



Н. В. Климина

дипломант конкурса ИНФО-2019 в номинации «Алгоритмы: вчера, сегодня, завтра»,
лицей имени Героя Советского Союза П. И. Викулова, г. Сызрань, Самарская область, Россия

МЕЖДИСЦИПЛИНАРНЫЙ ПОДХОД В ИЗУЧЕНИИ ПРОГРАММИРОВАНИЯ ЧЕРЕЗ СОЗДАНИЕ ЭЛЕКТРОННОЙ МУЗЫКИ В СРЕДЕ АУДИОПРОГРАММИРОВАНИЯ SONIC PI*

Аннотация

С развитием компьютерных технологий происходит и бурное развитие прикладной музыки, основанной на компьютерных технологиях: музыка для компьютерных игр, для рекламы, для озвучивания бытовой и другой техники, музыка в кино и анимации, телемузыка, академическая музыка. Это свидетельствует об актуальности такой области использования информационных технологий, как программный (алгоритмический) синтез звука, и о необходимости знакомства школьников с этим перспективным ИТ-направлением. О том, что музыка и математика связаны между собой, было известно с античных времен. Теперь мы можем наблюдать тесную связь музыки и информатики. И не просто обработку музыки в программах синтеза, а именно программное создание музыки с помощью кода. В статье показана связь музыкального искусства с искусством программирования, рассмотрена концепция среды аудиoprogrammирования Sonic Pi, продемонстрированы основные возможности этой среды, языковые средства программирования алгоритмов, структур данных и музыкальных композиций на языке Ruby.

Ключевые слова: алгоритмическая музыка, аудиoprogrammирование, программный синтез звука, синтезатор, сэмплы, реверберация, эхо, дисторшн.

DOI: 10.32517/2221-1993-2020-19-2-5-18

1. Алгоритмическая музыка. Аудиoprogrammирование

С появлением компьютеров и расширением их технических возможностей стало привычным применение компьютерной техники в творческих процессах. Не ис-

ключение и музыкальное творчество. В последнее время наблюдается бурное развитие компьютерной музыки, создаваемой с помощью различных технологий. Рабочее место современного композитора теперь включает в себя синтезатор и компьютер, а сам композитор похож на программиста, так как работает с музыкальным

* Материалы к статье можно скачать на сайте ИНФО: http://infojournal.ru/journals/school/school_02-2020/

Контактная информация

Климина Наталья Владимировна, учитель информатики, лицей имени Героя Советского Союза П. И. Викулова, г. Сызрань, Самарская область, Россия; адрес: 446001, Россия, Самарская обл., г. Сызрань, ул. Степана Разина, д. 1; e-mail: natklimina@yandex.ru

N. V. Klimina

Lyceum named after Hero of the Soviet Union P. I. Vikulov, Syzran, Samara Region, Russia

AN INTERDISCIPLINARY APPROACH TO LEARNING PROGRAMMING THROUGH THE CREATION OF ELECTRONIC MUSIC IN THE SONIC PI AUDIO PROGRAMMING ENVIRONMENT

Abstract

With the development of computer technologies, there is a rapid development of applied music based on computer technology: music for computer games, for advertising, for voicing household and other equipment, music in movies and animations, TV music, and academic music. This indicates the relevance of such an area of use of information technologies as program (algorithmic) sound synthesis, and the need for schoolchildren to familiarize themselves with this promising IT area. The fact that music and mathematics are interconnected has been known since ancient times. Now we can observe the close connection of music and informatics. And not just the processing of music in synthesis programs, but the programmatic creation of music using code. The article shows the relationship of musical art with the art of programming, considers the concept of the Sonic Pi audio programming environment, demonstrates the main features of this environment, language tools for programming algorithms, data structures and musical compositions in Ruby.

Keywords: algorithmic music, audio programming, program sound synthesis, synthesizer, samples, reverb, echo, distortion.

кодом. Одной из интереснейших технологий создания музыки является алгоритмический подход к созданию музыкальных композиций.

Алгоритмической музыкой (алгоритмической музыкальной композицией) называют музыкальный отрывок, последовательность или композицию, при создании которой использовались математические модели, правила и алгоритмы [1].

Об актуальности в современном мире такого направления музыкального творчества и развития компьютерных технологий, как **алгоритмический (программный) синтез звука**, свидетельствуют многочисленные исследования, проводимые в разных странах. В России это направление еще не приобрело значительные масштабы и популярность [1]. Однако в настоящее время композитору, звукорежиссеру и аранжировщику недостаточно быть только пользователем персонального компьютера — для эффективного применения информационных технологий в своем творчестве необходимо быть одновременно и программистом. Поэтому так важно получить знания в аудиопрограммировании всем будущим специалистам в области музыки. Базовые знания аудиопрограммирования будут полезны и школьникам, будущее которых не связано с музыкой. В этом случае знакомство с программным синтезом звука решает задачу гармоничного развития личности в условиях цифрового общества.

Для обучения музыкальному программированию нами была выбрана свободно распространяемая **среда аудиопрограммирования с открытым исходным кодом Sonic Pi**. Данный выбор обусловлен не только бесплатностью программы, но и ее богатым функциона-

лом. Программа обладает встроенным русскоязычным гидом, использует текстовый язык программирования Ruby, позволяет «кодить» музыку в режиме реального времени [4, 15]. Дистрибутив программы можно загрузить с официального сайта: <https://sonic-pi.net/>

У программы очень простой интерфейс, ничего лишнего (см. рис. 1).

В редакторе кода, как в текстовом редакторе, будем писать код на языке аудиопрограммирования.

Чтобы проиграть код, необходимо нажать кнопку **Run**. Кнопка **Stop** предназначена для остановки проигрывания мелодии, кнопка **Save** — для записи текстовой версии кода на внешний носитель в формате программы. Кнопка **Load** позволяет загрузить текстовую версию кода в открытую страницу программы (Buffer).

Одновременно в программе можно заполнить девять доступных страниц и переключаться между ними для проигрывания мелодий. С помощью контекстного меню можно копировать фрагменты кода и вставлять их в нужное место в любую страницу программы.

Для проигрывания простейшей мелодии используем команду проигрывания **play <номер ноты>** и команду паузы **sleep <время в секундах>**. Команды пишем в столбик.

Среда аудиопрограммирования Sonic Pi предоставляет большие возможности для создания и управления звуком. Время звучания мелодии, длительность нот, темп мелодии, амплитуда (громкость) звука, стереозвучание, использование готовых образцов в мелодии — все поддается настройке с помощью команд среды. Атрибуты-настройки записываются через запятую к основной команде и включают название настройки с двоеточием



Рис. 1. Интерфейс среды аудиопрограммирования Sonic Pi:
1 — редактор кода; 2 — управление воспроизведением; 3 — управление редактором;
4, 5 — визуализаторы сообщений; 6 — страницы редактора кода

и ее значение (например: **play 62, amp: 2**). У каждой команды свой набор настроек, хотя есть общие настройки для целых групп команд. Если после запятой нажать пробел, контекстное меню само предложит эти настройки в виде списка [6].

При создании мелодий используются как привычные алгоритмические конструкции, свойственные языкам программирования, так и уникальные, характерные именно для среды аудиопрограммирования Sonic Pi. Они позволяют расширить возможности программы при написании мелодий и украшении звука.

Приведенные ниже задания по синтезу звука* могут использоваться на занятиях внеурочной деятельности в V—XI классах или выборочно на уроках, посвященных изучению исполнителей алгоритмов. Предварительного знакомства с каким-либо языком программирования не требуется.

В заданиях указаны дополнительные сведения о командах и пр., которые пригодятся для изучения кода, его написания, экспериментов с программами.

Тексты программ обычно выдаются обучающимся в готовом виде (подготовлены учителем). Но школьники, имеющие музыкальную подготовку, могут сами создавать свои музыкальные фрагменты. То есть можно предложить учащимся три варианта выполнения задания:

- повторить предложенный учителем код;
- перекодировать обычную нотную запись музыкального фрагмента с листа в программный код;
- написать программный код, определив мелодию на слух.

2. Алгоритмическая конструкция «следование» и ее реализация в среде аудиопрограммирования Sonic Pi

Задание 1.

Создание музыкального отрывка с помощью линейного алгоритма.

При написании кода использовать музыкальные ноты в MIDI октавы 4 (см. табл.) [8, 11].

