

# Pyxel で、プログラミングを遊ぼう！

「楽しく作る」をデザインする

Let's Play Programming with Pyxel! — Designing the Fun of Making



北尾 崇 Takashi Kitao

✕ @kitao    ✉ takashi.kitao@gmail.com



今日は、Pyxel の話をします

Today I'd Like to Talk About Pyxel



INTRODUCTION

# 奇妙なアルバイト

A Strange Job Offer



# こんなアルバイト、いかがですか？

How About a Part-Time Job Like This?



アルバイト募集

PART-TIME POSITION AVAILABLE

|                |                                                                                     |
|----------------|-------------------------------------------------------------------------------------|
| 業務内容<br>Duties | 画面を見ながら、レバーを右に倒し続けてください<br>Watch the screen and keep pushing the lever to the right |
|                | 図形が表示されたら、ボタンを1回押してください<br>When a shape appears, press the button once              |
|                | 押し遅れると、最初からやり直します<br>If you press too late, you start over from the beginning       |
|                | その作業を、延々と繰り返してください<br>Repeat that, over and over, without end                       |
| 勤務時間<br>Hours  | 30時間<br>30 hours                                                                    |
| 報酬<br>Pay      | 4,900円<br>4,900 yen                                                                 |



# こんなアルバイト、いかがですか？

How About a Part-Time Job Like This?



## アルバイト募集

PART-TIME POSITION AVAILABLE

|                |                                                                                     |
|----------------|-------------------------------------------------------------------------------------|
| 業務内容<br>Duties | 画面を見ながら、レバーを右に倒し続けてください<br>Watch the screen and keep pushing the lever to the right |
|                | 図形が表示されたら、ボタンを1回押してください<br>When a shape appears, press the button once              |
|                | 押し遅れると、最初からやり直します<br>If you press too late, you start over from the beginning       |
|                | その作業を、延々と繰り返してください<br>Repeat that, over and over, without end                       |
| 勤務時間<br>Hours  | 30時間<br>30 hours                                                                    |
| 報酬<br>Pay      | <b>4,900円</b> ← 受け取る額ではなく、支払う額<br>4,900 yen Not what you earn — what you pay        |



# そのアルバイト、体験済みかも？

Have You Already Worked This Job?



| アルバイト 募集                     |                                                                                     |
|------------------------------|-------------------------------------------------------------------------------------|
| PART-TIME POSITION AVAILABLE |                                                                                     |
| 業務内容<br>Duties               | 画面を見ながら、レバーを右に倒し続けてください<br>Watch the screen and keep pushing the lever to the right |
|                              | 図形が表示されたら、ボタンを1回押してください<br>When a shape appears, press the button once              |
|                              | 押し遅れると、最初からやり直しです<br>If you press too late, you start over from the beginning       |
|                              | その作業を、延々と繰り返してください<br>Repeat that, over and over, without end                       |
| 勤務時間<br>Hours                | 30時間<br>30 hours                                                                    |
| 報酬<br>Pay                    | 4,900円 ← 受け取る額ではなく、支払う額<br>4,900 yen Not what you earn — what you pay               |



Super Mario Bros. © 1985 Nintendo

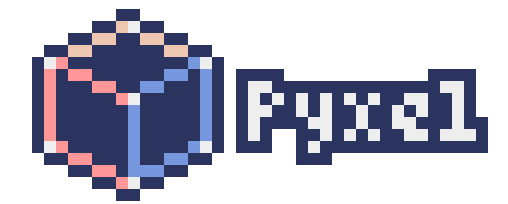
「作業」と「遊び」は紙一重。違いは、楽しくする工夫があるかどうか

Work and play are a hair apart. The difference is whether someone designed the fun



# ゲームデザインとは

What Is Game Design?



## プレイヤーに「楽しい」「面白い」と感じさせる仕組み

The workings that make players find it fun and interesting

### 目指す体験： The experience to aim for



やりたいことができる

You can do what you want



達成感が得られる

You get a sense of achievement

### 実現手段： How it is done



世界の構築

Building a world

計算

Math

ルール

Rules

物語

Story



フィードバック

Feedback

映像

Visuals

音

Sound

言葉

Words



試行錯誤の後押し

Encouraging trial and error

目標

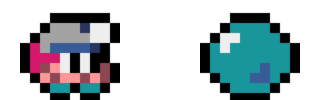
Objectives

ヒント

Hints

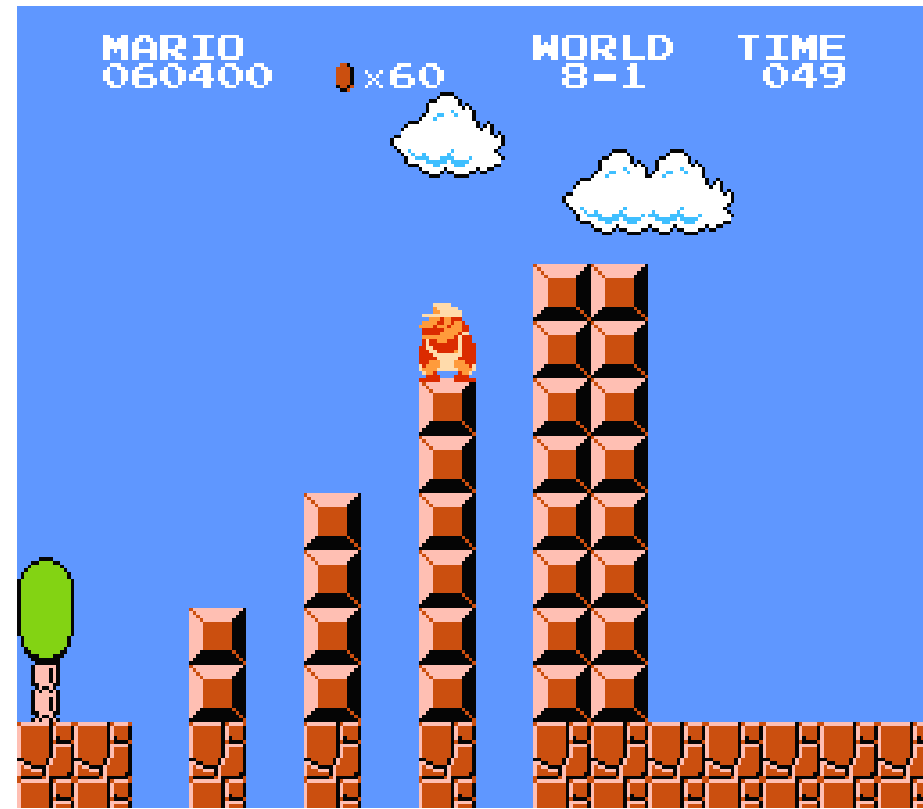
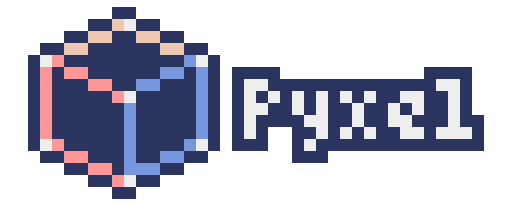
補正

Assists

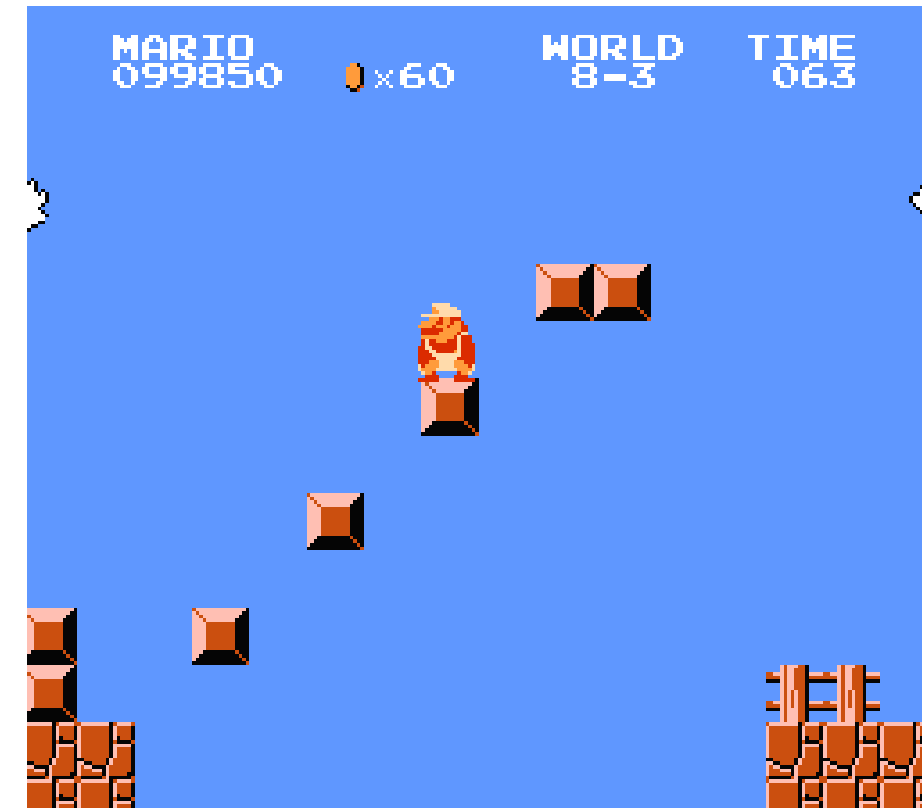


# 「達成感」は錯覚でも良い

An Illusion of Achievement Is Fine



8-1



8-3

Super Mario Bros. © 1985 Nintendo

難易度は同じ。見え方が違うだけ

The difficulty is the same. Only the look differs



# 自己紹介

About Me



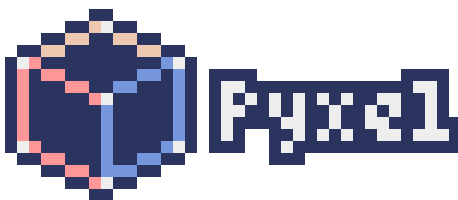
北尾 崇    きたお たかし    Takashi Kitao

|                          |                                                                                                                                                                                                                      |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 過去<br>Past               | 元ゲーム開発者<br>Former game developer<br><br>『METAL GEAR SOLID』 企画 / ムービー制作<br>Planner / Cutscene Production<br><br>『ZONE OF THE ENDERS』 シリーズ メインプログラマ / ゲームデザインユニットディレクター<br>Lead Programmer / Game Design Unit Director |
| 現在<br>Now                | ソニー株式会社 技術開発研究所 XR システム技術開発部 統括部長<br>General Manager, XR Systems Technology Development Dept., Technology Development Laboratories, Sony<br><br>XR（拡張現実・仮想現実・複合現実）と AI の研究開発<br>R&D in XR (AR/VR/MR) and AI          |
| 個人開発<br>Personal project | 2018年からレトロゲームエンジン Pyxel を開発<br>Developing the retro game engine Pyxel since 2018                                                                                                                                     |
| 著書<br>Book               | 『ゲームで学ぶ Python！ Pyxel ではじめるレトロゲームプログラミング』<br>Learn Python Through Games: Retro Game Programming with Pyxel<br><br>2025年1月 技術評論社刊<br>January 2025, published by Gijutsu-Hyoron Co., Ltd.                              |



# 自己紹介

About Me



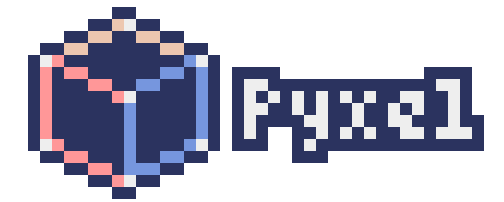
北尾 崇    きたお たかし    Takashi Kitao

|                          |                                                                                                                                                                                                                      |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 過去<br>Past               | 元ゲーム開発者<br>Former game developer<br><br>『METAL GEAR SOLID』 企画 / ムービー制作<br>Planner / Cutscene Production<br><br>『ZONE OF THE ENDERS』 シリーズ メインプログラマ / ゲームデザインユニットディレクター<br>Lead Programmer / Game Design Unit Director |
| 現在<br>Now                | ソニー株式会社 技術開発研究所 XR システム技術開発部 統括部長<br>General Manager, XR Systems Technology Development Dept., Technology Development Laboratories, Sony<br><br>XR（拡張現実・仮想現実・複合現実）と AI の研究開発<br>R&D in XR (AR/VR/MR) and AI          |
| 個人開発<br>Personal project | 2018年からレトロゲームエンジン Pyxel を開発<br>Developing the retro game engine Pyxel since 2018                                                                                                                                     |
| 著書<br>Book               | 『ゲームで学ぶ Python！ Pyxel ではじめるレトロゲームプログラミング』<br>Learn Python Through Games: Retro Game Programming with Pyxel<br><br>2025年1月 技術評論社刊<br>January 2025, published by Gijutsu-Hyoron Co., Ltd.                              |



# レトロゲームエンジン Pyxel（ピクセル）

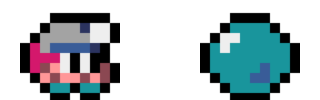
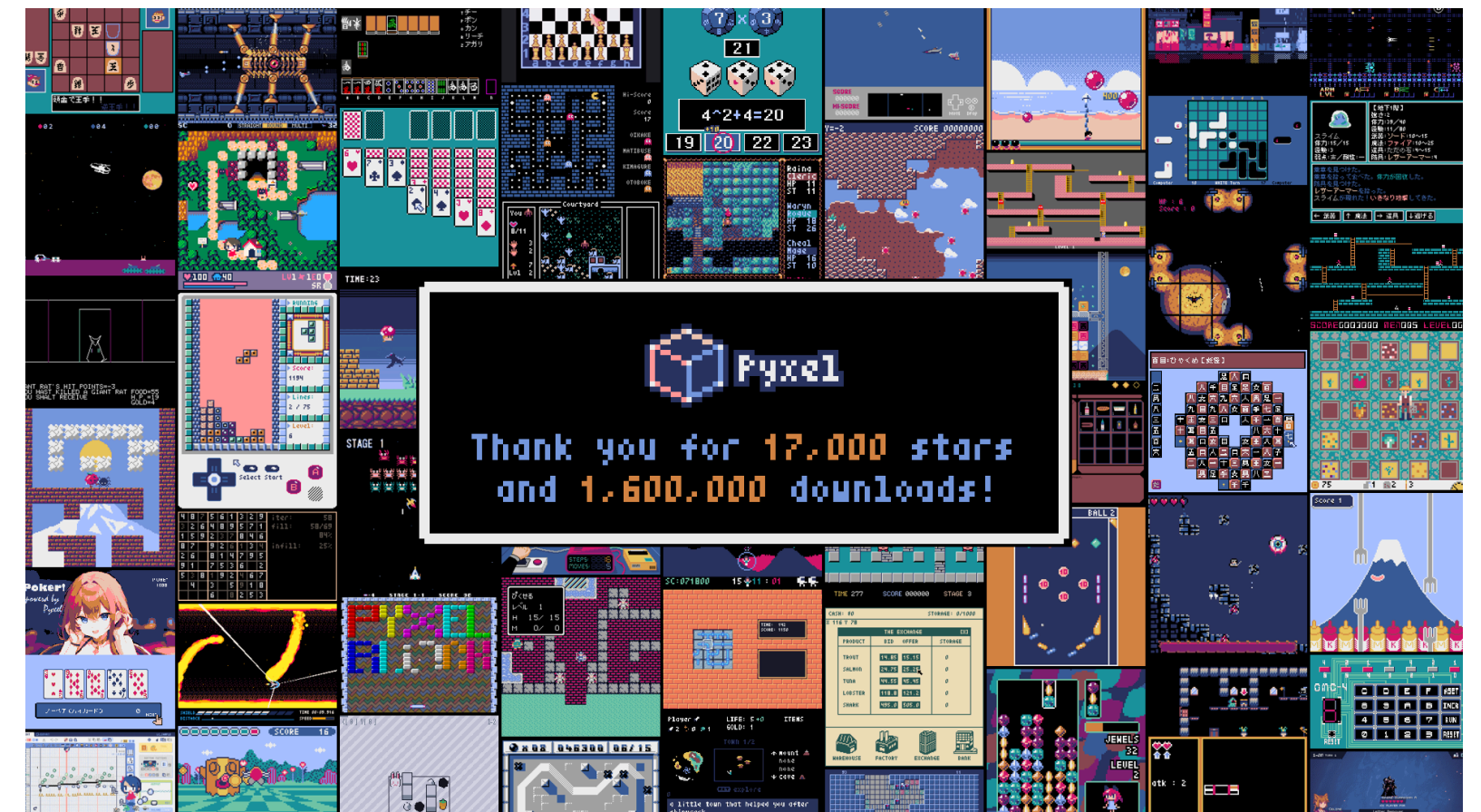
Pyxel (/ˈpɪksəl/), a Retro Game Engine



