ПРОЕКТИРОВАНИЕ ДИЗАЙНА АУДИОПРОДУКЦИИ В ПРОГРАММНОЙ СРЕДЕ AUDACITY® С ПРИМЕНЕНИЕМ ЯЗЫКА NYQUIST

Таран Василий Васильевич
К.культурологии, АНО ВО «Московский международный университет», докторант ФГБУН «Всероссийский институт научной и технической информации РАН»
allscience@lenta.ru

Аннотация. В статье рассматриваются вопросы проектирования аудиопродукции в открытой программной среде Audacity®. Систематически излагаются подходы к решению общих и некоторых частных проблем, возникающих при дизайн-проектировании аудиоматериала объектно-ориентированным и программным способами. На основе авторской инженерной практики компьютерного проектирования аудиопродукции проведен технический анализ наиболее важных программных компонентов и модулей, предполагаемых для проведения имитационного дизайн-проектирования аудиоматериала. Отдельное внимание уделяется терминальным возможностям программы в целях более точного управления процессами аудиообработки. Представлены и разобраны конкретные примеры и сценарные возможности встроенного языка программирования Nyquist, служащие демонстрационной основой в рамках имитационного подхода к компьютерному дизайн-проектированию аудиопродукции.

Ключевые слова: проектирование дизайна аудиопродукции, дизайн-проектирование, технические этапы дизайн-проектирования, программная среда Audacity®, язык программирования Nyquist.

PROJECT CONCEPTION OF AUDIO PRODUCTS IN AUDACITY® SOFTWARE ENVIRONMENT USING NYQUIST LANGUAGE

V. Taran

Summary. The problems of designing audio products in the free Audacity® software environment are considered. The approaches to solution of common and some specific problems arising when design and engineering of audio content by object-oriented and software methods are stated systematically. Technical analysis of major software components and modules intended for carrying out the simulated computer-aided audio material's design is performed on the basis of the author's engineering practice in the area of computer design. For more precise management of audio process operations special attention is paid to terminal capabilities of the program. Case studies and scenario features of built-in Nyquist programming language serving as a demonstration basis for the simulation approach to computer-aided design of audio products are presented and analyzed.

Keywords: design engineering of audio products, design engineering, technical stages of design engineering, Audacity® software environment, Nyquist programming language.

В связи с появлением различных музыкальных жанров и стилей, а также активно развивающегося на интернет-платформе подкастинга, интернет-радиовещания и прочих областей так или иначе связанных с аудиопроизводством, становится актуальной проблема проектирования дизайна аудиопродукции. Информационно-коммуникационные технологии, передовые наработки компьютерных наук (в России и центральной Европе — Информатики)[1] и производные от них различные специализированные программные средства диктуют новые стандарты создания и обработки аудиокомпозиций. Трансформация старых музыкальных жанров, их стилевой синтез и последующая постобработка фонограммы рождают совершенно новую аудиореальность.

Рынок продукции мультимедиа становится всё более насыщенным, изобретаются всё более изящные способы и подходы к производству аудиопродукции. Среди общепризнанных коммерческих разработок начинает расти конкуренция в области обработки звука, к примеру, программа WaweLab[2] от корпорации Steinberg

[1] Для более точного представления о терминологическом аппарате в области компьютерных наук и ИКТ, рекомендуется ознакомиться со статьёй: «Таран В. В. К вопросу о разграничении базовых понятий в контексте современного развития наук об информации / В. В. Таран // Меди@льманах МГУ.— 2014.— № 4.— С. 18–25».

[2] WaveLab — программный продукт компании Steinberg Media Technologies GmbH является профессиональным цифровым аудиоредактором широкого спектра действий, направленных на обработку аудиоматериала. Реализация концепции программы была предложена Филиппом Готье (Philippe Goutier). Распространяется на коммерческой основе. Подробнее о программе WaveLab в целом (и конкретно версии 4), а также некоторые технические рекомендации по использованию программы изложены в одноименной статье по адресу: www.web.archive.org/web/20150608220458/http://www.soundonsound.com/sos/may02/articles/wavelab4.asp (дата обращения: 29.05.2019).

Рис. 1.Логотип программы Audacity®. Графическая идея и название программы принадлежит Доминику Маццони (Dominic Mazzoni) и является зарегистрированной торговой маркой под его именем.

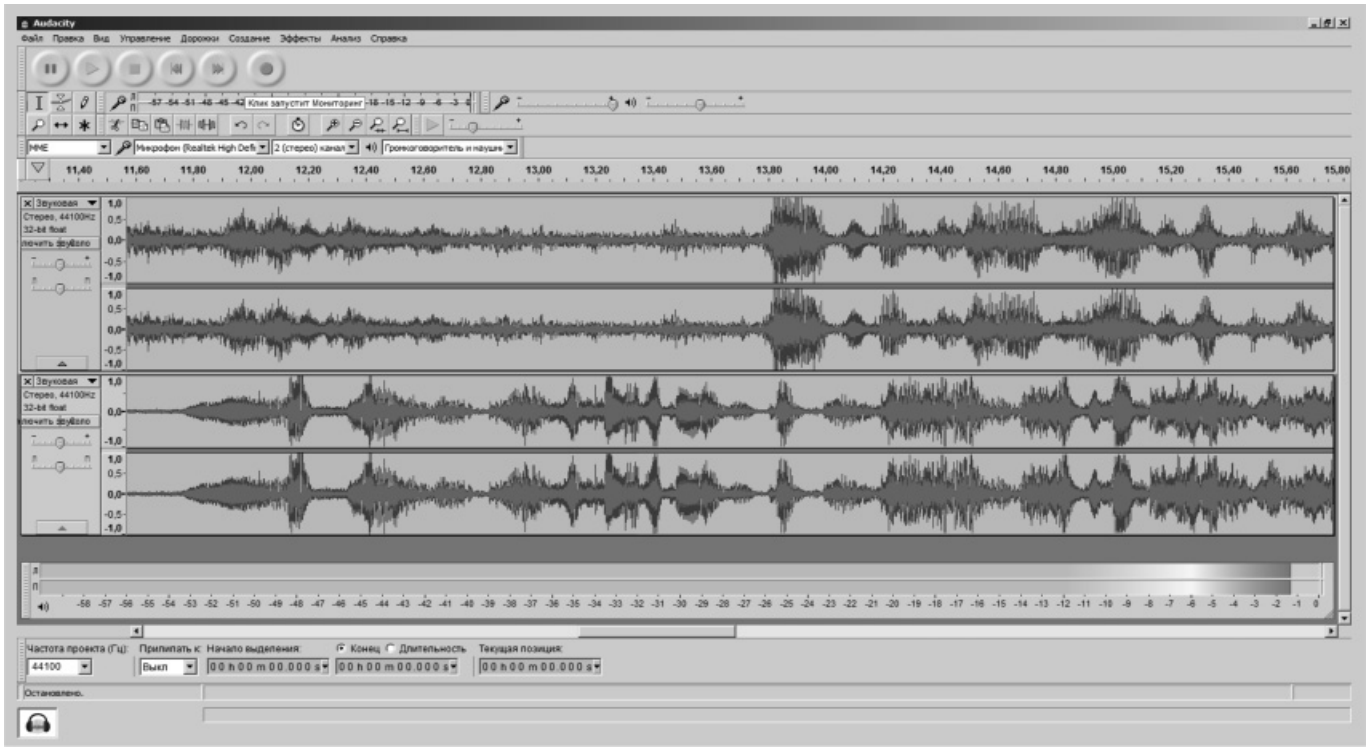

Рис. 2.Интерфейс программы Audacity® под управлением Microsoft Windows.

конкурирует с аналогичной коммерческой разработкой Soundforge от Sony[1]. Конечно, программы такого типа заточены под определенные конкретные задачи, к тому же у них довольно большая требовательность к ресурсам компьютера.

