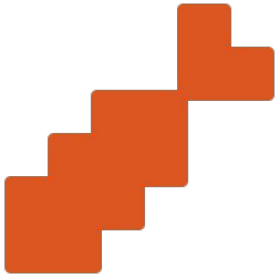
Ebitengine Ecosystem Overview

quasilyte @ podlodka 2023



About me

- Compiler engineer during the day
- Game developer during the night
- I also stream indie games from itch io
- Playing video games since ~2000

I used **Game Maker**, **Godot** and **Ebitengine** a lot.

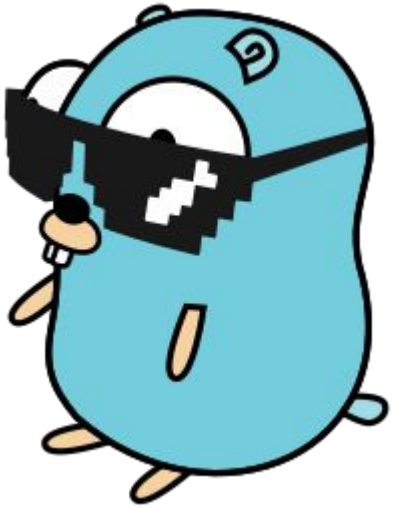
I also tried making games with **Defold**, **Phaser** and **SDL**.

Before we start

Please ⭐ Ebitengine (Eh-Bee-Ten-Gin) repository:

github.com/hajimehoshi/ebiten

It deserves some extra love.



Agenda

- Check out some of the games made with Ebitengine
- List Ebitengine features (as of today)
- Discover 3rd-party Ebitengine libraries
- Make some conclusions

For inspiration

Agenda

- Check out some of the games made with Ebitengine
- List Ebitengine features (as of today)
- Discover 3rd-party Ebitengine libraries
- Make some conclusions

Because this is the foundation of every game

Agenda

- Check out some of the games made with Ebitengine
- List Ebitengine features (as of today)
- Discover 3rd-party Ebitengine libraries
- Make some conclusions

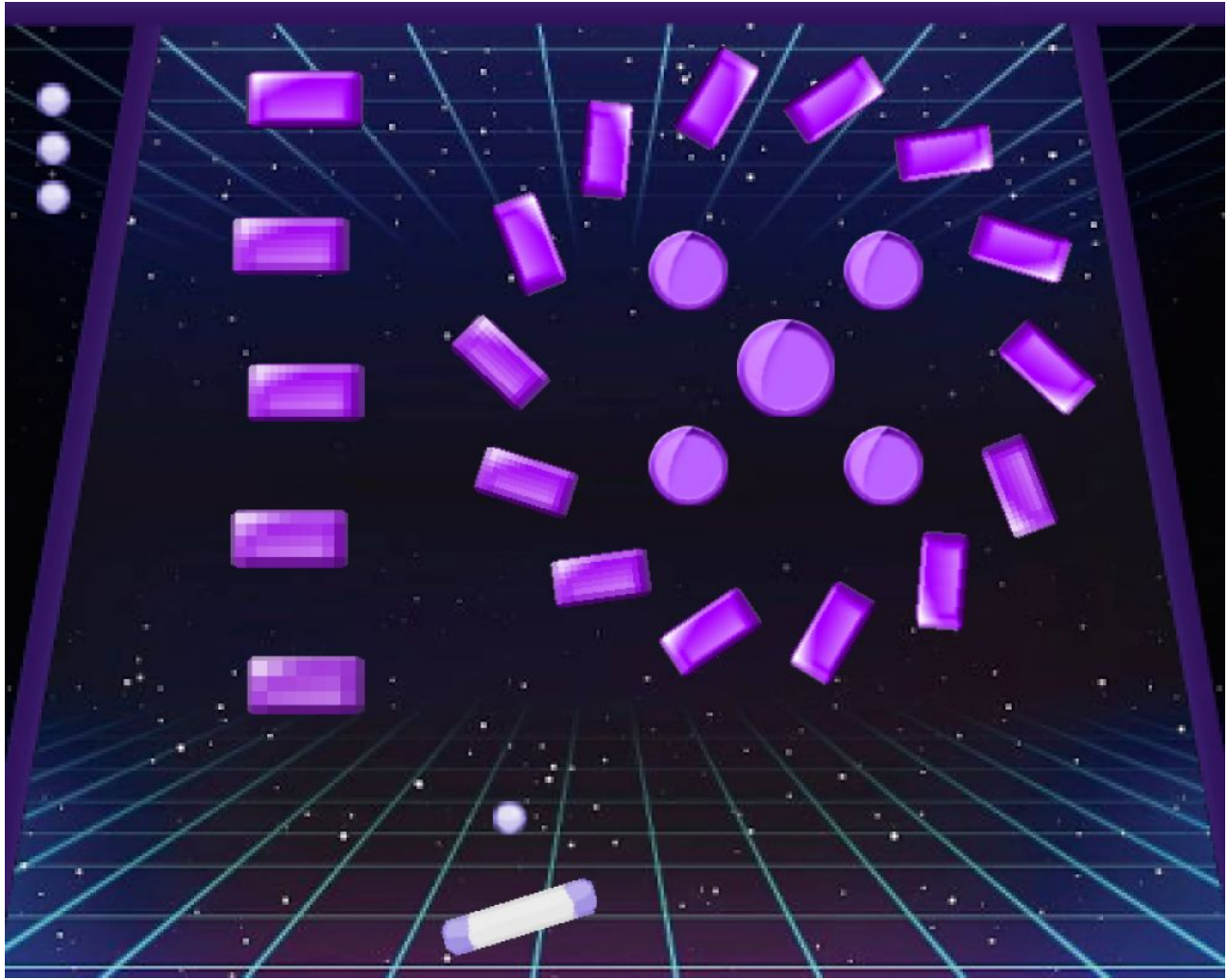
You'll probably need some of those

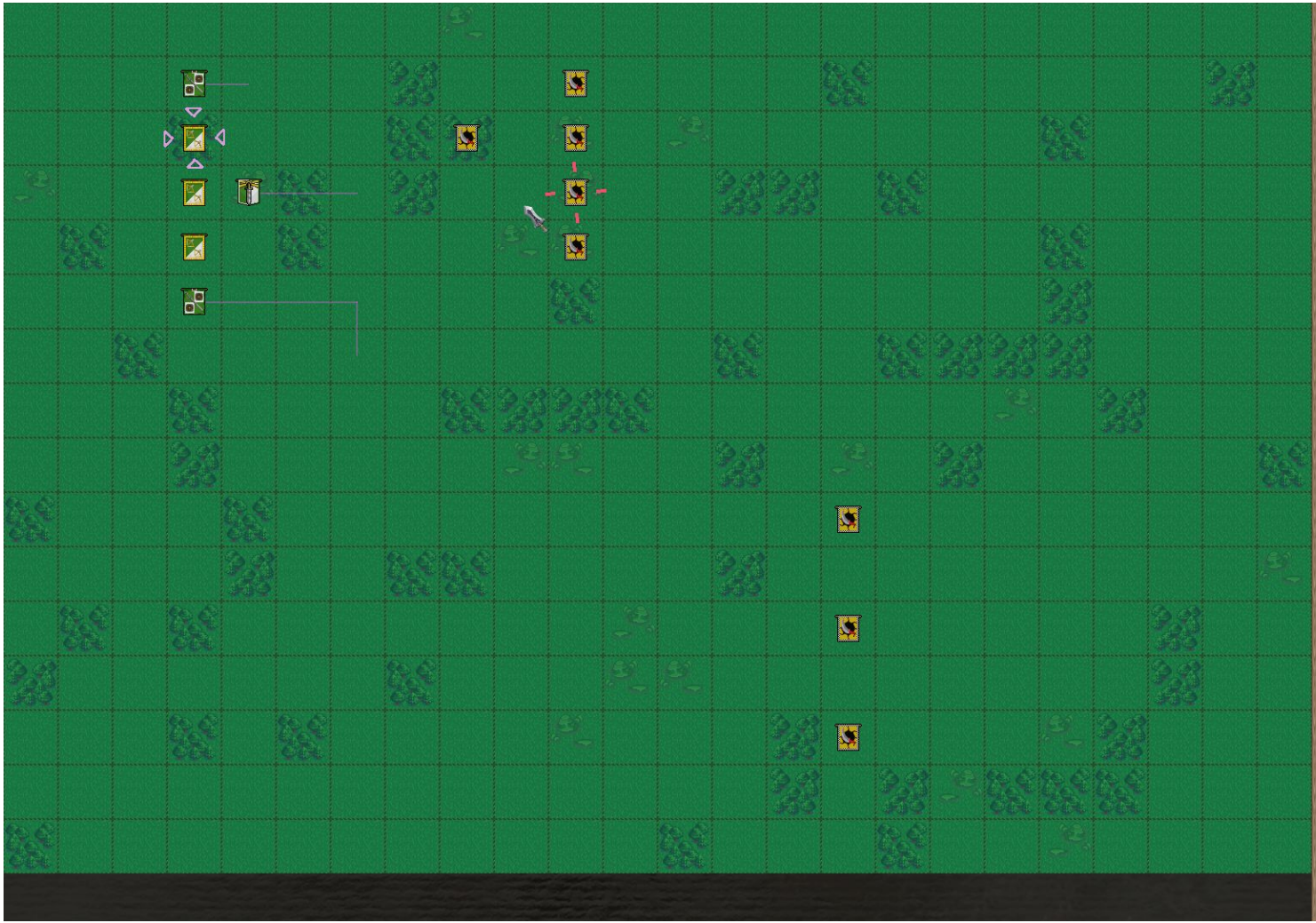
Agenda

- Check out some of the games made with Ebitengine
- List Ebitengine features (as of today)
- Discover 3rd-party Ebitengine libraries
- Make some conclusions

At least we'll try to!

