



# Go gamedev: XM music



quasilyte 2024



# My Go gamedev story

---

- I create games with Ebitengine
- I make libraries for gamedev in Go
- I write talks and articles about gamedev in Go
- [t.me/go\\_gamedev](https://t.me/go_gamedev) (Russian-speaking) creator

I'm using [Ebitengine](#) for around 2-3 years now

# Robo den

Press ENTER to continue





The background of the entire image is a screenshot from the game RoboDen. It depicts a top-down view of a snowy, mountainous landscape under a starry night sky. In the center, the title 'RoboDen' is displayed in a pixelated, green font with a blue outline. Below the title, the text 'Press ENTER to continue' is visible in a smaller, green font. A large black rectangular box with white text is superimposed over the middle of the image. The game scene includes various elements like snow-covered ground, green pine trees, small buildings, and several small, colorful robotic units scattered across the terrain. Some units are connected by blue lines, suggesting a network or communication system. The overall aesthetic is that of a classic 2D strategy game.

# RoboDen

Press ENTER to continue

**Desktop: Linux, Windows, MacOS**



The background is a pixelated space scene with a dark starry sky, white rocky planets, and a large green forested island. The title 'RoboDen' is centered at the top in a green, pixelated font. Below it, the text 'Press ENTER to continue' is displayed in a smaller green font. Two black rectangular boxes with white text are overlaid on the lower half of the screen. The game scene includes various elements like a blue beam of light, a small robot, and a large cluster of green trees on the island.

# RoboDen

Press ENTER to continue

**Desktop: Linux, Windows, MacOS**

**Mobile: Android**



The background of the image is a pixelated space-themed game scene. It features a dark starry sky with various floating islands, some covered in snow and others with green trees. Small alien-like creatures and spacecraft are scattered throughout the scene. The title 'Robodan' is displayed in a pixelated font at the top center.

# Robodan

Press ENTER to continue

**Desktop: Linux, Windows, MacOS**

**Mobile: Android**

**Also works in your browser ([itch.io](https://itch.io))**



The background is a pixelated space scene with a dark starry sky, white nebulae, and various floating islands. One island features green trees and small houses. Numerous small, colorful spacecraft are scattered throughout the scene. The title 'RoboDen' is centered in a pixelated font, with the 'o' replaced by a robot head icon.

RoboDen

Press ENTER to continue

**Desktop: Linux, Windows, MacOS**

**Mobile: Android**

**Also works in your browser ([itch.io](https://itch.io))**

**Has Steam integration (achievements, etc.)**

# Ebitengine audio for music

---

- Supports mp3 and ogg out-of-the box



# Ebitengine audio for music

---

- Supports mp3 and ogg out-of-the box
- Your own stream reader implementation is possible

# Ebitengine audio for music

---

- Supports mp3 and ogg out-of-the box
- Your own stream reader implementation is possible
- Works with 16-bit 2-channel PCM LE streams



# Ebitengine audio for music

---

- Supports mp3 and ogg out-of-the box
- Your own stream reader implementation is possible
- Works with 16-bit 2-channel PCM LE streams
- Works well on every platform I tested my games on

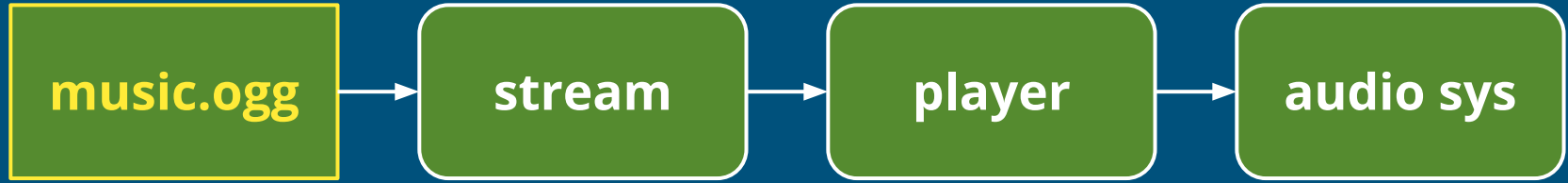
## Stereo 16-bit PCM Little Endian

---

- PCM are given to the audio driver as a final step
- OGG and MP3 formats allow compact storage
- A ~4 min PCM data can have a size of ~50MB

