

## - P/ECE によるゲームプログラミング講座 -

はじめに、

最近のゲームというのは、本当に豪華になっているなと思います。3DCGのムービーなどは当たり前のように使われ、キャラクターはフルボイスでしゃべり、音楽も、昔のBEEP音源で無理矢理奏でていたようなレベルとは比べようはずありません。

そんな状況ですから、「これからゲームを作るぞ」と思っている、なかなか容易なことでは無いのではと思います。そんなわけで、もっと気楽にゲーム作りというものを楽しんでみよう、という趣旨のもと、今回「P/ECEによるゲームプログラミング講座」をお送りしたいと思います。

### P/ECE って何？

という方もいらっしゃるでしょうから、まずはP/ECEの簡単な紹介から始めたいと思います。

“P/ECE”とは昨年未からアクアプラスより発売されている携帯端末機です。特徴としては、CPUの回路図をはじめ、そのほとんどの内部情報が公開されており、ユーザーは自由にプログラムを組むことができます。感覚的には、自分でゲームが作れるゲーム機といった感じでしょうか。

P/ECEのスペックをざっと挙げてみますと、

- CPU 24 MHz (32bit RISC)
- FLASHメモリ 512KB、メインメモリ SRAM256KB、スタックメモリ 8KB
- 画面解像度 128 x 88 白黒4階調
- PWM出力、PCM再生、波形メモリを利用したMML演奏
- 赤外線による、P/ECE同士のデータ交換

といった具合です。

画面解像度が若干低いようにも感じられるかもしれませんが、逆にこれくらいの方がかえって手軽にプログラムを組めるので良いのではないかと思います。

P/ECEには開発環境がついており、基本的には付属のCコンパイラ(gccを独自拡張したもの)によってプログラミングを行っていきますが、一般的なC言語のスタイルとはやや異なるスタイルをとっていますので、次にP/ECEにおけるプログラミングスタイルについて簡単に説明していきたいと思います。

なお、この記事ではC言語自体の解説等は行っておりません。それに関してよく分からない事がありましたら、本屋さんで書籍を漁るなり、分かる人に聞くなりしてください。

## P/ECE におけるプログラミングスタイル

先ほどにも述べましたように、P/ECE では基本的には C 言語を用いてプログラミングを行います。といて難しく構える必要はなく、ゲームを作るために必要な機能は一通り、固有の API として用意されていますので、たいていの場合、必要な API の呼び出しだけで事足りるでしょう。

P/ECE で開発するに当たって普通の C 言語と大きく異なるのは、通常の main 関数ではなく、その代わりとなる次の 3 つの関数を用いるということです。

- void pceAppInit(void)  
プログラムの開始時に 1 度だけ呼び出される関数で、変数の初期化、仮想画面の設定などは基本的にこの関数内で行うと良いでしょう。
- void pceAppProc(int cnt)  
システムから指定間隔毎に呼び出される関数で、プログラムの本体を記述します。1 ループ分の処理が終わった後は、必ずシステムにリターンされなければなりません。(while で無限ループなどは許されません)  
引数 cnt には、この関数が呼び出された回数が渡されます。
- void pceAppExit(void)  
プログラムの終了時に呼び出される関数で、メモリの解放等を行うと良いでしょう。

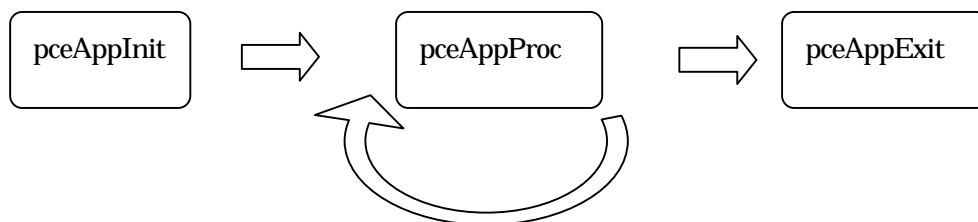
これら 3 つの関数は省略することはできず、必ずプログラム中に記述していなければなりません。

また、プログラムにシステムからの各種通知が発生したときに次の関数が呼ばれます

- void pceAppNotify(int type ,int param)  
引数 type ,param にはそれぞれ、通知タイプ、パラメーターが渡されます。

