モダンな言語

RustでもVJしたい！

kyoto.rs#1

2026/5/9 / taiseiue

自己紹介

- taiseiue
- X: @taiseiue / id:taiseiue
- Kyoto.cs (C#) / WSOFT
- Rustは今週初めた
- プログラミング言語
  作るのが趣味

導入

taiseiueの日記

7
taiseiue PRO
読者になる
@taiseiueをフォロー

検索
記事を検索

カテゴリー
イベント (6)
技術 (14)

イベント

RubyKaigi 2026に行きました

2026-05-09

id:taiseiue です。4/22~24に北海道は函館で開催されていた「RubyKaigi 2026」に参加しました。

RubyKaigi 2026
RubyKaigi 2026, #rubykaigi
rubykaigi.org 2 users

rubykaigi.org

参加にあたっては金銭面だけでなく、飛行機やホテルの手配までSTORESさんにお世話になりました。ありがとうございます!

https://d.taiseiue.jp/entry/2026/05/09/090625

matz / spinel

Type / to search

Code | Issues 7 | Pull requests 1 | Agents | Actions | Projects | Security and quality | Insights

spinel

Public

Watch 8 | Fork 46 | Starred 1.5k

master | 4 Branches | 0 Tags | Go to file T | Add file | Code

matz and claude fix(codegen): regex const initialised with .freeze resolv... c0828f5 · 4 hours ago 1,280 Commits

.github/workflows	ci: wrap cc with sccache + GitHub Actions cache backend	5 days ago
benchmark	bench: rename 10 files to bm_ prefix for naming consistency	last week
docs	docs: add Class object design memo + fix Windows test p...	8 hours ago
examples/ffi	examples(ffi/sqlite): toy blog on sqlite3	5 days ago
lib	fix: mark sp_argv strings during GC	12 hours ago
test	fix(codegen): regex const initialised with .freeze resolves ...	4 hours ago
.gitattributes	fix(test): normalize CRLF when reading .expected, force L...	last week
.gitignore	chore: ignore .exe variants of bootstrap binaries	last week
AUTHORS	Initial commit: Spinel AOT compiler project	2 months ago
LICENSE	docs: fix typo in LICENSE — Matsumomto → Matsumoto	last week
Makefile	fix: mark sp_argv strings during GC	12 hours ago
POLY-AS-SET.md	docs: stage 2 design — poly-as-set type representation	2 days ago
README.md	docs: add FFI specification + README pointer	5 days ago

About

No description, website, or topics provided.

Readme
MIT license
Activity
1.5k stars
8 watching
46 forks
Report repository

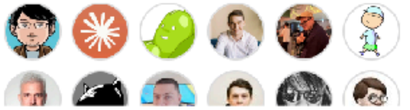
Releases

No releases published

Packages

No packages published

Contributors 25



https://github.com/matz/spinel

yuchi
@yuchi_io

DJタイムの時間に映像演出(VJ)をやったりイカ踊りの参考資料を出したりしました
楽しかったです～ありがとうございました🥳
#rubykaigi #rubyilluminations

午前1:34 · 2026年4月25日 · **1,210** 件の表示

2 26

返信をポスト 返信

もっと見つける

Xから

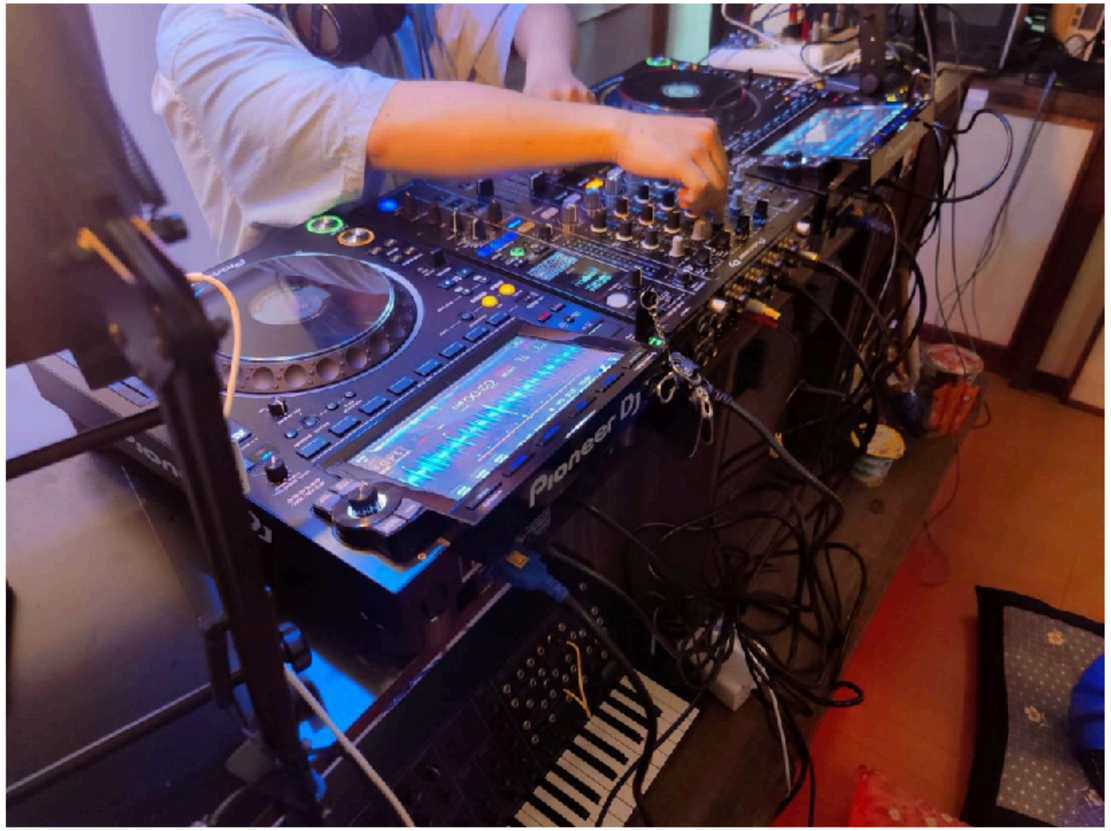
ひむら @himura4679 · 18時間
Posted to Hatena Blog
RubyKaigi 2026 に参加したぞっ！ - ひむ日記
blog.himura467.com/entry/2026/05/...
#はてなブログ #RubyKaigi_2026
#RubyKaigi

