DOI 10.37882/2223–2966.2020.04.37

ЯЗЫК ПРОГРАММИРОВАНИЯ NYQUIST: НАСТОЯЩЕЕ ВРЕМЯ И ПЕРСПЕКТИВЫ ЕГО РАЗВИТИЯ В ОБЛАСТИ КОМПЬЮТЕРНОЙ АУДИОИНЖЕНЕРИИ И АУДИОИНФОРМАТИКИ

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

Аннотация. В статье рассматривается общая теоретико-прикладная проблематика в области компьютерного программирования, связанная с разработкой утилитарных программ, микропрограмм и аудиоприложений на языке Nyquist, направленных на повышение эффективности обработки адиоматериала различной степени сложности в контексте развития современной аудиоинженерии и аудиоинформатики. Выделяются основные технические проблемы настоящего времени, препятствующие качественной обработке аудиоматериала программными средствами. Демонстрируются современные технические и вычислительные возможности языка Nyquist. Анализируется его собственная интегрированная среда разработки NyquistIDE. Обосновываются перспективы развития языка Nyquist как базисного языка, способствующего более эффективной обработке аудиоданных при проведении процедур, связанных с инженерно-технической и научно-исследовательской деятельностью. В заключении (на основе анализируемого научного материала) кратко систематизированы основные выводы проведённого исследования и сформулированы предложения по использованию данного технического средства.

Ключевые слова: язык программирования Nyquist, интегрированная среда разработки NyquistIDE, Lisp-ориентированный язык, функциональное программирование, аудиосигнал, компьютерная обработка аудиоматериала, компьютерная аудиоинженерия, аудиоинформатика.

NYQUIST PROGRAMMING LANGUAGE: CURRENT SITUATION AND PROSPECTS OF ITS DEVELOPMENT IN THE AREA OF COMPUTER AUDIO ENGINEERING AND AUDIO INFORMATICS

V. Taran

Summary. The paper deals with general theoretical and applied problems in the area of computer programming related to the development of utility programs, micro programs and audio applications written in Nyquist programming language and aimed at the efficiency improving when audio material of various complexity being processed in context of current development of audio engineering and audio informatics. Key modern technological problems hindering qualitative processing of audio material by program means are highlighted. Current technical and computational capabilities of Nyquist programming language are demonstrated. Its own NyquistIDE integrated development environment is analyzed. The future developments of Nyquist programming language as basic source contributing to more efficient processing of audio data during the procedure related to engineering and research activities are justified. In conclusion (based on the analyzed scientific material), the main findings of the study are briefly systematized and suggestions for the use of this technical tool are formulated.

Keywords: Nyquist programming language, NyquistIDE integrated development environment, Lisp oriented language, functional programming, audio signal, computer processing of audio material, computer audio engineering, computer audio informatics.

Современная аудиоинженерия, преимущественно базирующаяся на новых и перспективных теоретических достижениях в сфере компьютерных наук, а также их прикладной апробации, реализующейся через информационно-коммуникационные технологии (ИКТ), пестрят довольно внушительными передовыми наработками, созданными, в первую очередь, для упрощения процессов и различных прикладных процедур обработки аудиоданных. Ключевым звеном здесь является повышение эффективности обработки аудиосигнала, ставшее возможным благодаря увеличению вычислительных мощностей на уровне аппаратной инфраструктуры компьютера и на уровне интеллектуальной инфраструктуры (среда выполнения действий, языки программирования, искусственный интеллект), являющейся по своей сути базисной надстройкой, бе-

рущей на себя ответственность за контроль над управлением[1], а местами и (автоматизированному управлению) отдельными действиями либо группами действий, требующих адекватного принятия решений. Особым фактором повышения эффективности указанных выше процессов являются средства технической коммуникации[2] человек — компьютер и компьютер — компьютер, призванные обеспечить наиболее быстрый отклик при проведении монтажных работ над аудиоматериалом. Конечно, существует довольно обширный перечень языков программирования способный удовлетворить требования проектировщиков и программистов для наиболее востребованных отраслей народного хозяйства и промышленности в целом. Но когда речь заходит об отдельных (специализированных) областях исследования, коим является звук — дело уже обстоит сложнее. На сегодняшний день проектирование сложных программ по обработке звука и отдельных подключаемых модулей (плагинов), работающих в зависимости от типа программной микросреды (VST[3], RTAS[4] и т.п.), ведётся преимущественно на языках программирования высокого уровня, но существуют отдельные моменты, когда при проектировании необходимо совместить знания специалиста в области прикладных аудиотехнологий (звукорежиссер, оператор нелинейного аудиомонтажа, ассистент по звуку и т.д.) со знаниями и навыками специалиста в области программирования (прикладного программиста, системного программиста, инженера-технолога и т.д.) для более эффективного достижения поставленной при проектировании цели. На данном этапе возникает проблема интеграции знаний и накопленного опыта с прикладными навыками программирования. Однако на последующем этапе становится очевидно, что определенные технические нюансы в понимании процессов обработки аудиоматериала (при проектировании компьютерной программы или специализированного приложения) требуют чётких определений и уточнений, которые не всегда бывают понятны разработчику-программисту, и вот здесь возникает проблема технической коммуникации[5], обязывающей специалистов из разных областей научно-технических знаний качественно взаимодействовать между собой. Неправильное понимание сути вопроса впоследствии отражается на качестве проектируемой разработки и выливается, в свою очередь, в следующие негативные факторы: недоработка интерфейса проектируемой программы, неадекватное представление отдельных модулей интерфейса программы, несоответствие изложенных аудиоспециалистом требований, заложенных в алгоритм программы. Чтобы избавиться от тех проблем, которые мы перечислили во вводной части нашей статьи, многие из специалистов в интересах, которых лежит развитие компьютерной аудиоинженерии, аудиоинформатики и аудиоиндустрии в целом предпринимают попытки создания так называемых адаптивных языков программирования[6] с широким набором функций. Такие языки отвечают веяниям времени и служат интегратором между теорией и практикой компьютерного программирования. Такое положение дел также способ-

[1] ***Прим. автора.*** Контроль над управлением процессами обработки звука, является одной из самых важных и сложных процедур при проведении монтажных работ, направленных на создание нового аудиоматериала и улучшение его качества. Полагаясь на результаты акустического анализа компьютерных систем обработки звука, звукорежиссёр (оператор компьютерного аудиомонтажа) не всегда принимает решение по удалению нежелательных артефактов из аудиокомпозиции. Причиной этому служит перцепционное восприятие акустических нюансов человеком. Даже если машина (компьютер) руководствуясь точными алгоритмами поведения и математическими расчётами, показывает нежелательность присутствия абстрактного фрагмента частоты или пика, решение о его устранении принимает звукорежиссёр. Часто имеют место быть случаи, когда наличие акустического артефакта (по сути, мешающего восприятию полезного сигнала) наоборот является изюминкой аудиокомпозиции и его автоматическое удаление системами компьютерного мониторинга может изменить всю акустическую картину. Монтаж звука является крайне сложным ремеслом, и, к сожалению, компьютерные системы в данном случае пока остаются здесь бессильными (даже не смотря на интеллектуализацию языков программирования). Однако если говорить о цифровом звуке, то в процессах аналого-цифрового преобразования аудиосигнала, компьютерная интеллектуализация достигла больших высот, и является хорошим подспорьем для аудиоинженеров.

[2] ***Прим. автора.*** Под технической коммуникацией понимаются технические средства, способные интерпретировать и адаптировать вводимые команды и программные алгоритмические предписания в машинно-читаемый код. Такие продукты в информационно-коммуникационных технологиях и есть языки программирования, в соответствии с функционалом делимые на классы.

[3] VST (**V**irtual **S**tudio **T**echnology) — программная микросреда, позволяющая устанавливать межпрограммные связи с различными модулями обработки аудиосигнала посредством их подключения к различным аудиоредакторам. По сути, подключаемый модуль (плагин) является частью микросреды VST, что способствует его быстрому отклику в процессе выполнения необходимых процедур, направленных на формирование дизайна и улучшение качества аудиоматериала. В настоящее время поддержку и активное внедрение этой технологии осуществляет немецкая компания Steinberg.

[4] RTAS (**R**eal-**T**ime **A**udio**S**uite) — специализированная технология, выполняющая функции программной микросреды для аппаратно-программных студий семейства Pro Tools. Отличительной особенностью данной технологии (от других подобных технологий) выступает возможность использования напрямую вычислительных ресурсов процессора в обход DSP-карты, используемой в системах Pro Tools. Технология хорошо зарекомендовала себя в обработке аудиоданных повышенной сложности, благодаря хорошо адаптированной системе распределения аудиопотока, способствующей более эффективному решению многофункциональных задач. Технология разработана компанией Digidesign, однако в настоящее время поддерживается Avid Technology.

[5] ***Прим. автора.*** Здесь техническая коммуникация уже принимает гуманитарный характер, выражающийся в недопонимании специалистов разной направленности в процессе проектирования нового программного решения.

[6] ***Прим. автора.*** Адаптивные языки программирования (условное понятие), характеризующие направленность языка программирования в соответствии с той отраслью, в которой он будет успешно применим. Как правило, это узконаправленные машинные языки объектно-ориентированного типа, основные функции которых могут быть понятны и профильному специалисту в области программирования и профильному специалисту в соответствующей отрасли. В нашем случае это обработка аудиоданных, имеющих структуру различной сложности.

ствует более устойчивому развитию аудиоинформатики и компьютерной аудиоинженерии. И в связи с этим хотелось бы дать авторские определения данным научным направлениям[1]. Аудиоинформатика —наука о сборе, обработке, хранении и распространении аудиоданных различной степени сложности, изучающая общие закономерности звуковой информации и её технического преобразования. Современная аудиоинформатика базируется на передовых достижениях вычислительной техники, компьютерных наук и информационно-коммуникационных технологий, а также активно взаимодействует со смежными научными областями физической акустикой, кибернетикой, медициной и психологией (на уровне восприятия аудиоматериала)[2].

Компьютерная аудиоинженерия — научное направление и прикладная область деятельности, связанная с изучением специфики аудиоматериала, его акустических артефактов с целью повышения эффективности его обработки компьютерными аппаратно-программными средствами. Компьютерная аудиоинженерия пересекается с аудиоинформатикой, предметом изучения которой выступает звук в более расширительном понимании, но является более практико-ориентированной областью исследований, на основе которых создаются новые и усовершенствуются ныне существующие методы обработки аудиосигнала[3].

Одним из таких специалистов (… *в интересах которых лежит развитие компьютерной аудиоинженерии, аудиоинформатики и аудиоиндустрии*…), как было упомянуто выше, является американский учёный в области компьютерных наук и прикладной аудиоинженерии Роджер Данненберг[4] (Roger B. Dannenberg), работающий в школе компьютерных наук (School of Computer Science) при университете Карнеги-Меллон (Carnegie Mellon University), предпринявший попытку создания многофункционального языка программирования Nyquist, получившего своё название в честь известного американского исследователя шведского происхождения физика-акустика Гарри Найквиста[5] (Harry Nyquist) с целью автоматизации прикладных процессов обработки аудиоданных и повышения эффективности технической коммуникации между сотрудниками, задействованными в решении комплекса проблем, связанных с созданием и обработкой аудиоданных. Конечно, как и любой проект подобной сложности, требующий затрат на интеллектуальные ресурсы, в разное время поддерживался и дорабатывался разными специалистами[6]. Язык программирования

[1] ***Прим. автора.*** В связи с различными трактовками данных научных направлений, упоминающимися как в русскоязычной, так и в англоязычной научной литературе, и во избежание путаницы понятийного аппарата, автор считает необходимым уточнить определения понятий «аудиоинформатика» и «компьютерная аудиоинженерия». Авторские формулировки определений уточняют и дополняют традиционно сложившийся аппарат в области наук об информации и компьютерных наук в целом.

[2] По определению автора (Таран В. В., 2020 г.).

[3] По определению автора (Таран В. В., 2020 г.).

