個人開発集会 / 30分枠

Waterman Cometh

- ウォーターフォールの復権 -
Research → Spec → Oracle で Unity 開発を回してみた話

Unity 個人開発「ShapeSync」で、LLM エージェントを「速いコーダー」ではなく「開発チーム」として運用した記録

完成した方法論の宣伝ではありません。効いた範囲と弱点をそのまま話します。

発表者: zgock / 2026-xx-xx / ※ロゴ等の商標画像は使用していません

目安 1.5分

お前誰よ？

- ずごっく(zgock999)と申します
- わりと昔からopenSUSEユーザー会という組織でごにょごにょやってる OSSじじい
- OpenZFS/openSUSEにちょいちょいcommitしたり
- 最近は道楽でUnity Asset書いたり

ShapeSync なるものを作ってます

- Unity用のキャラクターメイキング /着せ替えAsset
- DAZ Studioライクな「ボーンごとモーフする Full Body Morphシステム」
- VRC民にわかりやすく言うと「リアルタイム版もちふぃったー」
- リアルタイムで「n種類のキャラクター体形を任意比率でリニア合成」
- リアルタイムでHumanoid avator計算、GPU演算による法線マップリアルタイム合成
- StackMachine/Forth言語によるレンダリングパイプライン不問の GPUテクスチャ演算
- リアルタイム変形するruntimeと、pure Humanoid prefab/Scene Objectへのコンパイラ

- 今日の本題ではないため、興味のある方は発表後に個別にお問い合わせを

目安 1.5分

始まりは単純でした

- 「キャラメイクシステム欲しい！」
- 「Asset Storeを見ても、俺の欲しいDAZ Studio型の奴がどこにもない」
- 「じゃあ作るか！」
- 「だが待てよ！？Asset Storeにないということは何か技術課題があるに違いない」
- 「そこの実証実験からだな！」
- 「OK AI。↑をUnityで実現しようと思うと何がいる？」
- 「あ、そんな感じ？やっぱ単純ではないよね。OK、それ要件書に起こして！」

何を言ってるかわからねえと思うが

- Unityのキャラメイクシステムを作るつもりが、
  いつの間にか
  「LLMと親和性の高い開発プロセスの研究」
  になっていた

- これはそんなおバカの観察日記です

Specを積み上げながらのボトムアップ開発

- Spec1:「そもそも出来るんかこれ？」と思いながら「ボーンごと BlendShapeを実装」
- 「あ、いけるわ。なら応用で表情もいけるな」で Spec2積み上げ
- 「うん、いける。BlendShapeをn本に増やそう」でSpec3
- 「あ、これいるね」でSpecをどんどん積み上げ
- 「これやりたいねえ、でもどうやるか見当つかんな」を Spec Xとして仮置き
- 気づいたらSpec20まで積みあがっていた

- 無意識に「後方リサーチ型開発」をやってたわけだけどそれは後述

最初（Spec1〜8あたり）はいつもの雑開発でした

- 要件書を雑に書いて、「実装計画書いて」＞レビューする＞「実装して」
- Ownership周りの取り決めが要件段階で超雑
- 実装中は良いが、「設計意図」「実装意図」が残っていかない
- 結局「こいついつも同じとこ間違えてんな」と思いつつコードレビュー

- 実装は1時間かもしれないが、レビューして動くものにするまでに12時間

そうだ、要件ちゃんと書こう

- 要件ドラフトを雑に書く
- 「OK AI。これで実装に入れる？何が足りない？」
- 「あっそう。Ownership。Rustのあれね？この情報のOwnershipは･･･こいつで」
- 「このデータを一時保持･･･あっそう。escrowって言うと君がわかりやすいのね？じゃあそれで」
- 「なんか要件定義書というよりも要件契約書になってきたな」
- なんだかんだとAI君と2人で細かい要件書を2〜3時間で仕上げる
- 「OK。これでまとまったね。実装計画に落として」