## レトロゲーム風のゲームを簡単に作るための、Python モジュール & ツール

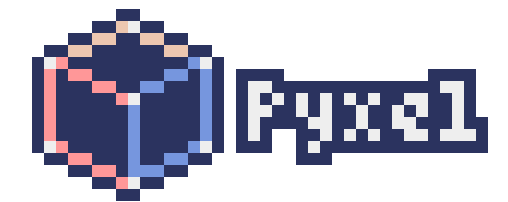
A Python module and tools for easily making retro-style games

|                        |                                                                                                                                                                             |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| コンセプト<br>Concept       | 気軽に楽しくプログラミング<br>Easy and fun programming                                                                                                                                   |
| 動作環境<br>Platforms      | Windows・Mac・Linux・Web ブラウザ<br>Windows, Mac, Linux, and web browsers<br><br>Chrome OS、Raspberry Pi、中華ゲーム機でも動作<br>Also runs on Chrome OS, Raspberry Pi, and Chinese handhelds |
| 同梱ツール<br>Bundled tools | イメージエディタ／サウンドエディタ<br>Image editor and sound editor                                                                                                                          |
| ライセンス<br>License       | MIT ライセンス<br>MIT License<br><br>商用利用も含め、無料で自由に使える<br>Free to use, including commercially                                                                                    |
| 採用実績<br>Adoption       | 世界中の教育機関やプログラミング教室で採用<br>Schools, coding classes, and more, worldwide<br><br>慶應義塾大学、大阪大学、Tento など<br>Keio University, Osaka University, Tento, and more                     |
| 公式サイト<br>Website       | <a href="https://github.com/kिताo/pyxel">https://github.com/kिताo/pyxel</a><br>Documentation, examples, and source                                                          |



# 現代版「レトロゲーム」

Retro Games Made Today



Pyxel 書籍付属サンプル『Cursed Caverns』

Cursed Caverns, a sample game from the Pyxel book

Music © Maki Kirioka



Pyxel 製の商用タイトル『Dungeon Antiqua』

Dungeon Antiqua, a commercial title made with Pyxel

© Shiromofu Factory



# Pyxel で、プログラミングを遊ぼう！ —— 「楽しく作る」をデザインする

## Let's Play Programming with Pyxel! — Designing the Fun of Making



本日は、Pyxel がどのようにプログラミングを「遊び」に変えているのか、その仕組みや考え方、そこに込めた想いについてお話しします

How Pyxel turns programming into play — the mechanisms, the thinking, and the wish behind them



### CHAPTER 1

アイデアを形にできると、うれしい

Bringing Your Ideas to Life Feels Great



### CHAPTER 2

みんなと作るのって、面白い

Making Things Together Is Fun



### CHAPTER 3

これからの Pyxel は、もっと楽しい！

Pyxel's Future Is Even More Exciting!





## CHAPTER 1

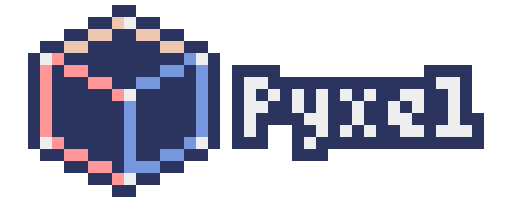
# アイデアを形にできると、うれしい

Bringing Your Ideas to Life Feels Great



# Pyxel を作った理由

Why I Made Pyxel



## 自分のために For myself

好きなものを作りたかった  
I wanted to build something I loved



やるならやはり好きなゲーム、  
題材は思春期を過ごしたレトロゲーム  
Then it had to be games —  
the retro ones I grew up with



レトロゲームを題材にしたゲームエンジンを自分で作ることにした  
I decided to build a retro-game engine of my own

## 息子のために For my son

手軽なプログラミング環境が欲しかった  
I wanted an easy programming environment



既存のゲームエンジンは複雑すぎ、  
レトロゲーム向けのものは Lua が主流だった  
The existing engines were too complex,  
and the retro ones mostly used Lua



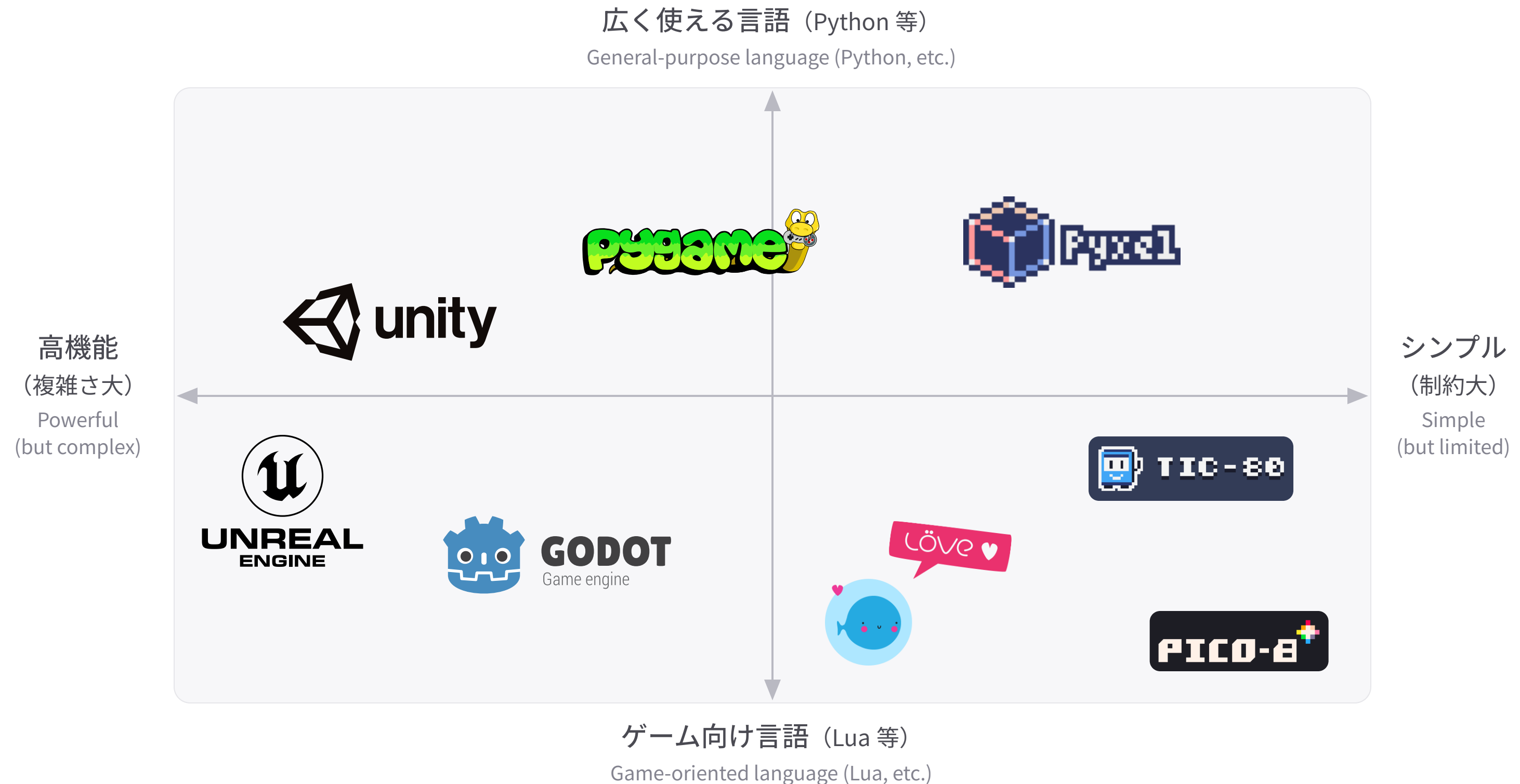
# ゲームエンジンの中の Pyxel の位置づけ

Where Pyxel Sits Among Game Engines



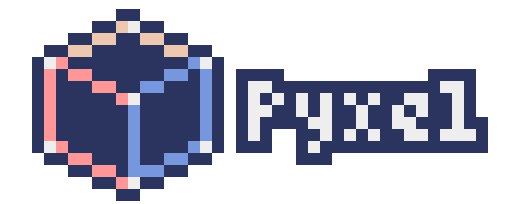
# ゲームエンジンの中の Pyxel の位置づけ

Where Pyxel Sits Among Game Engines



# 「これならできそう」と思わせる仕掛け

Making It Feel Doable

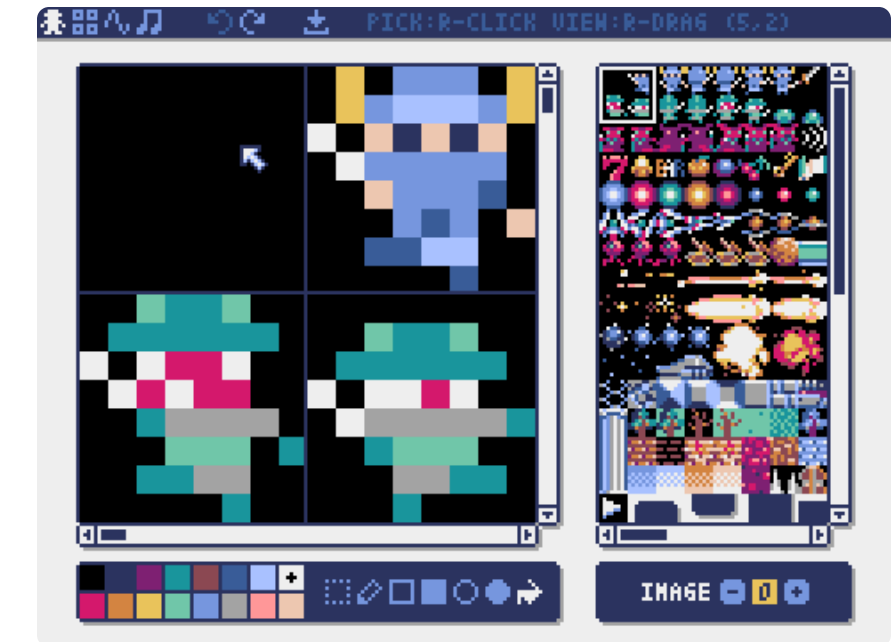


## 1 シンプルで学びやすい Python でプログラムを書ける

Write your program in Python — simple and easy to learn

Python は文法が素直で読みやすく、解説記事や入門書も多い

Plain syntax, readable code, and plenty of tutorials and books



## 2 気軽にインストールして、すぐに使い始められる

Install it without fuss and start using it right away

MIT ライセンスで無料。どの環境でも pip install だけで導入できる

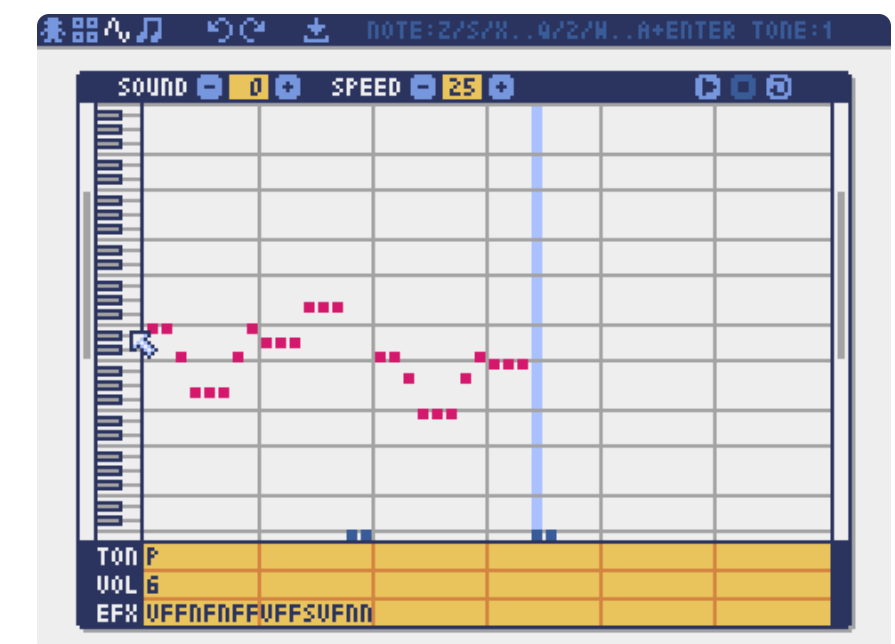
MIT-licensed and free; pip install alone works on any platform

## 3 必要ツールが同梱されており、絵や音が簡単に作れる

The tools come with it, so art and sound are easy to make

外部ツールの購入・習得が不要。簡単な操作でゲーム素材を作成できる

Nothing to buy or learn; simple controls make your game assets



## 4 あえて「レトロゲーム」らしい制約を入れている

And it deliberately keeps retro-game-style limits



# どちらも「すごい」。では「作れそう」なのは？

Both Are Impressive. So Which One Feels Doable?



3D モデリングツール  
3D modeling tool

Dragon phantom beast © Keita Okada / Villard Inc.



マイクラフト  
Minecraft

Mont-Saint-Michel © milk 猫 (@milk94698164)



# どちらも「すごい」。では「作れそう」なのは？

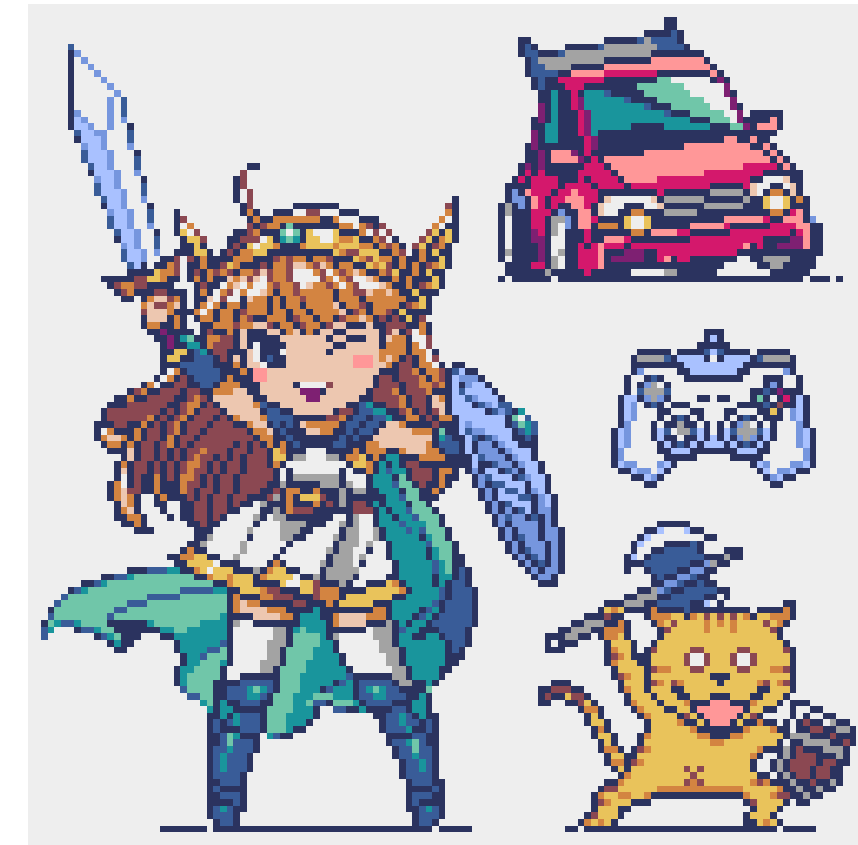
Both Are Impressive. So Which One Feels Doable?