My Ebitengine Games

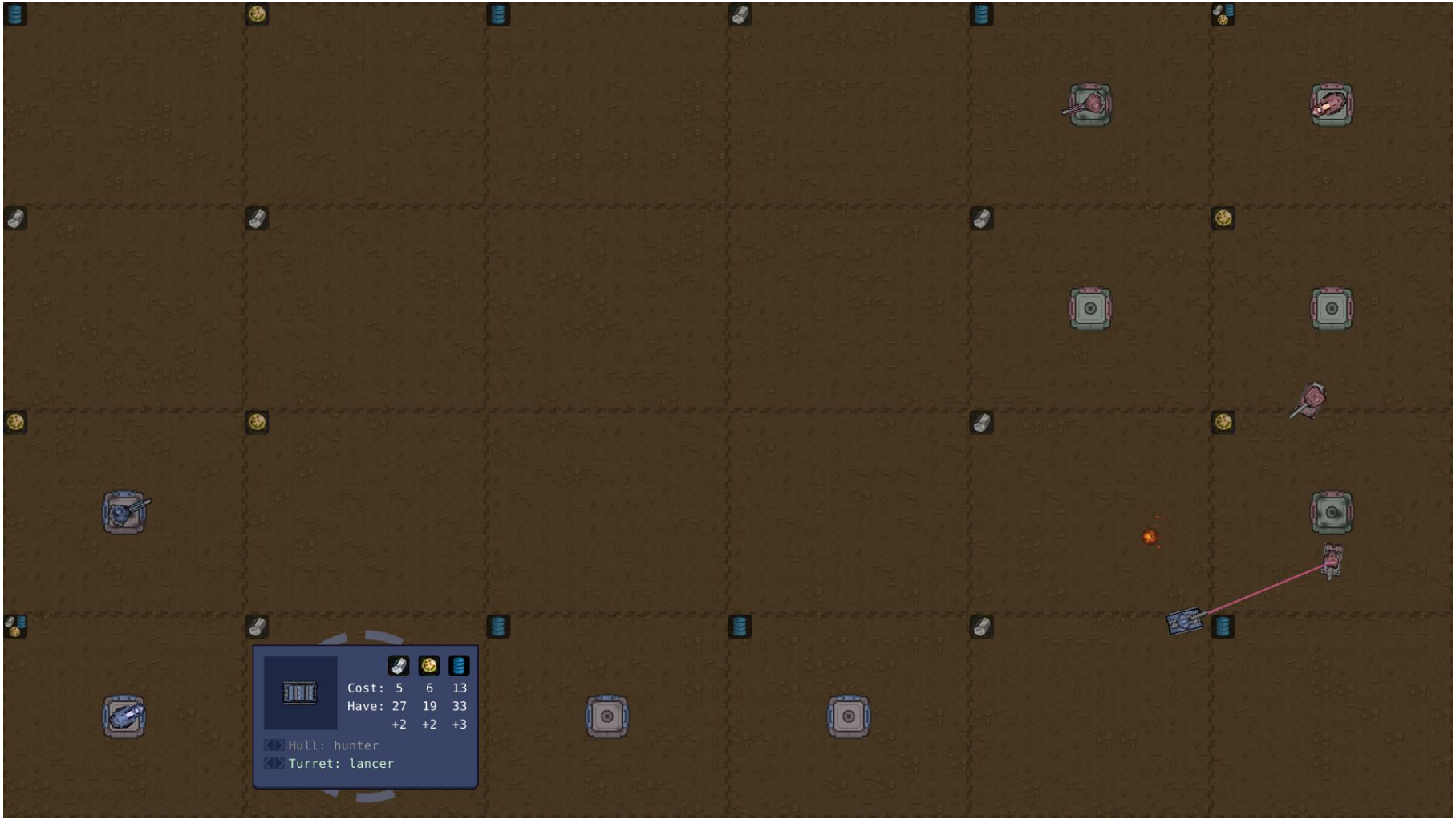


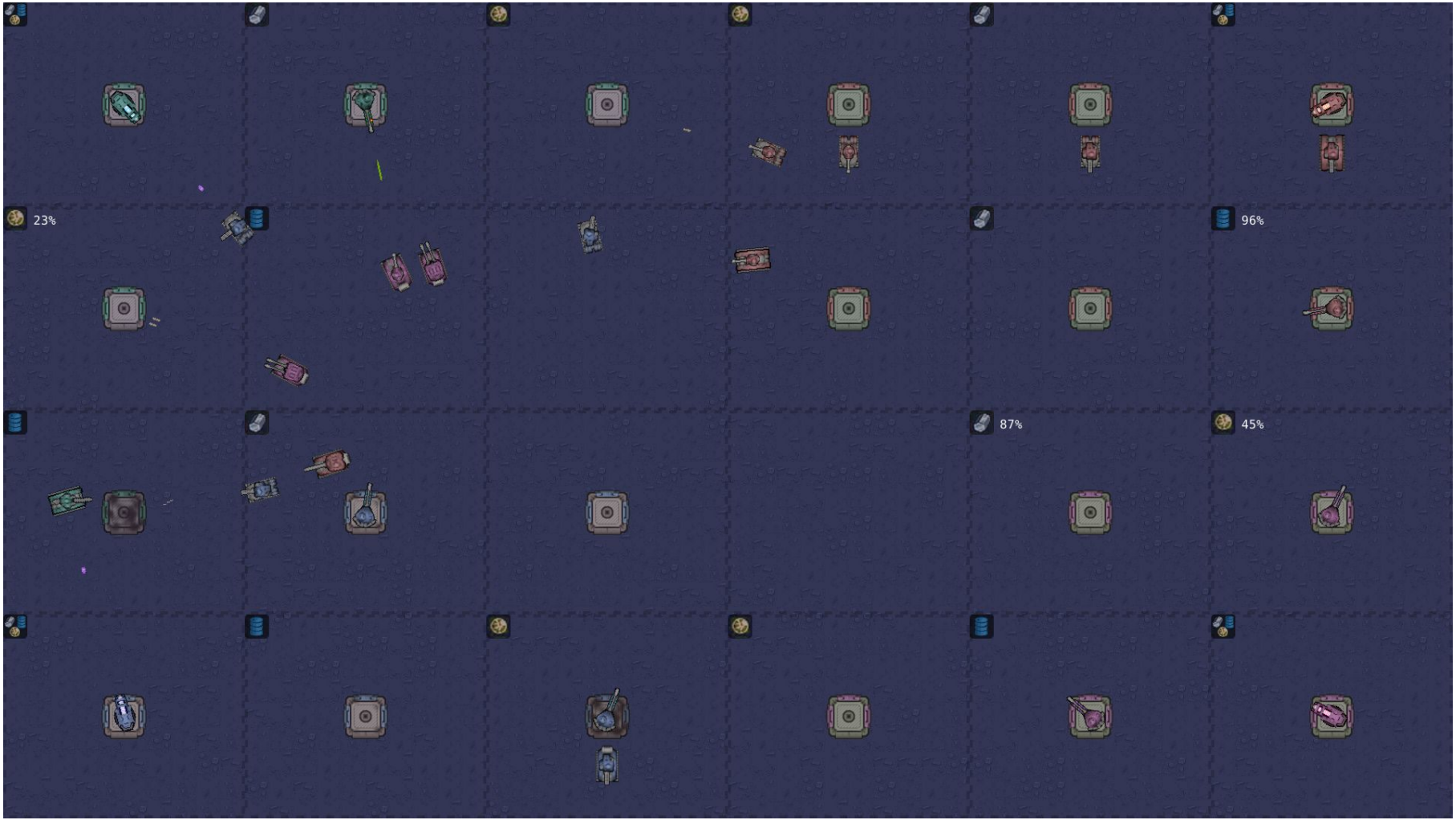

20
14
Mercenary Archers
Status: guard

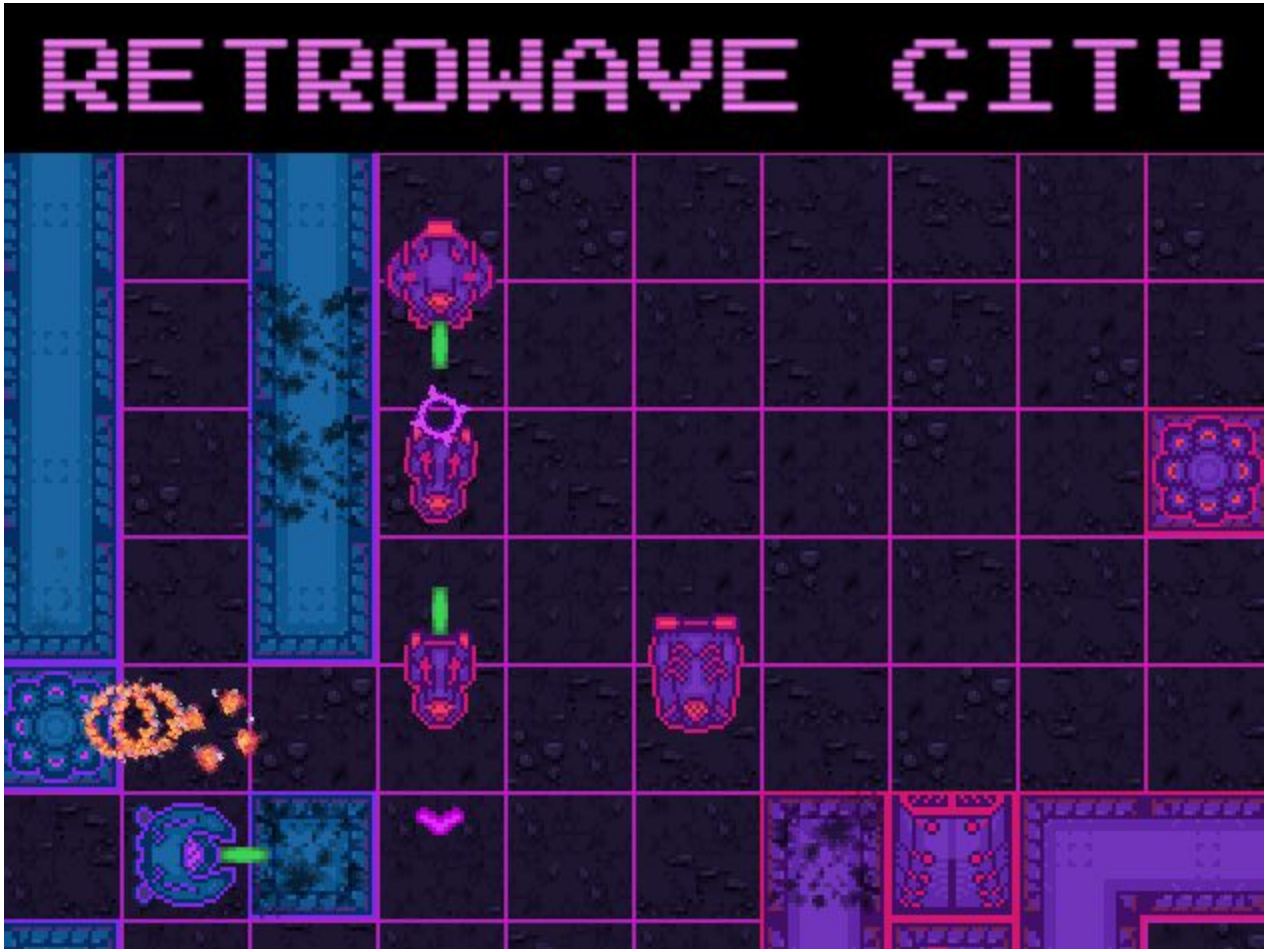
LETTER -> TRANSCRIPTION

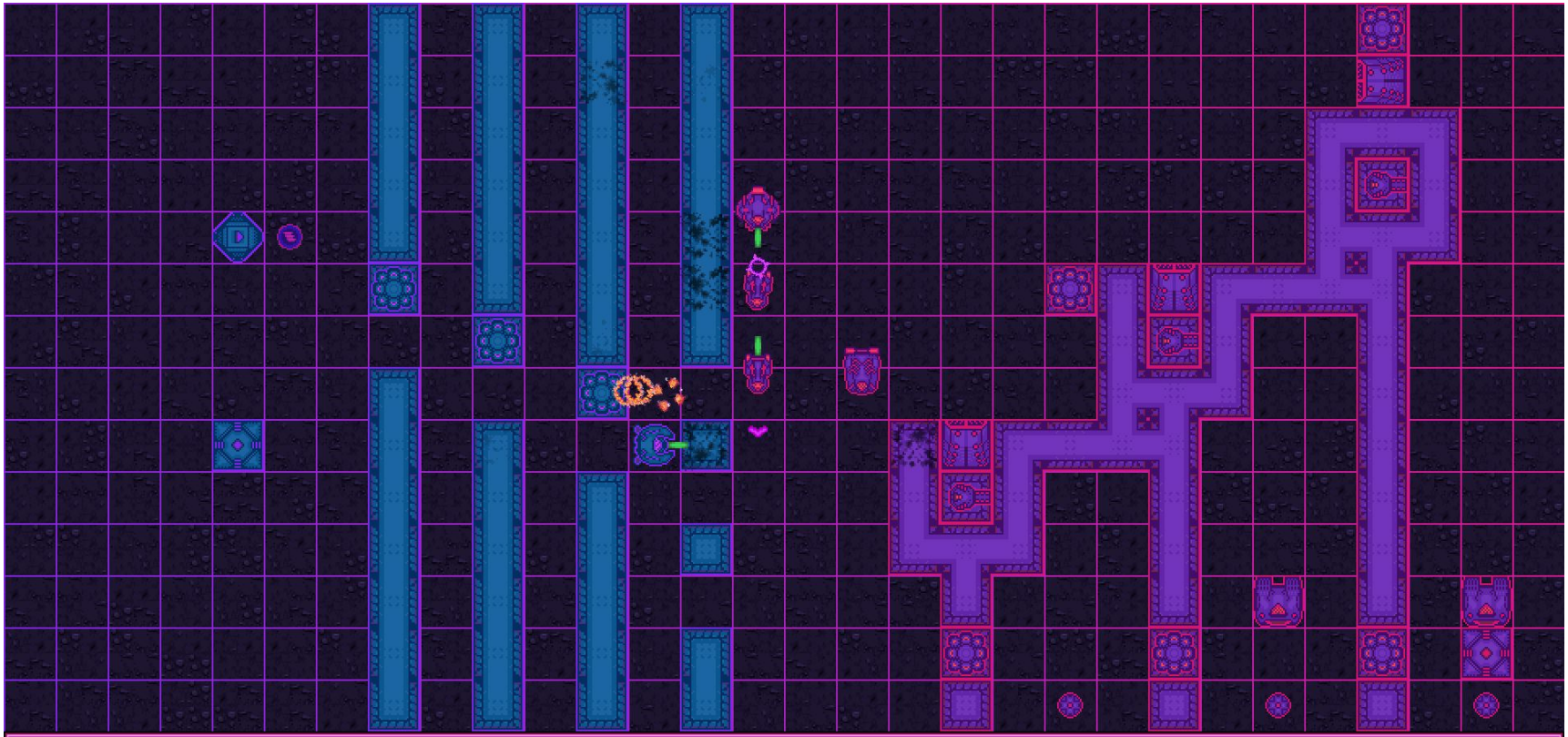
E	D	B	A
/ɛ/	/d/	/b/	/ɑ/


AUTOTANKS


Cost: 5 6 13
Have: 27 19 33
+2 +2 +3
Hull: hunter
Turret: lancer

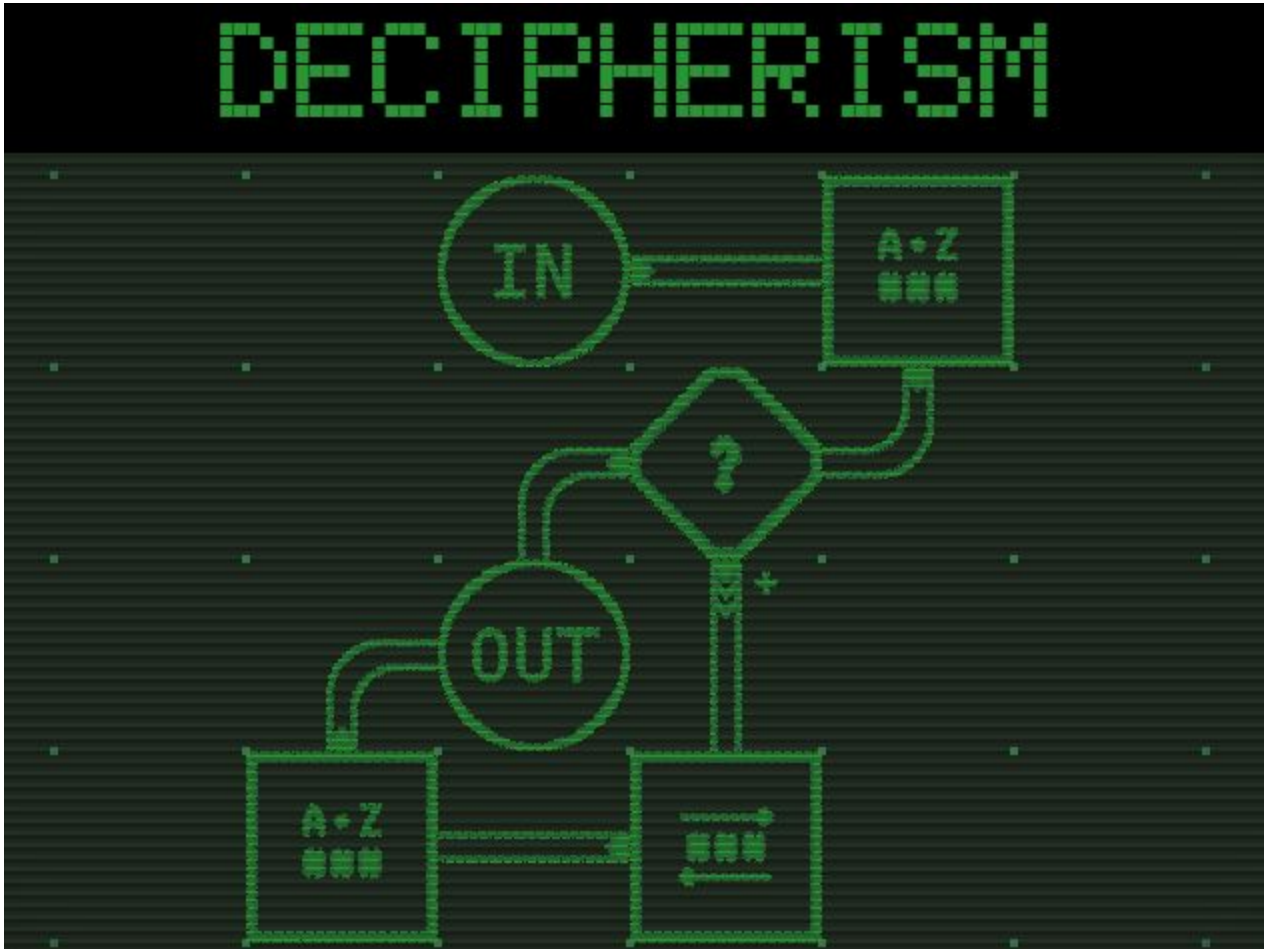

23%
96%
87%
45%

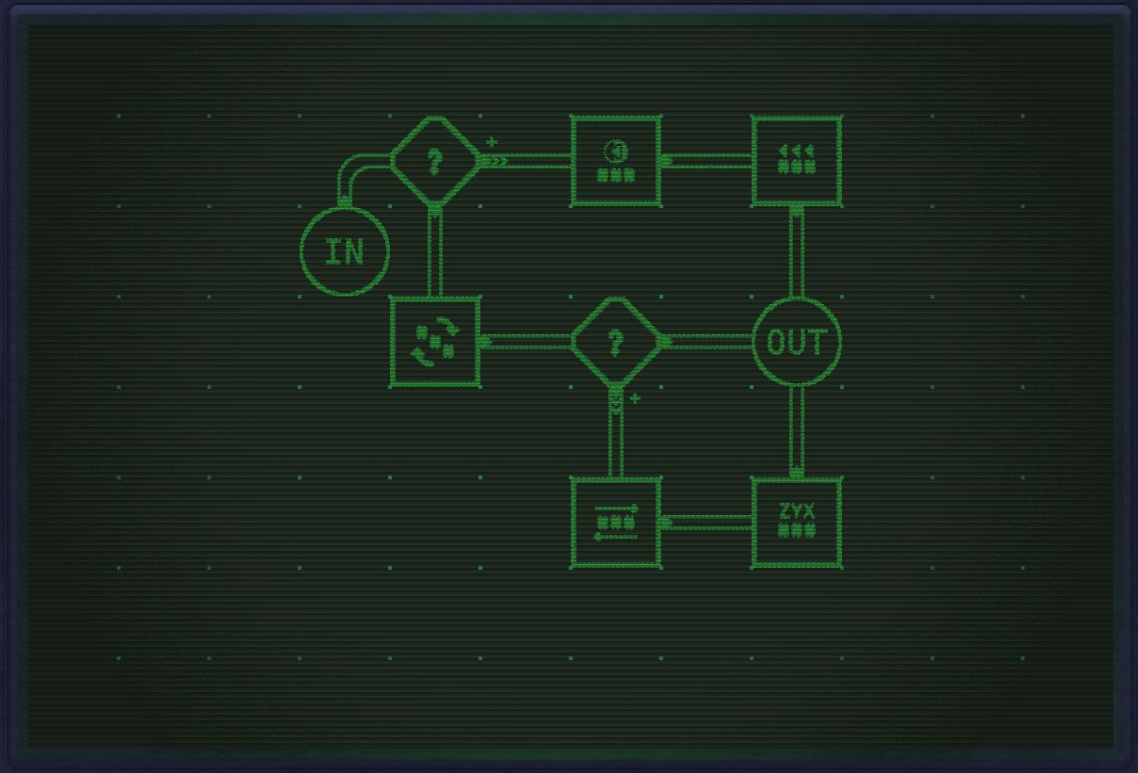

RETROWAVE CITY


PLAYER 1 | HP: 30 SPECIAL: 2


LEVEL 2
PLAYER 1
PRIMARY
GAUSS
SPECIAL
FLAMER
ARMOR
LEVEL 1
SPEED
LEVEL 2
ROTATION
LEVEL 2
BOOST
NONE
READY
CREDITS USED: 22/30

DECIPHERISM





ABC INPUT

? OUTPUT

READY STATUS

ACCESS KEYWORDS

XMVOVEZ OFF

IGRVY OFF

SLOW FAST

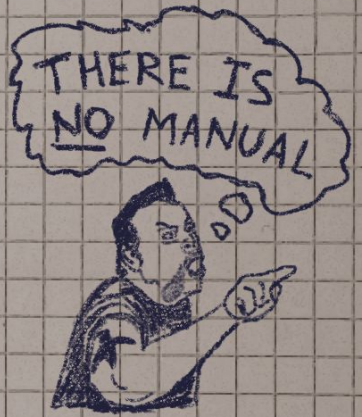
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Let The Hackery Begin

I am going to use the
