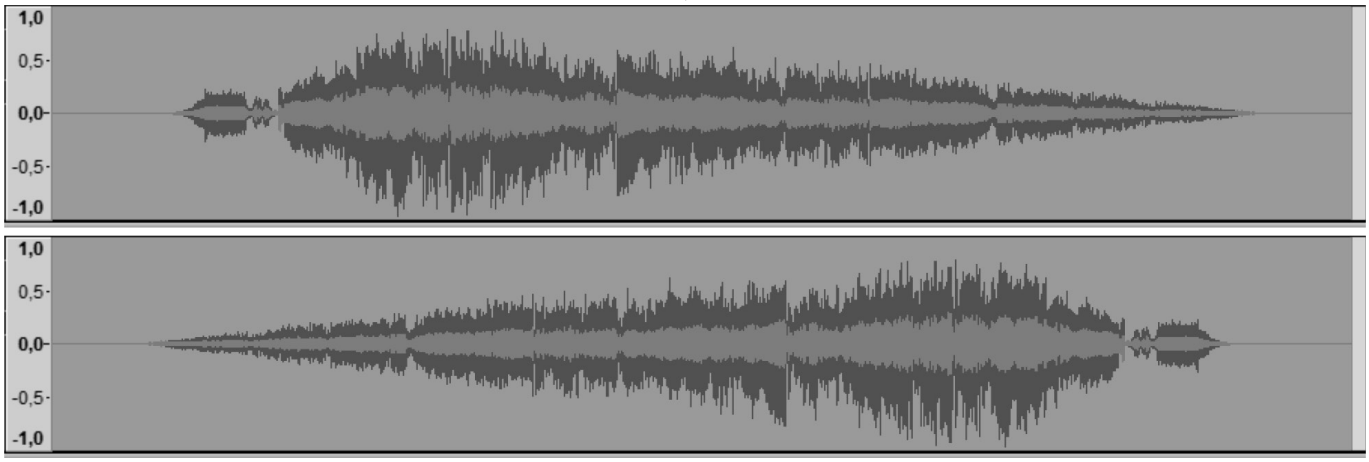


Рис. 20. Подготовленные к сведению аудиофрагменты, имеющие симметричную геометрическую форму спада аудиосигнала.

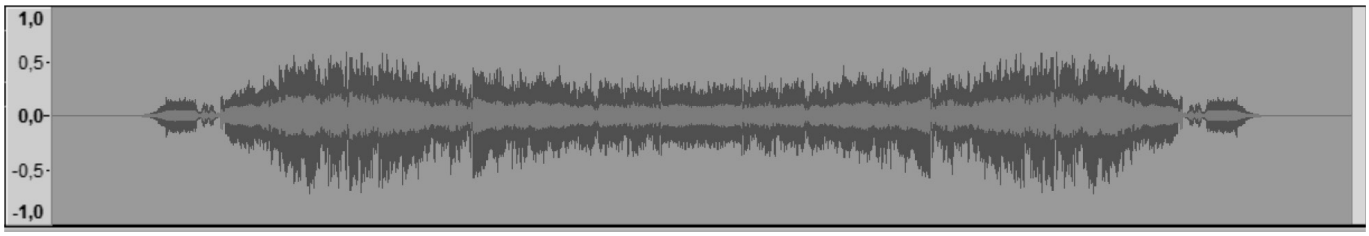


Рис. 21. Сведённый в единую монодорожку аудиофрагмент с симметричным узлом перехода.

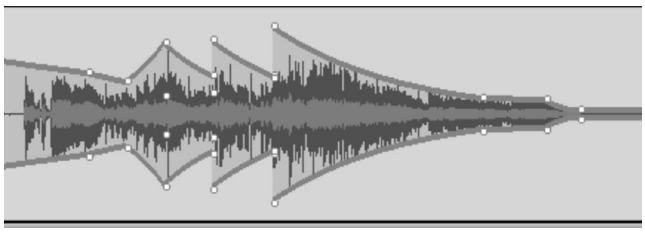


Рис. 22. Узел стыковки с расставленными точками.

Допустим, в нашем распоряжении имеется *два типа* музыкального материала. Первый тип относится к стилю *хип-хоп* (1), а второй (2) — *регги*, входная амплитуда первого аудиоматериала отличается по пиковой громкости от второго, но нам необходимо уровнять их амплитуду в стыковочном месте путём перекрестного затухания аудиосигнала (кроссфейдинга).

Есть два подхода к решению поставленной задачи. Способ первый (интерфейсно-ориентированный) — воспользоваться штатными инструментами выделения аудиосигнала и пошагово[1] снижать степень нагрузки на стековый участок, с начала, с одной стороны (конечный фрагмент аудиотрека), и затем, с другой — (начальная часть второго аудиофрагмента).

В результате мы получаем следующий амплитудный спектр, гармоники которого мягко пересекаются между собой.

Способ второй — (программно-ориентированный, программно-лингвистический) метод доработки кода штатного модуля Audacity® «Studio Fade Out» через «Nyquist Prompt». В принципе второй способ является наиболее точным, с точки зрения расположения узла стыковки аудиоматериала. Узел стыковки аудиоматериала это место соединения двух аудиофрагментов при вертикальном сведении аудиоматериала. От грамотно-

[1] Прим. автора. То есть применять один и тот же эффект (допустим — Fade In в начале трека, Fade Out в его конце) на определённом участке аудиофрагмента несколько раз.

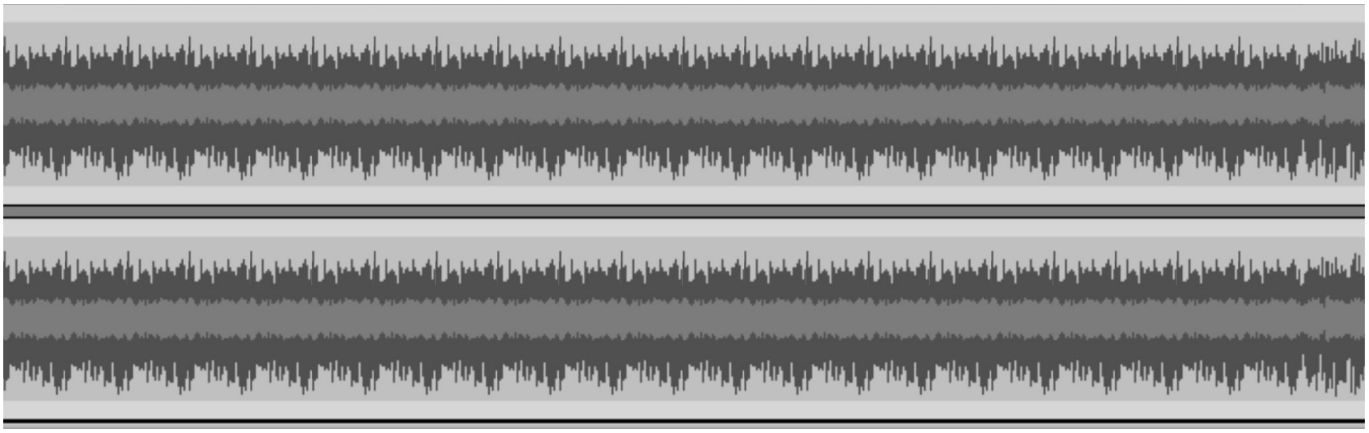


Рис. 23. Пример аудиоматериала с асимметричной формой звучания.

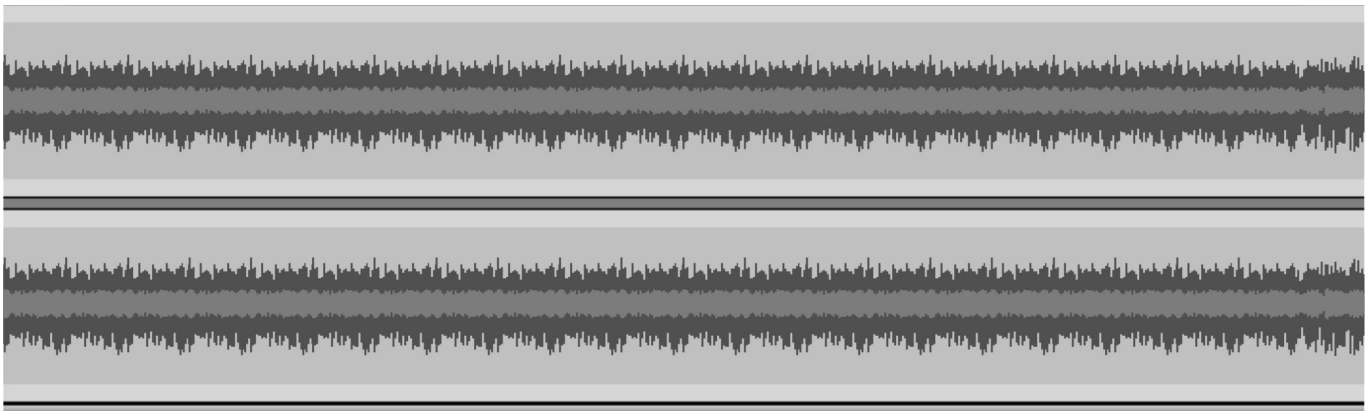


Рис. 24. Пример аудиоматериала с симметричной формой звучания.

го выбора расположения узла стыковки на тайм-линии будет зависеть качество сведённого аудиоматериала.

Возьмём аудиосигнал, имеющий на протяжении всего звучания симметричный ритм, импортируем его в Audacity®. Программа автоматически сформирует сонограмму, которая будет иметь следующий вид

Нам необходимо нормализовать аудиосигнал средствами Nyquist через приглашение NyquistPrompt, для этого мы можем воспользоваться стандартной формулировкой

```
(mult s (env 0.1 0.1 0.2 1.0 0.5 0.3 1.0))
```

На выходе мы получаем более адаптивную форму аудиосигнала

Однако в ходе процесса нормализации аудиосигнала мы не видим контрольных точек его огибающей. Пиковая громкость у всех композиций может сильно варьироваться и конечно, когда мы работаем с массивами аудиодорожек применение данной формулировки вполне оправдано и полезно, но бывают также случаи, когда структура аудиоматериала неоднородна[1]. В этих случаях необходимо манипулировать огибающей средствами контрольных точек. Для наглядности примера возьмём немного изменённый сгенерированный сигнал, отклик которого геометрически симметричен[2].

У аудиосигнала есть пики, которые мы хотим выделить для ясности пиковой громкости либо для подчёркивания фрагментарных акустических элементов[3], присутствующих на протяжении всего аудиоматериала.

[1] Прим. автора. Под неоднородной структурой аудиоматериала подразумевается сложно укомплектованный аудиосигнал, имеющий разный уровень спектрочастотной плотности, соответственно звуковое давление аудиоматерила будет разным.

[2] Обратите внимание на геометрию сонограммы, в первом случае отклик у аудиосигнала имеет асимметричную структуру, а во-втором случае — симметричную. В примере показан стереосигнал, в случае с моносигналом отклик дорожки будет также иметь симметричную поляризацию.

[3] Прим. автора. Фрагментарными акустическими элементами мы именуем отдельные гармоники, которые могут повторяться на протяжении всей акустической композиции. На сонограмме такой тип гармоник может быть выделен вершинами, которые имеют одинаковую либо отличную друг от друга амплитуду громкости.