[4] Роджер Данненберг (Roger B. Dannenberg) — учёный-исследователь в области компьютерных наук школы компьютерных наук на правах факультета при университете Карнеги-Меллона. В 1977 году он получил степень бакалавра искусств в Университете Райса, затем в 1979 году магистра наук в Кейсовском университете Западного резервного района (также именуется как Западный резервный университет Кейза и Университет Кейс Вестерн резерв. Университет особенно известен исследованиями в областях электрохимии и электрохимической инженерии). И наконец, в 1982 году — степень доктора философии в Университете Карнеги-Меллона. Его научные интересы распространяются на разработку и практическое применение языков программирования, а также использование компьютерных технологий при генерации, управлении и сочинении компьютерной музыки. Автор и идейный разработчик высокоуровнего языка программирования Nyquist. Долгое время был соруководителем компьютерного проекта (с англ. PianoTutor Project), направленного на внедрение теории понимания музыки и технологии экспертных систем в музыкальное образование. Доктор Данненберг ведёт активную преподавательскую деятельность и является членом Международной Ассоциации Компьютерной Музыки и ряда престижных организаций: IEEE, Sigma Xi, Phi Beta Kappa, Phi Mu Alpha, ACM.

[5] Гарри Найквист (Harry Theodor Nyquist) — американский исследователь в области электротехники и акустической физики, внёсший весомый вклад в теорию технической коммуникации. Получив в 1917 году учёную степень доктора философии (в области физики) в Йельском университете, поступает на работу в качестве прикладного инженера в Лабораторию Белла, где вплотную начинает заниматься теоретическими исследованиями в области теплового шума и апробацией на практике полученных результатов. Вместе с этим Гарри Найквист активно ведёт работу по исследованиям в области электротехники. Вносит вклад в разработку теоремы отсчётов (Уиттекера-Найквиста-Котельникова-Шеннона), в прикладном понимании — теорему дискретизации, которая играет огромную роль в цифро-аналоговом и анлогово-цифровом преобразованиях. Занимается изучением стабильности (устойчивости) в усилителях обратной связи, факсимильной связи, телевидения и телеграфии. На основе полученных результатов формулирует критерий устойчивости, в дальнейшем получивший название критерий Найквиста-Михайлова-Боде. В результате в 1932 году увидела свет публикация «О стабильности усилителей обратной связи». Критерий стабильности (устойчивости), разработанный Найквистом в настоящее время является базисом во время проведения исследований и научных публикаций по теории управления обратной связью. Также следует отметить, что ранние наработки Найквиста (по определению требований для широкополосной передачи информации) положили начало исследованиям другого известного учёного Клода Элвуда Шеннона (Claude Elwood Shannon), который впоследствии разработал теорию информации. Вклад Найквиста в данную теорию заключается в определении количества независимых каналов, которые могут быть размещены в телеграфном канале в единицу времени, ограниченного двойной пропускной способностью канала. Найквист опубликовал свои основные результаты в статьях «Факторы, влияющие на скорость телеграфа» (1924 г.) («Certain factors affecting telegraph speed») и «Некоторые вопросы теории телеграфной передачи (1928 г.) («Certain topics in telegraph Transmission Theory»). Последняя работа в англоязычной научной литературе известна как теорема выборки Найквиста-Шеннона.

[6] Среди специалистов, внёсших значительный теоретико-прикладной вклад в развитие языка программирования Nyquist, его распространение и внедрение, в разное время, следует упомянуть следующие фамилии: Адам Хартман (Adam Hartman), Андреас Пфеннинг (Andreas Pfenning), Джо Ньюкамер (Joe Newcomer), Дэйв Моватт (Dave Mowatt), Доминик Маццони (Dominic Mazzoni), Клифф Мерсер (Cliff Mercer), Крис Чоу (Chris Tchou), Морган Грин (Morgan Green), Нин Ху (Ning Hu), Илай Брандт (Eli Brandt), Педро Х. Моралес (Pedro J. Morales), Эдуардо Рек Мирранда (Eduardo Reck Miranda), Энн Люис (Ann Lewis), Эрих Нойвирт (Erich Neuwirth), Филипп Ям (Philip Yam), Перри Кук (Perry Cook), Гэри Пол Скавоне (Gary Paul Scavone), Дэйв Борел (Dave Borel), Стивен Мэнгиат (Stephen Mangiat), Фил Лайт (Phil

Nyquist[1] создавался с учётом всевозможных вариантов, предвосхищающих развитие событий при проведении процедур по обработке аудиоматериала. И поэтому его по праву можно считать наиболее эффективным специализированным средством, заточенным под конкретные задачи в области исследований и обработки компьютерного звука[2]. Идейной основой языка программирования Nyquist выступают опорные языки: Canon[3] и Arctic[4][1].

Light), Рик Таубе (Rick Taube), Джон Чаунинг (John Chowning), Хорхе Састре (Jorge Sastre), Чен Цихенг (Chen Ziheng), Крис Йели (Chris Yealy), Дерек Ди-Соуза (Derek D'Souza), Дарен Макук (Daren Makuck), Майкл Ривера (Michael Rivera), Дмитрий Портной (Dmitry Portnoy), Азаракш Кейпур (Azarakhsh Keipour).

[1] Nyquist — специализированный язык программирования, созданный в целях обеспечения качественной обработки аудиосигнала, составления композиций и компьютерного аудиосинтеза. Может применяться для разработки микропрограмм и утилит, способствующих адекватному решению задач, возникающих при решении проблем с дизайн-проектированием и композицией аудиоформ. ***Прим. автора.*** В некоторых электронных источниках и узкоспециализированной научно-технической литературе может упоминаться как NyquistPL, NyqPL, NPL (**N**yquist **P**rogramming **L**anguage), сокращения и приведённые аббревиатуры данного языка программирования также используются для соответствующих комментариев в программном коде. Язык Nyquist вносит весомый вклад в развитие компьютерной аудиоинженерии, аудиоиндустрии и аудиоинформатики в целом. Файлы программного кода языка Nyquist имеют расширение — *.ny.

[2] ***Прим. автора.*** Nyquist как язык программирования очень гибок и позволяет реализовывать заданные ему алгоритмы довольно чётко и точно, особенно это касается тех задач где требуются ***утилитарные функции,*** например, растяжение аудиоматериала во времени (от 1мс до [*… *] —установленных значений). Учитывая историю развития этого языка программирования, и исходя из собственной авторской практики разработки микропрограмм и утилитарных сценариев на языке Nyquist, можно сделать вывод, что это очень перспективная разработка, способствующая развитию аудиоинженерных технологий и развивающая аудиоинформатику в целом.

[3] Canon — язык программирования, созданный в помощь композиторам, работающим с электронной музыкой с целью создания аннотированных примечаний, содержащих информацию для аппаратных синтезаторов и компьютерных программ по синтезу звука, а также их контроля и управления над ними. Хорошо востребован, как практическое решение в условиях компьютерной студии по обработке аудиоданных, реализующих технологию MIDI. Язык поддерживает декларативный стиль программирования и одновременно является нотацией для музыкального множества и языком программирования. Canon позволяет вести высокоуровневую нотацию для последовательности музыкальных событий и структур. Canon, также как и Nyquist, базируется на Lisp. Операторы преобразований данного языка корректируют параметры громкости и продолжительности звучания аудиоматериала.

[4] Arctic — язык программирования, созданный для уточнения прикладных процедур обработки (точная настройка синтеза аудиоматериала, сведение и премастеринг — мастеринг аудиокомпозиций) аудиоматериала и обеспечения точного сопровождения управляющих систем, работающих в реальном времени. В отличие от более традиционных языков, используемых для управления в реальном времени, Арктик (Arctic) является языком, не фиксирующим текущее состояние, в котором взаимоотношения между системами ввода-вывода и промежуточными условиям выражаются в виде операций с изменяющимися во времени функциями. Арктик позволяет дискретным событиям (или условиям) вызывать и изменять ответы асинхронно, но так как программы не предусматривают наличия баз данных, проблемы синхронизации значительно упрощаются. Более того, программы Арктик непоследовательны и время ответов системы записывается явно. Для программиста это исключает необходимость беспокоиться за выполнением последовательности, которая приносит много трудностей при программировании в реальном времени.

Язык Nyquist основывается на нескольких основных технологиях программирования: Fugue[5], SAL[6],

[5] Fugue — язык программирования, созданный для формирования музыкальных электронных композиций и синтеза звука. В качестве основной идеи данного языка выступает концепция совмещения функционального стиля программирования с удобочитаемыми функциями и элементами, направленными на повышение качества и упрощения процедур, связанных с синтезом аудиокомпозиций. Можно констатировать, что автору удалось достигнуть поставленной цели в соответствии с исторической ретроспективой. Основной новизной данного проекта по праву можно считать расширение функциональной основы языка за счёт внедрения в его структуру функций и управляющих операторов, способствующих упрощенному выражению алгоритмов посредством операторов языка при обработке аудиоматериала в условиях проведения стационарного компьютерного звукового синтеза и создания музыкальных сопровождений. Язык органично вписался в концепции программирования того времени, тогда как при традиционном программном подходе требовалось знание и использование различных языков для решения подобных задач. Основой для языка Fugue выступает LISP. Поэтому при работе со структурами аудиоданных различной степени сложности у программиста (знающего синтаксис LISP) не возникнет трудностей при написании сценариев и алгоритмов действий. Хочется также отметить, что Fugue уже тогда поддерживал функции «поведенческой абстракции», которые сильно упрощали выражение «временного поведения». ***Прим. автора.*** Язык Fugue был ориентирован на UNIX-системы, при этом давал возможность проектировать виртуальные инструменты посредством объединения функций, и в этом отношении был более эффективен (по сравнению с теми же Music V, cmusic или Csound) именно при составлении оркестровых партий. К примеру, имелась возможность вызова виртуального инструмента для генерации нового сигнала (аудиоформы), путём описания простого выражения. Таким образом, можно было объединять простые выражения в сложные системы для создания целостной композиции.

[6] SAL (**S**imple **A**lgorithmic **L**anguage) — алгоритмический язык программирования специального назначения, спроектированный в целях обеспечения удобства трансляции алгоритмических моделей в машинный код. Язык может быть полезен электронным композиторам, которые хотят реализовать свои замыслы на компьютерной программно-аппаратной основе. ***Прим. автора.*** SAL можно назвать Lisp-ориентированным языком, поскольку он сочетает в себе многие характерные черты Lisp. Не- смотря на теоретические воззрения и концепции в области компьютерного программирования, на практике SAL всё же отличается от Lisp. Основное отличие SAL от Lisp заключается в упрощении структуры и стиля изложения кода самого языка. Язык SAL не требует применения префиксной нотации и не использует цитат (кавычек) при блокировании оценки. В результате получился очень оригинальный качественный алгоритмический язык достаточно лёгкий для изучения и его применения в сфере аудиообработки. Команды SAL отличаются краткостью и простотой в отличие от аналогичных Lisp-команд и эквивалентных Lisp-выражений. Программируя на SAL, аудиоинженер или звукооператор не взаимодействует непосредственно с Lisp, и поэтому не нуждается в дополнительном обучении отдельных команд Lisp как Read /Eval/ Print loop (REPL) или как восстанавливаться после ошибок Lisp. Поскольку SAL анализируется, из его строкового представления имеется возможность более быстрого и полезного отчета о программных ошибках композитору по сравнению с возможностями Lisp-анализатора. Примечательно, что язык SAL в англоязычной научно-технической литературе может толковаться по-разному. Традиционно как SAL «**S**imple **A**lgorithmic **L**anguage/ Стандартный алгоритмический язык программирования» и расширительно как SAL «**S**ecretly **A**nother **L**isp/ Еще один скрытый Lisp». Разница в трактовках имеет некоторый исторический подтекст. Автором языка является Sal Martirano — известный и влиятельный композитор, много лет преподававший в качестве профессора музыкальной школы при колледже изобразительных и прикладных искусств в Иллинойском университете в Урбане-Шампейне UIUC (**U**niversity of **I**llinois at **U**rbana — **C**hampaign, College of Fine and Applied Arts, School of Music).

