案ずるより産むが易し！
～Game Monkey Script と Xtal によるガンシュー開発～

株式会社バンダイナムコゲームス
湊　和久　　宇藤慶彦

本日はスクリプトドリブン開発
の
事例紹介をします

スクリプト言語活用の事例紹介:

CEDEC 2008

ゲームにおけるスクリプト言語の実情

続・ゲームにおけるスクリプト言語の実情

CEDEC 2009

RPGやADVの事例が多かった

本日のスクリプトドリブン開発事例は

業務用ガンシューティングです

デッドストームパイレーツの特徴

- 業務用の大型機
- PlayStation 3 をベースにした基板”システム357”を使用

デッドストームパイレーツの特徴

秒間
60フレーム

操舵輪
ミニゲーム

可動筐体

船
（乗り物）

事の起こり

いきなりですが、質問です

ゲームの面白さを作りこむのは
誰でしょう？

1.企画（ゲームデザイナー）
2.プログラマ
3.アーティスト

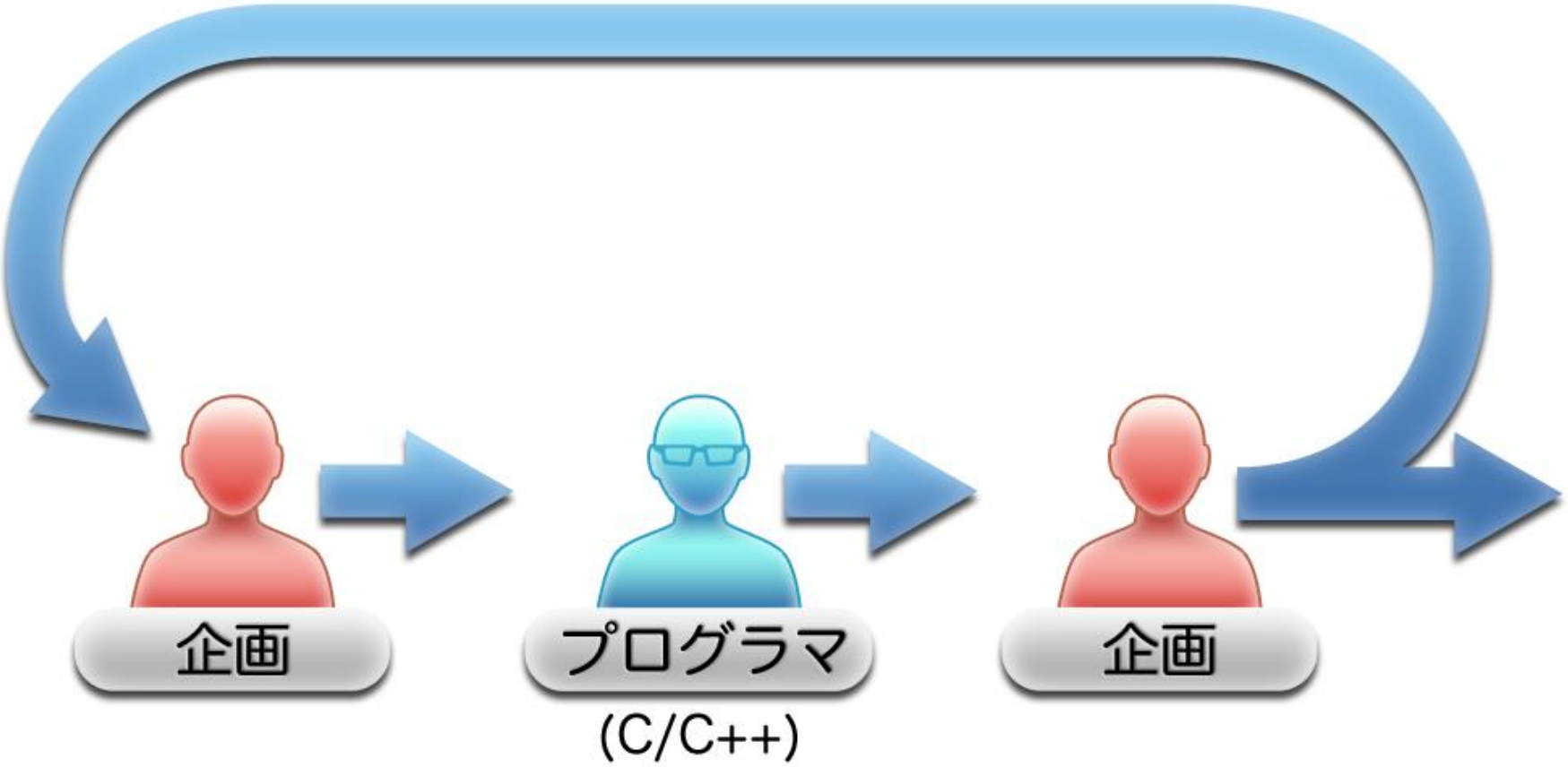
ゲームの作り方の一例

1. 仕様書を作ってプログラマに渡す
2. 出来てくるまで何日か待つ
3. 触って、修正点を見つける
4. 1へ戻る
5. たまに御破算になる

→**複数職種**によるイテレーション

ゲームの作り方の一例

複数職種によるイテレーション



[参考]Naughty Dogのゲームデザイナーの職責

Current Jobs

Administration

Job Title

- User Experience Designer
- Web Developer
- Web Designer

Art Department

Job Title

- Dynamics Artist
- Lighting Artist
- Marketing Artist
- Texture Artist
- Environment Artist
- Character Artist

Audio

Careers GO BACK

Game Designer

Responsibilities

- Responsible for the planning, level layout, setup and tuning of single-player levels, from high concept to object placement and scripting
- Responsible for designing and producing engaging and fun third-person action gameplay and levels
- Act as producer for levels you design, as well as other parts of the game, collaborating across disciplines to get work done and clear dependencies, ensuring deadlines are met, and championing aspects of the gameplay
- Work directly with artists, programmers, animators and other game designers to contribute to the vision of the game
- Responsible for level layout by creating simplified level geometry and performing extensive play-testing and iteration
- Work with programmers to develop tools on an ongoing basis

Requirements & Skills

- Experience designing single-player levels for console games
- Minimum 3 to 5 years of games industry experience
- Strong methodology and problem solving ability and with a focus on creative, fun and innovative solutions
- Strong and effective communication skills
- Exceptional team player with the ability to collaborate without losing sight of the gameplay vision
- Willingness to take design direction when offered