フルカラーのキャラクターイラスト

A full-color character illustration

© Toshio Noguchi



ドット絵のキャラクターイラスト

A pixel-art character illustration

© Toshio Noguchi © @helpcomputer0



# Pyxel の「レトロゲーム」制約

Pyxel's Retro-Game Limits



|                          | 一般的なゲームエンジン<br>A typical game engine                       | Pyxel<br>A retro game engine                                                                                                   |
|--------------------------|------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| 色<br>Color               | フルカラー —— 約1,677万色<br>About 16.77 million colors            | 16色・半透明なし<br>16 colors, no translucency<br> |
| 音<br>Sound               | 同時発音数も音色も実質無制限<br>Effectively unlimited voices and timbres | 同時発音数は4音／音色は4種類<br>4 voices, 4 waveforms                                                                                       |
| 画像<br>Images             | サイズも枚数も自由<br>Any size, any number                          | 256×256サイズの画像が3枚<br>3 banks of 256×256                                                                                         |
| 描画命令<br>Drawing commands | 用途ごとに膨大な API<br>A vast API for every purpose               | 基本の描画命令は10種類<br>10 basic drawing commands                                                                                      |



# Pyxel の描画命令

## Pyxel's Drawing Commands



画面をクリアする  
Clear the screen

点を描画する  
Draw a pixel

四角形を描画する  
Draw a filled rectangle

円を描画する  
Draw a filled circle

画像を描画する  
Copy from an image bank

文字を描画する  
Draw a string

`cls(col)`

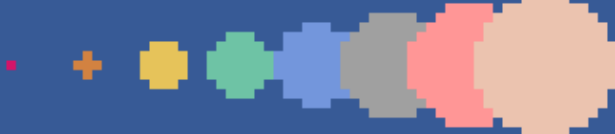
`pset(x, y, col)`



`rect(x, y, w, h, col)`



`circ(x, y, r, col)`



`blt(x, y, img, u, v,  
w, h, [colkey])`



`text(x, y, s, col)`

Elapsed frame count is 100  
Current mouse position is (0,0)

`line(x1, y1, x2, y2, col)`



`rectb(x, y, w, h, col)`



`circb(x, y, r, col)`



`bltm(x, y, tm, u, v,  
w, h, [colkey])`



`pal(col1, col2)`

直線を描画する  
Draw a line

四角形の枠線を描画する  
Draw a rectangle outline

円の枠線を描画する  
Draw a circle outline

タイルを並べて描画する  
Copy from a tilemap



# ゲームを作ってみよう① 図形を描く

Making a Game 1 — Drawing Shapes



描画命令で、キャラクターを描く

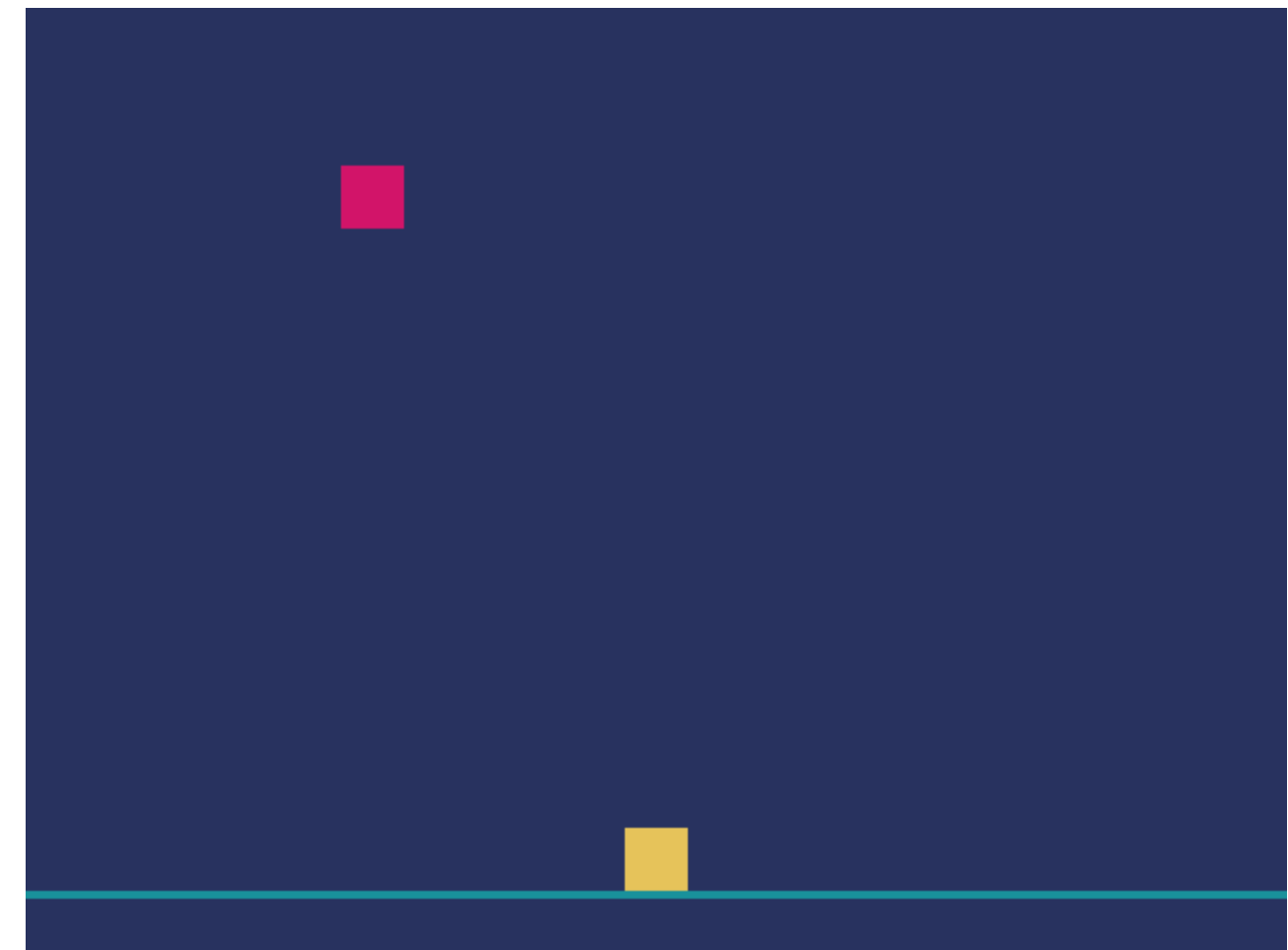
Draw the characters with drawing commands

```
import pyxel

pyxel.init(160, 120)

pyxel.cls(1)
pyxel.line(0, 112, 159, 112, 3)
pyxel.rect(76, 104, 8, 8, 10)
pyxel.rect(40, 20, 8, 8, 8)

pyxel.show()
```



# ゲームを作ってみよう② 図形を動かす

## Making a Game 2 — Moving the Shapes



座標を変えながら描画を繰り返すと、アニメーションになる

Repeat the drawing while shifting the coordinates, and it becomes animation

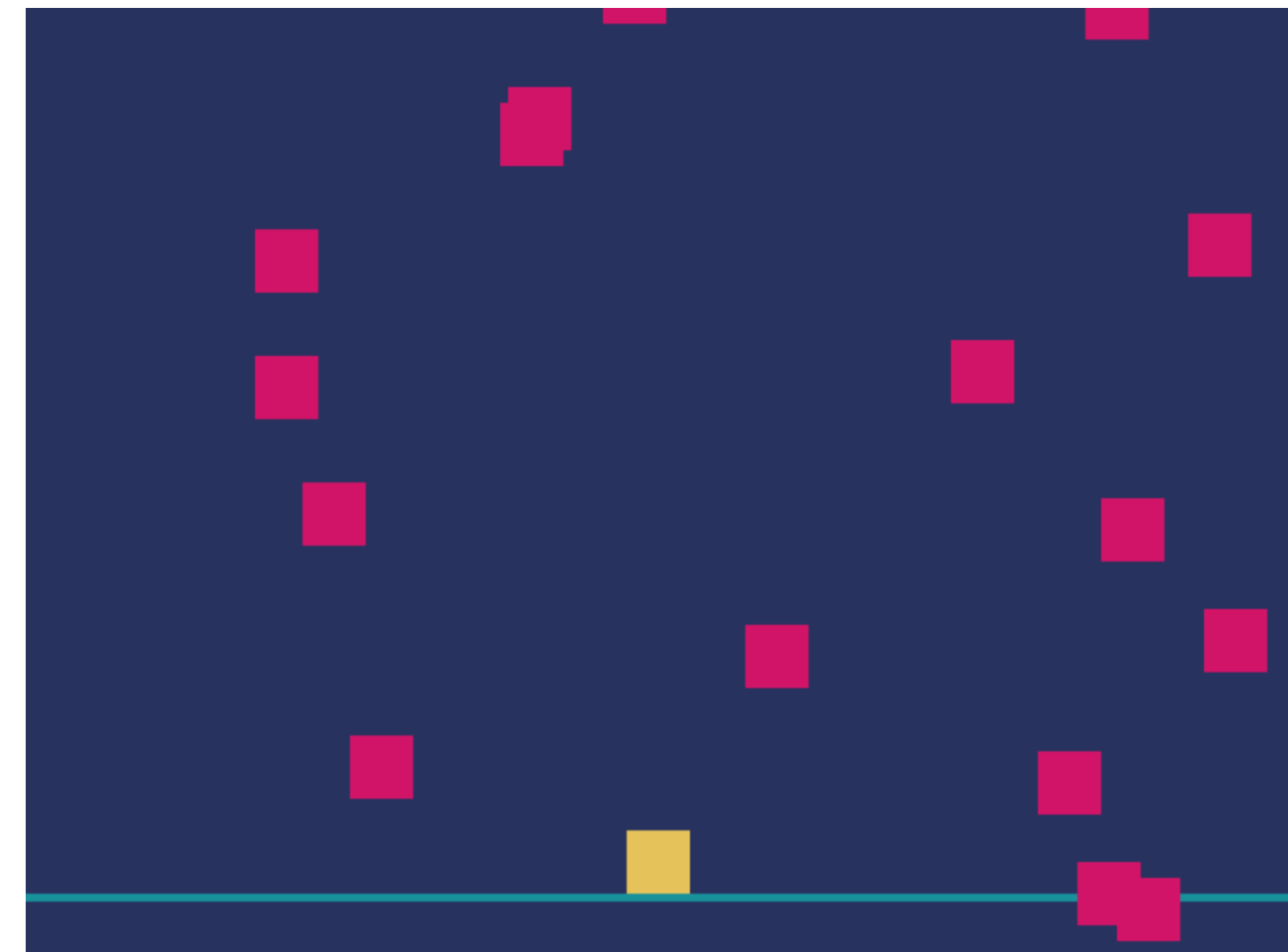
```
import pyxel

pyxel.init(160, 120)
enemies = [[pyxel.rndi(0, 152), -i * 16] for i in range(16)]

while True:
    pyxel.cls(1)
    pyxel.line(0, 112, 159, 112, 3)

    for enemy in enemies:
        enemy[1] += 2
        if enemy[1] > 120:
            enemy[0], enemy[1] = pyxel.rndi(0, 152), -8
        pyxel.rect(enemy[0], enemy[1], 8, 8, 8)

    pyxel.rect(76, 104, 8, 8, 10)
    pyxel.flip()
```



# ゲームを作ってみよう③ キーで操作する

## Making a Game 3 — Controlling It With Keys



### キー入力で、キャラクターを操作できるようにする

Take key input to make the character controllable

```
import pyxel

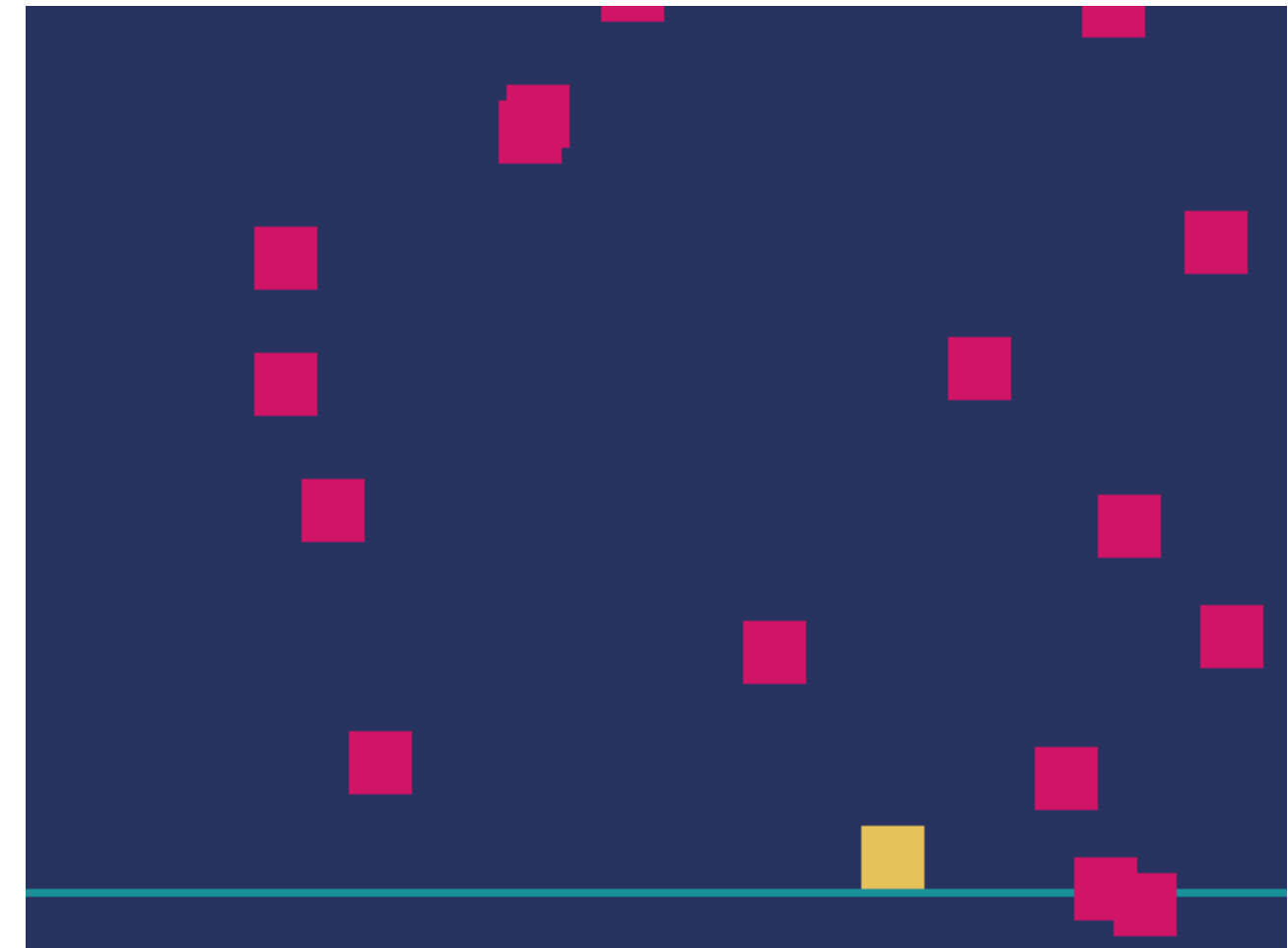
pyxel.init(160, 120)
x = 76
enemies = [[pyxel.rndi(0, 152), -i * 16] for i in range(16)]

while True:
    pyxel.cls(1)
    pyxel.line(0, 112, 159, 112, 3)

    if pyxel.btn(pyxel.KEY_LEFT):
        x -= 2
    if pyxel.btn(pyxel.KEY_RIGHT):
        x += 2

    for enemy in enemies:
        enemy[1] += 2
        if enemy[1] > 120:
            enemy[0], enemy[1] = pyxel.rndi(0, 152), -8
            pyxel.rect(enemy[0], enemy[1], 8, 8, 8)

    pyxel.rect(x, 104, 8, 8, 10)
    pyxel.flip()
```



