CodeGames

Playing with a Real-Time Coding Game

Buenos Aires Erlang & Elixir Factory Lite 2017

https://manas.tech

We are Manas.

We build unconventional software_

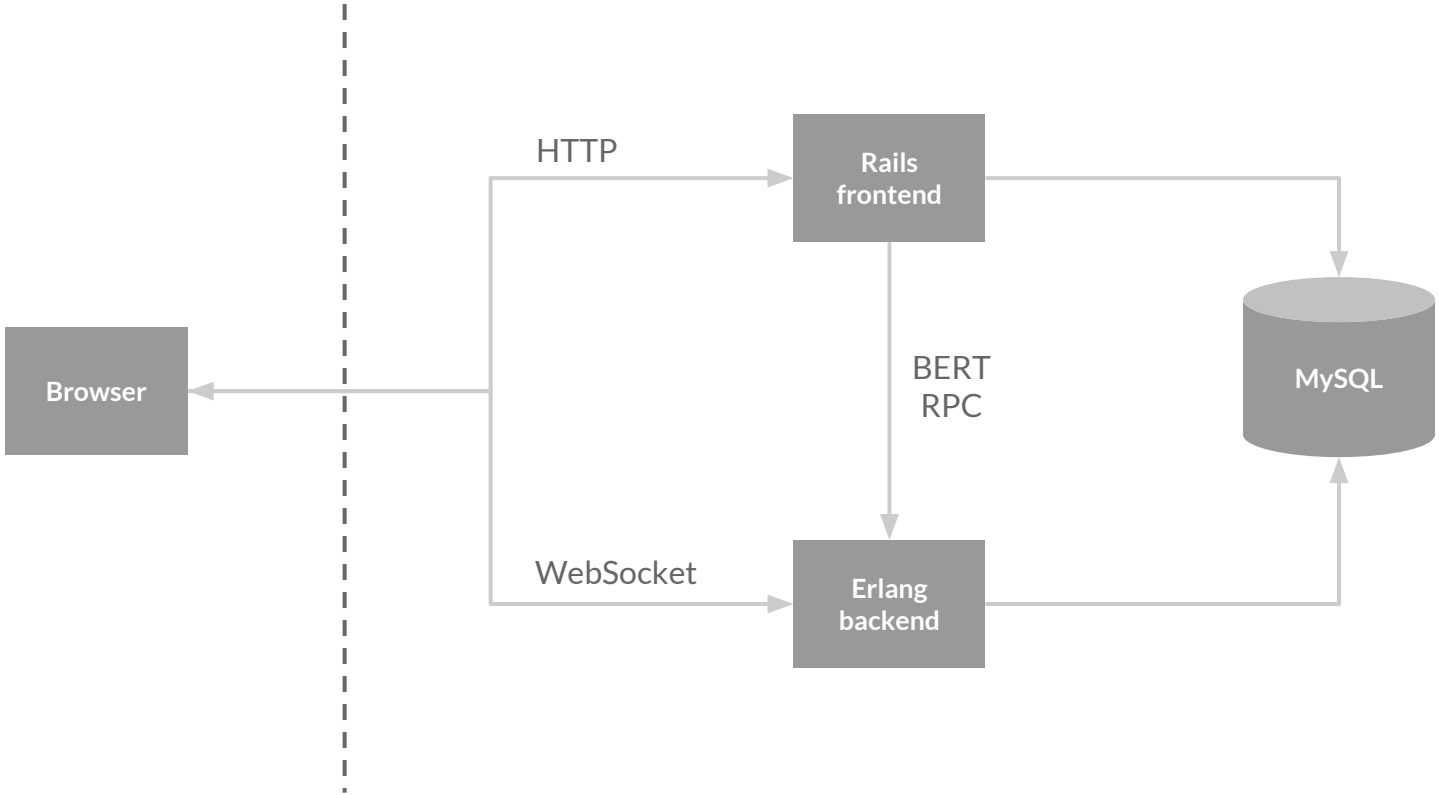
https://manas.tech

What is Code Games?