[参考]Naughty Dogのゲームデザイナーの職責

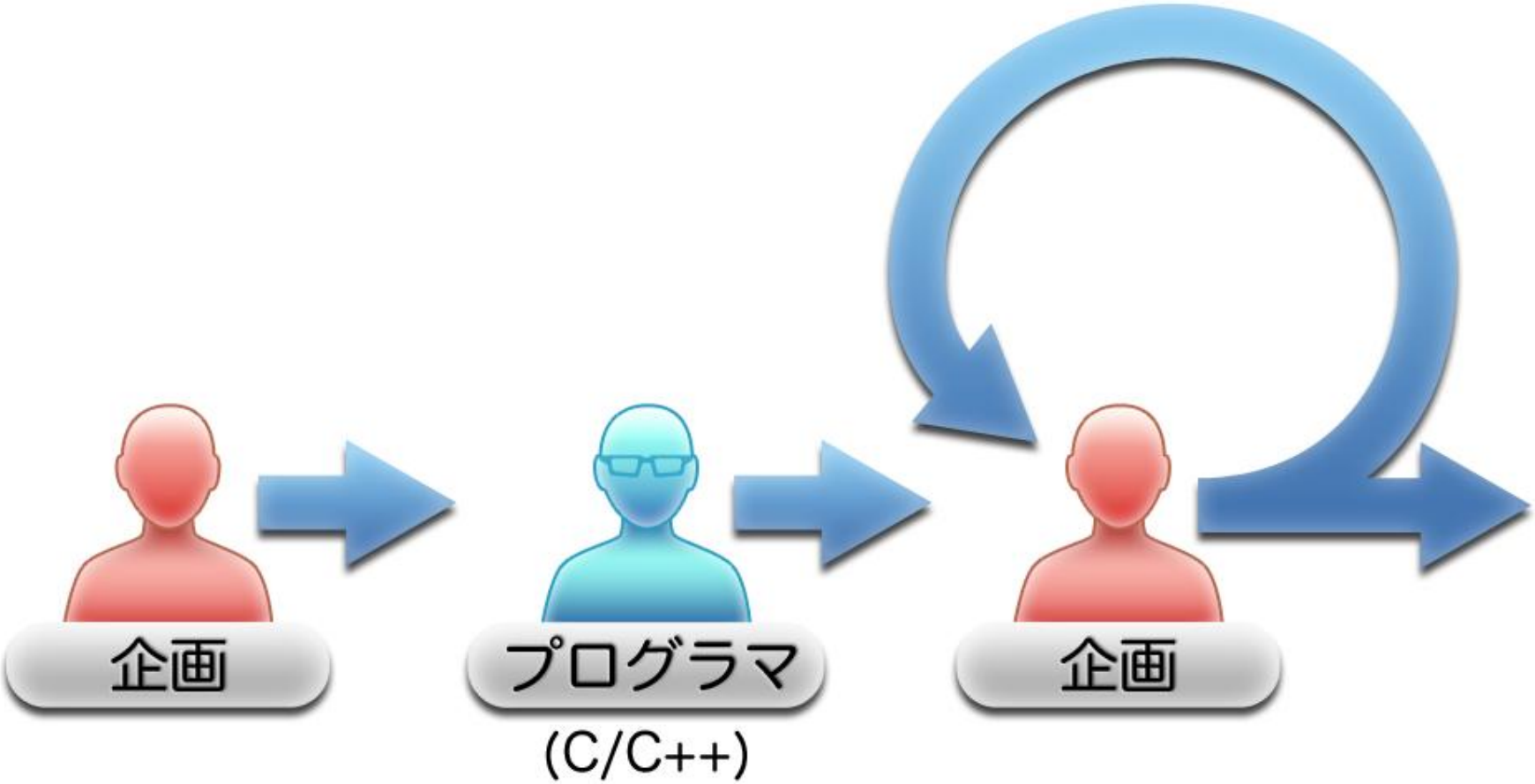
- ゲームレベルのプランニング、レイアウト、セットアップとチューニングの責任
  - コンセプト作成からオブジェクト配置＆スクリプティングまで及ぶ
- 魅力的で面白いゲームプレイとレベルを設計し、製作する責任
- シンプルなレベルジオメトリを作成し、広範囲のプレーテストとイテレーションを実行する責任。
- 担当するレベルのプロデューサーとしてふるまうこと！

パイレーツチームの原則的な方針

“ゲームの面白さを作りこむのは企画である”

