



HALF A DECADE OF RUST IN MOBILE GAMES

BY STEPHAN DILLY

WHO AM I?

- Stephan Dilly
- Worked for Ubisoft, InnoGames, Kolibri, GitButler & Kraken
- Founded Gameroasters (sold to CRATR.games)
- Founded [Rustunit.com](https://rustunit.com)
- Created [GitUI](https://gitui.org)
- Organiser of the [Beverly Meetup](https://bevy.dev/meetup)

GOALS FOR TODAY

- How did using Rust changed over time
- Some war stories
- Where are we today?
- What does the future hold?

OUTLINE

- Why Rust?
- Shared Rust Core Library
- Pure Rust Backends
- Hybrid Clients 2.0
- Modular Gamebackend
- Fullstack Rust Mobile Games



WHY RUST?

DREAM OF FULLSTACK

- Share critical logic with the backend
- Performance critical AI calculation in the Client
- Easily exposable via c-ffi
- No GC
- “If it builds it works”
- Dare to dream of Fullstack Mobile Dev

SHARED RUST CORE LIBRARY

GO BACKEND MEETS RUST

- calling C from go is horrible
- stringified json c-ffi

SHARED RUST CORE LIBRARY

GO C-FFI

The screenshot shows a dark-themed code editor interface, likely Visual Studio Code, displaying two files related to a shared Rust core library:

- cs4logic.h**: A C header file located at `backend > s4api > logic > inc > cs4logic.h`. It contains declarations for various functions and types, including `RatingDuel`, `get_time`, `get_ai_move`, `put_height`, `is_over`, `Rating ranking_new`, and `RatingDuel ranking_duel`.
- logic.go**: A Go source file located at `backend > s4api > logic > logic.go`. It includes the C header via `#include "inc/cs4logic.h"`. The code uses the `cgo` directive to link against the C library. It imports the C standard library (`<stdlib.h>`) and defines a function `GetAiMove` that takes parameters for player turn, other player, max depth, and state, and returns an integer value.

The left sidebar shows a file tree with the following structure:

- backend**
 - `goforward`
 - `pubsub`
 - s4api**
 - `.vscode`
 - api** (selected)
 - `bot`
 - `data`
 - `lambda`
 - logic**
 - inc**
 - cs4logic.h**
 - `lib`
 - `logic.go`
 - `types.go`
 - `logs`
 - `mock_api`
 - `mock_redis`
 - `mock_utils`
 - `push`
 - `rest`
 - `splunk`
 - `utils`
 - `validation`
 - `.gitignore`
 - `.htpasswd`
 - `AuthKey_67PU746K7W.p8`

SHARED RUST CORE LIBRARY

UNITY CLIENT MEETS RUST

- c-ffi is not too bad from C#
- cross compilation for iOS & Android rocks
- iteration is slow in Unity (reopen editor)
- stringified json c-ffi



SHARED RUST CORE LIBRARY

UNITY CLIENT MEETS RUST

```
client > unity > stack4next > Assets > Stack4 > Plugins > C# Bridge.cs

1  using System;
2  using System.Runtime.InteropServices;
3
4  namespace S4
5  {
6      static public class LogicNative
7      {
8          [Serializable]
9          public struct Move
10         {
11             public int player;
12             public int column;
13         }
14
15         [Serializable]
16         public struct FieldDefinition
17         {
18             public int height;
19             public int width;
20             public bool wrap;
21         }
22
23         [Serializable]
24         public struct FieldState
25         {
26             public FieldDefinition field;
27             public Move[] moves;
28         }
29     }
30
31     #if UNITY_EDITOR || UNITY_ANDROID
32     private const string DllName = "libcs4logic";
33     #else
34     private const string DllName = "__Internal";
35     #endif
36
37     [DllImport(DllName)]
38     public static extern int put_height(int column, string
39
40     [DllImport(DllName)]
41     public static extern int get_ai_move(int playerAtTurn,
42
43     [DllImport(DllName)]
44     public static extern string is_over(string data);
45
46     [DllImport(DllName)]
47     public static extern long get_time();
48 }
```

SHARED RUST CORE LIBRARY

CONCLUSION

- verbose and slow data passing across boundaries
- go c-ffi is slow
- hard to maintain
- go in the backend viable but adds another tech
- go error handling and package manager sucked at the time
- Slow Unity / C# json parsing

PURE RUST BACKEND

WHY NOW?

- Async Await stabilised
- More and more driver support for Rust (MongoDB)
- Client Project Opportunity to use Rust
- Warp + CosmosDB (via MongoDB)

PURE RUST BACKEND

WHAT COULD GO WRONG?

- Server hit by 1k players per second
- Crashed every few minutes
- Official MongoDB driver bug
- Connection Pool handling was broken



PURE RUST BACKEND

MONGODB DRIVER BUG

Decrement connection count when connection is dropped due to...

Brow

JG Decrement connection count when connection is dropped due to...

Brow

finished operation

main

v2.8.2 ... v1.--ignore

Stephan Dilly authored and saghm committed on Aug 19, 2020 1 parent ffa0b66 commit 4d4e007

Showing 2 changed files with 33 additions and 21 deletions.