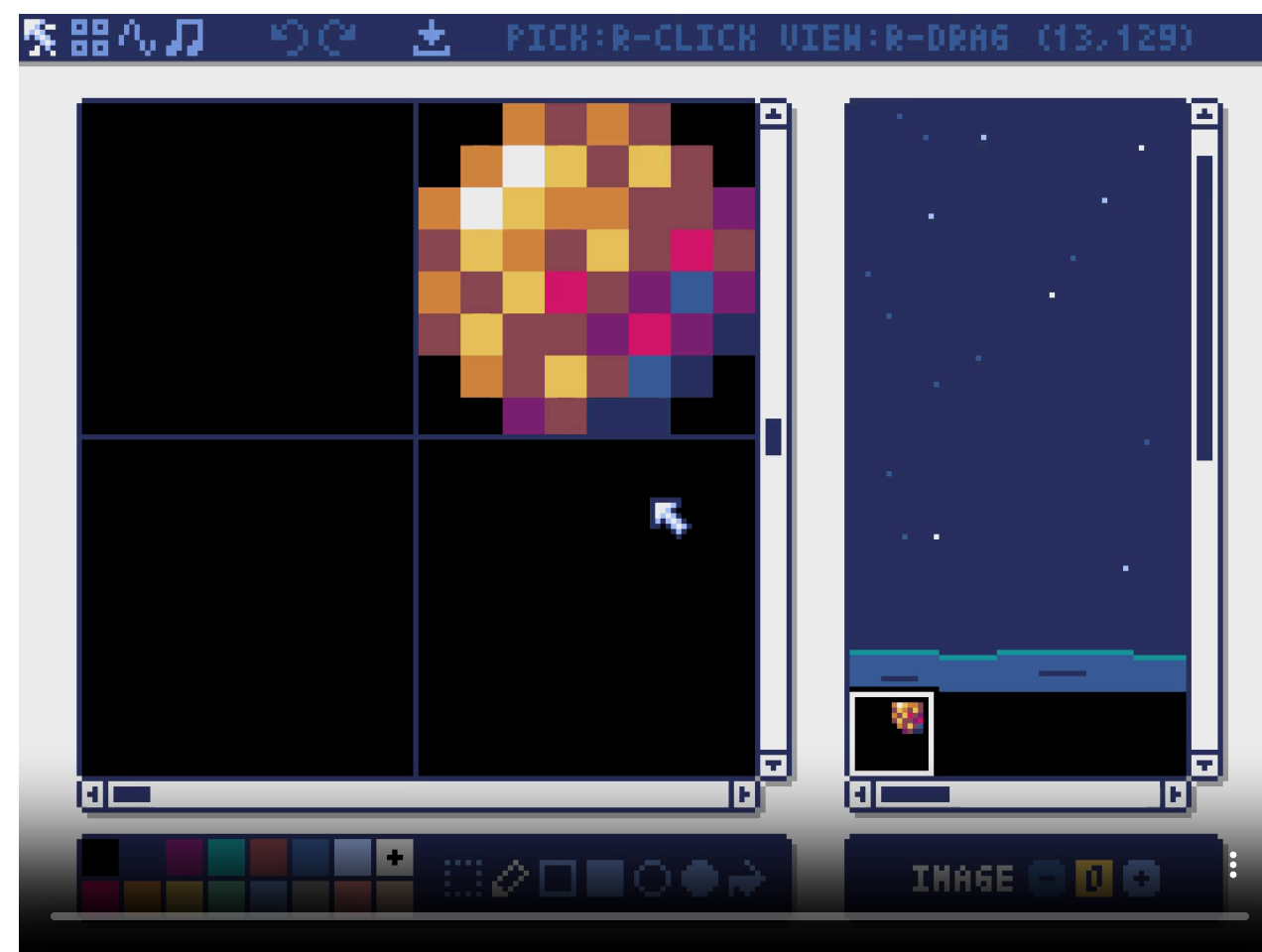
# ゲームを作ってみよう④ 絵と音を作る

Making a Game 4 — Making Art and Sound



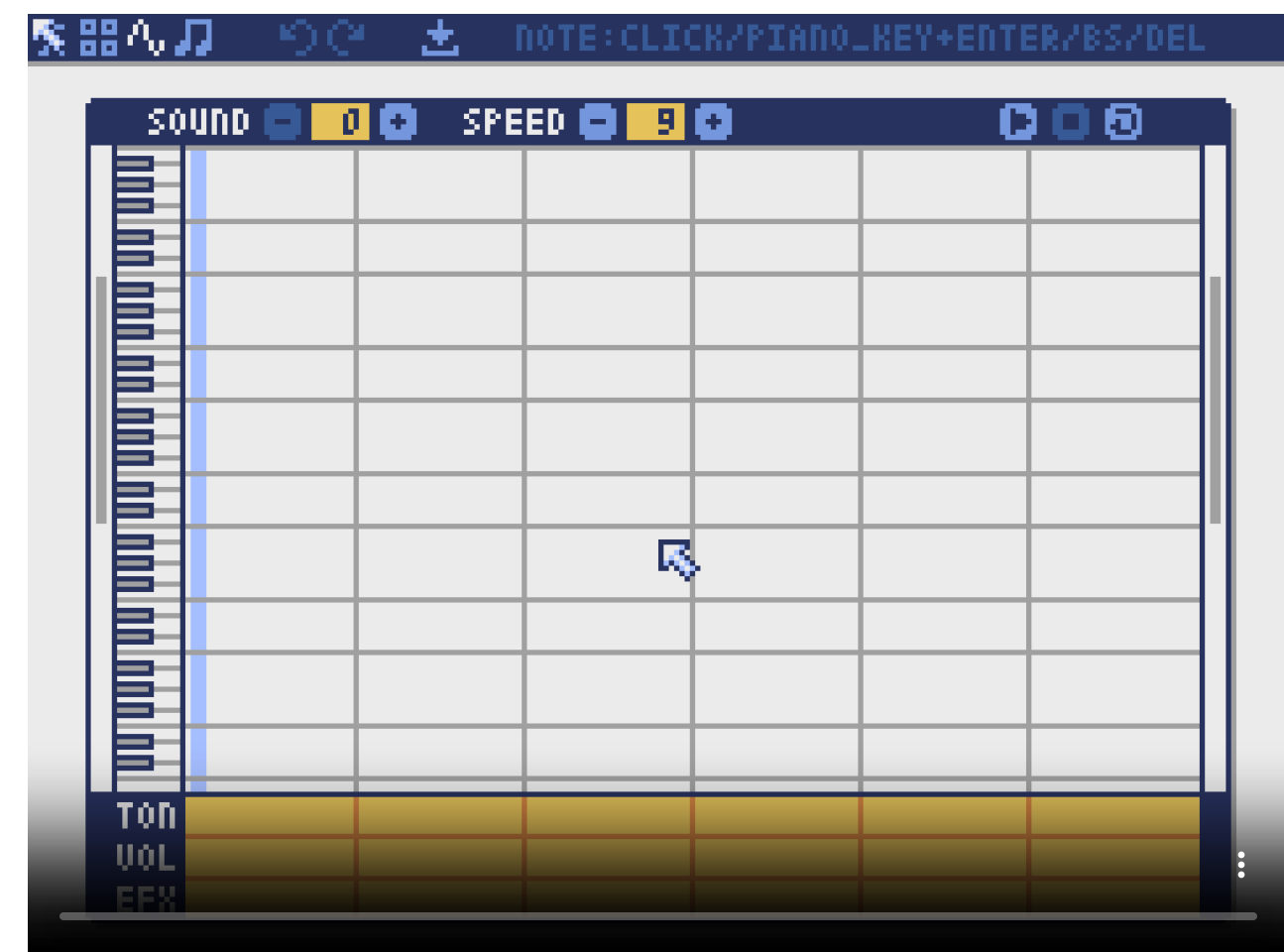
同梱の Pyxel Editor で、キャラクターと効果音を作る

Make the character and the sound effect in the bundled Pyxel Editor



イメージエディタ画面

The Image Editor screen



サウンドエディタ画面

The Sound Editor screen



# ゲームを作ってみよう⑤ 絵と音を入れる

Making a Game 5 — Adding the Art and Sound



```
import pyxel

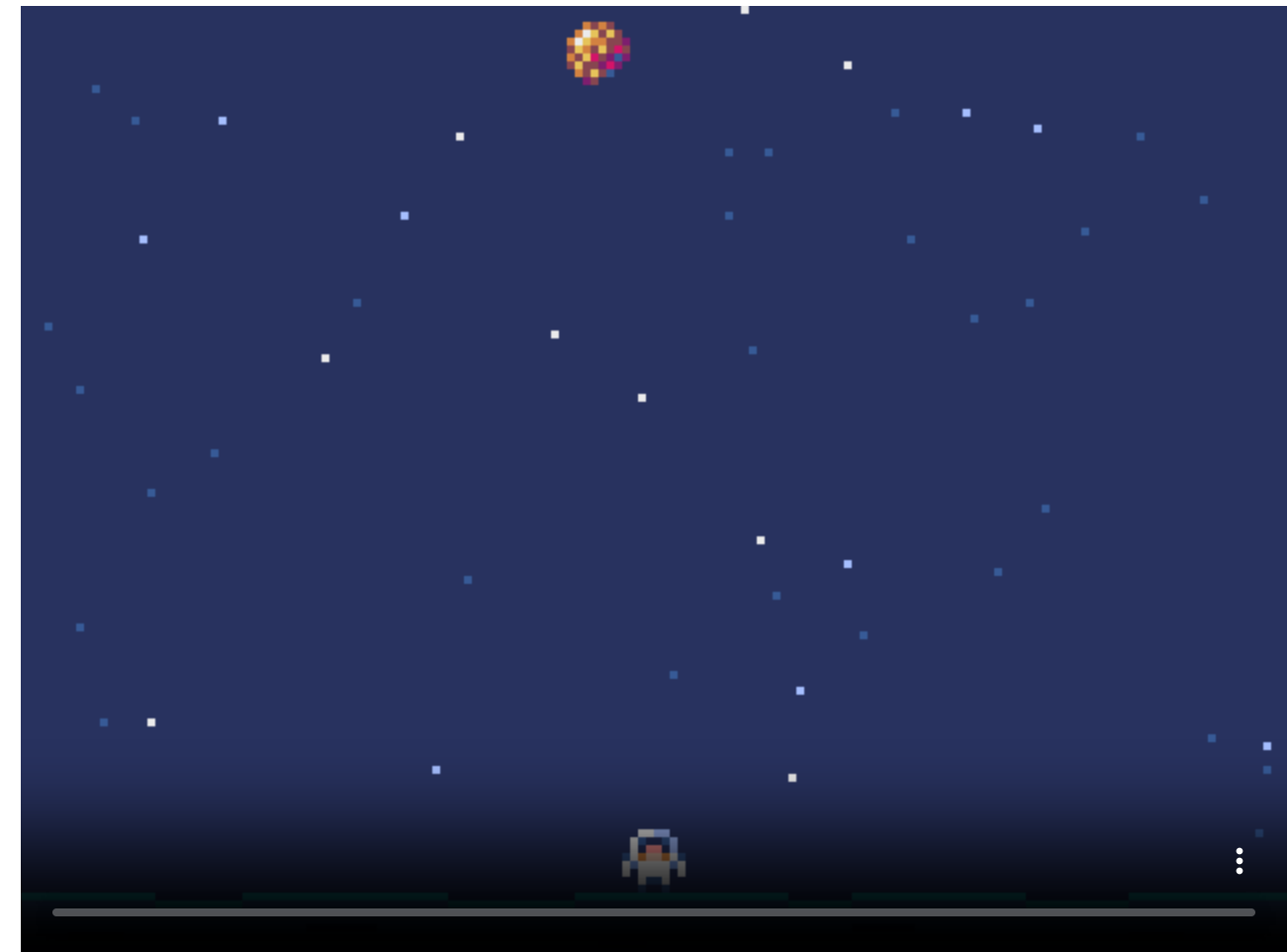
pyxel.init(160, 120)
pyxel.load("game.pyxres")
pyxel.gen_bgm(7, 0, 3, 0, play=True)
x = 76
enemies = [[pyxel.rndi(0, 152), -i * 16] for i in range(16)]
game_over = False

while True:
    pyxel.blt(0, 0, 0, 0, 0, 160, 120)

    if pyxel.btn(pyxel.KEY_LEFT) and not game_over:
        x -= 2
    if pyxel.btn(pyxel.KEY_RIGHT) and not game_over:
        x += 2

    for enemy in enemies:
        if not game_over:
            enemy[1] += 2
            if enemy[1] > 120:
                enemy[0], enemy[1] = pyxel.rndi(0, 152), -8
            if abs(enemy[0] - x) < 8 and abs(enemy[1] - 104) < 8:
                pyxel.play(3, 0)
                game_over = True
    pyxel.blt(enemy[0], enemy[1], 0, 8, 120, 8, 8, 0)

    pyxel.blt(x, 104, 0, 0, 120, 8, 8, 0)
    pyxel.flip()
```



# 制約は「学びの道具」でもあった

Limits Were Also a Tool for Learning



ないものは作り、できないことは工夫で越えた。Pyxel は「あの頃」の再現でもある

You built what didn't exist and improvised past what you couldn't do. Pyxel is a recreation of those days

**1986年** プログラムを始めた年

I started programming in 1986

調べたくてもネットは無い。雑誌やプロの作品から学び、BASIC が遅ければマシン語で書いた

No internet. You learned from magazines and the pros, and wrote machine code when BASIC lagged



**1998年** メタルギアソリッド（PS1）

MGS shipped on PS1 in 1998

性能では、思い描いた 3D に届かない。ポリゴンを減らし、足りない分は見せ方で補った

The hardware fell short of the 3D we wanted. We cut polygons and made up the rest in the look



**2001年** ゾーンオブエンドース（PS2）

Z.O.E shipped on PS2 in 2001

開発キットは未成熟で、必要なものが揃わない。ゲームより先に、作るための道具から作った

The dev kits were immature and nothing we needed was there. Before the game itself, we built the tools to build it



# 制約の外し方を遊びにする

Making the Way Around a Limit Into Play



Pyxel の存在意義は制約。その外し方を隠して、自由度と両立させている

The limits are why Pyxel exists. Hiding the way around them lets limits and freedom coexist

## ① 制約の解除方法を裏技化

Turned the way around a limit into a cheat code

配列の要素数操作や、隠しクラスのインスタンス登録で、色数や同時発音数を書き換えられる

Resize the arrays or register a hidden class instance, and the color and channel counts change

```
EXTRA_COLORS = [  
    0xFF004D, 0xFFA300, 0xFFEC27, 0x00E436,  
    0x29ADFF, 0x83769C, 0xFF77A8, 0xFFCCAA,  
    0x291814, 0x111D35, 0x422136, 0x125359,  
    0x742F29, 0x49333B, 0xA28879, 0xF3EF7D,  
]  
pyxel.colors[16:] = EXTRA_COLORS  
pyxel.channels.append(pyxel.Channel())
```



## ② 上級者向け API を分離

Split the advanced API off

データの直接操作など、専門知識が必要な API を通常時はリファレンスで非表示に。チェックを入れる则表示される

APIs needing specialized knowledge, like direct data access, are hidden in the reference until you tick the box

Search APIs...