Все немного иначе на рынке открытого свободно распространяемого программного обеспечения[2]. Здесь имеются очень достойные разработки, призванные решать широкий спектр задач, связанных с проектированием аудиопродукции и её редактированием. Одной из таких наиболее комплексных разработок является программа Audacity®[3]. По нашему мнению, Audacity® – абсолютный лидер в области аудиопроизводства по нескольким причинам.

Во-первых, это открытое программное обеспечение, которое с виду может казаться весьма ограниченным в функциях. Но при детальном рассмотрении становится,

[1] Sound Forge — программный продукт компании Sony Creative Software, профессиональный цифровой редактор звука широкого профиля. В настоящее время является программным пакетом компании MAGIX Software. Распространяется на коммерческой основе. С более подробными характеристиками можно ознакомиться по адресу: www.magix.com/us/music/sound-forge/sound-forge-pro/ (дата обращения: 29.05.2019).

[2] Рынок свободно распространяемого программного обеспечения (условное понятие), характеризующее отдельный сектор мультимедийного рынка, на котором распространяется программная продукция открытого типа либо условно-открытого типа.

[3] Audacity® — свободно распространяемая многофункциональная компьютерная программа для обработки, проектирования и финализации аудиоматериала. Наименование «Audacity» является зарегистрированной торговой маркой, логотип, позиционирование и начертательные элементы принадлежат Доминику Маццони (англ.: Dominic Mazzoni). Развитие программного продукта осуществляется командой «Audacity» (англ.: Audacity Team)

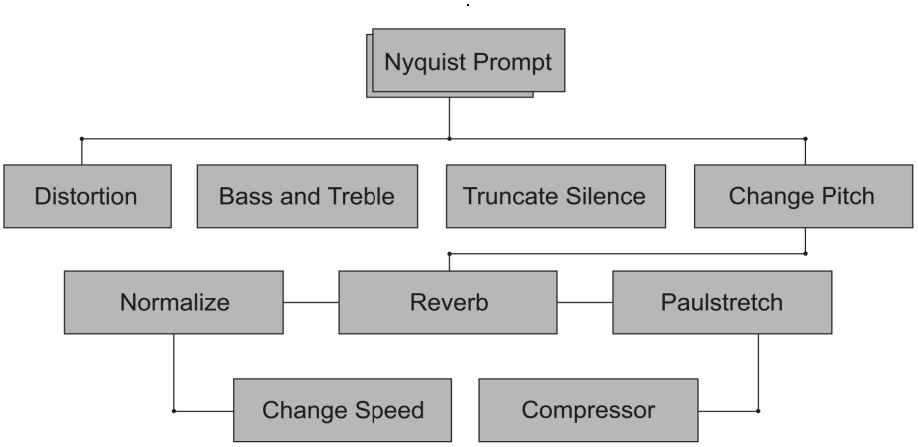

Рис. 3.Некоторые эффекты, управляемые командной оболочкой языка Nyquist, обрабатывающие аудиосигнал на одной секции.

очевидно, что это не так. Во-вторых, Audacity® — кроссплатформенное[1] программное обеспечение, поддерживающее различные операционные системы. В-третьих, по некоторым параметрам, к примеру, качество экспортируемой продукции Audacity® не то что не уступает, а превосходит аналогичные коммерческие продукты. В-четвертых, ядро программы Audacity® устроено таким образом, что его можно тонко подстроить под любые инженерные задачи и что самое важное — интегрировать с любыми приложениями. В-пятых, Audacity® имеет грамотно продуманный интерфейс именно с точки зрения работы над звуковым дизайном. Когда мы приводили первую причину лидерства Audacity® на рынке программного обеспечения в области постобработки аудио, то мы имели в виду встроенную поддержку языка Nyquist[2].

С нашей точки зрения, поддержка данного языка при проектировании аудиопродукции является очень важным решением, а местами и незаменимым. Реализация языка Nyquist позволяет устранять некоторые нюансы, которые (обычными) штатными средствами бывает очень трудно обойти. К тому же имеют место быть случаи, когда аудиопроект нуждается в уникальном (эксклюзивном) дизайне, но, увы, встроенных функций и эффектов по обработке звука здесь уже становится недостаточно.

Nyquist может реализовывать несколько функций одновременно, и с его помощью, возможно, сконструировать некую виртуальную программную карту обработки звука[3]. Карта, в свою очередь, может быть разделена на секции и подсекции. В каждой секции могут быть спрятаны эффекты как линейные, так и нелинейные, а в подсекциях параметры настроек каждой из них. Также карта обработки звука предполагает наложение нескольких эффектов на одну дорожку, в то время, как штатная база программы не предполагает таких опций. Итак, перед нами программная карта, которая интерактивно связана с отдельными секциями. При проектировании отдельных категорий звуковых файлов, например джинглов, часто приходится работать одновременно с несколькими программными эффектами, общей целью которых может быть более точная подстройка тонового баланса между каналами, локальная компрессия, расширение стереопанарамы и т.д. Вполне естественно, что для звукооператора необходимым условием при проектировании аудиопродукта любой сложности является точность контроля над всеми звуковыми нюансами, и именно здесь необходим линейный контроль над аудиодорожками.

Линейная обработка даёт возможность одновременного проигрывания большого числа аудиофрагментов и каждого из них по отдельности с наложением на них различных эффектов. Программа Audacity® обладает определенным спектром штатных эффектов, которые могут быть успешно применены при проектировании аудиоматериала разной направленности и разной степени сложности[4].

[1] Кроссплатформенность программы выражается в её способности успешно адаптироваться к различным операционным системам, включая такие как MS Windows, Apple Mac, и GNU/Linux.

[2] Nyquist — специальный язык программирования, созданный с целью облегчить рутинные процедуры при выполнении обработки аудиоматериала, а также обеспечить оптимальное управление и контроль над различными штатными модулями обработки данных внутри самой программы.

[3] Карта обработки звука — графическая (программная) карта расположения различных модулей обработки аудиоматериала, взаимодействующих между собой и по отдельности, призванная обеспечить визуальное сопровождение процессов аудиообработки и маршрутизацию действий при обработке аудио.

[4] Эффекты перечислены в соответствии с поставленными в статье задачами по проектированию аудиоматериала на свободной программной платформе. Поэтому часть эффектов, которая не имеет отношения к данным вопросам либо имеет далекое к ним отношение, в статье не приводится.

Среди них: Amplify[1], Auto Duck[2], Bass and Treble[3], Change Pitch[4], Change Speed[5], Change Tempo[6], Classic Filters[7], Click Removal[8], Compressor[9], Distortion[10], Echo[11], Equalization[12], Fade (In)[13], Fade (Out)[14], Invert[15], Noise Reduction[16], Normalize[17], Nyquist Prompt[18], Paulstretch[19], Phaser[20], Repair[21], Repeat[22], Reverb[23], Reverse[24], Sliding Time Scale/Pitch-Shift[25], Truncate Silence[26], Wahwah[27]. Все перечисленные эффекты могут быть органично вписаны в процессы производства аудиопродукции на этапе сведения и мастеринга дорожек. Для того чтобы было более понятно, в какой частотной области может быть применен каждый из эффектов, предлагаем представить данные в табличном виде (таблица 1).

Проектирование аудиопродукции является сложным этапом на пути к успешному внедрению конечного результата (финальная версия аудиопродукта). Для полного понимания процесса проектирования аудиопродукта необходимо на примере аудиоматериала разобрать последовательность действий, предпринимаемых для успешного достижения цели.

Самым важным проект-процессом на первом этапе является подбор аудиоматериала. Как и любой другой продукт интеллектуальной деятельности, аудиоматериал может обладать авторскими правами. Поэтому самым элементарным примером независимой звуковой формы может считаться голос человека, записанный на микрофон, либо материал, собранный непосредственно звуко-

[1] Amplify — штатный модуль обработки аудиоданных, применяемый при процессах усиления общей амплитуды звуковой формы либо увеличения громкости на отдельных её участках. При обработке материала со смешанным содержимым может применяться как расширитель звукового давления. Функционирует по принципу подбора высоких частотных значений при увеличении мощности и объема звука, при понижении частотных значений (ослабляет) звуковую форму. Имеет возможность ввода собственных значений пиковой амплитуды, которые коррелируются со значениями изначально заданной громкости. Имеется автоматический контроль пиковых перегрузок. Усиление от –50,0дБ до 50,0дБ (пиковая амплитуда –51,9382дБ до 48,0618дБ).