（by パイレーツ　テクニカルディレクター）

パイレーツチームの原則的な方針



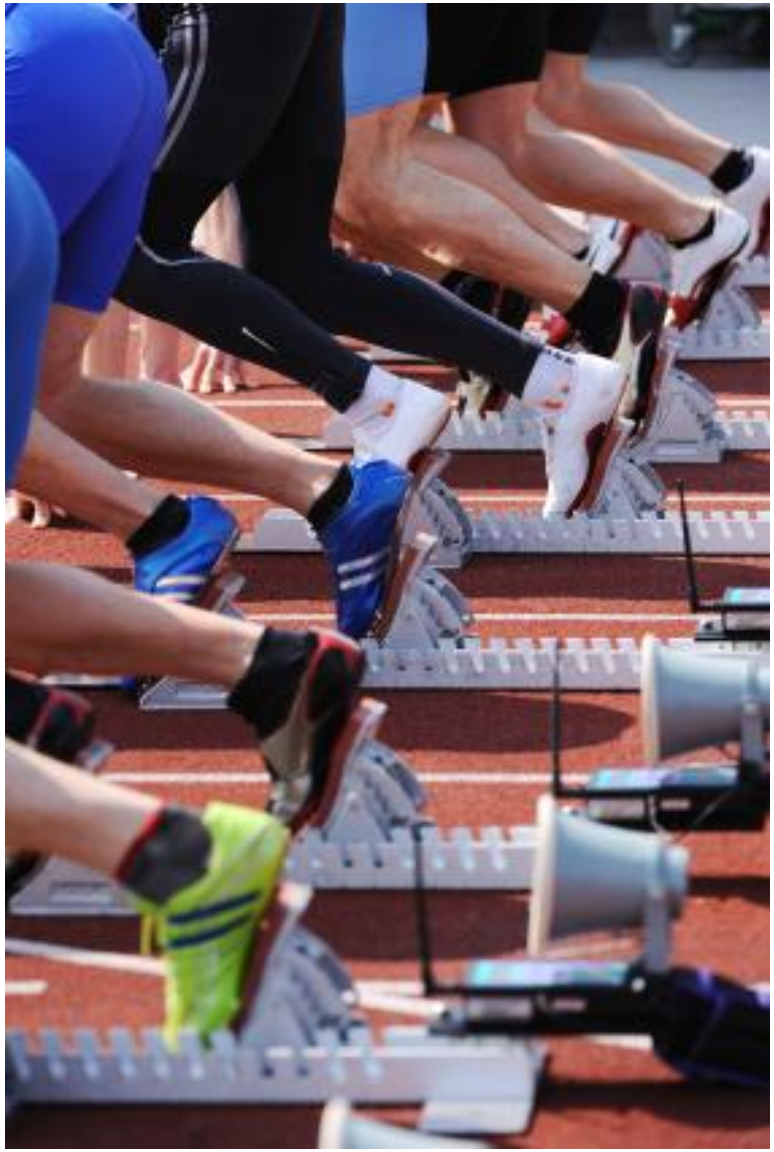
パイレーツチームの原則的な方針

- 従前の独自のスクリプト言語 SDL ではなく、フルスペックの汎用スクリプト言語 Game Monkey Script を採用
- 高レベルのゲームプレーコードはすべてスクリプトでプログラムすることを目指すシステムにした

パイレーツチームの原則的な方針

しかし、そのシステムを受け取る企画側には
スクリプトプログラム経験者は
ほとんどいなかったのでした。

パイレーツチームの原則的な方針



- プログラムチームは自由度の高いシステムを開発して、企画チームに渡す
- 企画チームはスクリプトプログラムで開発できる人員をトレーニングしつつ体制を整備する
- 協力して秒間60フレームを実現する