Whitespace Ignore whitespace Split Unified

Filter changed files

> 46 src/cmap/conn/mod.rs

src/cmap conn mod.rs

... @@ -300,6 +300,14 @@ impl ConnectionPoolInner {
 300 300 }
 301 301 }
 302 302
 303 + async fn dropped(&self, conn: Connection) {
 304 + let mut connection_manager = self.connection_manager.lock().await;
 305 +
 306 + connection_manager.close_connection(conn, ConnectionClosedReason::Dropped);
 307 +
 308 + self.wait_queue.wake_front();
 309 + }
 310 +
 303 311 async fn check_in(&self, mut conn: Connection) {
 304 312 self.emit_event(|handler| {
 305 313 handler.handle_connection_checked_in_event(conn.checked_in_event());
 306 314 }
 307 315 }
 308 316 }
 309 317 }
 310 318 }
 311 319 }
 312 320 }
 313 321 }
 314 322 }
 315 323 }
 316 324 }
 317 325 }
 318 326 }
 319 327 }
 320 328 }
 321 329 }
 322 330 }
 323 331 }
 324 332 }
 325 333 }
 326 334 }
 327 335 }
 328 336 }
 329 337 }
 330 338 }
 331 339 }
 332 340 }
 333 341 }
 334 342 }
 335 343 }
 336 344 }
 337 345 }
 338 346 }
 339 347 }
 340 348 }
 341 349 }
 342 350 }
 343 351 }
 344 352 }
 345 353 }
 346 354 }
 347 355 }
 348 356 }
 349 357 }
 350 358 }
 351 359 }
 352 360 }
 353 361 }
 354 362 }
 355 363 }
 356 364 }
 357 365 }
 358 366 }
 359 367 }
 360 368 }
 361 369 }
 362 370 }
 363 371 }
 364 372 }
 365 373 }
 366 374 }
 367 375 }
 368 376 }
 369 377 }
 370 378 }
 371 379 }
 372 380 }
 373 381 }
 374 382 }
 375 383 }
 376 384 }
 377 385 }
 378 386 }
 379 387 }
 380 388 }
 381 389 }
 382 390 }
 383 391 }
 384 392 }
 385 393 }
 386 394 }
 387 395 }
 388 396 }
 389 397 }
 390 398 }
 391 399 }
 392 400 }
 393 401 }
 394 402 }
 395 403 }
 396 404 }
 397 405 }
 398 406 }
 399 407 }
 400 408 }
 401 409 }
 402 410 }
 403 411 }
 404 412 }
 405 413 }
 406 414 }
 407 415 }
 408 416 }
 409 417 }
 410 418 }
 411 419 }
 412 420 }
 413 421 }
 414 422 }
 415 423 }
 416 424 }
 417 425 }
 418 426 }
 419 427 }
 420 428 }
 421 429 }
 422 430 }
 423 431 }
 424 432 }
 425 433 }
 426 434 }
 427 435 }
 428 436 }
 429 437 }
 430 438 }
 431 439 }
 432 440 }
 433 441 }
 434 442 }
 435 443 }
 436 444 }
 437 445 }
 438 446 }
 439 447 }
 440 448 }
 441 449 }
 442 450 }
 443 451 }
 444 452 }
 445 453 }
 446 454 }
 447 455 }
 448 456 }
 449 457 }
 450 458 }
 451 459 }
 452 460 }

PURE RUST BACKEND

USAGE IN WHEELIE ROYALE

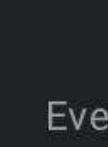
Wheelie Royale

Gameroasters GmbH

Contains ads · In-app purchases



10K+
Downloads



Everyone ⓘ

Install

Share

Add to wishlist

You don't have any devices

24.1 m
My Best: 109.6 m
3.409

Pos 100 / 100
8.4 m
My Best: 245.8 m
3.533

573.1 m
Pos 3 / 100
+79



App suppo

HYBRID CLIENTS 2.0

PROTOBUF TO THE RESCUE

- c-ffi down to a single function
- protobuf instead instead of json
- code generation for Rust and C#
- single memory buffer, no more GC hick-ups
- simple code sharing with backend

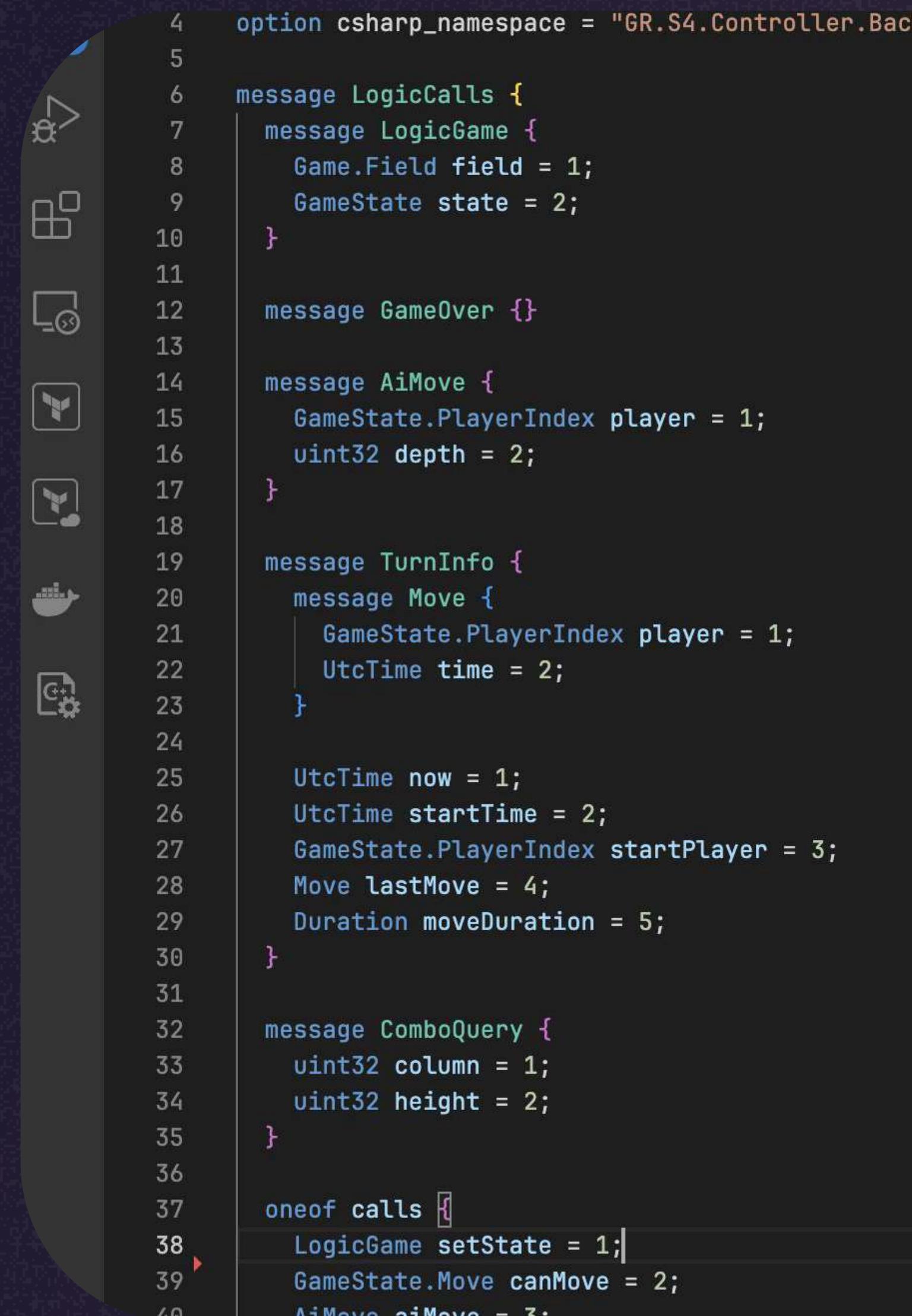
HYBRID CLIENTS 2.0

CODE #1

```
Unity > stack4 > Assets > _stack4 > Scripts > Native > C# Bridge.cs > ...
5  using GR.S4.Controller.Backend;
6  using UnityEngine;
7
8  namespace GR.S4.Native
9  {
10     2 references
11     static class Native
12     {
13         #if UNITY_EDITOR || UNITY_ANDROID
14             private const string DllName = "libs4clogic";
15         #else
16             2 references
17             private const string DllName = "__Internal";
18         #endif
19
20         [DllImport(DllName)]
21         1 reference
22         public static extern long get_time();
23
24         [DllImport(DllName)]
25         1 reference
26         public static extern bool logic_call(
27             byte[] buffer, long bufferSize, long inputSize, ref long outputSize);
28     }
29
30 }
```

