電算研アイデアソンハッカソン用
アクションゲーム仕様書

最終更新日:6/30

コーディング規約

変数

my_name のように小文字を使い、_を使って単語をつなげる

ゲッター

name という変数のゲッターは　name()

bool 型の場合は　isEmpty()　のように is 変数名とする。

ただし、直接自分のフィールド変数を返すものでない場合は普通の関数の名前をつける。

セッター

name という変数のセッターは set_name(String name)

関数

頭文字を大文字にする。　Draw()

.h の関数記述の順番

public→protected→private

役割分担

MapFactory,Map:Tenma

GameMain,Menu:三井

Field:森

Object,Character,Character以下のクラス,CharacterController,CharacterController以下のクラス:柳町

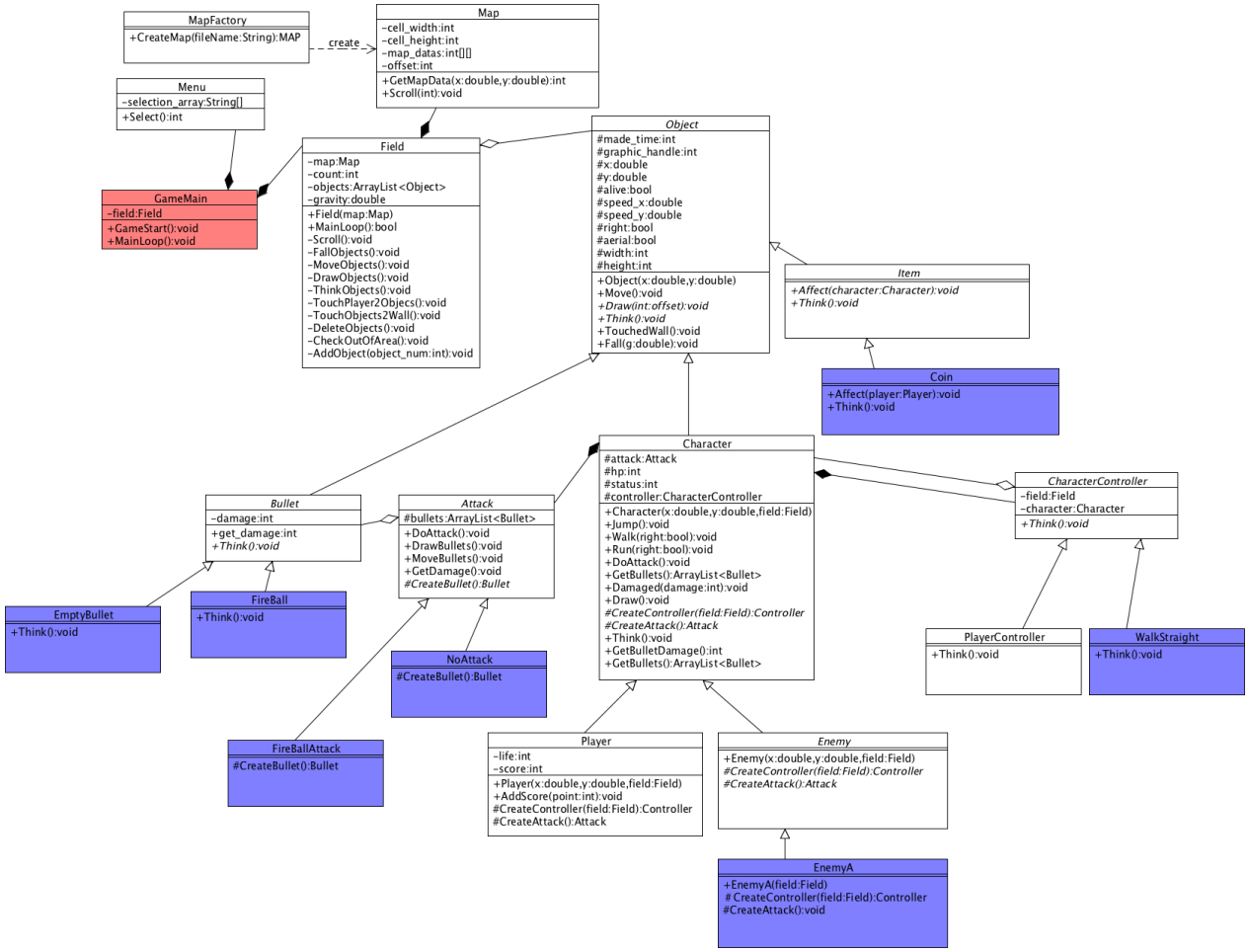
Item,Item以下のクラス,Bullet,Bullet以下のクラス,Attack,Attack以下のクラス:中井