[2] Auto Duck — эффект автоматического понижения амплитуды громкости. При проектировании часто применяется в области радиовещания, когда необходимо понизить громкость звуковой формы с целью наложения на неё дикторского голоса либо эффекта. Широко применяется в кинопроизводстве при озвучивании и дублировании аудиодорожки.

[3] Bass and Treble — эффект регулировки нижних частот, отвечающих за бас (Bass), в основном ниже 1000 Гц и высоких частот (Treble), контролирующих звонкость звуковой формы, выше 1000 Гц. Громкость регулирует значения +/-30 дБ.

[4] Change Pitch — эффект степени отклонения частоты и изменения звуковой формы, не затрагивающий области частот, отвечающие за темп аудиоформы и её скорость.

[5] Change Speed — изменение скорости воспроизведения аудиоформы, при замедлении скорости воспроизведения общий диапазон частот становится ниже, при увеличении скорости соответственно — выше.

[6] Change Tempo — эффект изменения темпа (общей продолжительности) аудиоматериала, не изменяющий степень отклонения частоты.

[7] Classic Filters — частотные фильтры, моделирующие аналоговый эффект в цифровой обработке аудиоматериала, а также служащие средствами контроля и выполнения некоторых измерительных функций.

[8] Click Removal — инженерный эффект, применяемый при обработке аудиозаписей, имеющих в спектральной составляющей щелчки, образуемые в результате статического электричества при перезаписи аудиоматериала с виниловых носителей.

[9] Compressor — специальный эффект, используемый при обработке аудиоматериала для частотного выравнивания общей аудиоформы. Компрессор уменьшает динамический диапазон аудиоформы с целью дальнейшего его усиления, при этом, не искажая частоты. По сути, выделяет сигнал (его спектр) и контролирует диапазон.

[10] Distortion — эффект для искажения аудиоформы. Алгоритм эффекта разрушает структуру аудиоформы и дает акустически-грязный эффект.

[11] Echo — эффект повторения фрагмента аудиоформы, задержка которого имеет фиксированный по времени интервал, определяемый степень повторяемости аудиофрагмента.

[12] Equalization — эффект-функция настройки нижних, средних и верхних частот в соответствии с тоновым аудиобалансом. Эффект позволяет управлять диапазоном частот, выравнивая каждый из них и приводя к соответствующим балансным значениям.

[13] Fade In — эффект плавного нарастания звука. Использует степень плавного повышения звукового давления, тем самым изменяя амплитуду подъема аудиоформы.

[14] Fade Out — эффект плавного снижения звука. Использует степень плавного снижения звукового давления, тем самым изменяя амплитуду спуска аудиоформы.

[15] Invert — неразрушающий эффект отражения аудиоформы за счет полярности.

[16] Noise Reduction — штатное средство подавления шума, понижает степень шума в полезном сигнале.

[17] Normalize — эффект, позволяющий нормализовать пиковую нагрузку в аудиоканале путем пересчета нижних, средних и высоких частот с возможной компенсацией баланса стереосигнала. В моносигнале происходит пересчет пиковых значений в сторону понижения либо увеличения плотности. Имеется возможность корректировки и точной подстройки отклонения аудиоформы (сигнала) от заданных центром значений от 0 … до 0,1… и т.д. (DC offset).

[18] Nyquist Prompt — терминальное управление программой Audacity с помощью поддерживаемого программой языка программирования Nyquist. Позволяет вводить команды Nyquist и задействовать программные функции для более гибкой и точной обработки аудиоматериала.

[19] Paulstretch — встроенный алгоритм, поддерживающий функцию изменения скорости аудиоматериала в не- зависимости от скорости воспроизведения.

[20] Phaser — эффект вращения фаз аудиоканала, комбинирует смещенные по фазе сигнал с оригинальным сигналом.

[21] Repair — функция позволяет редактировать аудиоматериал, извлекая из него небольшие частотные фрагменты, которые в последствии могут быть заменены близкими общечастотными значениями.

[22] Repeat — алгоритм повторения аудиофрагмента, создаваемый копии аудиофрагмента и распределяющий их в соответствии с заданным временем.

[23] Reverb — эффект реверберации, пересчитывает аудиосигнал с целью добавления искусственных задержек, создавая его копию и в соответствии с заданными значениями подмешивает её в оригинальный аудиоматериал.

[24] Reverse — функция инвертации аудиосигнала для достижения эффекта задом наперед.

[25] Sliding Time Scale / Pitch-Shift — специальный алгоритм, позволяющий изменять темп аудиоматериала и задавать промежуточную скорость воспроизведения.

[26] Truncate Silence — функция удаления нежелательных аудиофрагментов, которые идентифицируются алгоритмом по частотным характеристикам. Бывает полезна для обнаружения и удаления шумов между речевыми интервалами.

[27] Wahwah — дизайн-эффект моделирует различные акустические оттенки для инструментов и вокала.

Таблица 1. Аудиоэффекты, применяемые при проектировании аудиопродукции в зависимости от степени сложности.

Наименование эффекта	Область применения в аудиоинженерии	Амплитудно-Частотные характеристики Гц/дБ
Amplify	Сведение, аудиомастеринг, аудиоремастеринг, восстановление плотности звука	От 20 Гц до 20 000Гц
Auto Duck	Линейное и нелинейное сведение, компоновка аудиофайлов	От 20 Гц до 20 000Гц
Bass and Treble	аудиосведение, аудиомастеринг	От 20 Гц до +/ — 1000Гц
Change Pitch	Дизайн-аудиоформы, аудиосведение, аудиомастеринг	От 20 Гц до 20 000Гц
Change Speed	Дизайн-аудиоформы, аудиоремастеринг, аудиосведение	От 20 Гц до 20 000Гц
Change Tempo	Дизайн-аудиоформы, аудиосведение	От 20 Гц до 20 000Гц
Classic Filters	Анализ и измерение аудиоданных, аудиосведение	От 20 Гц до 20 000Гц
Click Removal	аудиоремастеринг	От 20 Гц до 20 000Гц
Compressor	аудиомастеринг, аудиосведение, Дизайн-аудиоформы	От 20 Гц до 20 000Гц
Distortion	Дизайн-аудиоформы	От 20 Гц до 20 000Гц
Echo	Дизайн-аудиоформы	От 20 Гц до 20 000Гц
Equalization	Дизайн-аудиоформы, аудиомастеринг, аудиоремастеринг, аудиосведение	От 20 Гц до 20 000Гц
Fade In	Дизайн-аудиоформы, аудиомастеринг, аудиоремастеринг, аудиосведение	От 20 Гц до 20 000Гц
Fade Out	Дизайн-аудиоформы, аудиомастеринг, аудиоремастеринг, аудиосведение	От 20 Гц до 20 000Гц
Invert	Дизайн-аудиоформы,	От 20 Гц до 20 000Гц
Noise Reduction	аудиоремастеринг	От 20 Гц до 20 000Гц
Normalize	Дизайн-аудиоформы, аудиомастеринг, аудиоремастеринг, аудиосведение	От 20 Гц до 20 000Гц
Nyquist Prompt	Дизайн-аудиоформы, проектирование аудиоматериала на языке программирования Nyquist, аудиомастеринг, аудиоремастеринг, аудиосведение	От 20 Гц до 20 000Гц
Paulstretch	Дизайн-аудиоформы,	От 20 Гц до 20 000Гц
Phaser	Дизайн-аудиоформы,	От 20 Гц до 20 000Гц
Repair	аудиоремастеринг,	От 20 Гц до 20 000Гц
Repeat	Дизайн-аудиоформы, аудиосведение	От 20 Гц до 20 000Гц
Reverb	Дизайн-аудиоформы, аудиосведение, аудиомастеринг, аудиоремастеринг	От 20 Гц до 20 000Гц
Reverse	Дизайн-аудиоформы, аудиоремастеринг	От 20 Гц до 20 000Гц
Sliding Time Scale / Pitch Shift	Дизайн-аудиоформы, аудиосведение	От 20 Гц до 20 000Гц
Truncate Silence	Аудиосведение, Дизайн-аудиоформы,	От 20 Гц до 20 000Гц
Wahwah	Дизайн-аудиоформы	От 20 Гц до 20 000Гц