システムづくり

質問です

ガンシューって
ざっくり言って**どんなゲーム？**

ガンシューに必要な最低限のもの

“狙って撃つ”
ことに集中できるFPS

ガンシューに必要な最低限のもの

レールに乗ったカメラと…

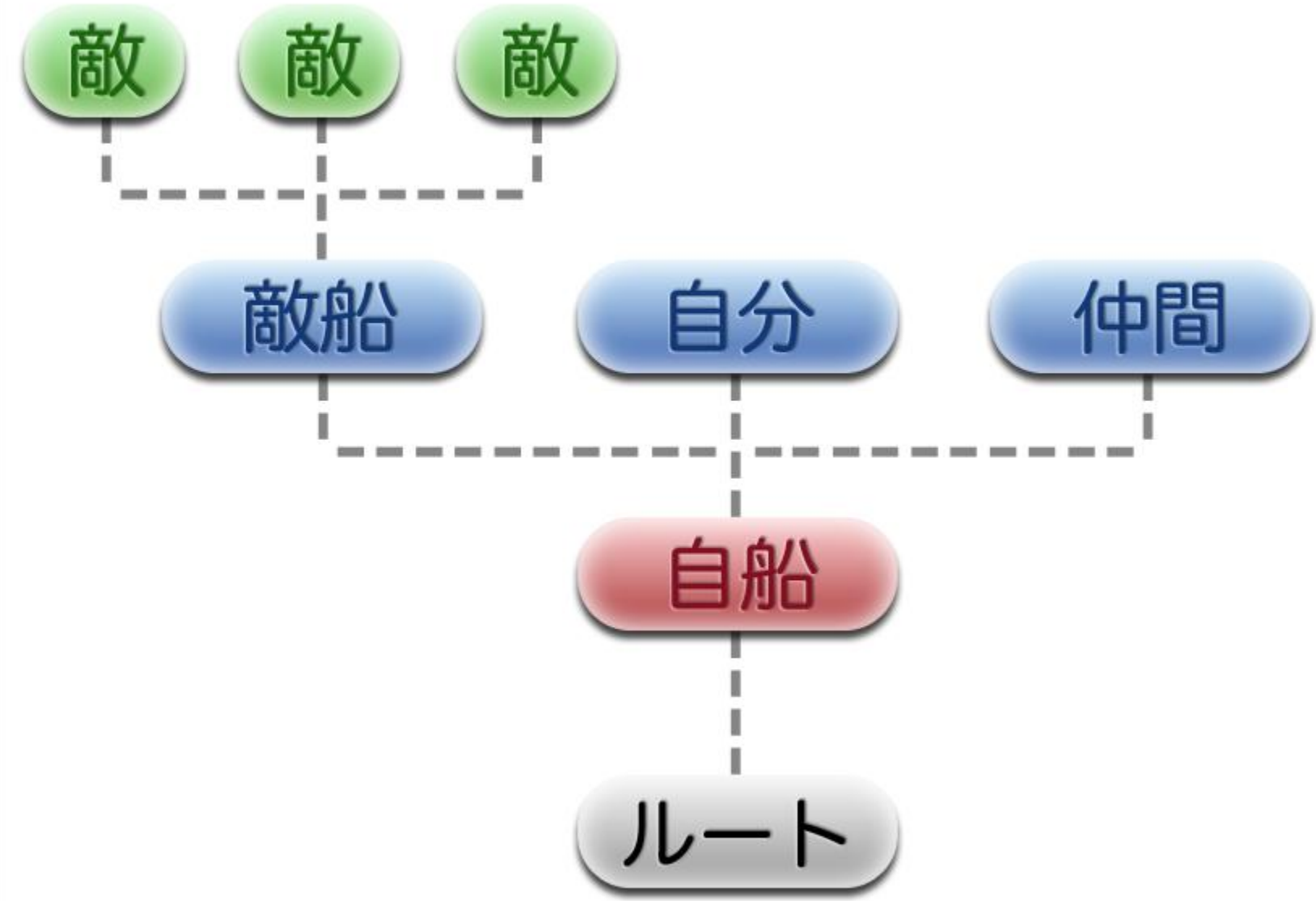
ガンシューに必要な最低限のもの

オブジェクトを配置して
操る仕組みがあればOK

基礎を支えるシステム」

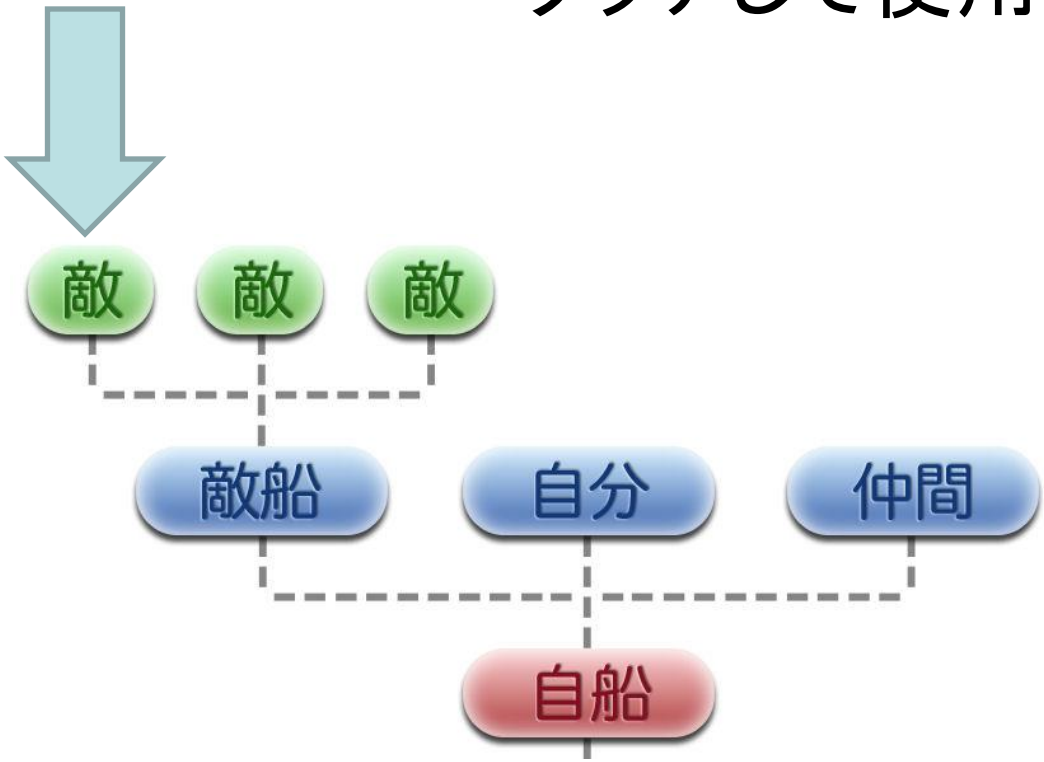
- シーングラフ
- ゲームオブジェクト
  - モーション
- ノードアニメーショントラック

シーングラフ



ゲームオブジェクト

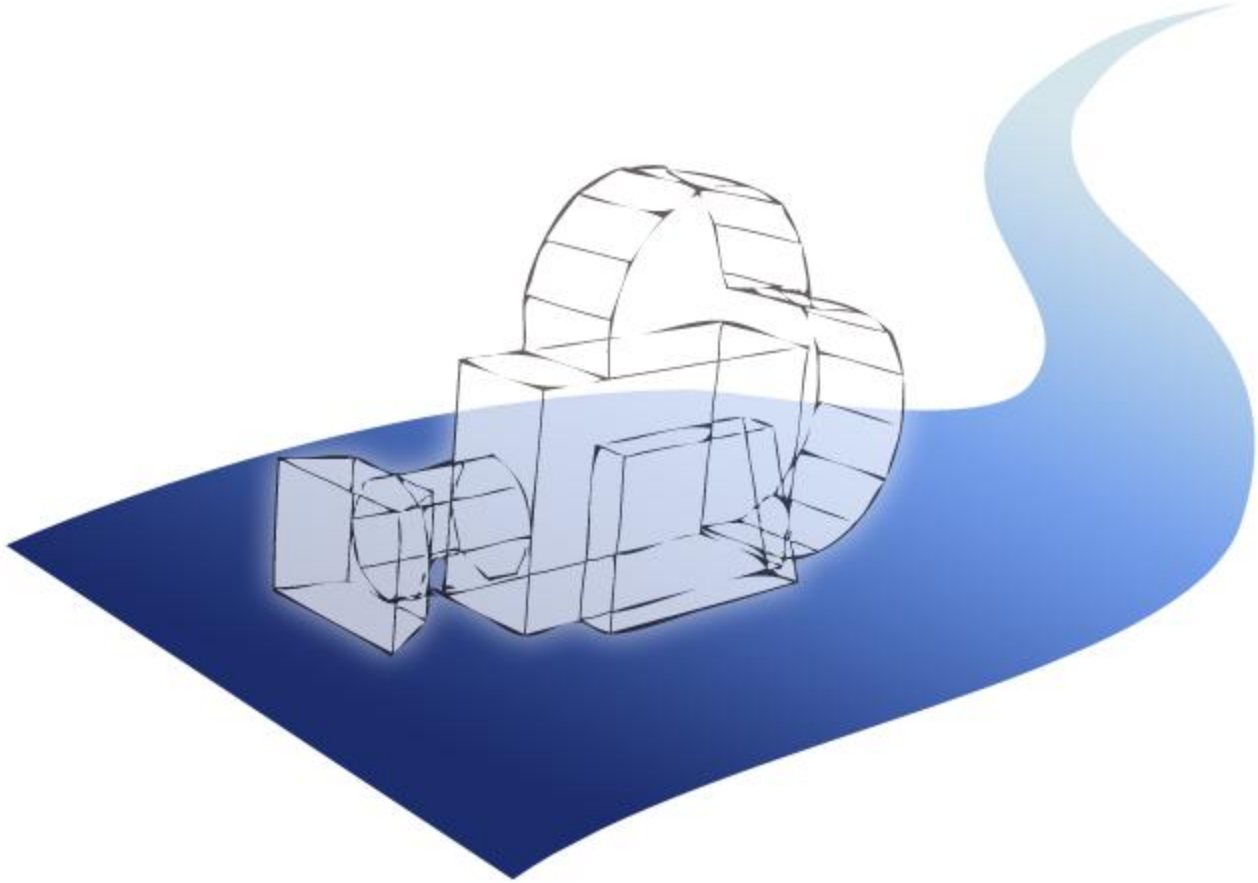
- キャラ、群衆、カメラ、ギミック、エフェクト、音源、マップなど
- スクリプトで生成し、ノードにアタッチして使用する