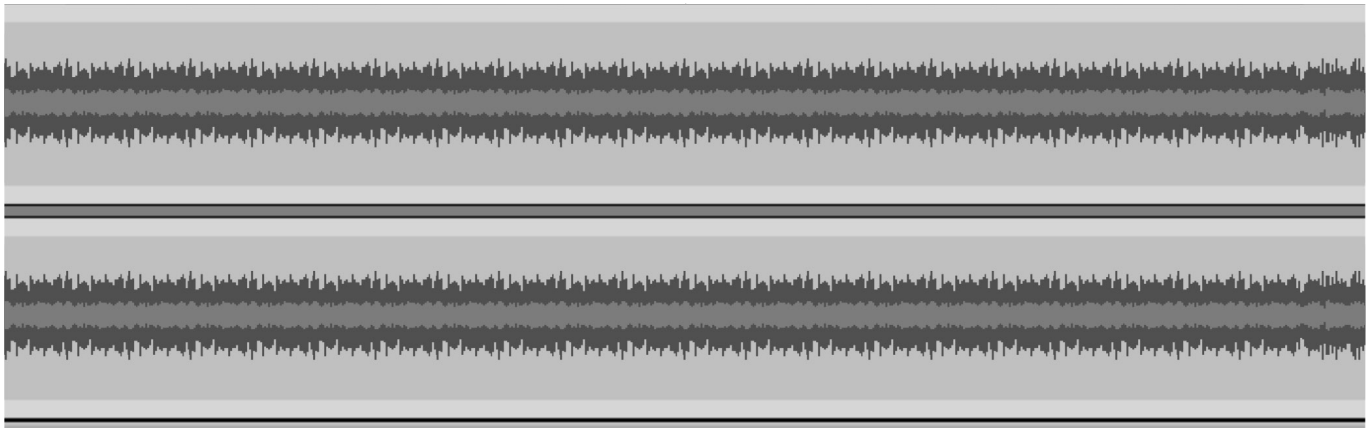


Рис. 25. Нормализованный симметричный аудиосигнал.

```
;nyquist plug-in
;version 4
;type tool
;name "Test Me"
;debugbutton false
;action "Applying TestMe..."
;author "Nemo Nemesis"
;release 2.3.0
;copyright "Public Domain"
;; test-me.ny by Nemo Nemesis, August 2018
(AUD-DO "SelectTime: Start=60 End=170")
(AUD-DO "SelectTracks: Track=1 TrackCount=2")
```

Код 25

Формально язык программирования Nyquist не имеет возможности беспрепятственно получить прямой доступ к огибающим в Audacity®, но данные операции могут быть преобразованы в команды AUD-DO. Поскольку язык программирования Nyquist можно использовать отдельно от аудиоредактора Audacity® и вести самостоятельные разработки в NyquistIDE для более функциональной его интеграции именно в программную среду Audacity®, начиная с версии 2.4.0[1], была внедрена поддержка Макросов. С этого момента AUD-DO стала выполнять функции, существенно расширяющие горизонты возможностей языка программирования Nyquist в Audacity®. AUD-DO как функция даёт возможность использовать широкий перечень макро пишущих команд из языка программирования Nyquist. Скриптовые команды передаются в AUD-DO в виде строк[2] для выполнения функциональной обработки. В качестве примера можно привести заглавный код фрагмента функции AUD-DO (код 25).

Nyquist в виде макроса будет представлен как (type tool) первая метка проекта, и формулируется следующим образом

```
(AUD-DO "SetLabel: Text=\"Foo\"")
```

Текстовая строка, находящаяся после AUD-DO, будет передана механизму выполнения сценариев для обработки сигнала. Одним из самых простых путей в решении вопроса с оптимизацией сценарного кода и сосредоточением его строк является использование команды Macros (Макрос) с целью добавления команд в макроячейки и изучения (отладки) уже добавленных строк.

[1] Прим. автора. Именно с версии Audacity® 2.4.0 стало возможным обеспечивать вызовы команд Nyquist-плагинов, не имеющих интерфейсно-ориентированных опций, макросами Nyquist.

[2] Прим. автора. Примерно такой же подход используется для реализации программных сценариев Audacity® (Python Scripting), команды также создаются в виде строк и передаются для функциональной обработки. Основным преимуществом Nyquist-макросов перед Python Scripting служит то, что Nyquist полностью встроен в оболочку Audacity®.

```
;amplitude-rms
;Envelopes
(/0.1.0.1.0)(/0.1.0.1.0)     /(setf ilowpass 8420.000)
(/0.1.0.1.0)(/0.1.0.1.0)
(/0.1.0.1)
```

Код 26

```
(/ 1616.000 sr)(/  2355.000 sr)(/ 6528.000 sr)
(/ 1516.000 sr)(/  2255.000 sr)(/ 6428.000 sr)
(/ 1416.000 sr)(/  2155.000 sr)(/ 6328.000 sr)
(/ 1316.000 sr)(/  2355.000 sr)(/ 6528.000 sr)
(/ 1216.000 sr)(/  2355.000 sr)(/ 6528.000 sr)
(/ 1116.000 sr)(/  2355.000 sr)(/ 6528.000 sr)
(/ 1616.000 sr)(/  2355.000 sr)(/ 6528.000 sr)
(/ 1516.000 sr)(/  2355.000 sr)(/ 6528.000 sr)
(/ 1416.000 sr)(/  2355.000 sr)(/ 6528.000 sr)
(/ 1316.000 sr)(/  2355.000 sr)(/ 6528.000 sr)
(/ 1216.000 sr)(/  2355.000 sr)(/ 6528.000 sr)
(/ 1116.000 sr)(/  2355.000 sr)(/ 6528.000 sr)
(/ 1616.000 sr)(/  2355.000 sr)(/ 6528.000 sr)
(/ 1516.000 sr)(/  2355.000 sr)(/ 6528.000 sr)
(/ 1416.000 sr)(/  2355.000 sr)(/ 6528.000 sr)
(/ 1316.000 sr)(/  2355.000 sr)(/ 6528.000 sr)
(/ 1216.000 sr)(/  2355.000 sr)(/ 6528.000 sr)
(/ 1116.000 sr)(/  2355.000 sr)(/ 6528.000 sr)
(/ 1616.000 sr)(/  2355.000 sr)(/ 6528.000 sr)
(/ 1516.000 sr)(/  2355.000 sr)(/ 6528.000 sr)
(/ 1416.000 sr)(/  2355.000 sr)(/ 6528.000 sr)
(/ 1316.000 sr)(/  2355.000 sr)(/ 6528.000 sr)
(/ 1216.000 sr)(/  2355.000 sr)(/ 6528.000 sr)
(/ 1116.000 sr)(/  2355.000 sr)(/ 6528.000 sr)
```

Код 27

Что касается импорта заранее подготовленных программных сценариев, то здесь, если вы имеете общее представление о синтаксисе Nyquist/Lisp, вы можете обнаружить, что синтаксис и структура кода, описываемая нами во фрагменте выше, не совсем похожа на обычный синтаксис Lisp. Команды magic string чувствительны к регистру. И построение строк кода выглядит не совсем элегантно. Но для многих команд и сценариев существует альтернативный синтаксис, доступный путём прямого импорта дополнительных функций Lisp[1]. Для таких операций предпочтительно использовать Audacity® версии 2.4.0, однако если вы работаете в более ранних версиях данной программы

[1] Прим. автора. Каждый из импортированных сценариев имеет имя, начинающееся с AUD — «…», с суффиксом имени пишущей команды. Параметры сценариев передаются в качестве аргументов ключевого слова. При использовании этих команд синтаксиса Lisp присутствует несущественное снижение производительности, поэтому для критически важных приложений (таких как пакетная обработка аудиофрагментов и небольших файлов) предпочтительнее использовать версии AUD-DO.

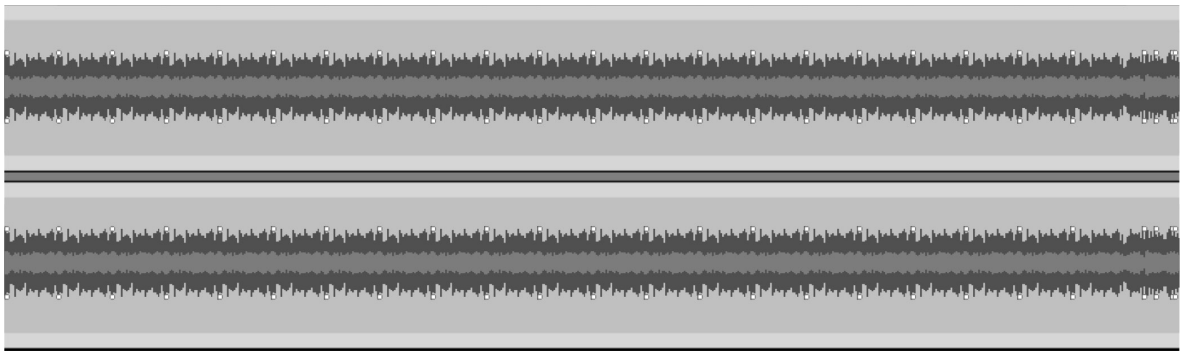


Рис. 26. Аудиоматериал, отмеченный контрольными точками, для поднятия частной амплитуды громкости.

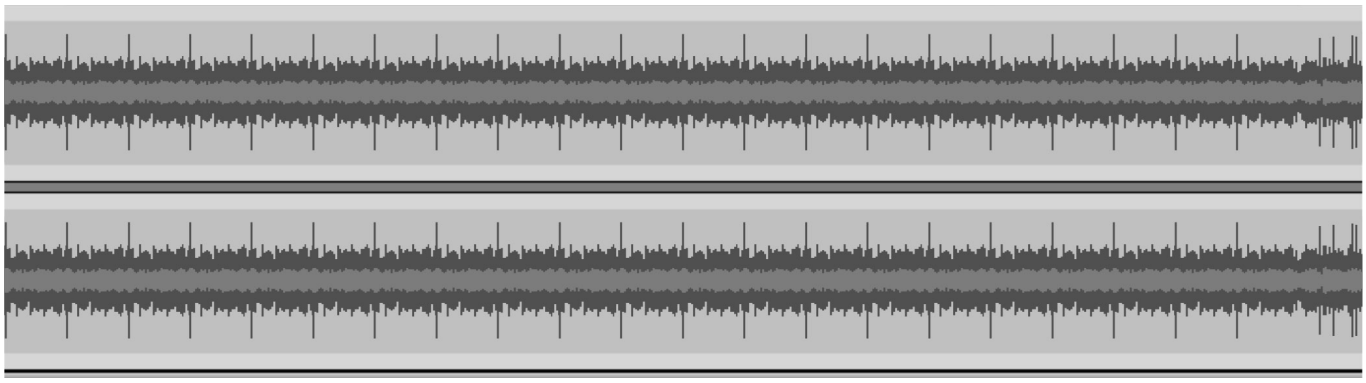


Рис. 27. Результат использования прозрачной огибающей. Пики, выделенные ранее, увеличили амплитуду громкости.