Таблица

Нумерация нот в MIDI

С (До)	D (Ре)	E (Ми)	F (Фа)	G (Соль)	A (Ля)	B (Си)
60	62	64	65	67	69	71

Используем:

- команду **play** — для проигрывания ноты;
- команду **sleep** — для выдерживания паузы между проигрыванием нот;
- параметр длительности звучания ноты **sustain** (можно экспериментировать с изменением значения параметра — этого и других);
- встроенный инструмент — синтезатор **dpulse** — для красоты звучания.

```
use_synth :dpulse
play 64, sustain: 0.125
sleep 0.5
play 64, sustain: 0.75
sleep 0.5
play 56, sustain: 0.125
sleep 0.5
play 60, sustain: 0.75
sleep 0.5
play 64, sustain: 0.125
sleep 0.5
play 67, sustain: 0.125
sleep 0.5
play 65, sustain: 0.75
sleep 2
play 65, sustain: 0.75
sleep 0.5
play 65, sustain: 0.75
sleep 0.5
play 64, sustain: 0.75
sleep 0.5
play 62, sustain: 0.75
sleep 0.5
play 62, sustain: 0.75
sleep 0.5
play 60, sustain: 0.75
sleep 2
play 64, sustain: 0.125
sleep 0.5
play 64, sustain: 0.5
sleep 0.5
play 56, sustain: 0.125
sleep 0.5
play 60, sustain: 0.75
sleep 0.5
play 64, sustain: 0.75
sleep 0.5
play 71, sustain: 0.125
sleep 0.5
play 69, sustain: 0.75
sleep 2
play 69, sustain: 0.125
sleep 0.5
play 67, sustain: 0.75
sleep 0.5
play 69, sustain: 0.125
sleep 0.75
play 69, sustain: 0.125
sleep 0.5
play 65, sustain: 0.75
sleep 0.5
play 64, sustain: 0.75
sleep 2
```

Результат выполнения задания 1 представлен на рисунке 2.

Задание 2.

Создание музыкального отрывка с помощью линейного алгоритма.

При написании кода использовать стандартную нотацию: С (До), D (Ре), E (Ми), F (Фа), G (Соль), A (Ля), B (Си).

* Соответствующие музыкальные фрагменты к заданиям (аудио-записи) можно скачать на сайте ИНФО: http://infojournal.ru/journals/school/school_02-2020/

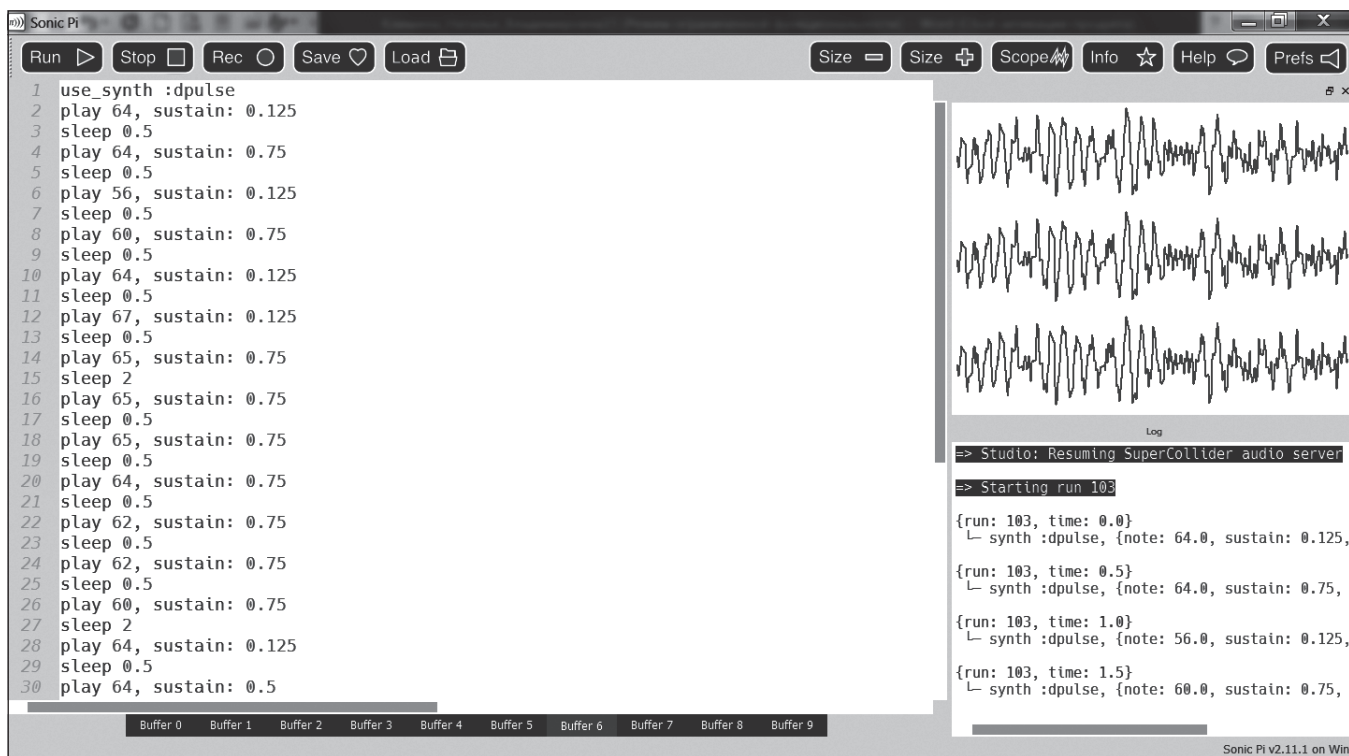


Рис. 2. Задание 1: фрагмент кода мелодии «Изгиб гитары желтой»

Используем команды:

play :d4 — играть ноту в октаве 4;

play :ds4 — играть ноту на полтона выше;

play :db4 — использовать бемоль.

```
play :c4
sleep 0.5
play :d4
sleep 0.5
play :es4
sleep 0.5
play :c4
sleep 0.5
play :c4
sleep 0.5
play :d4
sleep 0.5
play :eb4
sleep 0.5
play :c4
sleep 0.5
```

Результат выполнения задания 2 представлен на рисунке 3. Авторы данного музыкального фрагмента — Владимир Уваров и Роман Федоров, учащиеся ГБОУ СОШ № 4 г. о. Сызрань, 2019 год.

Задание 3.

Создание музыкального отрывка с помощью линейного алгоритма.

При написании кода использовать переменные.

Определим в начале кода переменную, которую будем использовать в параметре затухания проигрывания нот **release** и в длительности паузы:

```
a = 0.6
```

Отсутствие пауз между проигрыванием нот обозначает их одновременное проигрывание.

```
a = 0.6
play 61, release:a/2
play 73, release:a/2
play 68, release:a/2
sleep a/4*3
play 61, release:a/2
play 76, release:a/2
play 68, release:a/2
sleep a/4*3
play 61, release:a/2
play 73, release:a/2
play 68, release:a/2
sleep a/4*2
play 54, release:a/2
play 73, release:a/2
play 61, release:a/2
sleep a/4*3
play 54, release:a/2
play 76, release:a/2
play 61, release:a/2
sleep a/4*3
play 54, release:a/2
play 73, release:a/2
play 61, release:a/2
sleep a/4*2
play 57, release:a/2
play 73, release:a/2
play 64, release:a/2
sleep a/4*3
play 57, release:a/2
play 76, release:a/2
```

