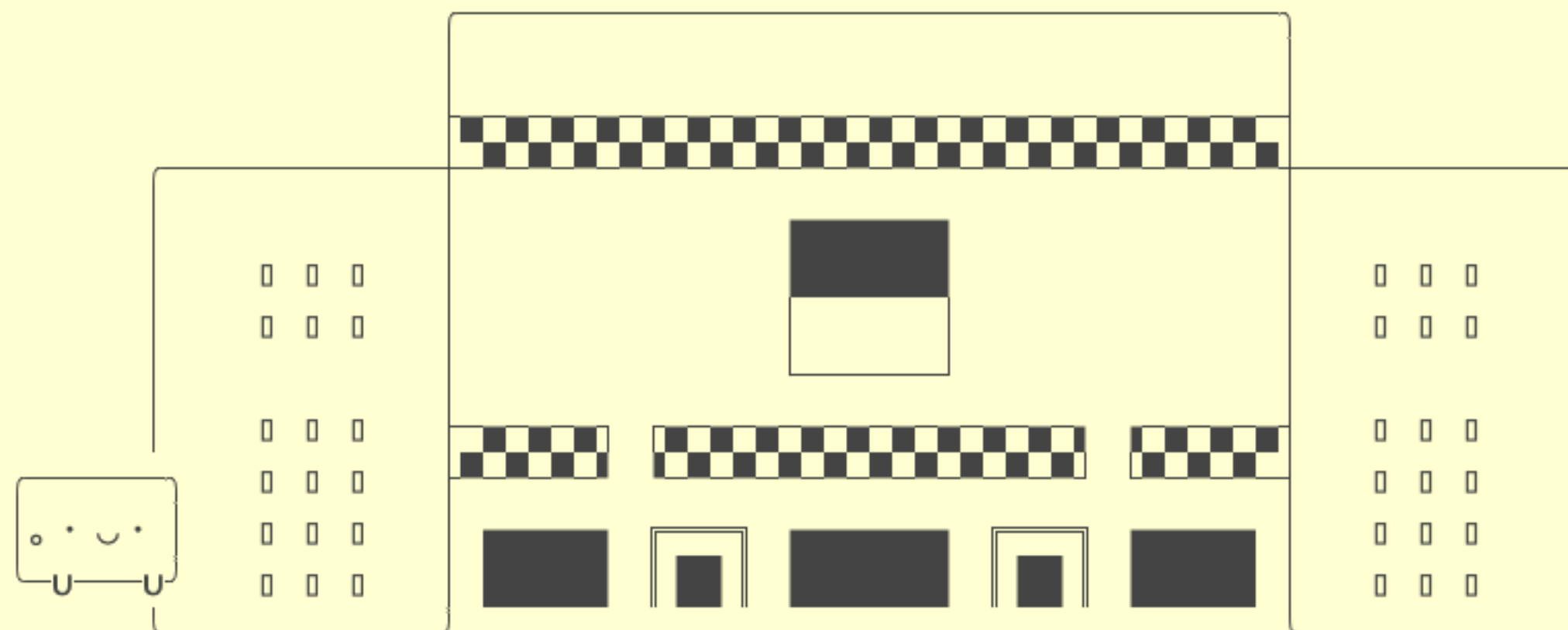
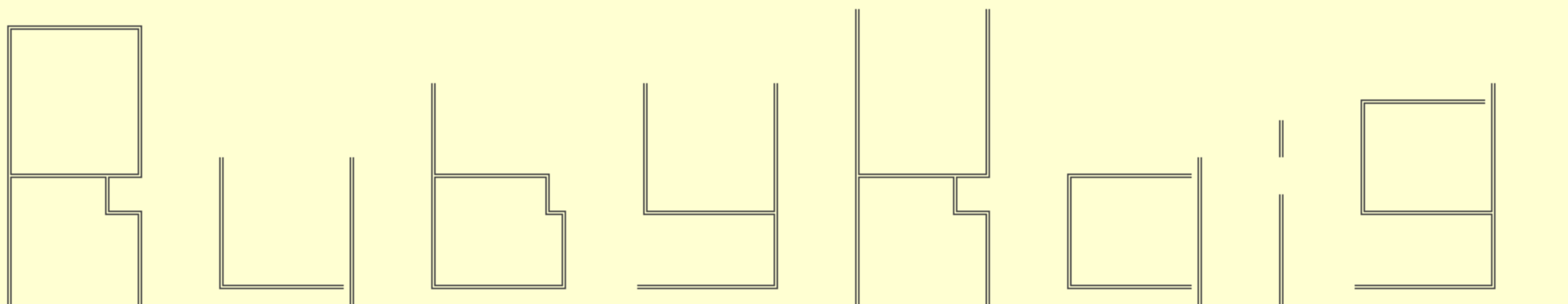
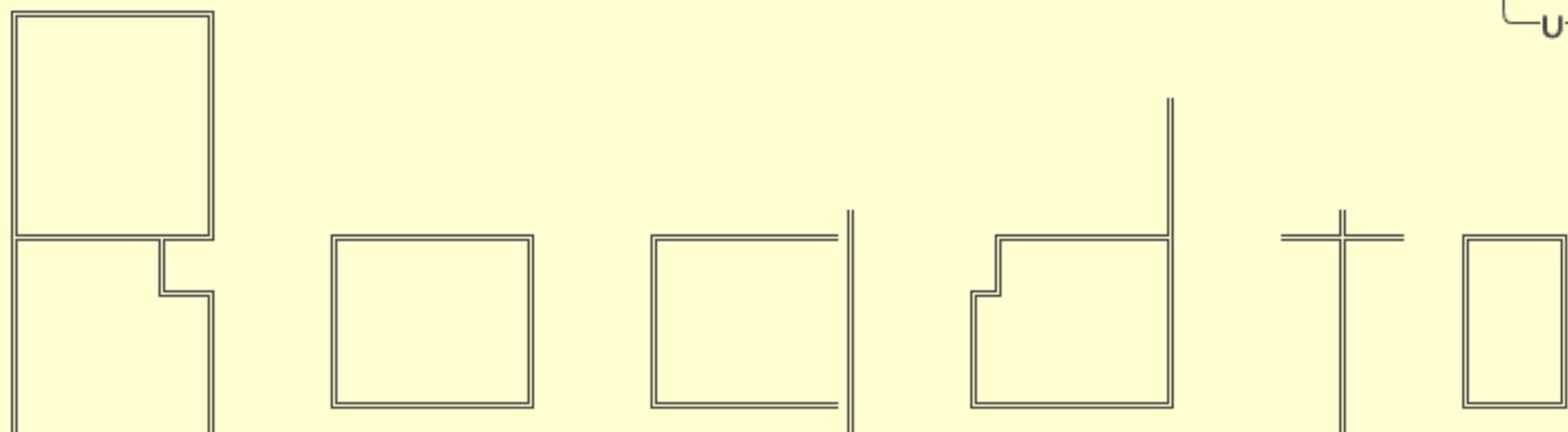