оператором[1]. Аудиоматериал условно можно разделить на два вида: *монофонический материал* и *материал со смешанным содержимым*. К первому виду относится любой материал, имеющий всего один канал спектро-акустической формы, ко второму виду относится многоканальная спектро-акустическая форма. Сочетание разных спектро-акустических форм называется процессом сведения. Сведение в аудиопромысле — очень важный этап производства аудиопродукции. Сам процесс сведения часто применяется при подборе материала. Когда условия диктуют необходимость производства новой звуковой формы из уже имеющихся форм, при соединении отдельных частей аудиоформ образуется совершенно новая оригинальная аудиоформа. Поэтому подпроцессы сведения на данном этапе называются стыковочными процессами[2]. Сами стыковочные процессы бывают двух типов и в зависимости от ситуации могут быть применены при звукопроизводстве. Первый тип такого процесса называется «горизонтальная стыковка». Горизонтальная стыковка применяется при склеивании двух и более аудиофрагментов, находящихся горизонтально на линейно-монтажной секции со схожими спектральными характеристиками. Второй тип — вертикальная стыковка применяется при вертикальном сведении двух аудиоформ, находящихся в горизонтальном положении и состоящих в линейной зависимости друг от друга. Стыковочные процессы дают возможность точного сведения звука и получения из него нужного материала. Второй этап — это проект-процессы, направленные на дизайн-технологии, применимые на стадии постпроизводства. Основными модульными элементами дизайн-технологии в настоящее время являются акустические спецэффекты. Спецэффекты дают возможность спроецировать виртуальную акустическую картину, для формирования акустического образа, запоминаемого человеческим мозгом. Спецэффекты могут применяться к аудиоматериалу, имеющему разную частотную амплитуду и разный смысловой оттенок. В эпоху компьютерной аудиоинженерии спецэффекты получили особую популярность среди специалистов. Эффекты делятся на *коммерческие* и *некоммерческие*. В нашем случае это *некоммерческие* штатные эффекты, интегрированные в программную среду Audacity®. Области применения штатных эффектов перечислены в табл. Но как быть, когда ситуация требует нестандартных или более креативных решений. Ведь дизайн звука практически безграничен, а произведение, получаемое после компьютерной обработки, всегда должно быть оригинальным. В этом смысле Audacity® отличается от своих коммерческих и некоммерческих конкурентов. В её программной среде интегрирована поддержка специального технического языка Nyquist посредством транслятора специальной командной строки Prompt, которая для специалистов в области дизайн-технологий, компьютерной аудиоинженерии и программирования очень важна. Данная опция позволяет управлять процессами производства и проектирования аудиопродукта с ювелирной точностью. Терминальный режим позволяет вводить команды для управления аудиопроцессом, а также интерпретировать разработанные автором алгоритмы действий для создания собственного подключаемого программного приложения по обработке звука. Написание программы зависит, прежде всего, от знания конкретного аудиоматериала, его вида и типа, а также от знаний диалекта языка программирования Nyquist.

Периодически технический синтаксис языка Nyquist обновляется, поэтому важно применять соответствующую синтаксису версию компьютерной программы (во избежание ошибки передачи языком программирования характеристик нелинейной звуковой волны при трансляции в программу)[3]. Важнейшей функцией при проектировании дизайна аудиопродукции является функция псевдолинейного воспроизведения аудиоматериала при его обработке. На штатной панели встроенных эффектов эту функцию берёт на себя команда Preview (Предпросмотр). Однако при использовании встроенной опции при мониторинге пиковых перегрузок[4] эффекты постобработки обрабатываемого сигнала могут давать задержку. К сожалению, данный аспект (на практике) не разрешим — при использовании штанных эффектов, но проектирование алгоритма (конкретных действий по обработке аудиосигнала) под встроенную среду выполнения команд (Nyquist Prompt) даёт возможность избежать ошибок мониторинга. В подобных аспектах и заключается основная сложность проектирования любого аудиопродукта.

Nyquist является специализированным языком программирования, производным от языка Lisp, унаследовавшим синтаксис и диалектическую основу XLISP. Сам язык и его модифицированный интерпретатор разработаны американским исследователем в области компью-

[1] При проектировании аудиопродукции важно обращать внимание на авторские права, поэтому самым лучшим выходом из положения является собственная монофоническая либо стереофоническая запись, преобразованная со звукоснимателя.

[2] Стыковочный процесс это сочетание различных фрагментов аудиоматериала и подбор соответствующих спектро-частотных характеристик с целью образовать новую акустическую форму.

[3] В зависимости от применяемой версии синтаксиса языка программирования Nyquist может быть построена командная программа, позволяющая реализовывать пакетные командные функции, т.е. к примеру, связать несколько эффектов в один или распределить их по отдельности.

[4] Пиковые перегрузки — это нежелательные спектро-частотные характеристики, образующиеся при некорректной оцифровке аудиоматериала либо применения к нему эффекта компрессии, значения которого намного превышают установленный порог чувствительности. Также данный эффект может проявляться при частой смене частоты дискретизации обрабатываемого материала.

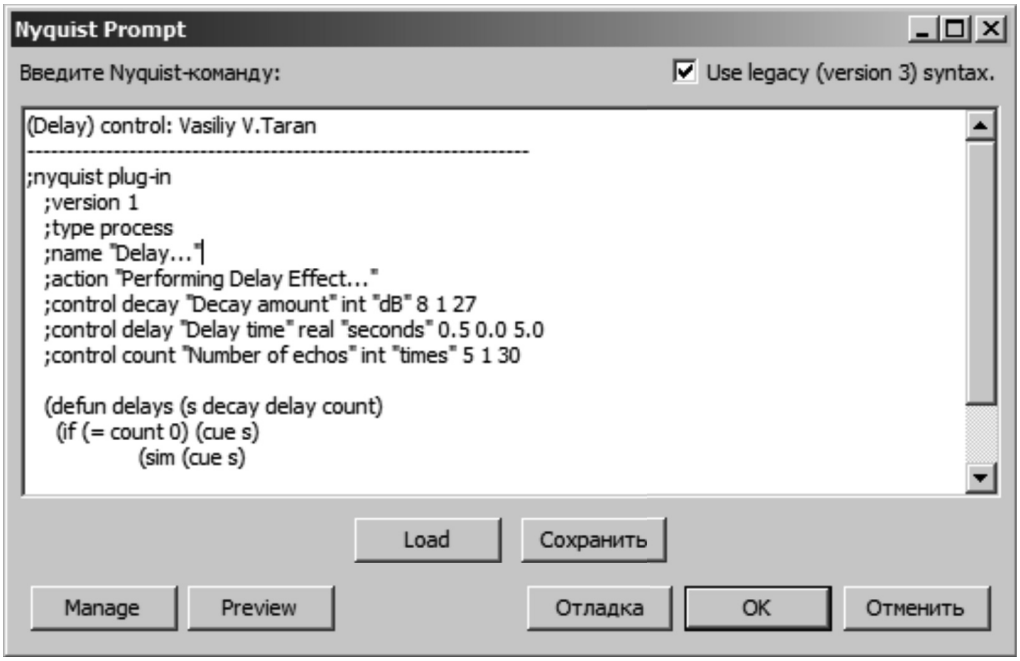

Рис. 4.Командная оболочка языка Nyquist в программной среде Audacity® с исполнением сценария задержки аудиосигнала

терных наук и аудиоинженерии Роджером Данненбергом (Roger Dannenberg) из школы компьютерных наук (на правах факультета) при университете Карнеги-Мелон (Питсбург, Пенсильвания) [1]. Свое название получил в честь американского инженера и исследователя в области технической коммуникации и связи, а также компьютерных наук Гарри Найквиста (Harry Nyquist)[1]. Благодаря богатому спектру команд язык может применяться для частотной обработки, генерации сигнала склейки, монтажа, анализа и синтеза аудиоматериала.

