Common Lisp による高速ゲーム開発入門

発表者:さくらんぼ

目次

- 自己紹介
- Lispを選んだ理由
- Common Lispのゲーム開発環境 & 能力
- 作業結果が簡単に確認できる環境
- 速度
- まとめ

自己紹介

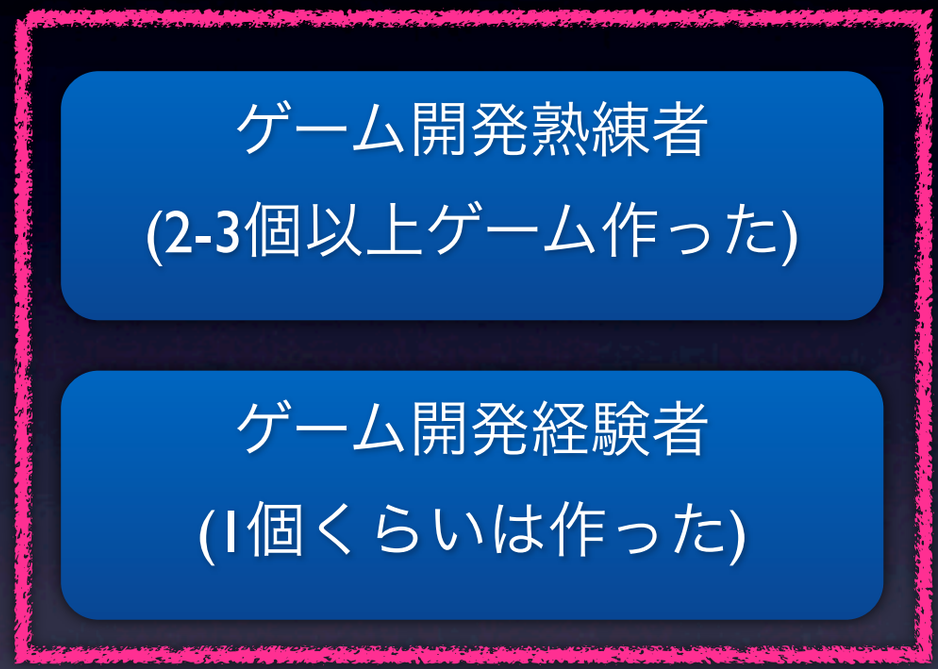
- おなまえ：さくらんぼ
- 同人サークル さくらんぼ亭 (代表)
- 絵(ドット絵、普通の絵)
- Twitter(@lambda_sakura)
  - 卑猥乙らしい
- C/C++ or Ruby → Common Lisp
- パチュリーかわいいよパチュリー

さくらんぼ亭について

- メンバーは5人
  - プログラマ:2人
  - ドッター:2人
  - 絵描き:1人
- 歴史
  - それ以前:小物を作ったり
  - 2006くらいから活動開始
  - 2006:東方掃除紀
  - 2007:東方萃汜紀
  - 2008:東方潜宅紀
  - 2009-2010:東方吸闘紀

対象者

- C/C++のゲーム開発経験者
- **効率よくゲーム開発したい**
- **Lisper**



ゲーム開発未経験

Lispを選んだ理由

自分の最近の課題

- ゲーム開発にあてる時間がない
    - 開発者の多くが社会人
        - C/C++はバッドノウハウが多くて辛い
    - メンバと作業時間帯が合わない
        - 作業結果がゲームに反映されるまでにタイムラグが発生

こんな環境が欲しい

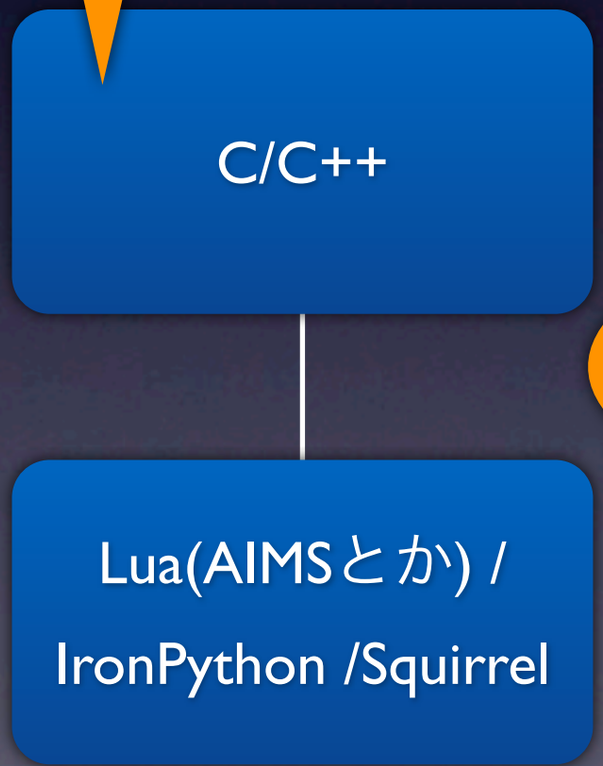
- 簡単に作業結果を確認できる環境
- 速度もそこそこ
- コーディングが効率的