# Ruby でつくる

## CLI横スクロール アクションゲーム



@makicamel  
TokyoWomen.rb #1  
2025.03.01

# 自己紹介

- @makicamel / 川原万季
- Ruby 💎 とビール 🍺 とお酒が好き
- (株)アンドパッド



# Ruby といえば

# Ruby といえば

- Ruby on Rails
- Web アプリケーション

昼間のお仕事で  
Ruby 書けてうれしい



# Ruby といえば

- Ruby on Rails
- Web アプリケーション

それ以外だと？



# Ruby といえば

# RubyKaigi

## 2024 MAY15-17

@NAHA, OKINAWA

那覇文化芸術劇場なはーと

📍 **NAHA CULTURAL ARTS THEATER NAHArt, Okinawa, Japan**

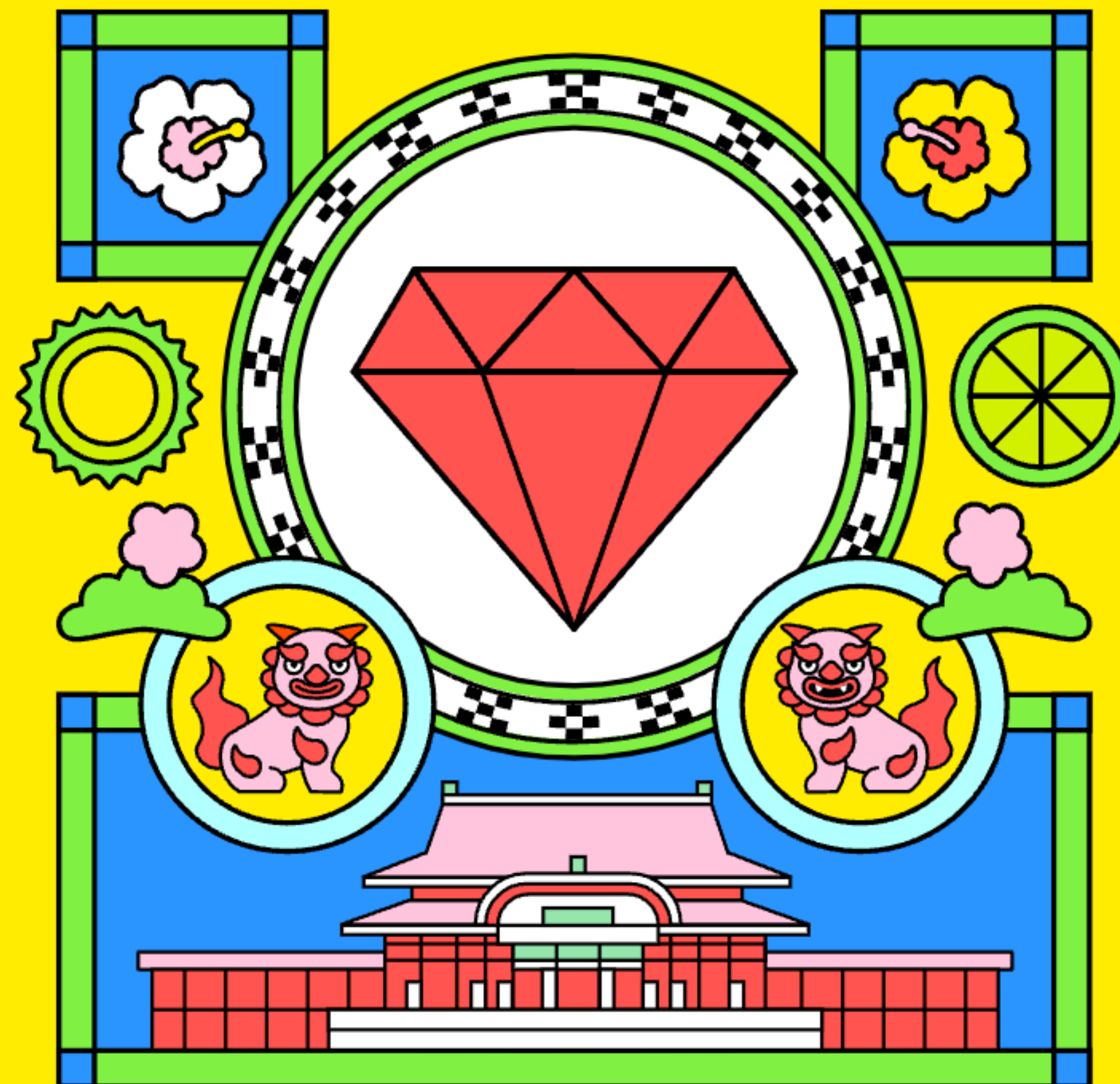
(In-person only)

### Keynote Speakers

Yukihiro "Matz" Matsumoto

tomoya ishida

Samuel Williams



RubyKaigi 2024 is over. Thank you everyone who made it.

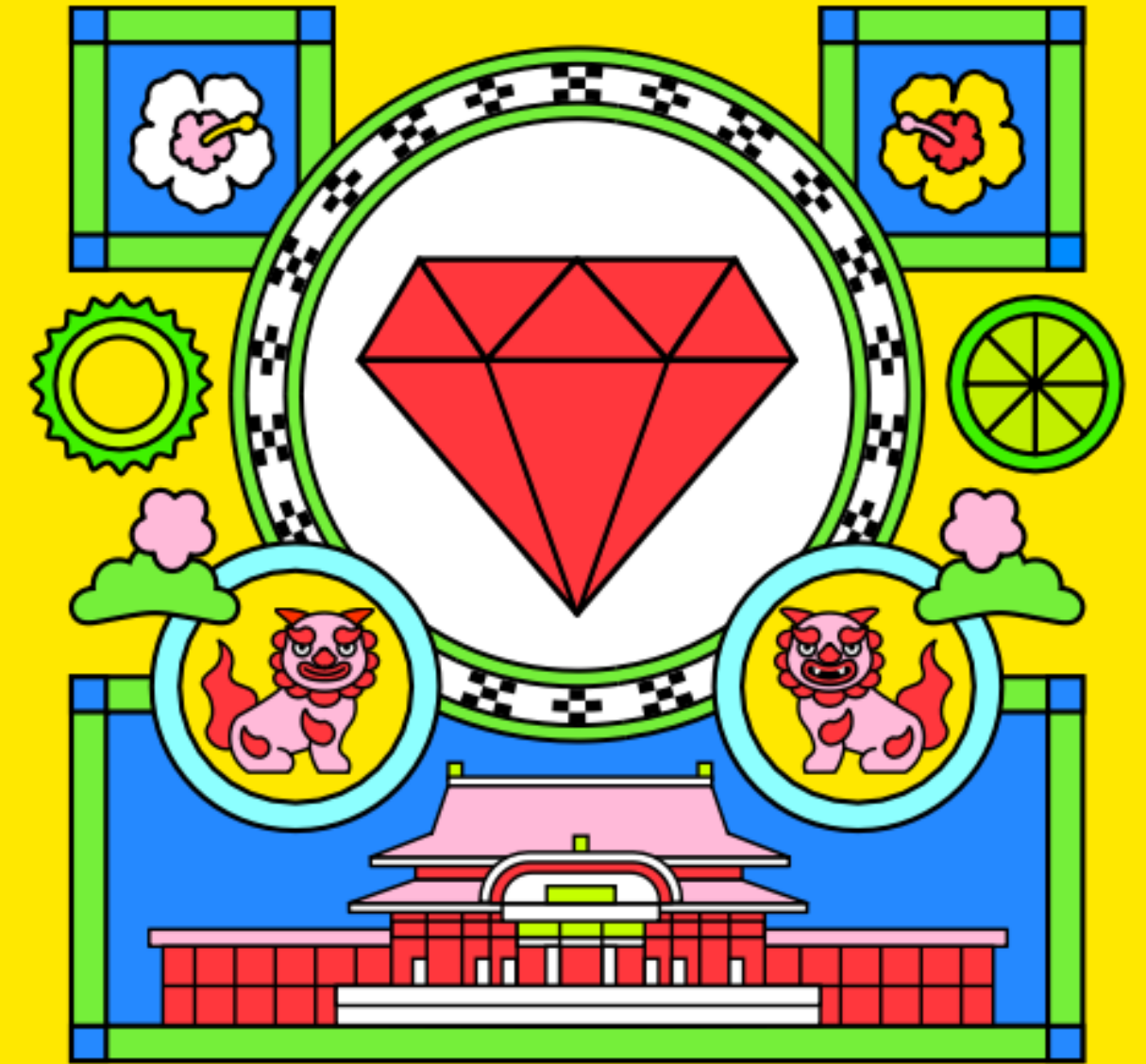
See you at RubyKaigi 2025 in **Matsuyama!**



# Writing Weird Code



**PEN @tompng**  
**tomoya ishida**



**RubyKaigi**  
**2024** **MAY15-17**  
@NAHA, OKINAWA



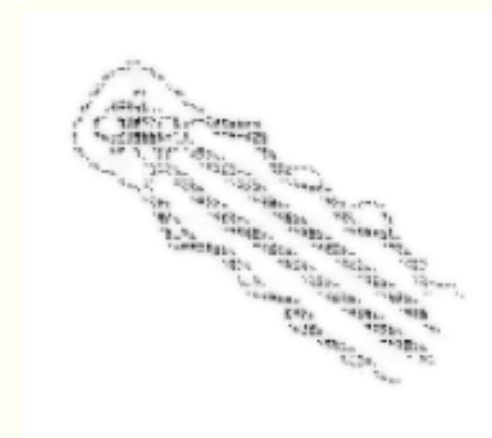
# Self TRICK 2024



Most Floral



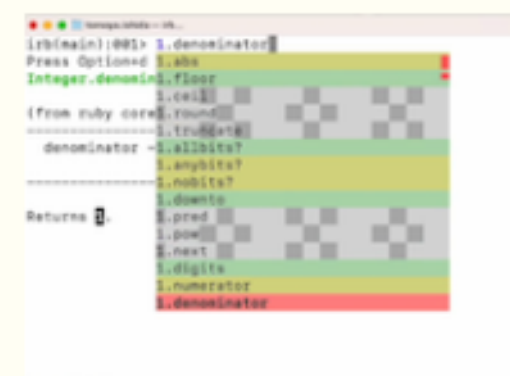
Most Dangerous



Most Fresh



Most Kaigi-ish



Most Wavy



Most Eternal



<https://github.com/tompng/selftrick2024>

格好いい…!!!