2.3.2[1] или 2.3.3, для импорта Lisp-сценариев можно воспользоваться следующей функцией

```
(aud-import-commands).
```

Поэтому внутрисценарно схема обработки аудиоматериала может выглядеть следующим образом (код 26).

Фрагмент кода показывает возможность применения прозрачной огибающей, с интервалами времени на каждом из которых присутствует отметка. Чуть ниже располагается таблица частот, демонстрирующая изменения амплитуды на выделенных участках (код 27).

Sr[2] — отметка частоты дискретизации. Первые шесть позиций варьируются к нулевой отметке в порядке убывания. Остальные секции таблицы остаются без изменений, единственное изменение — это начальная часть строчки, которая периодически переходит в нисходящую. Для того чтобы воспользоваться опциями временной структуры и определить шкалу затуханий, можно вывести следующую строчку

```
(diff s (env 0.0.1...))
```

Здесь diff определяет структуру времени, может работать в паре с «Returns» при необходимости возврата различий между установленными точками[3], параметр env — остаётся параметром огибающей и создаёт точку или другую разновидность амплитудной модуляции. Таким образом, по сути, схема описывает следующий набор действий:

1. выделить точки, отмеченные маркером;

[1] Прим автора. В Audacity® 2.3.2 и более поздних версиях следующие команды сценариев имеют эквивалентные функции Lisp:
- Создание меню эффектов.
- Встроенные эффекты.
- Таблицы Сценариев I.
- Таблица Сценариев II.

[2] Прим. автора. Sample rate как глобальная переменная содержит (*sound-srate*/*control-srate*=sound-srate-abs/control-srate-abs,), определяет частотную выборку управляющих сигналов (аудиосигнала), предопределяются в любой точке преобразования/sound-srate-abs (48000.0, osc(c4))/. Произведёт вычисления /48000кГц/, даже если общий уровень частоты по умолчанию будет иным.

[3] Прим. автора. Функция может определяться как sum a (prod –1 b), если есть условные a и b.

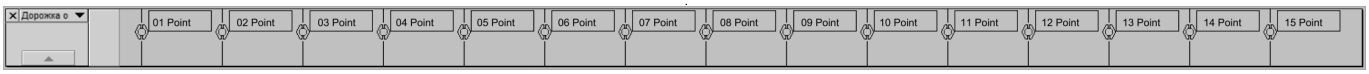


Рис. 28. Дорожка регулярной интервальной разметки в Audacity®.

2. взять все импульсные симметричные точки аудиосигнала (сформировать прозрачную огибающую[1]);
3. увеличить амплитуду пиковой громкости выделенных маркером точек, оставляя остальные точки на прежнем уровне.

На английском языке[2] данные действия будут изложены так:

1. label the marked points;
2. get all pulse symmetric points of the audio signal (create transparent envelope);
3. increase peak volume amplitude of the labeled points, leaving the remaining points at the same level.

В результате мы получаем увеличение амплитуды отмеченных точек, что может быть очень полезно при дизайне, мастеринге и реставрации аудиоматериала.

Для расположения узла стыковки[3] на тайм-линии мы воспользуемся Nyquist Prompt с целью расстановки точек[4] огибающей формы аудиосигнала. На этом этапе считаем важным отметить, что расстановку точек можно также выполнить с помощью интерфейсно-ориентированного инструмента «огибающая» [ ], а расстановку самих меток с помощью инструмента Regular interval labels (регулярная интервальная разметка).

Но здесь нужно принять во внимание тот факт, что инструмент «огибающая» расставляет точки хаотично по движению стрелки мыши. И тут есть несколько моментов:

1. Выделение с помощью мышки и клавиатуры имеет погрешность и не является точным.
2. Отсутствует возможность формирования симметрии расставляемых точек.
3. Отсутствует возможность выравнивания точек по диагонали трека.

Первый момент — заставляет аудиоинженера работать на слух[5], потому что точки ставятся приблизительно, и даже если кажется, что они стоят симметрично, то между ними есть значительный числовой интервал.

Второй момент — актуализирует проблему геометрии отклика аудиосигнала при плавном его спаде. Особенно характерно для стерео аудиосигнала.

Третий момент — делает невозможным процедуры ветвления[6] аудиоматериала, поскольку на глаз невозможно (крайне трудно) выравнивать отмеченные точками участки аудиокомпозиции. С Nyquist Prompt эта проблема становится разрешимой. Для наглядности импортируем обычный музыкальный аудиоматериал в редактор Audacity®, который будет иметь смешанный тип содержимого.

Стереосигнал поделён на две независимые дорожки, которые имеют нестандартный перекрёстный вариант перехода друг в друга с узлом стыковки, немного увеличивающим амплитуду на одной из фаз перехода. Определим себе задачу выделения одной

[1] Прим. автора. Под прозрачной огибающей подразумевается схема разметки аудиоматериала без использования инструмента выделения и его направляющих кривых. То есть, огибающая кривая присутствует на сонограмме только в виде точек. В случае с интерфейсным редактированием на сонограмме отражается и сама линия огибания, вдоль которой можно в хаотичном порядке выстраивать точки. Обратите также внимание, что снимки сонограммы в некоторых случаях имеют разный оттенок (в градациях серого) в независимости от темы, которая используется интерфейсом Audacity®. Здесь можно охарактеризовать два способа выделения спектро-акустической составляющей аудиоматериала интерфейсно-ориентированное выделение и программно-ориентированное выделение. В первом случае огибающая линия присутствует и отображается сама область выделения, во втором случае видны только точки и их геометрия расположения.

[2] Прим. автора. Можно использовать в качестве формулировок (либо дополнительных комментариев) в коде на английском языке, для описания алгоритма исполнения процесса и для строчных комментариев.

[3] Прим. автора. Узлы стыковки бывают разного вида одноранговые и двухранговые. Одноранговый узел стыковки предполагает три ступени плавных затуханий с пользовательским интервалом. Двухранговый узел стыковки подразумевает три и более ступеней затуханий с пользовательским интервалом*.

* Пользовательский интервал — это интервал, время которого заданно пользователем (оператором программы).

[4] Прим. автора. Расстановка точек необходима для оптимального баланса канальной пропускной способности аудиоматериала при его стыковке с другим аудиоматериалом. Точки огибающей могут регулировать степень плавного спада аудиопотока, а также скрывать некоторые акустические нюансы аудиоматериала при его горизонтальном сведении.

[5] Прим. автора. Работа «на слух» заключается в контроле акустического баланса (привязанного к точкам) оператором аудиомонтажа. Таким образом, выявление несовпадений по спектрочастотной амплитуде (провалы громкости, несбалансированные пики) определяются на слух, соответственно наладка такого типа аудиоданных требует большой отдачи со стороны звукорежиссёра (оператора аудиомонтажа).

[6] Прим. автора. Ветвление аудиоматериала это технологический процесс, подразумевающий выравнивание веток точек аудиоматериала с целью обеспечения единого баланса громкости фазы при финальном затухании аудиокомпозиции. То есть тогда, когда необходимо выдерживать временной интервал спада громкости, приводящий впоследствии к полному затуханию конечного фрагмента аудиоматериала.

Рис. 29. Аудиоматериал, имеющий смешанный тип содержимого со сложным узлом стыковки.

Рис. 30. Выделенный за счёт контрольных точек аудиоматериал, имеющий полярную структуру выделения.

```
;amplitude-rms
;Envelopes
L=(/0.1.0.1.0)(/0.0.0.0.0)   R=(/0.1.0.1.0)(/1.0.1.0.1)
   (/0.1.0.1)(/1.0.0.1)       (/0.1.0.1)(/1.0.0.1)
---------------|Fragment of the table|------------------
(/ 2614.000 sr)(/  3064.000 sr)(/ 5315.000 sr) (/ 1000.000 sr)
(/ 2514.000 sr)(/  3063.000 sr)(/ 5314.000 sr) (/ 1000.000 sr)
(/ 2414.000 sr)(/  3062.000 sr)(/ 5313.000 sr) (/ 1000.000 sr)
(/ 1614.000 sr)(/  2064.000 sr)(/ 4315.000 sr) (/ 1000.000 sr)
(/ 1613.000 sr)(/  2063.000 sr)(/ 4314.000 sr) (/ 1000.000 sr)
(/ 1612.000 sr)(/  2062.000 sr)(/ 4313.000 sr) (/ 1000.000 sr)
```

Код 28

из полярностей[1] стереосигнала (для дальнейшего поднятия уровня громкости пиков). Это будет небольшая часть левой и правой полярностей с отдельным эпизодом выделения пика узла стыковки (в этом случае для контроля перехода). При этом левая (L) и правая (R) стороны аудиосигнала будут иметь различную структуру разметки. Фрагмент, следующий за данными изменениями, останется нетронутым (т.е. не будет подвержен редактированию). Для этого обратимся к таблице частот подобно той, которая использовалась выше при симметричном выделении пиков (код 28).

[1] Прим. автора. В этом процессе под полярностями мы понимаем симметричную либо асимметричную геометрическую форму отражения аудиосигнала визуализируемого на сонограмме.

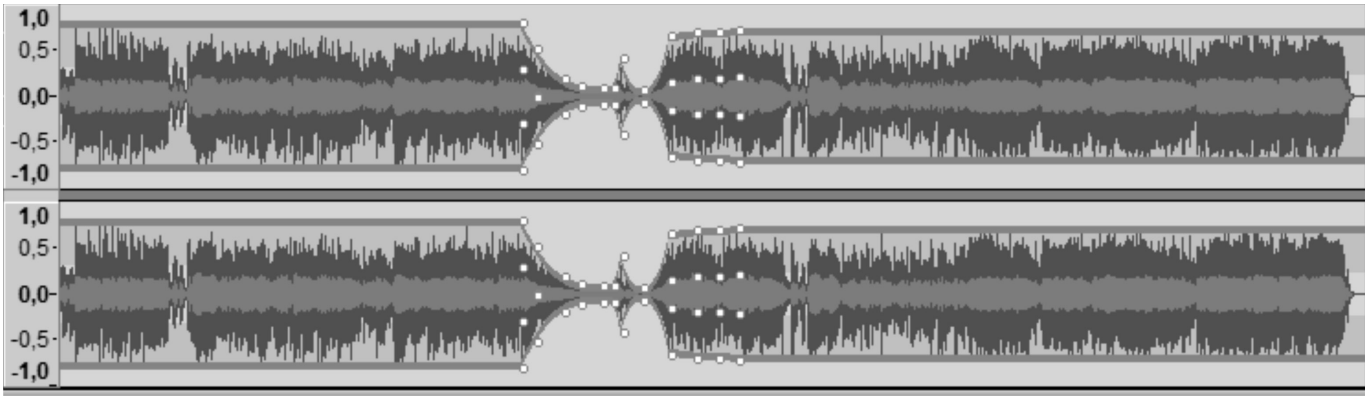


Рис. 31. Выделение того же аудиофрагмента интерфейсно-ориентированным способом, используя стандартные кривые, предусмотренные интерфейсом программы Audacity®.