モーション指定によるキャラ制御

- 起き上がる
- 剣を装備する
- 前へ歩く
- 前へ歩く
- 前へ歩く
- 止まる
- 剣を振りかざす

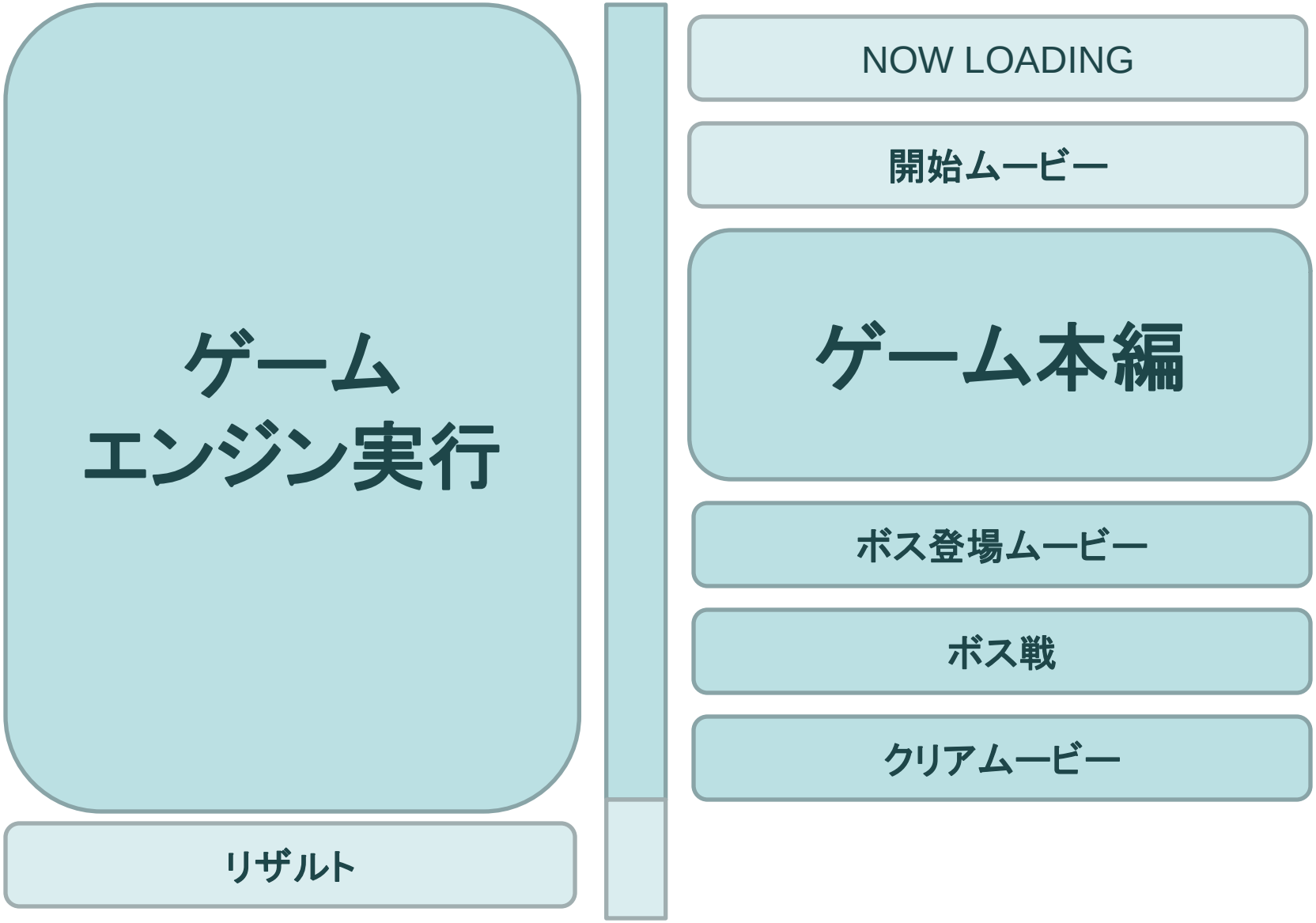
ノードアニメーショントラック



そのほかスクリプトから操作できるもの

アセットの ロード/アンロード	ムービーの 操作	HUDのON/OFF
カメラの操作	群衆の操作	数学計算
字幕の操作	サウンド/音声 操作	可動筐体操作

ゲーム中のシーケンスはスクリプト記述



企画側の取り組み

作業体制

ロケーションテスト前

Stage1　　メインスクリプタA＋サブスクリプタA

Stage2　　メインスクリプタB＋プログラマA

舵輪ゲーム・筐体可動　メインC＋サブB＋プログA

ロケーションテスト後

Stage3　　サブA＋サブC＋プログラマB

Stage4　　メインA＋サブD

Stage5　　メインB

舵輪ゲーム・筐体可動　メインC＋サブB＋サブE
＋プログラマA

作業体制

- 基本的にはメインとサブのスクリプタで1ステージを完成させる。

  ただし、シークエンス・処理が難しいシーン・ミニゲームはプログラマにスクリプトを書いてもらい、スクリプタが調整・修正を行う。

- ちなみに、スクリプタの経験はというと……

メインA（自分）：モーションヘッダを2年弱。プログ経験なし。

メインB：モーションヘッダを1年半。プログ経験なし。

メインC：今回が初のプロジェクト。趣味でCを。ハード設計・サウンド管理も並行して行う。

サブA：過去に別のガンシューを担当。

サブB：テキストエディタを触るのが初めて。

スクリプトで行ったこと

- キャラ・オブジェクトの配置、モーション指定
- カメラトラックの指定
- ロード・アンロード処理（処理落ち対策含む）
- ゲーム中のムービー・チュートリアルの差し込み
- ゲームの難易度自動調整システム
- スケルトンの自動待機・攻撃システム
- ミニゲーム（舵輪・宝箱選択など）全般
- 可動筐体制御