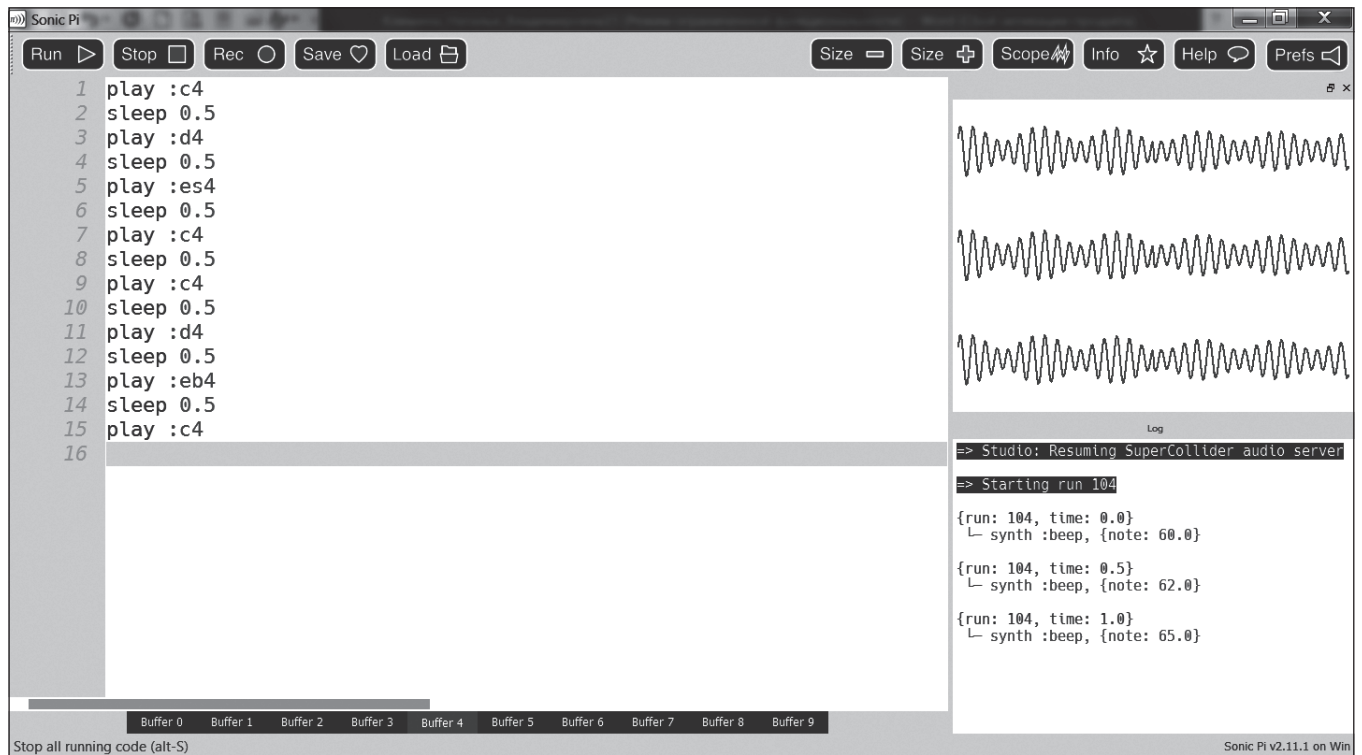


Рис. 3. Задание 2: авторский музыкальный фрагмент

```
play 64, release:a/2
sleep a/4*3
play 57, release:a/2
play 73, release:a/2
play 64, release:a/2
sleep a/4*2
play 59, release:a/2
play 75, release:a/2
play 66, release:a/2
```

```
sleep a/4*3
play 59, release:a/2
play 73, release:a/2
play 66, release:a/2
sleep a/4*3
play 59, release:a/2
play 71, release:a/2
play 66, release:a/2
sleep a/4*2
```

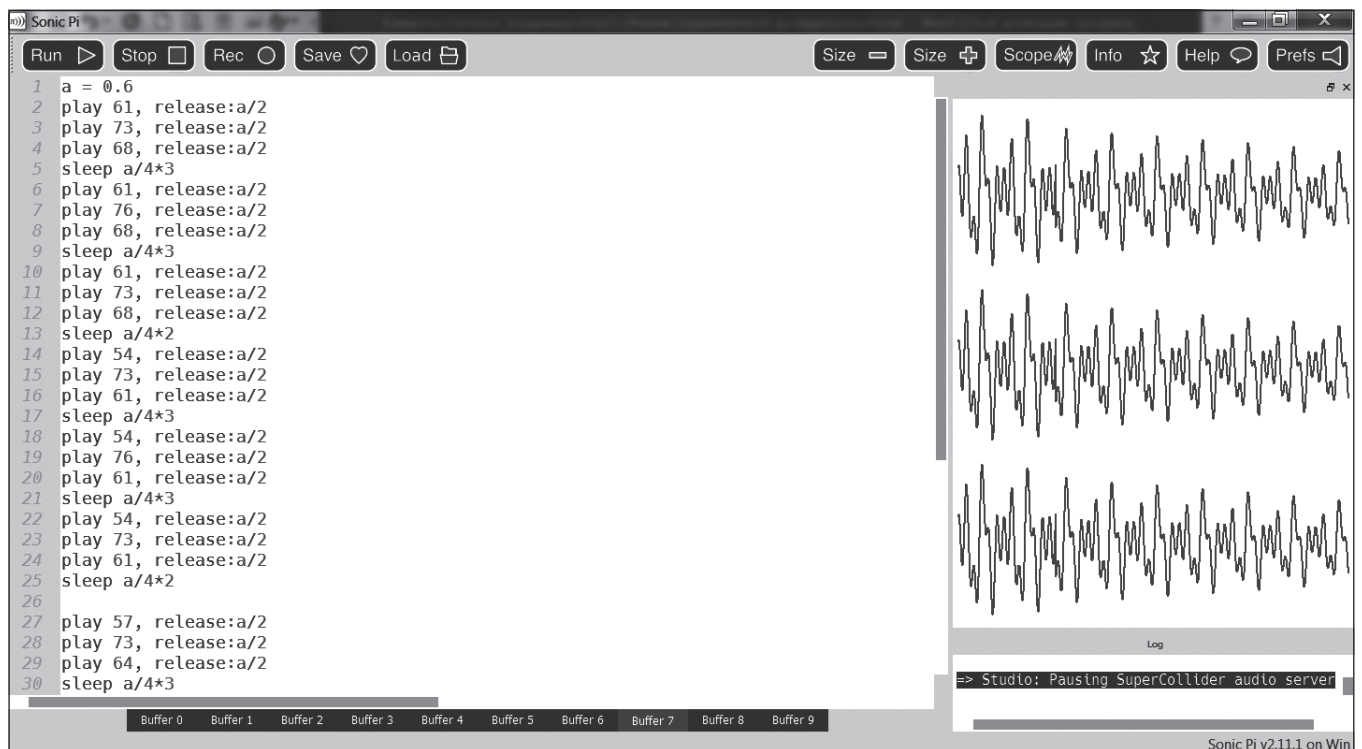


Рис. 4. Задание 3: Авторский музыкальный фрагмент. Применение переменной

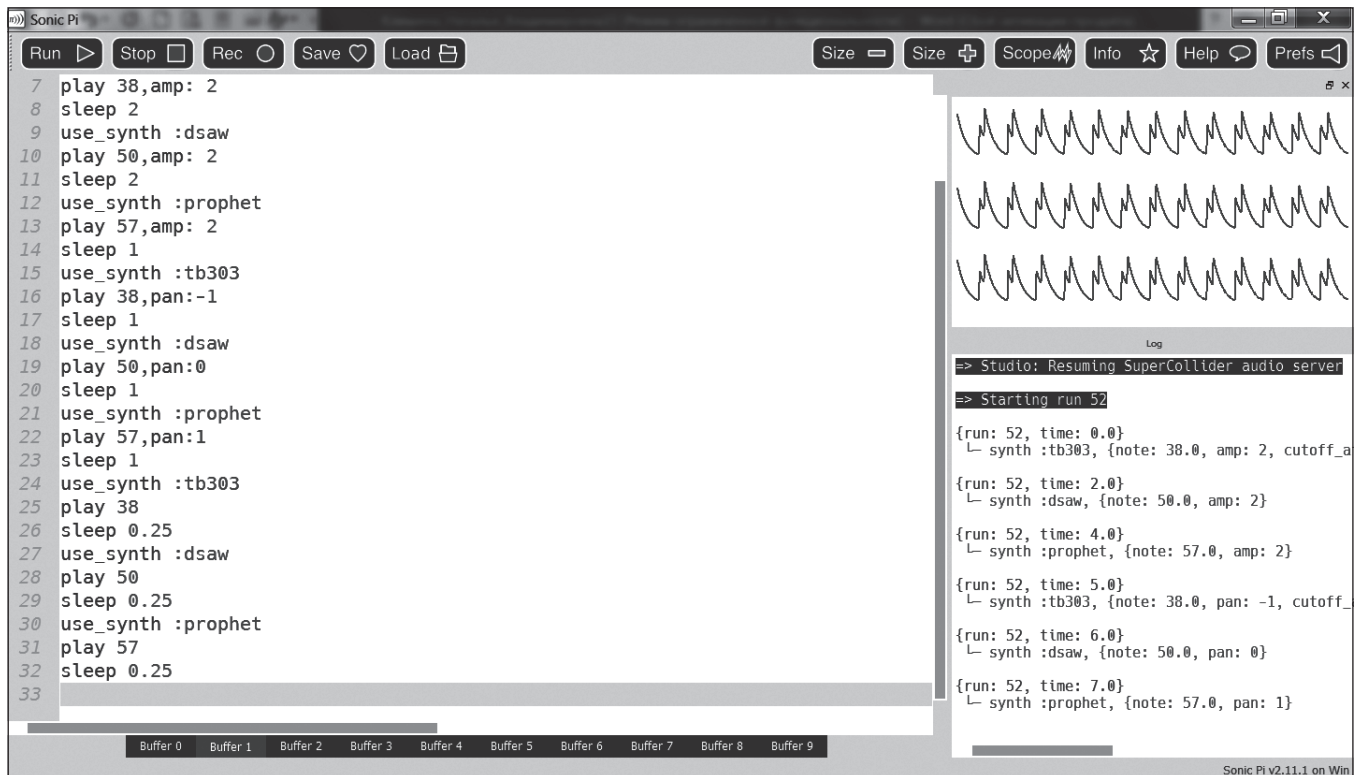


Рис. 5. Задание 4: Переключение синтезаторов, параметры панорамирования звука и увеличение амплитуды

Результат выполнения задания 3 представлен на рисунке 4. Авторы данного музыкального фрагмента — Владимир Уваров и Роман Федоров, учащиеся ГБОУ СОШ № 4 г. о. Сызрань, 2019 год.

Задание 4.

Создание музыкального отрывка с помощью линейного алгоритма.

При написании кода использовать встроенные синтезаторы, панорамирование звука и различные значения амплитуды.

Команда **use_synth** позволяет выбрать любой встроенный инструмент-синтезатор.

Параметр **amp** усиливает громкость звука при увеличении его значения.