- http://codegames.io
- Coding game
- Custom language
- Real-time
- Competitive player vs. player
- Started as an experiment in 2014

Demo

Overview of the architecture



Frontend architecture

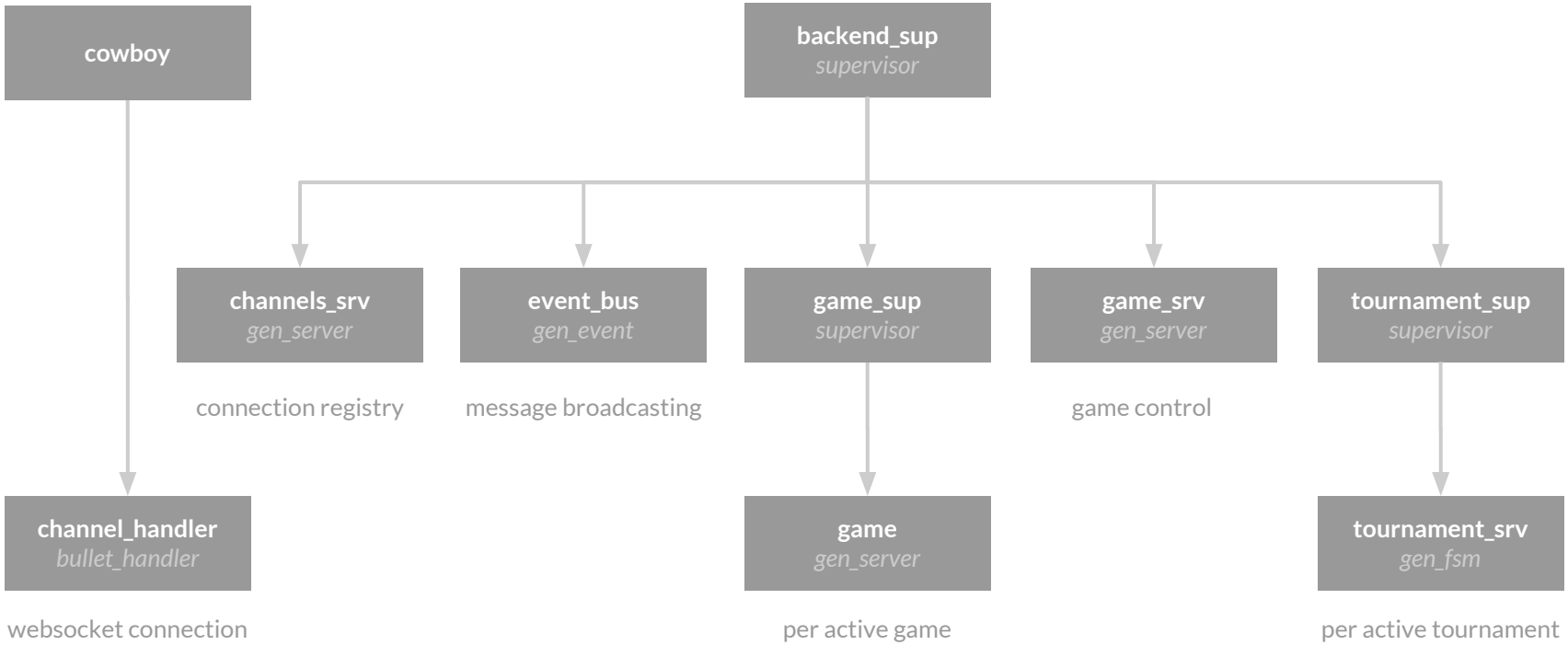
- Rails
  - User registration
  - Subscription to tournaments
  - Tournament progress
  - `bert_rpc`
- Javascript
  - HTML5 Canvas & Audio
  - Websockets
  - CodeMirror
  - A sprinkle of React