Decoder terminal to bypass
the Hexagon security systems.

It has no manual, so I'm going
to write one myself.

(There are more pages, read on.)



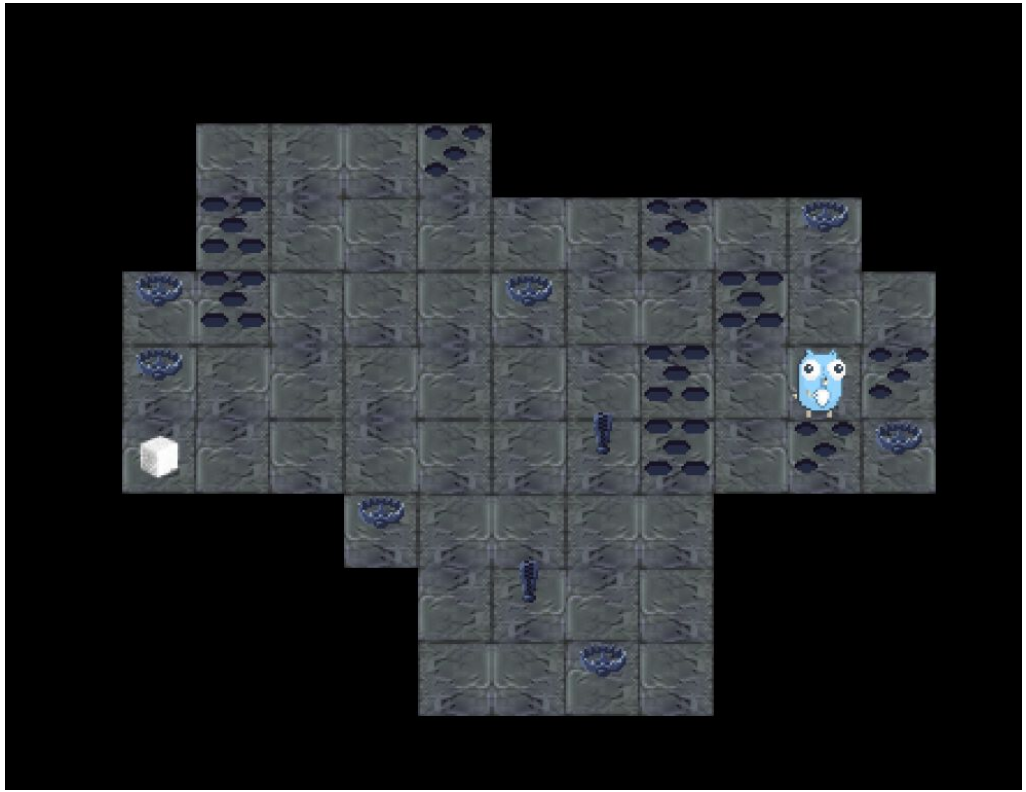
My published games

- Autotanks
- Retrowave City
- Decipherism

Live coding a small game using Ebiten (Russian)

- Part1
- Part2
- Sources

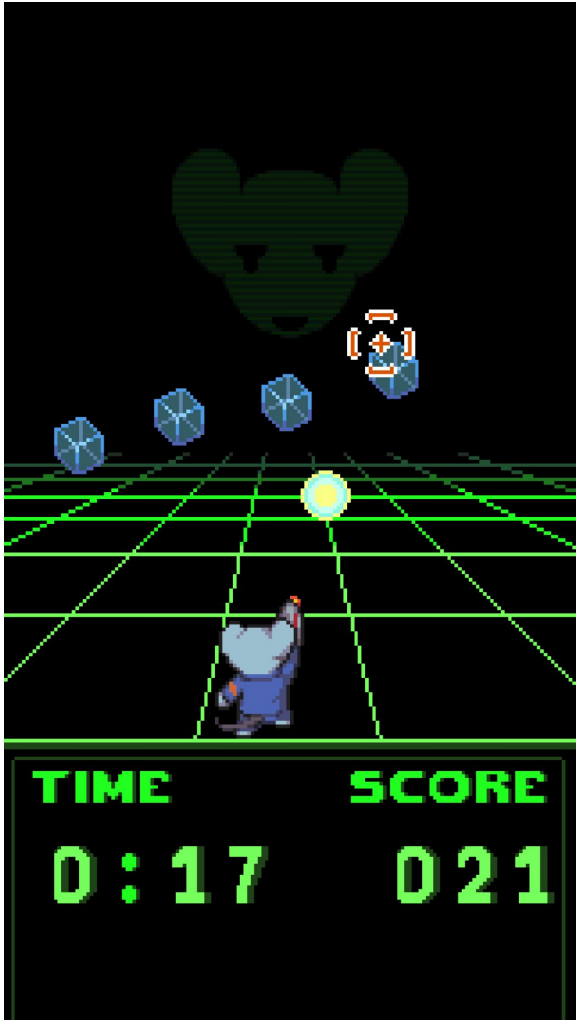
Subscribe @quasilyte on YouTube



Commercial Ebitengine Games


Bear's Restaurant


MOUSE
BUSTERS


TIME
SCORE
0:17
021


FISHING
PARADISO


0

Ebitengine & Nintendo Switch

Bear's Restaurant and Fishing Pradiso

are also available on Nintendo Switch!

More Ebitengine projects

See made-with-ebitengine collection on itch.io.

Also, Hajime usually re-twits Ebitengine projects, so check out his Twitter to find more interesting stuff.

Awesome-ebitengine repository lists some Ebitengine games (with links to their repositories).

Ebitengine community links

Most discussions happen at Discord channel (international, English).

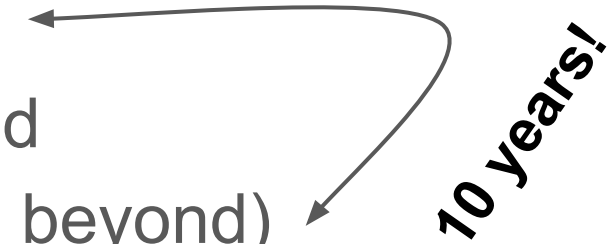
There is also /r/ebitengine.

I started a small Russian-speaking community in Telegram.

Ebitengine Core Overview

Ebitengine is not “something experimental”

- Repository created at **2013**
- June of **2016**: v1.0.0 is released
- It flourishes up to this day (and beyond)



Ebitengine is mature enough nowadays.

Ebitengine

- Game development framework written in Go
- Minimal game architecture restrictions
- Supports a fair range of platforms
- Mostly for 2D games

It feels like SDL for Go.

Ebitengine

- Game development framework written in Go
- Minimal game architecture restrictions
- Supports a fair range of platforms
- Mostly for 2D games

It feels like SDL for Go.