Параметр **pan** устанавливает слышимость звука:

- из левой колонки — при отрицательном значении (–1);
- из правой колонки — при положительном значении (1);
- из обеих колонок — по умолчанию или значению 0.

```
use_synth :tb303
play 38, amp: 2
sleep 2
use_synth :dsaw
play 50, amp: 2
sleep 2
use_synth :prophet
play 57, amp: 2
sleep 1
use_synth :tb303
play 38, pan: -1
sleep 1
```

```
use_synth :dsaw
play 50, pan: 0
sleep 2
use_synth :prophet
play 57, pan: 1
sleep 1
use_synth :tb303
play 38
sleep 0.25
use_synth :dsaw
play 50
sleep 0.25
use_synth :prophet
play 57
sleep 0.25
```

Результат выполнения задания 4 представлен на рисунке 5.

3. Алгоритмическая конструкция «цикл» и ее реализация в среде аудиопрограммирования Sonic Pi

Задание 5.

Создание музыкального отрывка с помощью арифметического цикла.

При написании кода использовать вложенные циклы, переменную для амплитуды и изменение ритма.

Команда **use_bpm** устанавливает ритм — количество ударов в минуту. По умолчанию ритм равен 60 ударам в минуту.

В приведенном ниже примере используется модифицированный код из справочника программы [16].

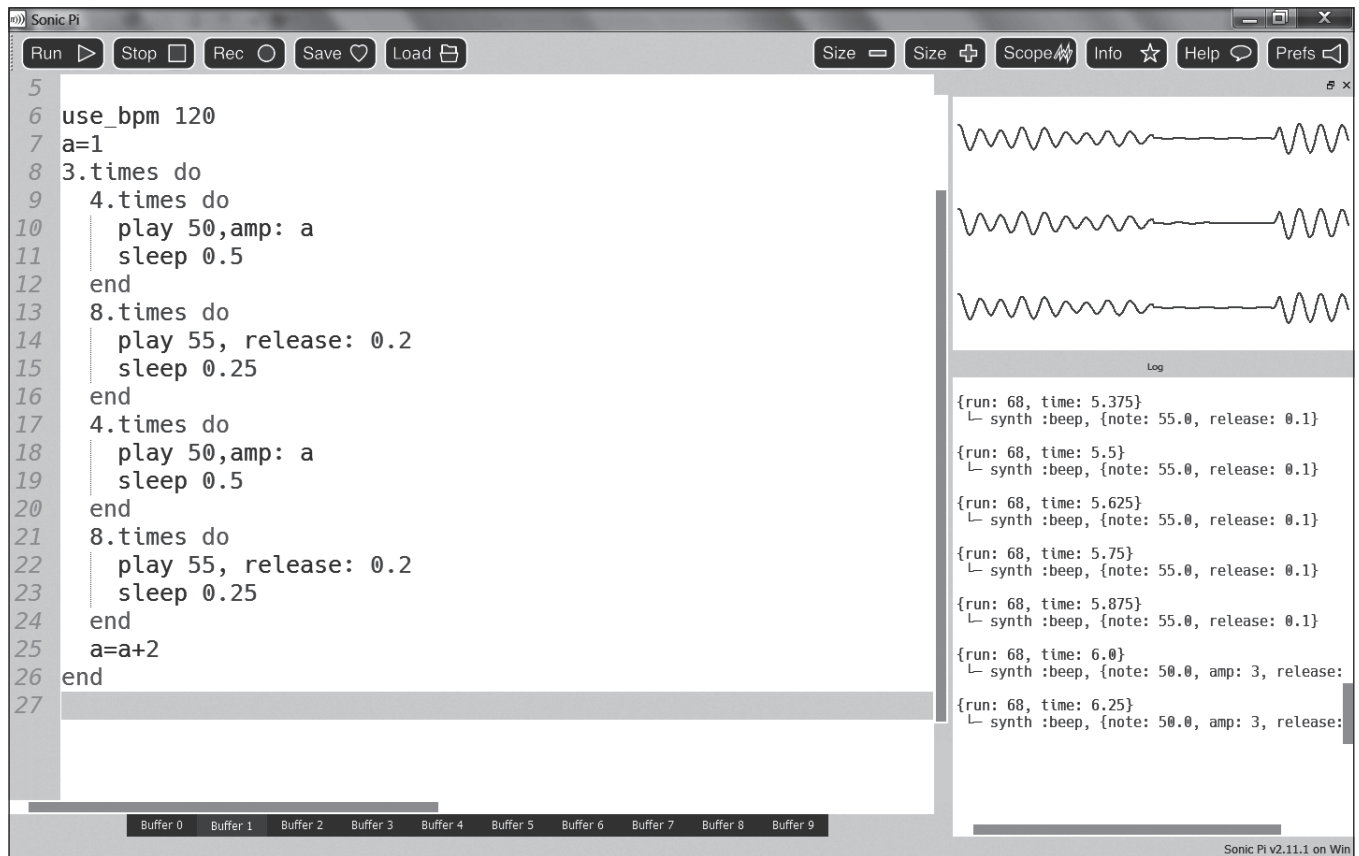


Рис. 6. Задание 5: Использование вложенных арифметических циклов

```
use_bpm 120
a=1
3.times do
  4.times do
    play 50, amp: a
    sleep 0.5
  end
  8.times do
    play 55, release: 0.2
    sleep 0.25
  end
  4.times do
    play 50, amp: a
    sleep 0.5
  end
  8.times do
    play 55, release: 0.2
    sleep 0.25
  end
  a=a+2
end
```

Результат выполнения задания 5 представлен на рисунке 6.

Задание 6.

Создание музыкального отрывка с помощью арифметического цикла.

При написании кода использовать сэмплы [7].

Сэмплы — предварительно записанные звуки и мелодии. Это простой способ разнообразить и украсить

музыку. В Sonic Pi есть множество встроенных сэмплов.

В программе они вызываются командой **sample**:

sample :ambi_choir — женский хор;

sample :ambi_lunar_land — прилунение;

sample :bass_hard_c — басовый долгий звук;

sample :drum_bass_hard — звук барабана;

sample :elec_triangle — звук музыкального треугольника;

sample :loop_amen_full — мелодия барабана с тарелками;

sample :perc_swash — звук свистящего кнута.

Параметр **rate** позволяет менять скорость воспроизведения сэмпла. Чем больше значение параметра, тем быстрее скорость и выше октава. При отрицательном значении параметра мелодия проигрывается от конца к началу.

Все параметры, применяемые к нотам, можно применять и к сэмплам: **amp**, **pan**, **release**, **attack**, **sustain**.

Приведенный ниже код является модифицированным из источника [7].

```
4.times do
  sample :ambi_choir, rate: 3
  sleep 0.5
end
sample :ambi_swoosh, rate: 0.3
4.times do
  sample :misc_crow, amp: 0.5
  sleep 1
end
```

Результат выполнения задания 6 представлен на рисунке 7.

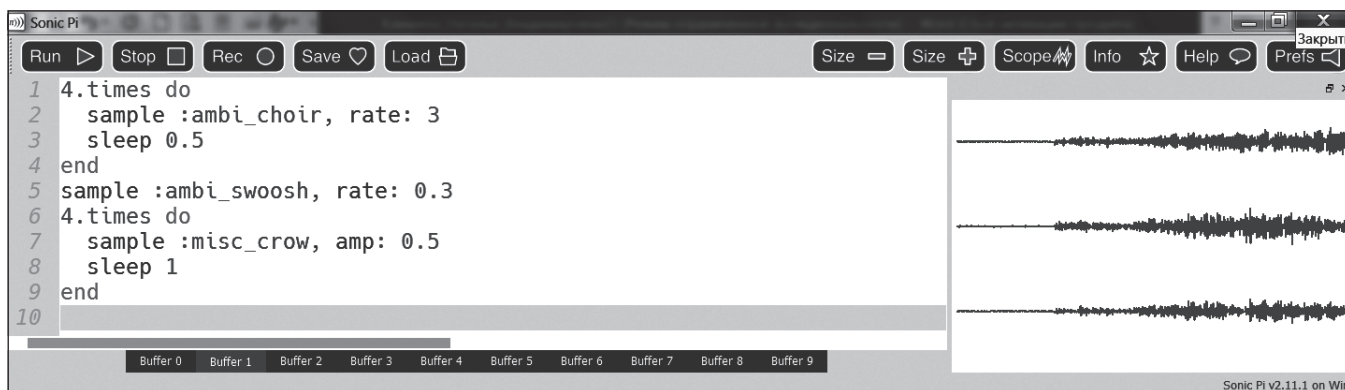


Рис. 7. Задание 6: Арифметические циклы и сэмплы

Задание 7.

Создание музыкального отрывка с помощью бесконечного цикла.

При написании кода использовать сэмплы с параметрами скорости проигрывания и амплитуды.

«Бесконечный цикл» — алгоритмическая конструкция, применяемая в среде аудиoprogrammирования Sonic Pi. Является стандартной именно для этой среды.

```

loop do
  sample :loop_amen
  sleep 1.8
  sample :ambi_lunar_land, amp: 1.5, rate: 3
end

```

Результат выполнения задания 7 представлен на рисунке 8.