blog.himura467.comから

https://x.com/yuchi_io/status/2047715913382957494

導入

- 最近スタジオを手伝ってます

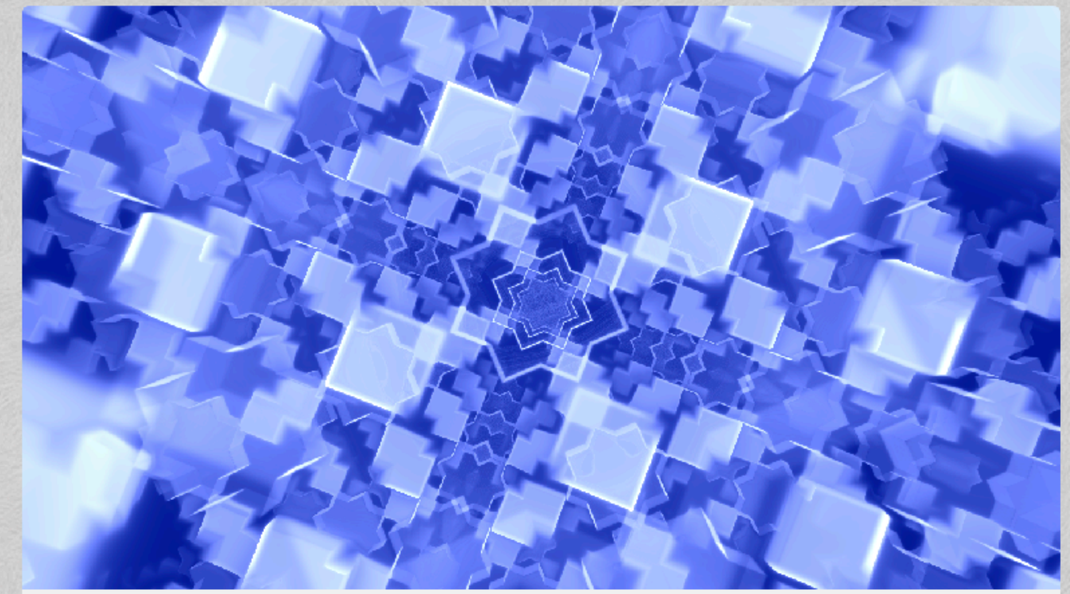


スタジオふみ

VJってやつどうやってるんですかね

shadertoyってやつをみたらええで

Shadertoy Search... Browse New Sign In



2.68 35.6 fps 1280 x 720 REC

Octagrams

512

Views: 516940, Tags: **raymarching**, **cineshader**

Created by **whisky_shusuky** in 2020-01-28

Inspired by arabesque.
https://cineshader.com/editor

Comments (18)

Sign in to post a comment.

renttec, 2026-05-06
very clean

arsenru87, 2026-02-09
cool

Image

Shader Inputs

```
precision highp float;


float gTime = 0.;
const float REPEAT = 5.0;

// 回転行列
mat2 rot(float a) {
    float c = cos(a), s = sin(a);
    return mat2(c,s,-s,c);
}

float sdBox( vec3 p, vec3 b )
{
    vec3 q = abs(p) - b;
    return length(max(q,0.0)) + min(max(q.x,max(q.y,q.z)),0.0);
}

float box(vec3 pos, float scale) {
    pos *= scale;
    float base = sdBox(pos, vec3(.4,.4,.1)) /1.5;
    pos.xy *= 5.;
    pos.y -= 3.5;
    pos.xy *= rot(.75);
    float result = -base;
    return result;
}

float box_set(vec3 pos, float iTime) {
    vec3 pos_origin = pos;
```

Compiled in 0.0 secs 1727 chars

iChannel0 iChannel1 iChannel2 iChannel3

https://www.shadertoy.com/view/tlVGDt

```
precision highp float;


float gTime = 0.;
const float REPEAT = 5.0;

// 回転行列
mat2 rot(float a) {
    float c = cos(a), s = sin(a);
    return mat2(c,s,-s,c);
}

float sdBox( vec3 p, vec3 b )
{
    vec3 q = abs(p) - b;
    return length(max(q,0.0)) + min(max(q.x,max(q.y,q.z)),0.0);
}

float box(vec3 pos, float scale) {
    pos *= scale;
    float base = sdBox(pos, vec3(.4,.4,.1)) /1.5;
    pos.xy *= 5.;
    pos.y -= 3.5;
    pos.xy *= rot(.75);
    float result = -base;
    return result;
}

float box_set(vec3 pos, float iTime) {
    vec3 pos_origin = pos;
```

← ポスト

たい
@taiseiue

Kyoto.rs #1 に参加を申し込みました！
kyotors.connpass.com/event/388753/?... #kyotors

午後1:30 · 2026年4月22日 · **206** 件の表示

1 2

返信をポスト

返信

RustをGLSLに変換して

VJできるようにしたらいいんや！

taiseiue / rs2glsl

Type / to search

Code Issues Pull requests 1 Agents Actions Projects Wiki Security and quality Insights Settings

rs2glsl Public

Pin Watch 0 Fork 0 Star 0

main 2 Branches 0 Tags Go to file T Add file Code

taiseiue エラーをトレースできるようにする 66af67f · 46 minutes ago 61 Commits

crates	エラーをトレースできるようにする	46 minutes ago
docs/reference	update docs	3 days ago
examples	pub	4 days ago
src	init project	last week
.gitignore	Initial commit	last week
Cargo.lock	エラーをトレースできるようにする	46 minutes ago
Cargo.toml	いらないクレートけした	4 days ago
LICENSE	Initial commit	last week
README.md	koyukiとかつくってみる	5 days ago

README MIT license

rs2glsl

https://github.com/taiseiue/rs2glsl

About

No description, website, or topics provided.

