Astra

Fable 5.1

Opus 5.5

Astra と Fable 5.1 と Opus 5.5 で放置ゲムワを 2 日で作った話

Potion Craft Idle (VRChat) ができるまで

個人開発集会 ／ おぷん ／ 2026-10-22

自己紹介

- **おぷん** (X: **x.com/opteyo**)
- 窓際情シス。最近はゲームのバイブコーディング
- VRChat のゲームワールドはこれが **2 作目**
- 最近のテーマ: ファンコミュニティづくり

今日の話

- 放置ゲーのワールドを **2 日** で作った
- 3 つのモデルの使い分け
- 公開後に起きたこと

作ったもの: Potion Craft Idle

ポーション自動生産 × 放置 × 属性反応

- 素材を集めて、ポーションを作って売る
- ポーションを投げて敵を倒す (油 + 火 = 爆発)
- 設備で自動化して、最後は全部が勝手に回る
- 1 人 1 工場。卓に座れば協力プレイ



錬金術工房のホール (Unity のシーンビュー)

遊び方 ワールド内のポスター (これも voxel スキルの絵)



座る → 集める → 作る → 売る → 自動化 → 戦う



付着 + 別の属性 = 反応 (爆発・凍結・腐食・感電・破砕)

2 日の内訳 git のコミット時刻から



3 つのモデルの役割分担

Astra (Codex)

voxel 生成スキルを作らせた

- アバターもワールドも敵も、このスキルで

Fable 5.1

VRChat 化の移植を一気に

- 最初のコミット (10/2) と公開日を担当

Opus 5.5

日々の実装の主力

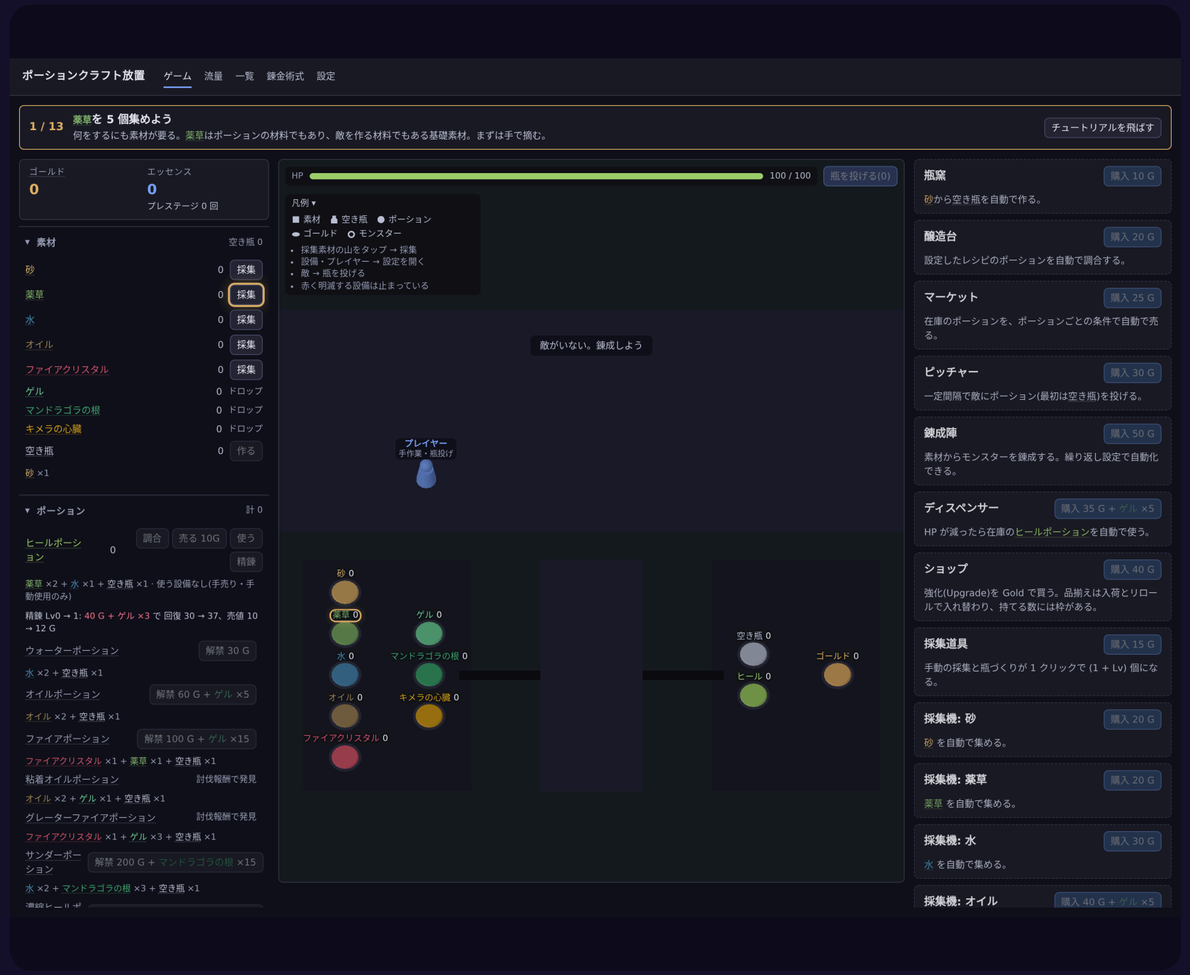
- VRChat 版 289 コミット中 212
- worktree で並列に

