教育言語Sunaba

平山尚

2013/04/24

何の話か

- プログラミング言語作ってみた。
  - 学生を実験台にしてみたのでその話をする。
  - 2010,2011,2012,2013の4年間、九州大学芸術工学部芸術情報設計学科3年生向け演習。180分x3。

何のために？

- 教育のため。
- 他の用途を完全に切り捨てたい。
  - 砂場で遊ぶのは幼児の間だけ。だから砂場。
  - できるようになったらさっさと他へ行け。
    - せいぜい100時間しか使わない、という前提。

教育言語？もうあるじゃん。

- デカすぎ。難しすぎ。多機能すぎ。
- 経験者の優位が大きすぎる。
- 実用性を捨て切れず教育に集中できてない。
- みんな英語苦手だろ？
- 絵や音出すの面倒すぎ。おもしろくない。

- 試しに作ってみた。

特徴

- 低機能
- 低抽象度→どうしようもなく具体的
- 実行の手間最小
- インストールの手間最小
- 覚えること最小
  - 早見を印刷してきた。
- 日本人に特化

コードとデモ見せる

- こんな感じ。

なぜ日本語？

- 英語みんな苦手。これはマジ。
  - whileがweileだったとしてもたぶんあまり変わらない、というくらい英語に親しみない。
- 英語で名前つけるのしんどい人は多い。
  - プロですらそうだろ？
  - ローマ字にするくらいなら日本語でいいじゃん
- それにアルファベットも使えるので、xとかaとかは使える。

なぜ→で代入？

- a=5は「aと5は等しい」だろ。
  - これを「aを5と等しくせしめよ」と読ませるのは不自然。
  - let a = 5とかしても日本人に対しては無力。
- 「a→5」で「aが5になる」と読んでもらう。
  - 去年は「a←5」で「aに5が入る」と読ませたが、今年改めた。主語が最初だろやっぱり。
  - 本当に「aが5になる」と書けるようにすることも検討中。

なぜpythonインデント？

- 2年前までC形式で{}だった。
  - 全くインデントしない奴が続出した！
  - 「見易さ」という評価基準がそもそも存在しないので、インデントしないのが合理的なのだ！
  - 見にくくて困った経験がなくてはインデントするはずがない！
- なので、python化して有無を言わせない。
  - でも、かなりの数がスペース1個。やっぱり面倒らしい。

「メモリ」って何だよ

- 「メモリ[65050]→990000」で画面に点出ます。
  - IOアドレスを露出させて、ライブラリ関数を不要にした。
  - 「メモリって名前のダイアルが7万個並んでて、うちいくつかは画素やキーボードにつながってる」という説明はどうしようもなく具体的。
  - 変数を理解しなくてもプログラムが書ける。
  - 「ポインタがわからねえ」とか言わせない。本当にただの整数だから。メモリ[a]のaがポインタ。

メモリを使う例

メモリ[0] → 0
メモリ[0] < 100 なかぎり
　メモリ[60000+メモリ[0]]→ 990000
　メモリ[0] → メモリ[0]+1

これで線が書ける。
なお、15分待ってこれが書けた学生は3/40人。
2年間Cやらjavaやらやってたはずなんだが。

「メモリ=マス目」は有害

- マス目なら「空白」がありうるが、メモリに空白はない。
  - この言葉が間違ったイメージを焼きつける。
- マス目は普通2次元に並ぶが、メモリは1次元。
  - このイメージも深層に焼きついて禍根となる。
- 「ダイアル」の方がマシ。もっといいメタファーがあればいいのだが。
  - ダイアルってもう絶滅寸前だからな…

変数→名前付きメモリ

- 変数って言って文字列入れたりするのを変だと思わない人は、頭がプログラミングに毒されている。
  - 「変数」って言うんだから数だろ！
  - 抽象的すぎてよくわかんない。どこにあんの？
  - スコープの概念がついてくると厄介。
- 「名前付きメモリ」なら間違いようがないだろ？「どっかの番号に名前つけただけ」

関数→部分プログラム

- y=f(x)と何の関係があるわけ？
  - 普通はそう思う。
  - 最初に関数を使う動機はプログラムの使い回しであって、数学で言う写像ではない。
- 「部分プログラム」なら間違いようがない。
- 引数や戻り値は必須ではない。なにせ「メモリ」を使えばやりたい放題である。
  - 遠からずその恐ろしさを思い知るのだが。

型は整数だけ

- 型とか初心者には無理ゲー。
  - 整数だけですら手に余るのに。
  - 文字列を使うようなプログラムは書かないと切り捨てた。
    - がんばれば書けるけどな。一応デバグ出力もある。しかし文字列リテラルや文字定数がないので現実的には無理。文字定数くらいは入れてもいいかなあ。