Readme
MIT license
Activity
0 stars
0 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 1

taiseiue Taisei Uemura

使ってみよう

```
taiseiue@lily koyuki-shader % cargo new sample_shader
    Creating binary (application) `sample_shader` package
note: see more `Cargo.toml` keys and their definitions at https://doc.rust-lang.org/cargo/reference/manifest.html
taiseiue@lily koyuki-shader % cd sample_shader
taiseiue@lily sample_shader %
```

```
taiseiue@lily koyuki-shader % cargo new sample_shader
    Creating binary (application) `sample_shader` package
note: see more `Cargo.toml` keys and their definitions at https://doc.rust-lang.org/cargo/reference/manifest.html
taiseiue@lily koyuki-shader % cd sample_shader
taiseiue@lily sample_shader % vi Cargo.toml
taiseiue@lily sample_shader % cat Cargo.toml
[package]
name = "sample_shader"
version = "0.1.0"
edition = "2024"

[dependencies]
rs2glsl-prelude = { git = "https://github.com/taiseiue/rs2glsl", package = "rs2glsl-prelude" }
rs2glsl-macros = { git = "https://github.com/taiseiue/rs2glsl", package = "rs2glsl-macros" }
rs2glsl-adapters = { git = "https://github.com/taiseiue/rs2glsl", package = "rs2glsl-adapters" }
rs2glsl-stdlib = { git = "https://github.com/taiseiue/rs2glsl", package = "rs2glsl-stdlib" }
```

```
taiseiue@lily sample_shader % cargo generate-lockfile
    Updating git repository `https://github.com/taiseiue/rs2glsl`
    Updating crates.io index
     Locking 12 packages to latest Rust 1.94.1 compatible versions
taiseiue@lily sample_shader %
```

```rust
use rs2glsl_prelude::*;
use rs2glsl_stdlib::*;
use rs2glsl_adapters::shadertoy;

fn pixel(frag_coord: Point, resolution: Point, _spectrum: Color, time: f32) -> Color {
    let uv = uv_from_frag(frag_coord, resolution);

    Color {
        r: 0.5 + 0.5 * cos(time + uv.x),
        g: 0.5 + 0.5 * cos(time + uv.y + 2.0),
        b: 0.5 + 0.5 * cos(time + uv.x + 4.0),
    }
}
```

```
taiseiue@lily rs2glsl % cargo run -p rs2glsl-cli -- /Users/taiseiue/source/github.com/taiseiue/koyuki-shader/sample_shader
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.12s
     Running `target/debug/rs2glsl-cli /Users/taiseiue/source/github.com/taiseiue/koyuki-shader/sample_shader`
const float PI = 3.14159265359;
const float TAU = 6.28318530718;

vec2 uv_from_frag(vec2 frag_coord, vec2 resolution);
vec2 aspect_uv(vec2 uv, vec2 resolution);
vec2 pixelize(vec2 uv, vec2 resolution, float size);
vec2 repeat2(vec2 p, float cell);
float hash11(float x);
float hash21(vec2 p);
vec2 hash22(vec2 p);
vec2 rotate(vec2 p, float angle);
float circle(vec2 p, float radius);
float box2(vec2 p, vec2 size);
vec3 palette(float t);
float clampf(float x, float lo, float hi);
vec3 mul_color(vec3 c, float k);
vec3 mix_color(vec3 a, vec3 b, float t);
void mainImage(out vec4 frag_color, vec2 frag_coord);
vec3 pixel(vec2 frag_coord, vec2 resolution, vec3 _spectrum, float time);

vec2 uv_from_frag(vec2 frag_coord, vec2 resolution) {
  return vec2((frag_coord.x / resolution.x), (frag_coord.y / resolution.y));
}

vec2 aspect_uv(vec2 uv, vec2 resolution) {
  return vec2(((((uv.x - 0.5)) * resolution.x) / resolution.y), (uv.y - 0.5));
}

vec2 pixelize(vec2 uv, vec2 resolution, float size) {
  return vec2(((floor(((uv.x * resolution.x) / size)) * size) / resolution.x), ((floor(((uv.y * resolution.y) / size)) * size) / re
solution.y));
```

```glsl
uniform float time;
uniform vec2 resolution;
uniform vec3 spectrum;
out vec4 frag_color;
const float PI = 3.14159265359;
const float TAU = 6.28318530718;

vec2 uv_from_frag(vec2 frag_coord, vec2 resolution);
vec2 aspect_uv(vec2 uv, vec2 resolution);
vec2 pixelize(vec2 uv, vec2 resolution, float size);
vec2 repeat2(vec2 p, float cell);
float hash11(float x);
float hash21(vec2 p);
vec2 hash22(vec2 p);
vec2 rotate(vec2 p, float angle);
float circle(vec2 p, float radius);
float box2(vec2 p, vec2 size);
vec3 palette(float t);
float clampf(float x, float lo, float hi);
vec3 mul_color(vec3 c, float k);
vec3 mix_color(vec3 a, vec3 b, float t);
void main();
vec3 pixel(vec2 frag_coord, vec2 resolution, vec3 _spectrum, float time);

vec2 uv_from_frag(vec2 frag_coord, vec2 resolution) {
  return vec2((frag_coord.x / resolution.x), (frag_coord.y / resolution.y));
}

vec2 aspect_uv(vec2 uv, vec2 resolution) {
  return vec2(((((uv.x - 0.5)) * resolution.x) / resolution.y), (uv.y - 0.5));
}
```

