

がんばれ！菜月さん！モジュール仕様書

## ソースファイル

がんばれ！菜月さん！ Ver1.11 (コメント付きソース)

<http://dij7.fc2web.com/alpha/data/natuki.zip>

ソースファイルはD.Kが作成したのですが、自由に使用してかまいません。

## 概要

過去に作成したソフトを含めたプログラムの構成をまとめる。  
構成は大きくは同じため、ここでは「がんばれ！菜月さん！」についての構成を記述する。  
ソフト作成に使用しているツールは「YGS2000」使用している。以下「YGS」と記述。  
製作者のホームページ：<http://yaneurao.hp.infoseek.co.jp/ygs/>

- 使用ツールについて

YGSは独自のスクリプト言語となっている。  
しかし書式はCに近く自由度も高い。Cプログラムが可能ならゲームソフト以外のWindowsのソフトも作成可能となっている。  
DirectXを利用するためのクラスが全てラッピングされ、ユーザはDirectXを意識することなくC言語の感覚でDirectX利用したコードを記述できるため使用している。

## 設計概要

YGSでは1スクリプトファイル内に1つのmain関数をもたせることができる。  
スクリプトジャンプ機能が用意されており、スクリプトから別スクリプトへ移行することが可能となっている。  
別スクリプトへジャンプすると、現在の動作しているコードは開放される。  
ジャンプ先のスクリプトファイルがコンパイルされオブジェクトがスタック領域へ作成されmain関数が実行される。  
ジャンプ前の変数、関数は開放されるため参照は出来ない。  
スクリプトファイルを画面単位で作成し、この機能を使用することで各画面が独立したクラスに近いイメージで作成することが可能となる。  
これにより画面の追加、削除時に発生する作業量は少量となり、別ソフトで作成した資産をそのままスクリプトファイルの入れ替えで組み込むことが可能となるため、コードの再利用性も高くなる。

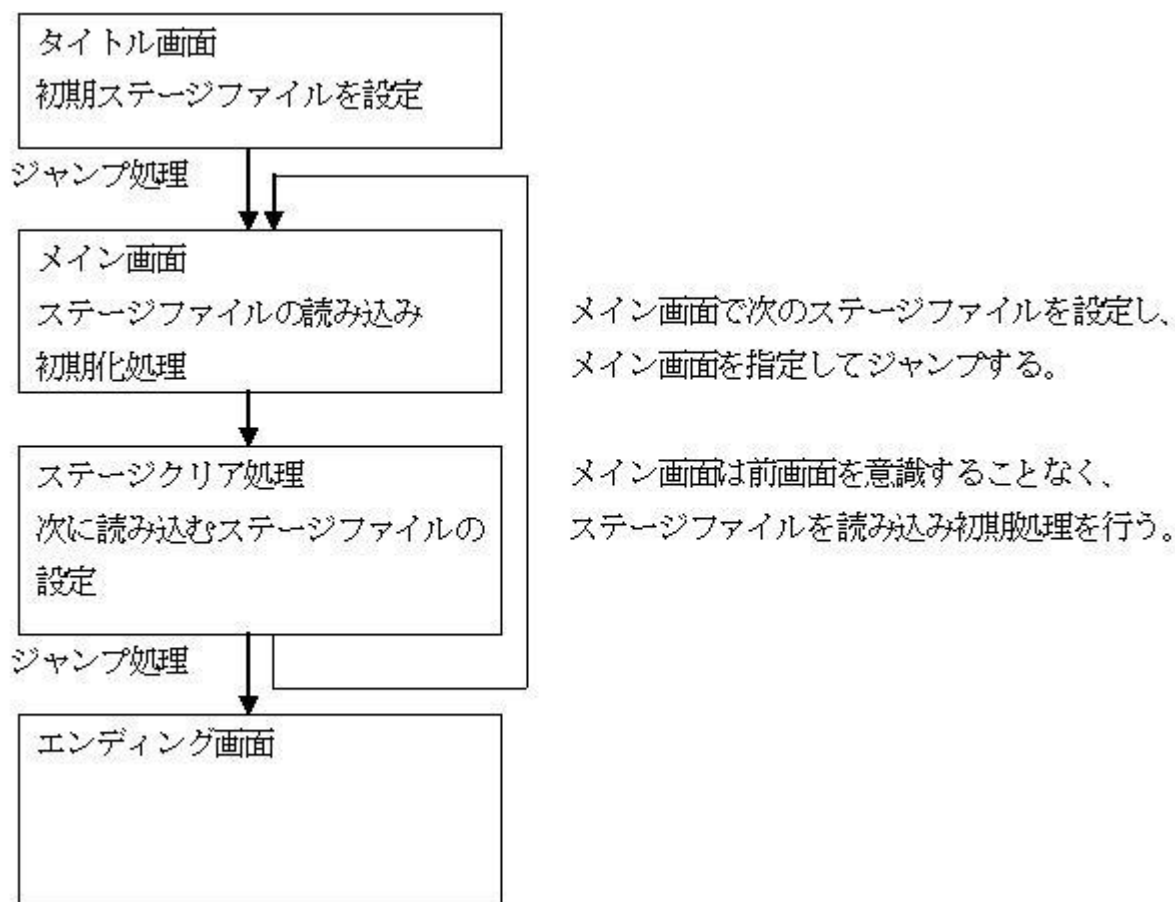
スクリプト間で参照できる共通のグローバル変数はgameflag[512]だけとなっている。  
各画面スクリプトファイルはここに次の画面遷移状態を保存し、画面管理スクリプトへジャンプを行う。  
各画面スクリプトファイルはgameflagの情報のみを参照して現在の状態へ初期化を行えばよい。  
そのためgameflagをインターフェースとして前に表示していた画面がどの画面であるかを意識したコードを書く必要がなくなり、完全に独立した形で各画面の作成作業を行うことができる。  
画面遷移の変更が容易となる。