HYBRID CLIENTS 2.0

CODE #2



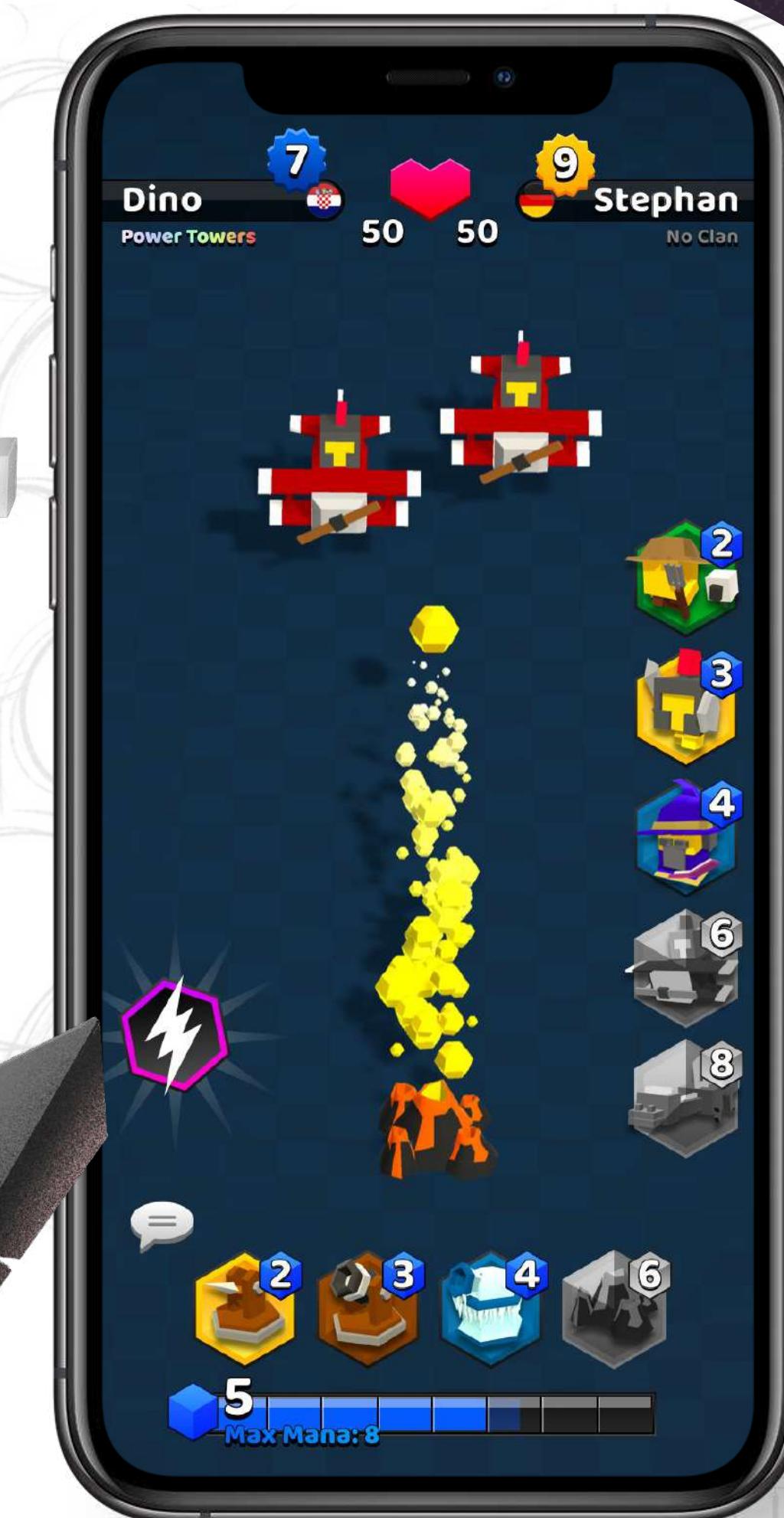
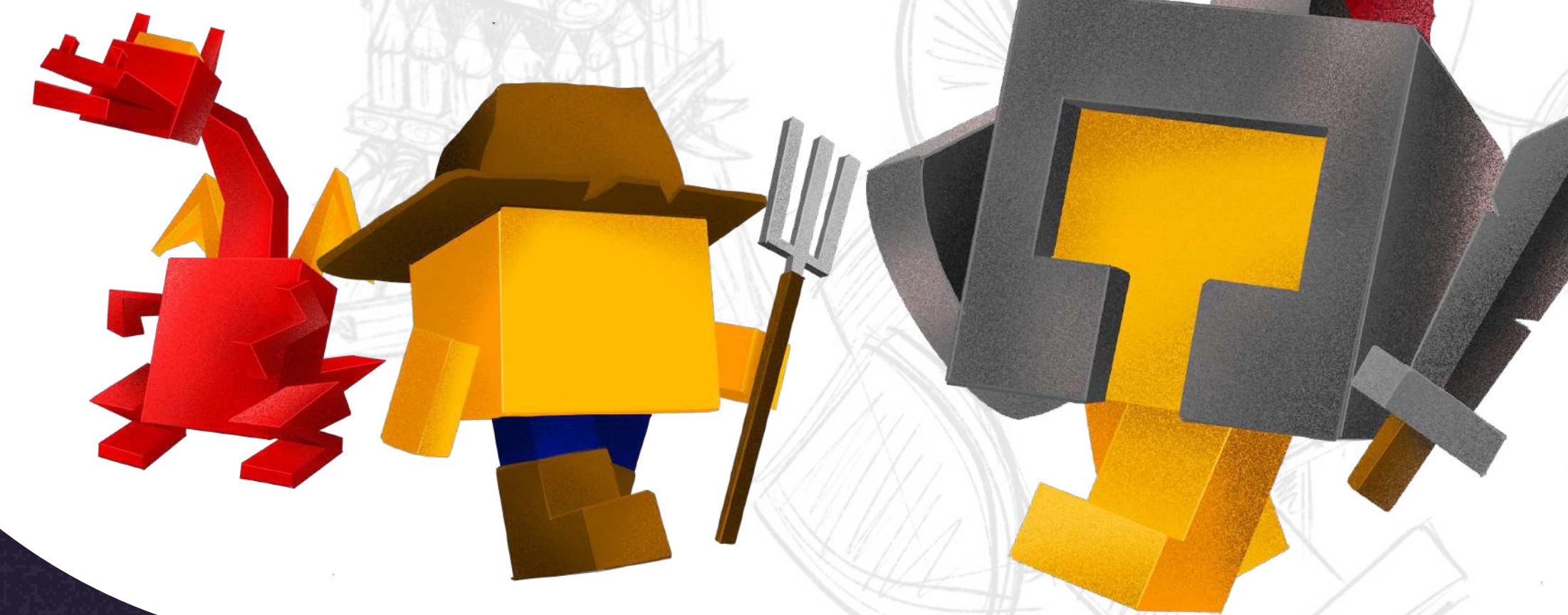
```
4 option csharp_namespace = "GR.S4.Controller.Backend";
5
6 message LogicCalls {
7     message LogicGame {
8         Game.Field field = 1;
9         GameState state = 2;
10    }
11
12    message GameOver {}
13
14    message AiMove {
15        GameState.PlayerIndex player = 1;
16        uint32 depth = 2;
17    }
18
19    message TurnInfo {
20        message Move {
21            GameState.PlayerIndex player = 1;
22            UtcTime time = 2;
23        }
24
25        UtcTime now = 1;
26        UtcTime startTime = 2;
27        GameState.PlayerIndex startPlayer = 3;
28        Move lastMove = 4;
29        Duration moveDuration = 5;
30    }
31
32    message ComboQuery {
33        uint32 column = 1;
34        uint32 height = 2;
35    }
36
37    oneof calls {
38        LogicGame setState = 1;
39        GameState.Move canMove = 2;
40        AiMove aiMove = 3;
41    }
42
43    message GetCubePlacement {
44        uint32 id = 1;
45    }
46
47    message CanMove {
48        GameState.Move move = 1;
49    }
50
51    message AiMoveResponse {
52        GameState.Move move = 1;
53    }
54
55    message WinnerCombos {
56        repeated CubePos positions = 1;
57    }
58
59    oneof kind {
60        Draw draw = 1;
61        WinnerCombos combos = 2;
62    }
63
64
65    message TurnInfo {
66        GameState.PlayerIndex player = 1;
67        UtcTime start = 2;
68    }
69
70    message Draw {}
71
72    message WinnerCombos {
73        repeated CubePos positions = 1;
74    }
75
76    oneof kind {
77        Draw draw = 1;
78        WinnerCombos combos = 2;
79    }
80
81
82    message TurnInfo {
83        GameState.PlayerIndex player = 1;
84        UtcTime start = 2;
85    }
86
87    message ComboQuery {
88        message ComboPos {
89            CubePos position = 1;
90            bool missing = 2;
91        }
92
93        repeated ComboPos positions = 1;
94    }
95
96        repeated Combo combos = 1;
97
98
99    oneof calls {
100        GetCubePlacement getCubePlacement = 1;
101        CanMove canMove = 2;
102        AiMove aiMove = 3;
103    }
104
105    message WinnerCombos {
106        repeated CubePos positions = 1;
107    }
108
109    oneof kind {
110        Draw draw = 1;
111        WinnerCombos combos = 2;
112    }
113
114    message TurnInfo {
115        GameState.PlayerIndex player = 1;
116        UtcTime start = 2;
117    }
118
119    message Draw {}
120
121    message WinnerCombos {
122        repeated CubePos positions = 1;
123    }
124
125    oneof kind {
126        Draw draw = 1;
127        WinnerCombos combos = 2;
128    }
129
130    message TurnInfo {
131        GameState.PlayerIndex player = 1;
132        UtcTime start = 2;
133    }
134
135    message Draw {}
136
137    message WinnerCombos {
138        repeated CubePos positions = 1;
139    }
140
141    oneof kind {
142        Draw draw = 1;
143        WinnerCombos combos = 2;
144    }
145
146    message TurnInfo {
147        GameState.PlayerIndex player = 1;
148        UtcTime start = 2;
149    }
150
151    message Draw {}
152
153    message WinnerCombos {
154        repeated CubePos positions = 1;
155    }
156
157    oneof kind {
158        Draw draw = 1;
159        WinnerCombos combos = 2;
160    }
161
162    message TurnInfo {
163        GameState.PlayerIndex player = 1;
164        UtcTime start = 2;
165    }
166
167    message Draw {}
168
169    message WinnerCombos {
170        repeated CubePos positions = 1;
171    }
172
173    oneof kind {
174        Draw draw = 1;
175        WinnerCombos combos = 2;
176    }
177
178    message TurnInfo {
179        GameState.PlayerIndex player = 1;
180        UtcTime start = 2;
181    }
182
183    message Draw {}
184
185    message WinnerCombos {
186        repeated CubePos positions = 1;
187    }
188
189    oneof kind {
190        Draw draw = 1;
191        WinnerCombos combos = 2;
192    }
193
194    message TurnInfo {
195        GameState.PlayerIndex player = 1;
196        UtcTime start = 2;
197    }
198
199    message Draw {}
200
201    message WinnerCombos {
202        repeated CubePos positions = 1;
203    }
204
205    oneof kind {
206        Draw draw = 1;
207        WinnerCombos combos = 2;
208    }
209
210    message TurnInfo {
211        GameState.PlayerIndex player = 1;
212        UtcTime start = 2;
213    }
214
215    message Draw {}
216
217    message WinnerCombos {
218        repeated CubePos positions = 1;
219    }
220
221    oneof kind {
222        Draw draw = 1;
223        WinnerCombos combos = 2;
224    }
225
226    message TurnInfo {
227        GameState.PlayerIndex player = 1;
228        UtcTime start = 2;
229    }
230
231    message Draw {}
232
233    message WinnerCombos {
234        repeated CubePos positions = 1;
235    }
236
237    oneof kind {
238        Draw draw = 1;
239        WinnerCombos combos = 2;
240    }
241
242    message TurnInfo {
243        GameState.PlayerIndex player = 1;
244        UtcTime start = 2;
245    }
246
247    message Draw {}
248
249    message WinnerCombos {
250        repeated CubePos positions = 1;