こちらは、省略することは可能です。

従って、大まかな流れとしては次の通りとなります。



## ゲームプログラミング実践編

それでは、実際にゲームを作っていくことにしましょう。ここでは、ゲームの基本（と筆者は信じている）インベーダーゲームの簡易版を作ることになります。

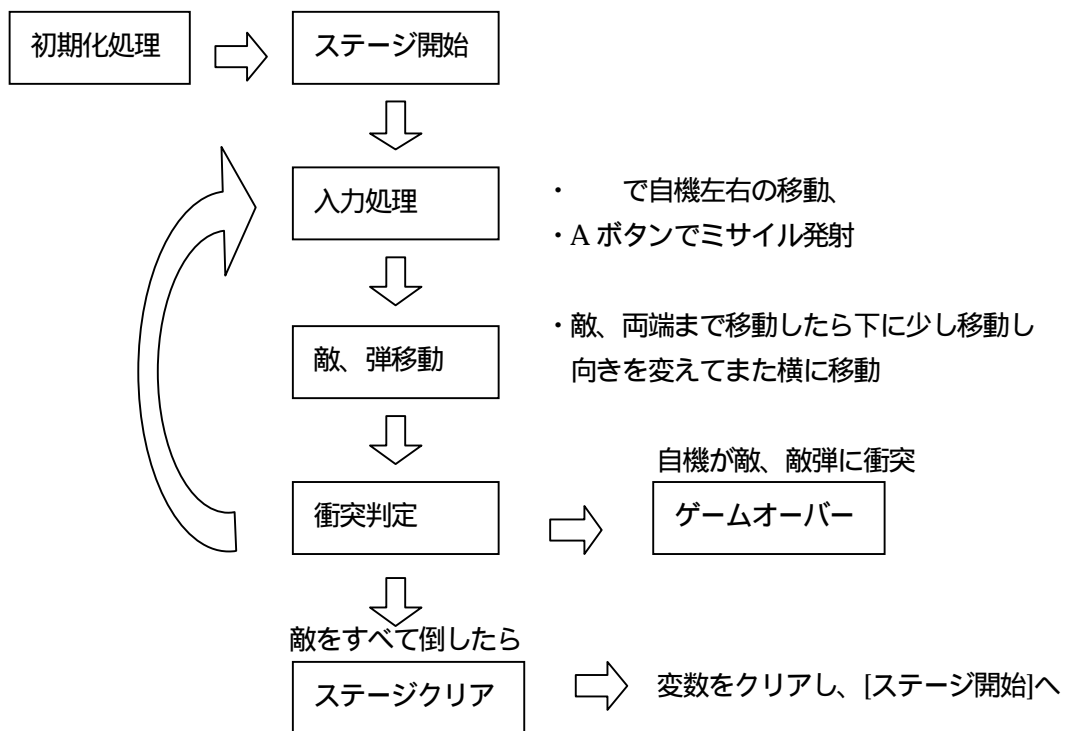
### 設計

まずゲームとしての最低限の要素、ルールを決める必要があります。インベーダーゲームのルールを単純に考えてみると、大体次のようになると思います。

1. 自機を左右に操作する事が出来る
2. 敵（インベーダー）は編隊を組み、左右に移動しつつ少しずつ降りてくる
3. 自弾、敵弾は真っ直ぐに進む
4. 自弾が敵に当たれば敵は消滅し、画面内の敵をすべて倒せばステージクリア
5. 自機に敵および敵弾が当たる、或いは敵が画面下へと到達したらゲームオーバー

とりあえず簡易版ということなので、これぐらいで良いでしょう。

では、以上のルールをもとに具体的なゲームの大まかな流れを考えてみましょう。



大筋ではこのような具合でしょう。

## 実装

プログラムの大まかな設計が出来ましたので、次は実際の処理を考えていきます。

まず、必要となる主要変数挙げてみると、次のようになります

自機用	tObject player
敵用	tObject enemy[横最大敵数][縦最大敵数]
自、敵弾用	tObject shot[2][最大弾数]
壁（衝突判定用）	tObject wall[2]
敵の向き	int dir
ゲームスピード	int speed
残敵数	int RestEnemy
スコア	int score
キャラクタ画像データ	PIECE_BMP pic_data
背景画像データ	PIECE_BMP bg_data
タイトル画像データ	PIECE_BMP title_data
画像情報テーブル	tObjType TypeTbl[]