またgameflagを保存する事で、そのままプレイヤーのセーブデータとすることが出来る。  
gameflagバッファにファイルを読み込むことで、各画面は必要な状態に初期化を行い表示されるため、セーブ機能の追加が容易となる。

前画面を意識させない設計によってステージデータの書き換えを実現している。

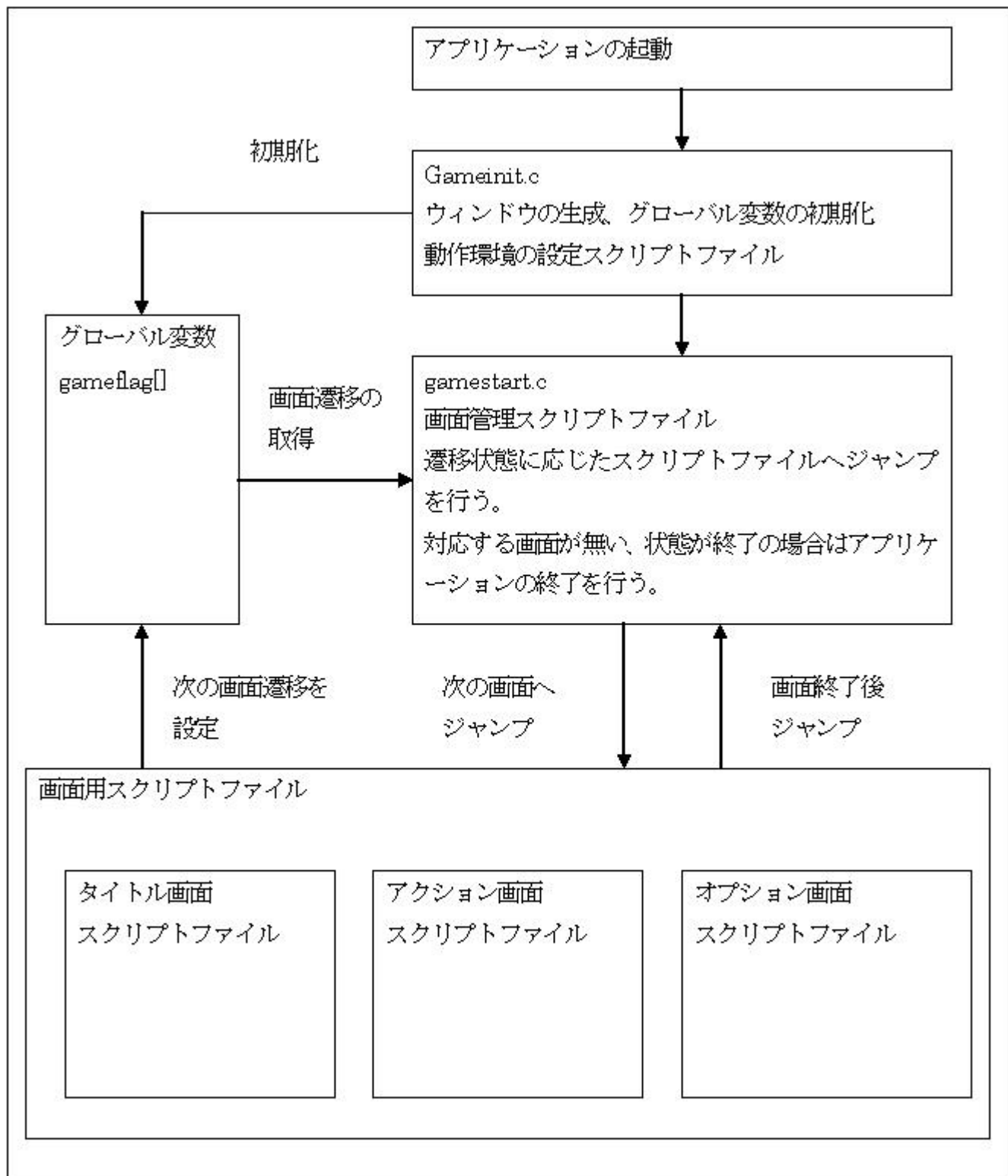
初期化処理を統一することで、初期化漏れなどの潜在的不具合の発生を防止する。  
以下に概要を示す。