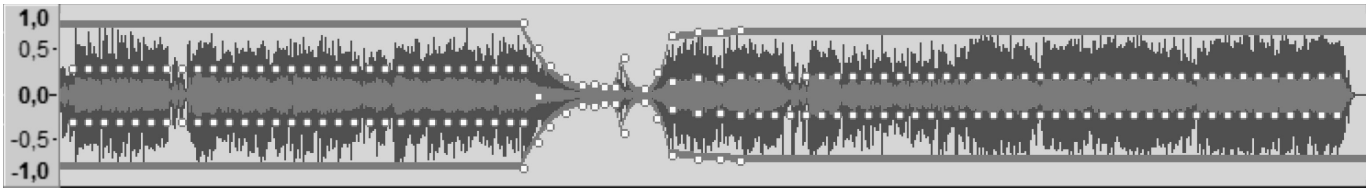


Рис. 32. Комбинированный способ выделения аудиофрагмента с использованием интерфейсно-ориентированных опций (стандартные огибающие) и опций программно-ориентированных (прозрачные огибающие).

Для упрощения изложения материала, учитывая объёмы научной статьи, мы привели начальный заголовок программной таблицы, характеризующий установленные на данном редактируемом аудиоматериале частоты. Небольшое отличие данного фрагмента таблицы от её предыдущего аналога состоит в том, что выделенные частоты, идущие в порядке возрастания (2614–5315), *незначительно снижаются* для перехода в более *нижние пределы* частот (2614–1614). Такое снижение позволяет применять любой из перечисленных в статье аудиоплагинов на любом выделенном точками фрагменте.

Мы добились выделения полярностей пиковых значений громкости с помощью программной таблицы, используя прозрачную огибающую.

Давайте посмотрим как будет выглядеть подобный результат (с использованием того же фрагмента аудиоматериала), в контексте применения интерфейсно-ориентированного инструмента «изменение огибающей»

Как мы видим, форма изменения огибающих выглядит несколько по-другому, точки, фиксирующие изменения значений по RMS и точки контролирующие отражение её пиков, зависят от видимых огибающих стереосигнал кривых. Поэтому в случае ультра точного компьютерного аудимонтажа больше подходит выделение с прозрачными кривыми. Однако в некотором смысле эти два вида выделения можно совмещать. В первом случае должно идти интерфейсно-ориентированное выделение аудиосигнала, а во-втором — программно-ориентированное с применением программной таблицы. В результате мы будем иметь более сложную (комбинированную) структуру выделения, как показано ниже

Итак, мы добились наиболее точных показателей выделения структуры аудиоматериала и его фрагментов[1], благодаря чему можно существенно повысить качество обработки звука в Audacity®, и расширить горизонты экспериментов со звуком. Ниже представлен интерфейсно-ориентированный способ работы с огибающей

[1] Прим. автора. Обратите также внимание на форму выделения RMS, в первом аудиофрагменте точки располагаются во внепиковой зоне (кроме одной точки), а во втором — во внутрипиковой. Это обстоятельство позволяет сделать следующие выводы:

1) Изменение математических параметров RMS можно вести независимо от центральной фазы (единственное, что нужно сделать, это поставить точку узла пересечения огибающей).
2) При сведении аудиодорожек либо их спектральном редактировании изменения, внесённые с помощью манипулирования амплитудой громкости, будут чётче фиксироваться спектрограммой, в дальнейшем это позволит убирать некорректные пики уже путём спектрального редактирования.
3) Редактирование RMS, так же как и в случае с пиками, можно вести симметрично, подгоняя отдельные частотные диапазоны под установленный оператором шаблон.

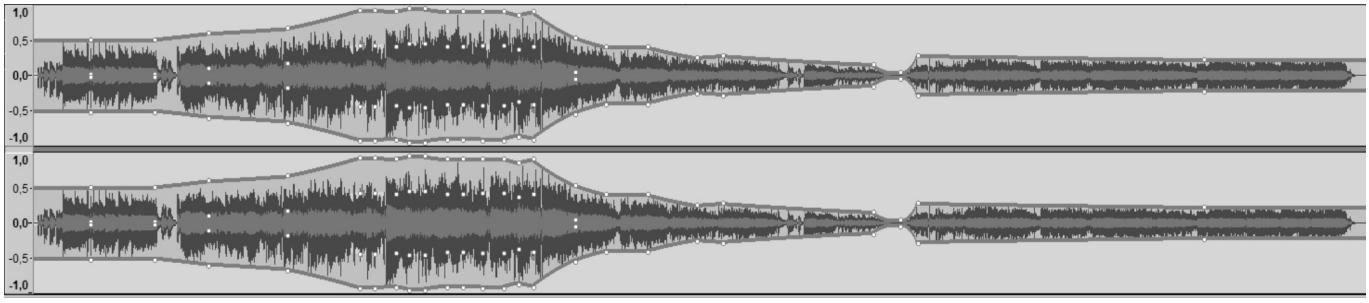


Рис. 33. Аудиоматериал со смешанным содержимым и управляемый стандартными кривыми (кривые огибающей).

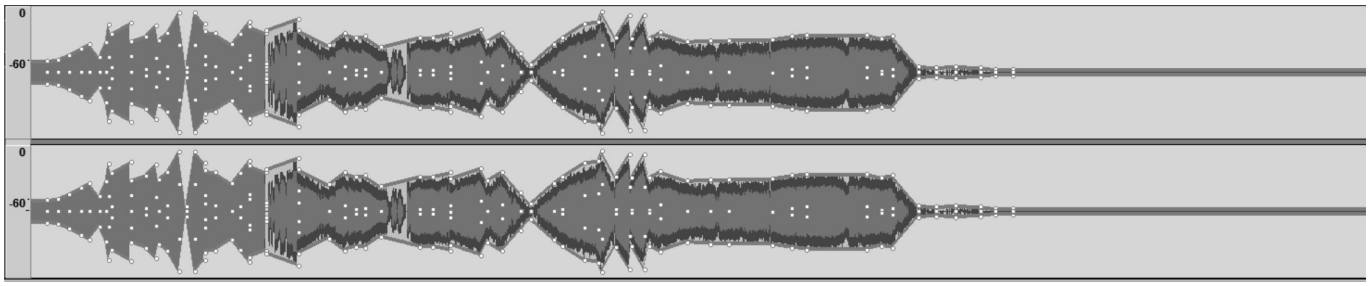


Рис. 34. Аудиоматериал со смешанным содержимым, имеющий нестандартную структуру RMS — результат совмещения выделений стандартной и прозрачной кривых.

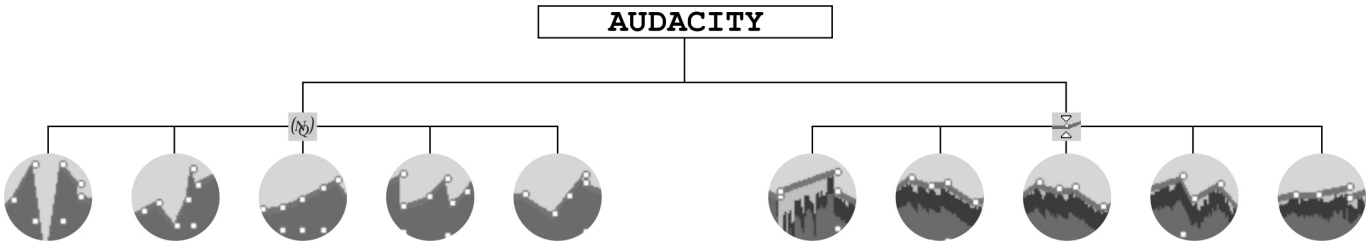


Рис. 35. Управление огибающими посредством контрольных точек. Слева — контроль, над точками, осуществляемый через приглашение NyquistPrompt. Справа — контроль, над точками, осуществляемый через интерфейс программы.

аудиосигнала (рис. 32), демонстрирующий развитость этой опции, и ещё ниже для наглядного представления показан комбинированный способ работы с обычной и прозрачной огибающими (рис. 33).

Читатель, наверное, обратил своё внимание на нестандартную структуру аудиоматериала в начале представленной на сонограмме композиции (Рис. 33). В действительности это стерео материал со смешанным содержимым, в начале которого присутствует искусственно сгенерированный однородный аудиосигнал, над отрезком которого не проходит обычная огибающая Audacity®, её свод начинается над оригинальным материалом, имеющим тип смешанного содержимого. Кроме того, искусственный сигнал не имеет пиков, поэтому его точки контролируют исключительно RMS. Первая часть точек на аудиофрагменте контролируется программной *таблицей частот*, а вторая часть точек контролируется *интерфейсной опцией* «изменение огибающей» (рис34).

Таким (комбинированным) подходом можно добиться наиболее востребованного результата при сведении аудиоматериала разных типов и жанров, к тому же «комбинированный подход» очень полезен при реставрации аудиоматериала, так как подобный аудиоматериал имеет большое количество акустических погрешностей, требующих заглушения либо полного удаления.

Удаление конечно лучше всего производить в спектральном режиме редактирования, а заглушение, при использовании комбинированного подхода, можно выполнять на обычной звуковой форме, располагая

```
sound-srate-abs 44100
ñontrol-srate-abs 44100
```
−159.03437/−167.82133

Код 29

$$\left[\begin{array}{c} 0.1\ 0.5\ 0.1 \\ \hline 1.0\ 5.0\ 1.0 \\ \hline \text{pwlv}\ 1\ 1\ 0 \end{array}\right] \rightarrow \left[\begin{array}{c} 1.1\ 0.1\ 0.1 \\ \hline 1.1\ 0.1\ 0.1 \\ \hline g0\ g1|{>}\,g0\ g1 \end{array}\right]$$

```
(defun r-cos (dur)
  (abs-env
    (mult 0.5
      (sum 1
        (osc (hz-to-step (/ (* dur 2))) dur *table* 90)))))
```

Код 30

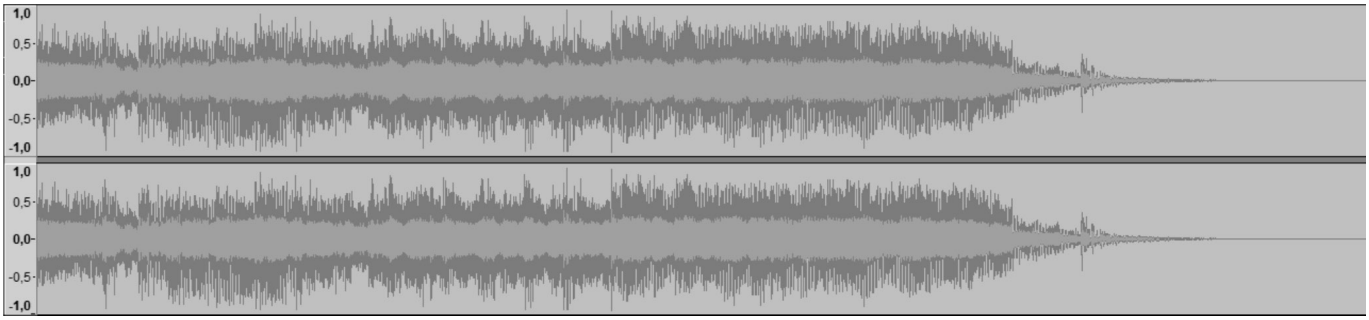


Рис. 36. Плавное затухание при отрицательной пиковой амплитуде
(Прим. автора. Имеется ввиду плавное понижение общей пиковой амплитуды громкости с учётом RMS.)