В процессе проектирования аудиопродукции бывает необходимо подключать уже существующие или создавать принципиально новые эффекты для обработки аудиоформы. Язык является кроссплатформенным и поддерживает операционную среду MS Windows, Linux, Mac OS [1,2]. Рассмотрим некоторые функции, способствующие более точному и креативному созданию некоторого вида контента. Не секрет, что среди профессионалов в области программного создания синтезированного музыкального материала большой популярностью пользуется MIDI[2]-технология. MIDI позволяет воспроизводить и синтезировать не физическую (запись, использующую аналого-цифровое преобразование), а виртуальную аудиоформу, т.е. технические сведения о последовательности электронных сигналов, соответствующих музыкальному шаблону. Широко используется программными виртуальными инструментами для достижения необходимого эффекта моделирования реальных музыкальных инструментов.

Синтез, получаемый в результате использования MIDI-технологии, позволяет использовать новые нестандартные результаты посредством моделирования аудиоформы и улучшает в целом дизайн-процессы [3,4].

[1] Гарри Найквист (англ. Harry Nyquist) — американский физик-акустик, инженер-электротехник, шведского происхождения, внёсший значительный вклад в теорию передачи и кодирования сигналов, является одним из участников разработки критерия устойчивости радиотехнических цепей с обратной связью (критерий Найквиста-Михайлова-Боде). С точки зрения компьютерной аудиоинженерии интерес представляют теоретические наработки Гарри Найквиста в области кодирования звука, результатом которых послужила теорема отсчётов, (в некоторых научных источниках: теорема выборки, теорема дискретизации, теорема Уиттекера-Найквиста-Котельникова-Шеннона), позволяющая работать с непрерывным по времени сигналом и выполнять процедуры его аналого-цифрового преобразования. По сути, работа с частотой дискретизации является важнейшим звеном в определении качества звучания аудиоматериала. Также особый вклад в развитие теоремы внесли известные учёные: К. Шеннон (Claude Elwood Shannon), Р. Хартли (Ralph Vinton Lyon Hartley).

[2] MIDI (Musical Instrument Digital Interface) — интерфейс и формат музыкальных инструментов на цифровой основе. Создан в 1982 году крупнейшими фирмами по производству музыкального оборудования, среди которых: Yamaha, Korg, Roland и другие. Интерфейс спроектирован с целью упростить задачи в области аранжировки звука классическими музыкальными инструментами посредством специальных данных, включающих в себя технические сведения о воспроизведении нотной таблицы, типа записи данных (моно/стерео), длительности звучания, громкости и прочих аудиохарактеристик. Также MIDI является аппаратным интерфейсом маршрутизации аудиосигнала, обладающим 16-ти канальным адаптером, для передачи независимых сигналов. Основной задачей MIDI как аппаратного интерфейса является связь различных цифровых музыкальных инструментов с вычислительной техникой.

С точки зрения языка Nyquist, последовательность действий для *чтения* MIDI-файла может быть выражена следующей формулировкой[1]:

```
(setf midi-file (open-binary "start.mid"))[2]
(seq-read-smf my-seq midi-file)
(close midi-file)
```

В то время как *запись* может выражаться так[3]

```
(setf midi-file (open-binary "copy.mid": direction: output))
(seq-write-smf my-seq midi-file)
(close midi-file)
```

Здесь open-binary — открытие бинарного файла start.mid — абстрактное название файла MIDI-потока. Последняя строка (close midi-file) указывает на закрытие MIDI-файла. Язык Nyquist и его прародитель XLISP, используя точный синтаксис, обладают достаточной базой для искусственной генерации сигнала с последующей ретрансляцией в MIDI. Рассмотрим пример подобной точной трансляции с сохранением данных в установленном диапазоне:

```
(defun midinote (seq time dur voice pitch vel)
(setf time (round (* time 1000)))
(setf dur (round (* dur 1000)))
(setf pitch (round pitch))
(setf vel (round vel))
(seq-insert-note seq time 0 (1+ voice) pitch dur vel))
```

Формулировка после добавленных замечаний

```
(defun test ()
(setf *seq* (seq-create))
(midinote *seq* 0.0 1.0 1 c4 100)
(midinote *seq* 1.0 0.5 1 d4 100)
(midinote *seq* 2.0 0.8 1 a4 100)
(setf seqfile (open-binary "test.mid": direction: output))
(seq-write-smf *seq* seqfile)
(close seqfile))
```

Символьная функция setf — устанавливает значение поля, seq — задает последовательность звуковых комбинаций. Если понадобится воспользоваться подключаемыми эффектами визуализации звуковой формы, возможно, обратиться к библиотеке openGL, это позволит визуализировать аудиоматериал в части его спектрального представления:

```
case "-openGL":
$pref:: Video:: displayDevice = "OpenGL";
$argUsed[%i]++;
```

Другим примером оформления звуковой формы, при дизайн-проектировании конечного аудиопродукта, может являться программный эффект скрипа иглы, возникающий на старых виниловых проигрывателях при появлении пыли, вызывающей эффект статического электричества, программно его можно записать следующим образом:

```
(defun ring (dur pch scl)
(let ((modstep1 (hz-to-step (* (step-to-hz pch) (sqrt 2.0))))
(modstep2 (hz-to-step (* (step-to-hz pch) (sqrt 11.0)))))
(stretch dur
(mult
(env 0.05 0.1 0.2 1 0.6 0.24 1)
(fmosc pch (mult
(pwl 0.07 1800 0.15 1000 0.4 680 0.8 240 1 100 1)
scl
(osc modstep1)
(osc modstep2)))))))
(play (ring 7.1 (hz-to-step 1) 1.2))
```

Здесь pwl[4](piece-wise linear) — является кусочно-линейной функцией, задает аппроксимированные значения (при контрольных точках) на единицу времени. sqrt[5] (square root) — вычисляет квадратный корень общего аудиофрагмента либо каждого по отдельности.

```
(play (sim
(scale 0.15 (at 2.9 (ring 7.1 (hz-to-step 1) 1.2)))
(scale 0.175 (at 4.9 (ring 5.1 (hz-to-step 2) 1.414)))
(scale 0.2 (at 6.9 (ring 3.1 (hz-to-step 4) 1.8)))))
```

[1] При проведении операций, связанных с программным проектированием аудиоматериала, использовались штатные рекомендации и формулировки, изложенные в Nyquist Reference Manual (Nyquist Version 2.36), существует обновленная база Nyquist Reference Manual 2013–2018 с соответствующей компиляцией языка (Nyquist Version 3.15). Некоторые формулировки, применяемые в версии языка (Nyquist 2.36), могут иметь более мобильное изложение в версии (Nyquist 3.15). Список подробных изменений см. в Nyquist Reference Manual Copyright 2013, 2014, 2015, 2016, 2017, 2018 by Roger B. Dannerberg.

[2] Некоторые образцы команд и примеры сценариев с авторскими дополнениями, используемые в статье, взяты с сайта-форума URL: www.audacity-forum.de/download/edgar/nyquist/nyquist-doc/examples/rbd/10-sequence-example.htm (дата обращения к электронному ресурсу: 12.06.2019).

[3] **Прим. автора**. Среда NyquistIDE (Nyquist 3.15) использует собственные выражения в синтаксисе Lisp. В принципе многие из примеров могут выполняться и на самом Lisp/XLISP. Однако, по мнению автора, выражения Nyquist упрощают обработку данных.

[4] pwl, pwl-list — может также использоваться для установления контроля аппроксимированных значений над частотными значениями верхнего и нижнего рядов.

[5] Можно использовать s-sqrt для обобщения функции квадратного корня, может применяться при вычислении квадратного корня каждого из отдельных аудиофрагментов для более точной выборки. Удобно использовать при многоканальном редактировании аудиоматериала, последовательно применяя данную функцию.