人がやったこと: 遊んで判断する・Unity でボタンを押す・プレイヤーの話を聞く

STEP 1 まず Web で面白さを検証する

- TypeScript + three.js を Cloudflare Pages に
- VRChat に呼ぶのは重い。**ブラウザなら URL を送るだけ**
- ルールは依存ゼロの純 TS エンジン
- CI で自動プレイして、バランスを機械で確かめる

9/29 に最初のコミット → 9/30 に公開



Web 版 potion-craft-idle.wbelucky.dev

ゲームの核になる錬金術式

- 設備の動きを書く、Rust 風の小言語
- ループ無し。**5,000 命令**で必ず止まる
- プリセット → ブロック → コードの 3 段

```
// Oil を当てた直後、
// 同じ敵へ Fire を重ねる
match event {
    Thrown { by: Pitcher1, target, ammo: Oil }
        if target.alive && potions.fire > 0
        => Throw(Fire, target),
    _ => Wait,
}
```

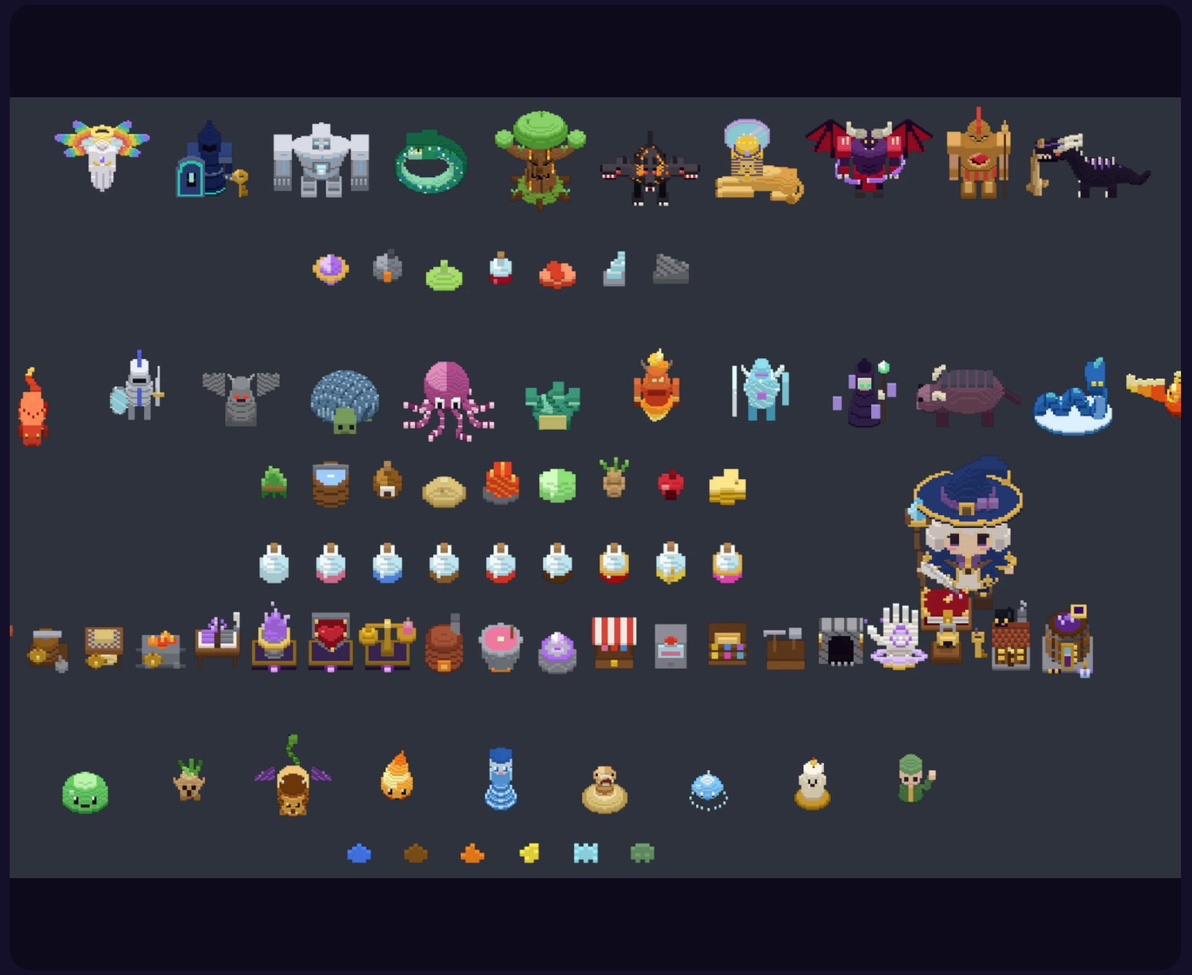
STEP 2 voxel 生成スキルで全部作る

スキル `voxel-world-art` (Astra で作成)

3 面図
前・横・後ろを描く
→
人が確認
OK まで作らない
→
voxel
Python で形を組む
→
書き出し
Blender → Unity

- 人が OK するまで 3D を作らない
- 一覧画像を見て採否を決める

171 モデル・約 11.2 万三角形



build_models.py --preview の一覧

STEP 3

昔のワールドを Fable に食わせて VRChat ワールドにする

入力: Web 版 + voxel スキル + 前作 ForbiddenPrefab

ルール: **基準は TS。Unity 側で数値を変えない**

TS の台本を C# で再生して一致を確かめる

TS (基準)
自動プレイの操作を台本に

→

C# (移植)
同じ操作を同じ順に流す

→

比較
Gold・在庫・HP・乱数が 1 つでも違えば失敗

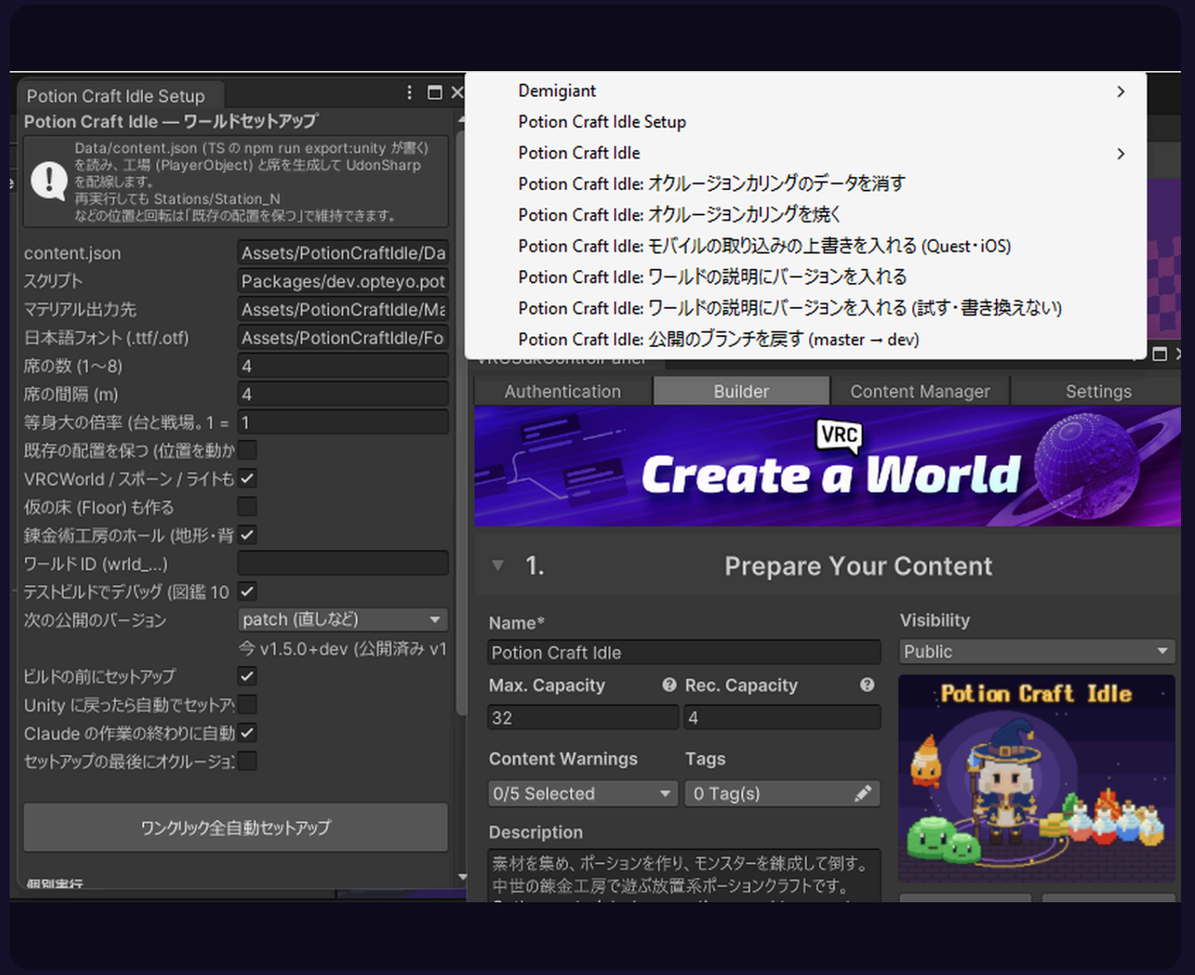
7,928
ステップ (1 時間の台本)