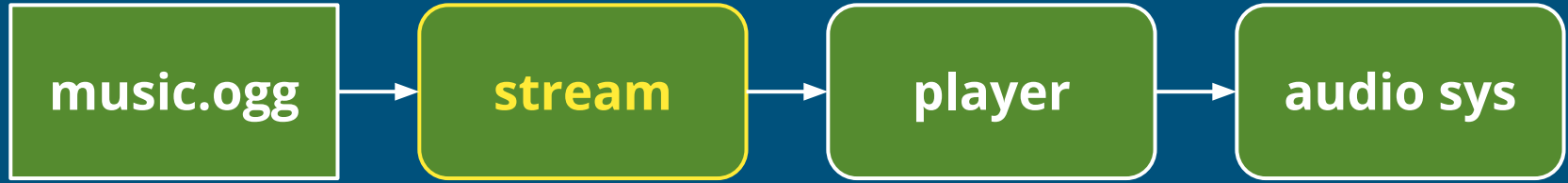
This is why most players “decode” OGG/MP3 into PCM on-the-fly, so you can avoid this large memory overhead.





music.ogg

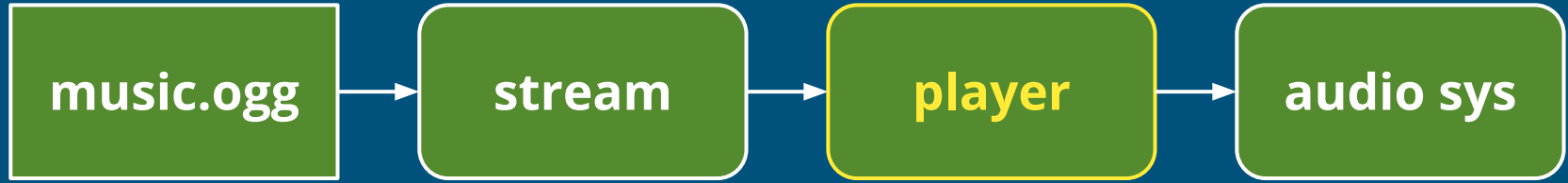
Contains the Vorbis-encoded music data.



## Stream

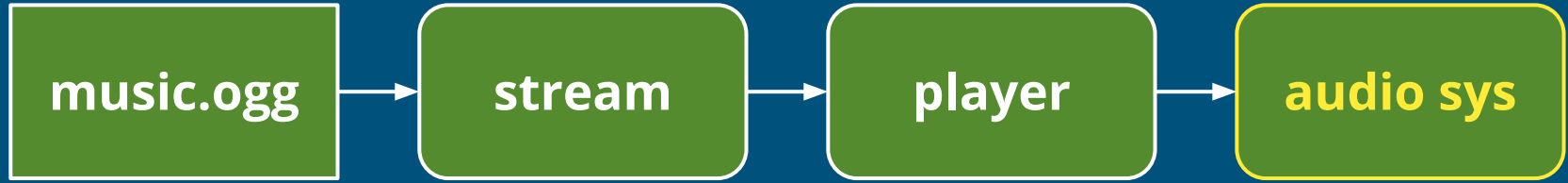
Reads OGG data and turns them into the 16-bit PCM LE bytes the player expects to get.





## (audio) Player

This is your audio system API object. It's a bridge between your stream implementation and the underlying audio system. Players are reusable, they wrap a single stream at a time. You can create tons of Player objects in your game.



## Audio system

This part is usually unseen for a game developer. We can assume that it's some kind of a low-level library that speaks to the audio systems on different platforms.

Why XM?



# Roboden music story

---

I used [Drozerix](#) tracks from [modarchive](#) for my Roboden game.

# Roboden music story

---

I used [Drozerix](#) tracks from [modarchive](#) for my Roboden game.

They were in XM format, so I converted them to OGG.

# Roboden music story

---

I used [Drozerix](#) tracks from [modarchive](#) for my Roboden game.

They were in XM format, so I converted them to OGG.

At some point, the game archive became quite big for a web build.

# Problems with OGG (and MP3)

---

- Large size (a problem for mobiles and web)



# Problems with OGG (and MP3)

---

- Large size (a problem for mobiles and web)
- Lack of the “sources” (they’re also “lossy”)

# Problems with OGG (and MP3)

---

- Large size (a problem for mobiles and web)
- Lack of the “sources” (they’re also “lossy”)
- Harder to do dynamic fancy stuff with the sound

# Let's go one step back

---

The “source” of my music (Drozerix tracks) is XM.

# Let's go one step back

---

The “source” of my music (Drozerix tracks) is XM.