なにか つくりたい



つくった

makicamel / road\_to\_rubykaigi

Type / to search

<> Code

Issues

Pull requests

Actions

Projects

Security

Insights

Settings

road\_to\_rubykaigi

Public

Pin

Unwatch 1

Fork 0

Star 0

main

Go to file

t

+

<> Code

makicamel

Initial release

✓

e5fad4a · 5 minutes ago

102 Commits

.github/workflows	CI with 3.4.1	last week
bin	Add entry point	2 weeks ago
lib	Note down the characters to gener...	11 hours ago
sig	bundle gem road_to_rubykaigi	2 weeks ago
spec	Fix CI	last week
.gitignore	Ignore Gemfile.lock	2 weeks ago
.rspec	bundle gem road_to_rubykaigi	2 weeks ago
.standard.yml	Tweak link rule	14 hours ago
CHANGELOG.md	Initial release	5 minutes ago
CODE_OF_CONDUCT.md	bundle gem road_to_rubykaigi	2 weeks ago
Gemfile	bundle gem road_to_rubykaigi	2 weeks ago
LICENSE.txt	bundle gem road_to_rubykaigi	2 weeks ago
README.md	Update README	11 hours ago
Rakefile	bundle gem road_to_rubykaigi	2 weeks ago

About

A retro ASCII action game where Rubyist overcomes a looming deadline and bugs on your way to RubyKaigi

Readme

MIT license

Code of conduct

Activity

0 stars

1 watching

0 forks

Releases

1 tags

Create a new release

Packages

No packages published

Publish your first package

Languages

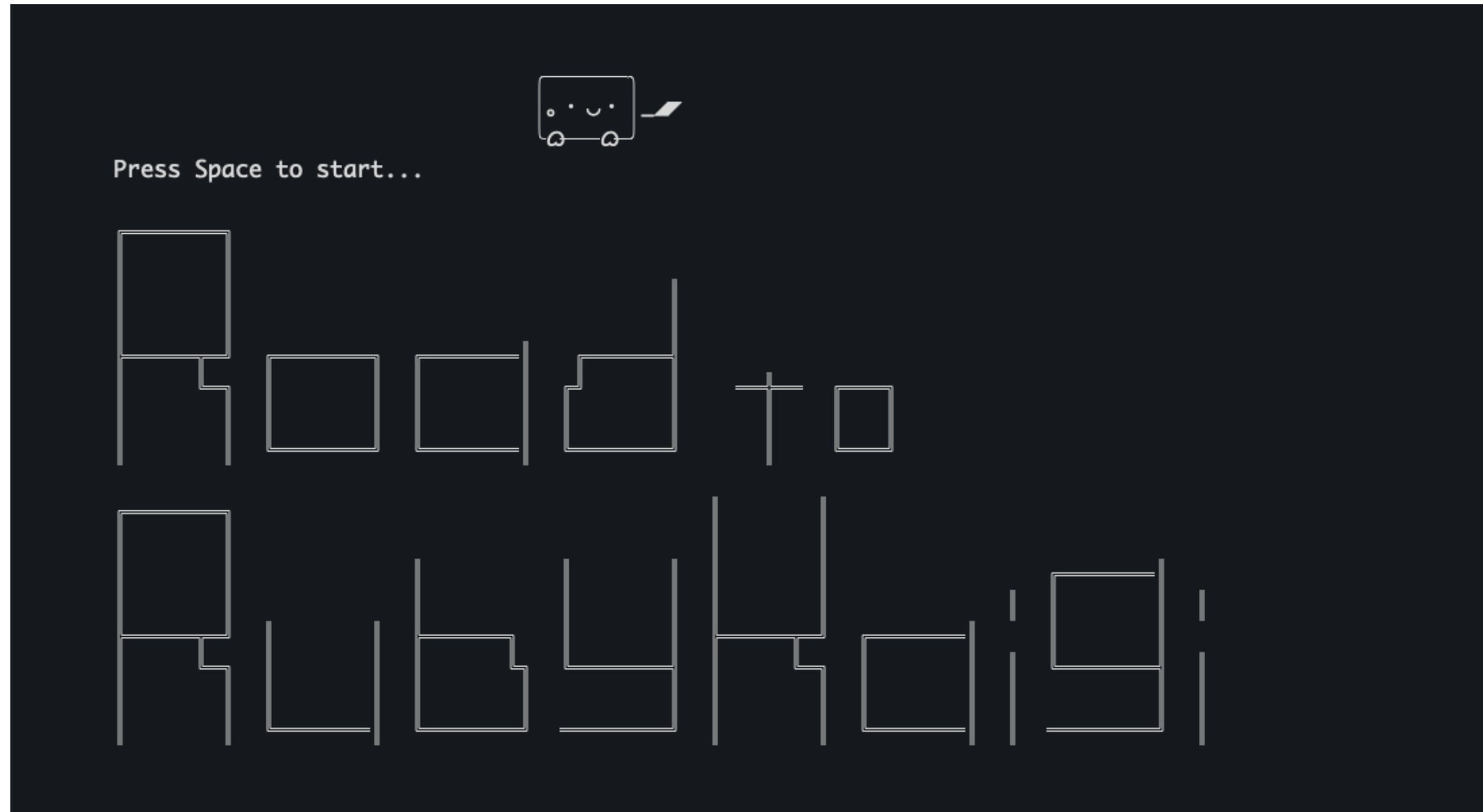
Road to RubyKaigi [https://github.com/makicamel/road\\_to\\_rubykaigi](https://github.com/makicamel/road_to_rubykaigi)

# Road to RubyKaigi

- ターミナル上で遊ぶ横スクロールアクションゲーム
- バグを倒しメ切から逃れて RubyKaigi 参加を目指す
- gem として配布
- `gem install road\_to\_rubykaigi` → `road\_to\_rubykaigi` で遊べる

# Road to RubyKaigi

- デモ



# Road to RubyKaigi のつくり方




# 基本編

# 端末ゲームプログラミングの基本

- アニメーション
- 入力
- ゲームループ

# 端末ゲームプログラミングの基本

- アニメーション 
- 入力
- ゲームループ

# 端末アニメーション

- パラパラ漫画
  - ①と②は足の位置が1コマ違う
    - 同じ位置に交互に表示させると足踏みしているように見える
    - 右の位置に移動し交互に表示させると歩いているように見える



# 端末アニメーション

- 位置
  - ANSI エスケープシーケンスで指定できる

# 端末アニメーション

- ANSI エスケープシーケンス
  - 端末上で色やカーソル位置などを制御するための文字列

# 端末アニメーション

- ANSI エスケープシーケンス
  - e.g. `\e[33m`: 次の文字からの文字色を黄色に変える

次の文字からの文字色を黄色にするシーケンス

helloが  
黄色く表示される

A terminal window with a dark background. The title bar shows three colored dots (red, yellow, green) and the text '-zsh' and 'ㄟ#2'. The prompt is a green '>'. The command 'ruby -e 'puts "\e[33m" + "hello"'' is entered. The output 'hello' is displayed in yellow. Below the command line, there is a status bar with '~ via [?]via [diamond icon] v3.4.2 on [cloud icon] (ap-northeast-1)'. A new green prompt '>' is visible below the status bar.

```
> ruby -e 'puts "\e[33m" + "hello"'
```

```
hello
```

```
~ via [?]via [diamond icon] v3.4.2 on [cloud icon] (ap-northeast-1)
```