☒ Advanced

FUNCTION **quit()**

Quit the Pyxel application.

FUNCTION **reset()**

Restart the Pyxel application from the beginning.

FUNCTION **title(title)** ADV

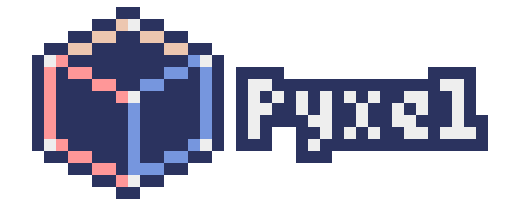
Set the window title.

► Details



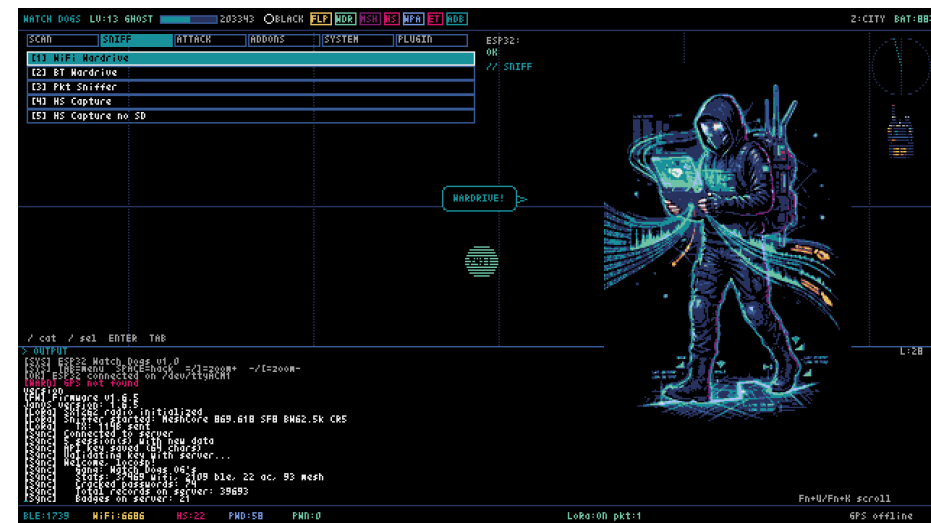
# Python ならではの広がり

## What Python Opened Up



日常の Python に、出来心で `import pyxel`。そこから愉快的作品が生まれている

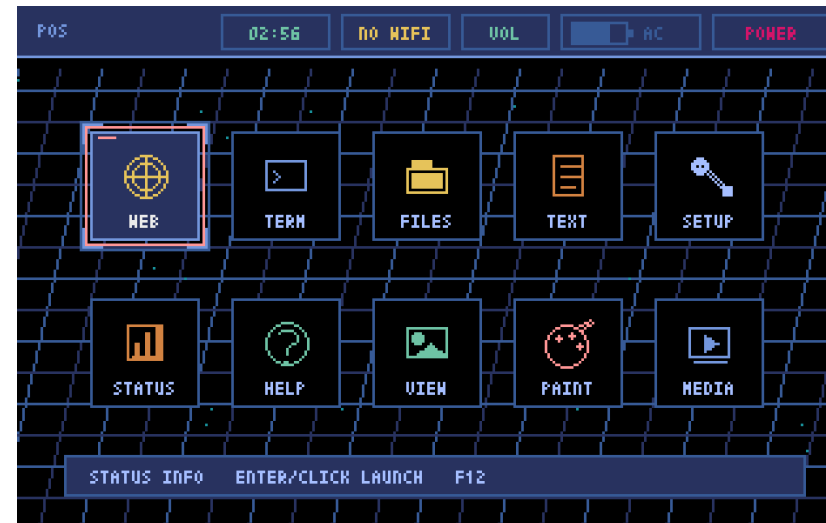
An idle import pyxel in everyday Python — and delightful works keep appearing



Watch Dogs Go © LOCOSP

ESP32 の基板をつなぎ、周りの Wi-Fi や Bluetooth を拾う。操作画面は Pyxel

An ESP32 board picks up nearby Wi-Fi and Bluetooth. The control screen is Pyxel



GPD Pocket POS © moto

7インチの PC で、デスクトップの代わりに起動。ブラウザも端末もここから開く

It boots on a 7-inch PC in place of the desktop. The browser and terminal open from here



Pen Apple © RuneBlaze

カードの効果は文章で書くだけ。それを LLM がゲームのルールにする

Cards are just sentences — an LLM turns them into the game's rules



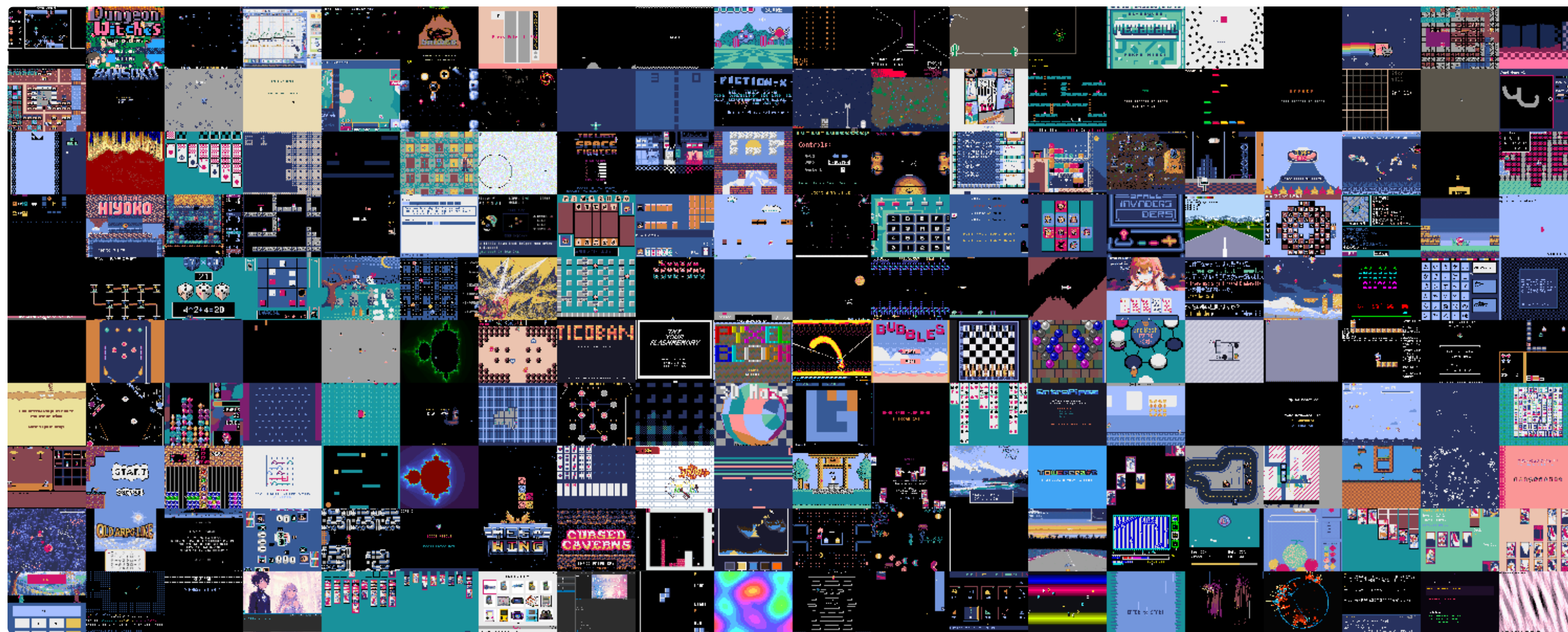
# アイデアを形にできると、うれしい

Bringing Your Ideas to Life Feels Great



## みんなのアイデアも形になり、Pyxel User Examples に集結

Other people's ideas took shape and gathered in Pyxel User Examples





## CHAPTER 2

# みんなと作るのって、面白い

Making Things Together Is Fun



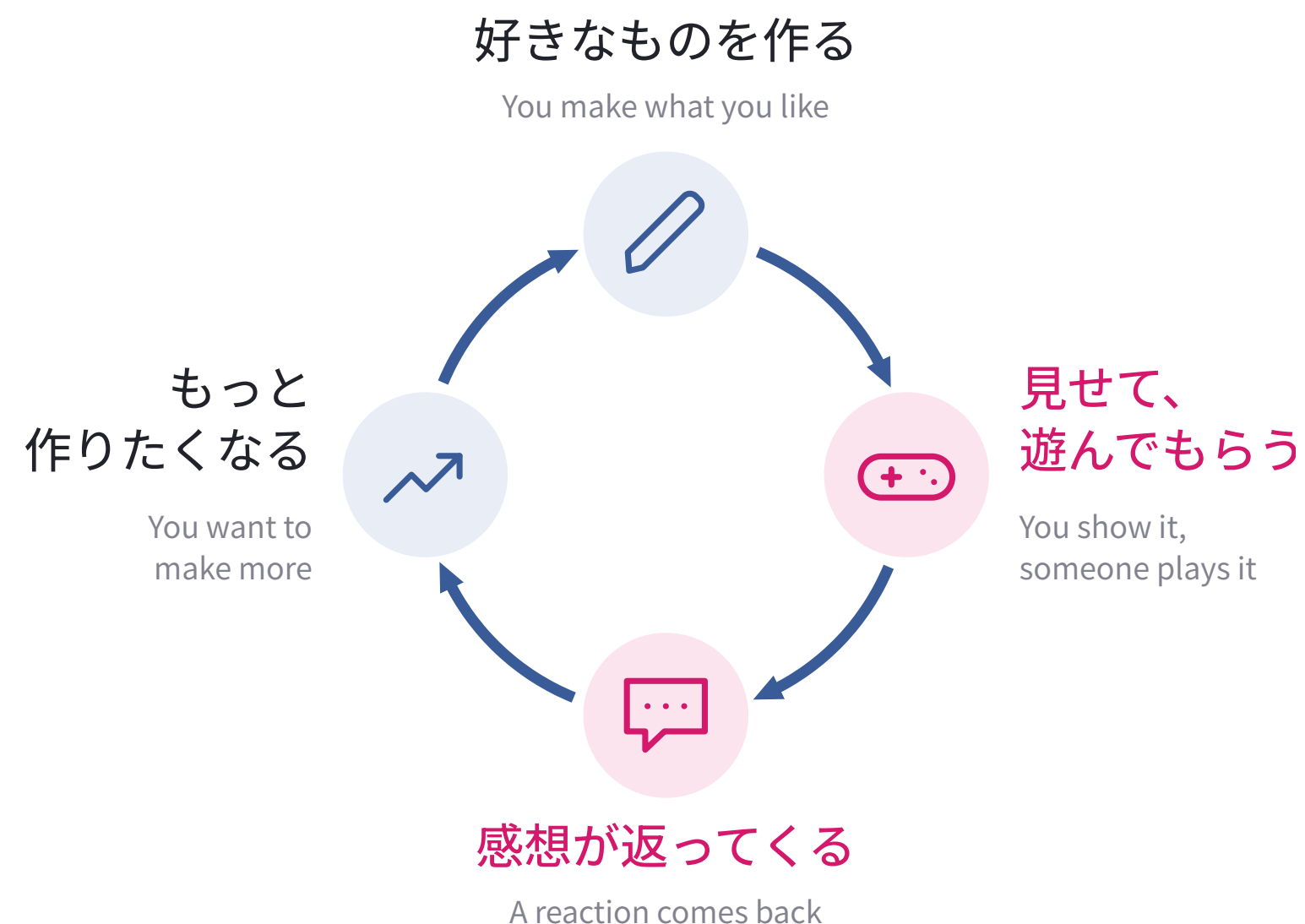
# 遊んでもらう楽しさ

The Fun of Having People Play



作って終わりにせず、共有することで、次の展開が始まる

Making it is not the end — share it, and the next chapter begins

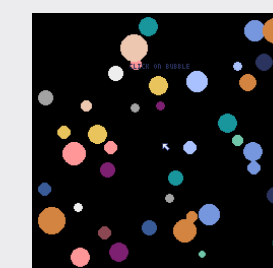


## Pyxel の「共有の仕組み」

How Pyxel makes sharing work

### スクリーンショット機能

Screenshot & screencast



Alt 1 画面を画像で保存  
Screenshot

Alt 3 直近10秒を GIF で保存  
Last 10s as a GIF

### 作品共有の場：Pyxel User Examples

A place to share your work — Pyxel User Examples

### 配布用形式：Pyxel アプリケーションファイル

A distribution format — the Pyxel application file



# Pyxel アプリケーションファイル

Pyxel Application File



必要なファイルを1つに詰めた Pyxel 専用のファイル形式。コマンド1つで作成・実行できる

A Pyxel-specific file format that packs everything into one file — one command to build, one to run



残る課題  
Still unsolved

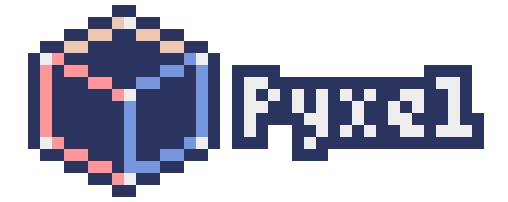
Python と Pyxel がインストールされた環境でないと、渡しても動かせない

Unless Python and Pyxel are installed, handing it over gets them nowhere



# Web 版 Pyxel への道のり

The Road to Pyxel on the Web



## コミュニティと周辺プロジェクトを巻き込んだ「みんなと作る」総力戦になった

It became an all-out effort — making things together with the community and nearby projects

- 2018年10月 **jahodfra** 氏が「**Brython** で動かせないか」と issue に書き込み  
jahodfra asks in an issue: could Brython run it?
- 2021年12月 13ヶ月かけて、コアエンジンの C++ から **Rust** への移行が完了する  
After 13 months, the core engine finishes moving from C++ to Rust
- 2022年5月 **km19809** 氏が数ヶ月かけて、コアエンジンの **WASM** 化に成功する  
km19809 spends several months and compiles the core engine to WASM
- 2022年8月 **maturin** の **messense** 氏と共に、Python 側も **WASM** 化する  
With messense of maturin, we compile the Python side too
- 2022年9月 **SDL** 向けに改造した **Pyodide** を同梱して、Web 版を公開開始  
Pyxel ships for the web with a Pyodide we modified for SDL
- 2023年4月 **Pyodide** の **Choi** 氏が **SDL** 対応を実装。改造版の同梱が不要になる  
Choi of Pyodide lands SDL support, and the modified build is no longer bundled