Image

Shader Inputs

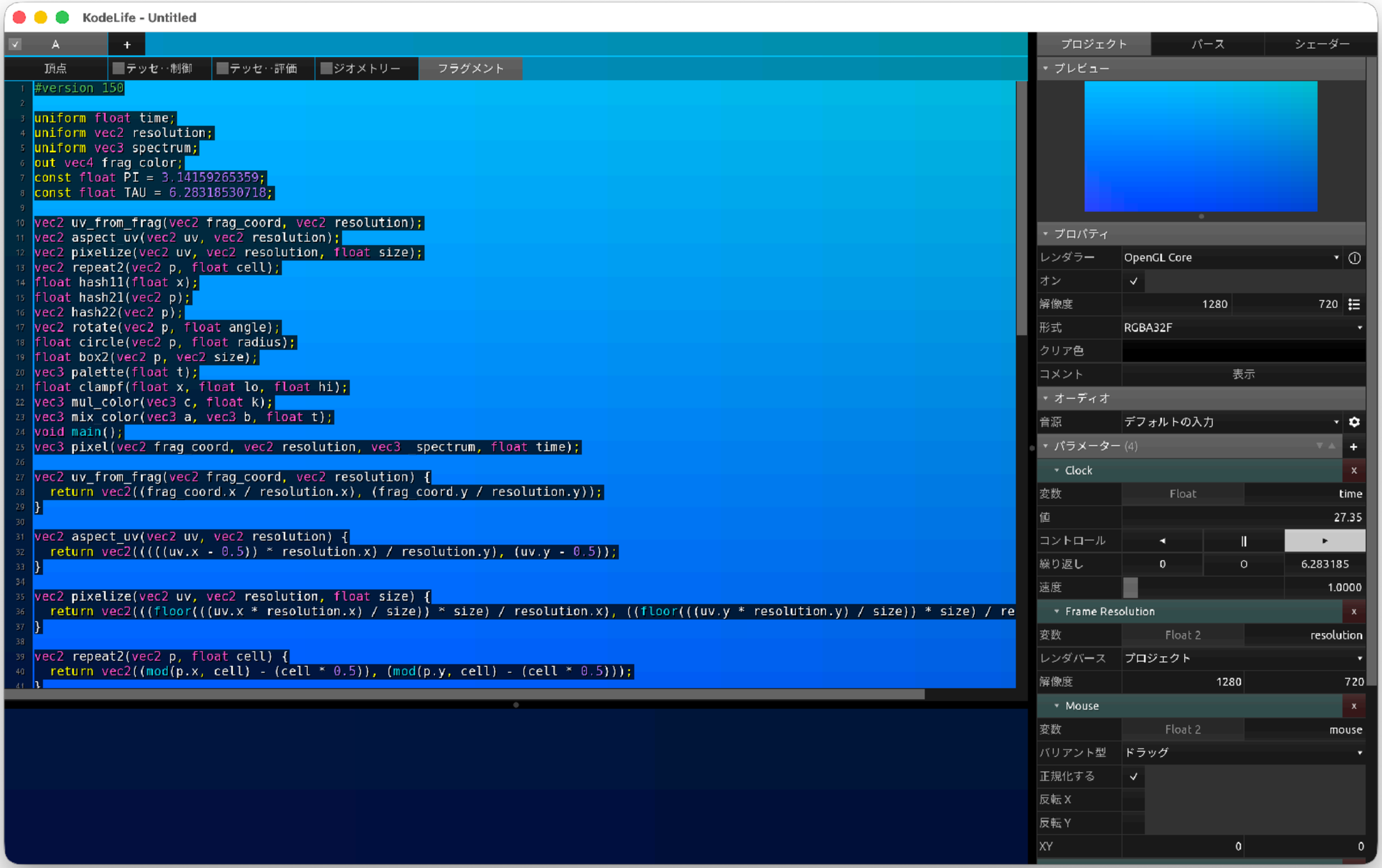
```
1  nst float PI = 3.14159265359;
2  nst float TAU = 6.28318530718;
3
4  c2 uv_from_frag(vec2 frag_coord, vec2 resolution);
5  c2 aspect_uv(vec2 uv, vec2 resolution);
6  c2 pixelize(vec2 uv, vec2 resolution, float size);
7  c2 repeat2(vec2 p, float cell);
8  at hash11(float x);
9  at hash21(vec2 p);
10 c2 hash22(vec2 p);
11 c2 rotate(vec2 p, float angle);
12 at circle(vec2 p, float radius);
13 at box2(vec2 p, vec2 size);
14 c3 palette(float t);
15 at clampf(float x, float lo, float hi);
16 c3 mul_color(vec3 c, float k);
17 c3 mix_color(vec3 a, vec3 b, float t);
18 d mainImage(out vec4 frag_color, vec2 frag_coord);
19 c3 pixel(vec2 frag_coord, vec2 resolution, vec3 _spectrum, float time);
20
21 c2 uv_from_frag(vec2 frag_coord, vec2 resolution) {
22 turn vec2((frag_coord.x / resolution.x), (frag_coord.y / resolution.y));
23
24
25 c2 aspect_uv(vec2 uv, vec2 resolution) {
26 turn vec2(((((uv.x - 0.5)) * resolution.x) / resolution.y), (uv.y - 0.5));
27
28
29 c2 pixelize(vec2 uv, vec2 resolution, float size) {
30
```

Compiled in 0.0 secs 2554 chars S ?

iChannel0 iChannel1 iChannel2 iChannel3

```rust
use rs2glsl_prelude::*;
use rs2glsl_stdlib::*;
use rs2glsl_adapters::shadertoy;

fn pixel(frag_coord: Point, resolution: Point, _spectrum: Color, time: f32) -> Color {
    let uv = uv_from_frag(frag_coord, resolution);

    Color {
        r: 0.5 + 0.5 * cos(time + uv.x),
        g: 0.5 + 0.5 * cos(time + uv.y + 2.0),
        b: 0.5 + 0.5 * cos(time + uv.x + 4.0),
    }
}
```

```rust
use rs2glsl_prelude::*;
use rs2glsl_stdlib::*;
use rs2glsl_adapters::kodelife;

fn pixel(frag_coord: Point, resolution: Point, _spectrum: Color, time: f32) -> Color {
    let uv = uv_from_frag(frag_coord, resolution);
    Color {
        r: 0.5 + 0.5 * cos(time + uv.x),
        g: 0.5 + 0.5 * cos(time + uv.y + 2.0),
        b: 0.5 + 0.5 * cos(time + uv.x + 4.0),
    }
}
```


