

# CTB by Charlie Flead v3.2

## Instructions

### **Table of contents.**

[Installation.](#)

[Advanced targetting.](#)

[Configuration of Individual Battle Commands.](#)

[Configuration of SuperArts.](#)

[Configuration of Summons.](#)

[Switching Party Members.](#)

[Steal and Thievery.](#)

[Flying Enemies and Actors.](#)

[Graphics.](#)

[Battle Formulas Basics.](#)

[Blue Magic.](#)

[ATB Mode.](#)

[Grandia Mode.](#)

[Custom States.](#)

## Installation.

In the following I will assume that the reader knows how to navigate and modify the database and how to open the script editor and copy/paste scripts.

STEP 1 Copy all the icons, pictures and window skins you find in the folders of the demo.

STEP 2 Copy all the scripts from “Bitmap, Draw Face” down to “Super Arts” into your project. They must be above the “Main” script and below all the other default scripts. If you have other custom scripts in your project, there is not a general rule about where to copy the battle system scripts with respect to them; it will be a case by case decision. Generally the mutual positions of custom scripts affect their behavior and correct operation. However, keep in mind that scripts that are not specifically made to work together, might not work together (!) no matter how you place them. I can only guarantee that this script works in a brand new project where it is the only added custom script.

STEP 3 Replace the “Configuration” script page with the one named “blank config.rtf” that you find in the demo main folder.

STEP 4 Open the database and go to the System tab. Copy all the elements below “# TAGS #” down to “Random Single Target”. Actually they are not all necessary for the system to work; you will probably need to add some, and you can change the names as long as your change is replicated in the scripts (more on this later). For now, you are safe if you copy them all, and keep their names unchanged. Locate the ELEMENTS\_TO\_IGNORE array in the Configuration script and put into this array all the positions taken by the elements you just copied.

STEP 5 Go to the Common Events tab and copy the events “Trincea” and “Altruismo”. The other common events are Summons and skills. You will learn how to create your own summons in a following section, hence ignore them now. Skills made for the demo can be studied as examples to make your own skills.

STEP 6 Go to the States tab and copy all the states after the “\*\* CUSTOM STATES \*\*” line.

STEP 7 Go to the Skills tab and copy all the skills from Drain to Retire. Note that some of them point to common events; if you copied the common events into positions that are different from the demo, you will have to adjust the common event field after you copy and paste. All of them have elements assigned to them (a technique that is usually referred as “tagging”); if you copied the elements into positions that are different from the demo, you will have to adjust the elements that are checked after you copy and paste. The other non-default skills you find in the demo are summons, superarts, or skills that exploit the states you copied in the previous step. You don't necessarily need them, you can use them as examples/templates. Also, you're not going to use Switch and Retire if you, respectively, don't enable switching and summons.

STEP 8 Read and follow the instructions in the section “Configuration of Individual Battle Commands”.

STEP 9 Configure “Animated Battlers - Enh ver. 13.3” (Animated Battlers by Minkoff, Enhancements by DerVVulfman). This is a script that this battle system uses and provides the animation system. It has its own configuration guide that you will find in the demo folder.

Done? Very well. Now you should be able to use the battle system without summons, superarts, switching and a few other things that will be explained in the following sections.

## **Advanced Targetting.**

Use the Left/Right arrow keys to switch between the party and the monsters when using skills tagged with the "Any Scope" element.

Use the key/button configured as Z (default is Shift) to switch between single and "all" target when using skills tagged with the "Single/All" element.

## Configuration of Individual Battle Commands.

Individual Battle Commands is a feature that lets you group skills in categories, so that they can be displayed under different submenus in the command window of actors during battles.

Let us assume that you want to create "Black Magic" and "White Magic".

Create two elements named "CMD White Magic" and "CMD Black Magic". For each skill belonging to the "White Magic" category, check the "CMD White Magic" element. Do the same with "Black Magic" skills and the "CMD Black Magic" element.

Finished. Yes. Simple, isn't it?

Remember: if "Name" is your category, use an element called "CMD Name". "Name" will show up in the command window of the actors, and it will be displayed only if that actor has at least one skill of that category.

### Individual Battle Commands in Menu

The battle system provides an optional edit of the Skills menu screen. This edit will show skills in separate categories (e.g. "White Magic", "Black Magic", "Techs", ...) and will allow the user to rotate through the categories by using the SHIFT key. Since this edit is expected to interfere with custom menu system, it can be disabled by putting

```
IBC_MENU_SUPPORT = false
```

in the configuration.