Ebitengine

- Game development framework written in Go
- Minimal game architecture restrictions
- Supports a fair range of platforms
- Mostly for 2D games

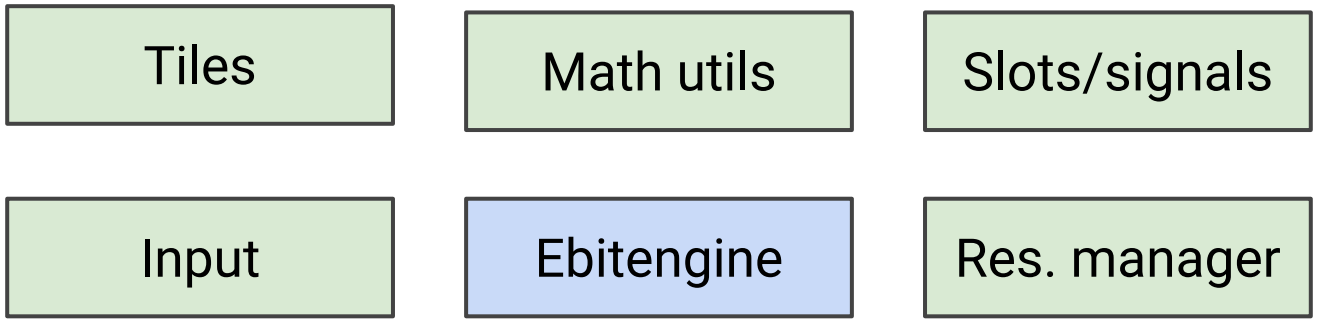
It feels like SDL for Go.

Ebitengine

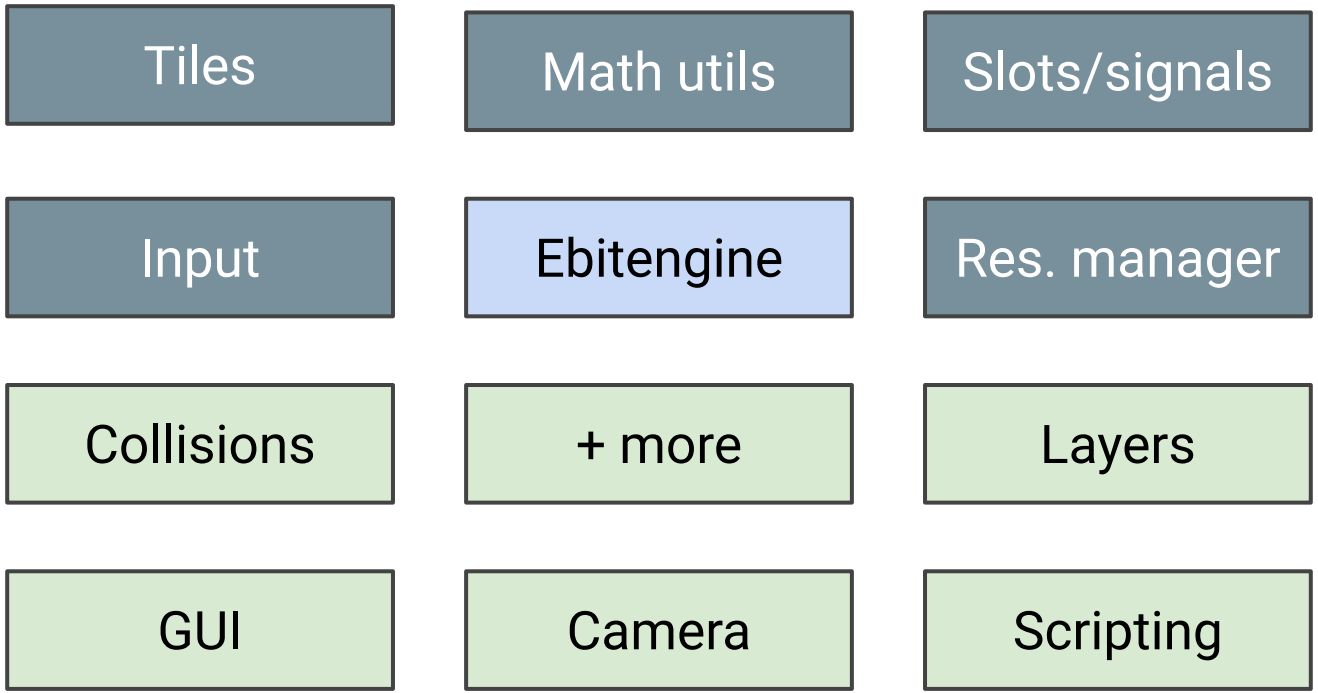
- Game development framework written in Go
- Minimal game architecture restrictions
- Supports a fair range of platforms
- Mostly for 2D games

It feels like SDL for Go.

We’ll start from here



Those are covered in my other presentation.
I'll mention them here too nonetheless,
but in much less details



We'll mostly focus on these today

Ebitengine game interface

- Separate update/draw loops
- Fixed TPS idiom (no “time delta”)
- You do ebiten.Run() explicitly

Ebitengine gives you a freedom of choice here.

You can choose whatever architecture you like.

Ebitengine game interface

- Separate update/draw loops
- Fixed TPS idiom (no “time delta”)
- You do ebiten.Run() explicitly

Ebitengine gives you a freedom of choice here.

You can choose whatever architecture you like.

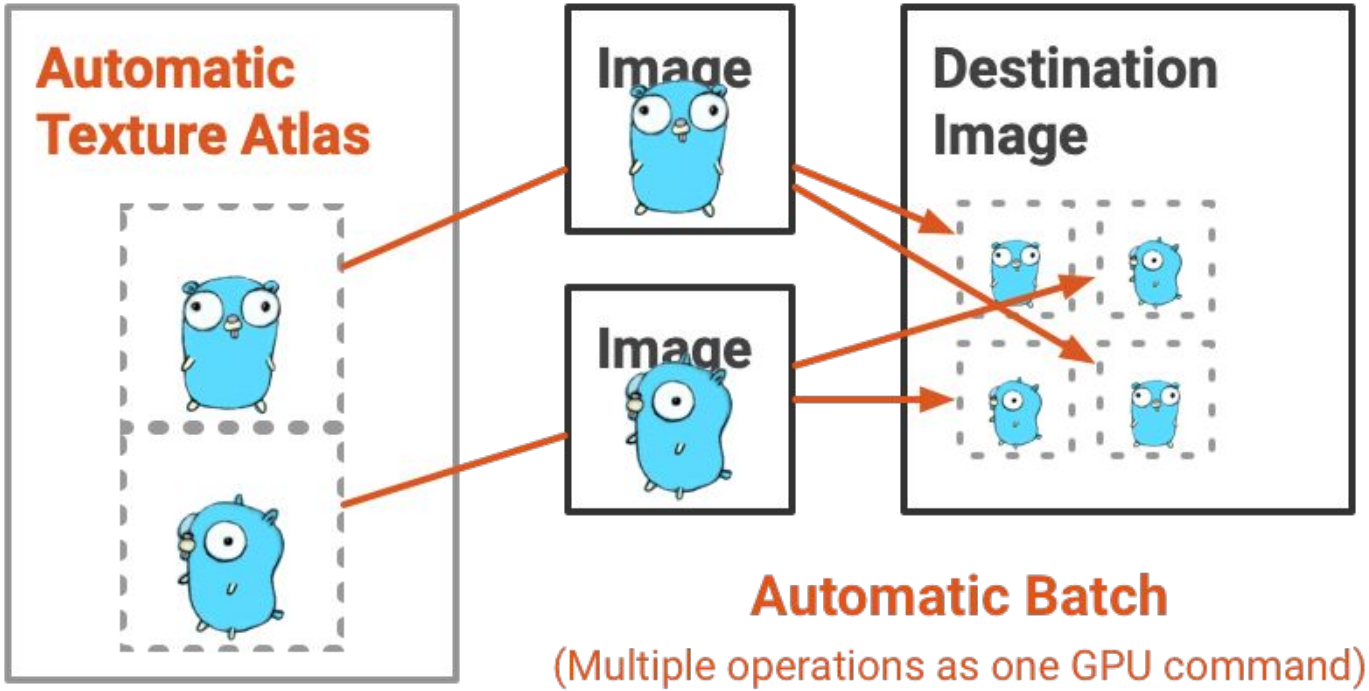
Ebitengine game interface

- Separate update/draw loops
- Fixed TPS idiom (no “time delta”)
- You do ebiten.Run() explicitly

Ebitengine gives you a freedom of choice here.

You can choose whatever architecture you like.

Ebitengine rendering



Ebitengine features: graphics

- Simple image rendering API
- Image transformations API
- Text rendering, good fonts support
- Shaders support
- Limited path drawing support (via vector package)

Ebitengine features: input handling

- Rich devices support (gamepads, keyboards, etc.)
- inpututil package that provides some helpers
- Intuitive API