251    }
252
253    oneof kind {
254        Draw draw = 1;
255        WinnerCombos combos = 2;
256    }
257
258    message TurnInfo {
259        GameState.PlayerIndex player = 1;
260        UtcTime start = 2;
261    }
262
263    message Draw {}
264
265    message WinnerCombos {
266        repeated CubePos positions = 1;
267    }
268
269    oneof kind {
270        Draw draw = 1;
271        WinnerCombos combos = 2;
272    }
273
274    message TurnInfo {
275        GameState.PlayerIndex player = 1;
276        UtcTime start = 2;
277    }
278
279    message Draw {}
280
281    message WinnerCombos {
282        repeated CubePos positions = 1;
283    }
284
285    oneof kind {
286        Draw draw = 1;
287        WinnerCombos combos = 2;
288    }
289
290    message TurnInfo {
291        GameState.PlayerIndex player = 1;
292        UtcTime start = 2;
293    }
294
295    message Draw {}
296
297    message WinnerCombos {
298        repeated CubePos positions = 1;
299    }
300
301    oneof kind {
302        Draw draw = 1;
303        WinnerCombos combos = 2;
304    }
305
306    message TurnInfo {
307        GameState.PlayerIndex player = 1;
308        UtcTime start = 2;
309    }
310
311    message Draw {}
312
313    message WinnerCombos {
314        repeated CubePos positions = 1;
315    }
316
317    oneof kind {
318        Draw draw = 1;
319        WinnerCombos combos = 2;
320    }
321
322    message TurnInfo {
323        GameState.PlayerIndex player = 1;
324        UtcTime start = 2;
325    }
326
327    message Draw {}
328
329    message WinnerCombos {
330        repeated CubePos positions = 1;
331    }
332
333    oneof kind {
334        Draw draw = 1;
335        WinnerCombos combos = 2;
336    }
337
338    message TurnInfo {
339        GameState.PlayerIndex player = 1;
340        UtcTime start = 2;
341    }
342
343    message Draw {}
344
345    message WinnerCombos {
346        repeated CubePos positions = 1;
347    }
348
349    oneof kind {
350        Draw draw = 1;
351        WinnerCombos combos = 2;
352    }
353
354    message TurnInfo {
355        GameState.PlayerIndex player = 1;
356        UtcTime start = 2;
357    }
358
359    message Draw {}
360
361    message WinnerCombos {
362        repeated CubePos positions = 1;
363    }
364
365    oneof kind {
366        Draw draw = 1;
367        WinnerCombos combos = 2;
368    }
369
370    message TurnInfo {
371        GameState.PlayerIndex player = 1;
372        UtcTime start = 2;
373    }
374
375    message Draw {}
376
377    message WinnerCombos {
378        repeated CubePos positions = 1;
379    }
380
381    oneof kind {
382        Draw draw = 1;
383        WinnerCombos combos = 2;
384    }
385
386    message TurnInfo {
387        GameState.PlayerIndex player = 1;
388        UtcTime start = 2;
389    }
390
391    message Draw {}
392
393    message WinnerCombos {
394        repeated CubePos positions = 1;
395    }
396
397    oneof kind {
398        Draw draw = 1;
399        WinnerCombos combos = 2;
400    }
401
402    message TurnInfo {
403        GameState.PlayerIndex player = 1;
404        UtcTime start = 2;
405    }
406
407    message Draw {}
408
409    message WinnerCombos {
410        repeated CubePos positions = 1;
411    }
412
413    oneof kind {
414        Draw draw = 1;
415        WinnerCombos combos = 2;
416    }
417
418    message TurnInfo {
419        GameState.PlayerIndex player = 1;
420        UtcTime start = 2;
421    }
422
423    message Draw {}
424
425    message WinnerCombos {
426        repeated CubePos positions = 1;
427    }
428
429    oneof kind {
430        Draw draw = 1;
431        WinnerCombos combos = 2;
432    }
433
434    message TurnInfo {
435        GameState.PlayerIndex player = 1;
436        UtcTime start = 2;
437    }
438
439    message Draw {}
440
441    message WinnerCombos {
442        repeated CubePos positions = 1;
443    }
444
445    oneof kind {
446        Draw draw = 1;
447        WinnerCombos combos = 2;
448    }
449
450    message TurnInfo {
451        GameState.PlayerIndex player = 1;
452        UtcTime start = 2;
453    }
454
455    message Draw {}
456
457    message WinnerCombos {
458        repeated CubePos positions = 1;
459    }
460
461    oneof kind {
462        Draw draw = 1;
463        WinnerCombos combos = 2;
464    }
465
466    message TurnInfo {
467        GameState.PlayerIndex player = 1;
468        UtcTime start = 2;
469    }
470
471    message Draw {}
472
473    message WinnerCombos {
474        repeated CubePos positions = 1;
475    }
476
477    oneof kind {
478        Draw draw = 1;
479        WinnerCombos combos = 2;
480    }
481
482    message TurnInfo {
483        GameState.PlayerIndex player = 1;
484        UtcTime start = 2;
485    }
486
487    message Draw {}
488
489    message WinnerCombos {
490        repeated CubePos positions = 1;
491    }
492
493    oneof kind {
494        Draw draw = 1;
495        WinnerCombos combos = 2;
496    }
497
498    message TurnInfo {
499        GameState.PlayerIndex player = 1;
500        UtcTime start = 2;
501    }
502
503    message Draw {}
504
505    message WinnerCombos {