Backend architecture

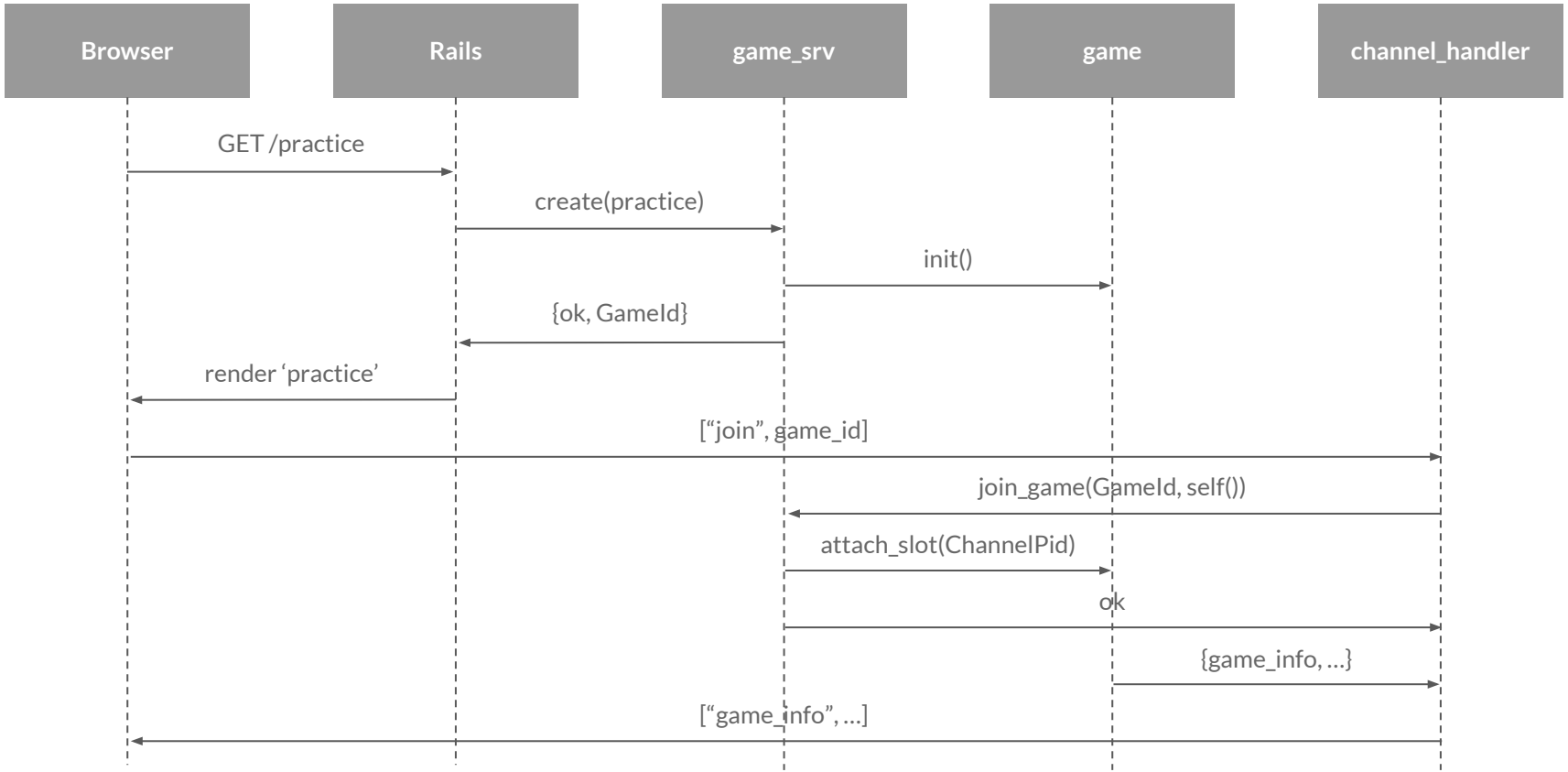
- Erlang application
  - Game simulation & streaming
  - User code execution
  - Managing games & tournaments
  - Message broadcasting
- Tech stack
  - OTP ftw!
  - Cowboy (+ bullet)
  - Leex + Yecc
  - Ernie