4. Алгоритмическая конструкция «ветвление» (полная форма) и ее реализация в среде аудиoprogrammирования Sonic Pi. Использование ветвления в цикле

Задание 8.

Создание музыкального отрывка с помощью бесконечного цикла.

При написании кода использовать сэмплы и условное выражение [12].

Используемая функция `one_in(n)` выдает случайное истинное значение с вероятностью $1 : n$. В зависимости от выбранного значения проигрывается тот или иной участок кода.

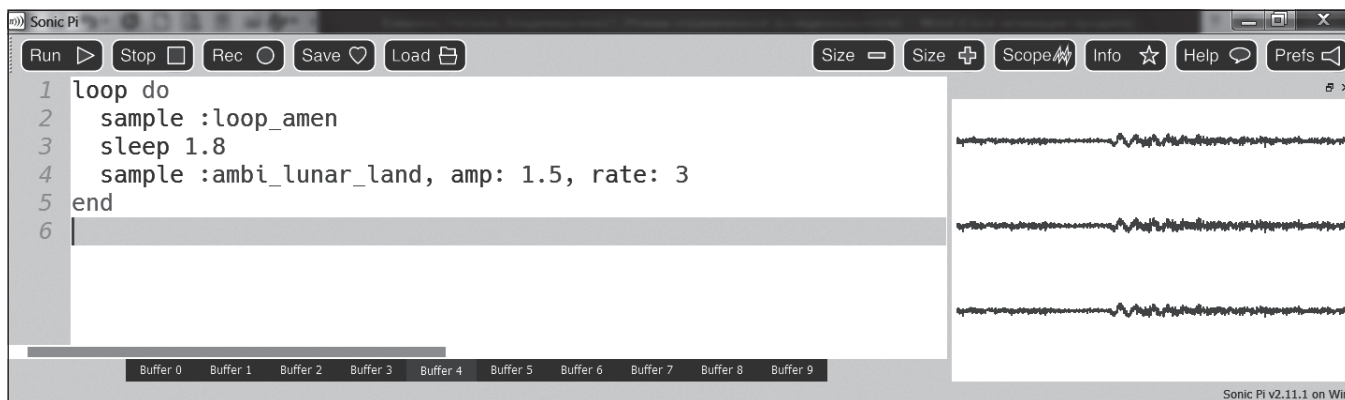


Рис. 8. Задание 7: Бесконечный цикл

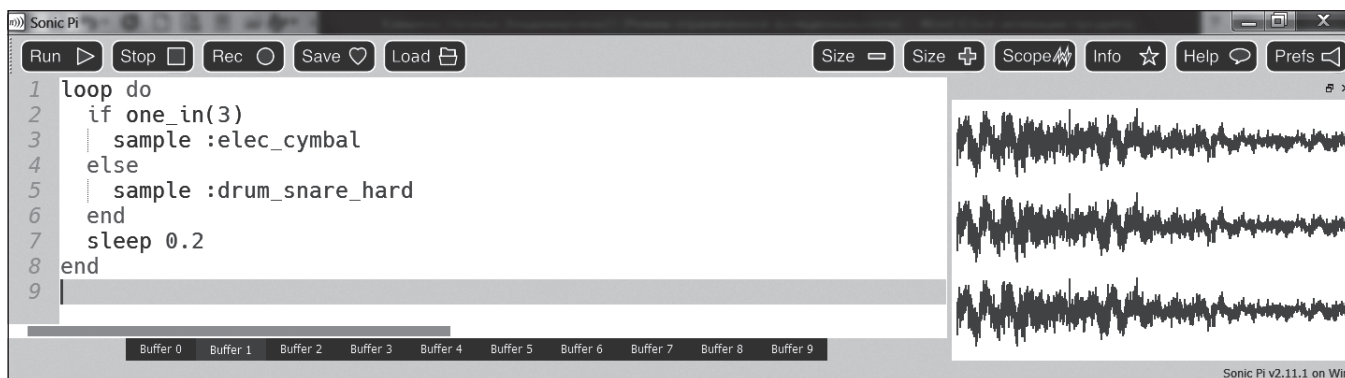


Рис. 9. Задание 8: Бесконечный цикл с условным выражением



Рис. 10. Задание 9: Использование структуры данных «список»

```
loop do
  if one_in(3)
    sample :elec_cymbal
  else
    sample :drum_snare_hard
    sleep 0.2
  end
end
```

Результат выполнения задания 8 представлен на рисунке 9.

5. Реализация структур данных в среде аудиoprogramмирования Sonic Pi.

Использование случайных чисел

Задание 9.

Создание музыкального отрывка с помощью арифметического цикла с использованием структуры данных «список» [7].

В теле команды паттерна указываем список нот через запятую, далее идет список пауз между нотами.

Параметр **r** — проигрывание тишины.

В представленном ниже фрагменте — модификация мелодии из источника [7].

```
use_bpm 120
use_synth :dpulse
3.times do
  play_pattern_timed [:E, :B3, :C, :D, :C, :B3],
    [1, 0.5, 0.5]
  play_pattern_timed [:A3, :A3, :C, :E, :D, :C],
```

```
  [1, 0.5, 0.5]
  play_pattern_timed [:B3, :C, :D, :E],
    [1.5, 0.5, 1, 1]
  play_pattern_timed [:C, :A3, :A3], [1, 1, 2]
  play_pattern_timed [:r, :D, :F, :A, :G, :F],
    [0.5, 1, 0.5, 1, 0.5, 0.5]
  play_pattern_timed [:E, :C, :E, :D, :C],
    [1.5, 0.5, 1, 0.5, 0.5]
  play_pattern_timed [:B3, :B3, :C, :D, :E],
    [1, 0.5, 0.5, 1, 1]
  play_pattern_timed [:C, :A3, :A3], [1, 1, 2]
  use_bpm current_bpm + 40
end
```

Результат выполнения задания 9 представлен на рисунке 10.

Задание 10.

Создание музыкального отрывка с использованием случайных чисел.

Функция **rrand** выдает случайное число из интервала (целое или действительное), функция **rrand_i** — только целое из интервала. Параметр **cutoff** делает срез (отсечку) определенной частоты ноты [3].

```
use_synth :bnoise
use_bpm 180
loop do
  play rrand_i(50,70), cutoff: 70
  sleep rrand(0.1, 1)
end
```

Результат выполнения задания 10 представлен на рисунке 11.

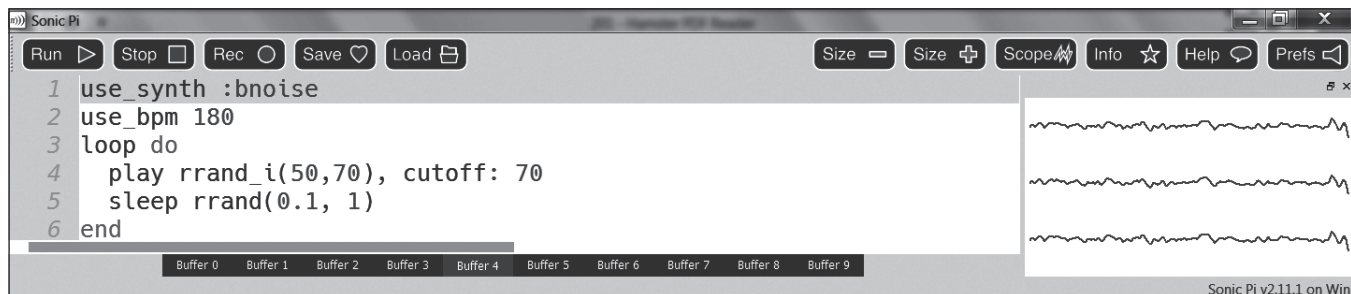


Рис. 11. Задание 10: Бесконечный цикл и случайные числа

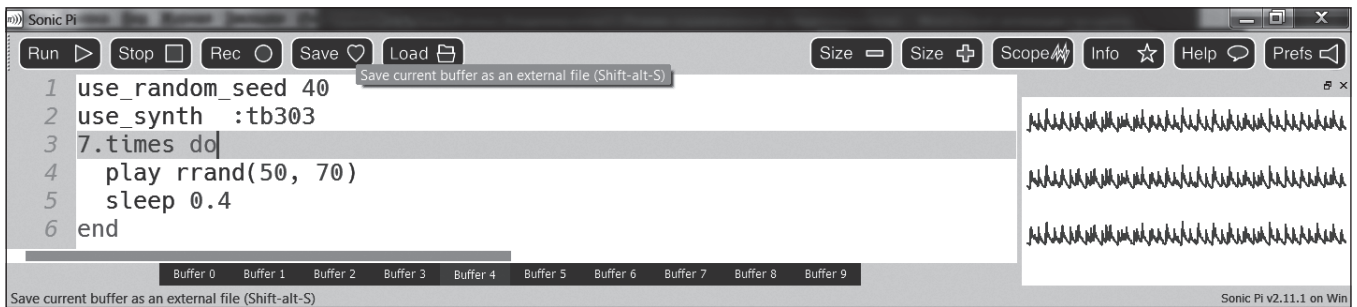


Рис. 12. Задание 11: Случайные числа с указанной начальной точкой выборки

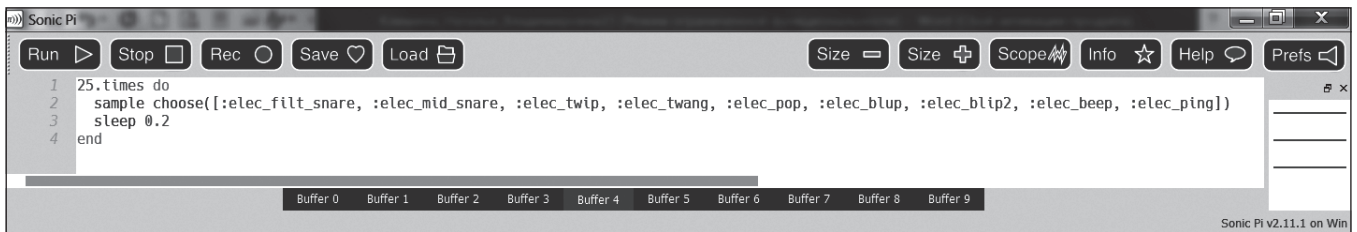


Рис. 13. Задание 12: Использование функции выбора из списка

Задание 11.