Ebitengine features: audio support

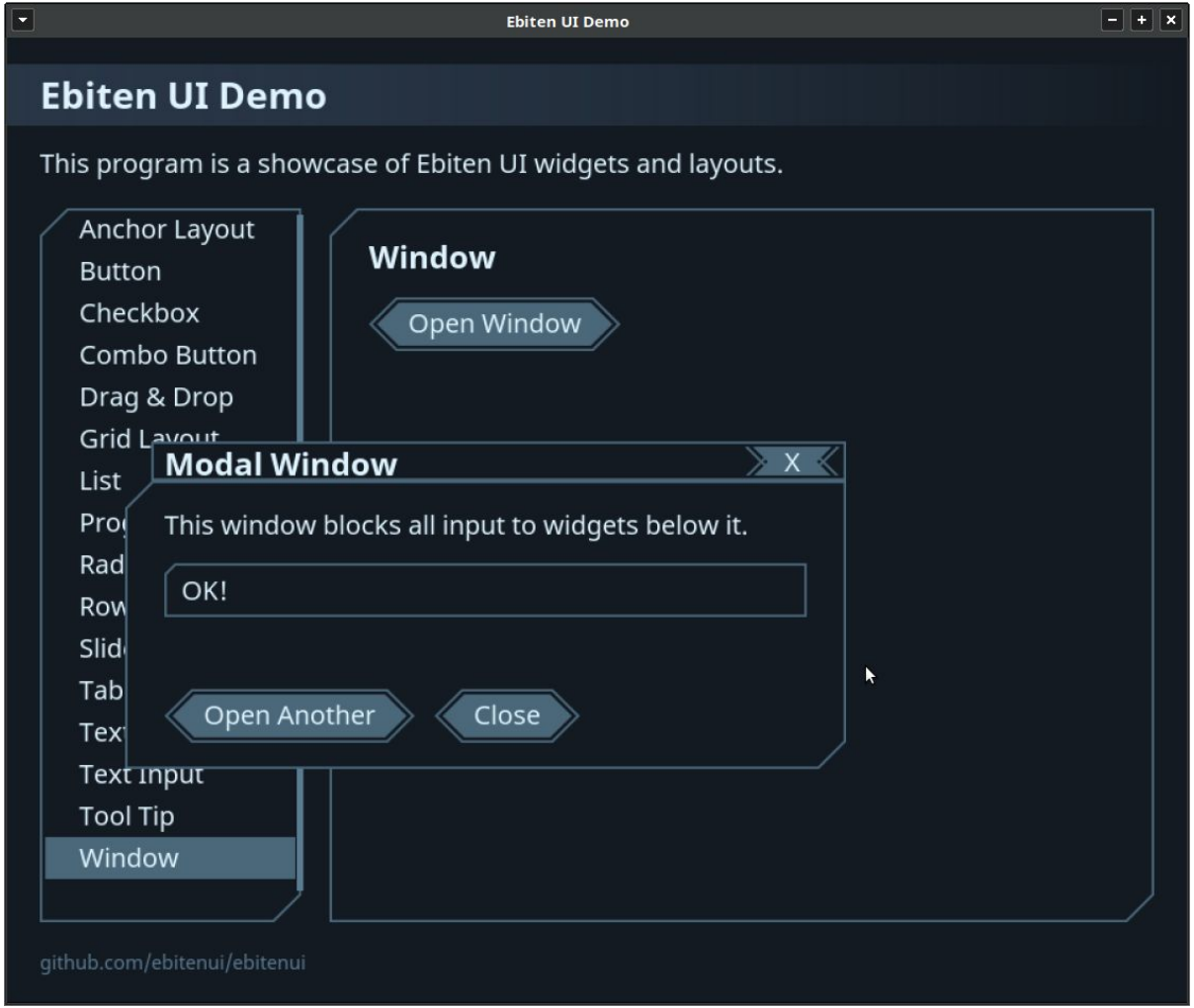
- Can work with ogg, mp3, wav, pcm
- Can re-encode with different sample rate

Graphical User Interface

ebitenui

github.com/ebitenui/ebitenui is a good library if your game requires more than just a couple of buttons.


Ebiten UI Demo
Ebiten UI Demo
This program is a showcase of Ebiten UI widgets and layouts.
Anchor Layout
Button
Checkbox
Combo Button
Drag & Drop
Grid Layout
List
Progress Bar
Radio Group
Row Layout
Slider
Tab Book
Text Area
Text Input
Tool Tip
Window
List
One
Two
Three
Four
Five
Six
Seven
Action 1
Action 2
Action 3
Eleven
Twelve
Thirteen
Fourteen
Fifteen
Sixteen
Seventeen
Disabled
github.com/ebitenui/ebitenui


Ebiten UI Demo
Ebiten UI Demo
This program is a showcase of Ebiten UI widgets and layouts.
Anchor Layout
Button
Checkbox
Combo Button
Drag & Drop
Grid Layout
List
Pro
Rad
Row
Slid
Tab
Tex
Text Input
Tool Tip
Window
Window
Open Window
Modal Window
X
This window blocks all input to widgets below it.
OK!
Open Another
Close
github.com/ebitenui/ebitenui

ebitenui API

- Uses a custom events system
- Uses functional options pattern a lot
- Accepts *ebiten.Image as an input
- Uses ints for measurements, coordinates

It may be incompatible with the rest of your game.

API wrapping can help here.

ebitenui API

- Uses a custom events system
- Uses functional options pattern a lot
- Accepts *ebiten.Image as an input
- Uses ints for measurements, coordinates

It’s not a real problem but I personally don’t like it.

ebitenui API

- Uses a custom events system
- Uses functional options pattern a lot
- Accepts *ebiten.Image as an input
- Uses ints for measurements, coordinates

This is very good. You can handle the image objects caching and loading on your side.

ebitenui API

- Uses a custom events system
- Uses functional options pattern a lot
- Accepts *ebiten.Image as an input
- Uses ints for measurements, coordinates

This may result in many float64->int conversions around the UI-related code. Wrappers could help here too.

Using ebitenui

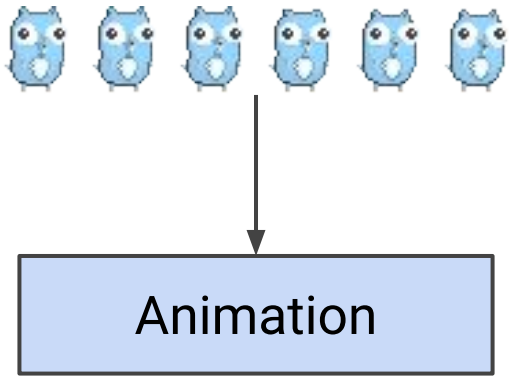
All in all, it looks like a promising library (use it!)

I highly recommend to wrap it, so you can benefit from the implementation while keeping the API surface more consistent with the rest of your game code.

Smash that ⭐ button! -> github.com/ebitenui/ebitenui

Animation / Tween

Spritesheet



Library options

- yohamta/ganim8 (anim8-inspired library)
- tanema/gween (tween)
  - SolarLune/gween (it’s a fork)

Both of them are good enough.

Tweens are useful for more than just animating things.

tanema/gween

- Uses float32 in its API
- Not Ebitengine-centric
- It’s quite lightweight

It doesn’t “play” the animation for you, but you can use it to interpolate the frames.

ECS (Entity Component System)

Disclaimer

I haven’t tried any of those.

Artem Sedykh kindly provided their feedback on the subject. Artem works on a new ECS framework (mizu), so this experience report is worthwhile.

Library options

- sedyh/mizu (54 ⭐)
- yohamta/donburi (95 ⭐)
- andygeiss/ecs (68 ⭐)
- leopotam/goecs (14 ⭐)
- 33blue/wecqs (archived, but could be a good read)

+ more!

sedyh/mizu

- WIP, but could be promising
- Less performance, but more convenience
- Querying support: pointer swap & generics

Documentation: good.

yohamta/donburi

- Has archetypes support
- The most performant choice right now
- Requires some boilerplate from the user
- Public API uses generics

Documentation: good.

andygeiss/ecs

- Quite fast
- Requires lots of boilerplate from the user
- Requires type assertions / casting

Documentation: good.

Benchmarks

Max objects at stable 60 fps:

```
ebitengine @5ee32bba: 40000 objects
donburi@d5a1431b:     33000 objects
gohan  @d6e94392:     28000 objects
mizu   @9387a0dd:     22000 objects
wecqs  @2a143f5e:     18000 objects
```

The state of ECS in Go

Since generics are relatively new concept, the authors of ECS are trying to see what are the limits and how can we create both convenient and efficient ECS system.

Perhaps Artem Sedykh will make a detailed talk about it one day. Let’s look forward to it.

Working with Text

Multi-language games

- It’s good when game can infer the default language
- In-game texts should not be hardcoded

It could be OK-ish to use English as a default, but not everyone know that language. You may need some very easy to understand language switch buttons to make the game more accessible for the players.

Matching the languages

To infer a good default language, you need to match the system language with dictionaries available first.

golang.org/x/text/language can be useful to match and compare the languages.

See also: golang.org/x/text/collate

ქართული

KARTULI

START

LEVEL: 1

CHALLENGE: ALPHABET

LANGUAGE: EN

ქართული

KARTULI

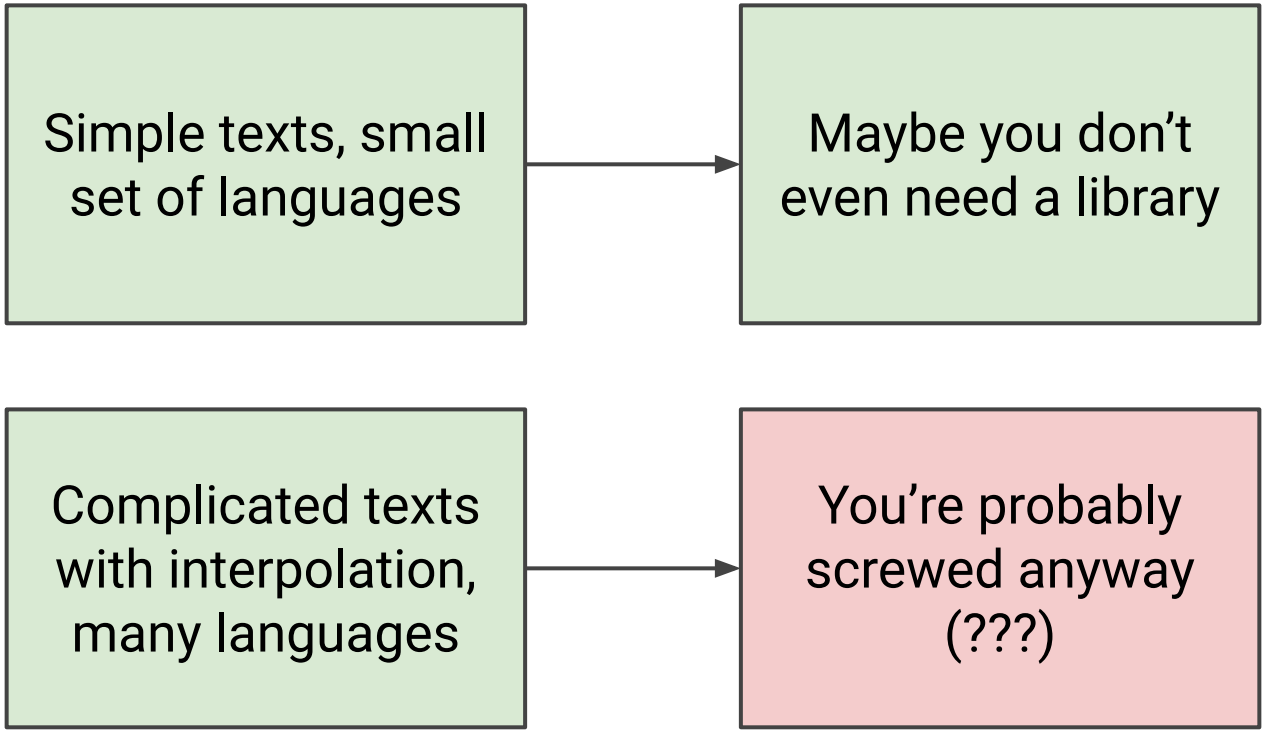
НАЧАТЬ

УРОВЕНЬ: 1

УПРАЖНЕНИЕ: АЛФАВИТ

ЯЗЫК: RU

How complex your game texts are?