XM file size: 71 KB

Converted OGG file size: 1.8 MB (~1843 KB)

It's about x25 times smaller!



# Roboden web archive size

---

With OGG music: ~18 MB

With XM music: 9 MB



# The modular music

---

- Smaller file size
- The music file itself is a source
- Almost the “code is data” approach

# The modular music

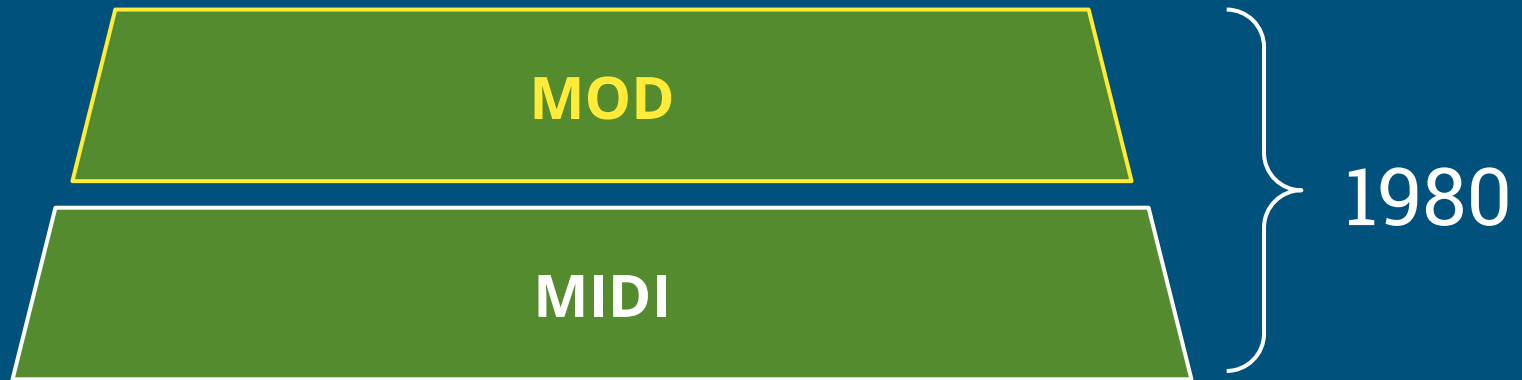
---



**MIDI**

# The modular music

---





# The modular music

---



# Some games that used modular music

---

- Deus Ex (2000, IT format)
- Unreal Tournament (1998, IT format)
- Age of Wonders (1996, IT format)
- Star Control 2 (1992, MOD format)
- Several first Final Fantasy games (MOD format)

...most modular formats can be converted to XM

# XM music format

---

Stands for “Extended MOD”.

It’s like MOD, but better (it’s even more compact thanks to the simple compression scheme).

## OGG

Stores the compressed music track data

## XM

Stores the instructions about how to play the music and samples data

## OGG

Stores the compressed music track data

Can't be edited by a human

## XM

Stores the instructions about how to play the music and samples data

Can be easily edited using a Tracker software

## OGG

Stores the compressed music track data

Can't be edited by a human

Can't be transformed on-the-fly during the playback

## XM

Stores the instructions about how to play the music and samples data

Can be easily edited using a Tracker software

Can be manipulated by a program in many ways



## OGG

Stores the compressed music track data

Can't be edited by a human

Can't be transformed on-the-fly during the playback

Avg. size is 3-7 MB

## XM

Stores the instructions about how to play the music and samples data

Can be easily edited using a Tracker software

Can be manipulated by a program in many ways

Avg. size is 50-500 KB

# Comparing XM, IT, S3M

---

- All of them are modular music formats

# Comparing XM, IT, S3M

---

- All of them are modular music formats
- XM and IT are less limiting than S3M

# Comparing XM, IT, S3M

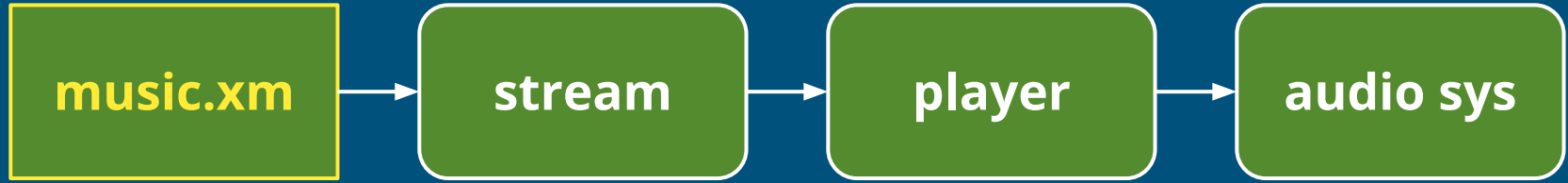
---

- All of them are modular music formats
- XM and IT are less limiting than S3M
- XM is more popular than the other two nowadays