## Configuration of SuperArts.

In order to use superarts you need two things: an element "Superart" and a second element "CMD Superarts".

Every skill that has to be a superart needs to be "tagged" with these two elements. It will appear under the Superart command and will be available only when the superart bar is filled.

So far, the following fill modes have been implemented.

Rage: gain Superart points by inflicting damage.

Pain: gain Superart points by suffering damage.

Empathy: gain Superart points by damage inflicted to allies.

Category: gain Superart points when skills of determined categories are used.

Critical: gain Superart points when with low HP.

Damage: a mix between Rage and Pain.

You can switch between the modes from the Status Scene, by pressing SHIFT, but in order to see that, you need a custom Status Scene script.

The speed at which the bar will fill depends on adjustable rates, one for each fill mode. You can set the default values in the configuration:

```
TYPE_0_RATE=2  
TYPE_1_RATE=0.75  
TYPE_2_RATE=1  
TYPE_3_RATE=0.25  
TYPE_4_RATE=0.1  
TYPE_5a_RATE=0.5  
TYPE_5b_RATE=0.5
```

During the game, you will be able to change them by using a Call Script command with, for example, the code:

```
$game_system.type_0_rate = 3
```

## Configuration of Summons.

There are little differences between the configuration of the different types of summons.

### Type 1 and 2 summons.

#### STEP 1

Type-1 summons do not remove the party (think about Warcraft), while type-2 do (think about FFX). In order to configure a summon, go in the common events tab of the database and create a new common event, call it as you like, and put a call script instruction like this:

```
"$game_party.summon(<type>,[ "actor1 name", "actor2 name", ... ])",
```

where <type> is 1 or 2.

Examples:

```
$game_party.summon(1, ["Kerya"])           # Type 1, summons Kerya
$game_party.summon(2, ["Kerya", "Adel"])   # Type 2, summons Kerya and Adel
```

You can put the command in more than one line:

```
$game_party.summon(2, ["Kerya",
"Adel"])
```

**VERY IMPORTANT: do not break the lines in the middle of an actor's name:**

```
$game_party.summon(2, ["Black
Night"])
```

is **WRONG** and will not work.

#### STEP 2

Then you must create a skill in the skills tab of the database, which your characters will use to summon, and make the common event field of this skill point to the common event you have just created.

All the summon skills must be tagged with an element whose name matches the name configured in the script as SUMMON\_NAME. The default name is "Summon". This name will not appear in the battle command window, where you will see the name of the category you assign to the skill (with a "CMD <category name>" element).

You will also need a skill to make your "Aeons" retire.

1. Make a new category by creating the element "CMD Retire";
2. create a skill "Retire" (or use any name you configured as RETIRE\_NAME);
3. make sure that all the "Aeons" learn the skill;
4. tag the skill with the "CMD Retire" element.

#### STEP 3

Finally, in order to correctly use your "Aeons", visualize their stats and making them learn skills (in the proper Scene), you need to add them to your party. Do it by using event script commands like the following:

```
$game_party.aeons.
push($game_actors[13])
$game_party.aeons.
push($game_actors[14])
```

NOTE: this step is not necessary if you have AUTO\_ADD\_ACTORS = true. However the use of this option has some side effects: when an actor that is not in the party learns a summon skill, the corresponding "Aeons" are added to the party. This may happen when you use the "Change Skills..." command event for actors not in the party and at the beginning of the game when all the actors in the database are processed and learn their starting set of skills. To avoid this side effect, make the actors learn the summon skills in the same moment that they join the party.

### Type-3 summons (Transformations).

#### STEP 1

Type-3 summons are used for transformations. Transformed actors will be removed and their new form will appear on the battlefield. Their form will return normal only after the end of the battle. In order to configure a type-3 summon, go in the common events tab of the database and create a new common event, call it as you like, and put a call script instruction like this:

Example:  

```
$game_party.summon(3, ["Ice Golem"])           # Type 3, transform into Ice Golem
```

You must specify type 3 and put only one actor in the list.

#### STEP 2

Then you must create a skill in the skills tab of the database, which your characters will use to transform, and make the common event field of this skill point to the common event you have just created.

Also type-3 summon skills must be tagged with an element whose name matches the name configured in the script as SUMMON\_NAME. The default name is "Summon". Again this is not the name that will appear in the battle command window.