Создание музыкального отрывка с использованием случайных чисел с заданной начальной точкой.

Команда `use_random_seed` меняет начальную точку выборки случайных чисел на указанное значение (сдвиг отсчета выборки).

```
use_random_seed 40
use_synth :tb303
7.times do
  play rrand(50,70)
  sleep 0.4
end
```

Сравните с:

```
use_synth :tb303
7.times do
  play rrand(50,70)
  sleep 0.4
end
```

Результат выполнения задания 11 представлен на рисунке 12.

Задание 12.

Создание музыкального отрывка со случайным проигрыванием из списка [16].

Функция `choose` позволяет проигрывать ноты, сэмплы и т. п. в случайном порядке, если они оформлены в список проигрывания. Это еще один вариант реализации структуры данных «список».

В приведенном ниже фрагменте используется модифицированный код из справочника среды Sonic Pi.

```
25.times do
  sample choose ([:elec_filt_snare,
    :elec_mid_snare, :elec_twip, :elec_twang,
    :elec_pop, :elec_blup, :elec_blip2,
    :elec_beep, :elec_ping])
  sleep 0.2
end
```

Результат выполнения задания 12 представлен на рисунке 13.

6. Реализация одновременного проигрывания нескольких мелодий. Потоки выполнения

Задание 13.

Создание музыкального отрывка с одновременным проигрыванием двух мелодий.

Использовать поток выполнения.

Записав мелодию в поток выполнения `in_thread`, мы обеспечим ее одновременное проигрывание с мелодией второго цикла.

Функция `chord` проигрывает аккорды во второй мелодии [10, 14].

```
in_thread do
  loop do
    sample :drum_heavy_kick, rate: 0.75, amp: 2
    sleep 0.3
    sample :drum_heavy_kick
    sleep 0.2
  end
end
loop do
  use_synth :subpulse
  sample :elec_soft_kick, rate: rrand(0.5, 1)
  play choose(chord(:e3, :minor)), release: 0.1,
    cutoff: rrand(60,80)
  sleep 0.25
end
```

Результат выполнения задания 13 представлен на рисунке 14.

Задание 14.

Создание музыкального отрывка с одновременным проигрыванием трех мелодий.

Использовать два потока выполнения.

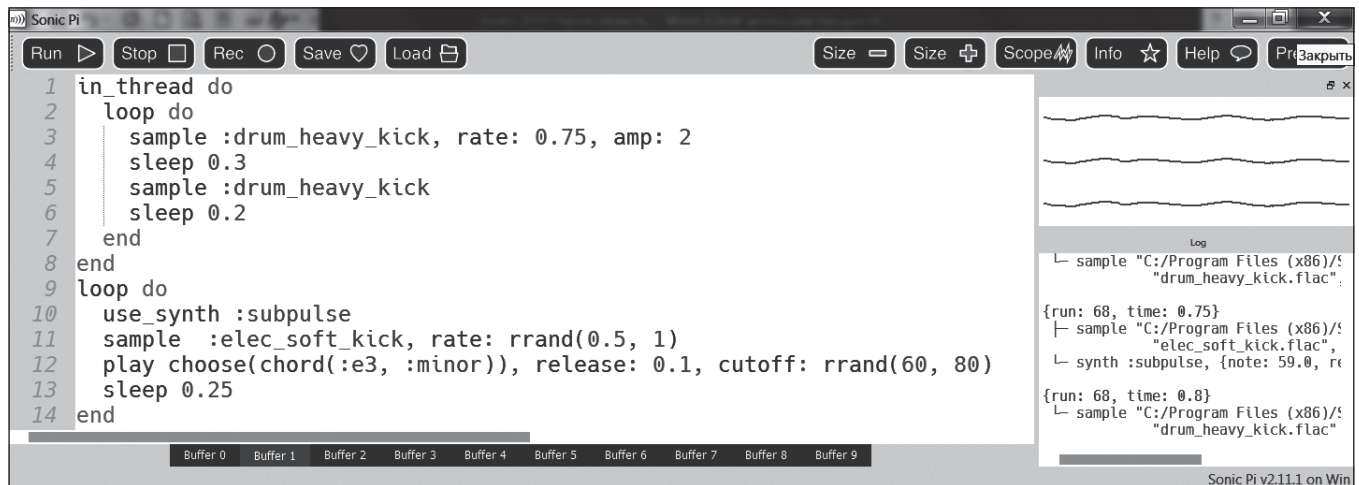


Рис. 14. Задание 13: Одновременное проигрывание двух мелодий

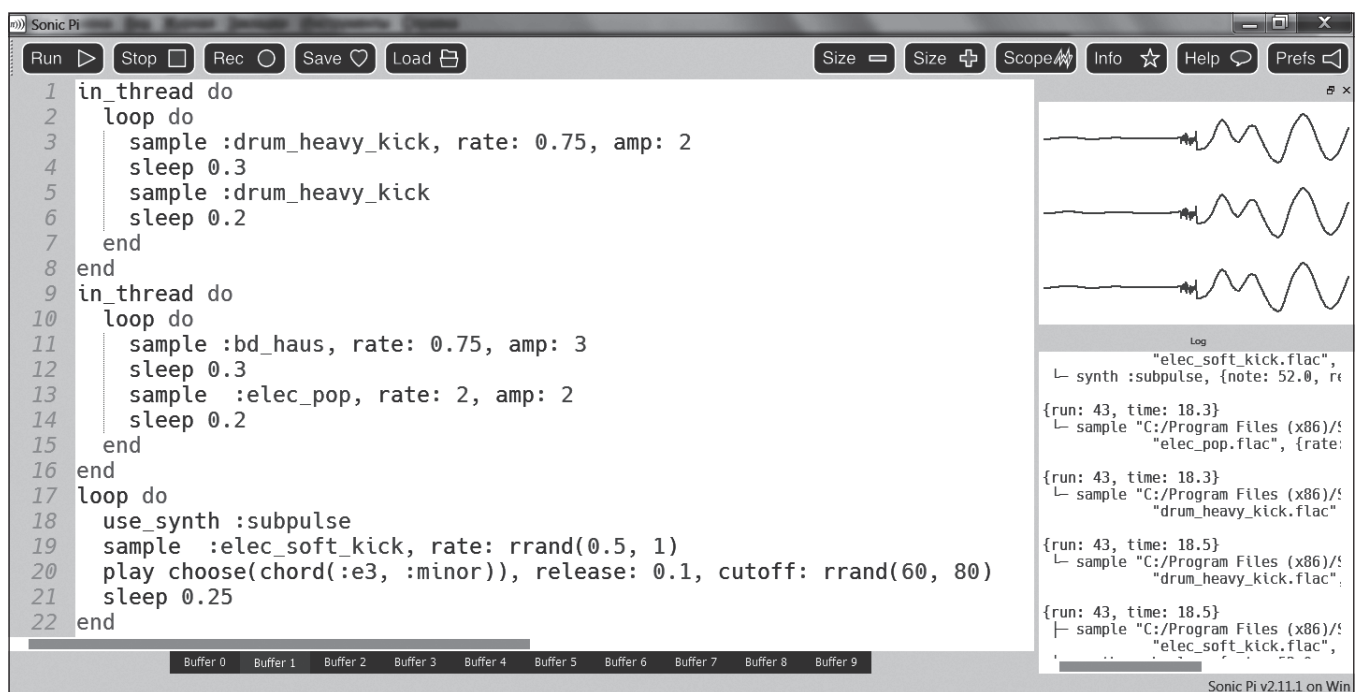


Рис. 15. Задание 14: Проигрывание трех мелодий одновременно

```
in_thread do
  loop do
    sample :drum_heavy_kick, rate: 0.75, amp: 2
    sleep 0.3
    sample :drum_heavy_kick
    sleep 0.2
  end
end
in_thread do
  loop do
    sample :bd_haus, rate: 0.75, amp: 3
    sleep 0.3
    sample :elec_pop, rate: 2, amp: 2
    sleep 0.2
  end
end
loop do
  use_synth :subpulse
  sample :elec_soft_kick, rate: rrand(0.5, 1)
  play choose(chord(:e3, :minor)), release: 0.1, cutoff: rrand(60, 80)
  sleep 0.25
end
```

```
play choose(chord(:e3, :minor)), release: 0.1,
cutoff: rrand(60,80)
sleep 0.25
end
```

Результат выполнения задания 14 представлен на рисунке 15.