Оператор sim вызывает многократные (одновременные) поведения, используется, когда необходимо воспроизводить <(**play** (sim > спектро-частотные группы (в примере их три) единовременно. Также данный оператор даёт возможность возврата результирующей суммы, этот аспект упрощает работу с вложенными преобразованиями. Преобразования объединяются в соответствии с правилами вложенных выражений:

```
sim(cue(a-snd),
loud(6.0, cue(a-snd) @ 3))
```

Здесь sim устанавливает масштаб звуковой амплитуды со смещением второго входа a-snd. Использование преобразований с вложенными выражениями дают возможность учитывать внутриспектральные характеристики (изменение тембра, смещение частоты в виртуальном канале) при реализации абстрактной плотности децибел, выражающейся в громкости. Данная ситуация даёт возможность уйти от примитивного умножения a-snd на коэффициент масштабирования. Поведение является сигналом, реализующим громкость простым амплитудным масштабированием. По сути эффект, получаемый в результате приведённых манипуляций, эквивалентен умножению db-to-linear (6.0). Преобразования в одинаковом порядке применимы к группам поведений:

```
loud(6.0, sim(cue(a-snd) @ 0.0,
cue(a-snd) @ 0.7))
```

scale — оператор управления масштабированием. Позволяет управлять точностью амплитудного масштабирования и применяется для оптимизации громкости в каналах, использующих входные и выходные числовые значения (0.15,17,21). hz-to-step — является функцией возврата шага для частоты в Гц, устанавливаемого числовым значением (1,2,4).

Приведенные примеры показывают, насколько точным может быть подход к созданию дизайна звуковых форм, причём, границы творчества при таком подходе ничем не очерчиваются, и все зависит от фантазии автора и основывается на грамотном программном подходе. Что касается синтеза, используемого для дизайн-проектирования, то некоторые оригинальные идеи изложены Р. Даннербергом на международной компьютерной музыкальной конференции [5]. Ещё одной важной проблемой, возникающей при проектировании аудиопродукта программными средствами, является проблема синхронизации звуковых дорожек проекта. Часто приходится искусственно подгонять ту или иную аудиодорожку под стандарты фонограммы. Интерфейс программы для таких действий довольно хорошо продуман, однако, когда речь заходит о некоторых тонкостях синхронизации, штатных функций начинает не хватать. Примером может служить ситуация, когда необходимо подгонять многодорожечную запись под определенный шаблон[1]. К тому же контроль над временным потоком является важным моментом при синхронизации сигнала по заданному темпу [6]. Допустим, у нас есть 8 аудиодорожек для сведения, из них семь содержат инструментальные партии и одна вокальную. Каждая из этих дорожек имеет частоту дискретизации 44 100 Гц, с потоковой разрядностью 32 бит. Для изменения значений частоты дискретизации при её воспроизведении предусмотрено специальное выражение (стартовая отметка):

```
(play (sound-srate-abs 44100.0 (osc c4)))

Промежуточная отметка
(play (sound-srate-abs 48000.0 (osc c4)))

Конечная отметка
(play (sound-srate-abs 88200.0 (osc c4)))
```

Запустив алгоритм повышенной автокорреляции, при частотном анализе секции звуковой формы, определим, что средним пределом частоты аудиофрагмента будет 347 Гц, в то время как пиковая нагрузка будет определена в 400 Гц. Необходимо уровнять в частоте каждый из аудиофрагментов таким образом, чтобы каждая дорожка, содержащая, к примеру, отдельную инструментальную партию при среднем пределе в 347 Гц, ровнялась допустимой пиковой нагрузке в 400 Гц. Конечно, в разумных пределах этого эффекта можно достичь штатной компрессией, установив диктуемые задачей параметры, однако при такой ситуации нужно коммутировать все дорожки через одну мастер-секцию[2], что при дальнейших процедурах постобработки может негативно сказаться на общей акустической панораме. Подробное описание штатно поддерживаемых частот описано в таблице 2.

[1] Бывают случаи, когда при сведении требуется синхронизация большого количества аудиодорожек, в такой ситуации очень трудно уследить за точными интонационными нюансами как инструментальной партии, так и вокальной. В отличие от противоположной ситуации, когда сводится в единое целое уже заранее сведенная инструментальная партия с вокалом, представленный процесс осложняется уровнем контроля за аудиодорожками.

[2] В силу разных вычислительных мощностей персональных компьютеров и рабочих станций, при условии использования различных аудиокарт (даже премиум-класса) не рекомендуется применять программные эффекты реального времени на одной мастер-секции для достижения поставленных задач в области сведения аудиоматериала. Поскольку программная оболочка (интерфейсы) различных модулей реального времени может неадекватно реагировать на вводимые функции оператором, в результате программная оболочка может отказать и проект ждет зависание. Чтобы избежать подобных ошибок, лучше применять эффекты реального времени поочередно и в зависимости от творческих задач, стоящих перед аудиоинженером. К сожалению, в Audacity® штатный иетерфейс микшерного пульта не поддерживает цепочное наложение аудиоэффектов на мастер секции, поэтому необходимость маршрутизации эффектов в этом случае очень актуальна. Аудиоэффекты в Audacity® накладываются по отдельности на редактируемую дорожку, если их несколько, то на каждую по отдельности.

Таблица 2. Параметры частоты дискретизации, поддерживаемые Audacity®.

Частота дискретизации	Ед. измерения	Разрядность*	CI /Nyquist PL**
8000	Гц	16/24 PCM, 32 BF	sound-srate-abs
11025	Гц	16/24 PCM, 32 BF	sound-srate-abs
16000	Гц	16/24 PCM, 32 BF	sound-srate-abs
22050	Гц	16/24 PCM, 32 BF	sound-srate-abs
44100	Гц	16/24 PCM, 32 BF	sound-srate-abs
48000	Гц	16/24 PCM, 32 BF	sound-srate-abs
88200	Гц	16/24 PCM, 32 BF	sound-srate-abs
96000	Гц	16/24 PCM, 32 BF	sound-srate-abs
176400	Гц	16/24 PCM, 32 BF	sound-srate-abs
192000	Гц	16/24 PCM, 32 BF	sound-srate-abs
352800	Гц	16/24 PCM, 32 BF	sound-srate-abs
3844000	Гц	16/24 PCM, 32 BF	sound-srate-abs

*Примечание: все параметры частоты дискретизации, поддерживаемые проектом, могут быть интерпретированы в соответствии с разрядностями, например частота 88200=16/24 PCM, 32 BF.
** Примечание: см. раздел обозначения данной статьи.

Можно также и устанавливать эффект компрессии на каждую аудиодорожку по отдельности, но в таком случае, при малейших ошибках компрессии, будет очень сложно выполнять процедуры декомпрессии сведенных аудиодорожек. В таких условиях полезным окажется следующий программный код:

```
(gate test () Fragment_audio
(setf *eq* 347.400 hz (seq-create))
(gate ((50 .2 1)(35 .2 1)(20 .2 1)(10 .2 1)
(setf s ' ((50 .2 1)(35 .2 1)(20 .2 1)(10 .2 1)
((50 .2 1)(35 .2 1)(20 .2 1)(10 .2 1)))
(setf s ' ((10.2 1)(20 .2 1)(35 .2 1)(50 .2 1)
((10.2 1)(20 .2 1)(35 .2 1)(50 .2 1))))
```

Для того, чтобы правильно соотнести частотные значения *eq*, которые впоследствии будут отвечать временному (пороговому) интервалу и логическому элементу gate можно использовать символ инкремента[1] (incf symbol), в некоторых случаях символ инкремента может быть представлен как элемент множества (incf (aref myarray i)). Либо воспользоваться программной версией компрессора, которая задает более точные параметры, нежели штатный модуль с графическим интерфейсом (Compressor):

```
;(set-control-srate 100)
;(set-sound-srate 100)
;(setf xx (pwl 0 1 1 0 1.1 1 1.8 0 2 1 3 0 5))
;(setf xx (pwl 0 1 1 .2 1.1 1 1.8 .2 2 1 3 0 5))
;(setf yy (snd-follow xx 0.1 0.25 1.0 30))
;(setf db-factor (/ 1.0 (log 0.00001)))
```

[1] Если используется как переменная — приобретает вид (incf i).