Type-3 summons do not and cannot use a "Retire" skill.

### STEP 3

Finally, in order to correctly use your "Transformations" and visualize their stats (in the proper Scene), you need to add them to your party. Do it by using event script commands like the following:

```
$game_party.transformations.  
push($game_actors[13])  
$game_party.transformations.  
push($game_actors[14])
```

NOTE: this step is not necessary if you have AUTO\_ADD\_ACTORS = true. However the use of this option has some side effects: when an actor that is not in the party learns a summon skill, the corresponding "Aeons" are added to the party. This may happen when you use the "Change Skills..." command event for actors not in the party and at the beginning of the game when all the actors in the database are processed and learn their starting set of skills. To avoid this side effect, make the actors learn the summon skills in the same moment that they join the party.

### **Custom BGM.**

A common event code like the following

```
bgm("003-Battle03");  
$game_party.summon(2, ["Adel", "Kerya"])
```

can be used to make a BGM start as the summoned actors appear. The bgm command also support the indication of a volume value:

```
bgm("003-Battle03", 50);
```

### **Custom Animation.**

A common event code like following

```
summon_animation(106);  
$game_party.summon(2, ["Getter 1"])
```

can be used to make an animation be played on every summoned actor as he appears. This setting will override the animation ID configured with SUMMONED\_ACTORS\_ANIMATION\_ID.

## Switching Party Members.

In order to use the switching feature you need a skill.

Simply create the new skill, and don't bother configuring all the parameters, the important thing is its name, which must be the same name configured as SWITCH\_NAME.

**YOU DON'T HAVE TO ADD THE SWITCH SKILL TO YOUR CHARACTERS, JUST CREATE THE SKILL IN THE DATABASE.**

Finally go in the configuration page of the script and locate this section:

```
#-----  
# PARTY ACTORS SWITCHING #  
#-----  
SWITCHING = true  
SWITCHED_ACTOR_ANIMATION_ID = 2  
# Applied to actors who are in the backup party when the battle ends  
SWITCHED_ACTORS_EXP_MULTIPLIER = 0.75  
SCENE_SWITCH_MENU_ENTRY = 0  
SCENE_SWITCH_RETURN_TO_MAP_ON_EXIT = true
```

You can turn on/off the swiching feature. You can configure an animation to be displayed when switching actors, and how much exp points the backup party members get at the end of the battle.

You can also decide whether exiting from the Switch Scene will open the Menu Scene or return the player to the Game Map. If you decide to open the Menu, you can configure the index of the menu command corresponding to the Switch Scene, which will be automatically selected.

In order to add/remove backup actors use a script command like this within some event:

```
$game_party.add_backup_actor(ID)  
$game_party.remove_backup_actor(ID)
```

where ID is the id of the actor in the database.

If you try to add a backup actor that is already in the backup party, nothing will happen.

If you try to remove a backup actor that is not in the backup party, nothing will happen.

## Steal and Thievery.

The user can define the objects that can be stolen from monsters. For each monster a common item and a rarity can be configured. Rarities can be stolen only when the character which uses the Steal skill has the Thievery state. A single item can be stolen each time and there is no limit to the number of times the user can steal from a monster. The success is random and depends on the comparison between the agilities of the character and the enemy. You will find a hash definition like this one in the configuration:

```
ENEMY_OBJECTS =  
                {  
                  35 => [1, 19],  
                  34 => [4, 15]  
                }
```

which uses the following syntax: *enemy\_id* => [*item\_id*, *rarity\_id*]

If an enemy does not appear in the hash the user won't steal anything.

You always have to specify an *item\_id*, but if you don't want a rarity use "nil" in the place of the *rarity\_id*. Example:

```
34 => [4, nil]
```

## **Flying Enemies and Actors.**

Flying enemies cannot be hit by physical attacks unless the attacker uses a ranged weapon (a bow, a gun, etc). Skills included in the MNK\_MOVING\_SKILL array in the configuration of Animated Battlers are considered as physical attacks and cannot hit flying enemies. Put the IDs of flying enemies in the FLYING\_ENEMIES array in the configuration.

Flying actors can hit flying enemies with no limitations. Put the IDs of flying actors in the FLYING\_ACTORS array in the configuration.

## Graphics.

There are lots of possibilities for those who want to customize the graphic layout of the BS.