そうだ、受け入れ条件書こう

- 「工場のラインって絶対受け入れ検査あるよね」
- 「hogeを実現するためにはfugaが必要、これ事前にわかるよなあ」
- 「これ書いておけば出荷前に達成してるかチェックすればいい。」
- 「何より俺のレビューが楽じゃん！」
- 「待てよ？そもそもこれあったらLLM君自身が自己レビュー出来るよね？」

- このあたりから実装では「続けて」「自己レビューして」ぐらいしか言わなくなった
  プロンプトエンジニアリング？何それ美味しいの？
  （正確にはプロンプトエンジニアリング相当を設計でやってるだけである）

誕生！ Waterman 0.0

- **Spec＞Implement>Code**
  - がっつり書いたSpecと、「人間とLLMどちらにも優しい中間言語」”実装計画”、実コードのセット
  - 実装計画には「実装記録」、「レビュー記録」をがんがん追記
  - あら不思議！「設計意図」「実装意図」が履歴として残る！
  - 「あれ？これ昔メインフレームの COBOLやFORTRANでさんざん書いたよね？泣きながら」
  - ウォーターフォールは「理想としては正しかった」、
    ただ「人間がやるにはコストが高すぎた」
    今はLLMのおかげでそのコストが激減した
  - だったらやろうよ！

ところでなんで「 Waterman」？

- The Stranglersのベーシスト、J･J･バーネルのソロアルバム「Euroman Cometh - ヨーロッパの復権 -」のオマージュです

- これで理解できた方、音楽談義がしたいので後で個別にご連絡ください

- ただ、もうすでに「名前の由来」なんかどうでもいいですよね？

このへんで「コードレビュー」がほぼ消えた

- 「中間言語」レベルできちんとしてれば「コードの誤り」はほぼない
- ほぼ

「あ？出来た？どう実装した？」
「あ、それ契約無視」
「あ、それ契約逸脱」
「あ、これ契約漏れだ。要件見直し！」
で、実装計画と要件の更新で片が付く
（というより、要件漏れ以外は自己レビューで　LLM自身が大半を見つけてしまう）

- ごくごく稀に「あれ？それでもおかしい？」時に初めてコードを覗く
- 昔なら「恐怖」でしかなかった「要件変更」も気軽に、かつ自信を持って行える

そうだ、Researchしよう

- 「んー、要件ドラフトを手で起こす時、詰まることがあるなあ」
- 「あ、これ潰すべき技術課題が見えてないせいだ」
- 「OK、AI。要件としてやりたいことはこれ。潰すべき技術課題出して」
- 「あ、やっぱそれか。ほな要件としてこの段いるね（書き書き）」
- 「んー、それこのSpecに入れるには重すぎるな。Spec X追加と（書き書き）」
- 「あー、これ正式手順として組み込んだ方がいいな」

- **かくして「要件フェイズ」の前に「リサーチフェイズ」が出来ました**

そうだ、Oracle作ろう

- 「んー、実装からHuman TestでどうしてもBlockerが出る。なんで？」
- 「あー、これ『あるべき姿』が実装時点で伝わってないせいだ」
- 「どっかの世界一大きい会社でさんざん言われたなあ、『あるべき姿』」
- 「OK AI。こういうテスト組み込もう。あ、これ Oracle Testって言うのね？じゃあそれで」

- どこかのOという会社とは全く関係がないです。
  あそこにはOpenSolarisを殺された私怨があるので

そしてWaterman 0.1へ

- Research＞Spec Draft＞Spec＞Implement＞Code＞Oracle Test＞Human Test
- Researchは人間が書いたSpec DraftへのLLMレビューでもある
  この時点で「Draftに何を合意として書き足せばSpecになるか」を洗い出す
- Spec Draft＞Specは「人間が決めないと進めようがない事柄」を埋めてのコンパイル作業
  人間がコンパイルされたSpecをレビューする
- Specを「実装計画」と言う名の「人間もLLMも読みやすい中間言語」にコンパイル
  ここも人間レビュー
- Implement＞Code＞Oracle TestはLLMの自己レビュー8割、人間レビュー2割
- Human Testは「UX等、人間じゃないと判断がつかない項目」をテストする
  「顧客受け入れ検査」