KodeLife - Untitled
A
+
頂点
テッセ‥制御
テッセ‥評価
ジオメトリー
フラグメント
#version 150
uniform float time;
uniform vec2 resolution;
uniform vec3 spectrum;
out vec4 frag_color;
const float PI = 3.14159265359;
const float TAU = 6.28318530718;
vec2 uv_from_frag(vec2 frag_coord, vec2 resolution);
vec2 aspect_uv(vec2 uv, vec2 resolution);
vec2 pixelize(vec2 uv, vec2 resolution, float size);
vec2 repeat2(vec2 p, float cell);
float hash11(float x);
float hash21(vec2 p);
vec2 hash22(vec2 p);
vec2 rotate(vec2 p, float angle);
float circle(vec2 p, float radius);
float box2(vec2 p, vec2 size);
vec3 palette(float t);
float clampf(float x, float lo, float hi);
vec3 mul_color(vec3 c, float k);
vec3 mix_color(vec3 a, vec3 b, float t);
void main();
vec3 pixel(vec2 frag_coord, vec2 resolution, vec3 spectrum, float time);
vec2 uv_from_frag(vec2 frag_coord, vec2 resolution) {
return vec2((frag_coord.x / resolution.x), (frag_coord.y / resolution.y));
}
vec2 aspect_uv(vec2 uv, vec2 resolution) {
return vec2(((((uv.x - 0.5)) * resolution.x) / resolution.y), (uv.y - 0.5));
}
vec2 pixelize(vec2 uv, vec2 resolution, float size) {
return vec2(((floor(((uv.x * resolution.x) / size)) * size) / resolution.x), ((floor(((uv.y * resolution.y) / size)) * size) / re
}
vec2 repeat2(vec2 p, float cell) {
return vec2((mod(p.x, cell) - (cell * 0.5)), (mod(p.y, cell) - (cell * 0.5)));
}
プロジェクト
パース
シェーダー
プレビュー
プロパティ
レンダラー
OpenGL Core
オン
解像度
1280
720
形式
RGBA32F
クリア色
コメント
表示
オーディオ
音源
デフォルトの入力
パラメーター (4)
Clock
変数
Float
time
値
27.35
コントロール
繰り返し
0
0
6.283185
速度
1.0000
Frame Resolution
変数
Float 2
resolution
レンダパース
プロジェクト
解像度
1280
720
Mouse
変数
Float 2
mouse
バリアント型
ドラッグ
正規化する
反転 X
反転 Y
XY
0
0

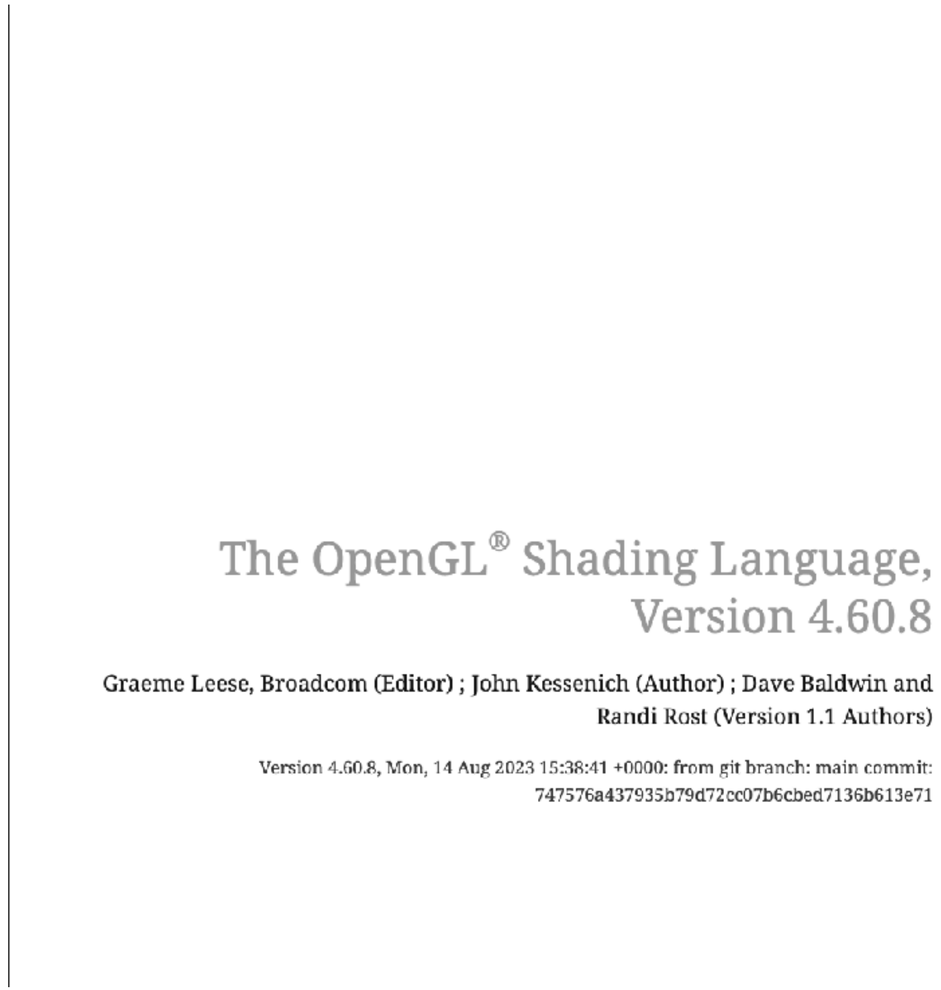
中身の紹介

rs2glsl

- RustのサブセットをGLSLにトランスパイルするトランスパイラ
- モダンな言語で書ける
- 複数の環境に応じたGLSLを出力できる

GLSL: OpenGL Shading Language

- 誤解を恐れずに言えばGPU上で動くプログラムを書けるプログラミング言語/API
- CPUで動くプログラムとやや世界線が違う


The OpenGL® Shading Language, Version 4.60.8

Graeme Leese, Broadcom (Editor) ; John Kessenich (Author) ; Dave Baldwin and Randi Rost (Version 1.1 Authors)

Version 4.60.8, Mon, 14 Aug 2023 15:38:41 +0000: from git branch: main commit: 747576a437935b79d72cc07b6cbed7136b613e71

C: Hello, World

GLSL: SampleCode

```
1  void mainImage( out vec4 fragColor, in vec2 fragCoord )
2  {
3      // Normalized pixel coordinates (from 0 to 1)
4      vec2 uv = fragCoord/iResolution.xy;
5
6      // Time varying pixel color
7      vec3 col = 0.5 + 0.5*cos(iTime+uv.xyx+vec3(0,2,4));
8
9      // Output to screen
10     fragColor = vec4(col,1.0);
11 }
```