47,725
ステップ (4 時間の台本)

250
錬金術式のテスト

Unity 拡張で、シーンの中身を全部自動配置

- メニュー 1 回でシーンを生成
- 部屋・卓・看板・**Udon の配線**まで全部コード
- 人は **Unity でボタンを押すだけ**



セットアップ画面と SDK のパネル

Unity を開かずにエージェントが自分で検証

typecheck.sh

Unity 同梱の Roslyn で、SDK の実 DLL に対してコンパイル

ハーネス

Udon のスクリプトを普通の .NET で動かす

1. C# を書く

2. 型検査

3. ハーネスで TS と照合

4. 人が Unity で Build & Test

ここだけ人

UdonSharp の罠は CLAUDE.md に貯める

- List<T> も LINQ も使えない
- int は **21 億で負になる**
- 例外は捕まえられず、部品ごと止まる

やり方

ハマるたびに **CLAUDE.md** に 1 行書く

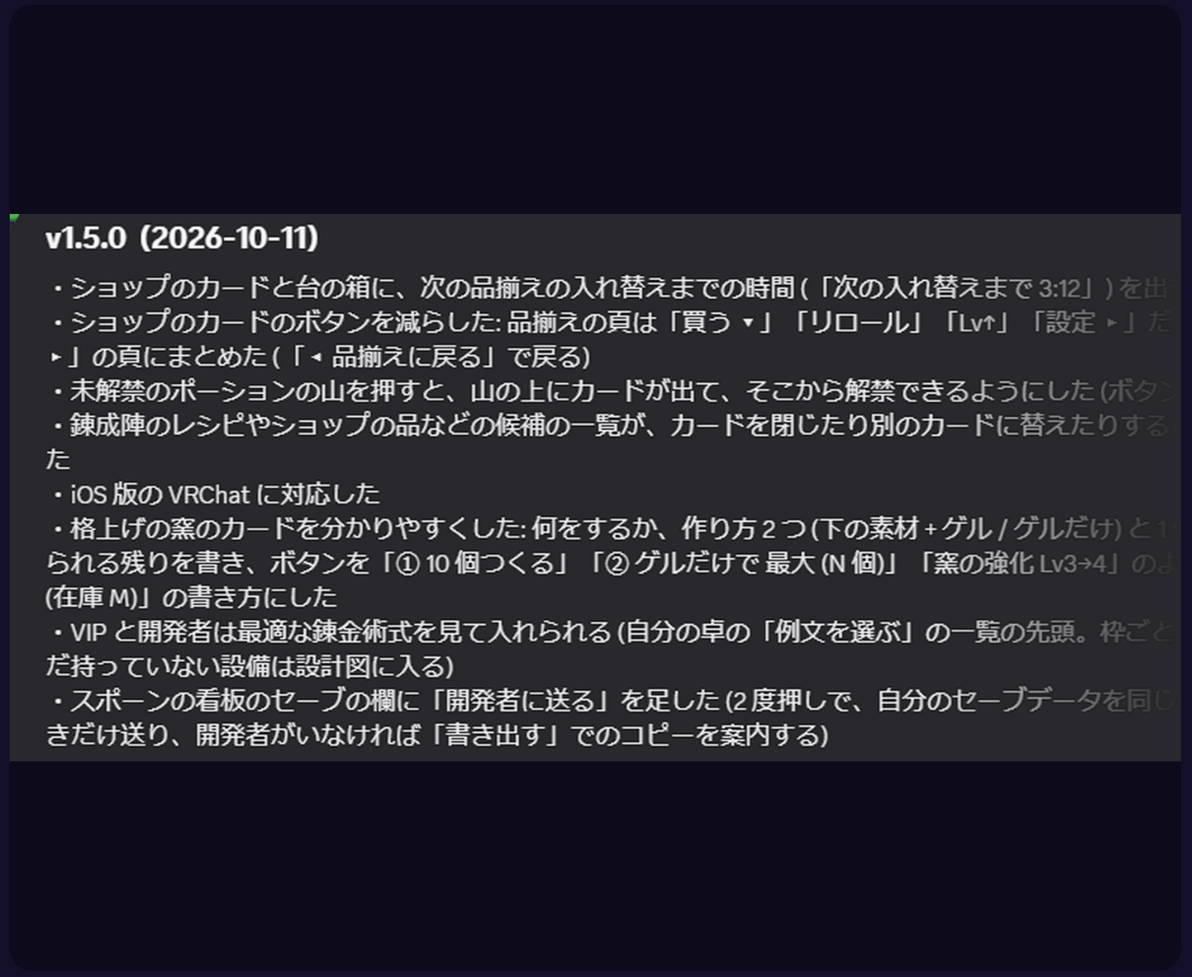
次からエージェントは踏まない

公開も「押すだけ」にする

Build & Publish を押すと

1. 版と日付を付ける
2. 全機種にアップロード
3. git tag を打つ
4. Discord・ワールドの説明・看板を更新

4 日で 18 タグ、公開日だけで 8 回



Discord に自動で投稿された更新履歴

重さとの戦い: Udon は C# の数百倍遅い

- 重い処理を **1 フレーム 1 部分**に分ける
- 遠い卓は止める。揺れは Animator に任せる
- 同期の送信量に予算を決める

1 フレームの Udon (v1.4.4)