7. Реализация подпрограмм в среде аудиопрограммирования Sonic Pi

Задание 15.

Создание музыкального отрывка с использованием функции пользователя [7].

Функция **define** с блоком **do-end** определяет функцию, созданную пользователем. В теле программы функцию пользователя вызывают по имени необходимое количество раз.

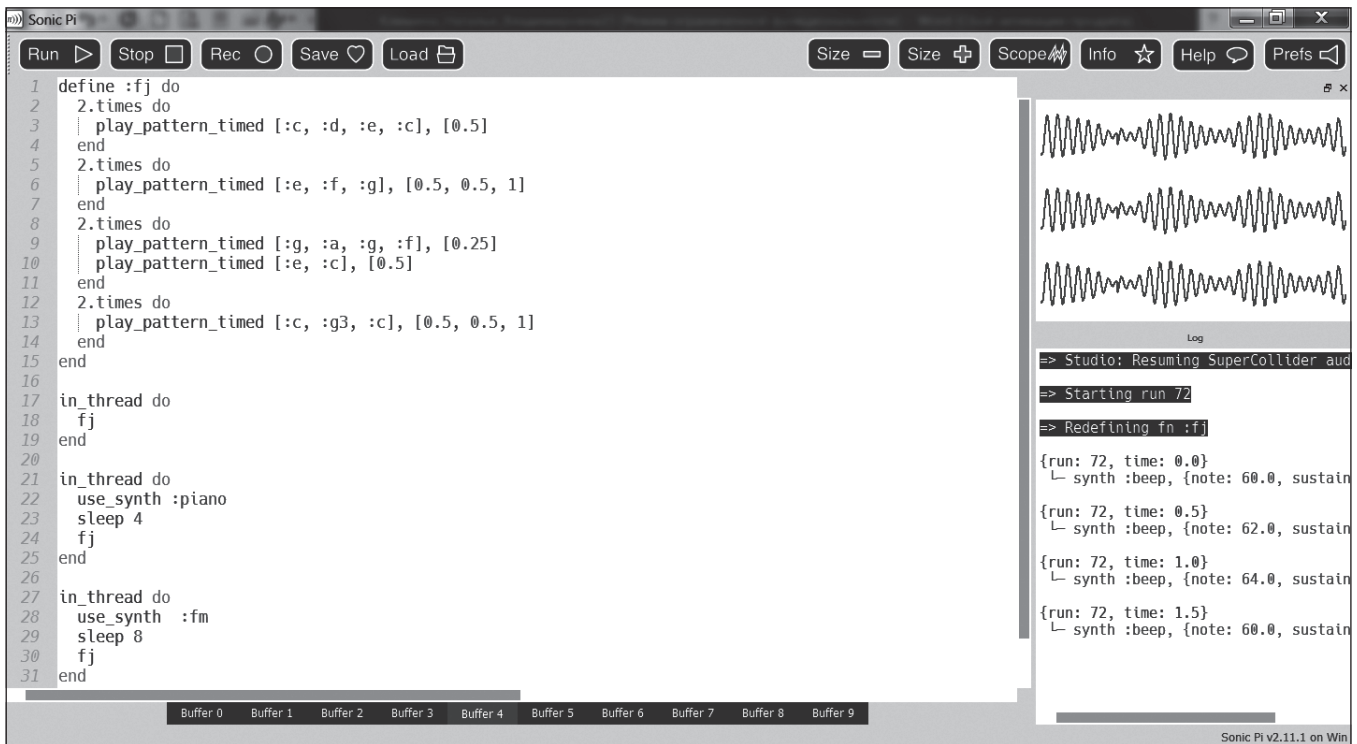


Рис. 16. Задание 15: Использование функции пользователя

В приведенном ниже фрагменте использован модифицированный код из источника [7].

```
define :fj do
  2.times do
    play_pattern_timed [:c, :d, :e, :c], [0.5]
  end
  2.times do
    play_pattern_timed [:e, :f, :g], [0.5, 0.5, 1]
  end
  2.times do
    play_pattern_timed [:g, :a, :g, :f], [0.25]
    play_pattern_timed [:e, :c], [0.5]
  end
  2.times do
    play_pattern_timed [:c, :g3, :c], [0.5, 0.5, 1]
  end
end

in_thread do
  fj
end

in_thread do
  use_synth :piano
  sleep 4
  fj
end

in_thread do
  use_synth :fm
  sleep 8
  fj
end

in_thread do
  use_synth :tri
  sleep 12
  fj
end
```

Результат выполнения задания 15 представлен на рисунке 16.

8. Реализация звуковых эффектов в среде аудиoprogramмирования Sonic Pi

Задание 16.

Создание музыкального отрывка с использованием эффекта реверберации [5].

Подключение плагина эффектов происходит с помощью функции **with_fx**.

В приведенном ниже фрагменте подключен эффект реверберации. Этот эффект позволяет сделать звук объемным. Параметр **room** позволяет установить максимальную объемность звука 1.

В примере также используется «живой» цикл **live_loop** имя **do-end**. Его особенность в том, что для перезапуска кода достаточно еще раз нажать **Run**, в то время как обычный код требует для перезапуска сначала остановить выполнение предыдущего кода. Это свойство используется музыкантами для редактирования кода «на лету».

Функция **scale** позволяет проигрывать гаммы.

Функция **sound.choose** вызывает случайное проигрывание элементов списка в переменной **sound**.

В примере использован модифицированный код из источника [5].

```
with_fx :reverb, room: 1 do
  live_loop :melody do
    use_synth :beep
    use_random_seed 66678
    sound= (scale :g3, :major_pentatonic, num_
octaves: 3)
    16.times do
```

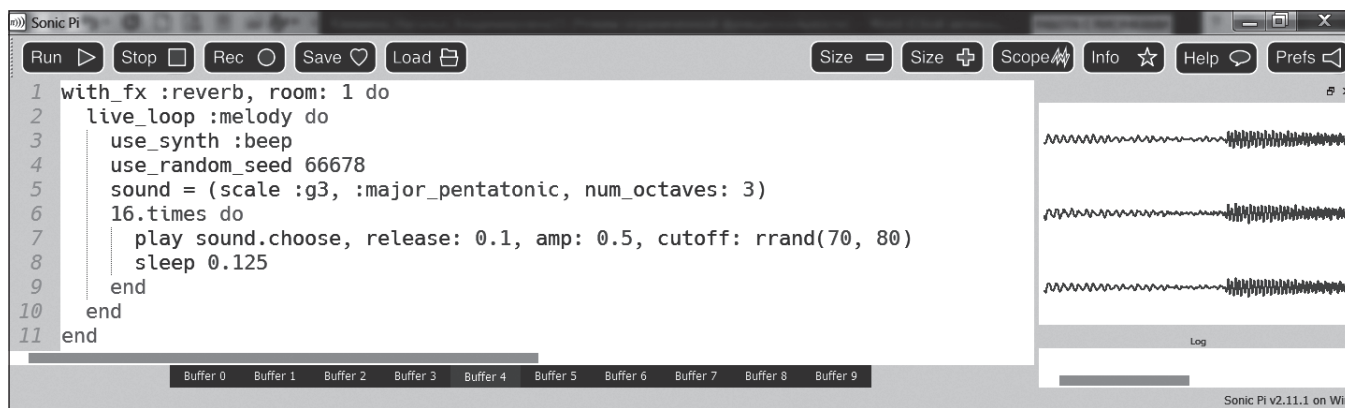


Рис. 17. Задание 16: Применение эффекта реверберации

```

      play sound.choose, release: 0.1, amp: 0.5,
cutoff: rand(70,80)
      sleep 0.125
    end
  end
end
end

```

Результат выполнения задания 16 представлен на рисунке 17.

Задание 17.

Создание музыкального отрывка с использованием эффекта «эхо» [12].

Параметр **phase** устанавливает длительность эха в долях такта.

В приведенном ниже фрагменте использован модифицированный код из источника [12].

```

live_loop :trance do
  sample :bd_gas
  with_fx :echo, phase: 0.125 do
    sample :drum_cymbal_soft, sustain: 0,
release: 0.1
    sleep 0.5
  end
end
live_loop :bas do
  use_synth :fm
  synth :mod_saw, note: :e1, release: 1, cutoff:

```

```

80
  sleep 2
end

```

Результат выполнения задания 17 представлен на рисунке 18.