Erlang Backend

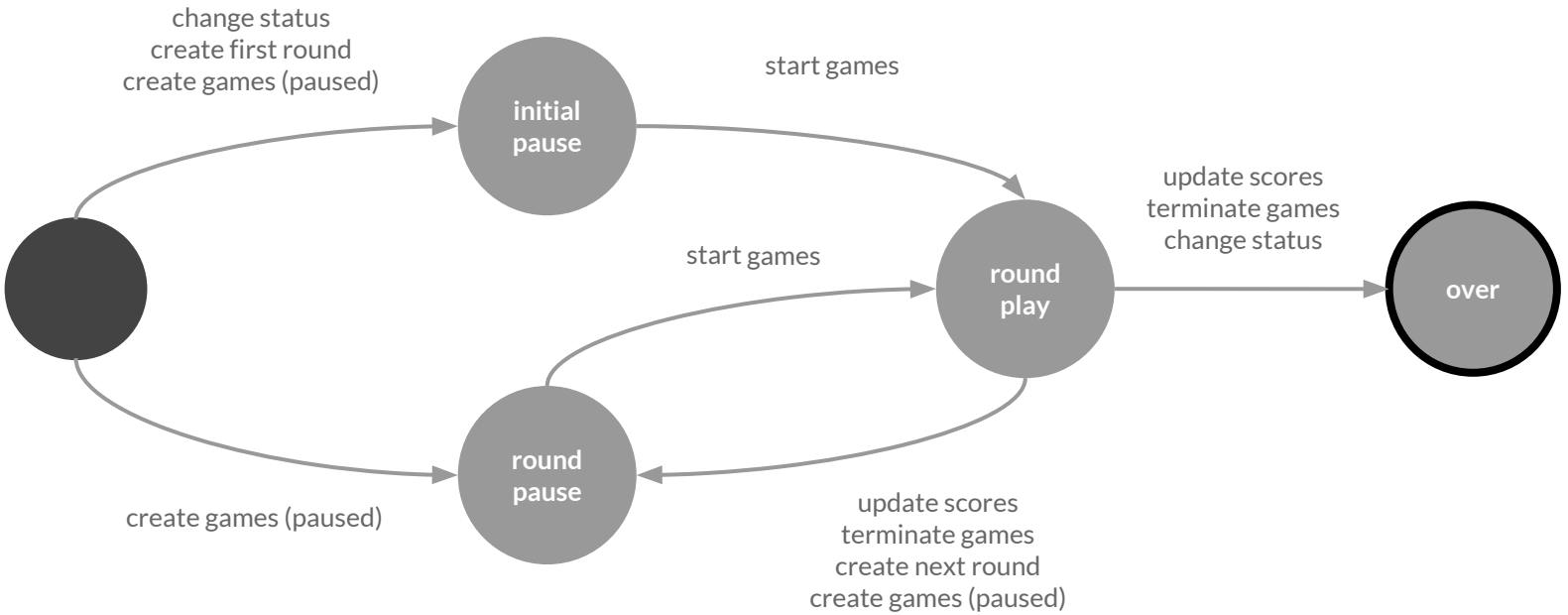
Supervision tree



Game instantiation



Tournament control



Game simulation loop

50 times per second

- Check collisions
- Move ships and projectiles
- Compute built-in variables
- Execute each player user-code
- Perform intended ship actions
- Broadcast updated state to players and spectators
- Enqueue next tick using `send_after`
- Collect stats

Codex: overview

- Deliberately simple and restricted language
- No variable declarations
- One data type: float
- No loops and no recursion
- Variables state is preserved across simulation ticks
- The loop is implicit

Codex: parsing and “compilation”

- Whenever the user changes the code
- Leex and yecc do the heavy lifting

Codex: grammar (partial)

```
Rootsymbol Program.
Program -> Expressions : '$1'.
Expressions -> '$empty' : [].
Expressions -> Expression newline Expressions : ['$1'] ++ '$3'.
Expressions -> Expression : ['$1'].
...
Expression -> Assign : '$1'.
Expression -> If : '$1'.
Expression -> Function : '$1'.
Expression -> Call : '$1'.
...
Assign -> Var '=' Expression : {assign, line_number('$1'), token_value('$1'), '$3'}.
If -> 'if' Expression newline Expressions 'end'
     : {'if', line_number('$1'), '$2', '$4', undefined}.
Function -> 'function' ident '(' FunctionArgs ')' newline Expressions 'end'
     : {function, line_number('$1'), token_value('$2'), '$4', '$7'}.
Call -> ident '(' CallArgs ')' : {call, line_number('$1'), token_value('$1'), '$3'}.
...
```

Codex: parsing and “compilation”

- Whenever the user changes the code
- Leex and yecc do the heavy lifting
- Check for arity mismatches
- Check for recursive calls
- “Link” the AST
    - Gather all user defined symbols (variables and functions)
    - Replace symbol references in AST with indexes
- Migrate user data

Codex: execution model

- Once per tick per player
- `NewContext = codex:run(Ast, Context)`
- Walks the “linked” AST recursively

Codex: execution

```
run(Ast, Context = #context{}) ->
  {NewContext, _Value} = eval(Ast, Context),
  NewContext.

eval(Ast, Context) ->
  try
    do_eval(Ast, Context)
  catch
    {return, ReturnValue} -> ReturnValue
  end.

do_eval([], Context) ->
  {Context, 0};
do_eval([Head|Tail], Context) ->
  {NewContext, _Value} = do_eval(Head, Context),
  do_eval(Tail, NewContext);
```

