|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 仕様書 | | 戦闘の仕組み | | |
| 種類 | 説明 | | 作成者 | たかなし |
| 日付 | 2014/7/21 | |

**1　内容**

　戦闘がどのように実装されているかの概要を説明する。

**2　戦闘を構成するクラス**

2. 1　人形と敵を表すもの

　実際に戦闘を行うのは、Game\_BattleDollクラスとGame\_BattleEnemyクラスのオブジェクト群である。この二つはいずれもGame\_BattleUnitクラスから派生しており、攻撃の宣言、ダメージの判定、などの味方にも敵にも共通の演算はこのクラスで定義されている。

　画面に見えているのは、Sprite\_BattleDollクラス（画面下、人形の画像）とSprite\_BattleEnemyクラス（敵グラフィック）である。この二つのクラスはそれぞれGame\_BattleDollクラスのオブジェクト、Game\_BattleEnemyクラスを参照するポインタを持ち、Game\_BattleXXXの状態を反映して自動的に描画される。

2. 2　その他戦闘画面を構成するもの

　Sprite\_BattleDoll、Sprite\_BattleEnemyの他に、画面には各種ウィンドウが表示される。主なウィンドウは以下の通り。

Window\_BattleMessage：戦闘中のメッセージを表示する。

BWindow\_AliceCommand：アリスのコマンドを表示する。

BWindow\_DollCommand：人形のコマンドを表示する。

BWindow\_FocusedEnemy：ウィンドウと言いつつ、実はウィンドウではない。敵を選択した時に出るカーソル。便宜上ウィンドウクラスから派生させ、描画は独自関数で行っている。

**3　戦闘フェイズ**

戦闘が今どんな状態かを制御するのが、Scene\_Battleクラスのphaze値である。

このphaze値は列挙体（enum）BATTLEPHAZEの変数であり、以下の値を取る。

BEFORE\_BATTLE, // 戦闘前（PRE\_BATTLEの前に入るダミー）

PRE\_BATTLE, // 戦闘前

*ALICE\_COMMAND, // アリスのコマンドを選択*

*ALICE\_COMMAND\_DO, // アリスのコマンドを実行*

*DOLLS\_COMMAND, // 人形のコマンドを選択*

*ENEMIES\_COMMAND, // 敵のコマンドを選択（即座に終了する）*

*BEFORE\_SORT, // 素早さソートの前に各コマンドの判定を行う*

*AFTER\_SORT, // ターンの最初（ソートの後）*

*BATTLE\_DO, // 戦闘*

*AFTER\_TURN, // ターン終了時*

POST\_BATTLE, // 戦闘終了

END\_BATTLE, // シーン終了(即座にフェードアウトしてシーン移動)

これらのフェイズのうち、斜体になっているものがいわゆる「1ターン」の流れである。味方か敵のどちらかが全滅するまで斜体のフェイズを繰り返し、終わったらPOST\_BATTLE（勝利宣言など）を経てシーンを終了させる。

各フェイズについての詳細は次項と合わせて述べる。

**4　「コマンド」と「アクション」**

4. 1　「コマンド」と「アクション」の違い

戦闘を構成するのが、「コマンド」と「アクション」である。

この二つはそれぞれGame\_UnitCommandクラスとGame\_BattleActionクラスに相当する。

「コマンド」は、いわゆる人形や敵の戦闘行動を指す。例えば「通常攻撃」「特技」「防御」「待機」などがコマンドである。

人形のコマンドはDOLLS\_COMMAND、敵のコマンドはENEMIES\_COMMANDフェイズで決定される。人形の行動はプレイヤーが選び、敵のコマンドは敵のデータベースから自動的に選択される。これについてはここでは詳しくは述べない。

これに対して「アクション」は、選ばれた「コマンド」に対して、その時々に起こる「動作」である。

例えば「通常攻撃」を行う場合、「○○の攻撃！」と表示され、続いて「○○に△のダメージ！」とダメージの表示とダメージの適用が行われる。このゲームでは、これを「攻撃宣言」というアクションと「ダメージ計算・適用」というアクションに分けている。基本的に、エフェクトの更新やメッセージウィンドウの更新を伴う行動は「アクション」に分かれていると考えて良い。

4. 2　「コマンド」の処理され方

ひとえに「通常攻撃」と言っても、実際の処理のされ方は様々である。例えば、

・コマンド実行前に敵の攻撃で倒れていて、コマンド自体が無効になる

・混乱していて、味方を攻撃してしまう

・かばわれて、ターゲットが狙った相手からずれる

・パッシブスキル（常時発動スキル）により通常攻撃が全体攻撃になる

・「反撃」コマンドにより攻撃の反動を受ける

・相手の回避行動により、攻撃が外れる

などである。このように、選択されたコマンドは戦闘時、様々なタイミングで処理を行う必要がある。この諸処理を行うのが、Scene\_BattleクラスのInterpretCommandという関数である。