контрольные точки над нежелательными артефактами аудиосигнала. Там где имеются ярко выраженные акустические артефакты на протяжении всего хронометража аудиоматериала, такой подход позволит, не затрагивая полезный аудиосигнал, приглушать и заглушать потенциально повреждённые участки аудиоматериала.

Ввод фрагментов программного кода, описывающего алгоритм спада с применением различных подходов выделения участков аудиосигнала (используя обычную и прозрачную) огибающие, позволит модифицировать алгоритмы и видоизменять геометрию плавного затухания аудиосигнала. При горизонтальном нелинейном сведении аудиотреков либо их отдельных фрагментов как раз очень актуальна возможность изменения траектории спада спектро-частотной амплитуды.

Допустим, мы хотим сделать плавное затухание аудиосигнала при общей пиковой амплитуде −159.03437дБ и при среднеарифметическом показателе среднеквадратического значения −167.82133 дБ. Устанавливаем параметры частоты дискретизации в соответствии с нашим проектом (код 29).

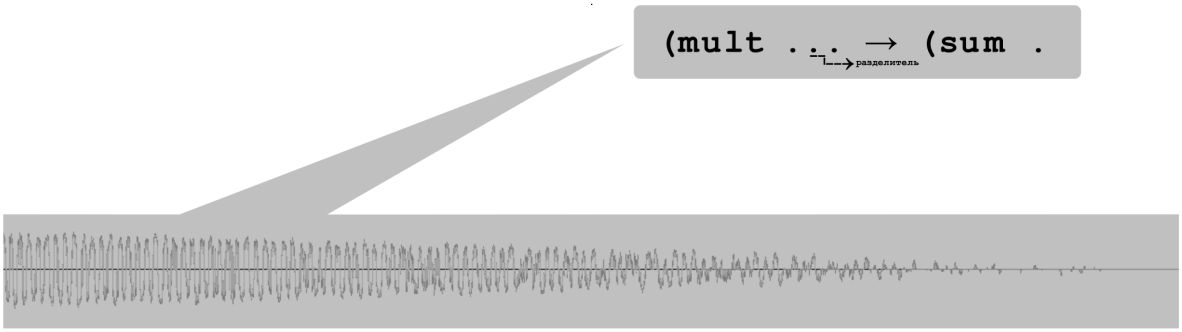
Дополняем программный код алгоритма студийного затухания согласно нижеследующей схеме, используя заголовки частотной таблицы (ячейки) таким образом, чтобы уровни громкости соответствовали параметрам mult[1] (код 30).

В результате мы получаем крутой, а затем мягкий склон частотной амплитуды в конце аудиокомпозиции с псевдокомпрессионным[2] эффектом сужения каналов на выходе[3] (рис. 34).

[1] Прим. автора. В зависимости от того, какой аудиоматериал будет выбран. Если это будет аудиоматериал с характеристиками моно, то значения g1 в правой части столбца заголовка таблицы поменяют очерёдность. Если аудиоматериал будет иметь характеристики стерео, то здесь всё остаётся без изменений, как показано на схеме.

[2] Прим. автора. Под псевдокомпрессионным эффектом здесь понимается сужение частотного диапазона выделенного фрагмента аудиоматенриала для достижения отрицательной пиковой амплитуды с сохранением оригинальной динамики RMS. В этом случае мы можем подчеркнуть (усилить, выделить) различные частоты, не затрагивающие основной RMS-поток. Такая настройка фейдинга позволит обратить внимание на определённые акустические блоки и подчеркнуть желаемые гармоники в затухающем аудиоматериале. Очень важно для современного компьютерного сведения аудиоматериала разной жанровой направленности.

[3] Прим. автора. Формируемый в результате манипуляций с ячейками таблицы «мягкий склон» иллюстрирует рис. 35, где в качестве основных инструментов




```
(defun r-cos (dur)
  (abs-env
    (mult 0.5
      (sum 1
        (osc (hz-to-step (/ (* dur 2))) dur *table* 90)))))
```

Рис. 37. Результат работы элементов управления Nyquist mult/sum.

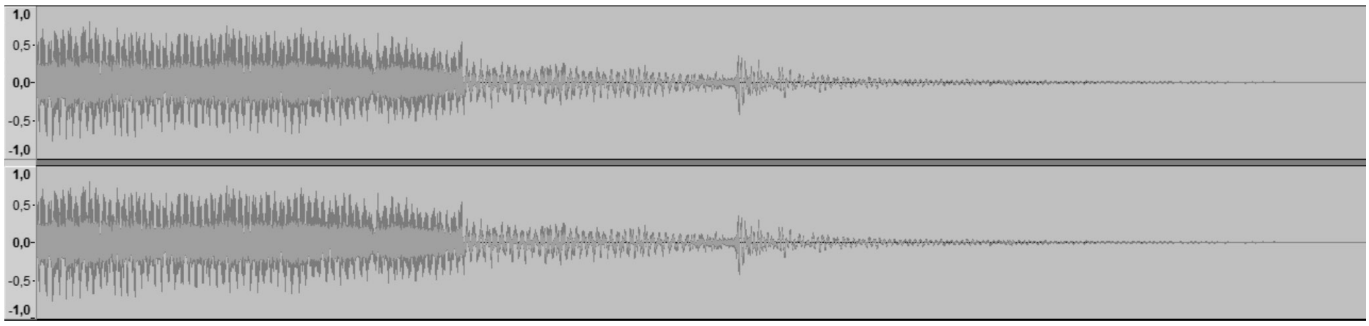


Рис. 38. Nyquist mult/sum — полный фрагмент затухания.

Основными элементами управления частотной таблицей здесь будут mult/sum.

К примеру, изменения числовых значений в положительную сторону 0.5 (mult) — в первом случае приведут к уплотнению спектрочастотной амплитуды, тем самым повысят её громкость относительно точки выделения аудиофрагмента, изменение значений в отрицательную сторону приведут к обратному эффекту, т.е. к затуханию аудиосигнала в противоположную сторону от установленного порога выделения. Sum здесь всё также остаётся классом суммы, который формирует сумму чисел по одной от каждого из двух других выбранных образцов. Соответственно при увеличении этого значения повысится громкость и увеличится плотность аудиосигнала. Это значение хорошо использовать в корреляции с mult для имитации жирности[1] аудиосигнала в период его затухания.

Данные операции достаточно хорошо разрешают ситуации с профессиональным (точным) сведением аудиодорожек. Что касается регулируемого спада спектро-частотной амплитуды в целом и спада внутрифазной амплитуды межфрагментарных акустических блоков, то здесь можно рассмотреть следующий пример:

На рисунке 37 изображён фрагмент сигналограммы в виде звуковой формы, тёмно-серым оттенком

управления ячейками выступают mult и sum. В перспективе настройка сонограммы может быть отсечена (масштабирована по хронометражу) по внутренней и общей фазам, согласно рис. 37.

[1] Прим. автора. При компьютерном аудиомонтаже звукозаписей, разделённых на различные дорожки особенно при их дальнейшем сведении, бывает необходимо подчеркнуть некоторые частоты при затухании аудиоматериала, процесс усложняется вдвойне, если выбранный фрагмент характеризуется смешанным содержимым аудиоматериала, и если, к примеру, нужно подчеркнуть элементы вокала или клавишного инструмента (на участке перехода от дорожки к дорожке), то обычно приходится прибегать к сторонним эффектам (это может быть компрессия, клиппинг, эквализация). Все перечисленные эффекты так или иначе видоизменяют содержимое аудиоматериала и могут ухудшать спектро-частоные характеристики других фрагментов обрабатываемого трека. Поэтому в данном случае мы используем обычный пересчёт громкости аудиоданных, не приводящий к существенной деформации аудиоматериала (не изменяющий не выделенных для обработки гармоник), эффект жирности здесь во многом будет зависеть от увеличения изначальной громкости. Это и есть та самая тонкая настройка, позволяющая (без применения сторонних плагинов) производить точную подстройку всех элементов сонограммы под общую настройку в соответствии с установленными значениями проекта.

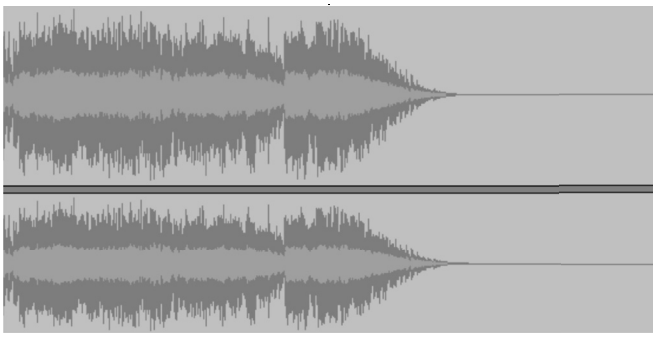


Рис. 39. Общая и внутренняя фазы и плавное затухание.

```
(defun filter (sig dur)
 (abs-env
   (let* ((nyq-hz (/ *sound-srate* 2))
          (f-out (r-cos (min (/ dur 2.0) 0.5)))
          (f-in (diff (snd-const 1 0 *sound-srate* dur) f-out)))
     (sim
       (mult f-out sig)
       (mult f-in (lp sig (pwlv nyq-hz dur 100)))))))

(defun r-cos (dur)
  (abs-env
    (mult 0.5
      (sum 1
        (osc (hz-to-step (/ (* dur 2))) dur *table* 90)))))

        (let ((dur (get-duration 1)))
  (cond
    ((< len 3) "Selection too short.\nIt must be more than 2 samples.")
    ((< dur 0.2) (mult s (r-cos dur)))
    (t (mult (filter s dur)(r-cos dur)))))
```

Код 31

обозначена общая фаза — (PH), светло-серым выделена внутренняя фаза (InPH). На индикаторах присутствуют соответствующие описания PH — фаза (phase), InPH — внутренняя фаза (Intraphase). Фаза представляет собой общую пиковую громкость, внутренняя фаза характеризует оригинальный уровень звукового давления (ядро громкости). Если нам необходимо провести калибровку громкости аудиоматериала в пределах установленного хронометража по InPH (RMS), то здесь можно использовать связку ((= g0 g1) g0), которую может характеризовать направление кривой (scale-curve g0 g1 (cosine-curve g0 g1)))))). Сдвиг общей фазы (PH) по громкости в сторону увеличения либо уменьшения амплитуды сигнала может контролироваться точками входа[1] (g0 g1) / (– g0 g1)).

Чуть ниже представлено полное описание алгоритма затухания (Studio Fade Out) на языке Nyquist (код 31).

[1] Прим. автора. Имеются в виду точки входа на увеличение либо на уменьшение (в соответствии с параметрами кода и выбранным алгоритмом затухания).

Здесь abs — абсолютное числовое значение. Expr — число. Returns — возврат абсолютного числового значения. Cond — контрольная конструкция, выполняющая условную оценку, может использоваться как связка cond pair, где pair — пара состоящая из (pred expr…). Pred — предикатное выражение (выражение с допущением). Expr оценивается, если допущение не является nil. Returns — возврат значения первого выражения, в котором допущение не является nil. Прежде чем подойти к выводам нашей статьи и для полноты научного обзора стоит отметить ещё один фиксированный модуль, имеющий отношение непосредственно к корректировке частотного спада аудиосигнала это сокращённый вариант модуля «Studio Fade Out» — «Fade In» → «Fade Out». Модуль представляет собой раздельные секции Fade In — плавное нарастание звукового давления, Fade Out характеризует плавный спад звукового давления.