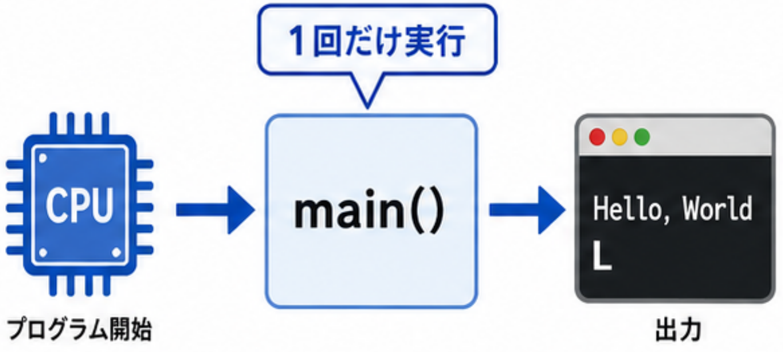
C言語の main と GLSL の mainImage の違い

Cはプログラム全体で1回、GLSLはピクセルごとに何度も並列実行される

❶ C言語

main はプログラム開始時に1回だけ呼ばれる

```c
#include<stdio.h>

int main(){
  printf("Hello, World");
  return 0;
}
```

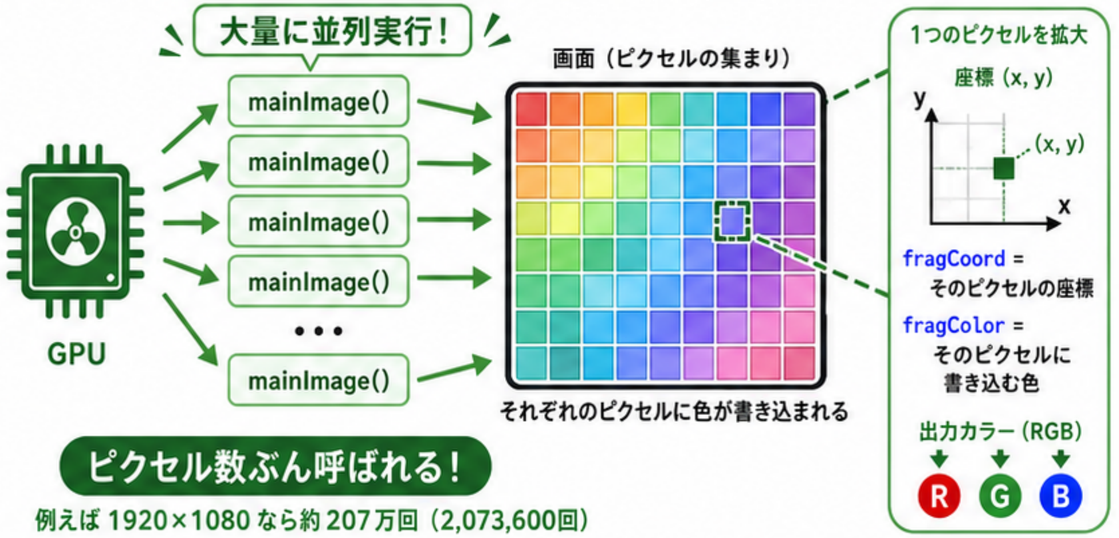


1回だけ実行（直列的）

❷ GLSL / Shader

mainImage は画面の各ピクセルごとに呼ばれる

```glsl
void mainImage( out vec4 fragColor, in vec2 fragCoord )
{
    vec2 uv = fragCoord/iResolution.xy;
    vec3 col = 0.5 + 0.5*cos(iTime+uv.xyx+vec3(0,2,4));
    fragColor = vec4(col,1.0);
}
```



ピクセル数ぶん呼ばれる！

例えば 1920×1080 なら約 207万回（2,073,600回）

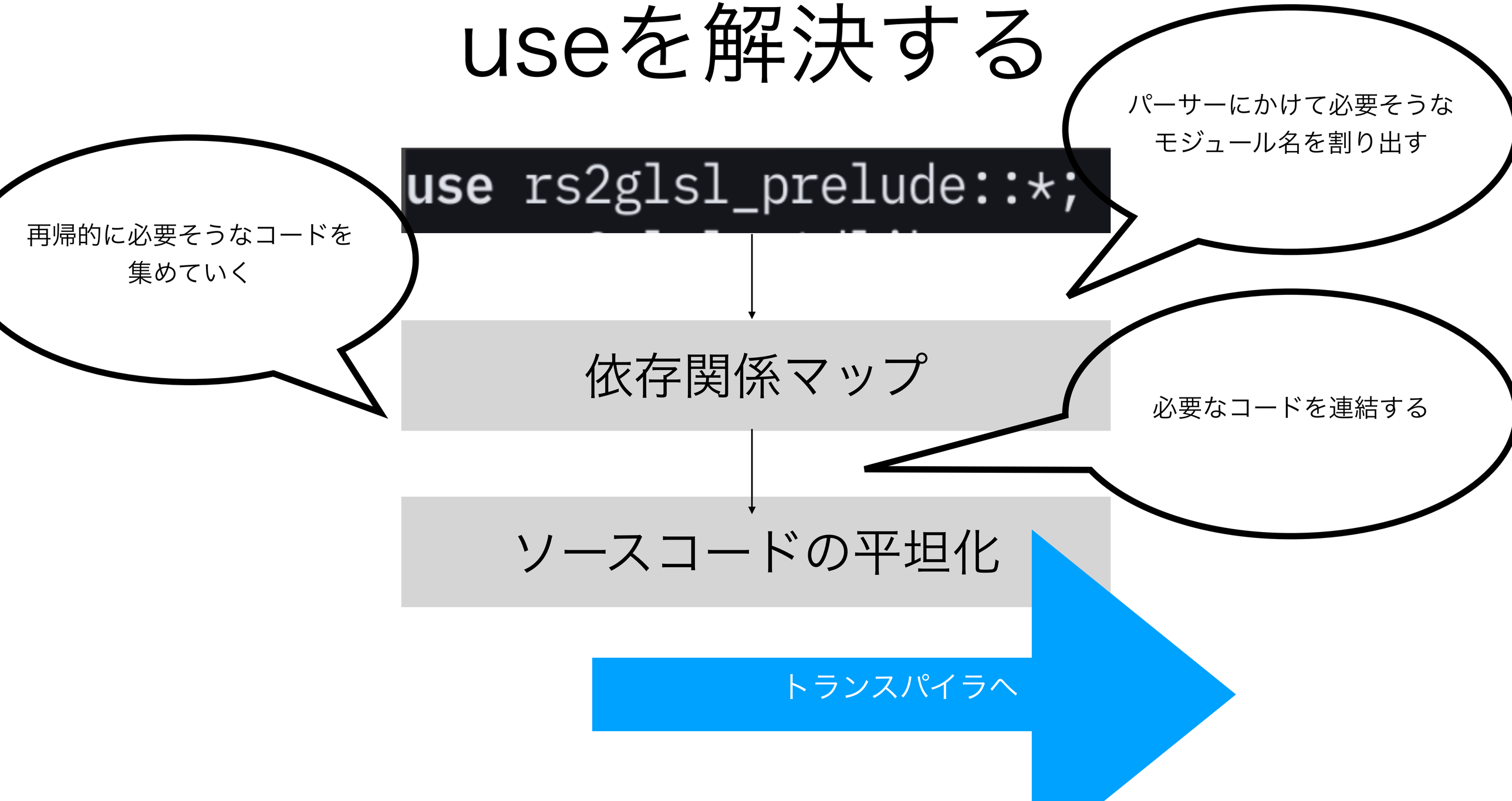
まとめ（比較）	C言語：1つのプログラムの流れを1回進める	1回だけ・直列的
まとめ（比較）	GLSL：画面上のたくさんの点を同時に計算する	大量に並列実行

```rust
use rs2glsl_prelude::*;
use rs2glsl_stdlib::*;
use rs2glsl_adapters::shadertoy;

fn pixel(frag_coord: Point, resolution: Point, _spectrum: Color, time: f32) -> Color {
    let uv = uv_from_frag(frag_coord, resolution);

    Color {
        r: 0.5 + 0.5 * cos(time + uv.x),
        g: 0.5 + 0.5 * cos(time + uv.y + 2.0),
        b: 0.5 + 0.5 * cos(time + uv.x + 4.0),
    }
}
```