候補として挙がったのが

Common Lisp

ちなみに組み込み言語は？

C/C++はお腹いっぱい



- 全部1つの言語でやりたい！
- C/C++は使うの止めたかった

2つも言語覚えるの?
速度とか心配

Common Lispをゲームで使う利点

Common Lispをゲームで使う利点

- チーム全体の開発効率向上
  - LL(Ruby,Python,Perl)と同程度の記述能力
    - コレクション(ベクタ、キュー、ハッシュなど)、クロージャ、関数型を提供
  - コンパイルが不要
  - DSL (Domain Specific Language) を構築して効率良くゲーム開発
    - Common LispはDSLの構築が容易
  - C/C++に若干負ける程度の実行速度

こんなん作れるよ！

参考

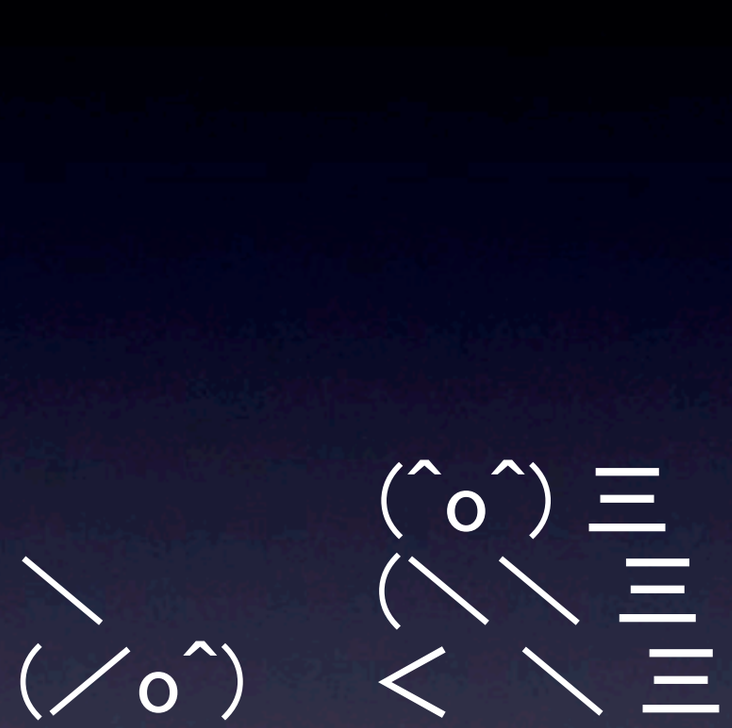
- 実作業時間1ヶ月強程度でした

Common Lispの誤解

- インタプリタ
- 関数型で記述
- 代入がない
- 逐次実行がない
- 型チェックがない

- Emacs?
- GCがあるから遅い

(自分の職場などで良く言われる事より)


へ(^o^)へ いいぜ
(^o^) てめえが何でも
思い通りに出来るってなら
(^o^) 三
(^o^) 三
(^o^) 三
まずはそのふざけた
幻想をぶち殺す

Common Lisp の実際

- コンパイルして実行する
- LLと同様に逐次で書く
- 型付けは動的ですが、型チェックは厳しい
- ネイティブのバイナリを生成
- 高速なGC(Garbage Collection)

Common Lispのゲーム

開発環境 & 能力

環境について

- PerlのCPAN(http://www.cpan.org/)みたいなもの
  - quicklisp
- ゲーム用ライブラリ(DirectX)
  - lispbuilder-sdl
- IDE(visual studioとか)
  - Emacs + SLIME

  - Eclipse + cusp

導入方法

- 公式サイト
  - http://code.google.com/p/lispbuilder/wiki/DownloadInstallationIntro
- 私のblog参照
  - http://d.hatena.ne.jp/sakura-1/
- Wikiつくってまとめ始めているので参考にしてください
  - http://light-of-moe.ddo.jp/~sakura/hiki/

lispbuilder-sdl使ったコードの例

ウィンドウを表示、Escキーが押されると終了するプログラム

```
(sdl:with-init ()
 (sdl:window 640 480 :title-caption "Window Test")
 (sdl:with-events ()
   (:quit-event () t)
   (:key-down-event ()
    (sdl:push-quit-event))
   (:idle ()
    (sdl:clear-display sdl:*black*)
    (sdl:update-display))))
```