Модуль Fade In является частью самой программы, однако основные его параметры изложены в интерпретации Nyquist и выражаются кодом 32:

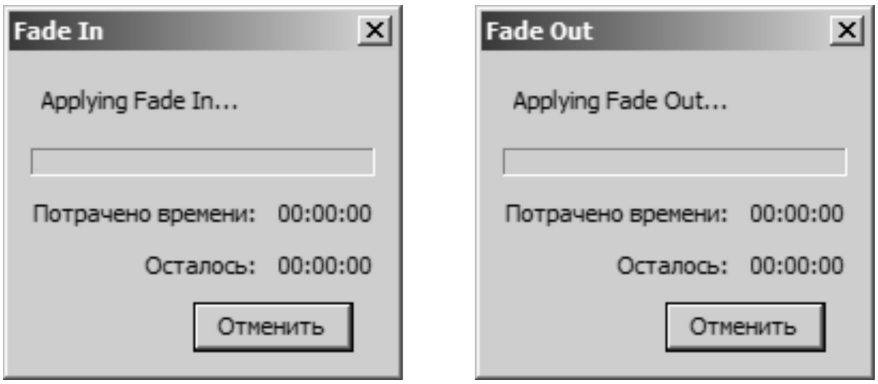


Рис. 40. Интерфейс модулей обработки аудиоданных Fade In/Fade Out.

```
(mult *track* (pwlv 0 1 1 1.1 1))
```

Код 32

```
(mult *track* (pwlv 0 1 2 1.1 0))
```

Код 33

```
(mult *track* (pwlv 1 1 0))
```

Код 34

```
(mult *track* (pwlv 2 1 0))
```

Код 35

Для эффекта акустической вспышки в начале трека можно использовать следующие настройки (код 33).

Строчка кода позволяет эмулировать постепенное линейное плавное нарастание аудиосигнала на входе.

Плавное снижение аудиосигнала обеспечивает модуль Fade Out, который выражается следующей строчкой (код 34).

Для достижения эффекта акустической вспышки[1] можно использовать следующую настройку (код 35):

И тот и другой модули можно корректировать через Nyquist Prompt, ход выполнения процесса характеризует следующая иллюстрация

При горизонтальном либо вертикальном сведении больших массивов аудиоданных может потребоваться многошаговое затухание (Multi-step fade out), которое обеспечит плавный спуск амплитуды аудиосигнала на определённых отметках (код 36).

Шаги определяются цифрами, которые являются идентификационными метками плавного фрагментарного спуска аудиосигнала на каждом из участков. К примеру, стандартная формулировка многошагового затухания будет выглядеть следующим образом (код 37).

[1] Прим. автора. Акустическая вспышка — специализированный акустический эффект, при котором амплитуда аудиоматериала (в отмеченном участке сигнала) резко возрастает на единицу времени, после чего резко спадает. Эффект акустической вспышки чаще всего употребим при сведении различных аудиодорожек, причём, в первую очередь, при инструментальном сведении, к примеру перкуссии, которая может иметь разные циклы ударов. Также данный эффект может применяться при сведении вокальных партий с бэк-вокалом для уплотнения некоторых участков общей акустической картины сведённых партий.

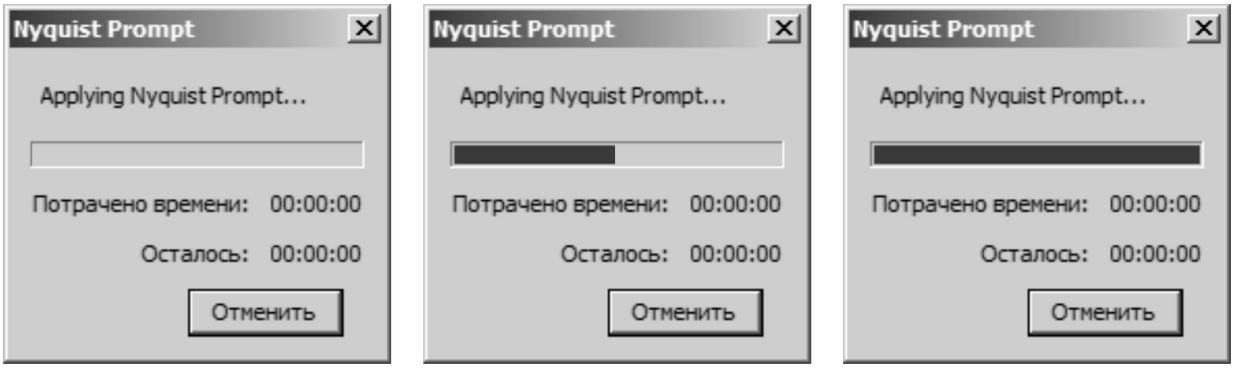


Рис. 41. Шкалы выполнения процесса модулем NiquistPrompt.

```
(_ _._ _.___ _._ _.___ _._ _.___ _.__ _.___ _._ _._))
```

Код 36

```
(_ _._ _.___ _._ _.___ _._ _.___ _.__ _.___ _._ _._))
 ↓ ↓ ↓ ↓ ↓↓↓ ↓ ↓ ↓ ↓↓↓ ↓ ↓ ↓ ↓↓↓ ↓ ↓↓ ↓ ↓↓↓ ↓ ↓ ↓ ↓
(1 0.4 0.775 0.8 0.447 0.9 0.316 0.95 0.224 1.0 0.0))
```

Код 37

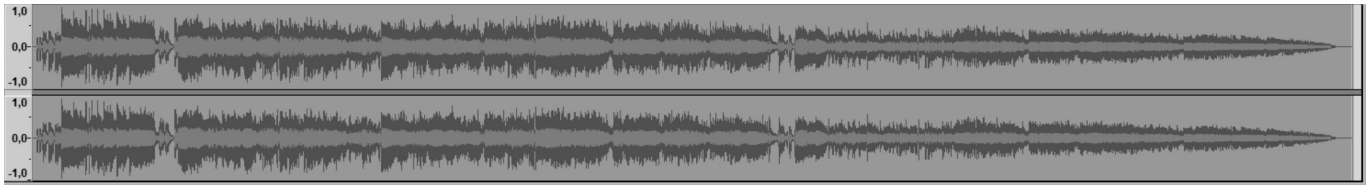


Рис. 42. Многошаговое затухание.

```
(mult *track* (pwlv 1 0.4 0.775 0.8 0.447 0.9 0.316 0.95 0.224 1.0 0.0))
```

Код 38

Сигналограмма примет следующий вид (рис. 42).

Полная строчка кода для ввода в NyquistPrompt будет следующей[1] (код 38).

Для того чтобы усилить крутизну эффекта (код 39).

Соответственно полная строчка кода для ввода в NyquistPrompt будет таковой (код 40).

[1] Прим. автора. При вводе команд на языке Nyquist параметр Use Legacy (version 3) syntax должен быть не активен. В противном случае оболочка NyquistPrompt не предпримет шагов для обработки аудиосигнала. При откате команды назад — действие ввода команд Nyquist не будут учитываться программой. Для лабораторных экспериментов автором статьи использовалась Audacity® версии 2.1.3. В других примерах использовалась текущая версия Audacity® 2.4.2.

Сигналограмма примет следующий вид (рис. 43).

В тех случаях, когда необходимо чтобы кривая синуса постепенно нарастала, противоположно многошаговому затуханию можно воспользоваться следующим формуляром (код 41).

Сигналограмма примет следующий вид (рис. 44).

Чтобы усилить центр и создать плавный спад в начале и в конце аудиокомпозиции, можно применить следующий код (код 42):

Сигналограмма преобразуется в следующий вид (рис. 45).

```
(0 0.0 0.000 0.0 0.000 0.0 0.000 0.00 0.000 0.0 0.0))
 ↓ ↓ ↓ ↓ ↓↓↓ ↓↓↓ ↓ ↓↓↓ ↓ ↓ ↓ ↓↓↓ ↓ ↓↓ ↓ ↓↓↓ ↓ ↓ ↓ ↓
(1 0.4 0.775 0.8 0.447 0.9 0.316 0.95 0.224 1.0 0.0))
 ↓ ↓ ↓ ↓ ↓↓↓ ↓↓↓ ↓ ↓↓↓ ↓ ↓ ↓ ↓↓↓ ↓ ↓↓ ↓ ↓↓↓ ↓ ↓ ↓ ↓
(1 0.5 0.775 0.4 0.500 0.9 0.700 0.95 0.740 1.0 0.1))
```

Код 39

```
(mult *track* (pwlv 1 0.5 0.775 0.4 0.500 0.9 0.700 0.95 0.740 1.0 0.1))
```

Код 40

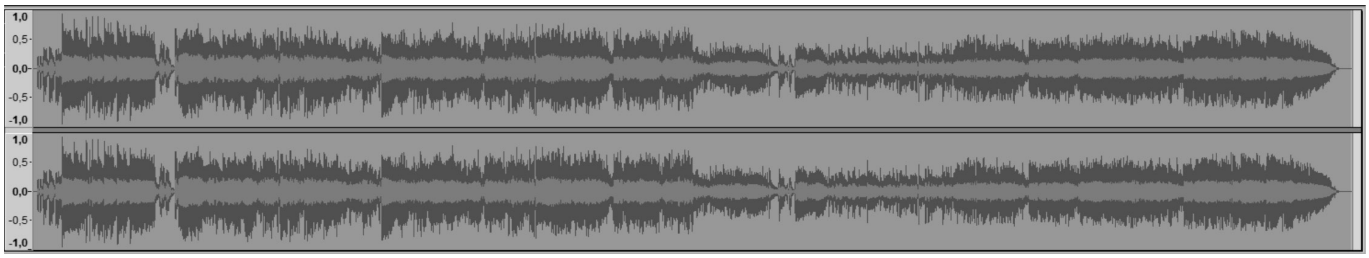


Рис. 43. Многошаговое затухание с соответствующей изложенному программному коду «крутизной».

```
(mult *track* 0.5
  (sum 1
    (osc (hz-to-step (/ (get-duration 2)))
      1 *table* -90)))
```

Кдо 41

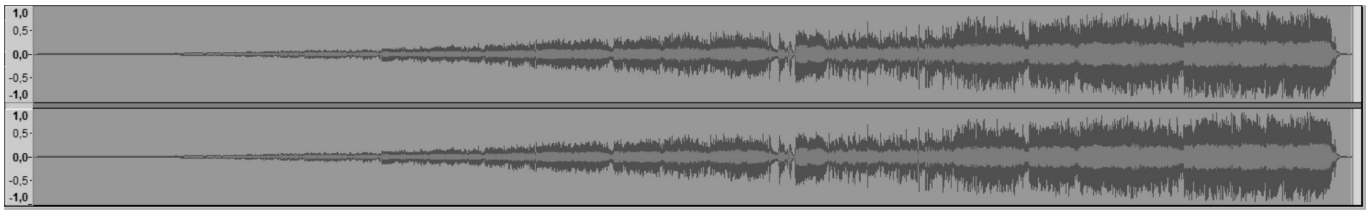


Рис. 44. Многошаговое затухание на входе аудиосигнала.