- ステージファイル読み込み処理フロー



以下にスクリプトファイルの関連を示す。

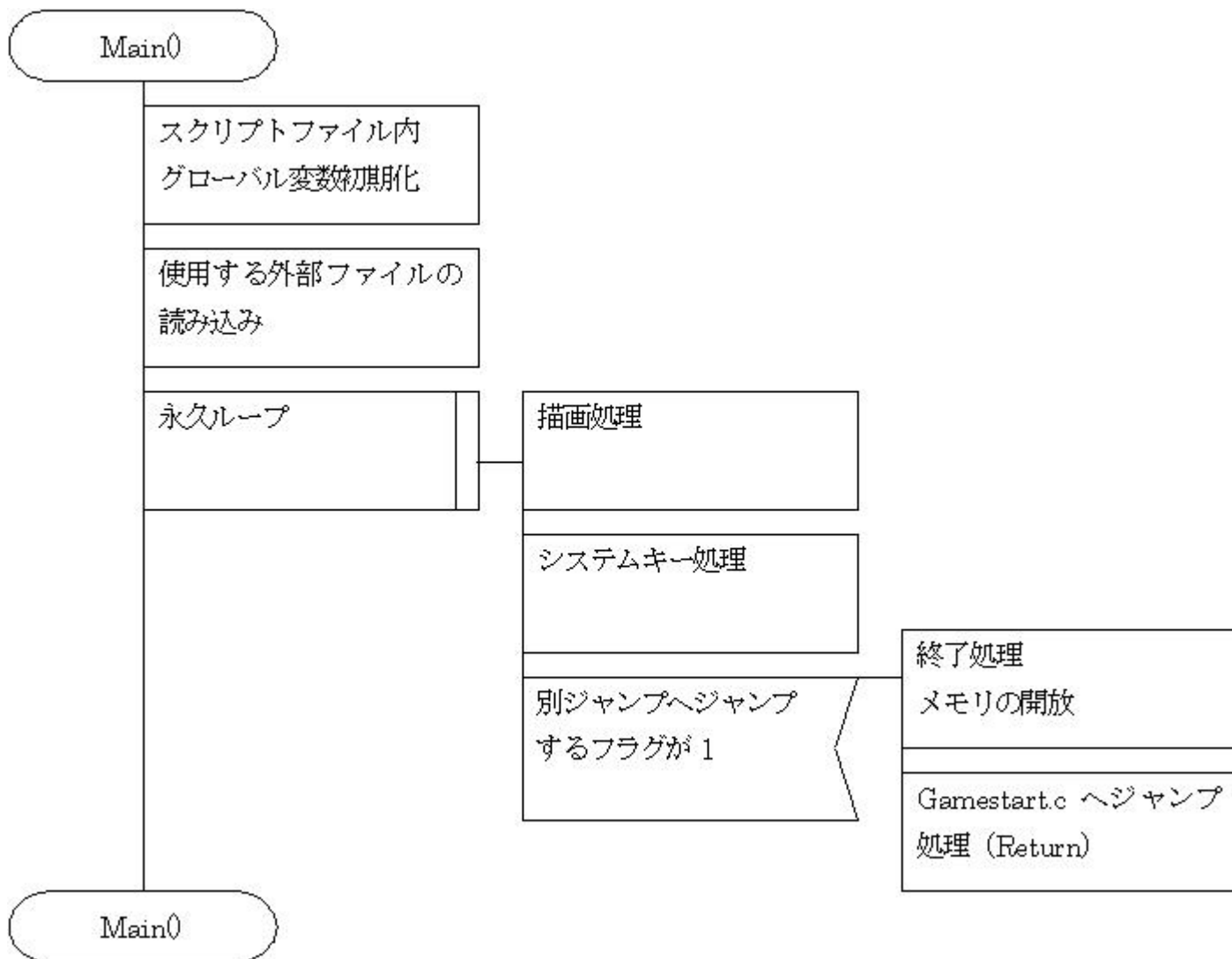
- スクリプトファイル関連図



## 画面スクリプトファイル

画面スクリプトファイルで行う処理は、全ての画面で同じとなる。  
以下に画面スクリプトファイルの処理フローを記述する。

- 画面スクリプトファイルの処理フロー