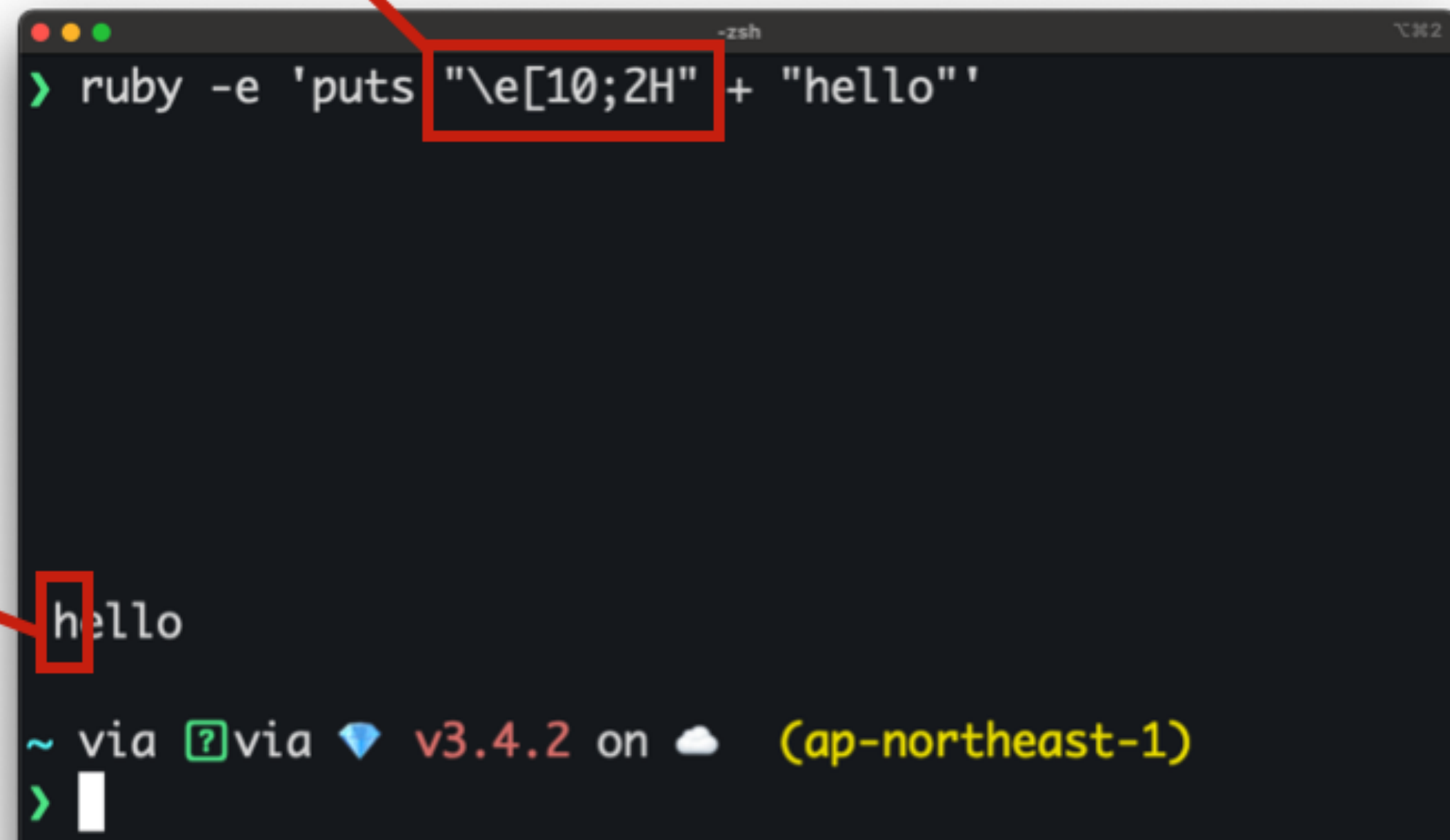
```
>
```



# 端末アニメーション

- ANSI エスケープシーケンス
  - e.g. `\e[1;1H`: 座標1;1の位置にカーソルを移動する
    - ※ `y;x`の順で指定する

x:2、y:10にカーソルを移動するシーケンス





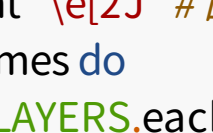
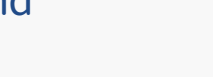


A terminal window with a dark background. The command prompt shows a Ruby command: `> ruby -e 'puts "\e[10;2H" + "hello"'`. The string `"\e[10;2H"` is highlighted with a red box. Below the command, the word `hello` is printed and also highlighted with a red box. The terminal status bar at the bottom shows the user is 'via' using 'via' v3.4.2 on a cloud instance named 'ap-northeast-1'.

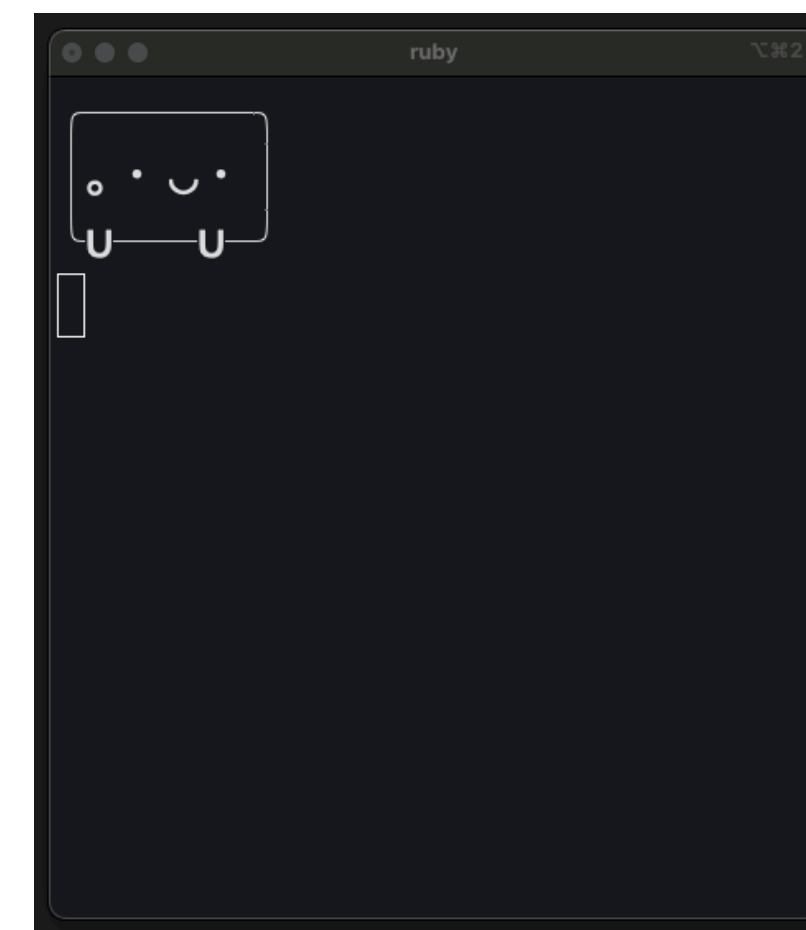
```
> ruby -e 'puts "\e[10;2H" + "hello"'  
  
hello  
  
~ via [?]via v3.4.2 on (ap-northeast-1)  
>
```

x:2、y:10 から  
helloが描画される


# 端末アニメーション

- その場で 5 回足踏みをするアニメーション

```
PLAYERS = [  
  <<~PLAYER,  
  ,  
  <<~PLAYER,  
  ,  
  <<~PLAYER,  
  ,  
  <<~PLAYER,  
  ,  
  <<~PLAYER,  
  ,  
  <<~PLAYER,  
  ,  
]  
print "\e[2J" # 画面クリア  
5.times do  
  PLAYERS.each do |player|  
    print "\e[1;1H" + player # 毎回1;1の位置に描画する  
    sleep 0.5  
  end  
end
```



# 端末ゲームプログラミングの基本

- アニメーション
- 入力 
- ゲームループ

# 端末入力

- キー入力を受け付けてキャラクターを操作する

# 端末入力

- 端末の入力モード
  - Cooked モード(カノニカルモード)
    - 入力文字列をバッファリングし改行でプログラムへ送信する
    - デフォルト
  - Raw モード(非カノニカルモード)
    - 入力文字列をバッファリングせずプログラムへすぐ送信する
    - ゲームなどリアルタイム性がほしいものの向き

# 端末入力

- 端末の入力モード
  - Ruby では `IO#raw` で Raw モードに変更する

```
STDIN.raw {  
  # 入力読み込み処理  
}
```

`IO#raw` <https://docs.ruby-lang.org/ja/latest/method/IO/i/raw.html>

# 端末入力

- IO#read\_nonblock
  - ノンブロッキングモードで IO からデータを読み込む

```
str = STDIN.readpartial(1) # ブロックする
puts :hello
puts str
# 文字入力するまで hello が表示されない

str = STDIN.read_nonblock(1, exception: false) # ブロックしない
puts :hello
puts str
# 文字入力前に hello が表示される
```

IO#readpartial <https://docs.ruby-lang.org/ja/latest/method/IO/i/readpartial.html>

IO#read\_nonblock [https://docs.ruby-lang.org/ja/latest/method/IO/i/read\\_nonblock.html](https://docs.ruby-lang.org/ja/latest/method/IO/i/read_nonblock.html)




# 端末入力

- Raw モード + ノンブロッキングモードで入力受け

```
STDIN.raw {  
  loop {  
    case STDIN.read_nonblock(3, exception: false)  
    when "\e[C" # right key  
      # キャラクターを右に1コマ動かす  
    when "\e[D" # left key  
      # キャラクターを左に1コマ動かす  
    when "q"  
      break  
    end  
    sleep 0.1  
  }  
}
```

# 端末ゲームプログラミングの基本

- アニメーション
- 入力
- ゲームループ 

# ゲームループ

- とは
- ゲーム実行中に無限に繰り返すメインループのこと

# ゲームループ

- 以下の処理を一定のフレームレートで繰り返す
  - 入力処理
  - 更新処理
    - e.g. キャラクターの状態変更、当たり判定
  - 描画処理

# ゲームループ

- ミニマムなゲームループ

```
PLAYERS = # ...
frame_index, x, x_min, x_max = 0, 10, 1, 20
STDIN.raw {
  loop {
    # 入力処理
    case STDIN.read_nonblock(3, exception: false)
    # 更新処理
    when "\e[C" # right
      x += 1
    when "\e[D" # left
      x -= 1
    when "\x03" # Ctrl+C
      exit
    end
    x = x.clamp(x_min, x_max) # 描画領域内に収める
    frame_index = (frame_index + 1) % 2
    # 描画処理
    print "\e[2J" # 画面クリア
    PLAYERS[frame_index].lines.each_with_index { |line, i| print "\e[#{i+1};#{x}H" + line }
    sleep 0.5 # 2FPS
  }
}
```