24〜45 ms

↓

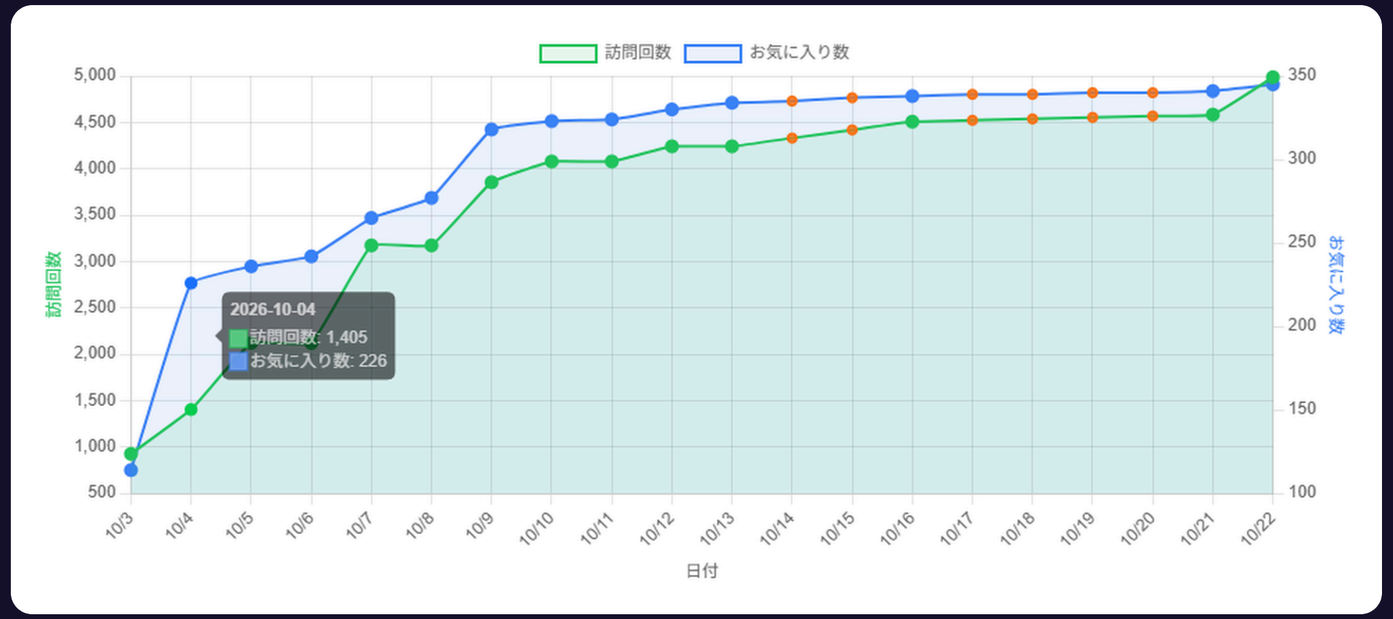
約 3 ms

公開 15 時間後: いつの間にかラボ抜け

約 1,500
Visits

約 150
いいね

- 宣伝: 何をしたか
- 今 (10/22) は約 5,000 訪問



訪問回数とお気に入り数の推移

1 日後: 別の集会でヘビープレイヤーに会う

- アバターの集会で、**プレイ 10 時間超**の人に会う
- 不具合報告と要望が無限に出る
- 21 億で負になる、25 日放置で止まる …
- **徹夜で実装**。1 時間で 2 回の hotfix