ウィンドウ表示

キーが押されたときの処理

1フレーム毎の処理

ゲーム開発に使えるCommon Lispの機能

- ハッシュ、ベクタ、リスト、キューなどよく使うものは言語が提供
  - コレクションの機能はCommon Lispの機能で十分
- CLOS(Common Lisp Object System)
  - クラス、総称関数といった機能を提供
    - オブジェクト指向もCommon Lispの機能で十分

動的ロード

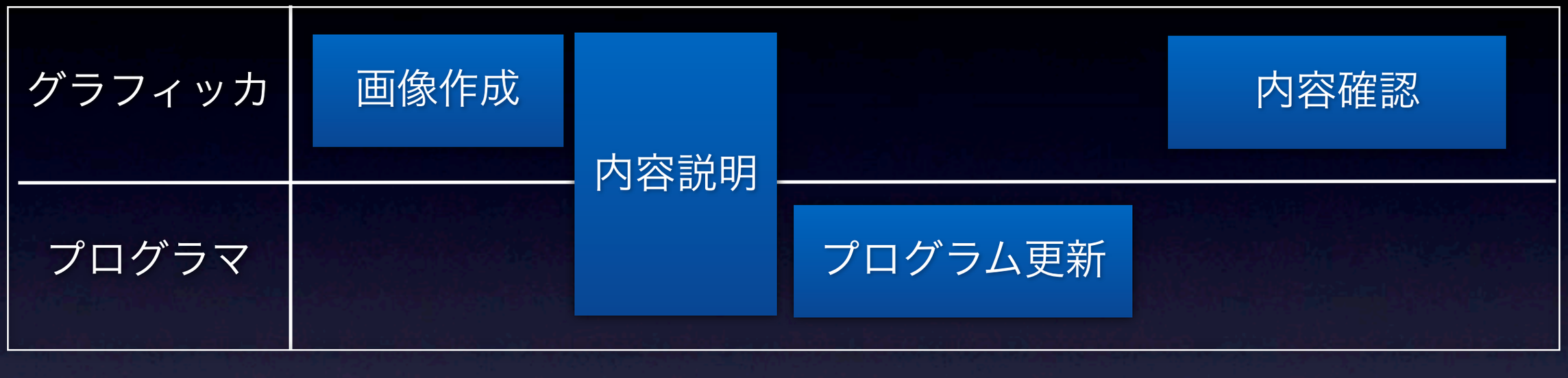
- 動作させながら特定の関数だけ更新
- 変数の更新も可能
- 動的束縛を利用すると参照の更新も可能

