ГЕНЕРАЦИЯ МУЗЫКАЛЬНЫХ НОТ С ИСПОЛЬЗОВАНИЕМ ЯЗЫКА ПРОГРАММИРОВАНИЯ TYPESCRIPT

Каминский А.В.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Можей Н.П. – канд. физ.-мат. наук

Описывается генерация музыкальных звуков при помощи расширенного алгоритма Карплюса-Стронга и ее применение на примере акустической гитары

Генерация сигналов - это процесс создания электрических или электромагнитных волн, которые могут быть использованы в различных устройствах, таких как радио, телевизоры, компьютеры и телефоны, для передачи информации, синхронизации устройств или генерации звуков и изображений.

Существует множество способов воспроизведения звуков музыкальных инструментов. Одним из способов является звукозапись различных аккордов, нот и последующее их воспроизведение. Однако такой подход не лишен недостатков, ключевым из которых является многократное чтение множества файлов, хранение их в памяти. Гораздо менее затратно генерировать звуки при помощи API языков программирования, а также различных алгоритмов.

Для решения задачи реализации генерации звуков будем использовать язык программирования TypeScript, а также фреймворк Angular. В качестве примера рассмотрим генерацию звуков гитары.

TypeScript предоставляет отличный интерфейс взаимодействия для воспроизведения звуков. Основным классом является класс AudioContext [1]. Объект класса можно представить как граф связанных между собой AudioNode. Этот интерфейс обеспечивает возможность управлять созданием узлов, составляющих AudioContext, а также регулировать обработку и декодирование звука. Необходимо создать AudioContext перед началом работы с аудио, поскольку все процессы обработки звука происходят внутри него.

После создания экземпляра класса AudioContext необходимо вызвать функцию createBuffer() для получения буфера аудио. В качестве параметров указывается количество каналов, используемых при проигрывании аудио, размер буфера, а также частота дискретизации.

Для получения доступа каналу используется функция getChannelData(), возвращающая объект типа Float32Array. Этот тип представляет собой массив дробных чисел, нормализованных в диапазоне от -1 до 1, представляющий сам звуковой сигнал.

Для генерации звуков гитарной струны будем использовать расширенный алгоритм Карплюса-Стронга, основывающийся на создании белого шума с последующим применением цифровых фильтров [2]. Схематическое представление алгоритма приведено на рисунке 1.

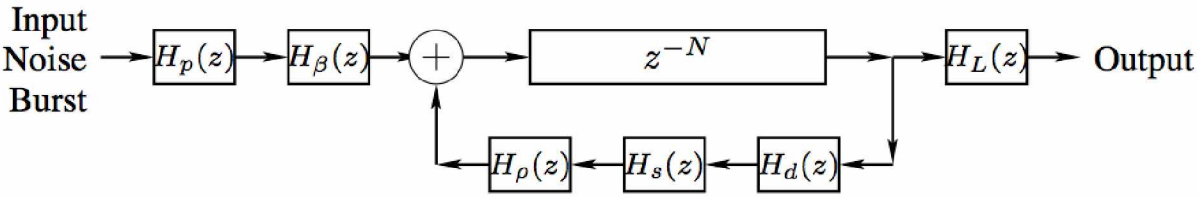

Рисунок 1 – схема расширенного алгоритма Карплюса-Стронга

После генерации шума в начале алгоритма, сигнал преобразуется при помощи ряда различных цифровых фильтров, основой которых служит сдвиг сигнала во времени. Фильтр низких частот представлен формулой:

$$H_p(z) = \frac{1-p}{1-pz^{-1}} \quad (1),$$

где p – коэффициент, влияющий на жесткость звука, а z – смещение значение сигнала во времени.

Гребенчатый фильтр представляется формулой:

$$H_\beta(z) = 1 - z^{-int\left(\beta * N + \frac{1}{2}\right)} \quad (2),$$

где β – место щипка струны, относительно всей струны, N – количество сгенерированных значений шума.

Демпинг-фильтр:

$$H_d(z) = (1 - S) + Sz^{-1} \quad (3),$$

где S - коэффициент натяжения. В оригинальном алгоритме используется значение 0.5. Фильтр пропускания по жесткости струн:

$$H_s(z) = z^{-K}\frac{\sim A(z)}{A(z)} \quad (4),$$

где $K \geq 0$, $A(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_N z^{-N}$. $\sim A(z)$ представляет собой функцию $A(z)$, у которой коэффициенты a_i расставлены в обратном порядке $\sim A(z) = a_N + a_{N-1} z^{-1} + a_{N-2} z^{-2} + \cdots + z^{-N}$.
Акустический фильтр первого порядка со струнной настройкой:

$$H_\rho(z) = \frac{C + z^{-1}}{1 + Cz^{-1}} \quad (5),$$

где C ∈ $[0, 1)$.

Фильтр низких частот с динамическим уровнем:

$$H_L(z) = \frac{\omega}{1+\omega}\frac{1+z^{-1}}{1-\frac{1-\omega}{1+\omega}z^{-1}} \quad (6),$$

где $\omega = \pi f / F_s$, где f – частота сигнала, а F_s – частота дискретизации.

Пример реализации части расширенного алгоритма Карплюса-Стронга, возвращающий массив семплов, представлен на рисунке 2. В результате реализации были задействованы все фильтры, кроме фильтра пропускания по жесткости струн (string-stiffness allpass filter) $H_s(z)$ с целью уменьшения времени выполнения алгоритма.

```
karplusStrong(rate: number): number[] {
  const noise = this.noise( n: this.N / rate);
  const hps = this.hP(noise,  p: 0.5);
  const hBs = this.hB(hps, noise.length,  beta: 0.5);
  const samples = [...hBs];
  for (let i = noise.length; i < this.N; i++) {
    samples.push(0)
    samples[i] = this.hZ(samples, noise.length, i);
    samples[i] = this.hD(samples, i,  s: 0.01);
    samples[i] = this.hRo(samples, i,  c: 0.4);
  }
  return this.hL(samples,  l: 0.32,  w: Math.PI * rate / this.N)
}
```

Рисунок 2 – пример реализации расширенного алгоритма Карплюса-Стронга

В результате, пропуская шум через последовательность фильтров, мы получаем звуки гитарной струны необходимой частоты. С целью улучшения звука можно добавлять фильтры для получения различных эффектов таких как хорус, задержка либо перегрузка.

Для решения поставленной задачи на основе описанного выше алгоритма создано программное средство, позволяющее создавать музыкальную табулатуру, проверять уникальность музыкального произведения и размещать готовые музыкальные произведения на площадке для пользователей. Данное приложение могут использовать как композиторы с целью сочинения музыкальных произведений, так и музыканты-любители для воспроизведения произведений по нотам.

Список использованных источников:

1.AudioContext. [Электронный ресурс] – Режим доступа: https://developer.mozilla.org/en-US/docs/Web/API/AudioContext.

2.Extended Karplus-Strong Algorithm [Электронный ресурс] – Режим доступа: https://ccrma.stanford.edu/realsimple/faust_strings/Extended_Karplus_Strong_Algorithm.html.

3.Extensions of the Karplus-Strong Plucked-String algorithm [Электронный ресурс] – Режим доступа:. http://musicweb.ucsd.edu/~trsmyth/papers/KSExtensions.pdf