# ゲームループ

- 「←」「→」を受け付けて左右に動くゲームができた 🎉




# Road to RubyKaigi 編



# 端末ゲームプログラミング実践編

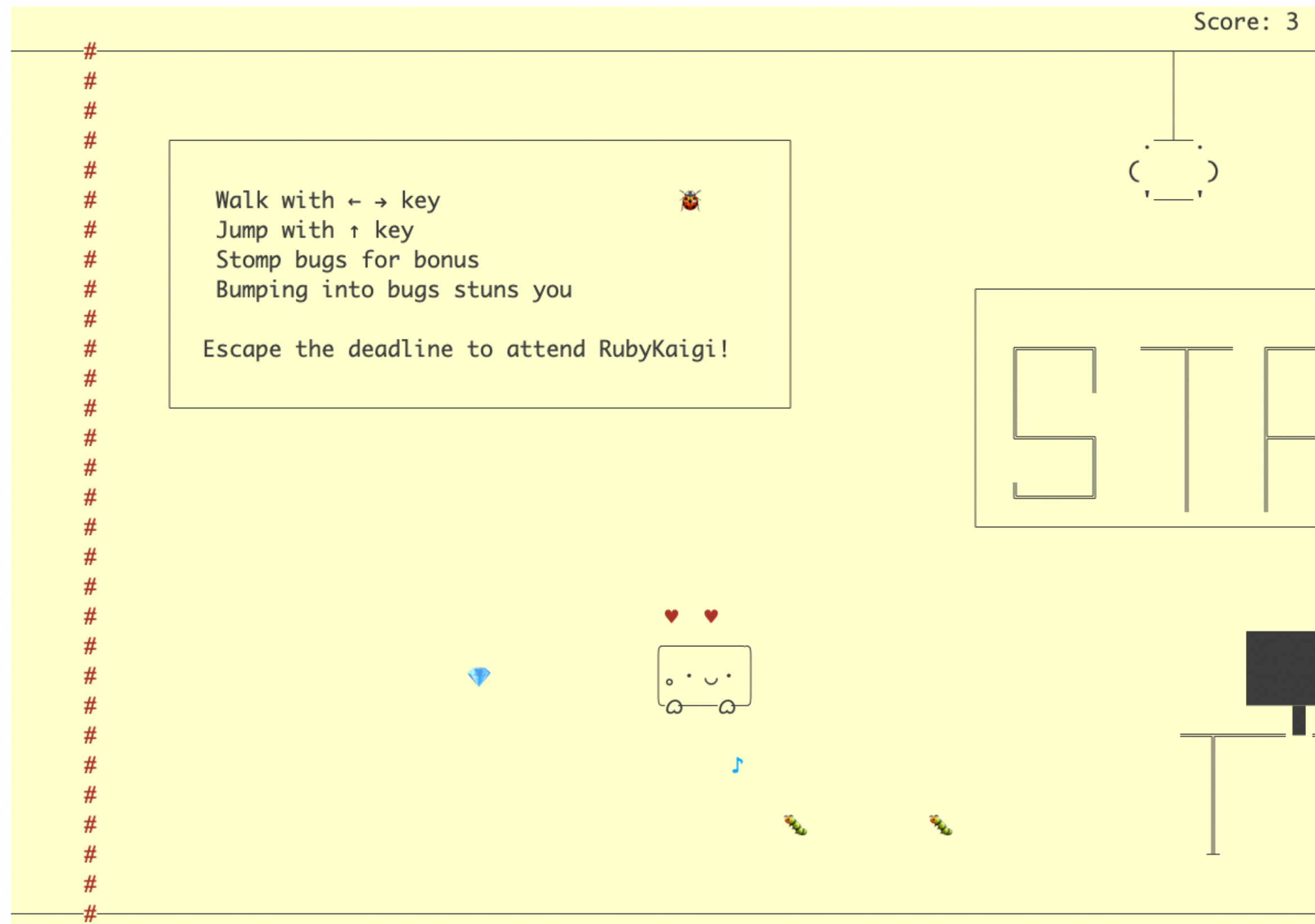
- レイヤー合成
- 物理シミュレーション

# 端末ゲームプログラミング実践編

- レイヤー合成 
- 物理シミュレーション

# レイヤー合成

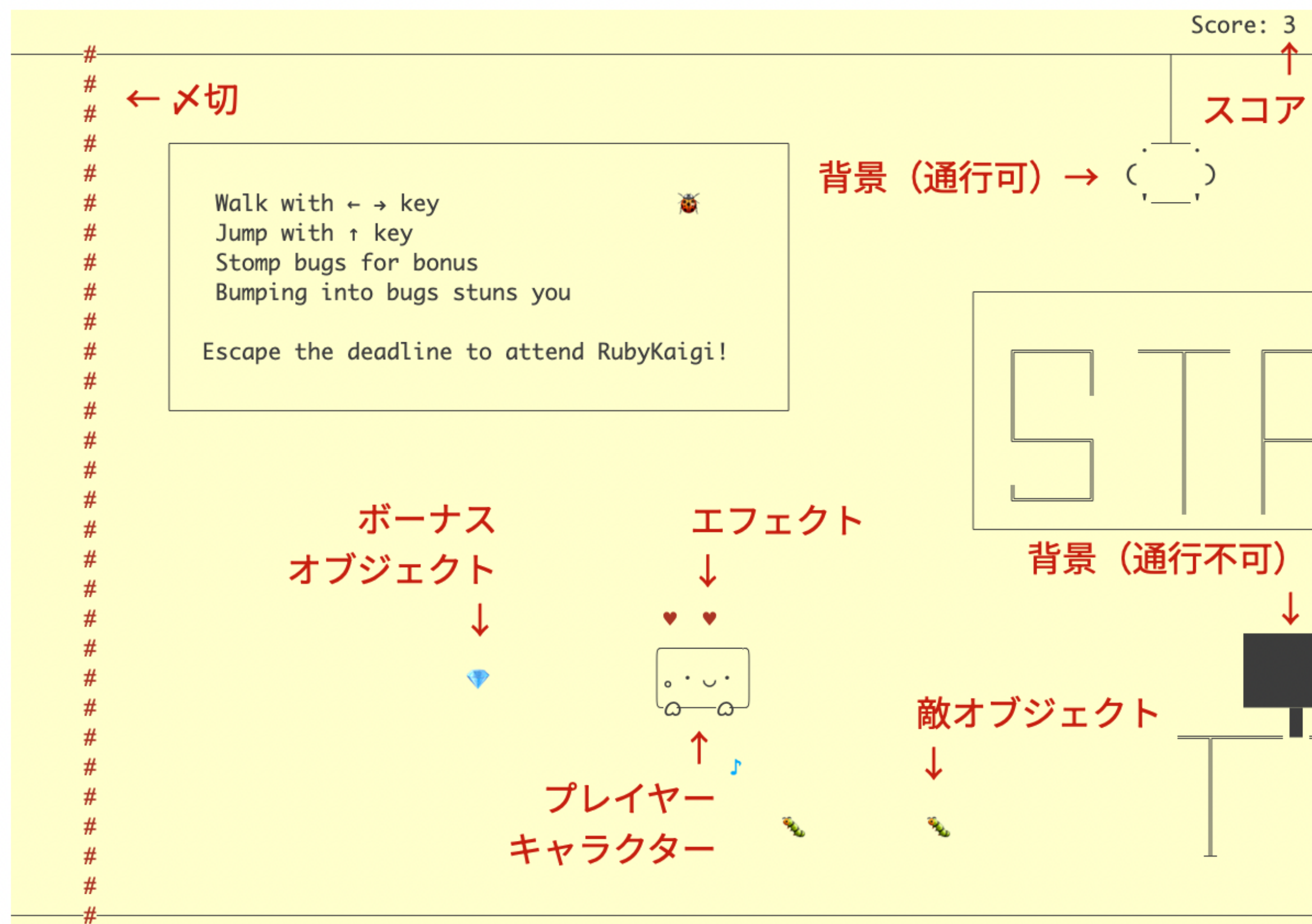
- オブジェクトの種類



# レイヤー合成

- オブジェクトの種類

- 8 種類



整理重要!!