これからの課題

- プログラム経験のない人が多く、人によってスクリプトの書き方に差異があり読みづらい

  →教育係が守らないといけない基本的なことを徹底する。ペアスクリプティングを用いる。

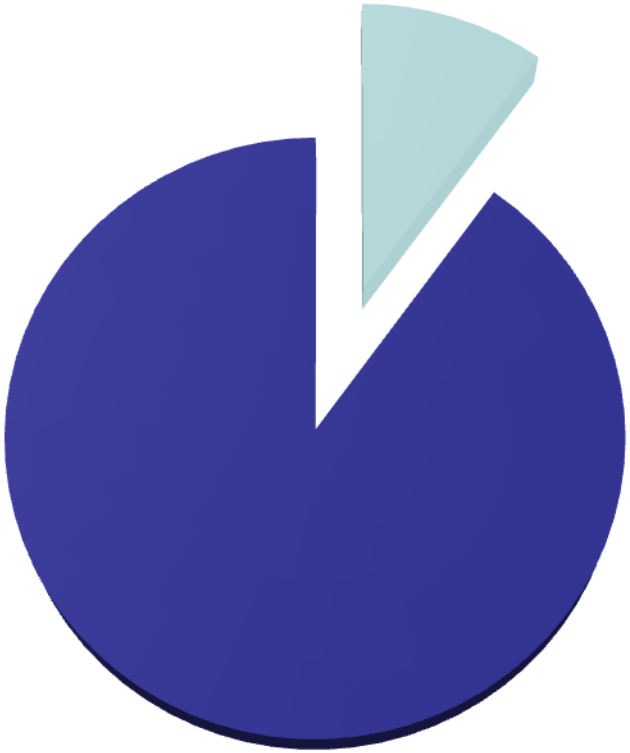
- 実験用のモーション作成を行うことが出来なかった

→MayaやXSIなどの基本オペレーションが出来るスクリプターがいると助かった。

秒間60フレームを目指して

60フレームを狙うには…

スクリプトエンジンのCPU予算8%



GMSはこんなときに重くなる

- 実行中のスレッドのコードが長い、もしくは重いビルトイン関数を呼んでいる
  - yield()で中断を！
- スレッドの数が多い
  - 塵も積もれば山となる
- インクリメンタルGCが稼働中である
- フルGCが動いてしまった

GMSのCPU負荷を減らすには

- スレッドの負荷を軽くする
- GCまわりの設定を工夫する

まずコルーチンを何とかしよう

動いてるスレッド10～20本

まずスレッドを何とかしよう

調べてみると
ゲームオブジェクト等の状態変化待ちが多

```
while ( gobj_get_ status(entity) == HOGE ) {
        yield();
};
```

ブロックとシグナル

- GMSの機能
  ブロックと**シグナル**による
  制御を活用することにした

ブロックとシグナル

- 指定したシグナル（数値や文字列）が届くまで、コルーチンを休止する
- ブロック状態のコルーチンは処理負荷が**ほぼゼロ**
- 実行中のコルーチン数を**数本程度**にすることができた
- かなりの負荷軽減効果

ちなみに

- 企画側にゾンビスレッドの削除も徹底してもらった
- キックした側のスレッドで、用済みのコルーチンをkillThread()

GCの負荷制御

- **インクリメンタルGC**と**フルGC**の2つがある
- **ソフトメモリ制限**と**ハードメモリ制限**の2つの設定を使ってGCの実行タイミングを制御する

GCの負荷制御

- インクリメンタルGC
  - 1フレームあたりの負荷を制限できるGC
  - 使用メモリ量が**ソフトメモリ制限を超えると**動き始め、数フレームにわたりタイムスライスで処理を行う
- フルGC
  - とても重いGC
  - 使用メモリ量が**ハードメモリ制限を超えると**動き、1フレームで一気に処理を行う

GCの負荷制御

- ハードメモリ制限は単純にメモリ予算に従って設定
- ソフトメモリ制限はポリシーに従って設定

ソフト制限を小さめに設定すると

- インクリメンタルGCが働きやすくなり、常時負荷が上がるが、フルGCが起こりにくくなる

ソフト制限を大きめに設定すると

- インクリメンタルGCが働きにくくなり、常時負荷が下がるが、フルGCが起こりやすくなる

GC制御

インクリメンタルGCとフルGCの稼働回数を
モニタできるようにする

GC制御

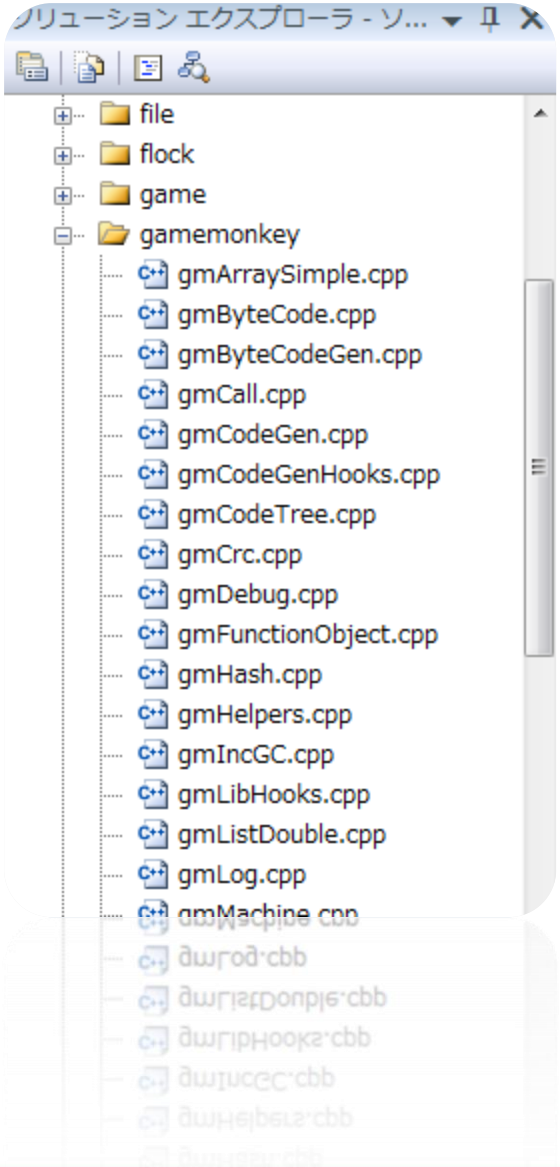
- インクリメンタルGCが常時動いても気にせず、フルGCを避けることにした
- 1フレームあたりのオブジェクトのトレース数、破棄数の上限を別途設定できる
  - インクリメンタルGCがオンになってからの負荷を微調整できる
- グローバル変数が解放されないため、ステージ後半にいくほど解放できないオブジェクトが溜まる