撮影スポットでヘビープレイヤーと

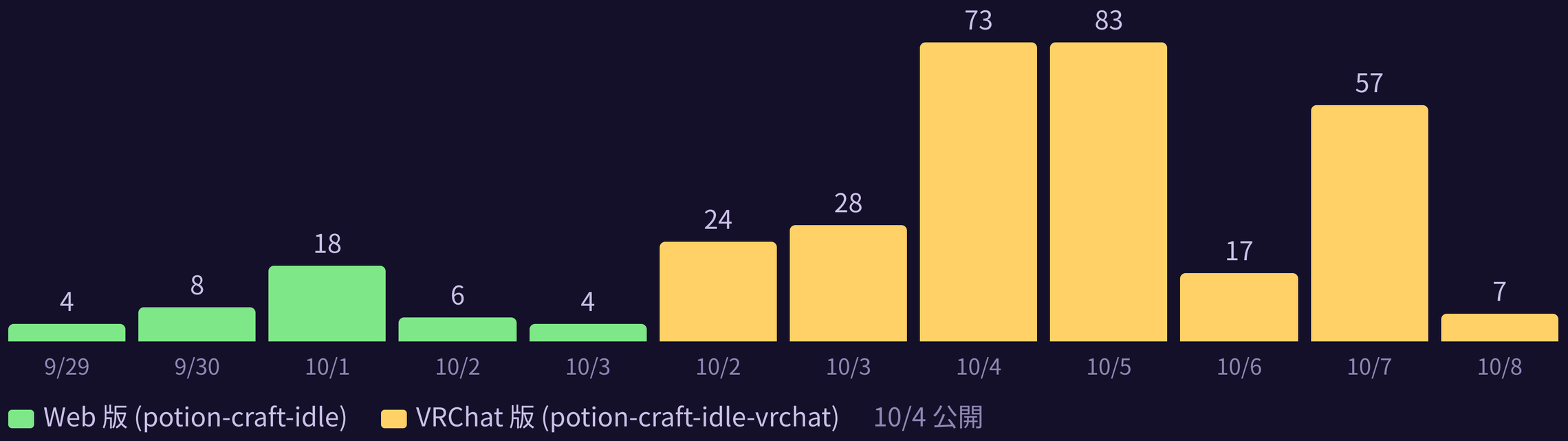
数字で振り返る

329
コミット

53k
行 (TS + C#)

232
Claude との共同コミット

1 日あたりのコミット数



学んだこと

1. **基準を 1 つ**に決めて自動で照合すれば、移植は任せられる
2. エージェントが**自分で回せるチェック**が速さの源
3. 面白さは Web で検証。でも**人は VRChat にいる**

遊びに来てください

VRChat: **Potion Craft Idle** ／ X: x.com/opteyo

ワールド

Discord