506        repeated CubePos positions = 1;
507    }
508
509    oneof kind {
510        Draw draw = 1;
511        WinnerCombos combos = 2;
512    }
513
514    message TurnInfo {
515        GameState.PlayerIndex player = 1;
516        UtcTime start = 2;
517    }
518
519    message Draw {}
520
521    message WinnerCombos {
522        repeated CubePos positions = 1;
523    }
524
525    oneof kind {
526        Draw draw = 1;
527        WinnerCombos combos = 2;
528    }
529
530    message TurnInfo {
531        GameState.PlayerIndex player = 1;
532        UtcTime start = 2;
533    }
534
535    message Draw {}
536
537    message WinnerCombos {
538        repeated CubePos positions = 1;
539    }
540
541    oneof kind {
542        Draw draw = 1;
543        WinnerCombos combos = 2;
544    }
545
546    message TurnInfo {
547        GameState.PlayerIndex player = 1;
548        UtcTime start = 2;
549    }
550
551    message Draw {}
552
553    message WinnerCombos {
554        repeated CubePos positions = 1;
555    }
556
557    oneof kind {
558        Draw draw = 1;
559        WinnerCombos combos = 2;
560    }
561
562    message TurnInfo {
563        GameState.PlayerIndex player = 1;
564        UtcTime start = 2;
565    }
566
567    message Draw {}
568
569    message WinnerCombos {
570        repeated CubePos positions = 1;
571    }
572
573    oneof kind {
574        Draw draw = 1;
575        WinnerCombos combos = 2;
576    }
577
578    message TurnInfo {
579        GameState.PlayerIndex player = 1;
580        UtcTime start = 2;
581    }
582
583    message Draw {}
584
585    message WinnerCombos {
586        repeated CubePos positions = 1;
587    }
588
589    oneof kind {
590        Draw draw = 1;
591        WinnerCombos combos = 2;
592    }
593
594    message TurnInfo {
595        GameState.PlayerIndex player = 1;
596        UtcTime start = 2;
597    }
598
599    message Draw {}
600
601    message WinnerCombos {
602        repeated CubePos positions = 1;
603    }
604
605    oneof kind {
606        Draw draw = 1;
607        WinnerCombos combos = 2;
608    }
609
610    message TurnInfo {
611        GameState.PlayerIndex player = 1;
612        UtcTime start = 2;
613    }
614
615    message Draw {}
616
617    message WinnerCombos {
618        repeated CubePos positions = 1;
619    }
620
621    oneof kind {
622        Draw draw = 1;
623        WinnerCombos combos = 2;
624    }
625
626    message TurnInfo {
627        GameState.PlayerIndex player = 1;
628        UtcTime start = 2;
629    }
630
631    message Draw {}
632
633    message WinnerCombos {
634        repeated CubePos positions = 1;
635    }
636
637    oneof kind {
638        Draw draw = 1;
639        WinnerCombos combos = 2;
640    }
641
642    message TurnInfo {
643        GameState.PlayerIndex player = 1;
644        UtcTime start = 2;
645    }
646
647    message Draw {}
648
649    message WinnerCombos {
650        repeated CubePos positions = 1;
651    }
652
653    oneof kind {
654        Draw draw = 1;
655        WinnerCombos combos = 2;
656    }
657
658    message TurnInfo {
659        GameState.PlayerIndex player = 1;
660        UtcTime start = 2;
661    }
662
663    message Draw {}
664
665    message WinnerCombos {
666        repeated CubePos positions = 1;
667    }
668
669    oneof kind {
670        Draw draw = 1;
671        WinnerCombos combos = 2;
672    }
673
674    message TurnInfo {
675        GameState.PlayerIndex player = 1;
676        UtcTime start = 2;
677    }
678
679    message Draw {}
680
681    message WinnerCombos {
682        repeated CubePos positions = 1;
683    }
684
685    oneof kind {
686        Draw draw = 1;
687        WinnerCombos combos = 2;
688    }
689
690    message TurnInfo {
691        GameState.PlayerIndex player = 1;
692        UtcTime start = 2;
693    }
694
695    message Draw {}
696
697    message WinnerCombos {
698        repeated CubePos positions = 1;
699    }
700
701    oneof kind {
702        Draw draw = 1;
703        WinnerCombos combos = 2;
704    }
705
706    message TurnInfo {
707        GameState.PlayerIndex player = 1;
708        UtcTime start = 2;
709    }
710
711    message Draw {}
712
713    message WinnerCombos {
714        repeated CubePos positions = 1;
715    }
716
717    oneof kind {
718        Draw draw = 1;
719        WinnerCombos combos = 2;
720    }
721
722    message TurnInfo {
723        GameState.PlayerIndex player = 1;
724        UtcTime start = 2;
725    }
726
727    message Draw {}
728
729    message WinnerCombos {
730        repeated CubePos positions = 1;
731    }
732
733    oneof kind {
734        Draw draw = 1;
735        WinnerCombos combos = 2;
736    }
737
738    message TurnInfo {
739        GameState.PlayerIndex player = 1;
740        UtcTime start = 2;
741    }
742
743    message Draw {}
744
745    message WinnerCombos {
746        repeated CubePos positions = 1;
747    }
748
749    oneof kind {
750        Draw draw = 1;
751        WinnerCombos combos = 2;
752    }
753
754    message TurnInfo {
755        GameState.PlayerIndex player = 1;
756        UtcTime start = 2;
757    }
758
759    message Draw {}
760
761    message WinnerCombos {