### Skills and Items windows.

```
#-----  
# SKILL and ITEM WINDOW #  
#-----  
SKILL_ITEM_WINDOW_WIDTH =          480  
SKILL_ITEM_WINDOW_HEIGHT =        24 * 6 + 32  
SKILL_ITEM_WINDOW_X =              32  
SKILL_ITEM_WINDOW_Y =             120  
SKILL_ITEM_WINDOW_ROW_SIZE =       24  
# Put "" to use the system skin  
SKILL_ITEM_WINDOW_WINDOWSKIN_NAME = ""  
SKILL_ITEM_WINDOW_WINDOWSKIN_OPACITY = 255  
SKILL_ITEM_WINDOW_TEXT_COLOR =     Color.new(255,255,255)  
# Value relative to row height  
SKILL_ITEM_WINDOW_TEXT_SIZE_OFFSET = 0  
SHOW_COST_WHEN_ZERO =              false  
# Put "" if unused  
SKILL_WINDOW_BG_PICTURE_NAME =     "skill bg"  
SKILL_WINDOW_BG_PICTURE_X =        -8  
SKILL_WINDOW_BG_PICTURE_Y =        -8  
# Put "" if unused  
ITEM_WINDOW_BG_PICTURE_NAME =      "item bg"  
ITEM_WINDOW_BG_PICTURE_X =         -8  
ITEM_WINDOW_BG_PICTURE_Y =         -8  
SKILL_COST_AUTOSORT =               true  
ITEM_COST_AUTOSORT =                true
```

You can personalize the Skill and Item Windows' sizes and positions, the row size, the name and the opacity of the window skin as well the text size and color.

Here you also decide whether the cost for skills with cost equal to zero must be displayed or not.

A custom background image can be used as well.

Finally, both skills and items can be automatically sorted based on their cost.

### Switch-like windows.

```
#-----  
# SWITCH WINDOW #  
#-----  
SWITCH_WINDOW_WIDTH =              (STATUS_WINDOW_STYLE==1) ? 480 : 512  
SWITCH_WINDOW_HEIGHT =              80 + 32  
SWITCH_WINDOW_X =                   (STATUS_WINDOW_STYLE==1) ? 160 : 128  
SWITCH_WINDOW_Y =                   368  
SWITCH_WINDOW_ROW_SIZE =            20  
# Put "" to use the system skin  
SWITCH_WINDOW_WINDOWSKIN_NAME =     ""  
SWITCH_WINDOW_WINDOWSKIN_OPACITY =  255  
SWITCH_WINDOW_TEXT_COLOR =          Color.new(255,255,255)  
# The following categories of skills will use the settings for the switch window  
# in order to not to cover the help window  
SWITCH_LIKE_CATEGORY_NAMES =         ["Summon", "Transform"]  
SWITCH_HELP_WINDOW_WINDOWSKIN_OPACITY = 255  
# Put "" if unused  
SWITCH_WINDOW_BG_PICTURE_NAME =     "" # "switch_bg"  
SWITCH_WINDOW_BG_PICTURE_X =        -6  
SWITCH_WINDOW_BG_PICTURE_Y =        -8  
SWITCH_HELP_WINDOW_X =              128  
SWITCH_HELP_WINDOW_Y =              166  
# Put "" if unused  
SWITCH_HELP_WINDOW_BG_PICTURE_NAME = "" # "switch_help_bg"  
SWITCH_HELP_WINDOW_BG_PICTURE_X =   -6  
SWITCH_HELP_WINDOW_BG_PICTURE_Y =   -8
```

The term refers to the Switch Window and the Skill Windows used for some categories of skills as specified in the SWITCH\_LIKE\_CATEGORY\_NAMES array.

These windows have their own settings (size and position, row size, text color, window skin name and opacity) because when used, a help window is typically shown with additional information (stats of the summons, or of the backup party members).

You can also set the opacity of the Help Window displayed to show additional information for summons, transformations and backup party members. As with Skills and Items windows, background images can be used for switch-like windows.