偉そうに0.1とか書いてますが

- Waterman 0.0の時点で無意識にやってたことを
  正式に手順にしただけです

ただ、「魔法のハンマー」ではない

- Oracleが最初から作れる問題ばっかりなら誰も苦労しねえよ！
- だからSpecに分解する段階で「 Oracleが作れる問題」まで噛み砕いていた
- 「Oracleが定義できるもの」: 仕様（ Spec1～20）
  「Oracleが定義できるまで揉む必要があるもの」: 構想・願望（　Spec X）
- Oracleが間違っていたら大惨事
  リサーチ段階で「 Oracleは何」はしっかり決めよう。実装時もそこは慎重に

前方リサーチと後方リサーチ

- 前方リサーチ（Waterman)
  解決する課題が単一であり、Oracleが明示できる問題に強い
  「何を作ればいいかはわかってる。でもどう作ったらいいかが不透明」
  こういう問題には向いてる

- 後方リサーチ（TDD/ProtoTyping)
  解決する課題が複合課題であり、Oracleを作ろうとすると発散する問題向き
  「何を作ったらいいかそもそもわからん」
  こういう問題に向いている
  （Spec1～10前後まではまさにこれだった）

では限定的なのかというとそうでもない

- 最初は何だって複合課題
  ただ、ちょっと洗うだけで単一課題に分解できる問題は案外多い
  「Oracle作れるか？」を意識しながら後方リサーチすることで
  後方リサーチ量を削減することは可能（なはず）

- 大事なものは「ウルトラマンメビウス」から学んだ
  「はじめは誰もヒーローじゃない
  　違う形のただちっぽけな星なんだ」
  （ウルトラマンメビウス OPよりの引用）

で、実際どうなのさ

- 設計/文書コストに振った分テスト /レビューコストは下がった
  Human Testが「本当にUX上の問題を洗うテスト」になった
  結果として「30分でテトリス！」は無理になった
  ただ、
  「12時間でホワイトボックス完備、
  　設計/実装意図までアーカイブしたドキュメント付きの
  　エンタープライズ級品質のテトリス」
  が作れるようになった
- 一番大きいと感じたのは
  「手書きなら一時間は悩むことになりそうな設計判断」
  が、リサーチの力を借りて10分で出来るようになったこと

恥さらしその1 (Spec1.md)

- 今見ると「これでよう作れたな」と思いますね

恥さらしその 2(Spec17.md)

- 同じプロジェクトの要件書だと思えませんね

恥さらしその 3(Implement17.md)

- すまない、1800行あるmarkdownを見せて本当にすまない

- codex/Terra君が12回の自己レビューで「もう大丈夫！」と言うまでに
  何度もwhitebox足したり契約違反を修正しています

- codex/Terra君がOracle Testで
  「誤差0.0002まで詰めました。計算順調整でまだ詰められます」
  と大学の研究生みたいなことを言ってます

恥さらしその 4(Research18.md)

- Claudeが勝手に、「過去設計の裏付けとそこで生じた契約漏れ」について分析してくれてます

ウォーターフォールに戻ろうとは言ってないですお

- 俺だってウォーターフォール大嫌いでした
- 写経のごとき「テスト仕様と期待値」を Excelに埋める仕事が好きな人挙手
- ただただ「たかが要件変更でこんな大騒ぎになるのは構造的問題では？」と思っていました
- でも「ウォーターフォールの理想自体が悪いわけじゃない」という気づき
- だから「ウォーターフォールのいいとこだけもらおう」という呼びかけです

まとめ（と予想される反応）

- DAZ StudioもどきをUnityで作ってます
「ふんふん」
- こういうプロセスで作って行きました
「ふんふん」
- なんか開発プロセスが固まったので ”Waterman”と名づけました
「？？？」
- 大事なものは全てウルトラマンメビウスにありました
「なに言ってんだじじい」

質疑

Questions & Discussion

想定される論点

- 自分の領域では Oracle をどう定義するか
- 文書コストが見合う規模の下限はどこか
- チーム開発（複数人）へ持ち込めるか
- モデル / ツールを変えても成立するか
- 「記憶のベンダーロックイン」対策の一つとしての効果

ご意見・反証は歓迎します