# レイヤー合成

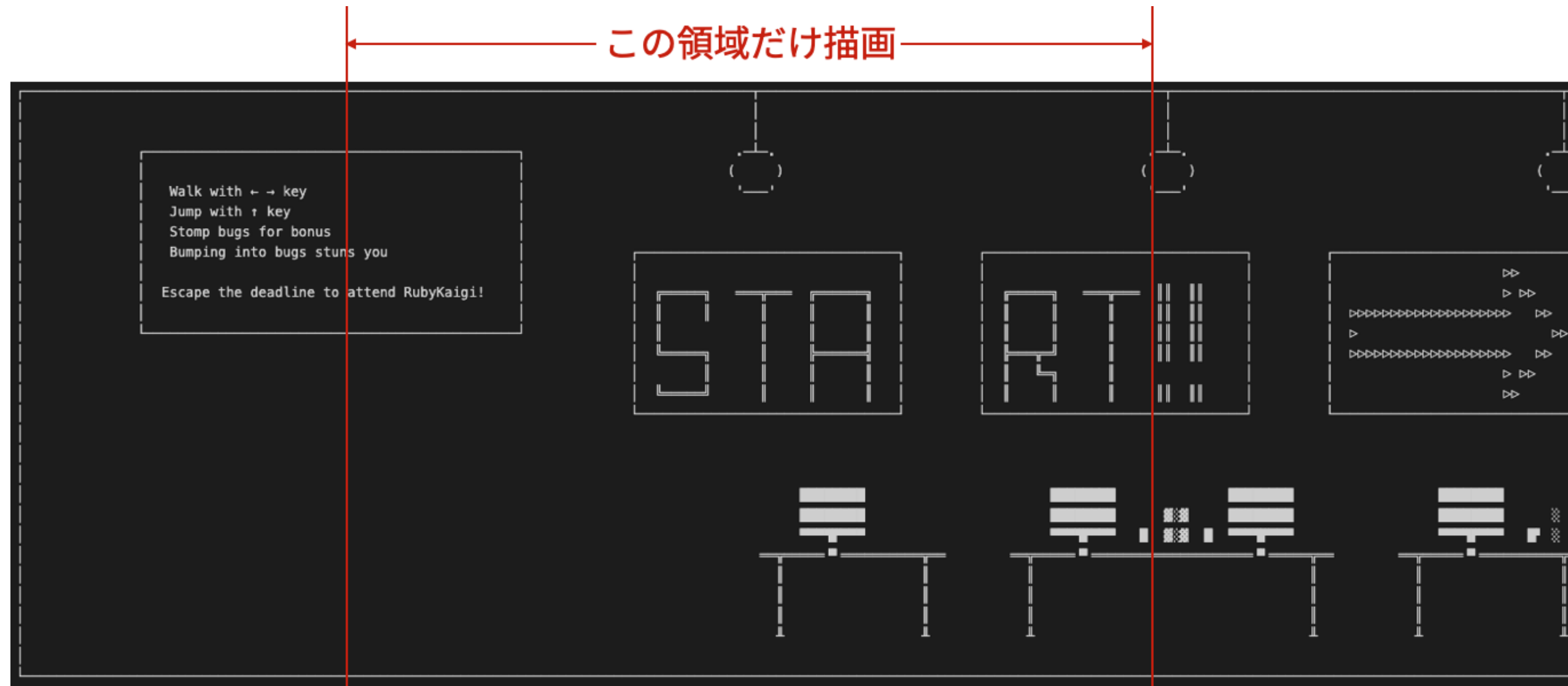
- オブジェクトをレイヤーとクラスで管理する
  - 背景レイヤー
    - 背景(アスキーアート)
    - マスク(通行可否情報)
  - 前景レイヤー
    - プレイヤー(ひとつ)
    - 敵オブジェクト(複数個) etc
  - その他
    - スコア

# レイヤー合成

- オブジェクトをレイヤーとクラスで管理する
- 描画時にマージする

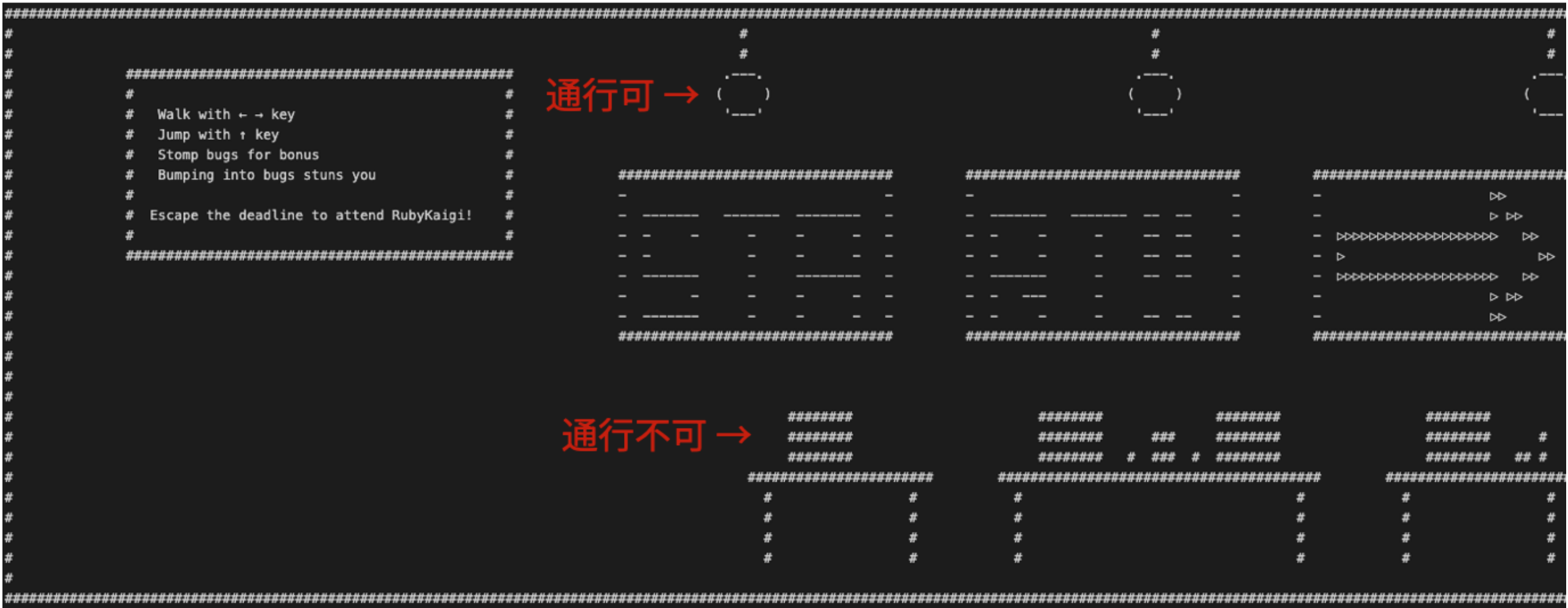
# 背景レイヤー

- 背景(アスキーアート)
  - アスキーアートを読み込み、描画領域分(100 x 30)を描画
  - 1文字を1つの「Tile」クラスのオブジェクトにする
    - 100 x 30 の2次元配列で管理



# 背景レイヤー

- マスク(通行可否情報)
  - 背景のアスキーアートからマスクデータを作成
  - 通行不可にしたい箇所をマスク文字(#)に置換
  - 該当座標がマスク文字(#)の Tile は通行不可オブジェクト





# 前景レイヤー

# 前景レイヤー

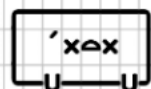
- プレイヤーキャラクター
- ボーナスオブジェクト
- 敵オブジェクト
- エフェクト
- ✕切



# 前景レイヤー

- 各クラス 描画領域分(100 x 30)の 2 次元配列を確保
  - 座標位置の配列に 1 文字ずつ格納

100 x 30  
プレイヤー用配列



100 x 30  
エフェクト用配列



# 前景レイヤー

- 「1 文字」
  - 1 文字として扱いたい単位

# 前景レイヤー

- 「1 文字」
    - 1 文字として扱いたい単位
    - "💎" は 2 文字として扱う
      - 半角文字 2 文字分の描画領域を使用するので
      - 配列の次の要素に NULL 文字 (制御文字) を入れて 2 文字分の領域を確保する
- ["💎", "\0"]


# 前景レイヤー

- 「1 文字」
  - 1 文字として扱いたい単位
  - "\e[33m" + "𐀀" + "\e[38;5;238m" は 1 文字として扱う
    - ANSI エスケープシーケンス自身は描画されないので

# レイヤー合成

- 背景レイヤと前景レイヤの各クラスの配列をマージ
  - 背景 → プレイヤー → 敵 → エフェクト ... のように後ろから順にマージ
- マージ済配列を描画する