Game Monkey Script の活用

Game Monkey Script の特徴

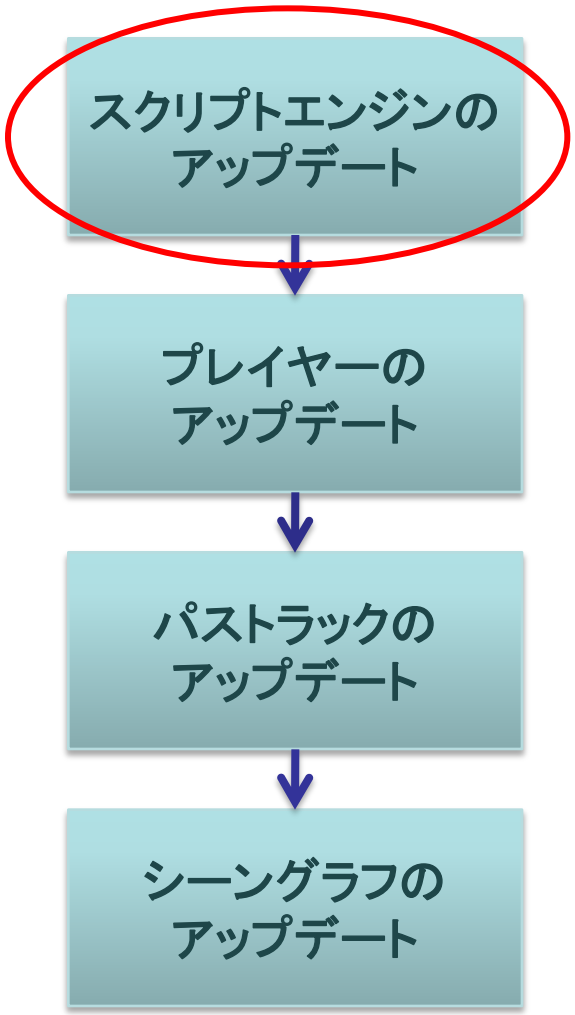
- 50kb 程度のコードベース
- **C ライクな構文**
- **（GMS自体が）オブジェクト指向**
  - C/C++側から何かと扱いやすい
- スタックマシン
- 関数はすべてラムダ
- プロトタイプベースのOOPもできる
- ユーザー型
  - こちらはクラスベースのOOPができる

ビルド方法



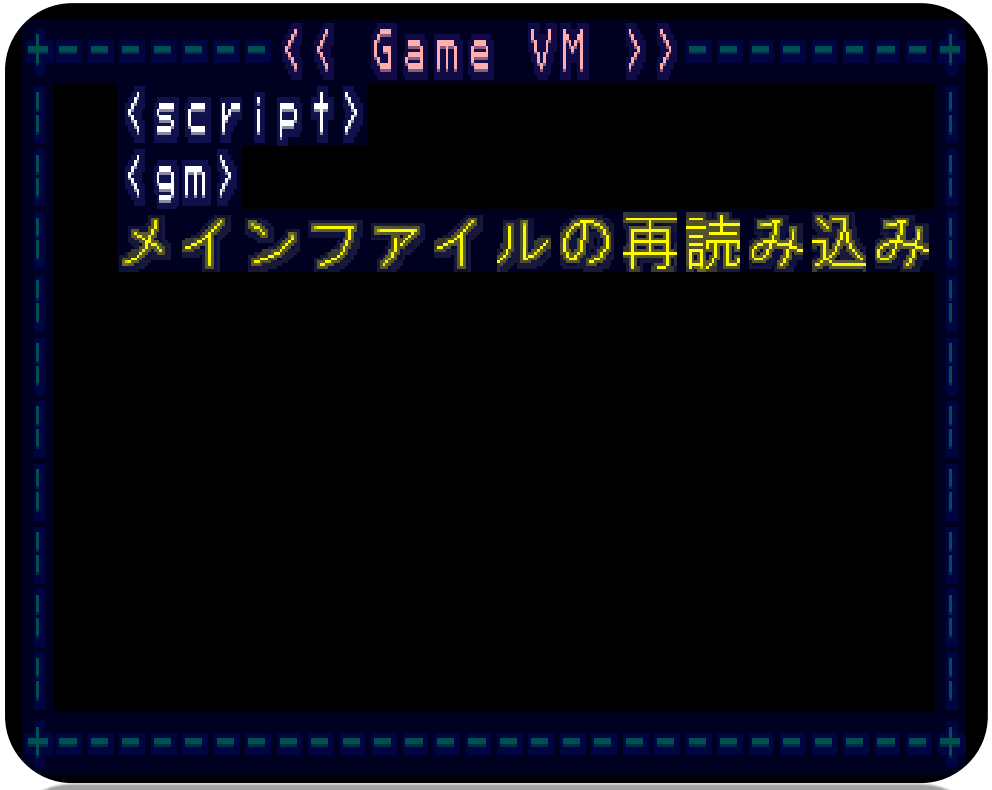
- プロジェクトにまぜるだけ
- PS3でも一発で通った
- UTF-8
- エンディアンは自動判別

実行上の特徴



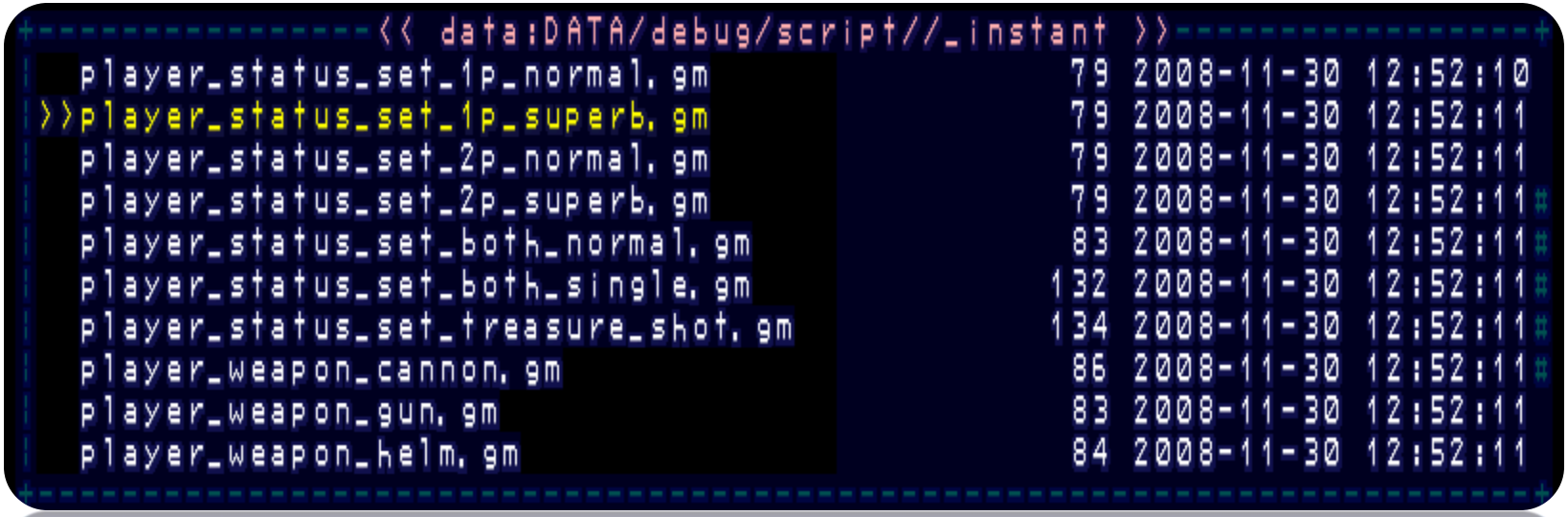
- VM実行時に実行中の全コルーチンをアップデート
- C/C++のコードを移すには向いてない
- 負荷は測りやすかった