# Comparing XM, IT, S3M

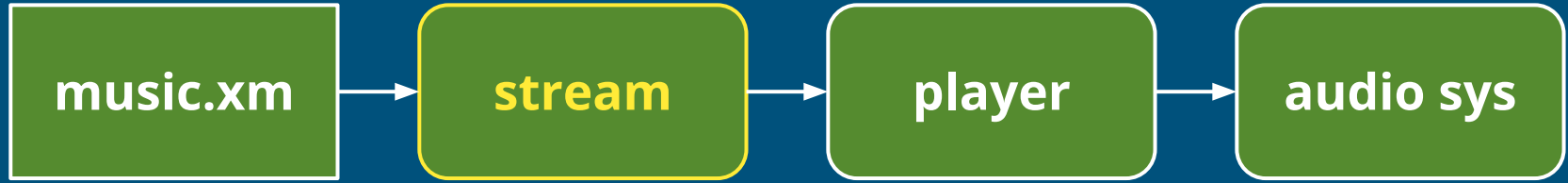
---

- All of them are modular music formats
- XM and IT are less limiting than S3M
- XM is more popular than the other two nowadays
- MilkyTracker can convert IT and S3M to XM



music.xml

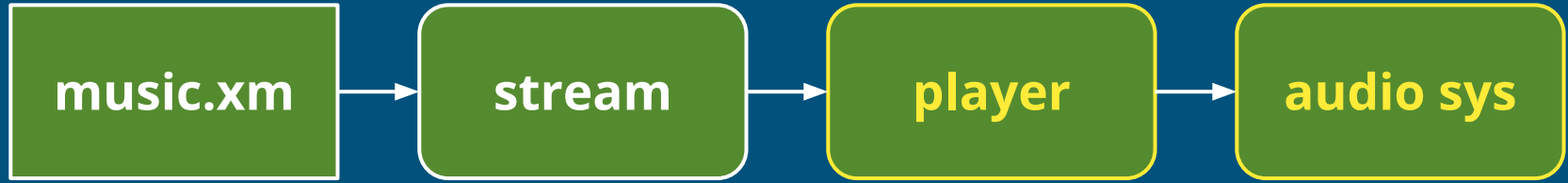
Contains the instructions for an XM-player. Also stores the necessary samples data inside the XM file.



## Stream

Plays a role of an XM-player. It evaluates the XM instructions and produces the output PCM bytes.





[same as with OGG]

# XM file layout

---



Header with metadata, etc.

# XM file layout

---



Samples

# XM file layout

---



Patterns (rows, notes)

# XM file layout



0, 1, 0, 0, 2, 3, 4, 4, 4, 5, 6, 7, 1, 8, 9, 2, 0, 1, 1



Pattern order (just an array of indexes)

# XM player for Go (Ebitengine-compatible)

---

[github.com/quasilyte/xm](https://github.com/quasilyte/xm)

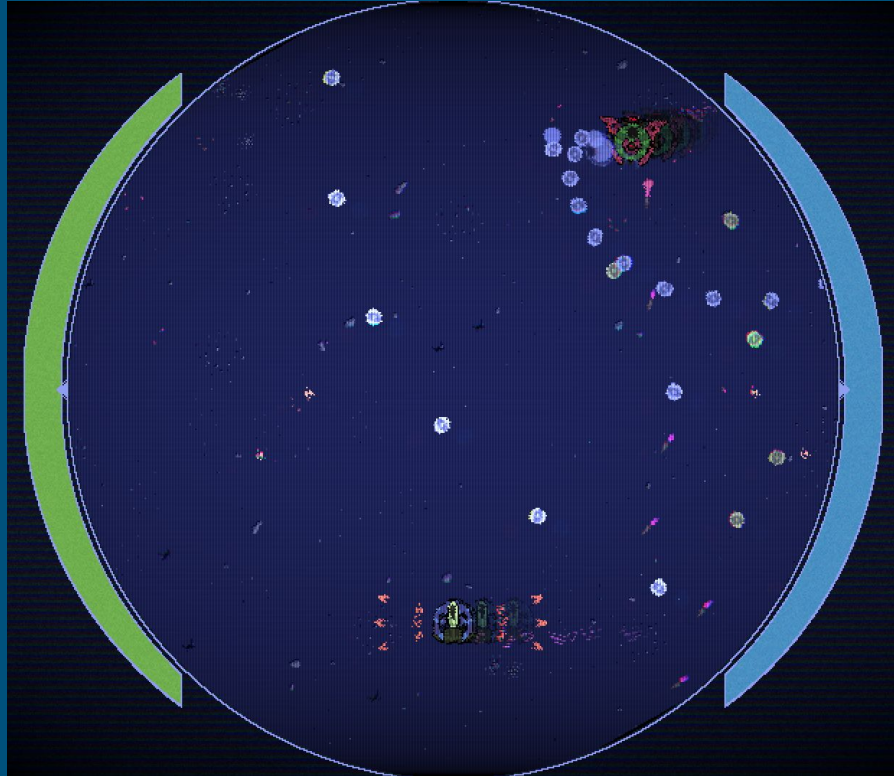
Used in [Roboden](#) and some other games of mine