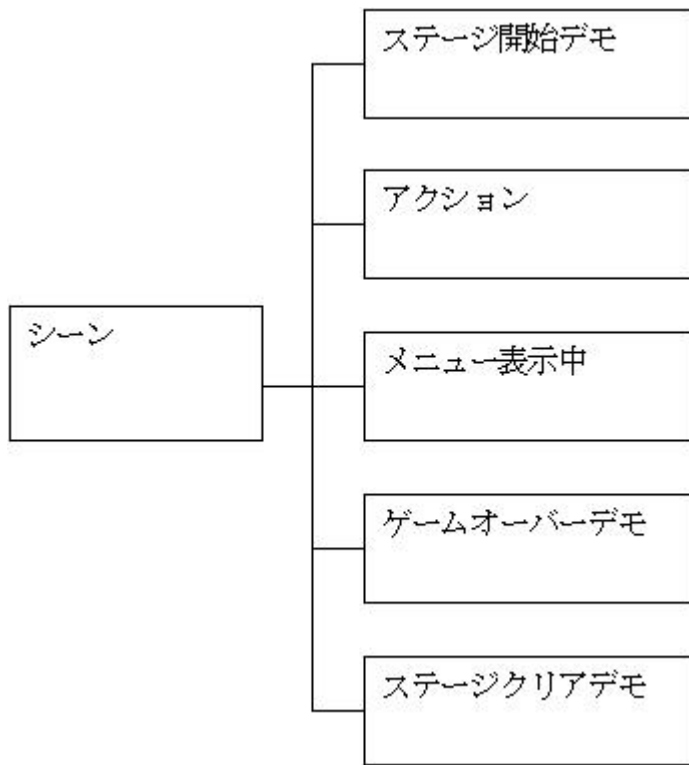


- 画面スクリプトファイル内でのシーン管理

gamemain.cでは内部でさらにシーン管理を行っている。  
スクリプトファイルをジャンプしてしまうと、以前のスクリプトファイルの情報を参照することは出来ないため、スクリプトファイル内の情報が必要な画面遷移は内部にシーン管理が可能な設計とする。