クラス設計

ゲッターセッターについてはだいたい省略してあるので、必要なものには追加してください。

クラス図

（赤がメイン関数から呼ばれるところ。青が具象クラス）



1. GameMain クラス
   ゲームのメインループのクラス
   - -field:Field
   - +GameStart():void
     - ゲームを始める。Menu クラスでプレイヤーにステージ選択を行ってもらい、MapFactory で選択したステージのマップを生成する。その後 Field クラスのインスタンスを生成。
   - +MainLoop():void
     - ループの処理はここでする。FPS は 60 になるように調整。
     - ループ内で Field の MainLoop()を呼び出して、ゲームのメインループにする。

2. Menu
   メニュー画面のクラス
   - # define MAX_SELECTION 1
     - 選択肢の個数。とりあえず 1 で
   - -selection_array:String
     - 選択肢の文字列の配列
   - +Select():int

        - 選択肢の文字列を画面に表示して、ユーザーの選択待ち(ここでループまわす)。選択した場合はその選択肢の番号を返す。
        - どんなメニュー画面にするかは任せる。
3. MapFactory
   テキストファイルからマップを生成するクラス
    - +CreateMap:(fileName:String):Map
        - 引数のファイル名で Map インスタンスを生成して返す
4. Field
   マップやキャラクターなどのオブジェクトのすべてを管理するクラス
    - -map:Map
    - -count:int
        - 始まってから何フレーム(ループ)経過したか
        - 時間制限に利用してもいい
    - -objects:Vector<Object>
        - Object の配列
        - 0 番目には Player
    - -gravity:double
        - 重力
    - +Field(map:Map)
        - コンストラクタ
        - count 初期化
    - +MainLoop():bool
        - ゲームのメインループで呼ばれるプロセスを並べる
        - FallObjects,MoveObjects,ThinkObjects,DrawObjects,CheckOutOfArea,TouchPlayer2Objects,TouchObject2Wall,DeleteObjects の順
        - ここでの処理が 1 ループに相当
        - ゲームが終了した場合、false を返す
    - -DrawObjects():void
        - 各 object に対して Draw 呼び出し
    - -ThinkObjects():void
        - 各 object に対して Think 呼び出し
    - -MoveObjects():void
        - 各 object に対して Move 呼び出し
    - -FallObjects():void
        - 各 object に対して、空中にいる場合に Fall 呼び出し
    - -Scroll():void
        - スクロール処理を行う
        - map に対してもスクロール処理するようメッセージを送る(Scroll 関数呼び出し)
        - 新しく読み込まれた場所がオブジェクトを持っていた場合、AddObject で生成と追加を行う。
    - -TouchPlayer2Objects():void
        - プレイヤーと敵及びアイテムの接触判定。
        - プレイヤー弾と敵、敵の弾とプレイヤーの判定も行う。
        - 接触している場合生存フラグを false にするなどの処理を行う。(アイテムならその効果が発動したり)

- -DeleteObjects():void
  - 生存フラグが false の Object を消去する。
  - NSArray の Objects から削除　かつ delete でメモリから削除
- -AddObject(object_num:int):void
  - 引数のオブジェクトを生成(new で生成)して、自身の配列に追加
  - 初期の描画域生成時と、Scroll()内で呼ばれる。
- -CheckOutOfArea():void
  - 描画画面外にオブジェクトが存在する場合、削除の処理を行う

5. Map
   マップデータを管理しているクラス
   - cell_width:int
     - マップの１セルの幅
   - cell_height:int
     - マップの１セルの高さ
   - enum
     - 列挙型。マップデータの数字が何を表すか
     - EMPTY:0 BLOCK:1 Enem1:2 ...
   - -map_datas:int[][]
     - マップチップデータ
   - -offset:int
     - 表示画面領域が全体マップからみてどこかを表すためのもの
     - セル単位ではなくピクセル単位（右に 30 スクロールするなら offset=30 にして、描画する時　(x-offset,y)が描画場所になる）
   - +get_map_datas():int[][]
     - ポインタを返すことになるはず
     - GetMapData だけでいーかも
   - +Scroll(point:int):void
     - offset を動かす offset+=point;
   - +GetMapData(x:double,y:double):int
     - 描画領域の x,y 座標を引数にとって、マップデータの配列の該当する場所に何があるかを返す
   - +Draw():void
     - map の描画を行う。
     - 背景を表示して、 map_data の値が 1 ならブロック。それ以外なら何も表示しない
6. AObject
   位置や画像を持った「物体」の抽象クラス
   - #graphic_handle:int
     - グラフィックハンドル（メモリに保存した画像の識別番号。描画する関数を使うときに必要になる）
     - カウンタか何かで同じ画像は１回だけ読み込んでメモリに置くようにする
   - #made_time:int
     - オブジェクトが生成された時間（フレーム）
   - #x:double
     - x 座標。(描画領域の、ではなくマップ領域全体からみた)

