Go 言語でスタンドアローンなゲームっぽいものを作るために

おおしまやすし
oshima-ya @ yagoto-urayama.jp
@oshimyja

はじめに

Go 言語を使って NetBSD 上で古風なドット絵ゲームのようなものを作れないかなと試行錯誤してます。

今現在できてること

- NetBSD のグラフィカルコンソール画面を Go 言語から操作できます。つまり自由に画面描画が可能です。
- USB の hid デバイスであるジョイパッドの入力を取得できます(現状は ELECOM の joypad 決め打ち)。
- 上の 2 つを使ってなんとなくスプライトっぽく扱えてドット絵キャラクタを動かしたり画面中弾だらけ(今 800 個)にしたり、なんかが飛んできたりします。
- これだけやって Raspberry PI B+(armv6 700MHz)で超 Low Resolution でだいたい 60fps 維持し CPU 使用率一桁程度。さすがに今時の CPU は速いです。
- 簡単に言えばほとんど絵を表示できてるだけです。

できてないこと

- いわゆるシューティングゲームにするためには最低限次のものが必要です。
  - 敵キャラを自律的に動かすルーチン
  - 当たり判定&スコア
  - マップ
  - シナリオ
  - 他

  よーやっと擬似スプライトができつつあるので Go ならホイホイ書けると思う。書けるといいな。
- USB の hid デバイス汎用化。Device Descriptor をちゃんとパースして各種入力機器に対応すべき。
- 音。何それおいしいの?

ここから駄文。

なぜ Go 言語なのか

まずは以下のような特徴があったから。

- 新しめのネイティブ言語。構文が割と簡単。C 知ってればそれなりにわかる。
- メモリ管理強力（GC あり)だったり並行処理がとりあえず楽(Go ルーチン)だったり。
- 標準ライブラリが多い。LL っぽく使える。そのくせ低レベル処理が可能。System call を直接呼び出せる。
- libc さえ不要。スタティック link の単一バイナリ、kernel さえあれば動く。
- 楽々クロスコンパイル。Windows 上でも作れる(いやほとんど使って無いけど)。
- BSD ライセンス(そこそこ重要。特に static link と合わせて)。
- Plan9! Plan9!(謎)。
- NetBSD でも動く(らしい。超重要←当初)。

次世代のお手軽言語にならんかなぁ、という漠然とした期待があったのが大きいような気がします。

なぜゲームなのか

Go 言語、強力なんだけどやっぱりサーバサイドでの利用やシステムツール等の裏方での利用がメジャーで、クライアントインターフェースは CLI、あるいは REST+Web ブラウザ(JS 等)でクライアントレンダリングが主流のようです。それはそれで現実的だし現代的だなぁと思うわけですが、もちっとお手軽にローカルで遊べないかな、自分がパソコン使い始めたのってやっぱりインタラクティブな画像というかゲームへの憧れだったりするし、初心に戻ってと思ったのです。

なぜ NetBSD なのか

普段使っているからです。Linux でも Windows でもいいんですが、普段使い慣れてるのは NetBSD だからです。
それに Linux とかだと、みんなもう色々やってるでしょ、と。NetBSD では誰もやってないでしょ。たぶん。

なぜ Raspberry PI なのか

NetBSD がそれなりに動いて画面出てちっこくてお手軽でパーソナルな感じだからです。もともと勉強用のコンピュータなんだからちょうどいいじゃない。
このプログラムは当然 PC でも動きます。というか PC で動かしてデバッグしてます(画面使ったりする関係で仮想マシン使ってますが)。ただ OS のインターフェースにバリバリに依存しているため、NetBSD 以外では動きません。そういう意味での Go 言語らしさは全くありません。言語同断ですね。
ただ PC の wsdisplay 画面描画遅いんですよなぜか。RPI 初代より遅い。これ C 言語で書いても。どーゆーことなの。

苦労した(してる)ところ

最初にやろうと思ってからもう 3 年以上かかってます。

- Go 言語を NetBSD/arm(EABI)で動かすために、Go 言語の NetBSD 向けポーティングをデバッグ。これどう考えても間違いなく今まで全く動いてなかっただろ!
  - kernel 由来の問題が多いよ(まだ残ってます。amd64 でも)
  - earmv6hf で vfp 使うとマトモに動かない(今は soft float な GOARM=5 で逃げてる)
  - オリジナルの Go ランタイム(Linux 以外)は armv7+SMP で armv6 までのバイナリ動かない。
  - これだけでだいたい 2 年以上かかってやっと最近 armv7 なら限定的にそこそこ動くようになったところ
- Go 言語的に苦労
  - ハードウェアに近いところは unsafe.uintptr だらけになってとてもよろしくない。こんなの Go じゃない。
  - お気楽 union 的なのがあると実はうれしいんだけどなー。
  - memcpy 的な何かないのかな。slice 相手じゃなくて mmap した先。
  - つまり自分の脳味噌が C 言語や低レベルに犯されててダメ。
  - syscall、意外と未実装多かったり限定的だったりしてめんどい。自分で実装できるのは偉いんだけど。
- NetBSD の機能に由来する苦労
  - 画面解像度変更等の汎用インターフェースが無い。起動前にファームで設定するしかない。
  - 垂直帰線同期のインターフェースが欲しいです。ちゃんと 60fps にしたいです。
  - (kernel のバグのせいらしいが)gdb 使えんとかネットワーク関連の kevent まわり、まだなんか怪しい?

今後について

飽きっぽいので飽きるかもしれません。飽きたらやめます。

ソースについて

今現在まだどこにも出してません。そもそも実験コードなので全然気に入らないところたくさんあって全面的に変わっちゃいそうです。既に 3 回、全面的にリライトしてるし。最近よーやく wsdisplay 操作するところだけは固まってきてるのでそういうところから出していこうかなぁ。でも wsdisplay(4)の IOCTL を呼び出すのと struct 定義しただけだから独自性なんてナッシング。

いじょ。