Здесь set-control устанавливает порог выборки введенного значения, set-sound-**srate** устанавливает вызов частоты дискретизации при низком её пороге, db-factor задает плотность воспроизведения звука, влияя на громкость, pwl задает перечень контрольных точек (0,0) и осуществляет кусочно-линейную конвертацию. Вычисления на разных типах процессоров, производимые внутри оболочки языка Nyquist на обычном персональном компьютере, обладающем средней мощностью, дают более высокие результаты скорости обработки аудиоинформации, чем аналогичные действия, обрабатываемые интерфейсно-ориентированным способом [7].

И последней, заключительной проблемой является процедура финализации[2] спроектированного аудиопродукта для дальнейшего его распространения. Известно, что конечный аудиопродукт может звучать на различных воспроизводящих системах по разному. Хотя частотные характеристики могут удовлетворять нормам и принятым стандартам аудиообработки. Проблема эта кроется, в первую очередь, в том, что частотные характеристики колонок и вообще динамиков могут быть раз-

[2] **Прим. автора**. Финализация — это финальный процесс обработки аудиоматериала с целью усреднения спектро-частотного акустического баланса, для оптимального его воспроизведения на различных воспроизводящих электро-акустических устройствах. К таким устройствам относятся: колонки, акустические системы, системы репродукции звука, громкоговорители. Каждые из этих устройств имеют свою частотно-балансную специфику зависящую, в первую очередь, от объёма репродуктора (динамика) и соотношений размера колонки. К примеру, портативные колонки всегда будут подчёркивать верхний спектр частот (от 10 000 Гц до 20 000 Гц), в то время как стационарные (большие) колонки опционально будут выделять низкие частоты (от 20 Гц до 100 Гц). Искусство инженера финализации аудио материала заключается в подборе идеального баланса нижних, средних и высоких частот, сохраняя при этом начальную частоту дискретизации.

ными, к тому же огромное значение здесь будет иметь и способ производства воспроизводящего устройства и качественные характеристики, такие как материал, из которого изготовлена мембрана, проводники, используемые для возбуждения звуковой волны, и магнитная основа. Поэтому при программном проектировании не может быть универсального подхода к процессам финализации.

Однако некоторые акустические нюансы можно смоделировать программно. Речь идет о программной эмуляции различных типов динамиков, используемых в аудиопроизводстве. Это имитация настенных колонок, напольных колонок, сабвуферов, наушников, репродукторов и т.п. Конечно, формат статьи не позволяет сделать подробных выкладок со всеми частотными характеристиками различных воспроизводящих звук устройств и привести подробные примеры моделирования на языке Nyquist по каждому из них. Но можно создать некие искусственные условия программной среды, благодаря которым будет возможна *эмуляция* тех процессов, о которых сказано выше. Чтобы создать такие условия нужно воспользоваться нестандартными возможностями языка Nyquist.

Нестандартные возможности языка Nyquist заключаются в некоторых особых функциях, призванных осуществить расстановку опорных точек для экспликации выделенных частот, соответствующих установленным значениям[1]; тем самым виртуально смоделировав эффект присутствия для аудиоинженера, целью которого является уравнивание аудиочастот и отсечение нежелательных частотных значений при мониторинге аудиоформы для успешного её воспроизведения на разных устройствах. В этой ситуации предлагаем следующий вариант решения поставленной задачи:

```
`(+ (* (+ (* (+ (–Value) (*Value, r)) (expt, h 2))
–102.91325–102.91349
–96.89265–96.89289
–93.37082–96.89289
–90.87205–102.91349
–96.89265–102.91349
–102.91325–96.89289
–93.37082–93.37107
–90.87205–93.37107
–90.87205–93.37107
–93.37082–93.37107
–102.91325–96.89289
```

Ряды математических значений представляют полосы частот, варьирующиеся в определённых пределах, каждая из которых является блоком битовых данных, которая может быть объединена в один массив. Пример объединения данных приведён ниже. Заголовок первой цепочки, требующий установки значений Value выборки частотного диапазона, похож на начало сценария, использующегося для синтезирования аудиофрагмента, однако в данном случае он устанавливает значения отсчета тех частот, которые должны быть отброшены либо уравнены в своих математических значениях при заданном интервале выборки. Подстановка кусочно-линейной функции может помочь при контроле рядов (верхний и нижний ряд).

```
(pwl 0 8 0.1–6 0.76 3 0.5 0 1–3 8))))
(pwl. . … . …. …. . .))))*
```

Поставленная задача имеет ещё и альтернативный вариант решения. Задача решается по коду, разработанному Роджером Даненбергом, без установления точных значений (процедура похожа на графический эквалайзер см. рис. 4), либо на локальное усреднение частот[2] для сглаживания амплитудного спектра, когда частоты объединяются в один узел), а не как в предыдущем примере, когда каждая частота регулируется по отдельности (синтаксис SAL).

```
(defun smooth-amplitude (frame)
(let* ((len (length frame))
(lenm1 (1- len))
(lenm2 (1- lenm1))
(rslt (make-array (length frame))))
(setf (aref rslt 0) (+ (* 0.75 (aref frame 0))
(* 0.25 (aref frame 1))))
(dotimes (i lenm2)
(let* ((ip1 (1+ i))
(ip2 (1+ ip1)))
(setf (aref rslt ip1) (+ (* 0.25 (aref frame i))
(* 0.5 (aref frame ip1))
(* 0.25 (aref frame ip2))))))
(setf (aref rslt lenm1) (+ (* 0.25 (aref frame lenm2))
(* 0.75 (aref frame lenm1))))
rslt))
```

[1] К установленным значениям относятся частотные характеристики физических воспроизводящих устройств. Это микроколонки, наушники открытого и закрытого типов, акустические системы, домашние, автомобильные и профессиональные. Каждый из заявленных устройств имеет свой собственный диапазон воспроизведения. К примеру, микроколонки, как правило, воспроизводят высокие частоты, низкие частоты становятся менее слышны. Крупные колонки воспроизводят спектр частот аудиоматиреала равномерно. А Наушники (в зависимости от типа) могут подчеркивать либо низкие, либо высокие частоты.

[2] *Свободная pwl-секция для подстановки верхних и нижних рядов [.]. Двузначные значения рядов могут быть от [.] и до — [.].
Локальное усреднение частот может представлять интерес при сведении аудиодорожек, имеющих различные типы канального воспроизведения моно/стерео, для расширения спектральных пиковых значений моноканала (например, вокал) для более оптимального представления формант. Фиксация частотного фильтра предполагает значения (0.25.5, 0.25).

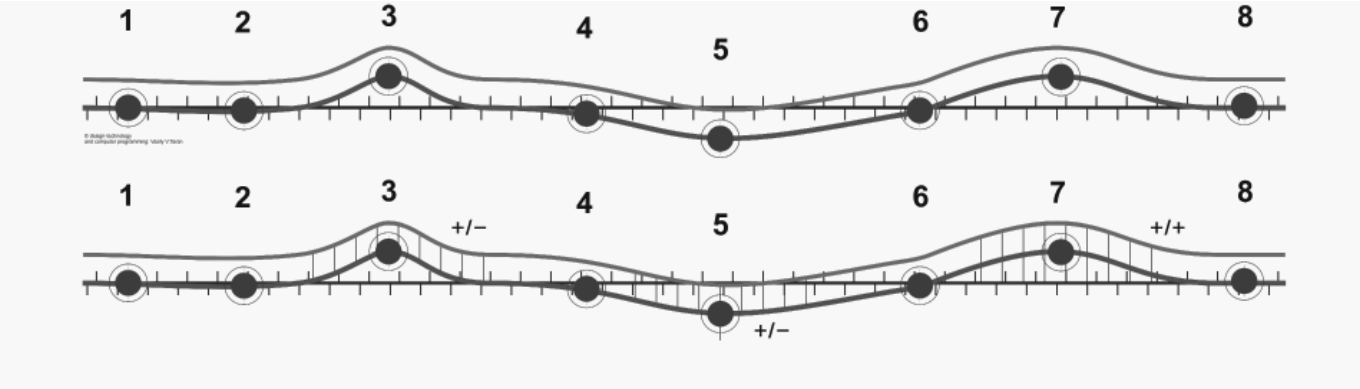

Рис. 5.Пример визуального контроля над управлением узловыми связками частот амплитудного спектра.