1バイト=32bit

- というか、「メモリには-10億から10億くらいまで入るから」としか言ってない。
- ビットとかバイトとか、それはプログラミングの本質ではない。本質は「演算器とメモリ」だ。
  - 2進法であることすら外に露出してない。
  - ビット演算、論理演算はない。
- そのうち世界のコンピュータがみんな64bitになったら64bitにしたい。

制御構造

- 「なかぎり(while)」と「なら(if)」のみ。
  - continue,break,return,switch,goto,for,foreachなどはない。
- ifすらなくても書けるのだが、さすがに辛いので、構文糖として用意してある。
  - ただし、elseif,elseはない。
- 十分書けるし、その方が訓練になる。
  - むしろバグりにくい印象すらある。

and?or?作れよ。

if (x > 0) * (x < 100) #and

while (y = 0) + (y = 1) #or

- 論理演算子なんていらなかったんだ！！
  - 本質じゃない。あれは単なる構文糖(短絡評価の問題を置くとすればだが)
- この仕様でifとwhileの動作の本質を理解でき、「演算子」の概念も理解できる。

実装

言語

- C++/CLIなDLL + C#なGUI
  - 実行側は速度が必要
  - コンパイラも速度が必要
  - GUIは速度より楽に書けることが重要

実行まで

- コンパイラが仮想マシン向けアセンブラを吐く。
- アセンブラをアセンブルしてバイナリ生成。
- バイナリを実行側に送りつけてから実行。
  - GUI側にコンパイラがあり、TCP/IPで実行側に送れる作りにはしてある。Androidで実行とか考えて。でもまだない。

VM

- スタックマシン。
  - レジスタは、PC、SP、FPの3本しかない。
- 命令は18個。
  - i,add,sub,mul,div,lt,le,eq,ne,ld,st,fld,fst,j,bz,call,ret,pop
  - ほぼギリギリ。あと2つしか減らせない。
  - ビット演算、論理演算はない。
  - 関数コールは最初はなかったが、後から入れた。

メモリ空間

- 0-39999: 自由領域+プログラム
- 40000-49999: スタック
- 50000-59999: ビデオ以外のIO
- 60000-:ビデオ
  - だから65050番に990000を入れると真ん中に赤い点が出る。
  - デフォルトでは100x100。100万色。色も10進。
  - グラフィクスはOpenGL1.1。CPUでテクスチャに描くだけ。2対応ならシェーダで色変換したいが。

メモリ狭くね？

- 番号を打ちこむ関係上桁は減らしたい。
  - 100x100あるので5桁にはなる。なので5桁。
  - 100x100の画素に馬鹿正直にストアするプログラムが3万命令あるので、それが入るようにはしたかった。→デモ
- スタック、IO、ビデオは万の位を変えたかったので、40000、50000、60000となった。アドレス直打ちが前提なので中途半端にしたくない。
- 案外入ります(32bit入るので)

音鳴るよ。

メモリ[55006] → 440 #周波数
メモリ[55009] → 10 #減衰率
メモリ[55012] → 5000 #強度。これでキック。

- 正弦波3チャネル用意しておいた。
- あとノイズ1チャネル足すかなあ。
- 実装はsin,cosを使わず調和振動子シミュレーション。ノイズも乗りにくく高速。XAudio2。

展望

本が出る

- 本が出る予定。
  - もうとっくに出てたはずなんだけどな…
  - これでテトリスを作るまでを徹底的に説明する。
  - A5、500ページ超。1500円を予定。
  - 「これでダメならお前はプログラマは無理」と誰もが言えるような最終兵器にしたい。
    - できる奴は完全無視。徹底的にしつこく丁寧に説明し、「これでできないなら仕方ない。本当に向いてないんだ」とあきらめられる出来にしたい。

野望

- 社内でも教育/選別に使いたい。
  - 暗記とパターンあてはめが通用しないので素の頭の良さが一瞬で露見する。
  - 逃げようもなく自力で実装することを要求される。
    - google力、コピペ力などは一切役に立たない。
    - なにせprintfすらない。デバグ？画面に点出せよ。
- 選別するにも、訓練するにも効果的であると信じる。
  - 英語風文法もあるよ！(if,while,def,const...)

Sunaba++

- 本が売れたら続編でオブジェクト指向編
  - VMを共有して、言語だけ変える。
  - クラス、メソッド、型定義、仮想関数、継承あたりを足す。
- 「売れる」というラインは数万部なのでだいぶ厳しいんですが…
  - なにせ安いので5万売れないと前作のもうけを超えない。10年かけて10万部が目標。

mac版....ッ!!!

- 誰か移植してよ...
  - 芸術工学部と言うだけあって学生の大半がmac派であり、家でやりたくてもできない。
  - mac買えって言われてもなあ。ObjectiveC面倒くさすぎる...
  - とりあえずOpenGLにしといたからさ。あと極力ヘッダ実装化してプラットフォーム依存部は分けておいた。
    - XAudio2をOpenALにしといたらもっと楽か?