具体的な企画
環境
フローチャート
簡易設計
素材
操作系

撮影した能力を使う
・メラは撮影銃的な

世界観
・スチームパンク風

スキルは定数（enum）

敵のバリエーション
・移動系
ハイジャンプ
ダッシュ
/
・攻撃
パンチ
切る
/

ストックは一回分

コーディングルール

・

| 命名規則名 | 日本語名 | 内容 |
| --- | --- | --- |
| **PascalCase** | **キャメルケース** | **先頭大文字，それ以降の単語区切りも大文字** |
| **camelCase** | **パスカルケース** | **先頭小文字，それ以降の単語区切りは大文字** |
| snake_case | スネークケース | すべて小文字，単語区切りをアンダースコア _ でつなぐ |
| SNAKE_CASE | 知らない | すべて大文字，単語区切りをアンダースコア _ でつなぐ |

※クラス、関数はキャメルケース、変数はパスカルケースで行う

※定数は、大文字のスネークケース

- 動作やクラス，格納する情報にふさわしい名前をつける
  - これが難しいので，リーダブルコードのような本が出る
- ハンガリアン禁止：メリットよりデメリットが多いことが知られている [C++ Code]
- 略称禁止：略称を使うと分からなくなる場合がある

## bool 変数

基本的に `is~`, `has~`, `can~`, `enable~` など `true` のときどうなっているかが明確に分かる名前にする. 付けがちな `flag~` は, true のときどちらかよく分からないときがある. `disable~` は禁止する.

長い条件文は関数化するか、bool でとる。

if 分のフォーマット

- 条件文が簡潔
- return が簡潔
- else がない
- 意図が明確(書式エラー, NULL など)
- 一行 if 文禁止{}を必ずつける。

3 項演算子は原則使わない

Nullptr は明記する

## コメントのつけ方

・関数

引数名や関数名は多少冗長になってもコメントを読まなくても意味が伝わるような名前にする

コメントは

//関数の動作

//第〇引数

//戻り値

の形式で書く。

値を変更しない関数や引数は const をつける

クラスの説明文は必ずヘッダの宣言部分の直前に書く

例

//クラスのコメント

Class hoge

{}

パスやファイル名は全部半角のみにする

フォルダのファイルを多少手間でもしっかり分ける（エディターとエクスプローラー両方）