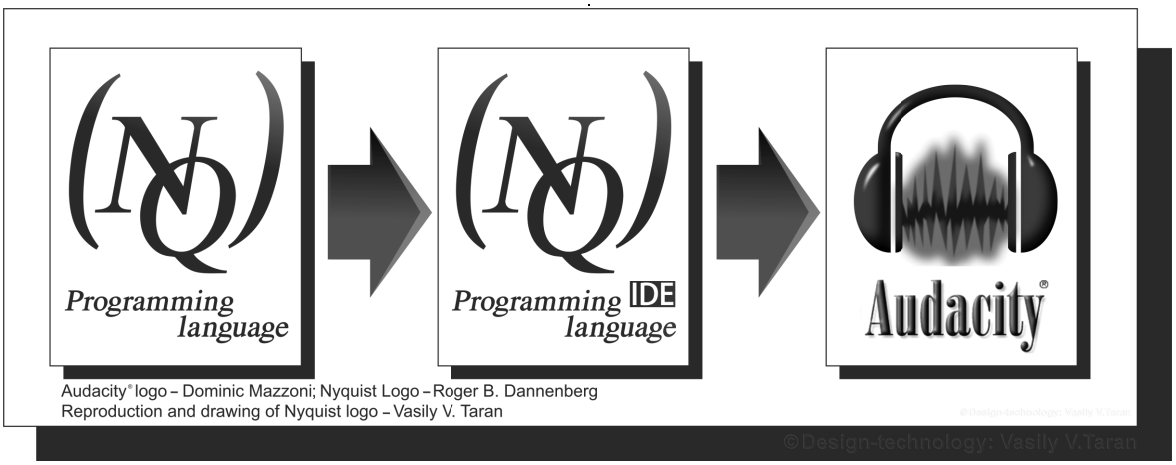

Рис. 1. Логотипы языка программирования Nyquist (его интегрированной среды разработки) и компьютерной программы обработки звука Audacity®, (стрелками показаны этапы интеграции технических средств) символизируют единый программный комплекс по обработке аудиоматериала*.

* Логотип Audacity® — является зарегистрированной торговой маркой и принадлежит Доминику Маццони (Dominic Mazzoni). Логотип языка программирования Nyquist — принадлежит Роджеру Даннебергу (Roger B. Dannenberg). В связи с тем, что в мировой научно-технической литературе данные логотипы практически не встречаются (либо встречаются в плохо читаемом виде), а программные продукты идентифицируемые ими хорошо зарекомендовали себя на рынке программного обеспечения в секторе обработки аудиоматериала, автор решил заново визуализировать логотипы данных компьютерно-технологических решений чтобы уточнить их идентификацию на рынке подобных программных продуктов.

Lisp[1], XLISP[2][1,2]. Официально язык Nyquist в его современном изложении демонстрирует стабильность, начиная с версии 2.36–7 и заканчивая версией 3.15. Однако его история насчитывает без малого 29 лет. Дело в том, что проектным его предшественником являлся язык Fugue версии 1.0, который уже в то время был способен манипулировать аудиосигналом, применяя поведенческую абстракцию [3]. Официально язык увидел свет в далёком 1991 году[3] в виду этого обстоятельства стоит уточнить, что над прообразом современного языка Nyquist совместно с Роджером Даннебергом работал и Крис Фрели (Chris Fraley) [4,5]. С тех пор Nyquist хорошо зарекомендовал себя как язык, способный производить простой и сложный аудиосинтез [6].

Сегодня Nyquist это динамически развивающийся кросплатформенный узкоспециализированный (готовый разрешать функционально задачи разного толку) язык программирования, поддерживающий функционально-ориентированный стиль[4] в области програм-

[1] Lisp (**LIS**t **P**rocessing language) — один из самых старых языков программирования, прочно закрепившийся на практике и по сей день. Отличительной чертой является ***полностью*** заключённая в скобки префиксная нотация. Язык Lisp получил широкое распространение (и хорошо себя зарекомендовал) среди систем искусственного интеллекта и компьютерной автоматизации. Программный код Lisp записывается в виде символьных выражений или скобочных списков. Название LISP происходит от «обработчик списков», связанные списки являются одним из основных структурных элементов Lisp, а исходный код Lisp состоит из отдельных списков. В конечном итоге Lisp-программы могут манипулировать исходным кодом как структурой данных, формируя макросистемы, которые позволяют программистам создавать новый синтаксис или новые доменные языки, встроенные в Lisp. Файлы программного кода на языке Lisp имеют расширение — *.lsp.

[2] XLISP — экспериментальный язык программирования, объединяющий некоторые особенности Common Lisp с объектно-ориентированной расширительной способностью. Он используется для экспериментов с объектно-ориентированным программированием на небольших компьютерах. Большинство функций Lisp встроено в XLISP. XLISP идентифицирует объекты 'Object' и 'Class' как простейшие. «Объект» ('Object') — это единственный класс, который не имеет суперкласса и, следовательно, служит корнем дерева иерархии классов. «Класс» ('Class') — это категория, для которой все остальные классы являются элементами (это единственный объект, который является примером самого себя).

[3] Поскольку по составу и своей структуре Fugue очень похож на современный язык Nyquist, с учётом доработок и некоторых изменений, он получил название — Nyquist.

[4] Функционально-ориентированный стиль (в программировании и теории компьютерных наук) — представляет собой парадигму программирования и характеризует собой стиль создания структуры и элементов компьютерных программ, который рассматривает вычисления как оценку математических функций и избегает изменения состояния и изменчивых данных. Часто именуется как «декларативная парадигма» (иногда декларативная модель) программирования, в которой программирование выполняется с помощью выражений или деклараций вместо операторов. В функциональном изложении кода выходное значение функции зависит только от её аргументов, поэтому вызов функции с одинаковым значением для

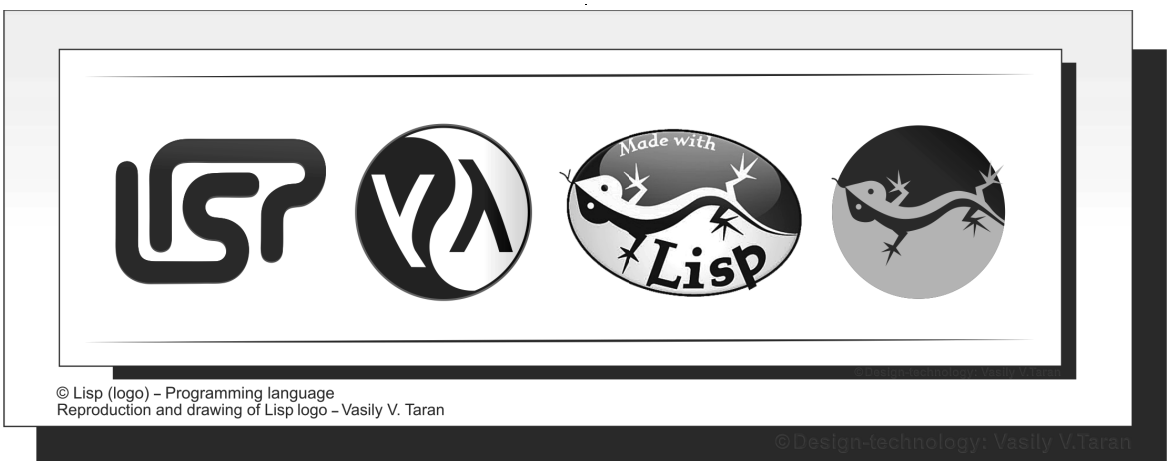

Рис. 2. Варианты графического представления языка программирования Lisp. Используются для идентификации программных решений, задействующих в своём коде Lisp. Поскольку Nyquist очень многое унаследовал от Lisp (диалект XLISP), то, по сути, он является Lisp-ориентированным языком.

мирования с развитой семантикой. Способен решать разные задачи на утилитарном (создание микросценариев для обработки аудиоматериала, разработка подключаемых утилит, адаптированных под выполнение задач средней сложности: учёт дефрагментации аудиодорожки, сбор спектро-частотной статистики, автоматизированная расстановка специальных меток-маркеров, с привязкой ко времени и частотным характеристикам и т.д.) и высоком уровнях (разработка полноценных подключаемых модулей для обработки аудиоматериала повышенной сложности)[1]. Язык Nyquist (в современном изложении)[2] может применяться в ситуациях, когда необходим нестандартный подход к решению проблем либо при невозможности разрешения абстрактной проблемы интерфейсно-ориентированным способом. Он способен выполнять следующий круг задач:

- Управлять аудиоконтентом в среде Audacity®.
- Осуществлять контроль и точность обработки аудиоматериала.
- Создавать композиции путём слияния аудиоформ.
- Программно моделировать ситуации, заданные звукооператором.
- Осуществлять компьютерный синтез звука.
- Выполнять анализ спектрочастотных характеристик.
- Реализовывать задачи по вводу-выводу аудиоматериала через Nyquist IDE.

При этом язык Nyquist интегрирует в себе возможности по работе с событиями и одновременно — с сигналами, а также поддерживает различные преобразования[3][7]. Язык Nyquist использует в своей работе интерпретатор Lisp и способен объединять простые и сложные выражения, образуя композиции, что предполагает его использования при обработке аудиоданных в режиме реального времени [8].

Кроссплатформенность языка Nyquist заключается в адаптивности к современным средам исполнения программного кода: Linux, Apple MacOS X, MS Windows NT, MS Windows 2000, MS Windows XP, MS Windows Vista. Исходя из собственного опыта автора, язык Nyquist также хорошо адаптируется и в средах: MS Windows 7, MS

аргумента всегда приводит к одному и тому же результату. В некоторой периодике употребление терминов «функциональное программирование» и «императивное программирование» носит схожий характер и даже отождествляется, однако это не совсем так. Основным отличием здесь служит то, что к аргументам функции состояние общей программы может влиять на результирующее значение функции. Устранение побочных эффектов, (то есть изменений состояния, не зависящих от входных данных функции), может облегчить понимание программы, что является одной из ключевых мотиваций для развития функционального программирования.

[1] Аудиоматериал повышенной сложности представляет собой аудиоданные большого объёма с разной степенью спектро-частотных характеристик. В качестве примера такого аудиоматериала может служить аудиозапись, которая имеет отличающуюся друг от друга амплитуду воспроизведения аудиосигнала на определённом участке аудиопотока. То есть, если запись длится 120 минут (и её средняя амплитуда 27 д.Б.), то из них, к примеру, есть выпадающие 24 минуты (средняя амплитуда которых 35 д.Б.— интервал «5»). Такой эффект часто наблюдается при записи различных потоковых аудиотрансляций в связи с микшированием разных аудиоканалов с разной степенью громкости. Наиболее часто такая картина наблюдается при записи потокового радиовещания. Несмотря на имеющийся в студии специальный компрессор, который на входе общего аудиосигнала выполняет его постоянную компрессию, рекламные блоки и отбивки могут превышать заданные нами абстрактные значения. В результате цифровая компрессия пересчитывает аудиосигнал с отличающейся амплитудой, а на сонограмме можно увидеть завышение громкости.

[2] ***Прим. автора.*** Имеется в виду последняя (текущая) версия языка Nyquist на 2018 год, версия 3.15.

[3] Преобразования помогают изменять продолжительность аудиофрагмента его время, темп и громкость.