### 構造体 tObject

int x	オブジェクトのx座標
int y	オブジェクトのy座標
int type	オブジェクトのタイプ番号
bool visible	存在フラグ

### 構造体 tObjType

int src_x	データ上のオブジェクトの画像開始x座標
int src_y	”y座標
int width	オブジェクトの幅
int height	”高さ
PIECE_BMP *data	使用画像データ

少し解説しますと、プログラム中から画像情報を意識させないために、画像情報を表す構造体 tObjType でキャラクタの画像情報を保持する配列テーブルを用意し、各種キャラクタを表す構造体 tObject 内に上記テーブル上の、そのキャラクタの現在画像を表すインデックス値”type”を保持します。

続いて、各処理内容について考えます。

- 初期化処理
  - ◇ 各種ゲーム変数の初期化、画像データ（キャラクタ、背景）を読み込み、仮想画面の初期化、pceAppProc の呼び出し間隔設定、各キャラクタの作成を行います。
- ステージ開始
  - ◇ 各キャラクタ用変数の初期化を行います。
- 入力処理
  - ◇ 左右キーが押されたとき、自機がゲームフィールド内にあれば、自機の位置情報に左右への増分値を加えます。
  - ◇ ボタンが押されたとき、自弾用変数に空きがあれば自弾を一つ発射（存在フラグを真、位置情報は自機と同じ位置）します。
- 敵、弾移動
  - ◇ 敵、弾のうちで存在フラグが真であるものに対して、位置情報に、その進行方向（敵：現在向いている方向、自弾：上方向、敵弾：下方向）へ増分値を加えます。
  - ◇ 敵編隊のうちで両端に位置する敵がゲームフィールド内のそれぞれの端に到達したとき、敵の位置情報に下方向への増分値を加え、進行方向を反転させます。
  - ◇ 敵移動時にランダムに敵弾を発射させます。
- 衝突判定
  - ◇ 自機・敵、自機・敵弾、敵・自弾の衝突判定を行い、前二者が真の場合、ゲームオーバー処理へ、後者が真の場合、当たった敵、自弾双方の存在フラグを偽にし、得点を加算、敵残数を一減らす。敵残数が0になったらステージクリア処理へ。
  - ◇ 処理の簡略化のため、ゲームフィールド上下部に不可視の壁キャラクタを作成し、壁・自弾、壁・敵弾、壁・敵の衝突判定を行い、前二者が真の場合、弾の存在フラグを偽にし、後者が真の場合、敵の侵入を許したとしてゲームオーバー処理へ。
- ステージクリア
  - ◇ クリアのメッセージを表示後、ゲームスピードを一段階上げ、ステージ開始処理へ。
- ゲームオーバー
  - ◇ ゲームオーバーのメッセージを表示後、変数を全て初期化し、初期画面へ。

大体このあたりで、ほぼプログラムの仕様は固まりました。

それでは、あとはコーディングしてゆくだけなのですが、紙面の都合上、ソースコードをここに載せるには少々無理がありますので、今回の記事ではひとまずここで区切りとさせていただきます。

なお、今回作成した簡易版インベーダーゲームのプログラム一式は <http://www.oucc.org/~mace/> 以下にしておりますので、拙いソースでもいいから見たいという奇特な方はそちらからダウンロードして下さい。

代わりといっちは何ですが、出来たゲームのキャプチャ画像を。



### 終わりに

プログラム設計というのは、「こうすればいい」と決まっているようなものではありません。今回サンプルとしてインベーダーゲームの作成手順を解説してみました。この方法が正しいとかそういう事ではなく、あくまでゲーム作成における一つのやり方を示したにすぎないという事です。

また、ゲームの作成において基本的な考え方というのは、どのような環境においてもそれほど違いはありません。(というか、根幹となるシステムが環境依存しているような設計などは、明らかに、設計段階で誤りがあるとしか思えません) 今回はP/ECEという環境を題材に解説をしてみました。ゲーム作成の(また、それに限らず多くのアプリケーションの開発において)大筋の考え方というのは他の環境にも運用する事が出来ると思います。

そんなわけで、この記事が、環境に関わらずこれからゲームを作ろうとする方にとっても、少しでも参考となる事が出来たら幸いです。

最後の最後に、記事に関して何か分からない点、および文句のある点がありましたら、私を捕まえて、小一時間問いつめて下さい。

文責： 似非の人こと、清水真之

[mace@oucc.org](mailto:mace@oucc.org)