バトルシーンにはphazeというフェイズ管理の変数の他に更にcommandPhazeという値があり、このcommandPhazeは以下の値を取る。

（Static\_Battle.hから抜粋）

COMMANDPHAZE\_NOPHAZE

COMMANDPHAZE\_BEFORE\_SORT

COMMANDPHAZE\_AFTER\_SORT

*COMMANDPHAZE\_FIX\_COMMAND*

*COMMANDPHAZE\_FIX\_TARGET*

*COMMANDPHAZE\_ASSERT*

*COMMANDPHAZE\_PRE\_ACTION*

*COMMANDPHAZE\_ACTION*

*COMMANDPHAZE\_CHECK\_DEATH*

*COMMANDPHAZE\_POST\_ACTION*

このうち、斜体は実際にコマンドを実行する際に使用する値であり、その他は、コマンドの実行とは異なるタイミングで使用される値である。以下に、それぞれざっと説明する。

・COMMANDPHAZE\_NOPHAZE

とりあえず考えなくて良い。

・COMMANDPHAZE\_BEFORE\_SORT

前項のコマンド選択において、コマンドを選択した後は、コマンドを素早さ順に並べ替え、その順にコマンドを処理していくということをする。

しかし、例えばスキル「先制攻撃」の時は、素早さ順に並び替える際に、素早さ補正の処理を行う必要がある。「並べ替えの際に素早さを補正する」という処理を、このゲームでは「並び替えの時だけ素早さを上げる」というステート（状態異常）で表現している。

そのため、「先制攻撃」というスキルを使う時は、このフェイズの時に、使用者を「素早さアップ」の状態異常にするという処理を行うことで、正しく並べ替えが行われるようにする。

・COMMANDPHAZE\_AFTER\_SORT

　コマンドの並べ替えが終わったあと、全てのコマンドの実行前に実行される。

　例えば「防御」コマンドでは、自分の行動順序によらずターンの最初から全てのダメージに対して防御補正を適用するが、それは、このフェイズにおいて防御の使用者を防御ステートにすることによって実現する。

・COMMANDPHAZE\_FIX\_COMMAND

　ここからは、各コマンド実行時に行われる内容。

　ここではまず、そもそもコマンドがそのまま実行されるかの判定を行う。パッシブスキルにより通常攻撃が全体攻撃になったり、自分がやられていることによってコマンドが無効になったりというのはこのフェイズで判断する。

・COMMANDPHAZE\_FIX\_TARGET

　コマンドのターゲットを決定するフェイズ。

　実は、全体攻撃などターゲットが複数あるいは不定の場合はCOMMANDPHAZE\_ACTIONにおいて再びターゲットが決定される他、「かばう」コマンドを実行した場合はそのかばう・かばわれるの判定はCOMMANDPHAZE\_ACTIONにおいて行う。全部COMMANDPHAZE\_ACTIONでやってもいいのでは？とも思うが、COMMANDPHAZE\_ASSERTでターゲットの名前を使ったりすることもあるだろうという判断で残している。

・COMMANDPHAZE\_ASSERT

「○○の攻撃！」「○○は××を使った！」など、コマンドを宣言するフェイズ。

・COMMANDPHAZE\_PRE\_ACTION

実際にコマンドを実行する前に行うフェイズ。

毒を受けていて攻撃前にダメージを受けるとか、そういう時に使う。

・COMMANDPHAZE\_ACTION

　行動を適用するフェイズ。通常攻撃であれば、「○○のダメージ！」と表示し、さらに実際にHPを減らすのがこのフェイズに相当する。

　このフェイズはやや特殊であり、コマンド（Game\_UnitCommandクラス）を一旦コマンド群（Game\_UnitSubCommandクラス）に解釈しなおす作業を最初に行う。すなわち、例えば「全体攻撃」を「各敵への一体ずつの攻撃」というコマンド群に分解し、「毒攻撃」を「攻撃」と「毒ステートの付加」に分解する。その上で分解された各コマンドについて解釈を行い、ダメージ表記や演算を行う。

例えば、「毒攻撃」は攻撃が外れたのに毒状態になる、というのはおかしい。そのため、「毒攻撃」は「攻撃」と「毒の付加」に分けてからそれぞれの成否判定を行うのではなく、全体として成功するか失敗するかを判定し、その上で失敗したら「失敗した」というコマンドのみに、攻撃は当たったが毒は付加されなかったら「ダメージ」というコマンドのみに、毒の付加に成功したら「ダメージ」「毒付加」にそれぞれ分解する。すなわち、分解された後の毒の付加は「確定した」毒の付加であり、毒付加アクションを実行する際に毒の付加の成否の判定を行うわけではない。

・COMMANDPHAZE\_CHECK\_DEATH

　各ユニットの死亡判定を行う。

　また、戦闘終了の判定を行う。

　基本的に、処理の内容は実行したコマンドの内容と関係ない。常にコマンドごとに、誰か死んでいないか、戦闘終了フラグが立っていないか、の判定を行う。

・COMMANDPHAZE\_POST\_ACTION

　コマンド後の判定を行う。たとえば、次の一撃のみ威力が上がるスキルは、使用時に「次の一撃強化」ステートを付加し、このフェイズにおいてそのステートを解除することで実装する。