Windows 8–8.1, MS Windows 10. Язык свободно совмещается с языками синтеза Csound[1], V[2], cmusic[3], это обстоятельство открывает возможности по использованию языка Nyquist, как вспомогательного элемента, при создании электронных аранжировок и синтезированных композиций [9]. Сам процесс создания аранжировок должен сопровождаться точностью алгоритма просчёта в классических (синтетических) языках программирования типа Music N, Cmusic или pcmusic, «электронный» музыкант проектирует инструменты, записывая специальные инструкции, определяющие взаимосвязи генераторов блоков и устанавливая таблицы поиска, содержащие формы сигналов для осцилляторов, огибающих кривых и других важных контроллеров. После того, как инструменты определены, музыкант должен составить список нот, чтобы играть на них. Каждая нота списка состоит из спецификаций времени *начала* и *длительности*, за которыми следует ряд значений параметров синтеза, требуемых инструментом, который будет её воспроизводить. Термин «нота» здесь относится к обобщенному звуковому событию любой сложности. Как инструменты, так и ноты обычно сохраняются в одном файле, называемом партитурой. Значения глобальных параметров синтеза, таких как частота дискретизации либо количество каналов, должны быть указаны в файле «Pcmusic.ini», который обычно должен находиться в текущем рабочем каталоге. Этот файл содержит информацию о конфигурации системы для pcmusic. Поэтому любителям рекомендуется не вносить серьезных изменений в «Pcmusic.init» файл, если они не полностью уверены в том, что они делают. И здесь хочется сказать, что Nyquist также требователен к каталогам и соблюдает чёткую иерархию в системе распределения обрабатываемых аудиоданных. Для свободного использования языка Nyquist необходимо иметь общие представления о языке Lisp. Nyquist совместим с синтаксисом XLISP[4] и позволяет исполнять команды, интерпретируемые LISP [10, 11, 12, 13]. Наравне с поддержкой синтаксиса XLISP язык Nyquist также поддерживает команды языка SAL [14]. SAL — несколько отличается от Lisp и предлагает расширенные утилитарные возможности, к примеру, открывает некоторые возможности по записи нотных данных с интерпретацией в Nyquist. Nyquist имеет свою собственную интегрированную микросреду разработ-

[1] Csound — язык программирования специального назначения, направленный на реализацию программных аудиопроектов средней и высокой сложности. Csound хорошо справляется с такими задачами как: нелинейная обработка аудиоматериала, линейная обработка аудиоматериала, написание утилит по контролю за обрабатываемым аудиоматериалом, статистика и учёт аудиоданных, автоматизация процессов сведения и премастеринга звука. Csound, также как и Nyquist, имеет свою микропрограммную среду, разработанную с помощью языка C. Язык является кроссплатформенным и распространяется по лицензии LGPL, что обеспечивает его открытость и возможность модификации его кодов. Язык также применяется и в синтезе аудиоформ, с помощью него можно успешно синтезировать различные аудиокомпозиции. Автор тестировал версию языка v.6.12.0.

[2] V — специализированный язык программирования, разработанный в 1966 году Максом Мэтьюсом (Max Mathews), работавшим в Центре акустических и поведенческих исследований при лаборатории Белла. Активное внедрение данного языка стало происходить ближе к 1969 году.

Прим. автора. Язык программирования V является одной из первых программных технологий (на основе Music N), призванных обеспечить вычислительными методами качественный и точный аудиосинтез. Язык проектировался с помощью Фортрана (FORTRAN), поэтому вобрал в себя некоторую его логику. Отличительной особенностью V от семейства языков Music N служила независимость от ассемблера. Особенно это касалось предыдущих версий языка Music IVF и Music IVBF, которые были ориентированы на ассемблер. Язык V внёс фундаментальный вклад в компьютерную аудиоинженерию, а именно в область компьютерного аудиосинтеза. Благодаря языку V были открыты новые возможности, имеющие далеко идущие перспективы, расширяющие инженерно-техническую практику в сфере обработки аудиоматериала. Хочется отметить, что язык V, помимо инженерной практики, внёс весомый вклад и в развитие творческих способностей музыкантов, благодаря ему, появилась возможность искусственно генерировать аудиопартии, тем самым значительно сокращая время на аранжировку. Не обошла данная разработка сферу образования и науки. В сфере образовательной деятельности язык начал укреплять позиции в направлении подготовки специалистов широкого профиля, деятельность которых так или иначе затрагивала разные аспекты изучения и производства аудиоматериала. В сфере науки язык (особенно на то время — конец 1960-х — и начало 1970-х годов прошлого столетия) открыл огромные возможности для более глубокого исследования процессов аналого-цифрового и цифро-аналогового преобразования звука и его акустических параметров. На основе данного языка были сформированы новые методы, позволяющие нелинейно вести симплексный аудиосинтез. В научно-технической и некоторой специализированной литературе по языкам программирования можно встретить различные написания данного языка, к примеру «Music V», «Music-V», «music V» или как у нас в статье — «V». Все упомянутые описания равнозначны и могут употребляться в научной периодике. Единственное, где может возникнуть путаница это употребление «V» не как принадлежности к версии (5), а как к языку «V». Речь идёт о статически-типизированном компилируемом языке программирования широкого профиля V, который очень похож на язык Go и испытывает влияние Oberon, Rust и Swift. Его написание ряд системных шрифтов (включая Times New Roman) идентифицируют также как и индекс 5 (V=5/ Music V) и (V=V).

[3] CMusic — специализированный язык программирования, созданный в Калифорнийском университете Сан-Диего (США) в 1980 году (F. Richard Moore, D. Gareth Loy). Представляет собой синтетический язык программирования похожий на Music-N. Имеет DOS-совместимую версию, которая называется pcmusic. В первоначальном варианте язык был заточен под UNIX-подобные операционные системы. Язык снабжён внушительной библиотекой генераторов блоков и подпрограмм обработки сигналов, которые реализовывались на протяжении многих лет. Он работает как компилятор, следовательно, не предназначен для работы в реальном времени. В pcmusic/Cmusic практически нет ограничений на сложность инструментов, потому что система не «беспокоится» о скорости обработки. Более медленные процессоры, естественно, требуют больше времени для компиляции, но более быстрые — необязательно улучшают инструментальные возможности языка. ***Прим. автора.*** В научно-технической периодике могут встречаться различные написания данного языка «Cmusic, cmusic, CMusic, pcmusic, PCmusic» — все написания равноправны, такой стиль изложения (в написании языка) объясняется его историческим развитием и связью с UNIX-подобными ОС.

[4] ***Прим. автора.*** Прежде всего, подразумевается версия XLISP 2.0. Версия XLISP 3.0 также имеет сходство с синтаксисом Nyquist, однако некоторые функции могут иметь различия в трактовках. См. литературу пункты [11,12]. О разнице функционала XLISP версии 1.7 см. пункт [10]. Чтобы больше узнать о процедурах символьных вычислений, можно ознакомиться со следующим руководством Touretzky, D. S. 1984. LISP: A Gentle Introduction to Symbolic Computation. New York: О некоторых списочных функциях можно прочитать здесь Tierney L. XLISP-STAT A Statistical Environment Based on the XLISP Language (XLISP-ST AT Version 1.0) / Technical Report No. 512, February 1988, School of Statistics University of Minnesota p.57

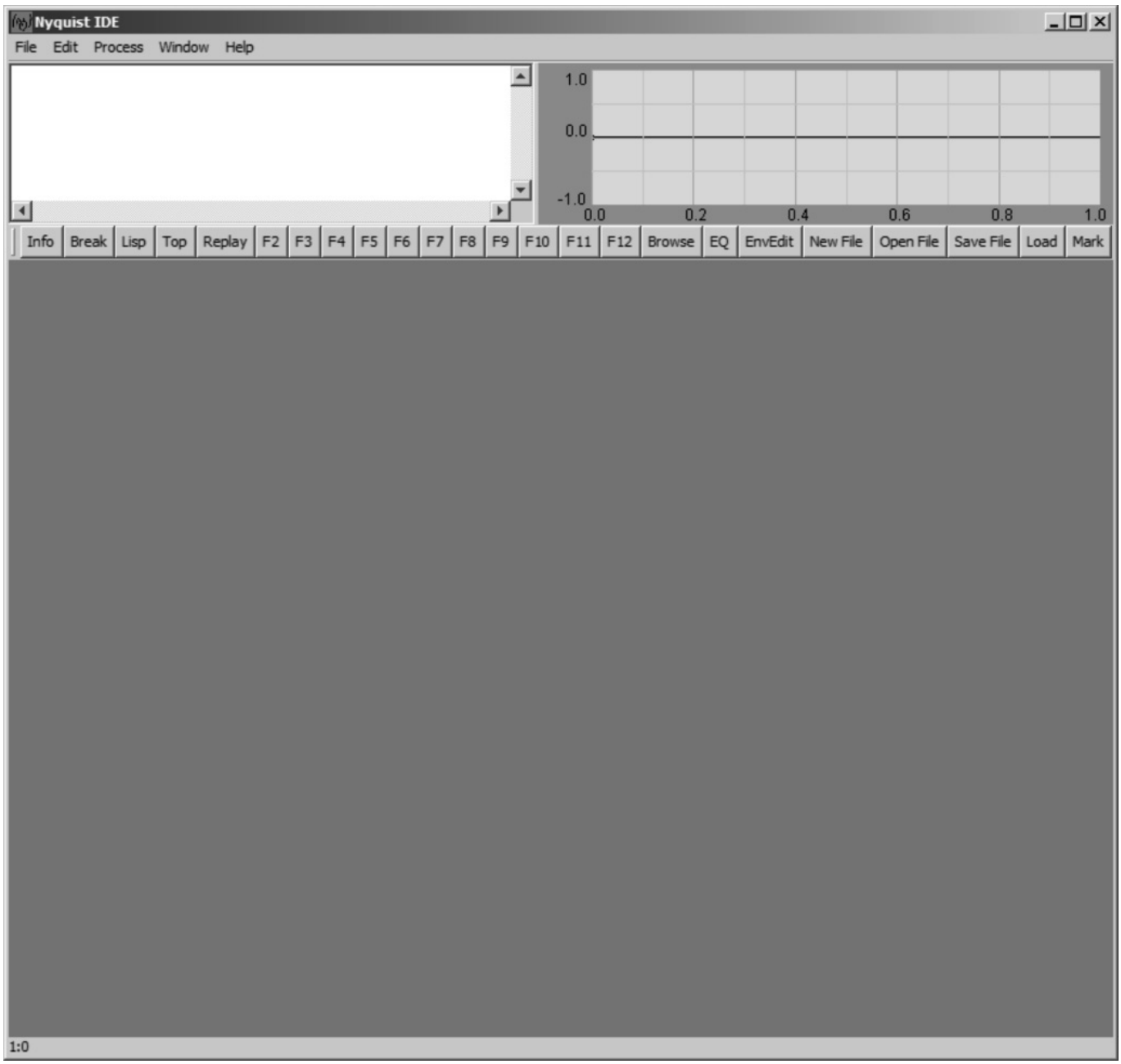

Рис. 3. Рабочее пространство автономной среды Nyquist IDE.

ки. Она позволяет выводить на экран весь перечень исполняемых процессов в отдельном окне и выводит подсказки Lisp —SAL по изложению синтаксиса в составляемом списке программных действий, тем самым позволяя ориентироваться в оценке выражений Nyquist в Lisp-синтаксисе и соответственно при оценке SAL-команд. Это очень удобно, поскольку работая в SAL-совместимом режиме, мы можем пользоваться исключительно арсеналом SAL-программирования, а Nyquist автоматически компилирует их в Lisp. Таким образом, специалисту достаточно знать один из изложенных выше языков программирования, чтобы начать реализовывать свои задумки и интерпретировать их в машинный код. Отдельно хочется сказать несколько хороших слов об интегрированной микросреде разработки NyquistIDE. Среда NyquistIDE является автономной средой исполнения команд и довольно хорошо продумана.