Задание 18.

Создание музыкального отрывка с использованием эффекта «дисторшн» [9].

Дисторшн — это эффект искажения звука, связанный с изменением амплитуды [2]. Без этого эффекта невозможно себе представить рок-музыку. Именно дисторшн придает ей глубину, силу и красоту звучания.

В приведенном примере используется структура данных «кольцо» — **ring**. Кольцо представляет собой список с извлечением элементов по кругу.

Функция **tick**, используемая в примере, служит для увеличения номера проигрываемого элемента кольца.

В примере используется модифицированный код из источника [9].

```

with_fx :distortion do
  live_loop :habanera do
    play (ring :d, :r, :r, :a, :f5, :r, :a, :r).
tick
    sleep 0.25
  end
end

```

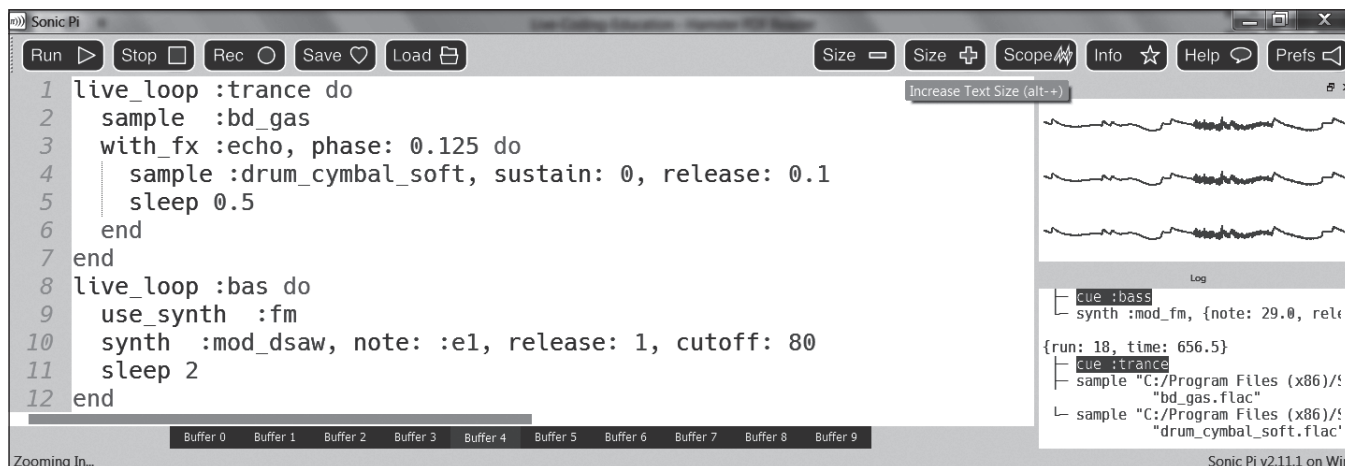


Рис. 18. Задание 17: Применение эффекта «эхо»

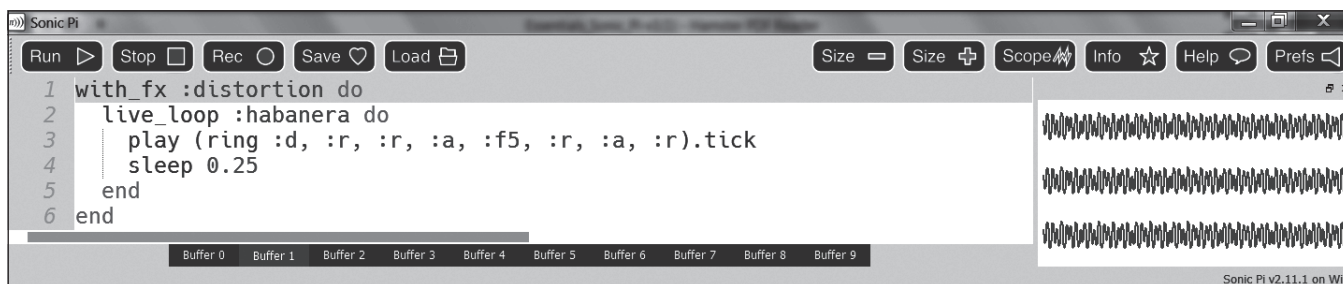


Рис. 19. Задание 18: Применение эффекта «дисторшн»

Результат выполнения задания 18 представлен на рисунке 19.

В приведенных выше заданиях отражены основные концепции создания электронной музыки в среде аудиoprogrammирования Sonic Pi. Рассмотрены основные возможности среды для создания алгоритмов мелодий.

Появление новых возможностей создания музыки с помощью алгоритмического подхода позволяет запрограммировать мелодии с использованием различных музыкальных инструментов, а также мелодии, имитирующие игру целого ансамбля или оркестра [13].

Список использованных источников

1. Алгоритмическая композиция. <https://algorithmiccomposition.ru>
2. Дисторшн и искажения звука в студии. <http://www.master-skills.ru/articles/mixing/118-dist>
3. Паундер Л. Ruby: Сочиним случайную музыку // LinuxFormat. 2016. № 4 (208). С. 90–91.
4. Паундер Л. Sonic Pi // LinuxFormat. 2014. № 7 (185). С. 17.

5. Паундер Л. Sonic Pi: Ну-ка, пошумим! // LinuxFormat. 2015. № 10 (201). С. 92–95.
6. Программирование музыки на Sonic Pi. <http://www.nofollow.ru/video.php?c=etuykXmZZxg>
7. Проекты на Sonic Pi. <https://projects.raspberrypi.org/en/projects?interests%5B%5D=music&software%5B%5D=sonic-pi>
8. Audio programming with Sonic Pi! https://www.deviceplus.com/how-to/raspberrypi-guide/raspberrypi_entry_017/
9. Code Music with Sonic Pi. https://www.raspberrypi.org/magpi-issues/Essentials_Sonic_Pi-v1.pdf
10. Generating Music with Sonic Pi. <https://dev.to/jessicagarson/generating-music-with-sonic-pi-57od>
11. Getting Started With Sonic Pi. <https://projects.raspberrypi.org/en/projects/getting-started-with-sonic-pi>
12. Live Coding Education. <https://sonic-pi.net/files/articles/Live-Coding-Education.pdf>
13. Sonic Pi: как программировать музыку на компьютере. <http://news.ifmo.ru/ru/education/trend/news/8397/>
14. Sonic-PI: Sound synth application for learning coding and music. <http://inspiredtoeducate.net/inspiredtoeducate/sonic-pi-sound-synth-application-for-learning-coding-and-music/>
15. Sonic Pi — The Live Coding Music Synth for Everyone. <https://sonic-pi.net/>
16. Tutorial. <https://sonic-pi.net/tutorial.html>

НОВОСТИ

«Роббо клубы» обучат программированию и робототехнике онлайн

Международная сеть школ робототехники «Роббо клуб» запустила дистанционные курсы по программированию, 3D-моделированию, схемотехнике и робототехнике. В компании считают, что это позволит сети удерживать учеников и привлечь новых.

Первые дистанционные курсы по программированию, 3D-моделированию и схемотехнике для разных возрастных категорий рассчитаны на три месяца. Для изучения этих дисциплин не предусмотрено использование оборудования: для ребенка нужно организовать рабочее место за компьютером и доступ в интернет. Дети могут заниматься в группах до пяти человек или индивидуально.

Далее ученикам предложат два варианта для продолжения обучения. Изучать робототехнику начнут те, кто купит робототехнический набор — «Роббо робоплатформу» и «Роббо лабораторию». Остальные смогут перейти на следующую ступень изучения программирования, 3D-моделирования и схемотехники. Кроме этого, ребята смогут подготовиться к участию в Международной Scratch-олимпиаде по креативному программированию и в «Роббо фестивале», которые традиционно проводятся онлайн.

Для организации дистанционного обучения в «Роббо клубах» после тестирования разных вариантов выбрали

платформу bluejeans. Она продемонстрировала лучшие характеристики по удобству для преподавателя и учеников, стабильности системы и стоимости. Ранее с помощью bluejeans специалисты методического отдела «Роббо клубов» проводили онлайн-обучение преподавателей сети.

«80 % учеников, которые из-за соблюдения профилактических мер перестали посещать кружки, продолжили заниматься онлайн, — отметила директор по франчайзингу сети «Роббо клуб» Екатерина Экало. — Мы видим интерес к формату дистанционного обучения в наших клубах как со стороны своих учеников, так и новых ребят. Мы технически готовы принять тех детей, чьи секции закрылись или не смогли оперативно предложить дистанционный формат обучения».

По словам Екатерины Экало, запуск дистанционного обучения не потребовал дополнительных денежных инвестиций. Были задействованы имеющиеся человеческие и технические ресурсы.

«В разработку наших образовательных методик мы вложили несколько миллионов долларов. Они применяются при обучении более 50 тыс. детей в 18 странах мира. Поэтому мы смогли оперативно адаптировать наши программы для онлайн-формата и запустить дистанционное обучение», — сказала Екатерина Экало.

(По материалам CNews)