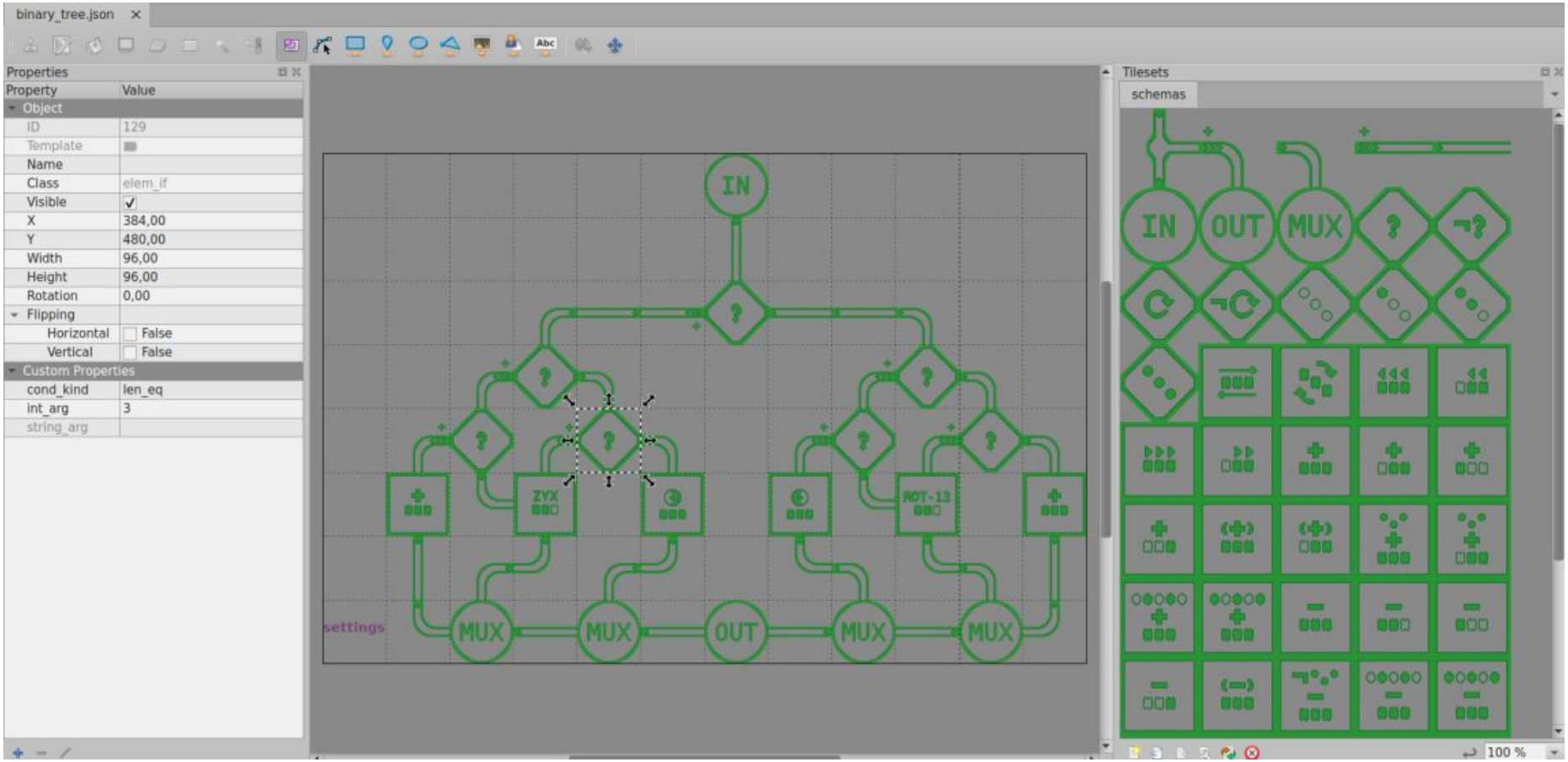


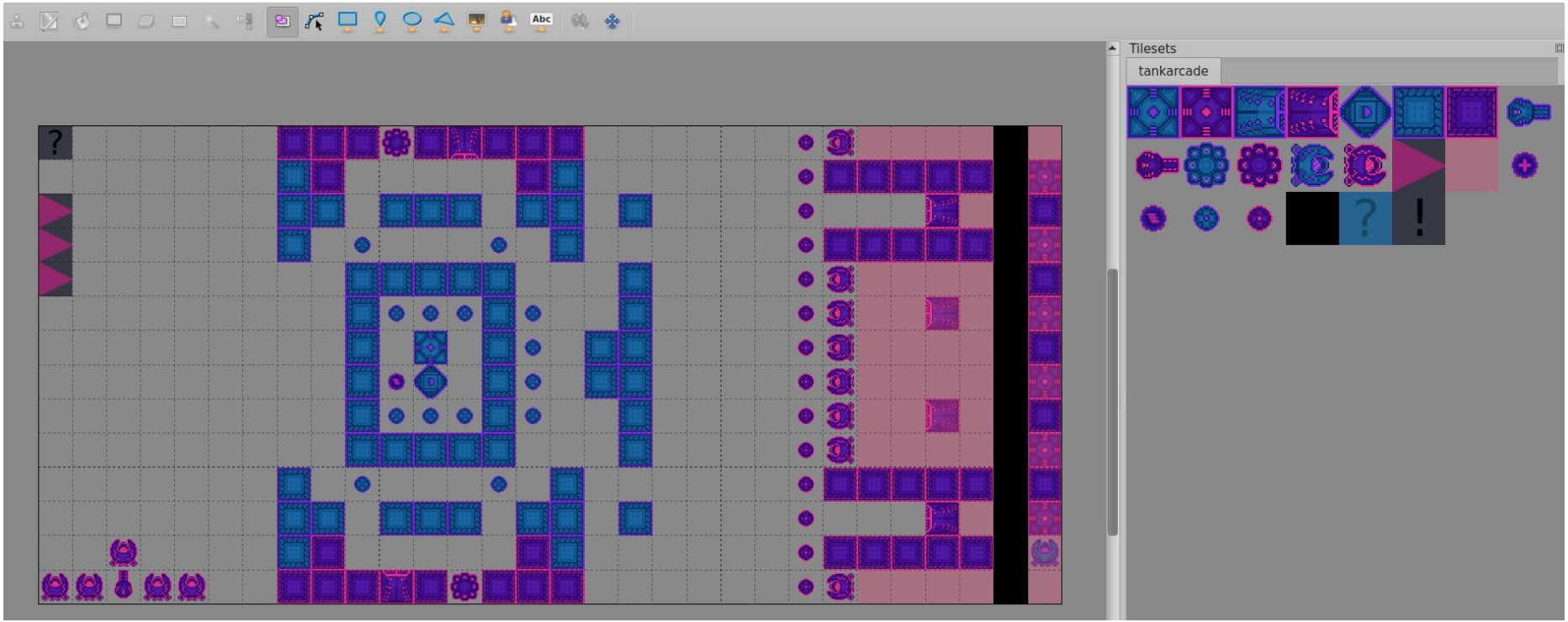
Tiles and Map Editor

Tiled!

- Tiled is a graphical map editor
- There is go-tiled package

Tiled also has simple JSON export format, so you can use it without any special libraries.


binary_tree.json
Properties
Property
Value
Object
ID
129
Template
Name
Class
elem_if
Visible
X
384,00
Y
480,00
Width
96,00
Height
96,00
Rotation
0,00
Flipping
Horizontal
False
Vertical
False
Custom Properties
cond_kind
len_eq
int_arg
3
string_arg
IN
?
ZYX
ROT-13
MUX
OUT
settings
Tilesets
schemas
100 %


Tilesets
tankarcade

Collisions Detection

Library options

- SolarLune/resolv (314 ⭐)
- jakecoffman/cp (201 ⭐)
- zergon321/cirno (20 ⭐)
- ByteArena/box2d (254 ⭐)

There are probably more libraries out there, but they escaped me successfully.

How to choose?

SolarLune/resolv is quite popular, has idiomatic API and it's easy to get started.

zergon321/cirno is another good choice, it has oriented bounding box support.

cp is more low-level and feature-rich.

Scripting

Why scripting?

Sometimes it simplifies the content creating for your team and your users.

User-created maps in WarCraft III wouldn’t be as great if there were no scripting support!

Scripting makes it possible to extend the game without having to re-build it (good for the players).

Yaegi

Good Go language support. Easy to integrate. A little bit slow right now, but it may improve in that regard.

github.com/traefik/yaegi

Scriggo

Less widely known, but it’s quite fast. Harder to integrate. May be less maintained nowadays.

github.com/open2b/scriggo/

Quasigo

My own creation. Used in ruleguard. Supports only limited subset of Go. I intend to keep it the fastest Go interpreter written in Go.

github.com/quasilyte/quasigo

Lua (or other languages)

You can use one of the few lua interpreters implementation for Go.

There are plenty of them.

I’m pretty sure there are dozens of various Lisps implementations too.

Path Finding

Library options

- beefsack/go-astar
- SolarLune/paths
- fzipp/astar

SolarLune/paths

- Grid cell size is easily configurable
- Very slow (1000+ allocs on one call, ~200k ns time)
- Supports diagonal moves
- Outdated readme, weird go.mod (import *path* case)

I can’t recommend it. :(

beefsack/go-astar

- Slow as well
- Inconvenient to use
- Seems unmaintained

I can’t recommend it. :(

fzipp/astar

Not very impressive either. :(

What to do?

- If your maps are small: do/use whatever
- If your maps are big, implement your own algorithm

An advice

You’re about to create a new path finding lib?