Среда NyquistIDE интегрирует в себе различный набор полезных функций по обработке аудиоматериала средней и высокой сложности. Работая автономно в среде NyquistIDE в операционной системе MS Windows, при

условии выполнения обработки аудиоданных (к примеру, компрессия аудиоканала), в фоновом режиме можно лицезреть командную строку[1] языка Nyquist, она выполняется как подзадача, поэтому необходимым условием непрерывной обработки данных является исполнение команд Nyquist одновременно со средой NyquistIDE. Рутинным способом использования NyquistIDE является редактирование и создание нескольких файлов. После необходимых процедур по редактированию данных происходит их загрузка. Чтобы приступить к процедуре загрузки данных, NyquistIDE сохраняет файл и устанавливает текущий каталог, Nyquist в каталог сохраняемого файла и только после этого передаёт команду загрузки Nyquist. В таких условиях можно заметить, что NyquistIDE в автоматическом режиме отправляет выражения Nyquist для их оценки. Это очень удобно с точки зрения функционального программирования, потому как можно увидеть команды и их результаты в окне вывода[2]. При загрузке окна отображаемого файл NyquistIDE задействует setdir с целью изменения текущего Nyquist-каталога. Такие правила позволяют сохранять одновременно две компьютерные микропрограммы. При стандартном подходе подобного рода операция предполагает хранение всех файлов проекта в одном каталоге с избеганием редактирования вручную текущего Nyquist-каталога[3]. Среда NyquistIDE предполагает работу с окнами. В зависимости от используемой операционной системы для окна может быть характерной ситуация, при которой окно попадает в положение (в зависимости от масштабирования), откуда его невозможно переместить в другое положение экрана. Специально для таких ситуаций в NyquistIDE предусмотрена команда меню Window (Окно), которая осуществляет автоматическую компоновку и масштабирование окон в режиме многозадачности[4]. Команда осуществляет настройку предпочтений для определения высоты списка завершения относительно высоты окна вывода информации. Интерактивность работы с NyquistIDE оставляет приятные впечатления, к примеру, когда возникает необходимость работы с функциями, то при вводе начальных имен функций среда NyquistIDE автоматически сформирует соответствие функций и некоторых параметров и представит их в виде списка. Эта интеллектуальная составляющая позволит привести в соответствие начальные буквы и подстроку полной функции с её именем. Эти операции контролирует опция Использования полного списка предпочтений завершения кода. Также из раздела локальной копии имеется возможность вывода справки по каждой из функций (команда IDEFAQ). В целях обеспечения контроля в процессе обработки аудиоматериала NyquistIDE можно создать интерактивный пульт управления с кнопками и ползунками, каждый из которых будет настроен на отдельный канал, при этом Nyquist позволит отправить значения в массив ползунков — можно настроить получение значений в режиме реального времени от массива ползунков как звук. Микросреда исполнения команд NyquistIDE построена на языке Java[5], поэтому было бы нелишним иметь заранее предустановленную среду исполнения команд Java[6]. По умолчанию командная строка Nyquist имеет путь обращения:

java-jar jnyqide/jNyqIDE.jar

Листинги программных событий сохраняются в расширении *.lsp, листинги на языке SAL имеют расширение *.sal. Сценарные команды и микропрограммы языка Nyquist имеют разрешение *.ny.

Для удобства использования NyquistIDE предусмотрены расширения, которые помогают работать с меню управления расширениями (Manage Extension Menu), предлагаем оригинальный их список[7], приведенный во внутренней документации NyquistIDE[8].

[1] Командная строка должна соответствовать версии языка программирования Nyquist.

[2] При выводе команд следует помнить, что вносить корректуру в синтаксис, а также изменять введённые команды или вводить новые команды в режиме окна вывода не представляется возможным по причине отсутствия функции редактирования команд в режиме окна вывода.

[3] ***Прим. автора.*** Фактически описанная ситуация показывает, что ***необходимо избегать*** вызова setdir в коде программы.

[4] ***Прим. автора.*** Масштабирование окон в процессе программирования сильно упрощает контроль над выполняемыми командами и отладкой программного кода. Справедливо отметить, что последние версии операционных систем MS Windows (8,8.1–10) поддерживают развитую систему графического представления окон (включая поддержку нескольких виртуальных рабочих столов и многооконный режим). Однако предыдущие системы корпорации Microsoft в ряде случаев не всегда корректно выполняли данные процедуры, и поэтому встроенная локальная функция компоновки окон очень выручает при работе над нагруженными проектами. Что же касается интеграции среды разработки NyquistIDE под другие платформы (Unix, Linux, MacOS), то здесь имеется своя специфика в зависимости от дистрибутива, поэтому наличие данной опции также имеет приоритет.

[5] Java — интерпретируемый язык программирования широкого профиля предназначенный для выполнения задач разной степени сложности, в частности язык способен удовлетворять потребности проектировщиков в области сетевого администрирования (клиент — сервер) и сетевой автоматизации на уровне взаимодействия WEB-приложений. Язык поддерживает концепцию объектно-ориентированности и лицензирован большинством крупных компаний на рынке программного обеспечения.

[6] Среда исполнения команд Java необходима для стабильного (автономного) функционирования NyquistIDE. Автор использовал 64-х разрядную версию Java(TM) Platform SE10.0.2.0. Некоторые ключевые команды, применяемые для редактирования файлов, являются стандартными и встроенными в среду Java, к примеру, команда Control-Z (команда — «Отменить»). В операционной системе MacOS X аналогичное действие выполняет команда Command-Z (команда — «Отменить»). Специальная команда Control-U «копировать в Lisp» (Command-U на MacOS) копирует выделение сектора программного кода и передает его на окно «ввода» точно также, как если бы оно было введено вручную.

[7] ***Прим. автора.*** Перевод аннотации ***расширений*** на русский язык Таран В. В. Список расширений детально представлен в справочном руководстве по языку Nyquist, (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.15).

[8] При инсталляции интегрированной программной микросреды NyquistIDE для исполнения команд языка Nyquist — данная инструкция располагается

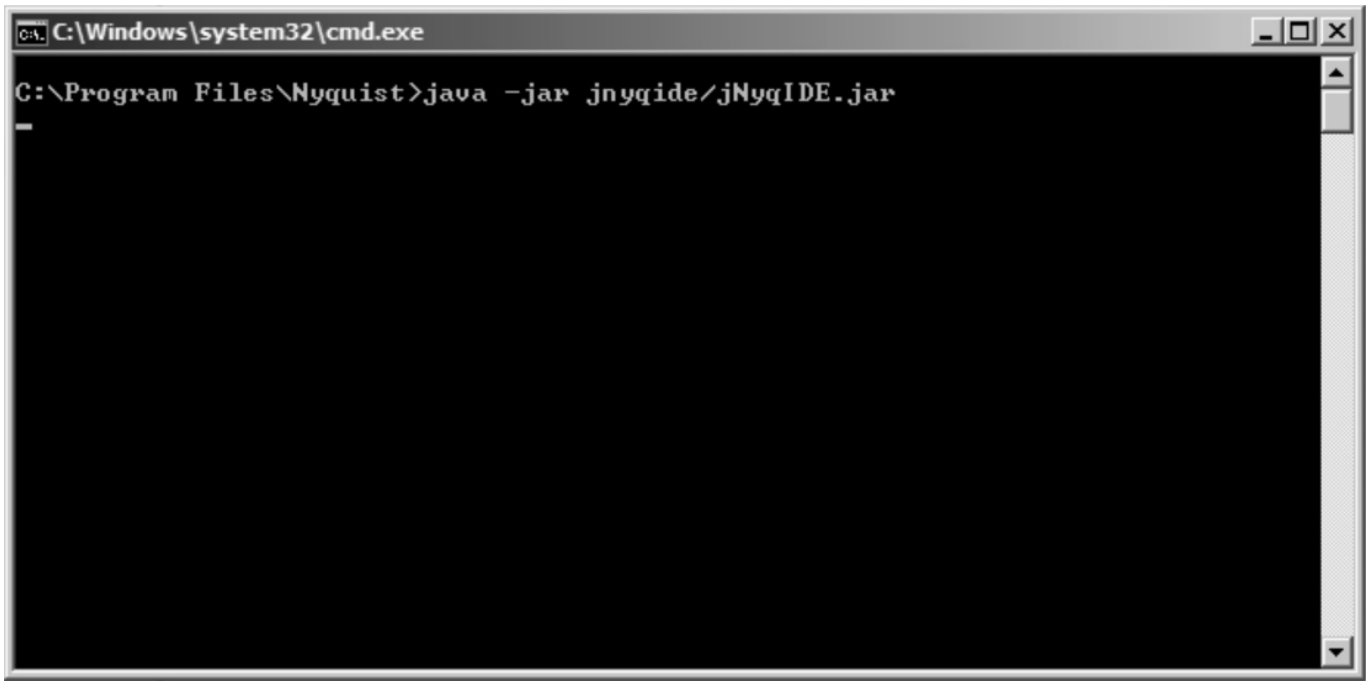

Рис. 4. Командная строка, загружаемая вместе с NyquistIDE.

Таблица 1. **Список расширений NyquistIDE, способствующих повышению эффективности обработки аудиоматериала (в зависимости от конкретных ситуаций)**

В принципе система управлением расширений или (диспетчер расширений) действительно могут существенно расширить горизонты Nyquist-программирования. Диспетчер расширений имеет функцию загрузки расширений[1], написанных на Lisp или SAL. На ряду с загрузкой расширений имеется возможность расширения самого аппарата программы NyquistIDE, для этого потребуется знание языка C или Java, а также представление о структуре кода программы, который по необходимости может быть перекомпилирован.

Некоторые специальные файлы могут быть включены в расширение. Файл вывода «разгрузка.lsp» — это файл Lisp, который загружается автоматически при запуске Nyquist. Nyquist просматривает подкаталоги каталога lib, чтобы найти все файлы «разгрузка.lsp». Как правило, файлы «разгрузка.lsp» используются для создания «заглушек» для функций в расширениях, так что код расширения в основном загружается в динамическом режиме по требованию, экономя время и пространство при запуске Nyquist. Расширения могут также включать nyquistwords.txt, который представляет собой описание функций в расширениях, используемых для создания списка вариантов автодополнения. Когда функция расширения появляется в списке вариантов автодополнения, за ней следует «ext:» и имя расширения, например, элемент автодополнения «speed-dial (list) ext: dtmf» означает, что функция быстрого набора определена в каталоге dtmf. По соглашению, любая функция в списке вариантов автодополнения должна быть непосредственно вызываемой, с записью в autoload.lsp, так что функция будет динамически загружаться. Чтобы отправить расширение для возможной публикации, лучше всего поместить файл(ы) расширения в локальный каталог и использовать настроенный локальный список расширений. В настройках Nyquist есть кнопка, с помощью которой можно назначить локальный файл для списка расширений. Он должен иметь тот же формат, что и extlist.txt по URL-адресу, указанному выше (вы можете просто открыть URL-адрес в своем браузере, чтобы увидеть его). На разрабатываемое расширение можно ссылаться в списке расширений, используя URL-адрес с файлом: / / protocol. Для контрольной суммы используйте любое значение. Диспетчер расширений вычислит правильную контрольную сумму и отобразит её, так что вы сможете скопировать её в свой extlist.txt. Нажать кнопку «обновить», чтобы получить её, и повторить попытку установить расширение.

по условному внутреннему адресу [путь к диску установки]:/Пользователи/учётная запись/nyquist/demos/index.htm. ***Прим. автора.*** Для удобства понимания ключевых расширений автором был осуществлён перевод с толкованием приведённых расширений.

[1] При установке расширения сначала загружается один файл, используя URL-адрес, полученный из списка расширений, и файл проверяется на наличие заголовка, который может точно определять дополнительные файлы для загрузки.

Таблица 2. Перечень основных элементов управления среды NyquistIDE с краткими описаниями предполагаемых действий.