XM-powered games



# TuneFire game (GameOff 2023)

---



	1	2	3	4
00	A-b 7		A-4 1	A-5 1
01				
02				C-b 1
03				
04		E-b 7	A-3 1	D-b 1
05				
06				E-b 1
07				
08	C-b 7		A-4 1	A-b 1
09				
0A				C-7 1
0B				
0C		E-b 7	A-3 1	D-7 1
0D				
0E				E-7 1
0F				
10	A-b 7		A-4 1	D-7 1
11				
12				C-7 1
13				
14		E-b 7	A-3 1	A-b 1
15				
16				E-b 1
17				
18	C-b 7		A-4 1	C-b 1
19				
1A				A-5 1
1B				
1C		E-b 7	A-4 1	E-5 1
1D				

} Pattern

	1	2	3	4
00	A-b 7		A-4 1	A-5 1
01				C-b 1
02				
03				
04		E-b 7	A-3 1	D-b 1
05				E-b 1
06				
07				
08	C-b 7		A-4 1	A-b 1
09				C-7 1
0A				
0B				
0C		E-b 7	A-3 1	D-7 1
0D				E-7 1
0E				
0F				
10	A-b 7		A-4 1	D-7 1
11				
12				C-7 1
13				
14		E-b 7	A-3 1	A-b 1
15				E-b 1
16				
17				
18	C-b 7		A-4 1	C-b 1
19				A-5 1
1A				
1B				
1C		E-b 7	A-4 1	E-5 1
1D				



Pattern's row

	1	2	3	4
00	A-b 7		A-4 1	A-5 1
01				
02				C-b 1
03				
04		E-b 7	A-3 1	D-b 1
05				
06				E-b 1
07				
08	C-b 7		A-4 1	A-b 1
09				
0A				C-7 1
0B				
0C		E-b 7	A-3 1	D-7 1
0D				
0E				E-7 1
0F				
10	A-b 7		A-4 1	D-7 1
11				
12				C-7 1
13				
14		E-b 7	A-3 1	A-b 1
15				
16				E-b 1
17				
18	C-b 7		A-4 1	C-b 1
19				
1A				A-5 1
1B				
1C		E-b 7	A-4 1	E-5 1
1D				

- Channel number

	1	2	3	4
00	A-b [7]		A-4 [1]	A-5 [1]
01				C-b [1]
02				D-b [1]
03		E-b [7]	A-3 [1]	E-b [1]
04				A-b [1]
05			A-4 [1]	C-7 [1]
06				D-7 [1]
07	C-b [7]		A-3 [1]	E-7 [1]
08		E-b [7]	A-4 [1]	D-7 [1]
09				C-7 [1]
0A			A-3 [1]	A-b [1]
0B				E-b [1]
0C		E-b [7]	A-4 [1]	C-b [1]
0D			A-3 [1]	A-5 [1]
0E				E-5 [1]
0F	A-b [7]		A-4 [1]	
10				
11			A-3 [1]	
12		E-b [7]	A-4 [1]	
13				
14			A-3 [1]	
15		E-b [7]	A-4 [1]	
16				
17	C-b [7]		A-3 [1]	
18			A-4 [1]	
19				
1A			A-3 [1]	
1B		E-b [7]	A-4 [1]	
1C				
1D				