useを解決する

- GLSLにはモジュールの概念がない
- 複数ファイル・別クレートの参照はトランスパイル前に連結される
  - たとえばlib.rsを参照したらmain.rs+lib.rsがトランスパイラに渡る

useを解決する


use rs2glsl_prelude::*;
パーサーにかけて必要そうな
モジュール名を割り出す
再帰的に必要そうなコードを
集めていく
依存関係マップ
必要なコードを連結する
ソースコードの平坦化
トランスパイラへ

```
use rs2glsl_prelude::*;
use rs2glsl_stdlib::*;
use rs2glsl_adapters::shadertoy;

fn pixel(frag_coord: Point, resolution: Point, _spectrum: Color, time: f32) -> Color {
    let uv = uv_from_frag(frag_coord, resolution);

    Color {
        r: 0.5 + 0.5 * cos(time + uv.x),
        g: 0.5 + 0.5 * cos(time + uv.y + 2.0),
        b: 0.5 + 0.5 * cos(time + uv.x + 4.0),
    }
}
```

トランスパイルしよう

- 構造体などの定義表を作る
- 式を見ながら等価なGLSLを出力する
- 構造体やintなどをマーシャリングする

おもろポイント: 構造体はベクトルに化ける

```
Color {
    r: 0.5 + 0.5 * cos(time + uv.x),
    g: 0.5 + 0.5 * cos(time + uv.y + 2.0),
    b: 0.5 + 0.5 * cos(time + uv.x + 4.0),
}
```

おもろポイント: 構造体はベクトルに化ける

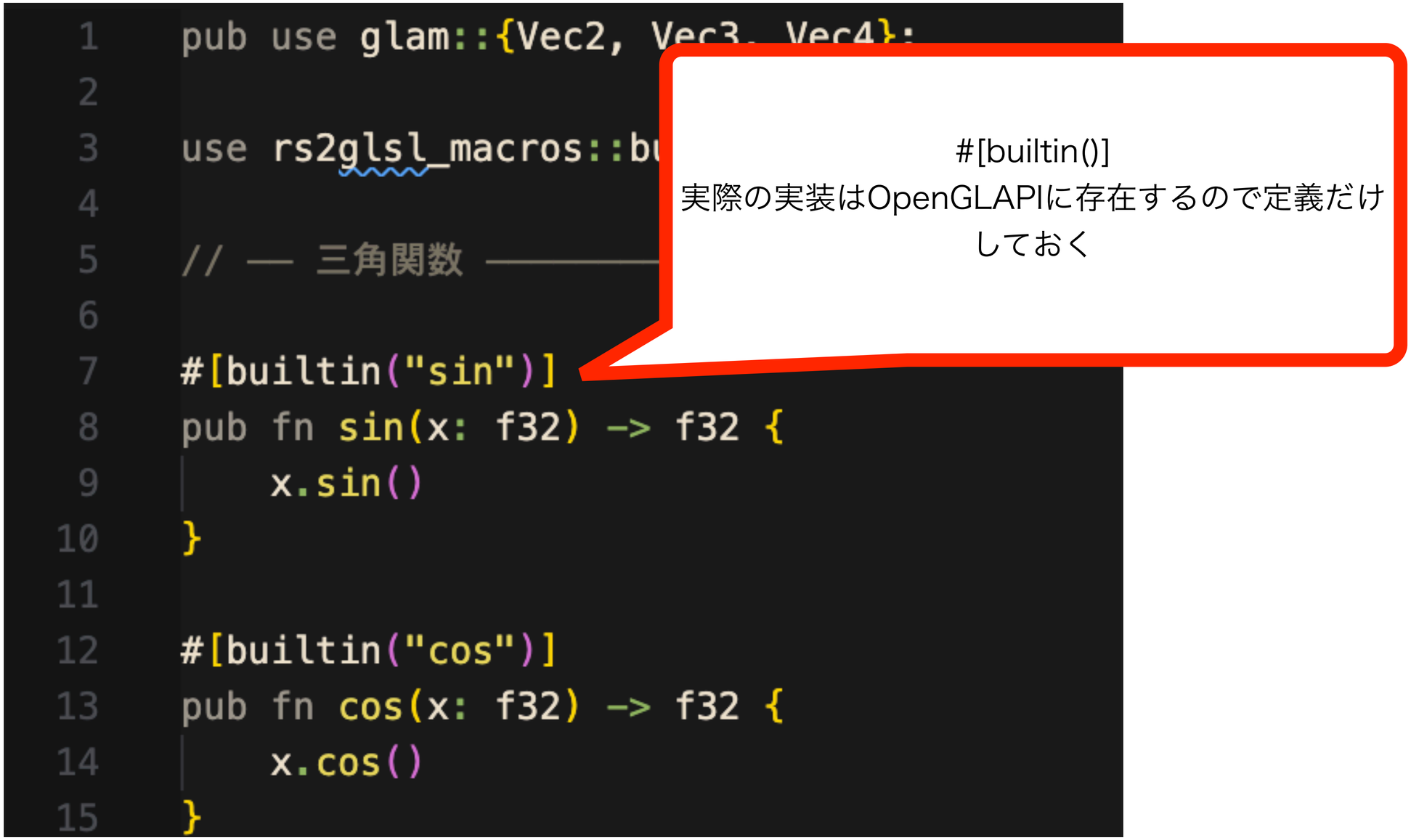
```
#[structlayout(vec3)]
pub struct Color {
    r: f32,
    g: f32,
    b: f32,
}
```