-  **Brython**  
Python を JavaScript に変換して動かす実装  
Runs Python by translating it to JavaScript
-  **Rust**  
Web 等の各種環境向けにコンパイルできる言語  
Compiles to many targets, including the web
-  **WASM**  
ブラウザが直接実行できる、機械語に近い形式  
A near-machine-code format browsers run directly
-  **Pyodide**  
Python 本体を WASM にコンパイルした実装  
CPython itself, compiled to WASM
-  **maturin**  
Rust 製の Python 拡張をパッケージにするツール  
Packages Rust-built Python extensions
-  **SDL**  
画面・音・入力を扱うライブラリ  
A library for screen, sound, and input



# Web 版 Pyxel の仕組み

## How Pyxel Works on the Web



- 1 Pyxel 本体を CDN サーバーに置き、インストールなしで実行  
Pyxel sits on a CDN — it runs with no install

pyxel.js スクリプトが、Python 実行環境（Pyodide）と Pyxel 本体を読み込む  
The pyxel.js script pulls in Pyodide (a Python runtime) and Pyxel itself

- 2 独自の HTML タグを定義し、HTML 内に Python を直接記述  
A custom HTML tag lets Python live right inside the HTML

ブラウザがタグを見つけると、中身のコードを取り出して Pyxel に渡す  
When the browser finds the tag, it hands the code inside to Pyxel

- 3 pyxapp を `pyxel app2html` コマンドで HTML に変換  
The pyxel app2html command turns a pyxapp into a standalone HTML file

pyxapp を Base64 エンコードし、復元コードとともに HTML に書き込む  
The pyxapp is Base64-encoded and written into the HTML with the code that restores it

```
<script src="https://cdn.jsdelivr.net/gh/kitao/pyxel/wasm/pyxel.js"></script>

<pyxel-run script="
import pyxel
pyxel.init(200, 150)
pyxel.cls(8)
pyxel.line(20, 20, 180, 130, 7)
pyxel.show()
"
></pyxel-run>
```

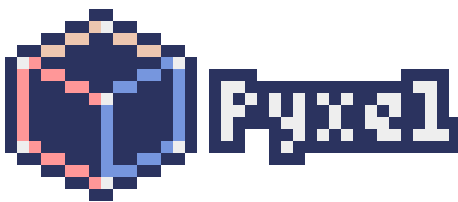
HTML ファイルを開く

Open the HTML file



# URL だけで遊べる究極形へ

The Ultimate Form: Play From a URL Alone



## Pyxel Web Launcher — GitHub に置いたファイルを、URL で指定して起動する

Pyxel Web Launcher — point a URL at a file on GitHub to launch it

`https://kitao.github.io/pyxel/web/launcher/?run=taro / my_repo / main / src/title`

ランチャーの場所  
the launcher

コマンド  
command

ユーザー  
user

リポジトリ  
repo

ブランチ  
branch

パス  
path



共有の準備は必要なく、GitHub で公開されていれば、そのまま遊べる

No setup to share — if it is public on GitHub, it just plays



# Web 技術で広がる可能性

What Web Tech Opens Up



他の技術と繋がり、別の環境に組み込める。Web ならではのアプリの形が見えた

Wired to other tech, embedded elsewhere — a kind of app possible only on the web came into view

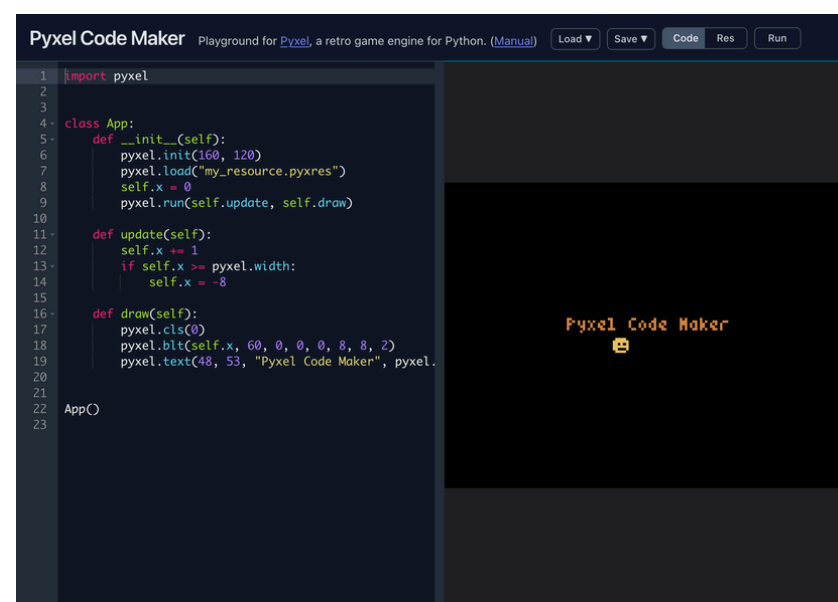


**Pyxel Code Maker**

Pyxel Code Maker

ブラウザ上で動く統合開発環境

An IDE that runs in the browser

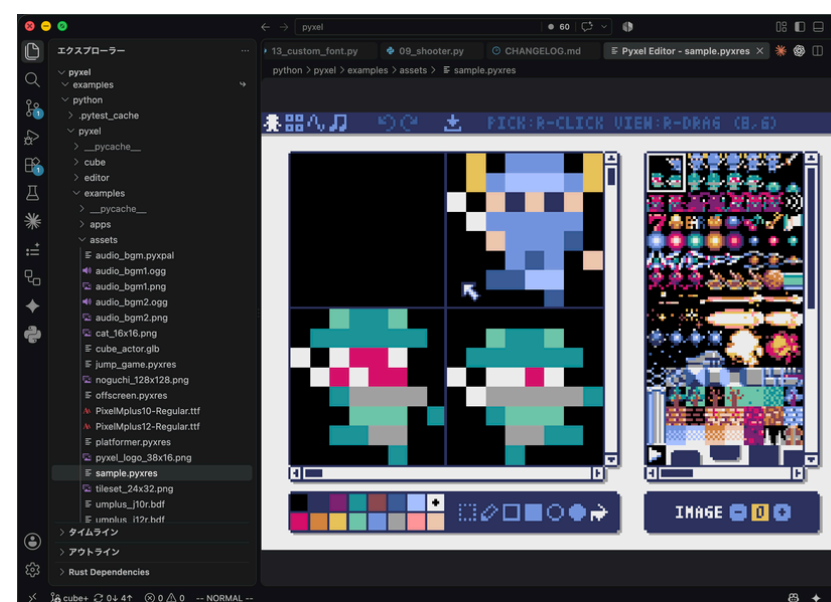


**Pyxel VS Code 拡張**

Pyxel VS Code extension

VS Code 上で動く実行環境

A runtime that runs inside VS Code

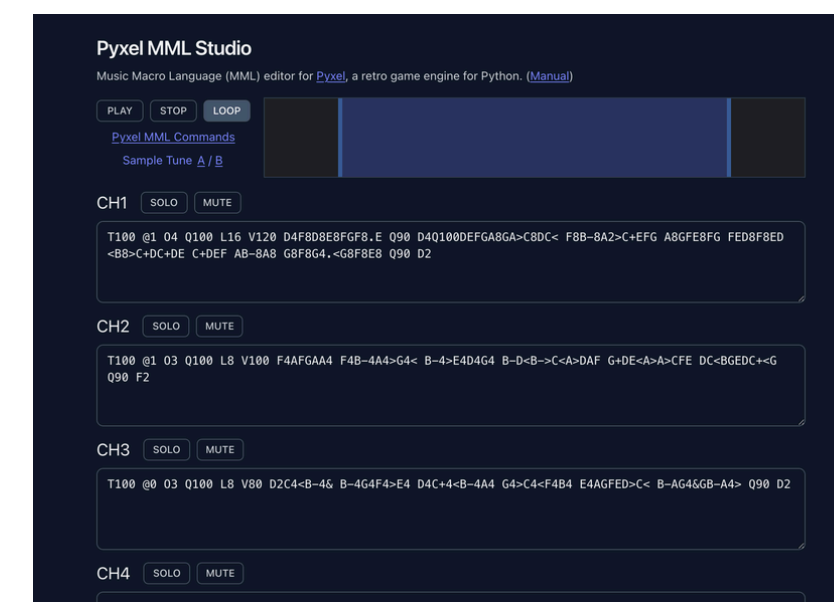


**Pyxel MML Studio**

Pyxel MML Studio

Pyxel MML の音楽制作ツール

A music tool for Pyxel MML



# Web 技術の応用① Pyxel Code Maker

Applying Web Tech 1 — Pyxel Code Maker



教室で最初につまずくのは、環境構築——なら IDE ごと Web アプリにしよう

The first wall in a classroom is setup — so make the IDE itself a web app

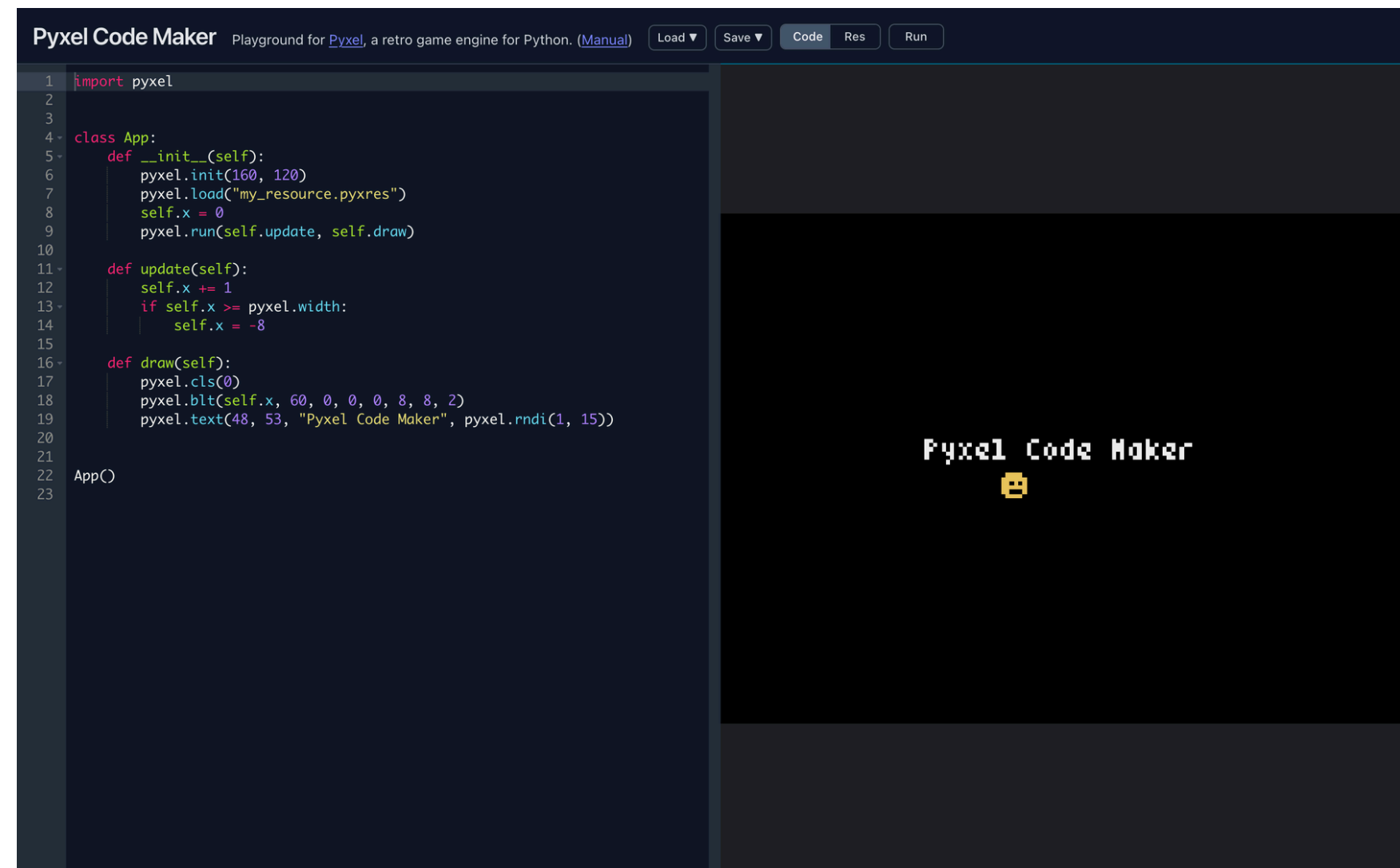


## Pyxel Code Maker

Pyxel Code Maker

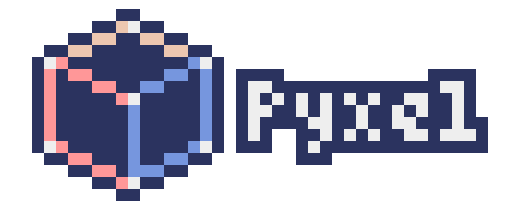
ブラウザ上で動く統合開発環境

An IDE that runs in the browser



# Web 技術の応用② Pyxel VS Code 拡張

Applying Web Tech 2 — Pyxel VS Code Extension



VS Code の中身はブラウザ —— じゃあ、Pyxel をそのまま動かせるはず

VS Code is a browser inside — so Pyxel should just run there

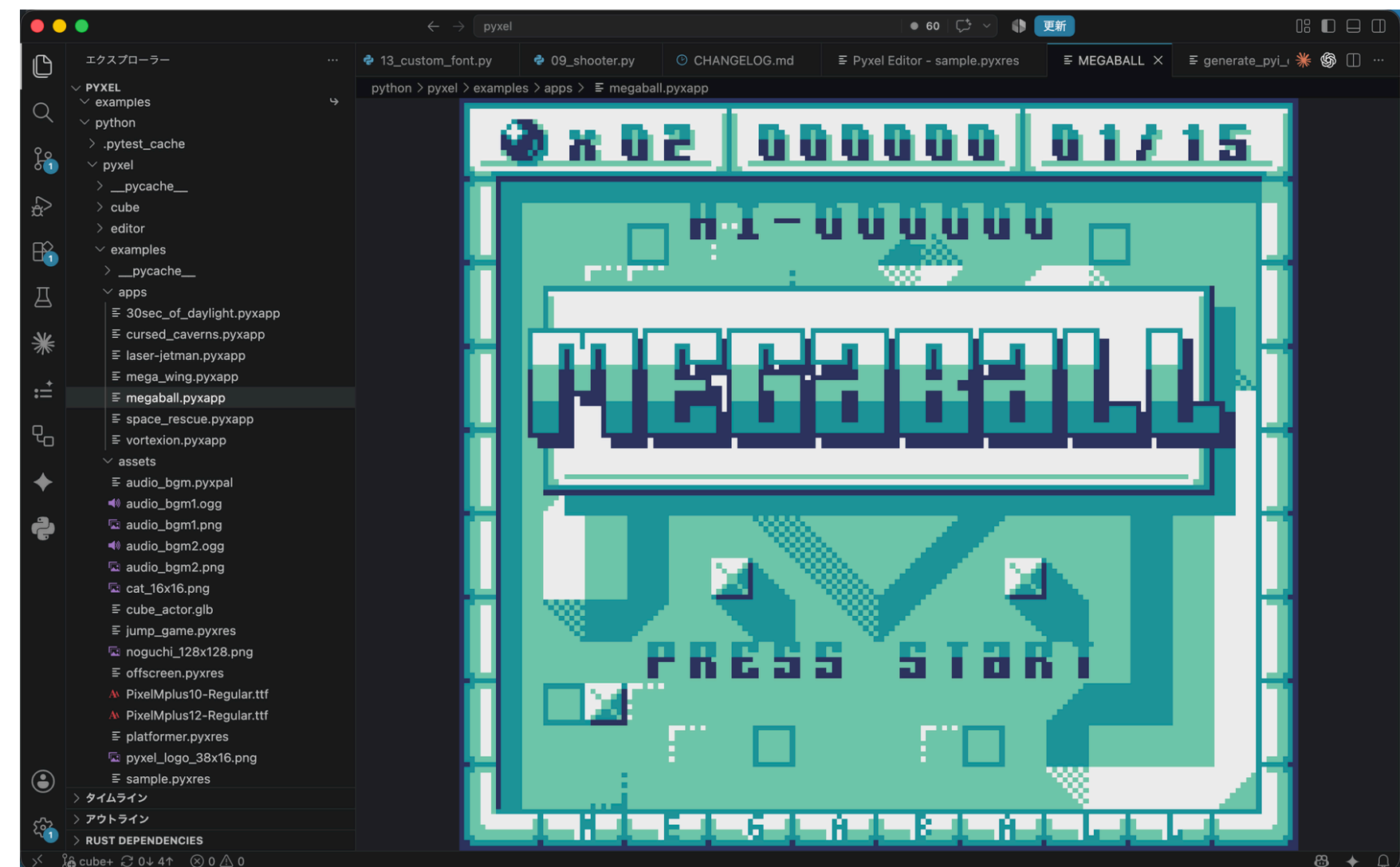


## Pyxel VS Code 拡張

Pyxel VS Code extension

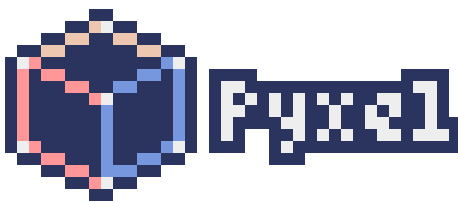
### VS Code 上で動く実行環境

A runtime that runs inside VS Code



# Web 技術の応用③ Pyxel MML Studio

Applying Web Tech 3 — Pyxel MML Studio



音楽が文字列で書けるなら —— 曲は QR コードに丸ごと入るのでは？

If music is just text — then maybe a whole tune fits in a QR code?