- Channel number
- Instrument ID



	1	2	3	4
00	A-b 7		A-4 1	A-5 1
01				C-b 1
02				D-b 1
03		E-b 7	A-3 1	E-b 1
04				A-b 1
05				C-7 1
06				D-7 1
07	C-b 7		A-4 1	E-7 1
08				D-7 1
09				C-7 1
0A		E-b 7	A-3 1	A-b 1
0B				E-b 1
0C				C-b 1
0D				A-5 1
0E				E-5 1
0F	A-b 7		A-4 1	
10				
11				
12				
13		E-b 7	A-3 1	
14				
15				
16				
17	C-b 7		A-4 1	
18				
19				
1A				
1B				
1C		E-b 7	A-4 1	
1D				

- Channel number
- Instrument ID
- Notes (pitch)

# Using music data as gameplay elements

	1	2	3	4
00	A-b 7		A-4 1	A-5 1
01				C-b 1
02		E-b 7	A-3 1	D-b 1
03				E-b 1
04	C-b 7		A-4 1	A-b 1
05		E-b 7	A-3 1	C-7 1
06				D-7 1
07				E-7 1
08	A-b 7		A-4 1	D-7 1
09		E-b 7	A-3 1	C-7 1
10				A-b 1
11		E-b 7	A-3 1	E-b 1
12	C-b 7		A-4 1	C-b 1
13				A-5 1
14		E-b 7	A-4 1	E-5 1

- Channel number
- Weapon type

# Using music data as gameplay elements

	1	2	3	4
00	A-b 7		A-4 1	A-5 1
01				C-b 1
02		E-b 7	A-3 1	D-b 1
03			A-4 1	E-b 1
04	C-b 7		A-4 1	A-b 1
05		E-b 7	A-3 1	C-7 1
06			A-3 1	D-7 1
07		E-b 7	A-4 1	E-7 1
08	A-b 7		A-4 1	D-7 1
09		E-b 7	A-3 1	C-7 1
10			A-3 1	A-b 1
11		E-b 7	A-4 1	E-b 1
12	C-b 7		A-4 1	C-b 1
13			A-3 1	A-5 1
14		E-b 7	A-4 1	E-5 1
15				

- Channel number
- Instrument ID
- Weapon type
- Weapon owner



# Using music data as gameplay elements

	1	2	3	4
00	A-b 7		A-4 1	A-5 1
01				C-b 1
02		E-b 7	A-3 1	D-b 1
03			A-4 1	E-b 1
04	C-b 7		A-4 1	A-b 1
05		E-b 7	A-3 1	C-7 1
06			A-3 1	D-7 1
07		E-b 7	A-4 1	E-7 1
08	A-b 7		A-4 1	D-7 1
09		E-b 7	A-3 1	C-7 1
10			A-4 1	A-b 1
11		E-b 7	A-3 1	E-b 1
12	C-b 7		A-4 1	C-b 1
13			A-4 1	A-5 1
14		E-b 7	A-4 1	E-5 1

- Channel number
- Instrument ID
- Notes (pitch)
- Weapon type
- Weapon owner
- Projectile power

# TuneFire game (GameOff 2023)

---

## Results



Criteria	Rank	Score*	Raw Score
<a href="#">Audio</a>	#29	3.800	3.800
<a href="#">Innovation</a>	#126	3.267	3.267
<a href="#">Overall</a>	#146	3.333	3.333
<a href="#">Graphics</a>	#167	3.333	3.333
<a href="#">Gameplay</a>	#203	2.933	2.933
<a href="#">Theme interpretation</a>	#267	2.733	2.733

# Drum Hero (WIP)



# Step 1: remove drums from the track

```
1  {
2      "Name": "Arilou Theme",
3      "Author": "Dan Nicholson",
4      "Instruments": {
5          "ClosedHiHat": 1,
6          "Snare": 2,
7          "Bass": 4
8      }
9  }
```

		1		2		3		4
00	F#5	1	●	F-4	3	C-4	4	●
01								
02	F#5	1	●	G-4	3			
03								
04	C-5	2	●	G-5	3			
05								
06	F#5	1	●	G-4	3			
07								
08	F#5	1	●	F-4	3	C-4	4	●
09								
0A	F#5	1	●	G-4	3	C-4	4	●
0B								
0C	C-5	2	●	G-5	3			
0D								
0E								
0F								
10	F#5	1	●	F-4	3	C-4	4	●
11								
12	F#5	1	●	F-4	3			
13								
14	C-5	2	●	F-5	3			
15								
16	F#5	1	●	F-4	3			
17								
18	F#5	1	●	F-4	3	C-4	4	●
19								
1A	F#5	1	●	F-4	3	C-4	4	●
1B								
1C	C-5	2	●	F-5	3			
1D								

```
for _, patternIndex := range t.Module.PatternOrder {
    p := &t.Module.Patterns[patternIndex]
    for j := range p.Rows {
        row := &p.Rows[j]
        for _, noteID := range row.Notes {
            n := module.Notes[noteID]
            kind := t.GetInstrumentKind(n.Instrument)
            if kind != edrum.UndefinedInstrument {
                // Skip this instrument. It will be played by the player.
                continue
            }
            // Remove this note from the track.
        }
    }
}
```