# 端末ゲームプログラミング実践編

- レイヤー合成
- 物理シミュレーション 



# 物理シミュレーション

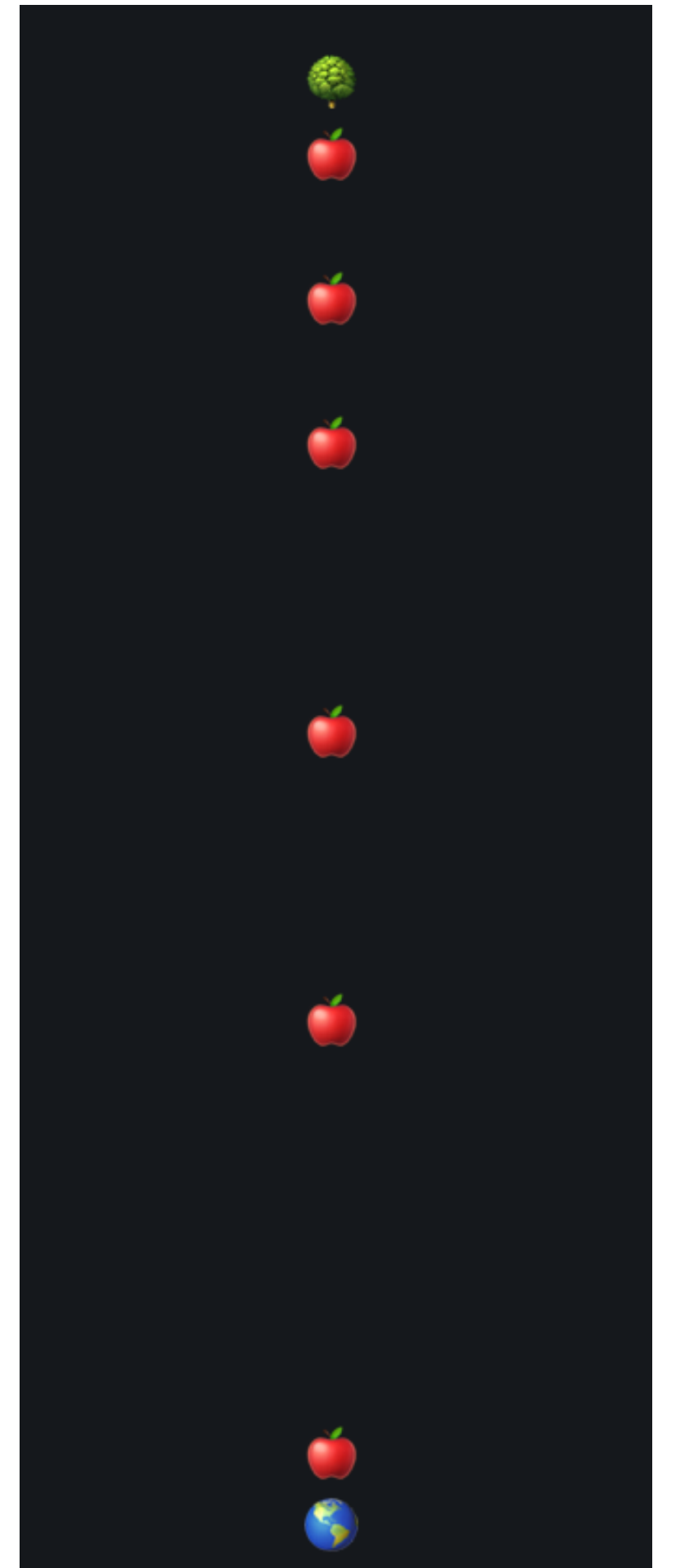
- とは
- 物理法則を計算することで現象を数値的に再現すること

# 物理シミュレーション

- e.g. 重力加速度に従って落ちるりんごのシミュレーション
  - 速度(m/s) = 速度(m/s) + 重力加速度(m/s<sup>2</sup>) \* 時間(s)
  - 移動距離(m) = 移動距離(m) + 速度(m/s) \* 時間(s)

```
height, goal_height = 1, 15
velocity = 0.0 # 速度 (m/s) 初速は0
step_second = 0.3 # ステップ時間 (s)
gravity = 9.8 # 重力加速度 (m/s2)

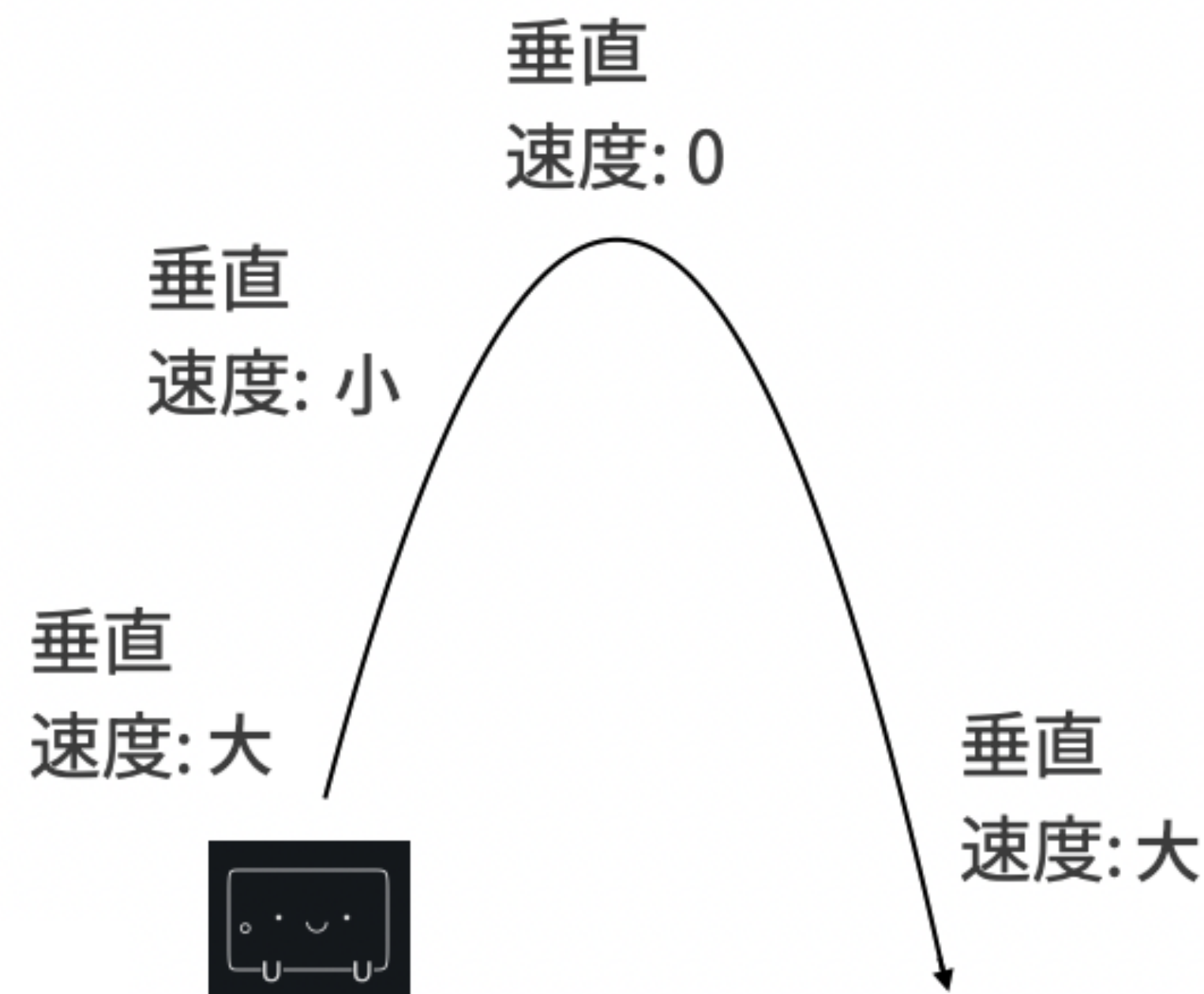
puts "\e[2J" + "\e[#{height};10H" + " 🌳"
while height < goal_height
  velocity += gravity * step_second
  height += velocity * step_second
  puts "\e[#{height.round.to_i};10H" + " 🍏"
  sleep step_second
end
puts "\e[#{height.round.to_i+1};10H" + " 🌍"
```