## Pyxel MML Studio

Pyxel MML Studio

### Pyxel MML の音楽制作ツール

A music tool for Pyxel MML

### MML (Music Macro Language)

音楽を文字列で書く記法

A notation for writing music as text

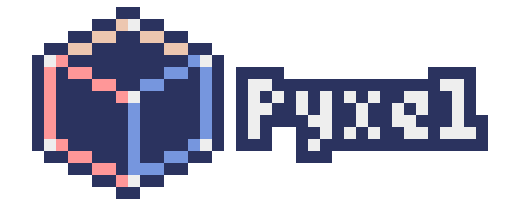
```
T120 04 L8 CDEFGAB>C
```

テンポ120・オクターブ4・8分音符で、ドレミファソラシド  
Tempo 120, octave 4, eighth notes — then the scale



# みんなと作るのって、面白い

Making Things Together Is Fun



## 教育機関への展開

Into schools and universities



- フランスの高校情報科（NSI）—— 国の公式教材に採用

Official course material for French high-school CS

- 大学の授業 —— 慶應 SFC・大阪大学・スペイン UC3M・ニュージーランド Unitec

Keio SFC, Osaka Univ., UC3M (Spain), Unitec (NZ)



## ゲームジャムへの展開

Into game jams

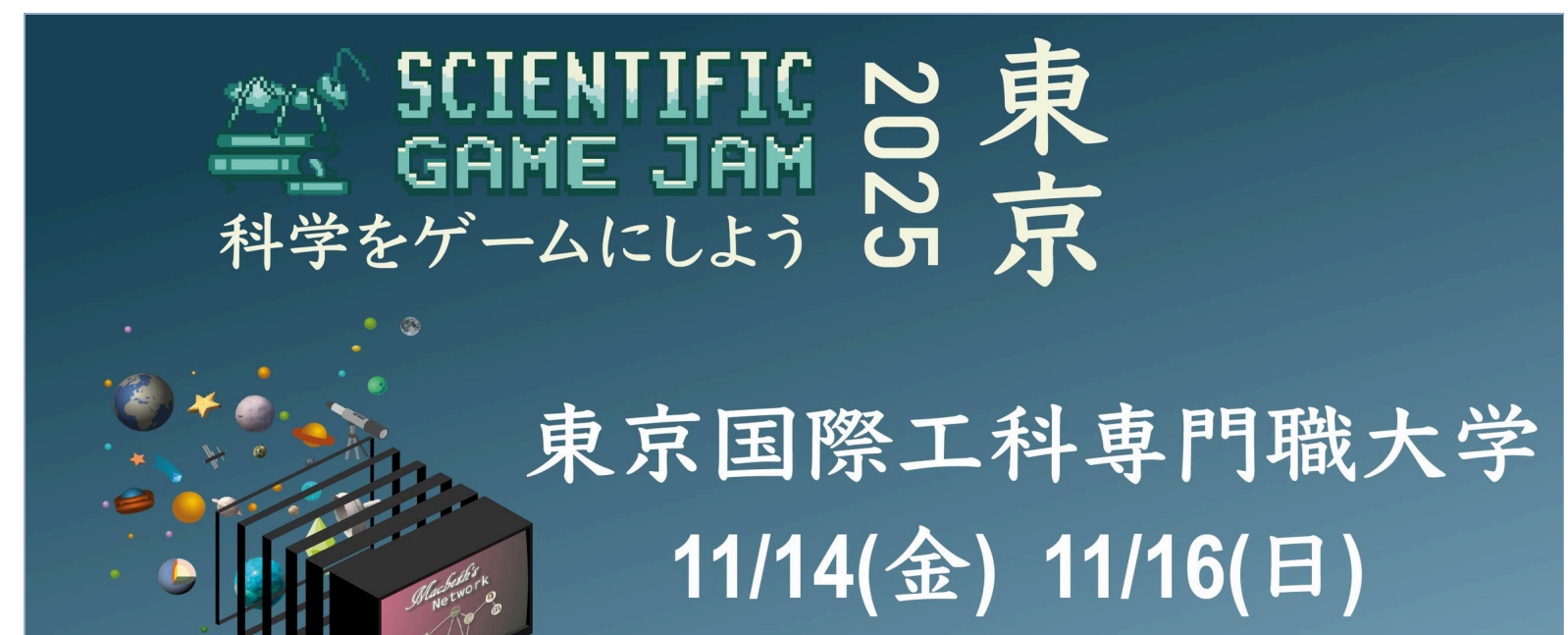


- Nuit du Code —— 61か国・609校が参加する6時間のプログラミングイベント

A 6-hour coding event — 609 schools in 61 countries

- Scientific Game Jam —— 科学をテーマにしたゲームジャム。高校生チームが Pyxel で制作

A game jam on science — a high-school team used Pyxel





## CHAPTER 3

# これからの Pyxel は、もっと楽しい！

Pyxel's Future Is Even More Exciting!



# AI 時代に「楽しく作る」は必要か？

Does the Joy of Making Still Matter in the Age of AI?

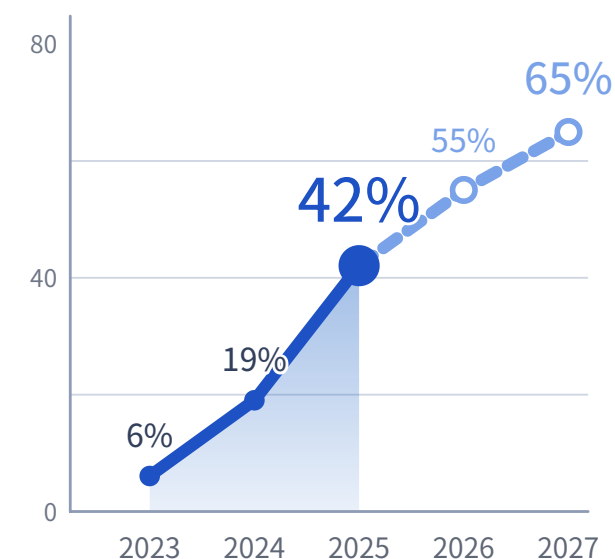


## AI が書くコードの割合は年々上昇中

The share of AI-written code climbs year after year

AI 生成・支援の  
コードの割合  
Share of AI-generated  
or assisted code

Sonar 「State of Code」 2026  
2026年以降は見通し  
Sonar State of Code 2026.  
2026 onward is an outlook.



## プログラミングとの関わり方が変わる

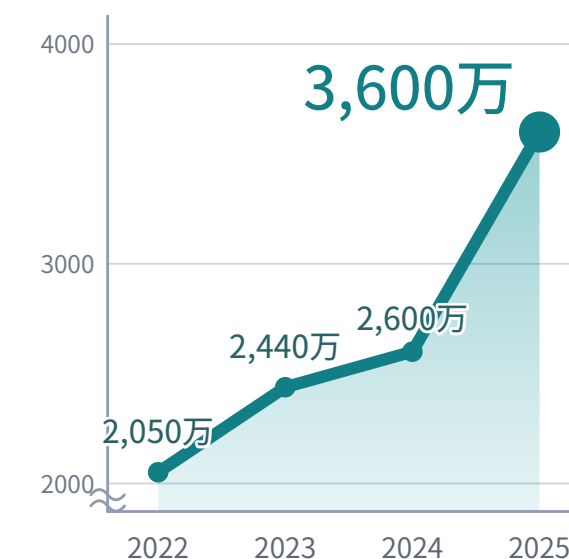
How people take part in programming is changing

## 新たに開発者となる人も増加中

And more and more people are becoming developers

GitHub に参加した  
新規開発者数  
New developers  
per year on GitHub

GitHub Octoverse  
2023・2024年は算出値  
GitHub Octoverse.  
2023 and 2024 are derived.



## 作りたい人は、実はたくさんいる

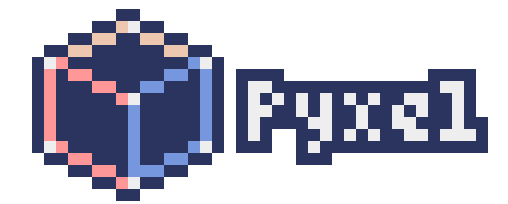
There are, in fact, plenty of people who want to make

何を作るか考え、方向づけるのは人の役割。「情熱を育てる」仕掛けがより重要に  
Setting the goal and giving direction is the human role — and devices that grow the passion matter more than ever



# 「楽しく作る」ための AI の立ち位置

Where AI Stands in the Joy of Making



AI はやる気を奪うものではなく、「作る楽しさ」を広げるパートナーになれる

AI does not take away the will to make — it can be a partner that widens the joy of making

## 1 チームメンバーの一員として As one of the team members

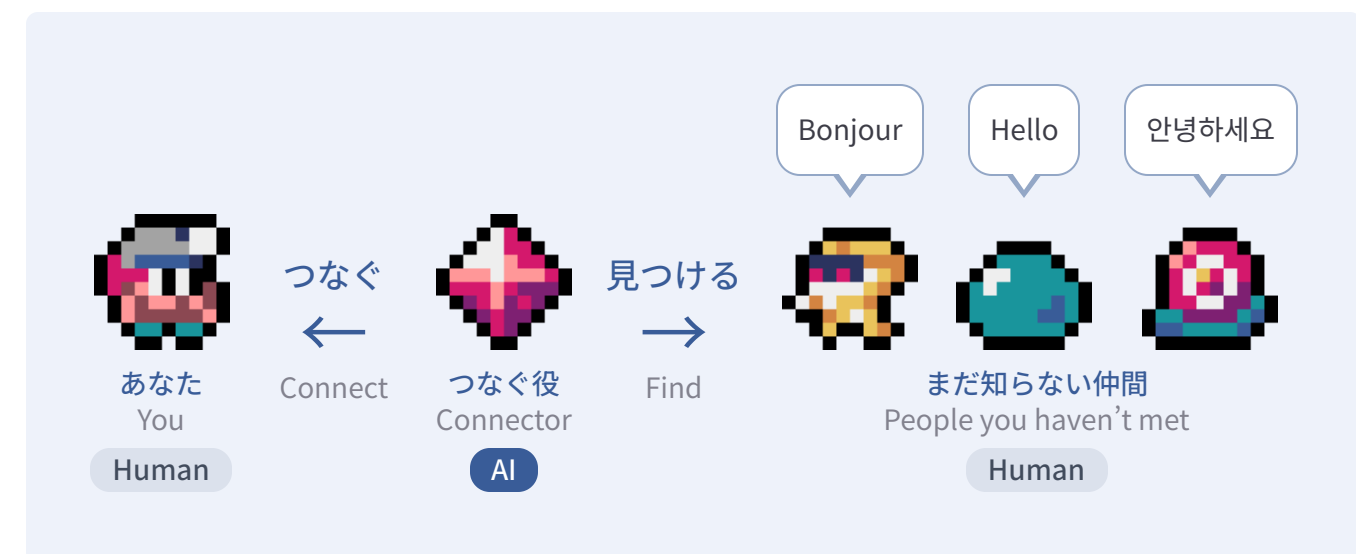
人と同じように担当を持ち、一緒に作る  
AI holds a role just like the others, and builds together



アイデアを出す (Generate ideas)    コードを書く (Write code)    動かして試す (Run and test)    改善する (Improve)

## 2 チームメンバーをつなぐ役として As the one who connects the team

AI が橋渡しをして、国や言葉の違う人と一緒に作る  
AI bridges people across countries and languages



仲間を見つける (Find people)    会話を翻訳する (Translate)    意図を伝える (Convey intent)    段取りを組む (Coordinate)



# AI 時代への Pyxel の対応

How Pyxel Is Meeting the Age of AI

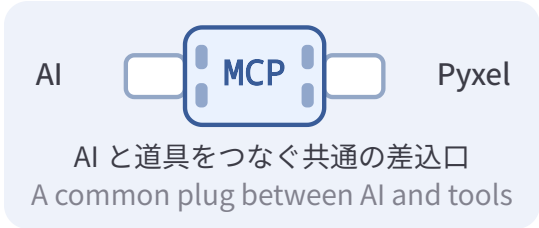


## 人の使い方は変えずに、AI のための道具を新たに開発

Without changing how people use it, new tools built for AI



**pyxel-mcp**  
MCP 準拠 / Built on MCP



AI と道具をつなぐ共通の差込口  
A common plug between AI and tools

**AI 用 Pyxel 操作ツールキット**  
A Pyxel toolkit for AI



**超早送り実行**  
Turbo run  
通常の300倍速で実行  
Runs at 300× normal speed



**事前ミス検出**  
Static checks  
動かす前に誤りを見つける  
Finds errors before it runs



**擬似コントローラー**  
Virtual controller  
キー操作を送り込む  
Feeds key presses in



**画面の差分比較**  
Frame diff  
1ドットの違いも見つける  
Catches a one-pixel change



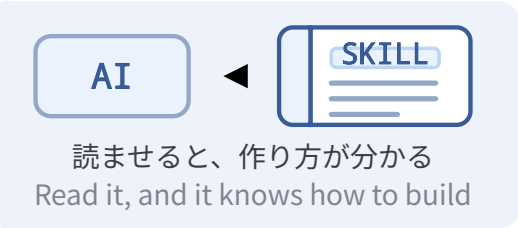
**ヘッドレス実行**  
Headless run  
ウィンドウを開かない  
No window at all