В некоторых случаях при сведении может пригодиться эффект волнообразных затуханий в течение всего сигнала, такой эффект будет полезен при создании волнообразных сонограмм, состоящих из сольных партий различных инструментов или вокала. Например, при сведении струнных инструментов — басовой (электро) и акустической гитар. Для этого можно воспользоваться следующей формулировкой кода (код 43).

Для усиления крутизны повышаем значение mult и sum и можем наблюдать изменение геометрии сонограммы в сторону уплотнения пиков (рис. 46).

В инженерной практике также может быть полезен эффект волнообразного конечного затухания, для этого можно использовать следующий код (код 44).

По умолчанию сигналограмма будет такой (рис 48).

Для того чтобы добиться широко распространённого в компьютерном монтаже аудиоматериала эффекта волнообразного сужения каналов на выходе, следует ввести код со следующими корректировками (код 45).

```
(mult *track* 0.3
  (sum 2
    (osc (hz-to-step (/ (get-duration 1)))
      1 *table* -80)))
```

Код 42

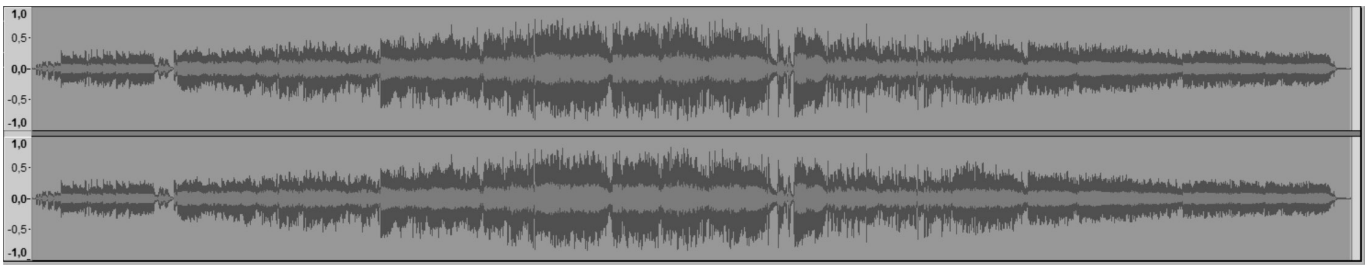


Рис. 45. Многошаговое затухание со смещённым к центру увеличением амплитуды аудиосигнала.

```
(setf wiggle (mult 0.2 (hzosc (/ 6.0 (get-duration 1)))))
(mult *track* (sum 1.0 wiggle))
```

Код 43

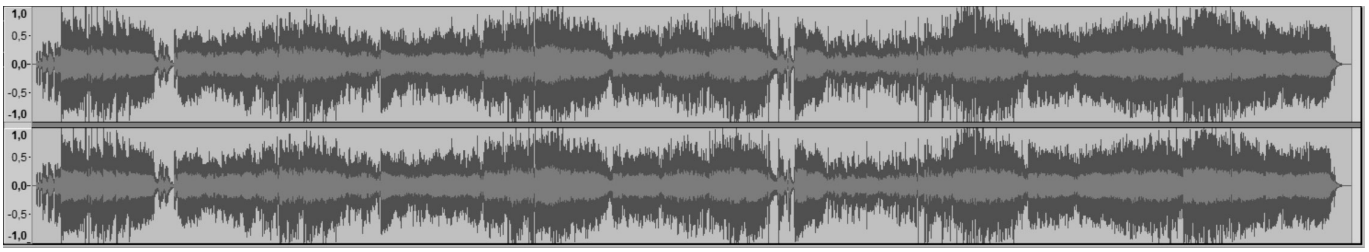


Рис. 46. Пример волнообразных многошаговых затуханий.

```
(setf wiggle (mult 0.5 (hzosc (/ 3.0 (get-duration 1)))))
(mult *track* (sum 1.1 wiggle))
```

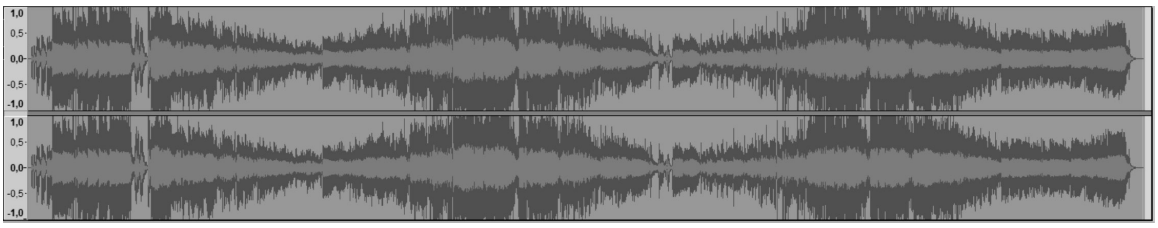


Рис. 47. Пример волнообразных многошаговых затуханий в соответствии с изменёнными по «крутизне» параметрами программного кода.

```
(setf wiggle (mult 0.2 (hzosc (/ 6.0 (get-duration 1)))))
(mult *track* (pwlv 1 1 0) (sum 1.0 wiggle))
```

Код 44

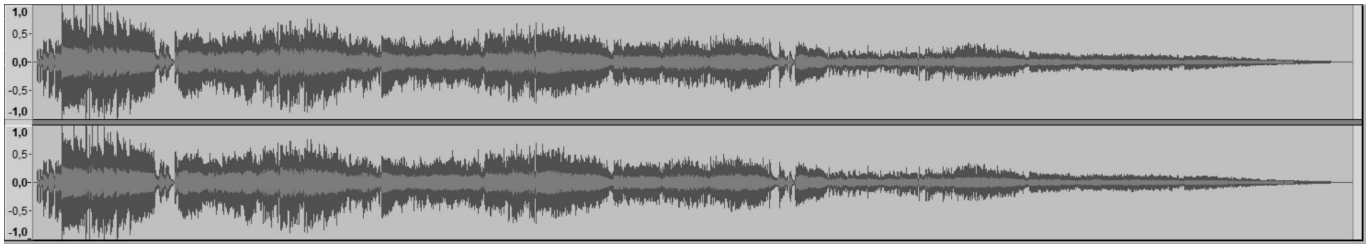


Рис. 48. Пример волнообразного конечного затухания.

```
(setf wiggle (mult 0.5 (hzosc (/ 4.0 (get-duration 1)))))
(mult *track* (pwlv 1 1 0) (sum 2.0 wiggle))
```

Код 45

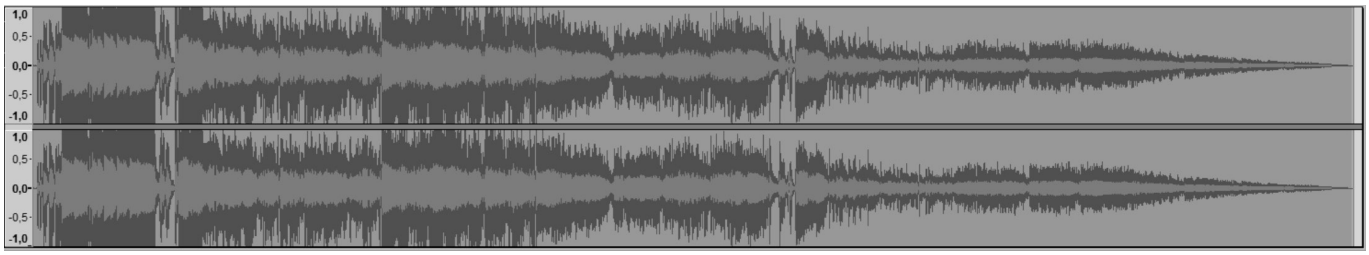


Рис. 49. Пример волнообразного сужения каналов на «выходе».

Сигналограмма примет несколько иной вид (рис. 49).

С помощью простых эффектов затуханий и их описаний на языке Nyquist можно достичь и некоторых других интересных эффектов, полезных при сведении аудиоматериала. В процессе подбора литературы для работы над данной статьёй автор наткнулся на интересную модификацию кода Nyquist[1], способную создавать интервал внутри выбранной дорожки на заданном пользователем участке с промежуточным нарастанием и затуханием аудиосигнала (рис. 48), код 50.

Здесь (totalDuration (/ len *sound-srate*)) соответствует количеству выборки. (segmentDuration (/ totalDuration 3)) характеризует параметры затухания, тишины и время спада. (decay-end (/ segmentDuration totalDuration)) определяет конец постепенного затухания. (ramp-start (/ (* segmentDuration 2) totalDuration)) выявляет конец тишины. (ramp-end (/ (* segmentDuration 3) totalDuration)) — конец постепенного нарастания аудиосигнала. (len+1 (/ (1+ len) len))) — Время Nyquist в len + 1 выборка. Далее смотрим следующие две позиции изменения и смещения сонограммы (рис. 49,50).

Таким образом, мы рассмотрели оснловные инженерно-технические возможности языка программирования Nyquist (в программной среде Audacity®) в области корректировки точности плавного снижения и повышения спектро-частотной амплитуды аудиосигнала.

[1] Прим. автора. Модификация размещена по электронному адресу: http://forum.audacityteam.org/viewtopic.php?f=39&t=53408 / Название эффекта — Dual-Fade. Автор edgar-rft (дата обращения к источнику: 15.01.2021). С изменениями и дополнениями с ресурса: https://www.christeck.de/2012/08/21/dual-fade-nyquist-plugin-for-audacity/ (дата обращения к источнику: 15.01.2021).

```
(defun dualfade (sound)
  (let* ((totalDuration    (/ len *sound-srate*))
         (segmentDuration  (/ totalDuration 3))
         (decay-end        (/ segmentDuration totalDuration))
         (ramp-start       (/ (* segmentDuration 2) totalDuration))
         (ramp-end         (/ (* segmentDuration 3) totalDuration))
         (len+1            (/ (1+ len) len)))
  (control-srate-abs *sound-srate*
      (mult sound (pwl 0.0 1.0 decay-end 0.0
      ramp-start 0.0 ramp-end 1.0 len+1)))))
(dualfade s)
```

Код 50

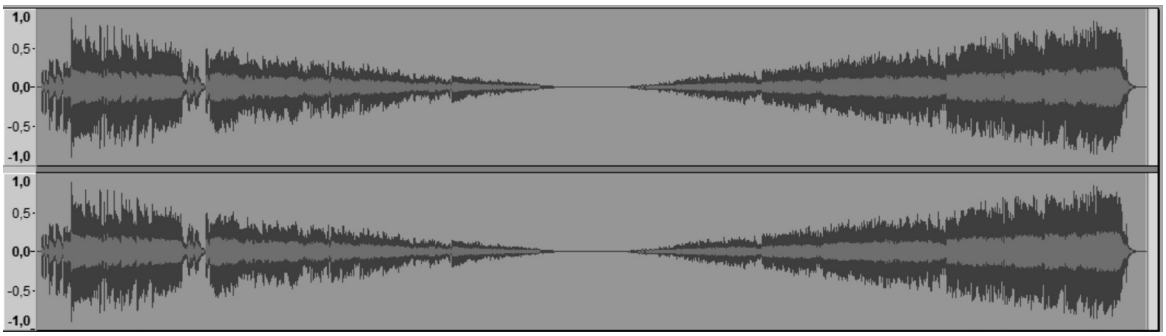


Рис. 50. Вариация использования интервальной тишины (с возможной коррекцией её хронометража).