## Step 2: extract selected instrument samples

---

Can be done programmatically or manually via Tracker software (like MilkyTracker).

## Step 3: create a note map

---

For every note “removed” from the track, remember its timings and other info like instrument type.

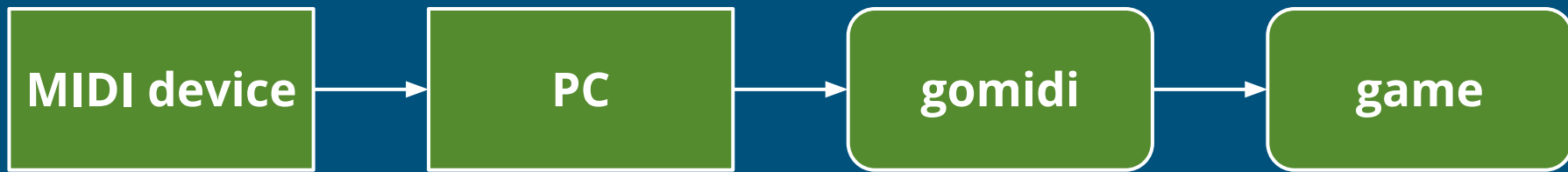
Render these note bars to the players when they need to play them.

## Step 4: read the MIDI stream

---

For every MIDI “play note” event play instrument’s associated sample.

[gitlab.com/gomidi/midi/](https://gitlab.com/gomidi/midi/)





# Summary

---

- The track is played without drums

# Summary

---

- The track is played without drums
- There is an interactive drum notes map

# Summary

---

- The track is played without drums
- There is an interactive drum notes map
- The drum will play original samples

# Summary

---

- The track is played without drums
- There is an interactive drum notes map
- The drum will play original samples
- Every drum stroke is /real/ and affects the song

# What else can we do?

---

- Collect player stats, like rhythm consistency
- Create tab sheets for an XM track automatically
- Play XM tracks at different speed and effects
- This is not limited to drums-only, any MIDI-device will do
- Record the player and build a combined XM track
- Build colored sound wave based on inst&chan index

# My XM player library for Go

---

- High performance (zero-alloc repeated plays)
- Sample interpolation & volume ramping support
- Dependency-free
- Ebitengine-compatible
- Exports XM files and parsers

[github.com/quasilyte/xm](https://github.com/quasilyte/xm)

# XM Performance

# XM playback

---

There are two main aspects to it:

1. Evaluating the effects/notes for a “tick”
2. **Rendering** the PCM bytes for the given tick

(1) is XM-specific, (2) is what any player would do

**Rendering** the PCM dominates the run time: 90-95%



# Benchmarks

---

We'll be comparing two libraries:

1. XM: [github.com/quasilyte/xm](https://github.com/quasilyte/xm)
2. OGG: [github.com/jfreymuth/vorbis](https://github.com/jfreymuth/vorbis)

# Benchmarks

---

We'll be using 3 different tracks:

1. Industrial Porn (Drozerix)
2. Old Bulls (Aruan); a MOD file converted to XM
3. Crush (Drozerix)

OGG player uses the converted XM->OGG file

# Benchmarks

---

There are 2 main parts of playing the music:

- Loading the file (preparing it to be played)
- Streaming the PCM bytes (playing the music)

# Benchmarks: decoding (ns/op)

---

Benchmark	OGG	XM	XM (lerp)
Decode1	6.27 ms	3.30 ms	3.46 ms
Decode2	4.95 ms	1.56 ms	3.58 ms
Decode3	5.03 ms	4.45 ms	4.98 ms

# Benchmarks: decoding (ns/op)

---

Benchmark	OGG	XM	XM (lerp)
Decode1	slowest	~90% faster	~80% faster
Decode2	slowest	~317% faster	~38% faster
Decode3	slowest	~13% faster	~same