762        repeated CubePos positions = 1;
763    }
764
765    oneof kind {
766        Draw draw = 1;
767        WinnerCombos combos = 2;
768    }
769
770    message TurnInfo {
771        GameState.PlayerIndex player = 1;
772        UtcTime start = 2;
773    }
774
775    message Draw {}
776
777    message WinnerCombos {
778        repeated CubePos positions = 1;
779    }
780
781    oneof kind {
782        Draw draw = 1;
783        WinnerCombos combos = 2;
784    }
785
786    message TurnInfo {
787        GameState.PlayerIndex player = 1;
788        UtcTime start = 2;
789    }
790
791    message Draw {}
792
793    message WinnerCombos {
794        repeated CubePos positions = 1;
795    }
796
797    oneof kind {
798        Draw draw = 1;
799        WinnerCombos combos = 2;
800    }
801
802    message TurnInfo {
803        GameState.PlayerIndex player = 1;
804        UtcTime start = 2;
805    }
806
807    message Draw {}
808
809    message WinnerCombos {
810        repeated CubePos positions = 1;
811    }
812
813    oneof kind {
814        Draw draw = 1;
815        WinnerCombos combos = 2;
816    }
817
818    message TurnInfo {
819        GameState.PlayerIndex player = 1;
820        UtcTime start = 2;
821    }
822
823    message Draw {}
824
825    message WinnerCombos {
826        repeated CubePos positions = 1;
827    }
828
829    oneof kind {
830        Draw draw = 1;
831        WinnerCombos combos = 2;
832    }
833
834    message TurnInfo {
835        GameState.PlayerIndex player = 1;
836        UtcTime start = 2;
837    }
838
839    message Draw {}
840
841    message WinnerCombos {
842        repeated CubePos positions = 1;
843    }
844
845    oneof kind {
846        Draw draw = 1;
847        WinnerCombos combos = 2;
848    }
849
850    message TurnInfo {
851        GameState.PlayerIndex player = 1;
852        UtcTime start = 2;
853    }
854
855    message Draw {}
856
857    message WinnerCombos {
858        repeated CubePos positions = 1;
859    }
860
861    oneof kind {
862        Draw draw = 1;
863        WinnerCombos combos = 2;
864    }
865
866    message TurnInfo {
867        GameState.PlayerIndex player = 1;
868        UtcTime start = 2;
869    }
870
871    message Draw {}
872
873    message WinnerCombos {
874        repeated CubePos positions = 1;
875    }
876
877    oneof kind {
878        Draw draw = 1;
879        WinnerCombos combos = 2;
880    }
881
882    message TurnInfo {
883        GameState.PlayerIndex player = 1;
884        UtcTime start = 2;
885    }
886
887    message Draw {}
888
889    message WinnerCombos {
890        repeated CubePos positions = 1;
891    }
892
893    oneof kind {
894        Draw draw = 1;
895        WinnerCombos combos = 2;
896    }
897
898    message TurnInfo {
899        GameState.PlayerIndex player = 1;
900        UtcTime start = 2;
901    }
902
903    message Draw {}
904
905    message WinnerCombos {
906        repeated CubePos positions = 1;
907    }
908
909    oneof kind {
910        Draw draw = 1;
911        WinnerCombos combos = 2;
912    }
913
914    message TurnInfo {
915        GameState.PlayerIndex player = 1;
916        UtcTime start = 2;
917    }
918
919    message Draw {}
920
921    message WinnerCombos {
922        repeated CubePos positions = 1;
923    }
924
925    oneof kind {
926        Draw draw = 1;
927        WinnerCombos combos = 2;
928    }
929
930    message TurnInfo {
931        GameState.PlayerIndex player = 1;
932        UtcTime start = 2;
933    }
934
935    message Draw {}
936
937    message WinnerCombos {
938        repeated CubePos positions = 1;
939    }
940
941    oneof kind {
942        Draw draw = 1;
943        WinnerCombos combos = 2;
944    }
945
946    message TurnInfo {
947        GameState.PlayerIndex player = 1;
948        UtcTime start = 2;
949    }
950
951    message Draw {}
952
953    message WinnerCombos {
954        repeated CubePos positions = 1;
955    }
956
957    oneof kind {
958        Draw draw = 1;
959        WinnerCombos combos = 2;
960    }
961
962    message TurnInfo {
963        GameState.PlayerIndex player = 1;
964        UtcTime start = 2;
965    }
966
967    message Draw {}
968
969    message WinnerCombos {
970        repeated CubePos positions = 1;
971    }
972
973    oneof kind {
974        Draw draw = 1;
975        WinnerCombos combos = 2;
976    }
977
978    message TurnInfo {
979        GameState.PlayerIndex player = 1;
980        UtcTime start = 2;
981    }
982
983    message Draw {}
984
985    message WinnerCombos {
986        repeated CubePos positions = 1;
987    }
988
989    oneof kind {
990        Draw draw = 1;
991        WinnerCombos combos = 2;
992    }
993
994    message TurnInfo {
995        GameState.PlayerIndex player = 1;
996        UtcTime start = 2;
997    }
998
999    message Draw {}
1000
1001    message WinnerCombos {
1002        repeated CubePos positions = 1;
1003    }
1004
1005    oneof kind {
1006        Draw draw = 1;
1007        WinnerCombos combos = 2;
1008    }
1009
1010    message TurnInfo {
1011        GameState.PlayerIndex player = 1;
1012        UtcTime start = 2;
1013    }
1014
1015    message Draw {}
1016
1017   
```