# ジャンプシミュレーション

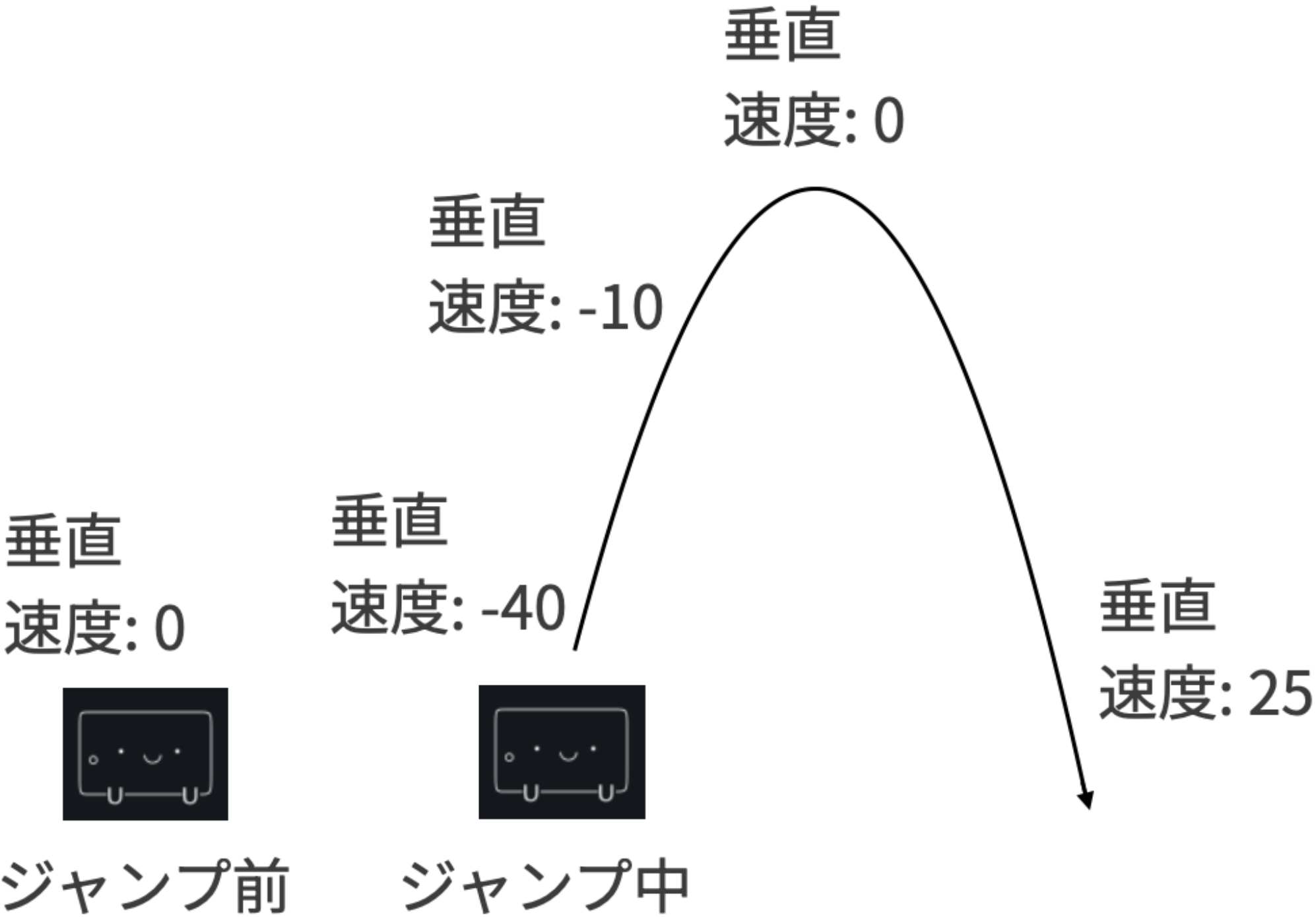
# ジャンプシミュレーション

- 上昇時
  - 初速が最も大きい
  - 高度が高くなるほど垂直速度が下がる
  - 垂直速度が 0 になると下降に切り替わる
- 下降時
  - 初速 0
  - 高度が低くなるほど垂直速度が上がる



※ 今回は垂直速度ではなく垂直速度の絶対値が大小します

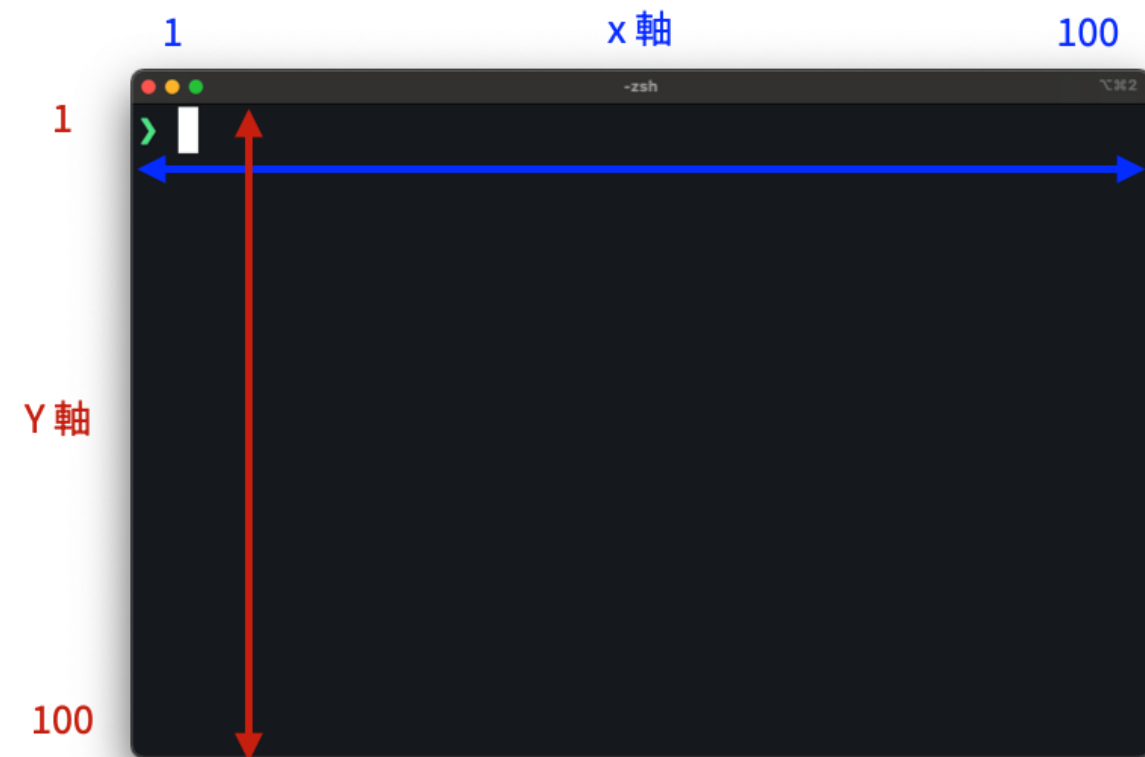
# ジャンプシミュレーション



垂直速度のみ持つと真上にジャンプする  
垂直速度と水平速度を持つと放物線状にジャンプする

# ジャンプシミュレーション

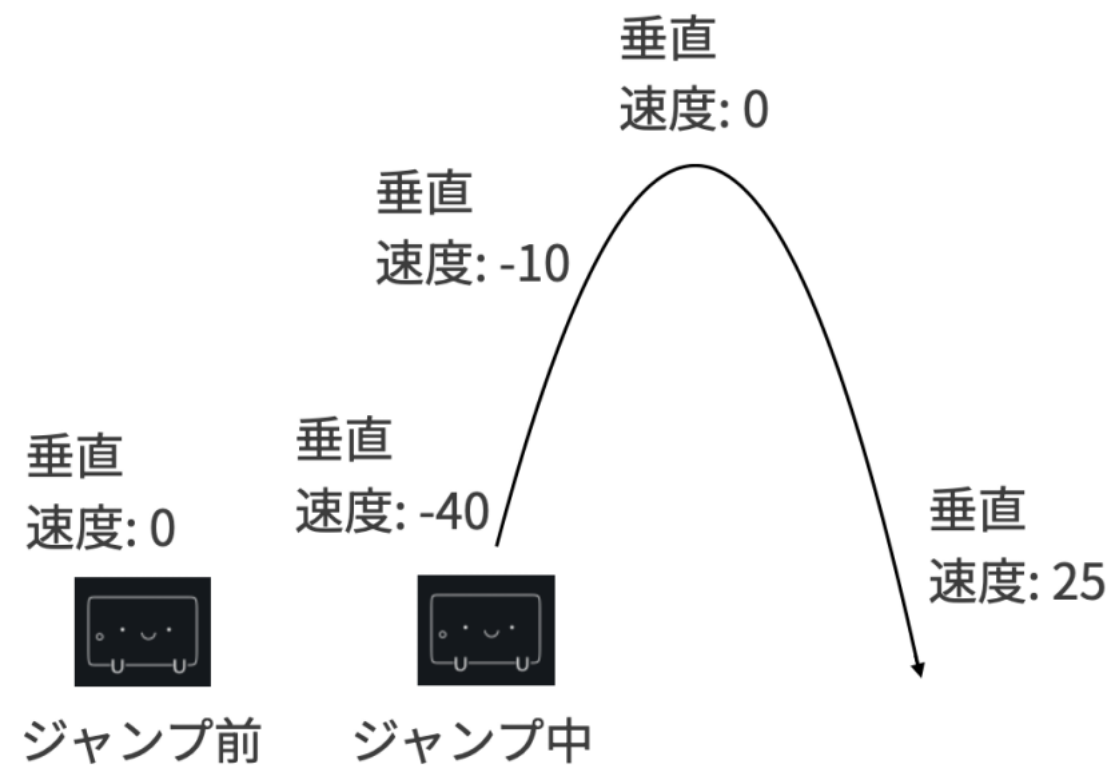
- ジャンプキーの入力を受付けると垂直速度をセット(初速)
- 初速は上向きなのでマイナス
  - 端末座標の y 軸は上が小さく下が大きい



```
# ジャンプ開始処理
# JUMP_INITIAL_VELOCITY: 垂直速度初速 (-40)
# @y_velocity: 垂直速度
def jump
  unless jumping?
    @jumping = true
    @y_velocity = JUMP_INITIAL_VELOCITY
  end
end
```

# ジャンプシミュレーション

- 垂直速度(コマ/秒) = 垂直速度 + (重力加速度(コマ/秒<sup>2</sup>) \* 経過時間(秒))
- 垂直位置(コマ) = 垂直位置 + (垂直速度(コマ/秒) \* 経過時間(秒))



```
# 更新処理
# JUMP_GRAVITY: 重力加速度 (80)
# @y_velocity: 垂直速度 (初期値は -40)
def update
  # ...
  @y_velocity += JUMP_GRAVITY * elapsed_time
  @y += @y_velocity * elapsed_time
```

# 物理シミュレーション

- 物理シミュレーションをすると表現の幅が増える



# Road to RubyKaigi のつくり方

- アニメーション
- 入力処理
- ゲームループ
- レイヤー合成
- 物理シミュレーション

# ゲームづくり

- すぐ動く、すぐたのしい
- Ruby すごい
  - 標準ライブラリが豊富
  - 文字列操作が得意
- 数学・物理が苦手でもなんとかなる

# Special Thanks

@youchan

**Enjoy Creating!**