Перечень расширений NyquistIDE	
arpeggiator	Описывает и выполняет функции в целях создания эффектов Арпеджио (последовательного исполнения звуков аккорда).
atonal	Алгоритмическая атональность. Расширение использует простые методы алгоритмической атональности, весьма эффективно и уже применялось в ряде музыкальных композиций, как для акустических, так и для электронных партий. С исп. Хорхе Састре, с англ. Джордж Састре (Jorge Sastre) применил этот код для создания атональных мелодий.
bandfx	Показывает, как использовать библиотеку bandfx в Nyquist. Множественные эффекты полосы разделяют входной сигнал на частотные полосы и создают различные эффекты (например, различные задержки) для каждой полосы.
cellautomata	Использует клеточное (кластерное) автоматическое устройство для создания отметок в Nyquist. Разработан Энн Льюис (Ann Lewis).
clipsoften	Содержит код для снижения низкочастотной вибрации на участке, где происходит отсечение.
compress	Обеспечивает динамический компрессор для речевого текста с шумовыми помехами в аудио формате.
convolve	Показывает, как использовать свернутые композиции в Nyquist.
distortion	Описывает, как использовать функцию формы в Nyquist для достижения искажения.
dtmf	Вводит в действие звуковой генератор DTMF (DTMF — это «сенсорные тона», посылаемые телефонами для набора номеров).
ext-template1	Является примером расширения Nyquist, использующего документацию в HTML.
ext-template2	Является примером расширения Nyquist, в котором документация находится в исходном тексте (программном коде).
fft	Описывает, как использовать функции FFT в Nyquist для выполнения спектральной обработки (использует синтаксическую структуру Lisp, см. также fftsal).
fftsal	Описывает, как использовать функции FFT в Nyquist для выполнения спектральной обработки (использует синтаксическую структуру SAL, см. также fft.).
fm-voices-chowning	Повторно реализует голоса на основе FM-синтеза по технологии, разработанной пионером компьютерной музыки Джоном Чаунингом (John M. Chowning).
gran	Выполняет детализированные синтетические функции.
intro	Вступительная часть справочного руководства Nyquist предлагает ряд небольших примеров. Они объединены в файлы examples.sal, и в файлы examples.lsp, которые вы можете загрузить и запустить в Nyquist.
labels	Вводит в действие код для чтения и написания файлов с меткой Audacity®.
lpc	Обучает, как использовать функции LPC особенно полезные для моделирования голоса и перекрестного синтеза.
mateos	Включает ряд примеров, выполненных на текущий момент в синтаксической структуре Lisp, однако они могут быть загружены и вызваны из программ SAL (например, описанный ниже учебник по Фазовому Вокодеру использует mateos/organ.lsp.). Директория mateos/содержит: bell.lsp, gong.lsp, organ.lsp и tuba.lsp. Автор примеров Дэниэл Матеос (Daniel Mateos).
pmorales	PMORALES — Педро Моралеса, содержит следующие программы: pjmg.lsp — некоторые вспомогательные функции: randi1, randi2, randh1, rndh2.
midi	Описывает, как читать и записывать стандартные MIDI-файлы.
moog	Выполняет роль имитатора мини-муга (аналогового синтезатора).
plight	Содержит образцы звука барабана и программное обеспечение драм-машины.
pvoc	Показывает, как использовать фазовый вокодер в Nyquist.
regression	Выполняет функции линейной регрессии.
reverse	Имитатор реверсивного устройства. Выполняет функции реверса звука и звуковых файлов.

Таблица 2 (продолжение). Перечень основных элементов управления среды NyquistIDE с краткими описаниями предполагаемых действий.

Перечень расширений NyquistIDE	
rhythm	Обеспечивает варианты, использующие шумовые импульсы и FM (частотную модуляцию), наряду с перегруппировкой и повторением сигнала, чтобы исследовать различные ритмические эффекты.
sdl	Предлагает компактный язык оценки SDL, который может быть использован в качестве стенографии для создания баллов Найквиста.
shepard	Это исходный файл Lisp, производящий тоны Шепарда. Существует также shepard/shepard. ny.(для тонов Шепарда) — плагин для Audacity®
sliders	Демонстрирует использование регуляторов для создания интерактивного управления звуком и композициями в Nyquist. Для более подробного ознакомления с данным расширением предусмотрен файл sliders/slider-demos.sal. Также данная информация хорошо описана в справочном руководстве по языку Nyquist.
spatial	Выполняет кодирующие функции звукового формата Dolby surround.
stk	Проигрывает различные инструменты STK, которые были перенесены в Nyquist. Этот файл — просто код Lisp, он должен быть весьма прост для чтения для того, чтобы понять названия функций и параметров. Некоторые выражения SAL добавлены в пояснениях и могут служить инструкцией по работе с ***наклонными*** в Lisp — [g4 (bow-env d)] и соответственно — в SAL [(g4, bow-env(d))], т.е. первый элемент каждого списка — это функция, а остальные элементы — параметры.
voice	Дает исходный код и примеры для модели исходного фильтра певческого голоса. Расширенный комментарий комментарии находится либо в версии voice/voice-synthesis.sal, либо в версии voice/voice-synthesis.lsp.
vosim warble	VOSIM — Voice Simulator (имитатор речи) приводится исходный код и примеры метода синтеза «VOSIM».
wind	Представляет определенный код для создания звуков ветра.

Nyquist имеет хорошо продуманный интерфейс обозревателя событий, поэтому нажимая на кнопку «Обзор» или используя команду меню Window: Browse, NyquistIDE отобразит окно браузера, которое предварительно загружается с несколькими командами Nyquist с целью создания звуков.

Имеется возможность настройки параметров, прослушивание звуков и «захвата» выражения, которое создает звук. Во многих случаях выражение проверяет, определены ли необходимые функции, загружая файлы, если это необходимо, перед воспроизведением звука.

Если вы хотите использовать звук в своей собственной программе, вы можете во многом упростить процесс, непосредственно загрузив необходимый файл только один раз в начале вашего файла[1]. Чтобы дать полное представление об основных элементах управления приведём подробные параметры кнопочной панели:

Неотъемлемой частью конкретных решений на языке Nyquist являются подключаемые программные LADSPA-модули[2] обработки аудиоданных, портированные под микросреду исполнения сценариев Nyquist Prompt (портирование позволяет управлять Linux-плагинами в среде MS Windows)[3] в программе Audacity®. Они рассчитаны на решение задач следующего уровня:

- Усиления моносодержимого дорожки

[1] ***Прим. автора.*** Последняя версия Nyquist поддерживает сочетание Lisp и SAL, поэтому можно оказаться в положении, когда код из браузера выражен на одном языке, в то время как вы работаете на другом. Лучший способ справиться с этим — поместить код для звука, который вам нужен, в функцию, определенную в Lisp (*.lsp) или файле Sal (*.sal). Загрузите файл (из Lisp, используйте команду sal — load для загрузки файла SAL) и вызовите оператор-функцию с выбранного вами языка.

[2] LADSPA (**L**inux **A**udio **D**eveloper's **S**imple **P**lugin **A**PI) — интегрируемая с редакторами обработки звука программная микросреда, объединяющая разночастотные фильтры обработки сигнала и различные цифровые эффекты. По сути, служит стандартом интерфейса прикладного программирования (API) для обработки аудиосигнала и наложения на него различных спецэффектов, предусмотренных алгоритмической базой подключаемого к интерфейсу модуля. Использование данной технологии регулируется LGPL-соглашением (GNU Lesser General Public License).Технология LADSPA имеет своё продолжение в более модернизированной программной микросреде LV2. По сути LV2 вбирает в себя все предыдущие функции LADSPA, но несколько расширяет горизонты цифровой обработки аудиматериала.

[3] Nyquist Prompt одинаково хорошо осуществляет поддержку синтаксиса языка Lisp и языка SAL. Располагаемая после установки программы инструкция по адресу [C:/Program Files (x86)/audacity/help/manual/man/nyquist_prompt.html] гласит: Если вводимый вами код Lisp/SAL неадекватно распознается, либо вообще не распознаётся средой исполнения команд, программа автоматически выведет сообщение о невалидности кода с подсказкой о создании исправления. ***Прим. автора.*** Существуют LADSPA-модули с собственным инсталлятором под MS Windows, они позволяют напрямую выгружать модули в хост-папку аудиоредактора.

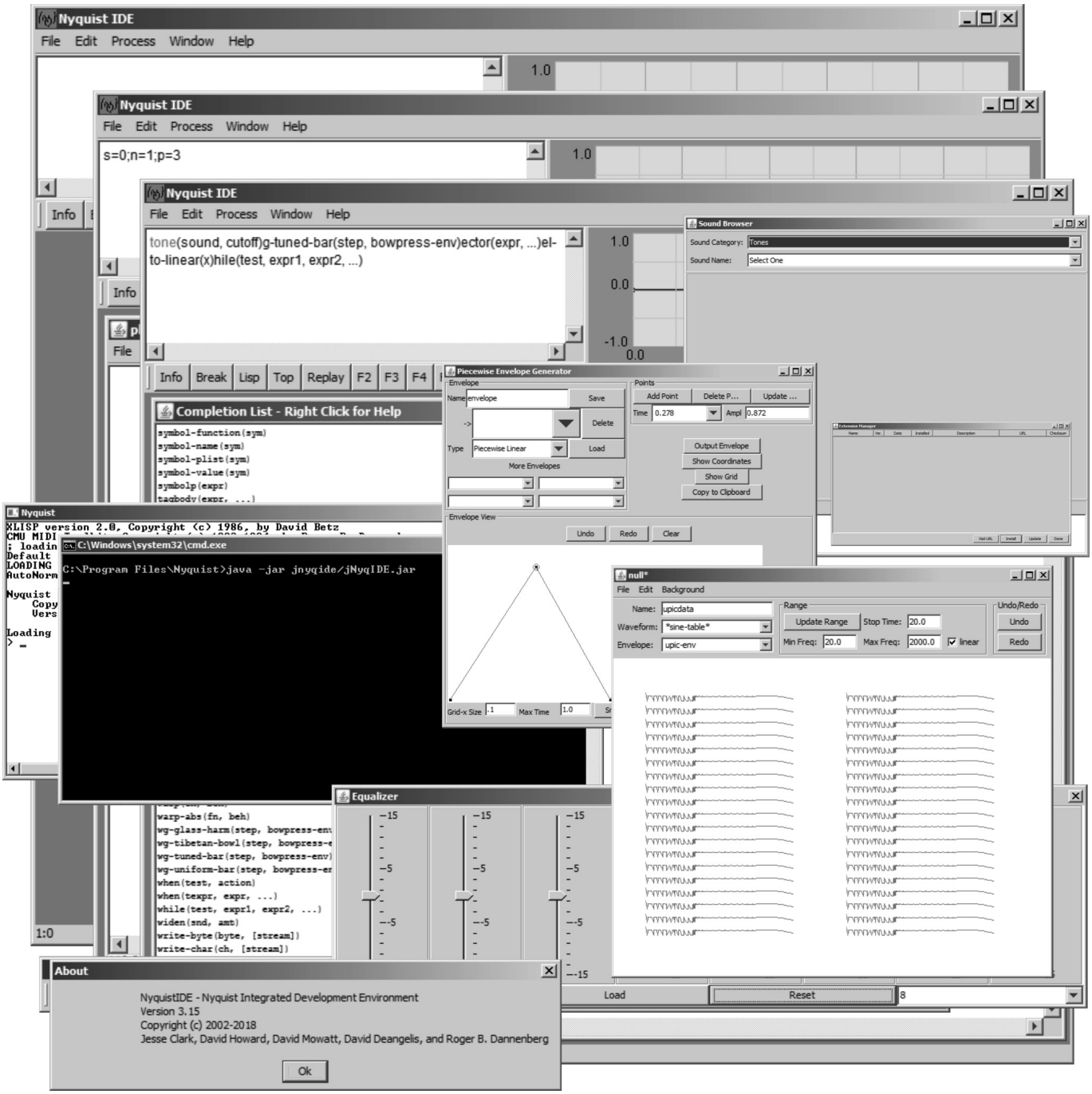

Рис. 5. Рабочее пространство автономной программной среды NyquistIDE в активном состоянии.

Рис. 6. Плавающая Панель управления средой исполнения программного кода NyquistIDE.