動的ロードのデモ

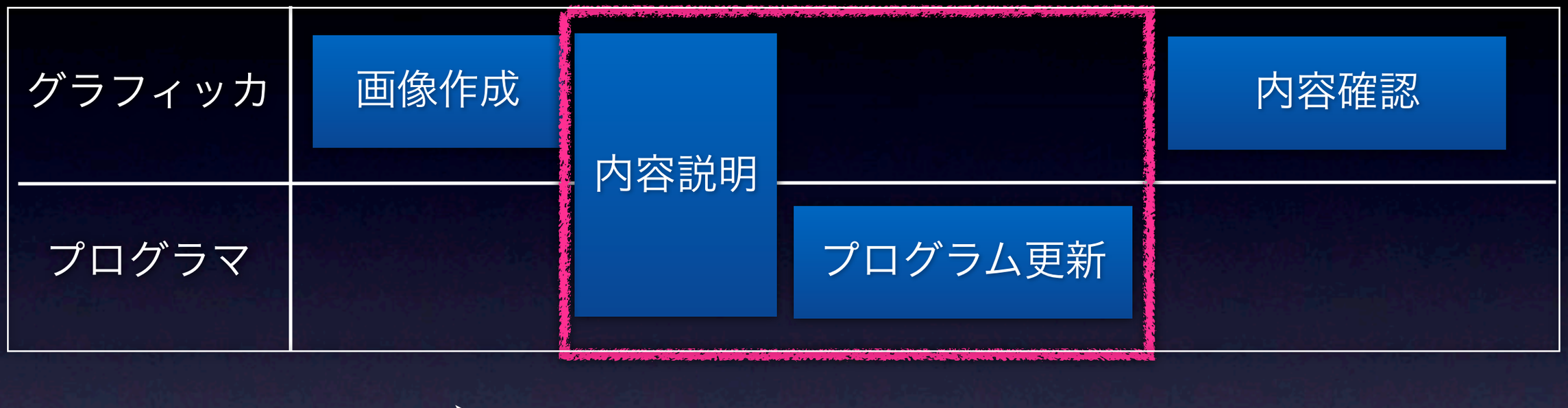
- 関数の内部をリアルタイムに書き換えて実行してみせます

作業結果が簡単に確認できる環境

普通のゲーム開発の流れ



普通のゲーム開発の流れ



グラフィッカが作業結果を把握するまでにかかる時間
この時間を減らしたい

DSL(Domain Specific Language)がある開発の流れ

グラフィッカ	画像作成	DSL記述	内容確認
プログラマ			

時間の進み

グラフィッカが自分で記述して確認！

プログラマの作業が軽減

DSL(Domain Specific Language)がある開発の流れ

グラフィッカ	画像作成	DSL記述	内容確認
プログラマ	画像作成	DSL記述	内容確認

時間の進み

グラフィッカが自分で記述して確認！

プログラマの作業が軽減

むしろ転職してグラフィッカに

DSL構築の手法

- DSLのタイプは大きく分けて2(+1)種類
  - 外部DSL：独自言語をテキストとして読み込んで解釈
  - 内部DSL：言語内に別言語を構築する
    - Rubyなどの場合とCommon Lispの場合で大分雰囲気が違う

構築方法比較


外部DSL
内部DSL
DSL層
DSL層
DSL層
パーサ
言語層
(マクロ)
言語層
言語層
言語層
Common Lisp

マクロとDSL

- Common Lispでは内部DSLはマクロで構築する
  - マクロは任意の式を別の形のCommon Lispの式に変形する機構
    - 変形の処理にCommon Lispの関数を呼び出せる
      - 条件に応じて変形する形を変えることが可能
      - ただの文字列置換ではない

DSL比較表

		作成者	利用者
外部DSL	利点	なし	DSLの文法は自由
	欠点	構文解析器とか作るの大変	なし
内部DSL	利点	構文解析器不要。言語の機能を利用できる	なし
	欠点	なし	文法は言語の文法に制約。記述する側に言語の知識が必須
内部DSL (Common Lisp)	利点	構文解析器不要。言語機能も利用可能	DSLの文法は自由
	欠点	上級者向け。プログラマに慣れが必要	なし

DSL比較表

		作成者	利用者
外部DSL	利点	なし	DSLの文法は自由
	欠点		なし
内部DSL	利点	構	なし
	欠点	なし	文法は言語の文法に制約。記述する側に言語の知識が必須
内部DSL (Common Lisp)	利点	構文解析器不要。言語機能も利用可能	DSLの文法は自由
	欠点	上級者向け。プログラマに慣れが必要	なし

手軽に
グラフィッカーの望むDSL文法
で記述させることが可能

LispのDSLの例

```
(Recipe
 :name "Milky Gravy"
 :consists
 (add
    1 lb “flour”
    200 grams “milk”
    1 gram “nutmeg”)
  (steps
     "mix ingredients"
     "cook for some amount of time"))
```

速度

それでは

みなさんお待ちかね！

プログラムファイト
レディーゴー！

闘ってもらう人たち

- ディフェンディングチャンピオン
  - C/C++
- 挑戦者
  - Common Lisp(SBCL)
  - Common Lisp(Clozure CL)
  - Python(2.6.1)
  - Ruby(1.8.7)

2種類の比較をするよ！

- 配列(vector)に1000万個を挿入してみた
- オブジェクトを1000万個作ってみた

結果

	挿入	オブジェクト生成
C/C++	0.614	1.317
Common Lisp (SBCL)	0.672	1.172
Common Lisp (CCL)	1.074	30.186865
Python	計測不能	計測不能
Ruby	4.837042	224.851941

単位:秒

うーん

- Pythonおそすぎじゃねw
- Rubyも遅い...。
- SBCL早すぎワロタ
- CCLはなんでオブジェクトの作成だけおそいのだろう

速度まとめ

Common Lispは

ゲーム作るには

十分早い！

Common Lispの課題

- 情報が英語
- Windows環境のサポートが弱め
- 処理系が多くて混乱する
- ファイルとかディレクトリの扱いが他の言語とたまに違う

まとめ

- Common Lispで開発すると
  - 言語機能でラクチン
  - DSL構築もラクチン
  - 速度も十分
- みんなでCommon Lispでゲーム開発！

ご清聴

ありがとうございました

Q & A

覚えてる範囲で

- SDL + Common Lisp って相当マルチプラットフォームなのでは？
- Yes. 開発者はMac/Linuxで開発。配布はWindowsという形態を実現している

- SDLの2Dの描画機能では貧弱すぎでは
- OpenGLを叩けるので、描画機能に不満があればそちらを呼べる

- グラフィッカーの望むような文法のDSL
を提供するのは困難なのでは
- Common Lispのマクロなら十分実現可能である

- オススメのLisp実装は？
  - trunkのClozure CL
    - 64bit Windowsでも動作するので。