HYBRID CLIENTS 2.0

SHOWCASE: TOWERRANGERS

Tower Rangers

Idle tower defense meets
real-time duels



MODULAR BACKEND

ATLAS SERVER COMPONENTS

- Authentication, SSO (FB, SIWA, Google)
- IAP Validation
- Push Notifications
- PubSub Realtime matches
- AB Testing
- Friends Lists, Leaderboards
- Tracking

MODULAR BACKEND

ATLAS STACK

- WARP
- API crates for 3rd parties (FB, SIWA)
- reusable, composable WARP layer
- static typed DI (using HList)
- find opensource at:

<https://github.com/gameroasters/atlasserver>

MODULAR BACKEND

EXAMPLE: HLIST

```
fn sculpt<Ts, Indices>(
    self
) -> (Ts, <HCons<Head, Tail> as Sculptor<Ts, Indices>>::Remainder)
where
    HCons<Head, Tail>: Sculptor<Ts, Indices>,
```

Consume the current HList and return an HList with the requested shape.

`sculpt` allows us to extract/reshape/sculpt the current HList into another shape, provided that the requested shape's types are contained within the current HList.

The `Indices` type parameter allows the compiler to figure out that `Ts` and `Self` can be morphed into each other.

Examples

```
use frunk_core::{hlist, HList};

let h = hlist![9000, "joe", 41f32, true];
let (reshaped, remainder): (HList![f32, i32, &str], _) = h.sculpt();
assert_eq!(reshaped, hlist![41f32, 9000, "joe"]);
assert_eq!(remainder, hlist![true]);
```

```
impl CustomModule for S4Module {
    type Resources = Hlist![
        Arc<PlayerStateResource>,
        Arc<TrackingResource>,
        Arc<UserLoginResource>,
        Arc<RankingResource>,
        Arc<PushNotificationResource>,
        Arc<S4FcmSendResource>,
        Arc<SsoResource>,
        Arc<WofResource>,
        Arc<RealtimeSenderArenaResource>,
        Arc<FriendsResource>,
        Arc<SpectateResource>,
        Arc<AdsResource>,
        Arc<LobbyResource>,
        Arc<GameSpawnerResource>,
        Arc<CheatsResource>,
        Arc<S4BroadcastResource>,
        Arc<ShopResource>
    ];
}
```



MODULAR BACKEND

EXAMPLE: REALITY

MODULAR BACKEND

EXAMPLE: WARP TYPE MADNESS

```

    - customModule for S4Module {
fn create_filter<S: atlasserver::ModuleResources<Self>>(
    >() And<And<And<And<And<And<And<..., ...>, ...>, ...>, ...
    .and(friends) And<And<And<And<And<And<And<..., ...>, ...>, ...>, ...
    .and_then(fun: facebook_friends_filter_fn);

    let spectate_filter: AndThen<And<And<And<And<And<..., ...>, ...>, ...>, ...> = warp::path!("s4" / "spectate")
        .and(userlogin::session_filter(user_resource.clone()))
        .and(warp::post())
        .and(pbwarpc::protobuf_body::<schema::SpectateRequest>())
        .and(spectate)
        .and_then(fun: spectate_filter_fn);

    let wof: BoxedFilter<(Box<dyn Reply>,>) = create_filters_wof(wof_res, &user_resource);
    let lobby: BoxedFilter<(Box<dyn Reply>,>) = create_filters_lobby(lobby_res, &user_resource);
    let siwa: BoxedFilter<(Box<dyn Reply>,>) = create_filters_siwa(&user_resource, sso_res);

    let tracking_filter: AndThen<And<And<And<And<And<..., ...>, ...>, ...>, ...> = warp::path!("tracking")
        .and(userlogin::session_filter(user_resource))
        .and(warp::post())
        .and(pbwarpc::protobuf_body::<schema::TrackingRequest>())
        .and(warp::any().map(fun: move || tracking_resource.clone()))
        .and_then(fun: tracking_filter_fn);

    let time_filter: Map<And<And<impl Filter<Extract = ..., Error = ...>, ...>, ...> = warp::path!("time").map(time_filter_fn);

    upgrade_item_filter AndThen<And<And<And<And<And<And<..., ...>, ...>, ...>, ...>, ...>
        .or(lobby) Or<AndThen<And<And<And<And<..., ...>, ...>, ...>, ...>
        .or(wof) Or<Or<AndThen<And<And<And<..., ...>, ...>, ...>, ...>
        .or(siwa) Or<Or<Or<AndThen<And<And<..., ...>, ...>, ...>, ...>
        .or(get_kotd_filter) Or<Or<Or<Or<Or<AndThen<And<..., ...>, ...>, ...>
        .or(ranking_filter) Or<Or<Or<Or<Or<Or<AndThen<And<..., ...>, ...>, ...>
        .or(select_cube_filter) Or<Or<Or<Or<Or<Or<Or<AndThen<..., ...>, ...>, ...>
        .or(matchmaking_filter) Or<Or<Or<Or<Or<Or<Or<Or<AndThen<..., ...>, ...>, ...>
        .or(get_state_filter) Or<Or<Or<Or<Or<Or<Or<Or<Or<AndThen<..., ...>, ...>, ...>
        .or(game_get_filter) Or<Or<Or<Or<Or<Or<Or<Or<Or<..., ...>, ...>, ...>, ...>
        .or(game_resign_filter) Or<Or<Or<Or<Or<Or<Or<Or<Or<Or<..., ...>, ...>, ...>, ...>
        .or(game_leave_filter) Or<Or<Or<Or<Or<Or<Or<Or<Or<Or<..., ...>, ...>, ...>, ...>
        .or(game_move_filter) Or<Or<Or<Or<Or<Or<Or<Or<Or<..., ...>, ...>, ...>, ...>
        .or(game_rematch_filter) Or<Or<Or<Or<Or<Or<Or<Or<Or<..., ...>, ...>, ...>, ...>
        .or(claim_reward_filter) Or<Or<Or<Or<Or<Or<Or<Or<Or<..., ...>, ...>, ...>, ...>
        .or(aet state other filter) Or<Or<Or<Or<Or<Or<Or<Or<Or<..., ...>, ...>, ...>, ...>

