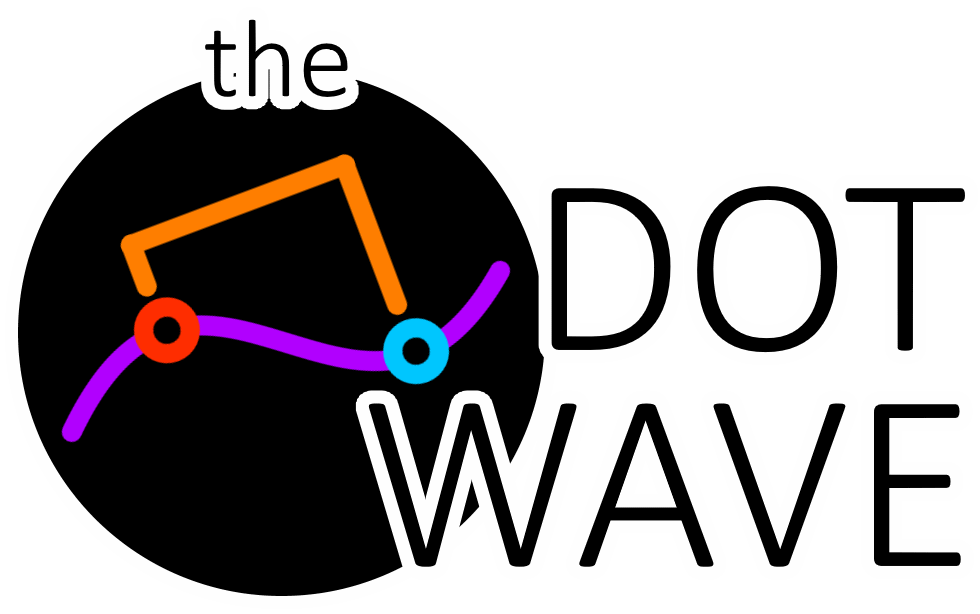
LIMIDE DEV

DotWave 0.1





vk.com/limide

vk.com/jkulvich

@jkulvich

# Содержание

[Содержание 2](#_Toc471501732)

[Описание полей, методов и классов 3](#_Toc471501733)

[Примеры 5](#_Toc471501734)

# 

# Описание полей, методов и классов

В пространстве имён LMDDotWave находятся 3 класса:

**- DotWave**

**- DotWaveInfo**

**- DotWaveWorker**

Если кратко, то в задачи этих классов входит следующее:

**DotWave** – Проверка корректности набора байт .wav файла, предоставление байт в удобной форме (DotWaveInfo), а так же обратная упаковка в байт-массив который может быть непосредственно записан или воспроизведён .wav плеером.

**DotWaveInfo** – Класс описывающий структуру будущего или считанного файла. Предоставляет данные в удобной форме. Позволяет устанавливать основные параметры файла.

**DotWaveWorker** – Класс для простейшей обработки и генерации аудио.

Обратите внимание, что классы **DotWave** и **DotWaveWorker** являются полностью статическими, как и методы в них.

Также, давайте условимся, что в контексте данного руководства я буду иметь ввиду под **каналами** и **аудио**. В первом случае – это всегда одномерный массив типа UInt32[], во втором – это двумерный массив типа UInt32[,]

Полная сводка:

|  |  |
| --- | --- |
| **DotWave** | |
| **Методы** | |
| String : IsWave(byte[] bytes) | Проверяет, является ли заданный набор байт форматом PCM .wav  В положительном случае вернёт NULL, в ином – строку с описанием того, почему это не .wav |
| DotWaveInfo : GetWaves(byte[] bytes) | Вернёт объект типа DotWaveInfo полностью описывающий данный файл, получая массив байт файла (В некоторых случаях сможет прочитать и повреждённый файл, который не прошёл проверку IsWave) |
| Byte[] : SetWaves(DotWaveInfo DWI); | Вернёт упакованный массив байт который может быть записан в файл .wav для последующего воспроизведения |
| **DotWaveInfo** | |
| **Конструкторы** | |
| DotWaveInfo() | Инициализирует новый, пустой экземпляр объекта DotWaveInfo с параметрами по умолчанию |
| DotWaveInfo(DotWaveInfo DWI) | Инициализирует копию объекта DWI |
| DotWaveInfo(UInt32[,] Data, UInt32 SampleRate, UInt16 BitsPerSample ) | Инициализирует новый экземпляр объекта с заданным массивом данных, заданной частотой дискретизации и заданной глубиной звучания |
| **Методы** | |
| UInt32 : GetDuration() | Вернёт продолжительность звучания файла в миллисекундах |
| **Свойства** | |
| UInt32 : SampleRate | Частота дискретизации (по умолчанию 44100) |
| UInt16 : BitsPerSample | Количество бит на звуковой кадр, может принимать значения 8, 16, 24 или 32. (по умолчанию 16) |
| UInt32[,] : Data | Звуковые данные |
| **Свойства (только чтение)** | |
| UInt32 : ChunkID | Равен тексту «RIFF» или 0x52494646 |
| UInt32 : ChunkSize | Равен общей длине файла -8 байт. |
| UInt32 : Format | Равен тексту «WAVE» или 0x57415645 |
| UInt32 : Subchunk1ID | Равен тексту «fmt » или 0x666d7420 |
| UInt32 : Subchunk1Size | Для формата wav(PCM) всегда 16 |
| UInt16 : AudioFormat | Для формата wav(PCM) всегда 1 |
| UInt16 : NumChannels | Количество ауди-каналов (1-моно, 2-стерео, 3-расширенное стерео и тд.) |
| Uint32 : ByteRate | Количество байт которое читается за одну секунду воспроизведения |
| UInt16 : BlockAlign | Количество байт требующееся для одного сэмпла |
| UInt32 : Subchunk2ID | Равен тексту «data» или 0x64617461 |
| UInt32 : Subchunk2Size | Равен количеству байт в области данных |
| **DotWaveWorker** | |
| **Методы** | |
| **Работа с каналами в аудио** | |
| UInt32[,] : MergeToAudio(Object[] channels) | Собирает из предоставленного массива каналов один массив типа аудио |
| Object[] : SplitAudio(UInt32[,] audio) | Разбирает аудио на составляющие каналы |
| UInt32[] : GetChannel(UInt32[,] audio, Int32 num) | Возвращает канал из аудио по его номеру |
| UInt32[,] : AddChannel(UInt32[,] audio, UInt32[] channel) | Добавляет к аудио новый аудио-канал |
| UInt32[,] : RemoveChannel(UInt32[,] audio, Int32 num) | Удаляет из аудио указанный аудио-канал (остальные каналы смещаются на его место) |
| UInt32[,] : ReplaceChannel(UInt32[,] audio, Int32 num, UInt32[] channel) | Заменяет указанный аудио-канал новым |
| UInt32[,] : InsertChannel(UInt32[,] audio, Int32 num, UInt32[] channel) | Вставляет новый аудио-канал на указанную позицию смещаю остальные |
| **Работа с каналами** | |
| UInt32[] : GetSub(UInt32[] channel, UInt32 start, UInt32 length) | Вернёт копию части канала начиная с заданной позиции и заданной длинны |
| UInt32[] : JoinChannels(Object[] channels) | Объединит несколько каналов переданных в массиве объектов |
| UInt32[] : DeleteSub(UInt32[] channel, UInt32 start, UInt32 length) | Удалит часть канала начиная с заданной позиции и заданной длинны |
| UInt32[] : InsertSub(UInt32[] channel, UInt32[] subchannel, UInt32 pos) | Вставит один канал в позицию на другом канале, при этом сдвинув остальную часть |
| UInt32[] : RewriteSub(UInt32[] channel, UInt32[] subchannel, UInt32 start) | Перезапишет часть в канале другим каналом |
| **Работа с аудио** | |
| UInt32[,] : GetSub(UInt32[,] audio, UInt32 start, UInt32 length) | Вернёт копию части аудио начиная с заданной позиции и заданной длинны |
| UInt32[,] : JoinAudios(Object[] channels) | Объединит несколько аудио переданных в массиве объектов |
| UInt32[,] : DeleteSub(UInt32[,] audio, UInt32 start, UInt32 length) | Удалит часть аудио начиная с заданной позиции и заданной длинны |
| UInt32[,] : InsertSub(UInt32[,] audio, UInt32[,] subaudio, UInt32 pos) | Вставит один аудио в позицию на другом аудио, при этом сдвинув остальную часть |
| UInt32[,] : RewriteSub(UInt32[,] audio, UInt32[,] subaudio, UInt32 start) | Перезапишет часть в аудио другим каналом |
| **Обработка каналов / аудио (данные функции могут принимать и возвращать как каналы, так и аудио)** | |
| UInt32[\*] : SetVolume(UInt32[\*] au/ch, float vol) | Изменяет громкость аудио или канала на указанный коэффициент |
| UInt32[\*] : SetLength(UInt32[\*] au/ch, UInt32 length) | Изменяет длину канала или аудио, это влияет на скорость воспроизведения |
| UInt32[\*] : SetSpeed(UInt32[\*] au/ch, float speed) | Устанавливает коэффициент скорости аудио или канал, влияет на длину канала |
| UInt32[\*] : Merge(UInt32[\*] au/ch, UInt32[\*] au/ch) | Объединяет звук двух каналов или аудио в один канал или аудио |
| UInt32[\*] : LinearSoundUp(UInt32[\*] au/ch) | Линейно повышает громкость звука на всём протяжении аудио или канала |
| UInt32[\*] : LinearSoundDown(UInt32[\*] au/ch) | Линейно понижает громкость звука на всём протяжении аудио или канала |
| UInt32[\*] : ParabolicSoundUp(UInt32[\*] au/ch) | Параболически повышает громкость звука на всём протяжении аудио или канала |
| UInt32[\*] : ParabolicSoundDown(UInt32[\*] au/ch) | Параболически понижает громкость звука на всём протяжении аудио или канала |
| UInt32[\*] : Reverse(UInt32[\*] au/ch) | Реверсирует массив данных |
| **Генерация каналов** | |
| UInt32[] : GenerateBeep(float freq, UInt32 sampleRate, UInt32 length) | Генерирует сигнал заданной частоты и длинны (пределы в которых я что-то слышал: 50 – 15000), делает корректировку по частоте воспроизведения |
| UInt32[] : GenerateNoise(UInt32 length) | Генерирует цифровой шум заданной длинны |
| UInt32[] : GenerateInterference(float chance, UInt32 length) | Генерирует треск с заданным шансом (0-1) и длинной |

# Примеры

Ну тут мне стало резко лень… не сегодня.