リロードの実装



破棄関数と再初期化関数を呼び
スタートに指定されている関数から再スタート

即時実行スクリプト



ファイラーを用意して、
選択されたスクリプトを実行する

インゲームコンソール

１行コードとして処理するお手軽コンソール
業務用機だと、稀に役に立つ

ラムダの活用

```
event.frame = 10;
event.func = function(foo) {
        ……..
};
```

```
gobj_set_callback(skelton, function(obj)  { …. });
thread(function(obj)  { …. });
```

~~高次関数に大活躍~~

コールバック等をその場で書くのに大活躍

デバッガ

NO PRINTING

ターゲット側のソケット通信部を書くだけ
渋い活躍

拡張すると便利だったもの

- 無限ループの検出
  - 1万回くらいループしていたらエラーにする
  - XTALにはデフォルトでついてます
- 初期化していない変数の使用に対する警告
- メッセージボックスを使ったコルーチン間通信
  - ブロックに入る前にシグナルが発行されてもOKなので、ロード待ちの制御などで役に立つ

XTALの活用

XTALの活用

C/C++サイドでもリロードしたい……

XTALの活用

皆さん、
エディット＆コンティニューはうまく使えていますか？

GMSではだめなのか？

- プロトタイピングには使っていたが……
- GMSはクラスっぽいことはできるが、C++で書いたところを置き換えるにはつらい
- 中断したコルーチンがVMのアップデート時に再開してしまう

GMSではだめなのか？

- そこは使い分け！
- そこで**XTAL**！

XTALの特徴

一言で言うなら**モダン**です

XTALの特徴

- クラスベースオブジェクト指向
- C ライクな構文
- 動的な型
- **強力なバインダ**
- クロージャのサポート
- レジスタマシン
- **リロードのサポート**

- （社内に開発者） 

for に対する else とか

```
for ( i: 0 ; i < enemy_nums; i++ ) {
        …
}
else {
        …
}
```

XTALのユニークな構文や機能

if first とか

```
for ( i: 0 ; i < enemy_nums; i++ ) {
        if ( firststep ){
                ….
        }
}
```

XTALのユニークな構文や機能

no break節とか

```
for ( i: 0 ; i < enemy_nums; i++ ) {
        if ( … ){
                break;
        }
}
nobreak {
        // break でループを抜けなかった場合のみここを通る
        …
}
```

どうですか　このループに対する**愛！**

XTALのユニークな構文や機能

いま実行している関数を指す変数があるとか

```
// 再帰
callee(…);
```

XTALのユニークな構文や機能

- リロードのサポート
  - クラスが変更されたとき、**元のクラスのインスタンスを、変更後のクラスのインスタンスへ切り替える**ことができる
  - GMSではできない
- もしかして機能
  - TYPOかもしれない場合、類似したオブジェクトを検索して候補をリストしてくれる

パイレーツでの使用例

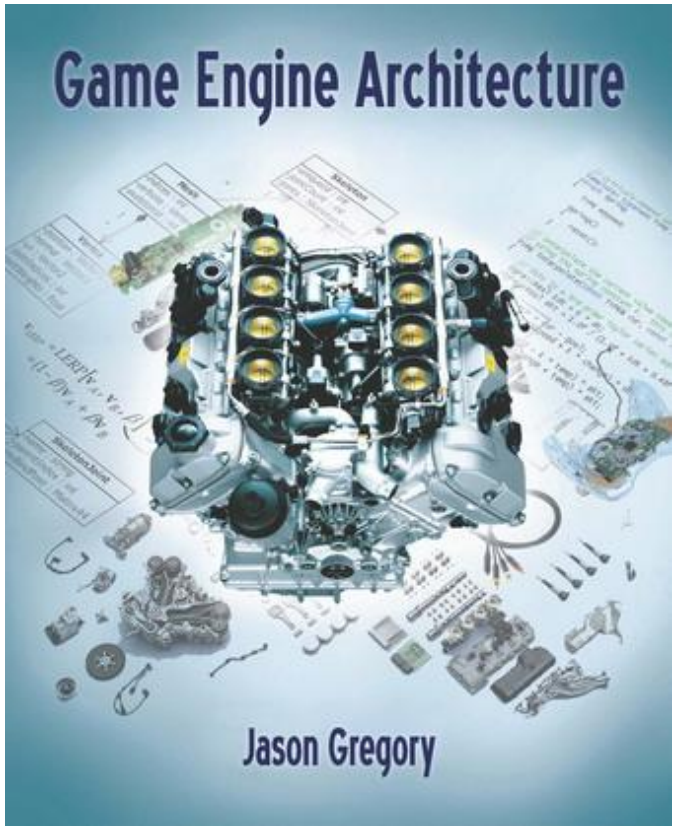
- 敵の投擲武器/飛翔体の実装
- C/C++で書かれたシーケンスの一部

総括

案ずるより産むが易し…!

- 何とかなるなった
- スクリプタがいないからやれないのではない、やらないからいないのではないか
- 技術的なこともオブラートに包まなかったが、企画は理解した
- スクリプトで重ければC/C++に持ってくればよいので何とかなる
  - ただ、GMSのブロック＆シグナルの仕組みは助かった

御清聴ありがとうございました



- スクリプトとゲームエンジンの絡みに興味がおありでしたら……
- Game Engine Architecture
- ソフトバンクさんのブースで一部を翻訳した抜粋版リーフレットを配布しています

Q & A