**画像・音声ファイル保存**  
Image and audio export  
画面を PNG、音を WAV で  
Screens as PNG, sound as WAV



**pyxel-skill**  
Agent Skill 形式 / An Agent Skill



読ませると、作り方が分かる  
Read it, and it knows how to build

**AI 用 Pyxel 作業手順書**  
A Pyxel working guide for AI

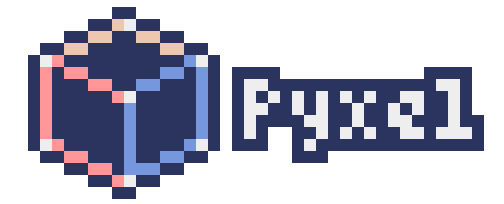
**SKILL.md**

**## 作る手順** Workflow  
1. 遊べる最小の形を見きわめる Find the smallest playable scope  
2. 入口・操作・目的・やり直しを作る Build one complete slice  
3. **validate** で誤りと警告を確かめる Check errors and warnings  
4. **run** でコマ0から動かし画面を撮る Run from frame 0 and capture  
5. ログを読み、撮った画面を見る Read the log, look at the frame  
6. 見つけた不具合だけを直す Fix only observed defects  
7. 操作方法と結果を報告する Report the controls and results



# 「新しいレトロ」への挑戦

Taking On a New Kind of Retro



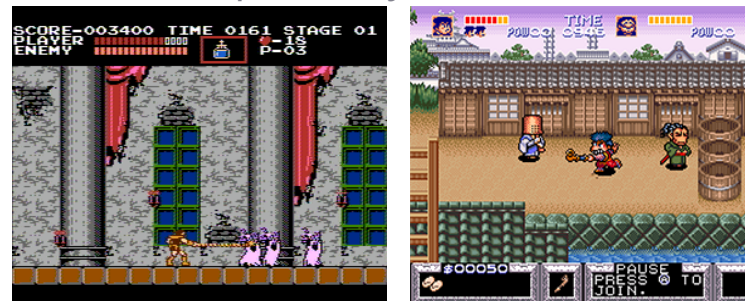
PS2 でさえ、もう26年前。Pyxel で「レトロ 3D ゲーム」が作れてもいい頃

Even the PlayStation 2 is 26 years old — about time Pyxel let you make retro 3D games



## レトロ 2D 時代 The retro 2D era

ドット絵・平面 (従来の Pyxel)  
Pixel art, flat plane (Pyxel so far)



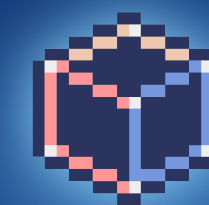
悪魔城ドラキュラ © 1986 KONAMI  
がんばれゴエモン © 1991 KONAMI

## レトロ 3D 時代 The retro 3D era

ローポリゴン・立体空間  
Low poly, 3D space



METAL GEAR SOLID © 1998 KONAMI  
ZONE OF THE ENDERS © 2001 KONAMI



## Pyxel Cube

Pyxel 3.0 で追加される  
レトロ 3D モジュール  
A retro-3D module  
added in Pyxel 3.0



# Pyxel Cube の使い方

## How to Use Pyxel Cube

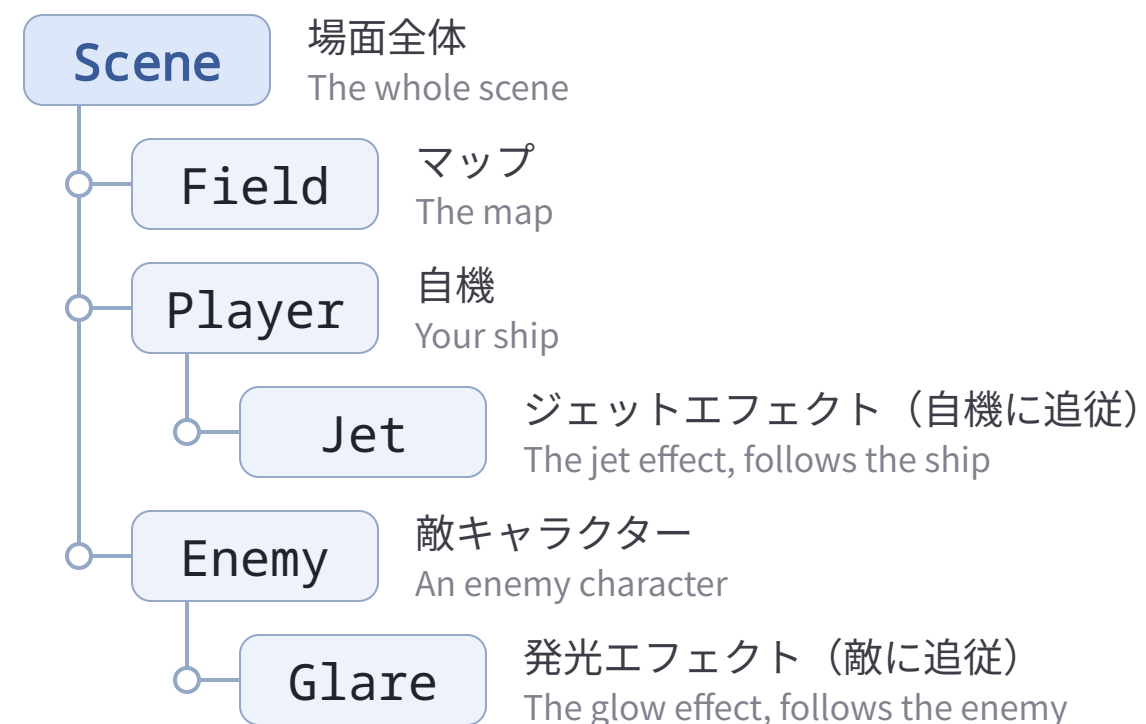


ノードツリーで要素を配置。ノードの書き方は、2D 版と同じ考え方

Arrange things with the node tree. Writing a node follows the same idea as in 2D

### ノードツリー

The node tree



### Enemy ノードの中身

Inside the Enemy node

```
from pyxel.cube import Node, Vec3

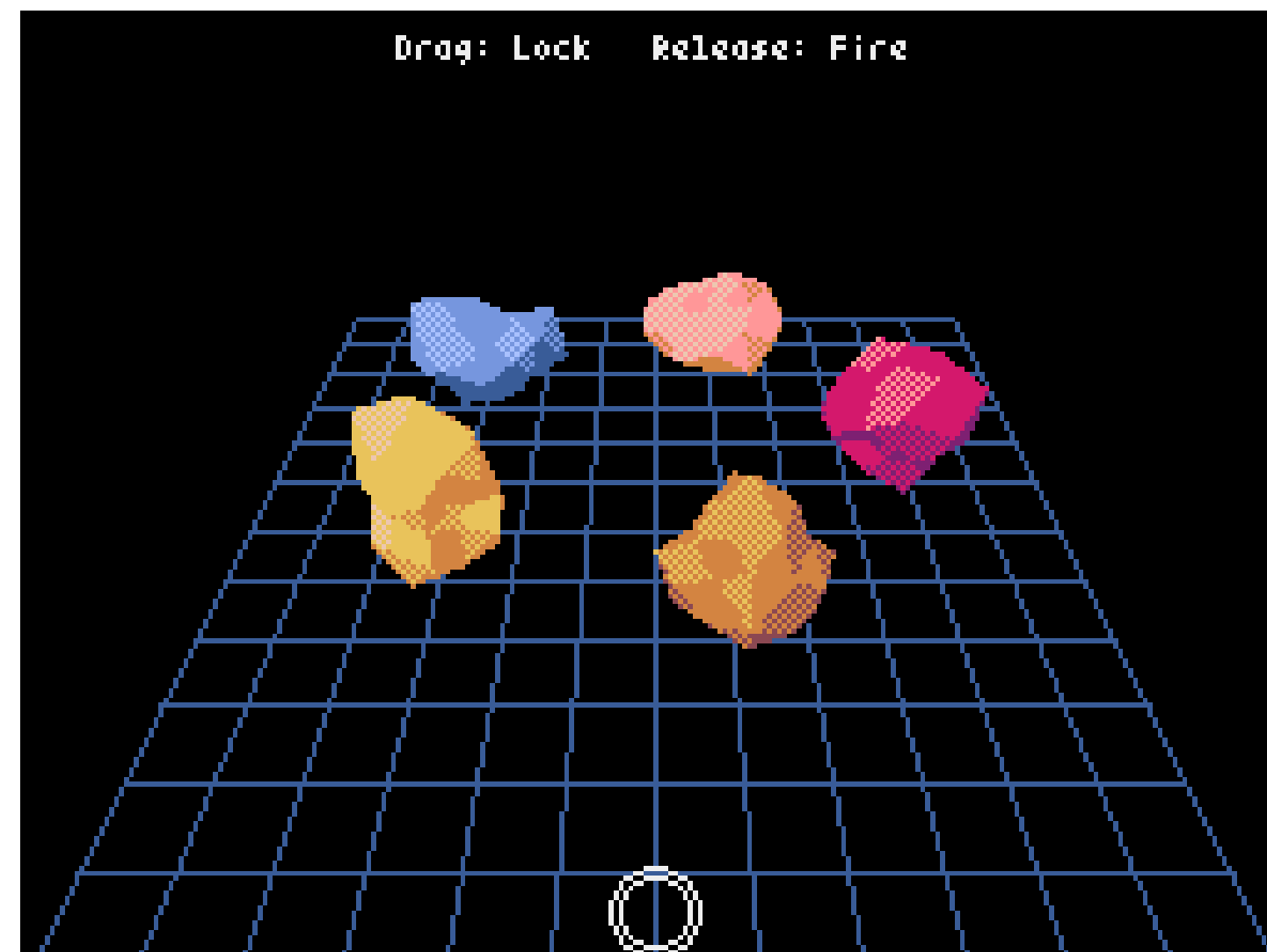
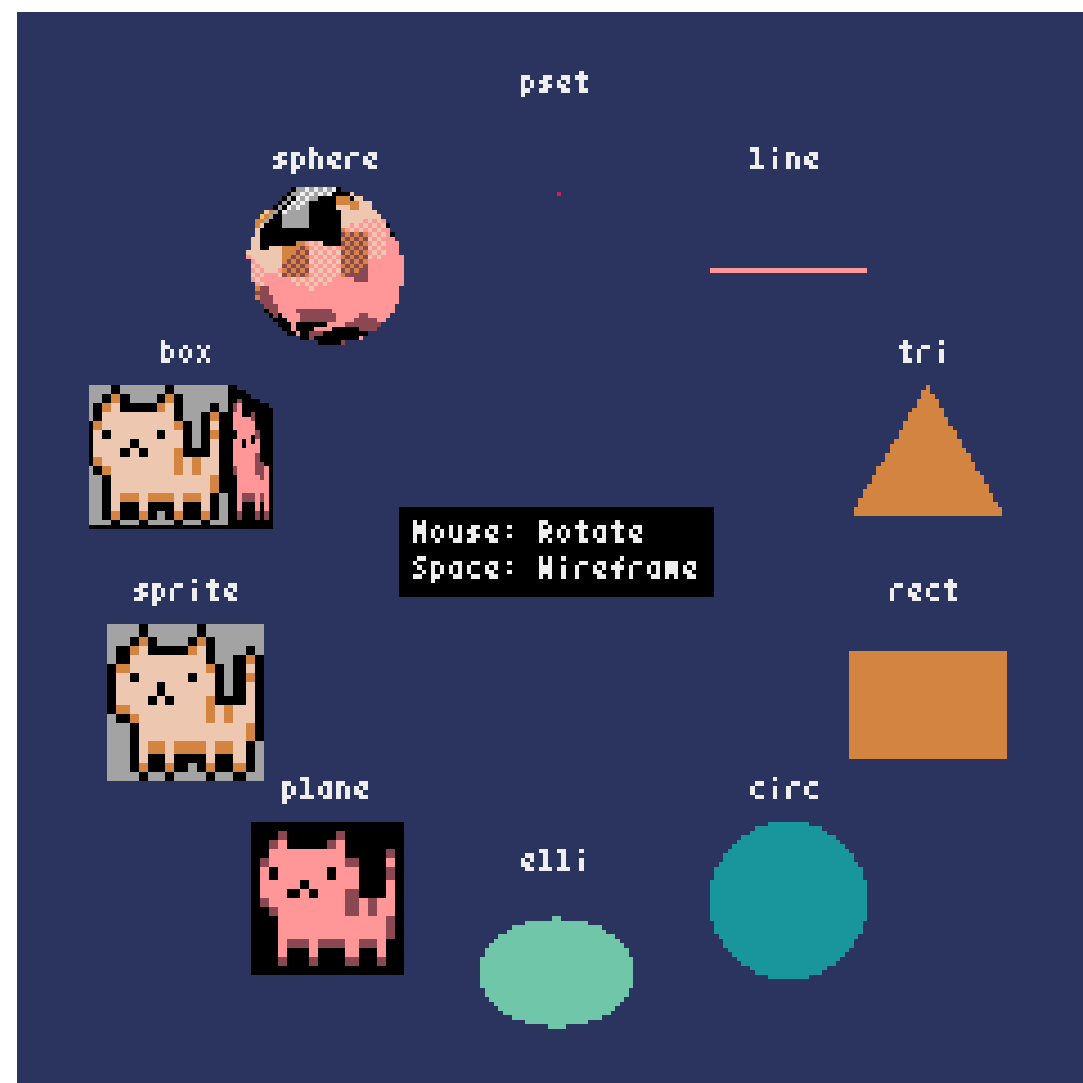
class Enemy(Node):
    def on_update(self):
        self.transform = self.transform.translate(
            Vec3(0.05, 0, 0))

    def on_draw(self):
        self.circ(Vec3.ZERO, 0.5, 11)
        self.text(Vec3(0, 0.8, 0), "enemy", 7)
```



# Pyxel Cube 動作画面

Pyxel Cube in Action



近日公開予定

Coming soon



# これからの Pyxel は、もっと楽しい！

Pyxel's Future Is Even More Exciting!



## Pyxel で AI がもっと身近に

AI feels closer with Pyxel

AI は新しい仲間になれる

AI can be a new teammate



pyxel-mcp と pyxel-skill で  
AI と Pyxel が効率よく連携

pyxel-mcp and pyxel-skill let an AI  
and Pyxel work together efficiently



## Pyxel は「新しいレトロ」へ

Pyxel heads for a new kind of retro

3D のゲーム機も、レトロの領域に

3D consoles are entering retro territory too



Pyxel 3.0 で Pyxel Cube を追加  
2D と同じ書き方で、3D も作れる

Pyxel Cube arrives in Pyxel 3.0 —  
written the same way as 2D, now in 3D



AI 時代も、Pyxel で気軽に楽しくプログラミング！

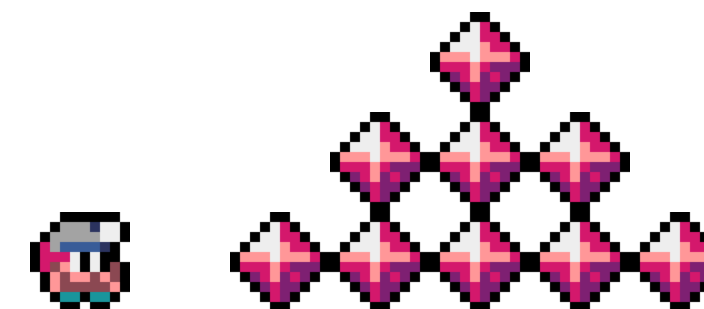
Even in the age of AI, easy and fun programming with Pyxel!





おわりに

In Closing



# 「楽しく作る」は、デザインできる

The Fun of Making Can Be Designed

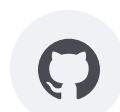
これからも一緒に、Pyxel と Python で  
プログラミングを遊んでいきましょう！

Let's keep playing programming together — with Pyxel and Python!



@kitao

Pyxel 関連のニュースを発信しています  
News around Pyxel



<https://github.com/kitao/pyxel>

スターをつけて応援よろしくお願いします！  
A star would mean a lot

発表資料 Slides