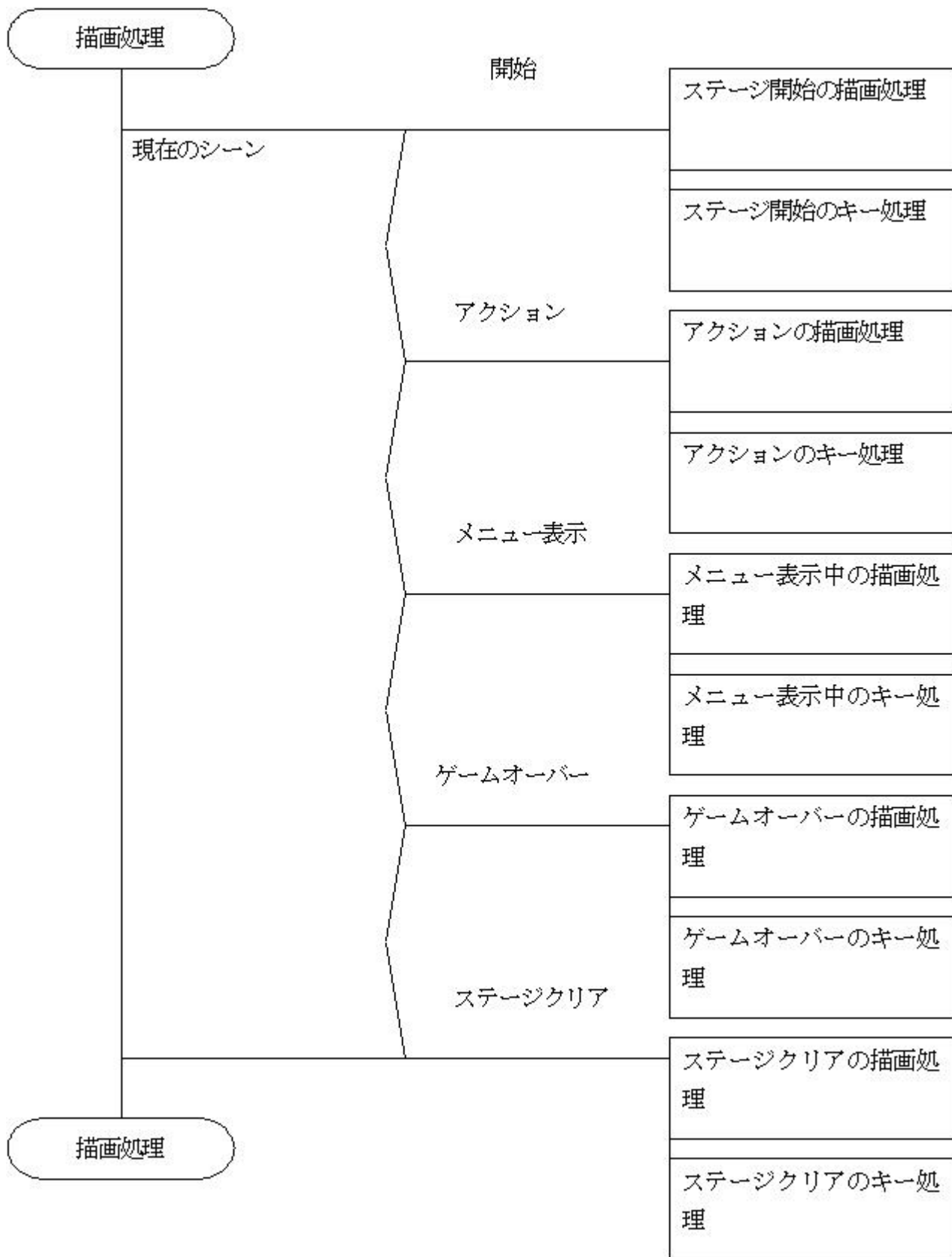
以下にシーンを示す。

- シーン関連図



シーン管理は描画処理とキー入力処理を、それぞれのシーン用に作成することで実現する。以下にgamemain.cのシーン管理部フローを示す。

- シーン管理部フロー



シーンに合わせた描画処理とキー入力処理を作成することで、シーンの追加ができ描画処理、キー処理をやめることでシーンの削除を行う事が出来る。シーン追加時に作成するべき処理を具体的にすることができる。

## 表示オブジェクト

プレイヤー、プレイヤー、アイテム、敵、破片など画面に表示され、独立して動作を行う。

- 各オブジェクトで使用するメモリ領域

メモリ領域はグローバル変数として全て静的に確保する。  
使用するメモリ空間を明確にし、メモリリークやメモリ不足、想定外のキャラクター数による不具合などの潜在的な不具合を防ぐ事を目的とする。  
YGSでは構造体が使用できないため、全て1次元の配列として取得している。  
1次元の配列を構造体に見立てて参照する。オブジェクト変数へのインターフェース関数を用意しオブジェクトポインタ (= オブジェクトインデックス) を引数として参照を行う事とする。

- オブジェクトの生成

各オブジェクト用の変数にそれぞれにオブジェクトを作成するインターフェース関数を用意する。  
引数は、種類、初期位置などを渡す。  
空きバッファを検索し、見つかった場合にはオブジェクトはバッファ内に生成され、以降消滅までオブジェクトが自動的に動作を行う。  
空きバッファが存在しない場合、生成は行わない。

- オブジェクトの表示、動作

各オブジェクトの表示関数を用意する。表示関数は描画処理関数で呼び出される。  
オブジェクト変数を検索し、生成されているオブジェクトが発見された場合、パラメータから表示と、動作を行う。  
当たり判定を行う場合は存在数の少ないオブジェクトが、存在数の多いオブジェクトを検索し当たり判定を行う。