Часто используемые кнопки	Пояснения выполняемых действий
Info — Print	Информация об использовании памяти Nyquist, включая количество свободных ячеек «минусов», количество накоплений «мусора», общее количество ячеек минусов, общий объём буферной памяти выборки и объём памяти в буферах свободных выборок.
Break — Send	Символ разрыва в XLISP. Его можно использовать для ввода отладчика (цикл прерывания) во время работы программы. Для того чтобы продолжить надо набрать (co).
SAL/Lisp — Switch modes	Sal/Lisp — переключение режимов. Кнопка определяет режим (SAL или Lisp), при котором возможны переключения, но не изменения текущего режима работы. Например, если вы находитесь в режиме Lisp и хотите ввести команду SSL, сначала нажмите кнопку SSL.
Top — Enters (top) into Nyquist	Вводы верхнего уровня (Top — Enters) в Nyquist. Если размер приглашения ввода XLISP равен 1> или некоторому другому целому числу, за которым следует «>», нажатие верхней кнопки завершит цикл отладки и вернется к приглашению верхнего уровня.
Replay — Enters (r) into Nyquist	Вводы воспроизведения (r) в Nyquist. Эта команда воспроизводит последний вычисленный звук.
F2-F12 — Enters (f2) etc. into Nyquist	Вводы F2-F12 — вводит (f2) и т.д. в Nyquist. Эти команды не являются встроенными и позволяют пользователям определять свои собственные действия.
Browse — Equivalent to the Window: Browse menu item	Обзор — эквивалент окна: пункт меню обзор.
Equivalent to the Window: EQ menu item	Эквивалент окна: пункт меню EQ.
EnvEdit — Equivalent to the Window: Envelope Edit menu item	EnvEdit — эквивалент окна: Пункт меню «Редактирование конверта».
NewFile — Equivalent to the File: New menu item	Эквивалент Файла: Новый пункт меню. Открывает новое окно редактирования файла для создания и загрузки Lisp или программного файла SAL.
OpenFile — Equivalent to the File: Open menu item	Эквивалент Файла: Пункт меню открытия. Открывает существующий список Lisp или программный файл SAL для редактирования и загрузки.
SaveFile — Equivalent to the File: Save menu item	Эквивалент Файла: Пункт меню сохранения (находится в строке меню окна редактирования). Сохраняет содержимое окна редактирования в связанный с ним программный файл.
Load — Equivalent to the File: Load menu item	Загрузка — эквивалент Файла: пункт меню загрузки (находится в строке меню окна редактирования). Выполняет операцию сохранения, а затем отправляет команду в Nyquist, которая загружает файл как программу.
Mark — Sends a Control-A to Nyquist	Метка — передачи управления — A (вызов Control-A) на Nyquist. Во время воспроизведения звука отображается и записывает приблизительное время в аудиопотоке.

- Усиления стереосодержимого дорожки
- Фильтрации нижних частот
- Фильтрации верхних частот
- Ввода и вывода задержки аудиосигнала
- Корректировка ввода белого шума

Для наглядности процесса обработки аудиоматериала через Nyquist Prompt с точными математическими параметрами предлагаем пример программного сценария, написанного Роджером Данненбергом для выполнения эффекта импульсивной задержки с открытием дианового окна и постоянным числом эхо:

```
; nyquist plug-in
; version 1
; type process
; name "Delay…"
; action "Performing Delay Effect…"
; info "Demo effect for Nyquist by Roger Dannenberg.\
nThis effect
creates a fixed number of echos.";(should be all on one
line)
; control decay "Decay amount" int "dB" 6 0 24
; control delay "Delay time" real "seconds" 0.5 0.0 5.0
; control count "Number of echos" int "times" 5 1 30
```

```
(defun delays (s decay delay count)
(if (= count 0) (cue s)
(sim (cue s)
(loud decay (at delay (delays s decay delay (– count 1)))))))
(stretch-abs 1 (delays s (– 0 decay) delay count))
```

Тот же самый код только с изменёнными параметрами:

```
; nyquist plug-in
; version 1
; type process
; name "Delay…"
; action "Performing Delay Effect…"
; control decay "Decay amount" int "dB" 6 0 24
; control delay "Delay time" real "seconds" 0.3 0.0 3.0
; control count "Number of echos" int "times" 3 1 30
(defun delays (s decay delay count)
(if (= count 0) (cue s)
(sim (cue s)
(loud decay (at delay (delays s decay delay (– count 1)))))))
(stretch-abs 1 (delays s (– 0 decay) delay count))
```

Примечательно, что числовые параметры задачи отклика (эхо) должны располагаться на одной строке, в случае разрыва строк параметры эффекта могут быть интерпретированы неправильно. Если программа распознает одну из строк управления, то она тут же запустит диалоговое окно для запроса у пользователя параметров подключаемого модуля (плагина). Каждый из параметров включает информацию о текстовом поле для вводимых значений и ползунка для регулировки уровня отклика. После ввода пользователем необходимых для работы плагина данных последнее (финальное) значение будет преобразовано в переменную Nyquist с именем, которое и определяет строку управления. Обратите внимание, что строка информации выводится на экран вверху диалогового окна и параметр \n становится новой строкой. Параметры в строке управления влияют на ограничение параметра. Каждая из строк управления должна соответствовать следующим восьми элементам в хронологическом порядке:

1. Слово «управление»
2. Имя элемента управления — это имя переменной Nyquist, которая будет установлена, когда пользователь будет манипулировать диалогом.
3. Метка слева от элемента управления.
4. Тип значения: либо int (целое число — integer), либо real (реальный).
5. Метка справа от значения (обычно это единицы измерения типа «Гц» или «дБ»).
6. Значение по умолчанию/начальное значение параметра
7. Минимальное значение параметра.

Язык Nyquist и его технический арсенал, включающий огромные возможности по ультратонкому редактированию аудиоматериала, дают основания предполагать, что разработка микроприложений, а также создание утилитарных сценариев на данном языке имеют далеко идущие перспективы.

Язык Nyquist динамично развивается, вобрав в себя лучшие наработки смежных языков программирования (Arctic, Csound, LISP, cmusic, Canon, SAL, XLISP, Fugue), учитывающих различные теории и парадигмы программирования соответствующие временам их развития и применения. Вобрав в себя лучшее из технологий программирования, синтезировав их технологическую базу, язык Nyquist по праву может считаться эффективным и устойчиво-развивающимся языком программирования в сфере аудиотехнологий. Nyquist может применяться при различных процедурах и разрешать многие проблемы при исследовании структуры звука (определении его спектро-частотных характеристик), монтаже и синтезе аудиоматериала [15].Чтобы усилить понимание структуры языка (проследить его идейные предпосылки), предлагаем ознакомиться с рядом статей по смежной тематике [16,17,18,19,20,21,22,23,24,25, 26]. Анализ технической документации языка программирования Nyquist показал, что в будущем Nyquist-программирование может широко использоваться (при условии соблюдения лицензионных соглашений) в открытых системах обработки аудиоданных, для решения различных технических задач с целью повышения эффективности и оптимизации обрабатываемого аудиосигнала, а также для обеспечения оперативного контроля и качественного управления над всеми процедурами обработки аудиоматериала. Проведённый технический анализ основных компонентов данного языка, анализ мировой научно-технической литературы и авторская практика в области программирования могут свидетельствовать о том, что в ближайшей перспективе язык способен решать массу задач и разрешать довольно большое количество проблем, среди которых могут быть следующие:

- Интеграция подключаемых модулей в интерфейсно-ориентированную среду обработки аудиоданных.
- Адаптация аудиопотоков (их ввод и вывод) в компьютерной среде обработке аудиоданных.
- Управление базами аудиоданных, включая их сортировку по спектрочастотным характеристикам.
- Осуществление визуализации процессов обработки аудиосигнала совместно с внешними программными технологиями (включая языки программирования).
- Контроль качества обработанной аудиопродукции (для обеспечения более эффективных пока-

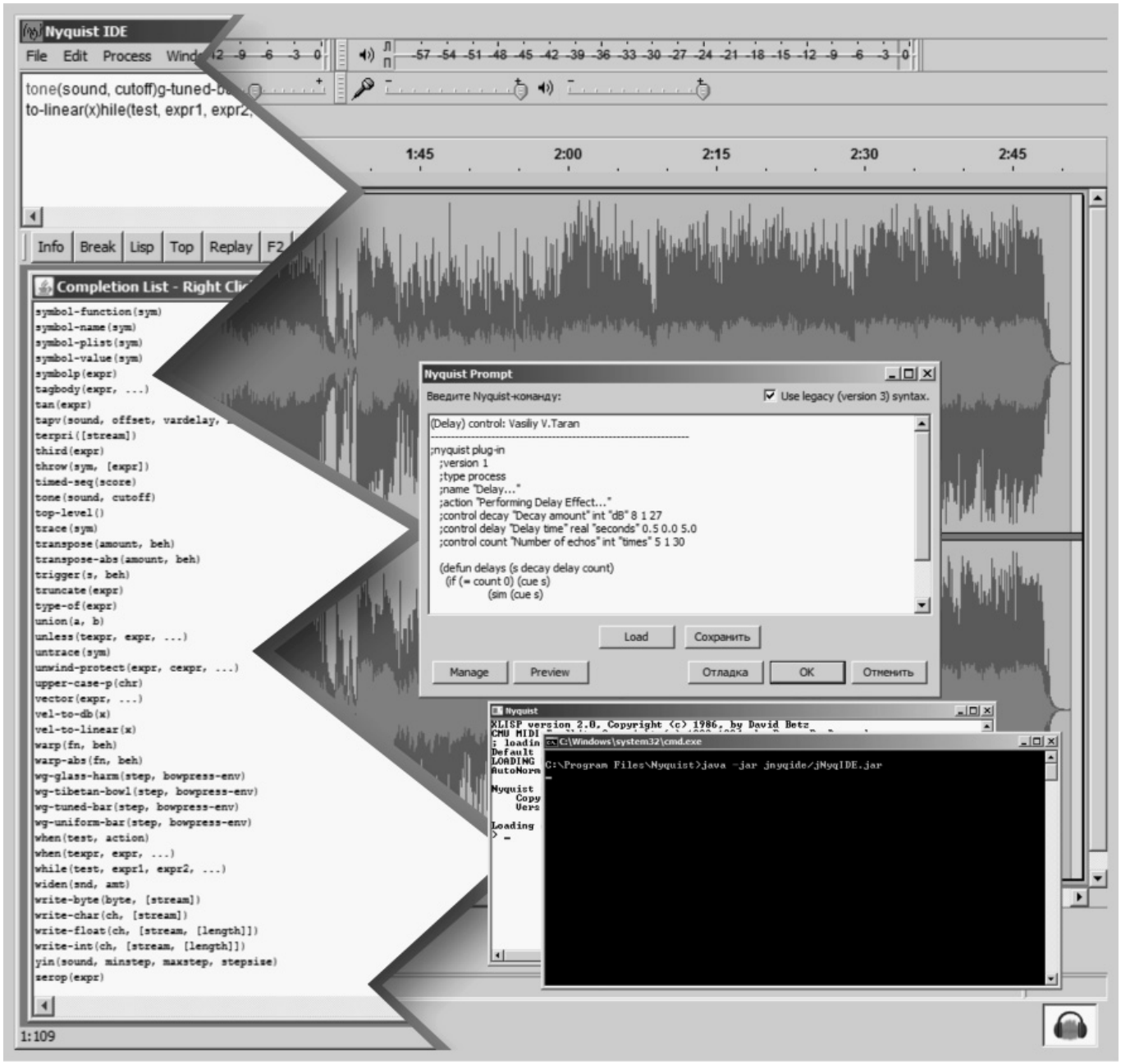

Рис. 7. Иллюстрация демонстрирует интеграционные возможности языка программирования Nyquist. Nyquist, NyquistIDE, Nyquist Prompt и Audacity®, представляют собой единый программный комплекс по обработке аудиоматериала.

зателей её экспорта на различные носители данных).

- Настройка и точная подстройка амплитудных значений сонограммы (включая расстановку динамических маркеров по всему спектру аудиоформы).
- Разграничение спектральных блоков при компьютерном синтезе аудиоматериала, с возможностью коррекции стереопанорамы.