Codex: execution

```
do_eval({var, Line, {bif, BifIndex}}, Context) ->
  do_eval({call, Line, {bif, BifIndex}, []}, Context);
do_eval({var, _Line, {biv, BivIndex}}, Context = #context{bivs = Bivs}) ->
  Value = element(BivIndex, Bivs),
  {Context, Value};

do_eval({assign, _Line, {user, UserIndex}, Exp}, Context) ->
  {NextContext, Value} = do_eval(Exp, Context),
  NewVars = setelement(UserIndex, NextContext#context.vars, Value),
  {NextContext#context{vars = NewVars}, Value};

do_eval({'if', _Line, Cond, Then, Else}, Context) ->
  {CondContext, CondValue} = do_eval(Cond, Context),
  NormalizedValue = normalize_boolean(CondValue),
  case NormalizedValue of
    0 -> do_eval(Else, CondContext);
    _ -> do_eval(Then, CondContext)
  end;
```

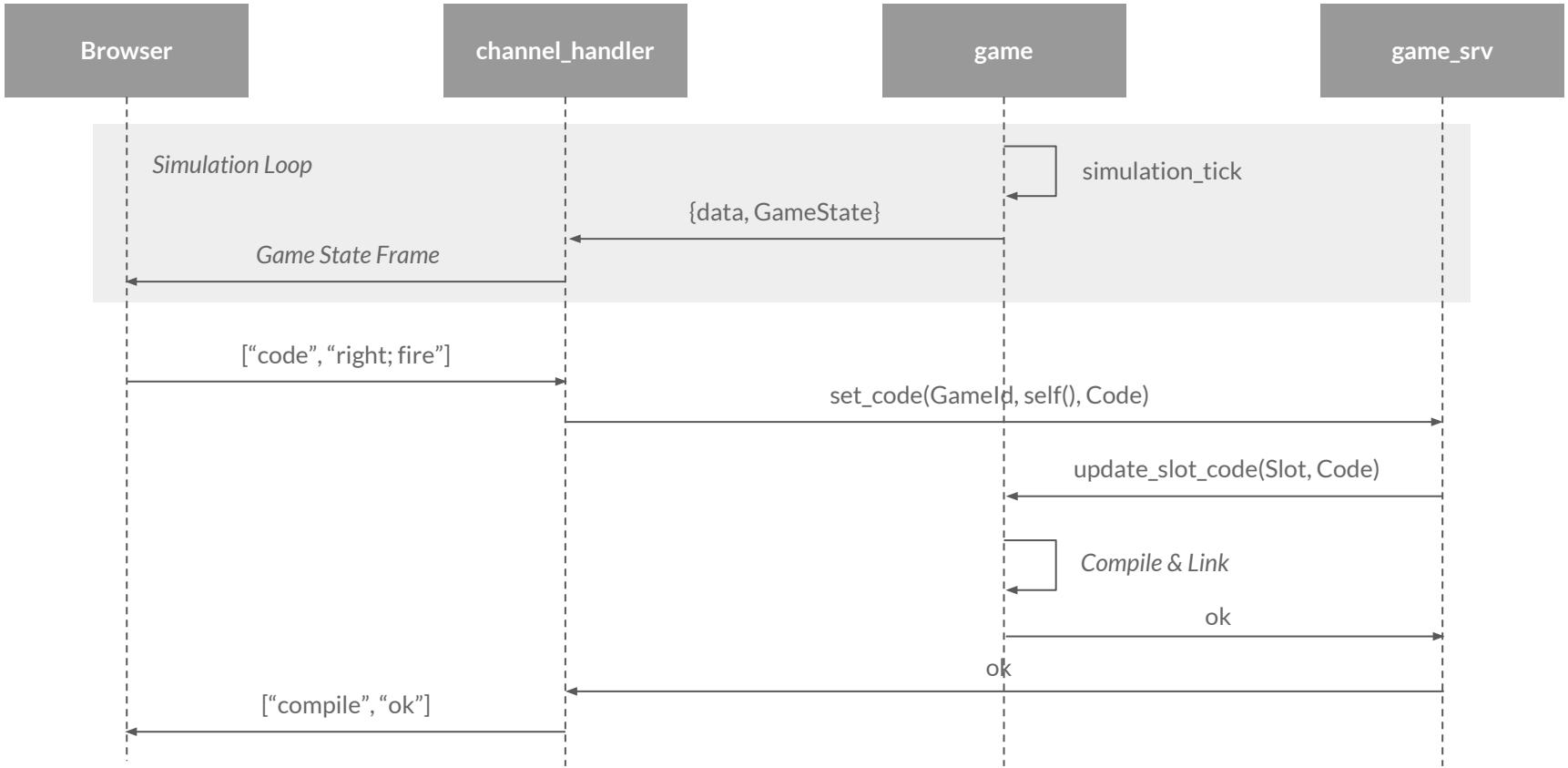
Codex: execution

```
do_eval({call, Line, {bif, BifIndex}, CallArgs}, Context = #context{bifs = Bifs}) ->
  Fun = element(BifIndex, Bifs),
  eval_bif(Fun, CallArgs, Context, Line);
do_eval({call, _Line, {user, UserIndex}, CallArgs}, Context = #context{vars = Vars}) ->
  {function, _, _, FunArgs, Body} = element(UserIndex, Vars),
  {BeforeContext, CallValues} = multi_eval(CallArgs, Context),
  BeforeVars = BeforeContext#context.vars,
  CallVars = prepare_call_vars(FunArgs, CallValues, BeforeVars),
  CallContext = BeforeContext#context{vars = CallVars},
  {AfterContext, Value} = eval(Body, CallContext),
  AfterVars = AfterContext#context.vars,
  RestoredVars = restore_call_vars(FunArgs, BeforeVars, AfterVars),
  FinalContext = AfterContext#context{vars = RestoredVars},
  {FinalContext, Value};
end;
do_eval({return, _Line, Exp}, Context) ->
  throw({return, do_eval(Exp, Context)}).
```