- Try avoiding redundant allocs (slices, etc)
- Do not use “container/heap”
- Whether possible, use generics instead of interface{}

The devil is in the details.

An advice

You’re about to create a new path finding lib?

- Try avoiding redundant allocs (slices, etc)
- Do not use “container/heap”
- Whether possible, use generics instead of interface{}

The devil is in the details.

An advice

You’re about to create a new path finding lib?

- Try avoiding redundant allocs (slices, etc)
- Do not use “container/heap”
- Whether possible, use generics instead of interface{}

The devil is in the details.

Vector (2D) Math

Library options

- quartercastle/vector (used in resolv)
- quasilyte/gmath (used in my games)

There are probably tons of those.

Pick the one you like or write your own.

quartercastle/vector

- Used in resolve (yeah, I already said that)
- Uses float64 slices for vectors

I haven’t tried it since [ ]float64 for Vec2D is a red flag for me. This type is too ubiquitous in 2D games.

quasilyte/gmath

- Godot-inspired API
- gmath.Vec is a simple X,Y struct (no slices involved)
- Compatible with ebitengine-input package
- Includes utility things like convenient Rand sources

gmath works for me, maybe it’s good enough for you too.

quasilyte/gmath

- Godot-inspired API
- gmath.Vec is a simple X,Y struct (no slices involved)
- Compatible with ebitengine-input package
- Includes utility things like convenient Rand sources

gmath works for me, maybe it’s good enough for you too.

quasilyte/gmath

- Godot-inspired API
- gmath.Vec is a simple X,Y struct (no slices involved)
- Compatible with ebitengine-input package
- Includes utility things like convenient Rand sources

gmath works for me, maybe it’s good enough for you too.

quasilyte/gmath

- Godot-inspired API
- gmath.Vec is a simple X,Y struct (no slices involved)
- Compatible with ebitengine-input package
- Includes utility things like convenient Rand sources

gmath works for me, maybe it's good enough for you too.

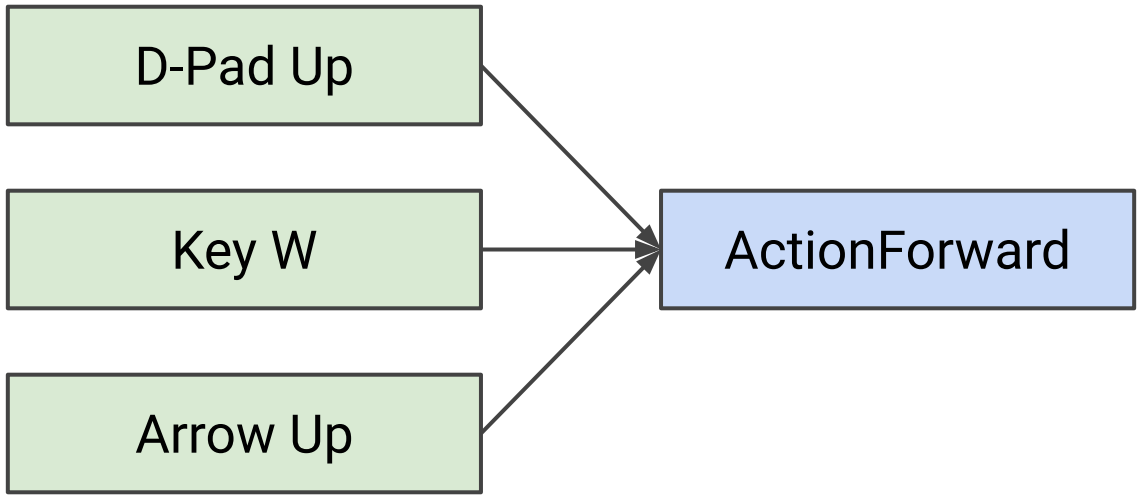
User Input Mapping

Why extra library for the input handling?

- Input-device agnostic event checking
- N-to-M action-key mapping support
- Convenient re-configuration during the run-time
- Virtual (simulated) input events
- Easier modifiers bindings (ctrl+c, etc)

```
go get github.com/quasilyte/ebitengine-input
```

N keys to 1 action mapping



Signals/Slots

Why bother and use signals/slots?

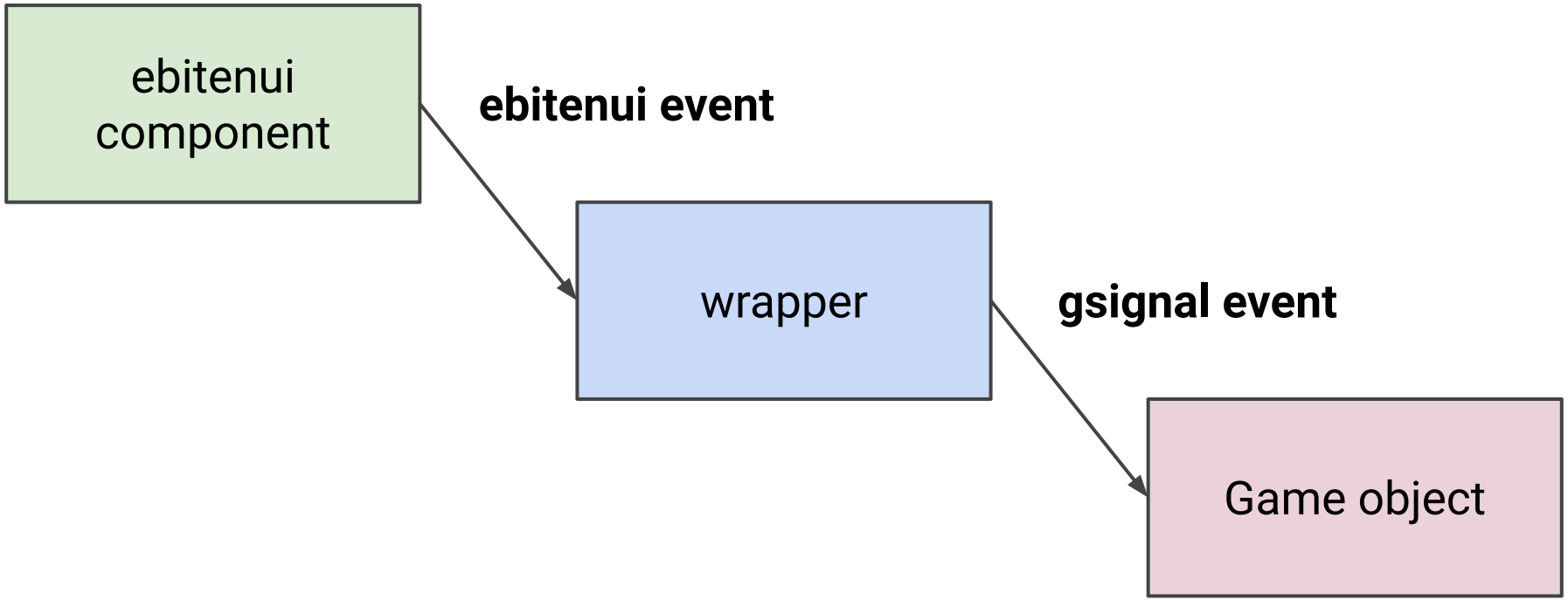
- It reduces the objects coupling
- It's an elegant event listener solution for Go
- Signals are a familiar concept (Godot, Phaser, Qt, ...)

quasilyte/gsignal

- Godot/Qt inspired signals library
- Amortized to zero allocs in most cases
- Efficient Connect/Disconnect/Emit
- Type safe API with generics

```
go get github.com/quasilyte/gsignal
```

Combining with ebitenui events



Resource Manager / Loader

quasilyte/ebitengine-resource

- Resource caching
- Int-based resource keys => efficient and convenient
- Allows metadata binding

```
go get github.com/quasilyte/ebitengine-resource
```

Conclusions

My gamedev starter kit

```
go get github.com/hajimehoshi/ebiten
go get github.com/ebitenui/ebitenui
go get github.com/SolarLune/resolv
go get github.com/quasilyte/gmath
go get github.com/quasilyte/gsignal
go get github.com/quasilyte/ebitengine-input
go get github.com/quasilyte/ebitengine-resource
```

What else you might need?

- Some scene tree framework
- Viewports/camera if your game needs it
- Layers, things like YSort, etc.
- ADs/Telemetry things for real products maybe

Ebitengine: pros

- Allows you to write games in Go!
- Supports many platforms
- Relatively simple installation (few external deps)
- Easy to get started (but maybe hard to master?)
- Feature reach core (shaders, audio, etc.)
- Good docs, many examples
- An active community

Ebitengine: cons

- Mobile platform targets could be challenging
- Low performance in wasm build
- Hard to write a library without Ebitengine dependency
- Very limited touch (low-level) input support

Do I recommend you to try out Ebitengine?

- Yes

Just give it a try and see if it clicks with you.

Pros: Ebitengine

Cons: None (wink)

How to keep the code sane?

Using multiple 3rd party libraries will make your code look less consistent due to different API styles.

I suggest to wrap 3rd party libraries and use them via your own APIs. This will also help you to change the 3rd party dependency later if you'll ever want to do that.

This is better than writing everything from scratch.

Want to get involved? Want to help Ebitengine?

- Consider becoming an Ebitengine sponsor
- Contribute to Ebitengine (or submit bug reports)
- Credit/mention **#ebitengine** in your game or content
- Contribute to cool libraries or create your own
- Be an active community member (Discord channel)

Created a game? Want to share it with community?

- Tweet using **#ebitengine** tag
- Post in **show-and-tell** channel in Discord
- Create a reddit post in **/r/ebitengine**
- For itch.io, add game to made-with-ebitengine
- Open source? => submit to awesome-ebitengine
- Let me know and I'll stream it

If you love Godot, why do you use Ebitengine?

I like Go more than gdscript or C#.

It’s not only about the language:

- I like Go communities
- I like the 3rd party tooling (and the “go tool” itself)
- I am feeling at home here

ENGLISH CONTENT

QUASILYTE

INDIE GAMEDEV & GAMING

Ebitengine Ecosystem Overview

quasilyte @ podlodka 2023