- #y:double
  - y 座標
- #alive:bool
  - 生存しているかどうか。生存しているなら true。死亡しているなら false
- #speed_x:double
  - x 軸のスピード (どれだけ移動するか。x =x+speed で移動)
- #speed_y:double
  - y 軸のスピード
- # right:bool
  - オブジェクトが右向きなら true
- #aerial:bool
  - オブジェクトが空中に存在しているなら true
- #width:int
  - オブジェクトの画像の幅
- #height:int
  - オブジェクトの画像の高さ
- +Move():void
  - x と y に対して、スピード分移動する。
- +Draw(offset:int):void
  - x,y から描画の位置を offset を用いて算出し、グラフィックハンドラの画像で描画する。
- +Think():void
  - 抽象関数
  - どのように動くかを決定する部分。基本的に自身のスピードをいじる
- +Fall(g:double):void
  - オブジェクトのスピードに対して重力加算
- +TouchedWall():void
  - オブジェクトが壁と当たった時に呼ばれる。速度を 0 に。

7. Item
   アイテムの抽象クラス
   - +Affect(character:Character):void
     - 抽象関数
     - Character のフィールド変数をいじってアイテムの効果を与える。
   - +Think():void
     - オーバーライドした抽象関数。下のクラスで実装
8. Character
   - #attack:Attack
   - #hp:int
     - 体力
   - #controller:CharacterController
   - attack:Attack
   - +Character(x:double,y:double,field:Field)
     - コンストラクタ　引数から初期位置を決定
     - CreateController を呼び出して controller に参照を持たせる
     - CreateAttack を呼び出して attack に参照を持たせる。
   - +Jump():void

        - 飛ぶ。　例 speed_y=-10;
        - 引数指定してどれくらいの高さで飛ぶかやってもいいかも
    - +Walk(right:bool):void
        - 歩く。引数は右向きかどうか。右向きなら speed_x を変化させる。左なら逆。
    - +Run(right:bool):void
        - 走る。上と同じ
    - +DoAttack():void
        - 攻撃する
        - 具体的にどうするかは Attack に委譲
    - +Damaged(damage:int):void
        - 引数の整数の分だけダメージを受ける（hp を減らす）
        - 0 になれば alive を false に
    - +CreateController(field:Field):Controller
        - 抽象関数
        - コントローラーを生成して、参照する
    - +CreateAttack():void
        - 抽象関数
        - Attack を生成して参照する
    - +Think():void
        - 具体的にどうするかは controller に委譲
9. Player
   プレイヤークラス
    - -life:int
        - 残機
    - -score:int
        - スコア
    - +Player(x:double,y:double)
        - コンストラクタ
    - +AddScore(point:int):void
    - #CreateController():void
    - #CreateAttack():void
10. Enemy
    エネミーの抽象クラス
    - +Enemy(x:double,y:double,field:Field)
        - コンストラクタ
    - +CreateController():void
    - +CreateAttack():void
11. Attack
    攻撃をする抽象クラス(どんな Bullet を飛ばすかは下位クラスに任せる)
    - -bullets:Vector<Bullet>
    - +DoAttack():void
        - CreateBullet()で bullet を生成し、bullets に追加する
    - +DrawBullets():void
        - bullets に対して描画
    - +MoveBullets():void

        - bullets を動かす
    - -CreateBullet():Bullet
        - 抽象関数
        - Bullet の下位クラスを作成する

12. Bullet
    - -damage:int
        - 敵にぶつかったときどれだけダメージを与えるか
    - +Think():void

    弾の抽象クラス。Attack クラスで操作する
13. CharacterController

    キャラクターを動かすクラス
    - #character:Character
    - #field:Field
    - *+Think():void*
14. PlayerController

    キーボード入力からプレイヤーを動かすクラス
    - +Think():