Codex: execution model

- Once per tick per player
- `NewContext = codex:run(Ast, Context)`
- Walks the “linked” AST recursively
- Context: the ship, built-in & user functions and vars
- Functions and vars are kept in tuples
- Hence the need for the link step
    - More complex than a dict/map
    - Needs user data migration
    - For a reasonable number of vars, a tuple is *much* faster

Game streaming



Game state streaming

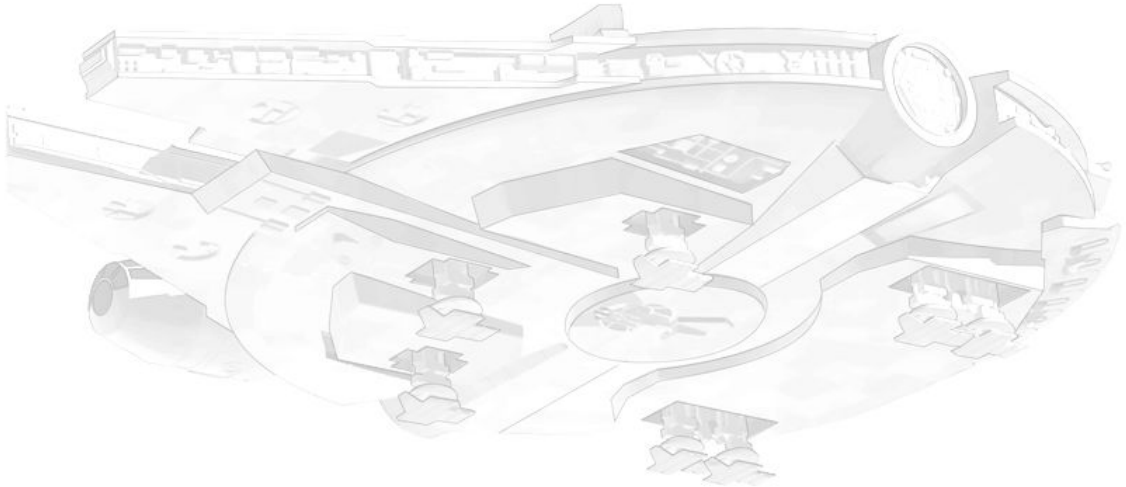
- Server-side: unbounded asynchronous messages
- One websocket frame per game tick state
- Extra frames for control messages (eg. game control and players joining/leaving)
- Spectator channels are monitored by the game process
- Player channels are monitored by the `game_srv` process
- JS side uses a ring buffer

Lessons learned and future

- Player fun & engagement
  - Evolve the rules and the language
  - Increase the depth of the game
  - Lots of ideas in many directions
- Technical
  - Executing arbitrary user code is always risky
  - Leverage Erlang and OTP to scale horizontally
  - Current implementation is CPU bound
  - Large number of spectators could make it IO bound
- Business model
  - Iterate and try again

Conf Tournament!

Monday, July 3rd 18:00



Subscribe using the coupon code `ERLANGROCKS`

Thanks!

 ggiraldez@manas.tech

 @ggiraldez