```

HALF A DECADE OF RUST IN MOBILE GAMES

MODULAR BACKEND

SHOWCASE: STACK4

Stack4

Connect Four in 3D

Download on the
App Store

GET IT ON
Google Play



FULLSTACK RUST MOBILE GAMES

HELLO BEVY

- ECS-first, multithreaded, fast
- Modular using Plugins
- Cross Platform: win,mac,linux,ios,android,web...
- Free and OpenSource
- Huge community (19k on Discord)

FULLSTACK RUST MOBILE GAMES

ZOOLITAIRE



FULLSTACK RUST MOBILE GAMES

BACKEND NOW USING AXIUM

- Axium is more ergonomic
- Easier to modularise using Tower
- HLIST-free DI using Extractors
- Reuses components of Atlas

FULLSTACK RUST MOBILE GAMES

BACKEND NOW USING AXIUM

```
1    #[instrument(skip(app))]
2    pub async fn mod_get_question(
3        Path((id: String, secret: String, question_id: i64)): Path<(String, String, i64>,
4        State(app: Arc<App>): State<SharedApp>,
5    ) -> std::result::Result<impl IntoResponse, InternalError> {
6        tracing::info!("mod_get_question");
7
8        Ok(Json(app.get_question(id, Some(secret), question_id).await?))
9    }
10
11    #[instrument(skip(app))]
12    pub async fn get_question(
13        Path((id: String, question_id: i64)): Path<(String, i64>,
14        State(app: Arc<App>): State<SharedApp>,
15    ) -> std::result::Result<impl IntoResponse, InternalError> {
16        tracing::info!("get_question");
17
18        Ok(Json(app.get_question(id, None, question_id).await?))
19    }
20
21    #[instrument(skip(app))]
22    pub async fn mod_edit_question(
23        Path((id: String, secret: String, question_id: i64)): Path<(String, String, i64>,
24        State(app: Arc<App>): State<SharedApp>,
25        Json(payload: ModQuestion): Json<shared::ModQuestion>,
26    ) -> std::result::Result<impl IntoResponse, InternalError> {
27        tracing::info!("mod_edit_question");
28
29        Ok(Json(
30            app.mod_edit_question(id, secret, question_id, payload)
31                .await?,
32        ))
33    }
34}
```

FULLSTACK RUST MOBILE GAMES

BACKEND NOW USING AXIUM

```
15
16 let admin_routes = Router::new()
17 .route("/admin/login", get(admin_login_handler))
18 .route("/admin/logout", post(logout_handler))
19 .route("/logout", post(logout_handler));
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76 let event_routes = Router::new()
77 .route("/:id", get(handle::getevent_handler))
78 .route("/:id/pwd", post(handle::set_event_password))
79 .route("/add", post(handle::addevent_handler))
80 .route("/editlike/:id", post(handle::editlike_handler))
81 .route("/addquestion/:id", post(handle::addquestion_handler))
82 .route("/question/:id/:question_id", get(handle::get_question));
83
84 #[rustfmt::skip]
85 let mod_routes = Router::new()
86 .route("/:id/:secret", get(handle::mod_get_event))
87 .route("/upgrade/:id/:secret", get(handle::mod_premium_upgrade))
88 .route("/capture/:id/:order", get(handle::mod_premium_capture))
89 .route("/delete/:id/:secret", get(handle::mod_delete_event))
90 .route("/question/:id/:secret/:question_id", get(handle::mod_get_question))
91 .route("/questionmod/:id/:secret/:question_id", post(handle::mod_edit_question))
92 .route("/:id/:secret", post(handle::mod_edit_event));
93
94 #[rustfmt::skip]
95 let router = Router::new()
96 .route("/api/ping", get(handle::ping_handler))
97 .route("/api/version", get(handle::version_handler))
98 .route("/api/error", get(handle::error_handler))
99 .route("/api/payment/stripe/webhook", post(stripe_webhooks::handle_webhook))
100 .route("/push/:id", get(push_handler))
101 .nest("/api/event", event_routes)
102 .nest("/api/mod/event", mod_routes)
103 .nest("/api/admin", admin_routes)
104 .layer(auth_layer)
105 .layer(session_layer)
106 .layer(SetSensitiveRequestHeadersLayer::new(once(header::COOKIE)))
107 .layer(SentryHttpLayer::with_transaction())
108 .layer(NewSentryLayer::new_from_top())
109 .layer(TraceLayer::new_for_http())
110 .layer(setup_cors())
111 .with_state(Arc::clone(&app));
```

FULLSTACK RUST MOBILE GAMES

ZOOLITAIRE: OPENSOURCE

- `bevy_ios_impact`
- `bevy_ios_review`
- `bevy_ios_alerts`
- `bevy_ios_notifications` (soon)

SEE [HTTPS://GITHUB.COM/RUSTUNIT](https://github.com/rustunit)

THANK YOU

- Join the bevy meetups: bevygamedev.com
- Chat with us on discord: rustunit.com
- We are open for Internships, get in touch
- Meet us at Oxidize Conf in Berlin

GET ZOOLITAIRE NOW:
[HTTPS://ZOOLOITAIRE.COM](https://zoolitaire.com)