Рис. 5.иллюстрирует возможность связки частот, объединённых в узлы, упрощающие контроль управления над амплитудным спектром. При необходимости введения энного элемента множества (для разделения ветки потока) предусмотрена символьная функция expr **n** и трактовка записи, используемая синтаксис Lisp, примет следующий вид (aref expr n). Запись, используемая SAL-синтаксис (как в примере выше), будет иметь несколько другой вид setf (aref expr n)/(setf (aref rslt[1] …). Алгоритм, описанный на Nyquist с расширением Lisp, также помогает отсечь низкие частоты на общем (фронтальном) канале и на моно канале:

```
;; NYQ: LOWPASS2 — обрабатывает один канал
(defun nyq: lowpass2 (x hz q)
(if (or (> hz (* 0.5 (snd-srate x)))
(< hz 0))
(error "cutoff frequency out of range" hz))
(let* ((w (* 2.0 Pi (/ hz (snd-srate x))))
(cw (cos w))
(sw (sin w))
(alpha (* sw (sinh (/ 0.5 q))))
(a0 (+ 1.0 alpha))
(a1 (* –2.0 cw))
(a2 (– 1.0 alpha))
(b1 (– 1.0 cw))
(b0 (* 0.5 b1))
(b2 b0))
(nyq: biquad-m x b0 b1 b2 a0 a1 a2)))
```

При бинарном отображении схема имеет следующий вид

Фрагмент воспроизводимого файла с уравненными частотами без привязки к данным

```
0000C248 72 44 6C50 74 53 CD E6 D7 7B0B2A 01 00 00 00
0000C258 E5 58 51 00 BC B4 4D00 00 CA 0A 00 3B03 D9 5F
0000C268 B2 66 4D00 00 CE00 00 8D8F 28
```

Фрагмент воспроизводимого файла с уравненными частотами с привязкой к данным

```
0000C248 72 44 6C50 74 53 CD E6 D7 7B0B2A 01 00 00 00 rDlPtSHжЧ{•*••••
0000C258 E5 58 51 00 BC B4 4D00 00 CA 0A 00 3B03 D9 5F eXQ•jґM••K••;•Щ_
0000C268 B2 66 4D00 00 CE00 00 8D8F 28
```

Стоит отметить, что нечто похожее достигается эффектом компрессии, состоящей из разных частотных диапазонов. Классический вариант такой компрессии представляется как независимое компрессирование нижних, средних и верхних частот. Для оптимизации таких процессов, а также для манипуляции с выделенными диапазонами (включая их разграничение), предусмотрена специальная библиотека bandfx.lsp она упрощает действия по налаживанию частотных характеристик.

Проведенный анализ возможностей программной среды Audacity® показал её функциональность и пригодность для проектирования аудиопродукции для коммерческого и некоммерческого постпроизводства. Вопросы проектирования аудиопродукции в условиях бурного развития мультимедийного рынка являются особо актуальными. Большинство представленных на рынке систем обработки аудиоконтента как свободно распространяемых, так и коммерческих обладают ограниченным набором штатных функций для решения задач проектирования аудиоматериала.

[1] **Прим. автора**. rslt является глобальной переменной языка Nyquist, поддерживающего синтаксис Lisp. В правилах Lisp глобальная переменная заключается в *_* при возврате функции больше чем одного значения следует процедура помещения *rslt* в список дополнительных значений. В ветке XLISP — *RSLT*. Операция напоминает средство многократного возврата значений на языке Common Lisp.

Система Audacity® обладает профессиональными возможностями для проектирования аудиоконтента. Перечисленные в статье штатные средства и встроенные модули позволяют решать огромное количество задач, возникающих при аудиопроектировании. Для повышения эффективности обработки аудиоматериала и для решения узкопоставленных задач введена терминальная оболочка, позволяющая программисту-проектировщику вводить предусмотренные языком Nyquist команды, а также составлять микропрограммы и исполнять их в программной среде Audacity®.

В статье также показаны некоторые примеры имитационного дизайн-проектирования, программно моделирующие разные акустические эффекты и некоторые физические условия с целью создать качественную акустическую основу для дизайн-проектирования.

Данная статья позволяет по новому взглянуть на возможности аудиоредактора Audacity®, а проведенный анализ и демонстрация некоторых примеров использования программной основы для моделирования различных ситуаций, возникающих при обработке аудиопродукции, будут полезны программистам и аудиоинженерам при проведении инженерных работ, связанных с записью и корректировкой аудиоматериала, а также его дизайн-проектированием.

Обозначения

[Audacity®] — Зарегистрированная торговая марка (бренд — принадлежащий Доминику Маццони, англ.: Dominic Mazzoni) на рынке свободно распространяемого программного обеспечения, в целях уважения к авторским наработкам в статье упоминается с верхним регистром.

[+/–] — балансное соотношение узла *верхних частот* поднимает огибающую в соответствии с уровнем децибелов.

[+/+] — балансное соотношение узла *верхних и нижних частот* поднимает огибающую в соответствии с уровнем децибелов.

[*] — оператор «*умножения*» (включающий звуки).

[.lsp] — расширение файла данных, записанных на языке программирования Lisp. По тексту статьи можно встретить разное написание языка Lisp/LISP. Обусловлено это историческими традициями развития языка, старое написание (преимущественно использующееся с середины 60-х годов прошлого века) LISP, а в современной практике программирования Lisp. Таким образом, допустимым являются два написания: Lisp/LISP=**Lis**t **P**rocessing/ **LIS**t **P**rocessing.

[NYQ/NQ] — допустимое сокращение языка программирования Nyquist. Может применяться во внешних подключаемых источниках обработки аудиоданных.

[.NY] — расширение файла данных, записанных на языке программирования Nyquist.

[CE /Nyquist^{PL}**]** — искусственно введённое понятие (для разграничения функций и операторов) в целях упрощения обозначений элементов контроля. **C**ontrol **I**tems /**N**yquist — **P**rogramming **L**anguage, по тексту может обозначаться как **CINPL**.

[<>] — условные угловые скобки для отделения некоторых функций и операторов языка Nyquist во избежание путаницы с другими печатными символами.

[+]– оператор «*сложения*» (включающий звуки).

[@]– оператор «*сдвига фазы*» времени.

[>]– оператор «*больше чем*».

[<]– оператор «*меньше чем*».

[–]– оператор «*вычитания*» (включающий звуки).

ЛИТЕРАТУРА

1. 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
2. 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
3. Dannenberg R.B. «The Implementation of Nyquist, a Sound Synthesis Language», Computer Music Journal, 21(3) (Fall 1997), pp. 71–82.
4. Dannenberg R.B. «Machine Tongues XIX: Nyquist, a Language for Composition and Sound Synthesis», Computer Music Journal, 1997, 21(3), pp. 50–60.
5. Dannenberg R.B. «The Implementation of Nyquist, A Sound Synthesis Language», in Proceedings of the 1993 International Computer Music Conference, International Computer Music Association, (September 1993), pp. 168–171.
6. Dannenberg R.B. «Time-Flow Concepts and Architectures For Music and Media Synchronization», in Proceedings of the 43rd International Computer Music Conference, International Computer Music Association, 2017, pp.104–109.
7. Dannenberg R., Thompson N. Real-Time Software Synthesis on Superscalar Architectures / Computer Music Journal, 21 (3), (Fall 1997), pp. 83–94.

Журнал «Современная наука: актуальные проблемы теории и практики»