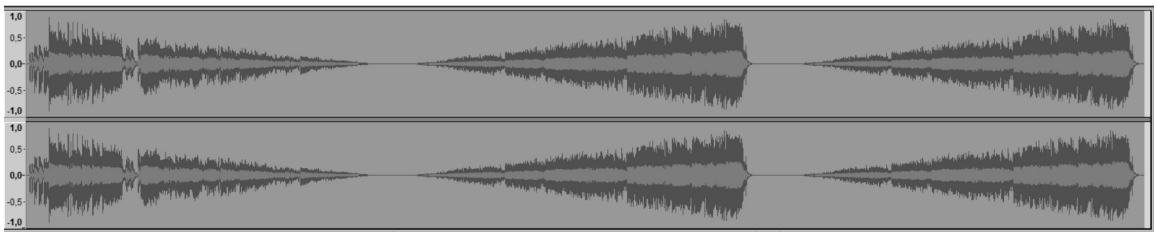


Рис. 51. Вариация использования интервальной тишины: спад ⟶ интервал ⟶ нарастание ⟶ интервал ⟶
нарастание ⟶ интервал.

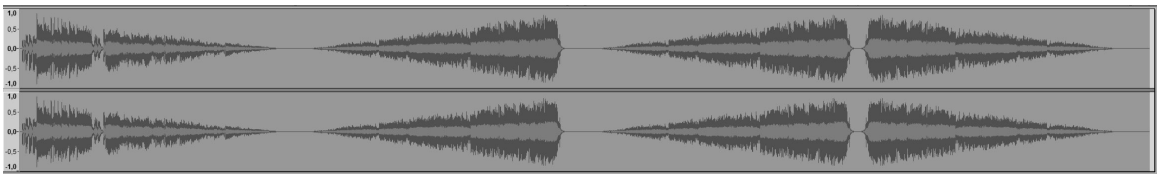


Рис. 52. Вариация использования интервальной тишины: спад ⟶ интервал ⟶ нарастание ⟶ интервал ⟶ нарастание ⟶ интервал ⟶ спад. Интервальная тишина правого аудиофрагмента сдвинута по хронометражу общей дорожки.

Автор попытался систематизировать и упорядочить все основные визуальные инструменты интерфейса программы Audacity®, отвечающие за редактирование и точную настройку выделенных фрагментов аудиоматериала, при этом, не забывая про терминальные функции, открывающие дополнительные возможности по балансировке отладке и точной настройки отдельных подключаемых модулей программы. В принципе перечисленные технические процедуры редактирования аудиоматериала можно отнести к гибридному способу обработки аудиоданных. Именно в силу совместного и поочерёдного использования элементов интерфейса подключаемых (штатных) модулей обработки аудиосигнала.

Примеры, рассматриваемые в нашей научной статье, могут послужить хорошей визуально-схематической инструкцией для специалистов в области компьютерного сведения звука и в сфере прикладной аудиоинженерии и аудиоинформатики, органично дополняющей традиционные практики корректировки, компоновки, рендеринга и распределения аудиоматериала.

Автор надеется, что такой гибридный подход качественно улучшит инженерно-техническую отрасль обработки аудиоматериала, а также сократит время, затрачиваемое звукоинженерами на компоновку аудиоматериала.

Приложение 1

Единичные и двоичные операторы
для формирования выражений

1. [+] — сложение, со звуковым сопровождением.
2. [–] — вычитание, со звуковым сопровождением.
3. [*] — умножение, со звуковым сопровождением.
4. [/] — деление (из-за случаев деления на ноль без звукового сопровождения).
5. [%] — числовая характеристика (остаток от деления).
6. [^] — потенцирование.
7. [=] — равенство (использует Lisp equal для «несписочных» значений и сравнивает последовательно элементы списка в обратном порядке).
8. [!=] — не равно.
9. [>] — больше чем.
10. [<] — меньше чем.
11. [>=] — больше или равно.
12. [<=] — меньше или равно.
13. [~=] — приблизительно равно. Числа являются приблизительно равными, если они располагаются в пределах расстояния *~=tolerance* друг от друга. *~=tolerance* изначально равно 0,000001. Не числовые значения сравниваются при помощи функции XLISP equal, а списки сравниваются поэлементно в обратном порядке, используя ~=.
14. [&] — логический «и».
15. [|] — логический «или».
16. [!] — не логический (единичный).
17. [@] — сдвиг интервала.
18. [@@] — сдвиг интервала в абсолютное время.
19. [~] — растяжение времени.
20. [~~] — растяжение времени до абсолютного коэффициента растяжения.

Приложение 2

Математические функции языка программирования Lisp/Niquist и их эквиваленты на языке программирования SAL

Lisp	SAL	Описание функции
(+ a b)	a + b	сложение
(– a b)	a — b	вычитание
(***** a b)	a ***** b	умножение
(**/** a b)	a **/** b	деление
(**truncate** *expr*)	**truncate** (*expr*)	округлить выражение до целого числа (нижняя целая часть числа)
(**float** *expr*)	**float** (*expr*)	целое выражение с плавающей точкой*
(**rem** a b …)	**rem** (a b…)	остаток списка чисел
(**min** a b …)	**min** (a b …)	минимум
(**max** a b …)	**max** (a b …)	максимум
(**abs** expr)	**abs** (expr)	абсолютное значение числа
(**random** n)	**random** (n)	случайное целое число от 1 до n – 1
(**rrandom**)	**rrandom** ()	случайная величина с плавающей точкой от 0 до 1
(**sin** expr)	**sin** (expr)	синус
(**cos** expr)	**cos** (expr)	косинус
(**tan** expr)	**tan** (expr)	тангенс
(**expt** expr)	**expt** (expr)	показатель степени (экспонент) (**a** в степени **b**)
(**sqrt** expr)	**sqrt** (expr)	квадратный корень
(< a b)	a < b	тест на **a** меньше, чем **b**
(<= a b)	a <= b	тест на **a** меньше или равно **b**
(> a b)	a > b	тест на **a** больше, чем **b**
(>= a b)	a >= b	тест на **a** больше или равно **b**
(= a b)	a = b	тест на равенство **a** и **b**
(/= a b)	a /= b	тест на неравенство **a** и **b**

* Прим.автора. В англоязычной научно-технической литературе дословно эти команды обозначают с «плавающей точкой», отделяет десятичные доли от целой части. В русскоязычной литературе — «плавающая запятая». Точка/запятая используются как разделители целого числа и его десятичных долей. Символы в языке программирования Nyquist (такие как, имена переменных и имена функций) не чувствительны к регистру.

ЛИТЕРАТУРА

1. Таран В.В. Проектирование дизайна аудиопродукции в программной среде Audacity® с применением языка Nyquist/ В.В. Таран// Современная наука: актуальные проблемы теории и практики. Серия: Естественные и технические науки.— 2019.— № 10.— С. 159–171. [ISSN2223–2966].
2. Таран В.В. Компьютерный аудиосинтез штатными средствами Audacity® с возможностью имитационного дизайн-моделирования на языке Nyquist/ В.В. Таран// Современная наука: актуальные проблемы теории и практики. Серия: Естественные и технические науки.— 2020.— № 1.— С. 115–129. [ISSN2223–2966].
3. Таран В.В. Компьютерная очистка аудиоматериала штатными средствами программы Audacity® (программно-ориентированный подход) / В.В. Таран// Современная наука: актуальные проблемы теории и практики. Серия: Естественные и технические науки.— 2020.— № 9.— С. 112–128. [ISSN2223–2966]. (DOI 10.37882/2223–2966.2020.09.37).
4. Таран В.В. Язык программирования Nyquist: настоящее время и перспективы его развития в области компьютерной аудиоинженерии и аудиоинформатики / В.В. Таран// Современная наука: актуальные проблемы теории и практики. Серия: Естественные и технические науки.— 2020.— № 4.— С. 135–153. [ISSN2223–2966]. (DOI 10.37882/2223–2966.2020.04.37).
5. Таран В.В. Актуальные проблемы развития аудиоинженерии в России и их философско-техническое обоснование /В.В. Таран// Материалы X международной научно-практической конференции Фундаментальная наука и технологии — перспективные разработки [Технические науки], North Charleston, USA.— 2016 г., Том 1, стр. 108–123.
6. Таран В.В. Сравнительный анализ качества передаваемой информации форматами MP3, OGG, AIFF, WMA для оптимального выбора трансляции в Интернете/В.В. Таран // Материалы XVII международной научно-практической конференции Академическая наука — проблемы и достижения [Технические науки], North Charleston, USA.— 2018 г., Том 1, стр.78–94 [ISBN: 978–1729559000].
7. Audacity® 2.1.3 Manual, web-version: https://wiki.audacityteam.org/wiki/Release_Notes_2.1.3; https://wiki.audacityteam.org/wiki/New_features_in_Audacity_2.1.3 [дата обращения к электронному ресурсу: 15.01.2021].

8. Audacity® 2.4.2 Manual, web-version: https://manual.audacityteam.org/ [дата обращения к электронному ресурсу: 15.01.2021].
9. Carla Schroder The book of Audacity®: record, edit, mix, and master with the free audio editor / No Starch Press, Inc.— 2011, p.359. (ISBN-10: 1–59327–270–7; ISBN-13: 978–1–59327–270–8).
10. Touretzky, David S. Common LISP: a gentle introduction to symbolic computation /Carnegie Mellon University///Copyright (c) 1990 by Symbolic Technology, Ltd.////Published by The Benjamin/Cummings Publishing Company, Inc.p.587 (ISBN0–8053–0492–4).
11. Seibel Peter. Practical COMMON LISP /APRESS — 2005, p.528 (ISBN1–59059–239–5).
12. Dannenberg R.B. Nyquist Reference Manual Version 3.16 // Carnegie Mellon University — School of Computer Science/ Pittsburgh, PA 15213, U.S.A. 2013–2020 WEB-version, URL: http://www.cs.cmu.edu/~rbd/doc/nyquist/ [дата обращения к электронному ресурсу: 15.01.2021].
13. Dannenberg R.B. Nyquist Reference Manual Version 3.15 // Carnegie Mellon University — School of Computer Science/ Pittsburgh, PA 15213, U.S.A. 11.08. 2018 p.276.
14. Dannenberg R.B. Nyquist Reference Manual Version 3.09 // Carnegie Mellon University — School of Computer Science/ Pittsburgh, PA 15213, U.S.A. 27.12. 2014 p.297.
15. Dannenberg R.B. Nyquist Reference Manual Version 2.36 // Carnegie Mellon University — School of Computer Science/ Pittsburgh, PA 15213, U.S.A. 05.03. 2007 p.205.

Журнал «Современная наука: актуальные проблемы теории и практики»

Российская академия наук