おもろポイント: 構造体はベクトルに化ける

```
return vec3(
        (0.5 + (0.5 * cos((time + uv.x)))),
        (0.5 + (0.5 * cos(((time + uv.y) + 2.0)))),
        (0.5 + (0.5 * cos(((time + uv.x) + 4.0))))
);
```

```
use rs2glsl_prelude::*;
use rs2glsl_stdlib::*;
use rs2glsl_adapters::shadertoy;
```

```rust
pub use glam::{Vec2, Vec3, Vec4};

use rs2glsl_macros::builtin;

// ── 三角関数 ──────────────────────────────────────────

#[builtin("sin")]
pub fn sin(x: f32) -> f32 {
    x.sin()
}

#[builtin("cos")]
pub fn cos(x: f32) -> f32 {
    x.cos()
}
```


pub use glam::{Vec2, Vec3, Vec4};
use rs2glsl_macros::bu
// ── 三角関数 ──
#[builtin("sin")]
pub fn sin(x: f32) -> f32 {
x.sin()
}
#[builtin("cos")]
pub fn cos(x: f32) -> f32 {
x.cos()
}
#[builtin()]
実際の実装はOpenGLAPIに存在するので定義だけしておく

```
use rs2glsl_prelude::*;
use rs2glsl_stdlib::*;
use rs2glsl_adapters::shadertoy;
```

```rust
use rs2glsl_macros::structlayout;
use rs2glsl_prelude::*;

pub const PI: f32 = 3.14159265359;
pub const TAU: f32 = 6.28318530718;

#[structlayout(vec2)]
pub struct Point {
    x: f32,
    y: f32,
}

#[structlayout(vec3)]
pub struct Color {
    r: f32,
    g: f32,
    b: f32,
}

impl Add for Point {
    type Output = Point;

    fn add(self, rhs: Point) -> Point {
        Point {
            x: self.x + rhs.x,
            y: self.y + rhs.y,
        }
    }
}

impl Sub for Point {
    type Output = Point;

    fn sub(self, rhs: Point) -> Point {
        Point {
            x: self.x - rhs.x,
            y: self.y - rhs.y,
        }
    }
}
```

```
use rs2glsl_prelude::*;
use rs2glsl_stdlib::*;
use rs2glsl_adapters::shadertoy;
```

adapter

- GLSLはホストアプリケーションによってエントリポイントが違う
- ホストアプリケーションによって渡されるパラメータも違う

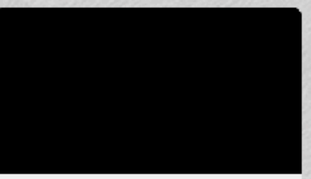
Browse

Image

Shader Inputs

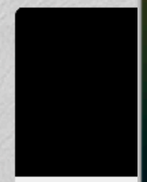
```
void mainImage( out vec4 fragColor, in vec2 fragCoord )
{
    // Normalized pixel coordinates (from 0 to 1)
    vec2 uv = fragCoord/iResolution.xy;

    // Time varying pixel color
    vec3 col = 0.5 + 0.5*cos(iTime+uv.xyx+vec3(0,2,4));

    // Output to screen
    fragColor = vec4(col,1.0);
}
```

Compiled in 0.0 secs 158 chars


iChannel0


iChannel1


iChannel2


iChannel3

Apps and Plugins Tutorials

A +

頂点 テッセ··制御 テッセ··評価 ジオメトリー フラグメント

```
#version 150

uniform float time;
uniform vec2 resolution;
uniform vec2 mouse;
uniform vec3 spectrum;

uniform sampler2D texture0;
uniform sampler2D texture1;
uniform sampler2D texture2;
uniform sampler2D texture3;
uniform sampler2D prevFrame;
uniform sampler2D prevPass;

in VertexData
{
    vec4 v_position;
    vec3 v_normal;
    vec2 v_texcoord;
} inData;

out vec4 fragColor;

void main(void)
{
    vec2 uv = -1. + 2. * inData.v_texcoord;
    fragColor = vec4(
        abs(sin(cos(time+3.*uv.y)*2.*uv.x+time)),
        abs(cos(sin(time+2.*uv.x)*3.*uv.y+time)),
        spectrum.x * 100.,
        1.0);
}
```

adapter

- ホストアプリ間の差異を埋めてくれるくん
- ホスト独自のエントリポイント→pixel関数を呼んでくれる
- 使っても使わなくてもいい(ホスト固有の変数や関数も呼べる

時間あったら
トランスパイルして動かす

モダンな言語

RustでもVJしたい！

kyoto.rs#1

2026/5/9 / taiseiue