На основе пятилетней практики в области программирования на языке Nyquist с учётом сравнения его версий, автором данной статьи выделены только наиболее значимые (с его точки зрения) задачи и проблемы, которые в состоянии разрешить данный язык. Так как язык программирования распространяется по GPL-лицензии, важно отметь, что он может использоваться в качестве опорного языка программирования в проведении комплексных научных исследований, затрагивающих все

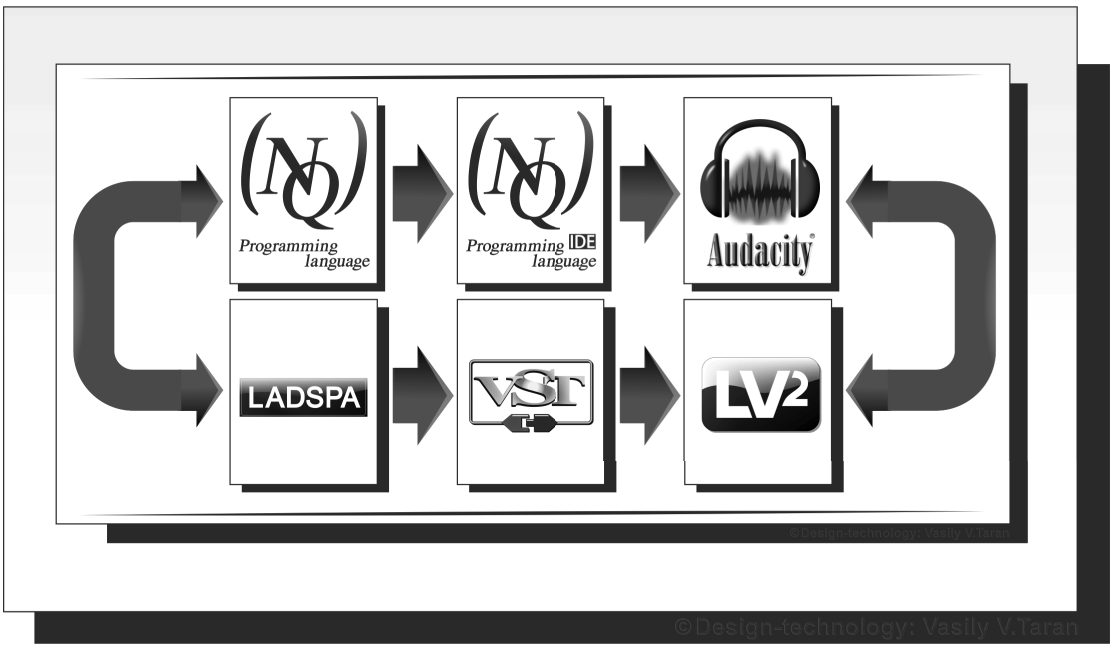

Рис. 8. Замкнутый цикл нелинейной обработки аудиоданных с возможностью их экспорта из программы Audacity®. Иллюстрация демонстрирует интегрируемость различных технологий с программной средой обработки аудиоматериала Audacity® и показывает более высокую степень эффективности обработки аудиоматериала данным аудиоредактором.

сферы компьютерной аудиоинженерии. Язык хорошо подходит для решения прикладных задач и может быть использован как вспомогательная основа при инженерных работах со звуком при сведении и мастеринге аудиоматериала. На его основе удобно демонстрировать процессы обработки аудиоданных различной степени сложности в стенах высших учебных заведений, экспериментировать с аудиоданными, занимаясь прикладными и фундаментальными научными исследованиями в области звука. Язык также значительно расширяет горизонты аудиоинформатики и компьютерной аудиоинженерии в целом, органично вписываясь в её структуру и дополняя водораздел прикладной обработки аудиоданных.

Приложение. Информация об авторских правах и лицензии для Nyquist

© 2000–2002, Роджер Б. Данненберг (Все права защищены).

Перевод оригинального текста лицензии на русский язык и использованная техническая терминология — Таран Василий Васильевич

Перераспределение и использование в исходной и двоичной формах, с модификацией или без неё, допускаются при соблюдении следующих условий:

- При перераспределении исходного кода должно сохраняться указанное выше уведомление об авторских правах, данный список условий и следующее заявление об отказе от ответственности.
- При перераспределении исходного кода должно сохраняться уведомление об авторских правах, список условий и отказ от ответственности, все три из которых приведены ниже в разделе «ИНФОРМАЦИЯ ОБ АВТОРСКИХ ПРАВАХ И ЛИЦЕНЗИИ ДЛЯ XLISP».

Перераспределения в двоичной форме должны воспроизводить вышеуказанное уведомление об авторских правах, данный список условий и следующее заявление об отказе от ответственности в документации и / или другие материалы, обеспечиваемые во время распределения.

Перераспределения в двоичной форме должны воспроизводить уведомление об авторских правах, список условий и отказ от ответственности, все три из которых приведены ниже в разделе «Сведения об авторских правах и лицензии на LISP» в документации и / или в других материалах, обеспечиваемых во время распределения.

Ни имя Роджера Б. Данненберга, Университета Карнеги-Меллона, ни имена любых участников не могут использоваться для одобрения или продвижения про-

дуктов, полученных от этого программного обеспечения, без специального предварительного письменного разрешения.

Это программное обеспечение предоставляется владельцем авторских прав и участниками «как есть» и любые явные или подразумеваемые обязательства, подразумеваемые гарантии товарности и пригодности для конкретной цели являются оговариваемыми. Ни при каких условиях правообладатель или участники не несут ответственности за любые прямые, случайные, специальные, типичные или косвенные убытки (включая, кроме прочего, приобретение заменяющих товаров или услуг; потерю данных или прибыли; или перерыв в коммерческой деятельности независимо от причин их вызвавших и видов ответственности, будь то договор, строгая ответственность или деликт (в том числе небрежность или что-либо другое), возникающие в любом случае от использования этого программного обеспечения, даже если имеется предупреждение о возможности такого убытка.

Прим. Тарана В. В.

Специальные библиотеки **libsndfile** и **liblo,** которые задействует язык **Nyquist в процессе обработки аудиоматериала, не облагаются лицензиями и соглашениями.**

Пожелания автора-разработчика языка Nyquist — Роджера Данненберга

Пожалуйста, присылайте мне исправления ошибок и предложения по улучшению по адресу, указанному ниже. Пожалуйста, не распространяйте модифицированные версии Nyquist без разрешения. Любое перераспределение Nyquist должно включать уведомление о том, что Nyquist может быть получен бесплатно по адресу (http://www.cs.cmu.edu / ~music). Автор просит в качестве любезности отправить ему копию любого продукта, который активно использует Nyquist. Связь с автором осуществляется по текущему почтовому адресу (roger.dannenberg@cs.cmu.edu). Данная просьба основывается на том, что автор заинтересован в получении информации об использовании своего программного продукта. В тех случаях, когда Nyquist используется для создания музыки и проведения исследований, пожалуйста, подтвердите это с выражением признательности в примечаниях к программе, разделах признательность (acknowledgements) и в тех местах, где обычно выражаются благодарность и признательность.

E-mail: roger.dannenberg@cs.cmu.edu

ЛИТЕРАТУРА

1. 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.
2. 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.
3. Dannenberg R. and Fraley C. «Fugue: A Signal Manipulation System with Lazy Evaluation and Behavioral Abstraction», International Computer Music Conference, Computer Music Association (October 1989), pp. 76–79.
4. Dannenberg R. B. Fugue Reference Manual Version 1.0 // Carnegie Mellon University — School of Computer Science/ Pittsburgh, PA 15213, U.S.A. (20.08.1991), p. 90.
5. Dannenberg R. B. Languages for Computer Music / Frontiers in Digital Humanities // November 2018, Volume 5 (Article 26), p.13, doi: 10.3389/fdigh.2018.00026, (https://www.frontiersin.org/journals/digital-humanities).
6. Dannenberg R.B. «The Implementation of Nyquist, a Sound Synthesis Language», Computer Music Journal, 21(3) (Fall 1997), pp. 71–82.
7. Dannenberg R.B. 1997. «Machine Tongues XIX: Nyquist, a Language for Composition and Sound Synthesis», Computer Music Journal, 21(3), pp. 50–60.
8. Dannenberg and Mercer, «Real-Time Software Synthesis on Superscalar Architectures», in Proceedings of the 1992 International Computer Music Conference, International Computer Music Association, (October 1992), pp. 174–177.
9. Vercoe B. «Csound: A Manual for the Audio Processing System and Supporting Program», MIT Media Lab, MIT, Cambridge, Mass., 1986., 120 p.
10. Betz D.M. XLISP: An Experimental Object-oriented Language — Version 1.7 114 Davenport Ave. Manchester, NH 03103 June 2, 1986, p.45.
11. Betz D.M. XLISP: An Object-oriented Lisp — Version 2.0 (The basic manual for Lisp) / 127 Taylor Road Peterborough, NH 03458 February 6, 1988, p.51 (Электронный вариант: https://www.audacity-forum.de/download/edgar/nyquist/nyquist-doc/xlisp/xlisp-man/xlisp-man-index.htm, доп. ссылка: [http://www.cs.cmu.edu/~rbd/doc/nyquist236/part17.html]).
12. Betz D.M. XLISP: An Object-Oriented Lisp — Version 3.0 (28.01.1997), 18 Garrison Drive Bedford, NH 03110, p. 41.
13. History, management and technical instructions of a programming language XLISP (The XLISP family) [электронный ресурс] — URL: http://www.edm2.com/index.php/The_XLISP_family (дата обращения к источнику: 15.11.2019).
14. Taube H.K. (2007) SAL: A simple algorithmic language in common music, Paper presented at International Computer Music Conference, ICMC2007, Copenhagen, Denmark, 27.08.07–8/31/07 pp. 121–124.
15. Dannenberg R. and Ning Hu «Discovering Musical Structure in Audio Recordings» in Anagnostopoulou, Ferrand, and Smaill, eds., Music and Artificial Intelligence: Second International Conference, ICMAI 2002, Edinburgh, Scotland, UK. Berlin: Springer, 2002, pp. 43–57.

16. Dannenberg R. «The Canon Score Language», Computer Music Journal, 13 (1) (Spring 1989), pp. 47–56. (©1989 MIT).
17. Dannenberg, R., McAvinney P., Rubine D. «Arctic: A Functional Language for Real-Time Systems», Computer Music Journal 10 (4, 1986.), pp. 67–78.
18. Rubine D. and Dannenberg R., 1987. «Arctic Programmer's Manual and Tutorial», Carnegie Mellon Computer Science Department Technical Report CMU-CS-87–110.
19. Nishino H, Osaka N, and Nakatsu R. «The Microsound Synthesis Framework in the LC Music Programming Language» Computer Music Journal, vol. 39, № 4 (Winter), 2015, pp. 49–79.
20. Vercoe, B. «The Canonical CSound Reference Manual — Version 5.07. Edited by J. Fitch (J. ffitch), J. Piché, P. Nix, R. Boulanger, R. Ekman, D. Boothe, K. Conder, S. Yi, M. Gogins, A. Cabrera, F. Pinot, and A. Kozar., 1980 p. (http://www.csounds.com/manual/html/).
21. Dannenberg R. and Brandt, E. (1996b). A Portable, High-Performance System for Interactive Audio Processing. In Proceedings of the 1996 International Computer Music Conference (ICMC96), pp. 270–273 (International Computer Music Association).
22. Dannenberg R.B. A Perspective on Computer Music Computer Music Journal, Vol. 20, № 1 (Spring, 1996), pp. 52–56 (http://www.jstor.org/stable/3681271).
23. Dannenberg R. and Neuendorffer T. «Sound Synthesis from Real-Time Video Images». Proceedings of the 2003 International Computer Music Conference. San Francisco: International Computer Music Association, pp. 385–388.
24. Herrera P., Yeterian A., and Gouyon F. Automatic classification of drum sounds: a comparison of feature selection methods and classification techniques in Proc. Int. Conf. Music and Artificial Intelligence (ICMAI), LNAI2445, 2002, pp. 69–80.
25. Dannenberg R. B. Nyquist Reference Manual Version 3.08 // Carnegie Mellon University — School of Computer Science/ Pittsburgh, PA 15213, U.S.A. (26.02. 2013), p.237.
26. Dannenberg R., Kotcher R., «AuraFX: A Simple and Flexible Approach to Interactive Audio Effect-Based Composition and Performance» in Proceedings of the 2010 International Computer Music Conference, San Francisco: The International Computer Music Association, (August 2010), pp. 147–152.

Журнал «Современная наука: актуальные проблемы теории и практики»

Московский международный университет