# Benchmarks: playing (ns/op)

---

Benchmark	OGG	XM	XM (lerp)
Play1	4245 ms	1235 ms	Same as previous
Play2	4292 ms	540 ms	Same as previous
Play3	2609 ms	1627 ms	Same as previous

# Benchmarks: playing (ns/op)

---

Benchmark	OGG	XM	XM (lerp)
Play1	slowest	~343% faster	Same as previous
Play2	slowest	~795% faster	Same as previous
Play3	slowest	~160% faster	Same as previous

# Benchmarks: playing (allocs/op)

---

Benchmark	OGG	XM	XM (lerp)
Play1	444097	0	0
Play2	447999	0	0
Play3	163519	0	0



# Benchmarks: conclusion

---

- XM players are not slow
- XM players can be zero alloc

If XM-style music fits your game, use it directly instead of converting it to OGG (or MP3)

# XM lib internals

# Stages separation

---

- Decoding: compile the XM module
- Playback: generate PCM bytes from the module

Compilation happens only once.

A module can be played multiple times.

This library favors the playback efficiency (zero alloc).

# Sample loops

---

A sample can “loop”:

- Forward loop
- Ping-pong loop (bidirectional)

# Sample loops

---

A sample can “loop”:

- Forward loop
- Ping-pong loop (bidirectional)

This means there are 3 “modes”: no loop, forward, pingpong

# Sample loops

---

A sample can “loop”:

- Forward loop
- Ping-pong loop (bidirectional)

This means there are 3 “modes”: no loop, forward, pingpong

We can unify all of them (for branchless performance)

# Ping-pong loop

---



Played as 0, 1, 2, 3, 4, 3, 2, 1, ...

# Unrolled ping-pong loop

---



Loop start



Loop end

Now we only have “forward” loops



# Sample interpolation (lerp, etc.)

---

There are (at least) two ways:

- A genuine interpolation during a playback
- A precomputed subsamples approach

My library uses the latter

# Precomputed subsamples

---

- Injects subsamples during the track compilation

# Precomputed subsamples

---

- Injects subsamples during the track compilation
- Requires more memory due to the extra samples

# Precomputed subsamples

---

- Injects subsamples during the track compilation
- Requires more memory due to the extra samples
- Has zero CPU cost during the playback

# Precomputed subsamples

---

- Injects subsamples during the track compilation
- Requires more memory due to the extra samples
- Has zero CPU cost during the playback
- Can be sample-size dependent (adaptive)

# Original sample

---

0	1	2	3	4
---	---	---	---	---

# With 1 sub-sample injected

---



# Volume ramping

---

Only a few first bytes of the “tick” require ramping.

Process “tick” in two loops: with and without ramping.



```
n := s.module.bytesPerTick
const rampBytes = 2 * 2 * numRampPoints

for i := 0; i < rampBytes; i += 4 {
    // ... generate PCM with ramping
}

// 80-90% of bytes don't need ramping:
for i := rampBytes; i < n; i += 4 {
    // ... generate PCM without ramping (super fast)
}
```

# Closing Words

# Using other modular music formats

---

These formats can be converted to XM easily:

- MOD -> XM (I use MilkyTracker for this conversion)
- S3M -> XM (MilkyTracker and modplug)
- IT -> XM (MilkyTracker)

Amiga frequencies can be converted to linear too.

# Links

---

- [XM file format overview](#)
- [A tiny XM player implementation in C](#)
- [MilkyTracker sources](#) (implements XM as well)
- [Modarchive](#) (modular music collection)
- [My XM library for Go](#)
- [Ebitengine Discord channel](#) (international)

# What I want you to remember from this talk

---

- Game development in Go is a thing (try it out!)

# What I want you to remember from this talk

---

- Game development in Go is a thing (try it out!)
- Modular music (esp. XM) is still relevant

# What I want you to remember from this talk

---

- Game development in Go is a thing (try it out!)
- Modular music (esp. XM) is still relevant
- You can play the XM music in Ebitengine directly

# What I want you to remember from this talk

---

- Game development in Go is a thing (try it out!)
- Modular music (esp. XM) is still relevant
- You can play the XM music in Ebitengine directly
- Modular music can sound cool (Deus Ex OST, Drozerix)



# What I want you to remember from this talk

---

- Game development in Go is a thing (try it out!)
- Modular music (esp. XM) is still relevant
- You can play the XM music in Ebitengine directly
- Modular music can sound cool (Deus Ex OST, Drozerix)
- XM players are not slow (see benchmarks)



# Go gamedev: XM music



quasilyte 2024