### Actor Command window.

```
#-----  
# ACTOR COMMAND WINDOW #  
#-----  
# Put "" to use the system skin  
ACTOR_COMMAND_WINDOWSKIN_NAME = ""  
ACTOR_COMMAND_WINDOW_WINDOWSKIN_OPACITY = 0  
# 160 recommended with status window style 1, 128 recommended with status  
# window style 2  
ACTOR_COMMAND_WINDOW_WIDTH = (STATUS_WINDOW_STYLE==1)?160:128  
ACTOR_COMMAND_WINDOW_X = 0  
ACTOR_COMMAND_WINDOW_TEXT_COLOR = Color.new(255,255,180)  
ACTOR_COMMAND_WINDOW_TEXT_BG_COLOR = Color.new(0,0,0)  
# Value relative to row height  
ACTOR_COMMAND_WINDOW_TEXT_SIZE_OFFSET = 0  
# Put "" if unused  
ACTOR_COMMAND_BG_PICTURE_NAME = "style 3 actor_command_bg"  
ACTOR_COMMAND_BG_PICTURE_X = 0  
ACTOR_COMMAND_BG_PICTURE_Y = -8  
ACTOR_COMMAND_CURSOR_MEMORY = true  
LEFT_ACTOR_COMMAND_WINDOWSKIN_OPACITY = 0  
# Put "" to use the system skin  
LEFT_ACTOR_COMMAND_WINDOWSKIN_NAME = ""
```

For the Actor Command Window, you can decide the name and opacity of the window skin, its width, its position, the color for the text and for the text border as well as the text size. You can also place a background image behind the Actor Command Window and here you can set its name. You can turn on and off the cursor memory. The Left Command Window is used to show the Defend, Skip, Escape and Status commands. You can define its window skin name and its opacity.

### Face of the active battler.

```
#-----  
# BATTLER WINDOW #  
#-----  
BATTLER_FACE_VISIBLE = false  
BATTLER_FACE_SMALL = false  
BATTLER_WINDOW_Y = 368  
BATTLER_WINDOW_X = 20  
BATTLER_FACE_SIZE = 80  
BATTLER_FACE_OPACITY = 128  
BATTLER_FACE_WINDOWSKIN_OPACITY = 0
```

The Battler Window displays the face of the active battler. Here you decide to make it visible or not. When `BATTLER_FACE_SMALL` is false the system will look for the picture `Graphics/Faces/<character_name.png>`, whereas with `BATTLER_FACE_SMALL` true the system will first look for the picture `Graphics/Faces/<character_name-small.png>`, and then `Graphics/Faces/<character_name.png>` when the first one cannot be found. Here you can set the position and size of the window, the opacity of its window skin and the opacity of the face image.

### Faces of the actors in the battle status.

The `STATUS_FACE_VISIBLE` is true, the Status Window displays pictures for the faces of the party members. When `USE_BATTLE_STATUS_FACES` is false, for each party member, the system looks first for the picture `Graphics/Faces/<character_name>-small.png`, and then for `Graphics/Faces/<character_name>.png`. When `USE_BATTLE_STATUS_FACES` is true, for each party member, the system looks for the picture `Graphics/Faces/<character_name>-battlestatus.png` and displays it unresized.

### Turns window.

```
#-----  
# TURNS WINDOW #  
#-----  
TURNS_WINDOW_VISIBLE = (true and not (ATB_MODE or GRANDIA_MODE))  
# Number of displayed turns  
BATTLERS_QUEUE_SIZE = 10  
# Height of each box
```

```

TURN_HEIGHT = 22
# Use pictures for the party actors
TURN_USE_PICTURES_ACTORS = true
# Use pictures for the enemies
TURN_USE_PICTURES_ENEMIES = false
# Y-coordinate for the turns window
TURNS_WINDOW_Y = 80
# X-coordinate for the turns window
TURNS_WINDOW_X = 526
# Put "" if unused
TURNS_BG_NAME = "" # "turns_bg"
TURNS_WINDOWSKIN_OPACITY = 0
TURNS_WINDOW_VERTICAL = true
# The following values are used when TURNS_WINDOW_VERTICAL is false
# Width of each box
TURN_WIDTH = 40
# Y-coordinate for the turns window
TURNS_WINDOW_H_Y = 64
# X-coordinate for the turns window
TURNS_WINDOW_H_X = 0
TURNS_WINDOW_HEIGHT = 64
UPDATE_ONLY_CHANGES = true

```

First, you can decide to not use a Turns Window. In that case you can set `TURNS_WINDOW_VISIBLE = false` and get rid of it.

If you are going to use this window you can decide how many turns it will display by setting the value of `BATTLERS_QUEUE_SIZE`. If you disable the window, setting `BATTLERS_QUEUE_SIZE=1` is recommended as this will make the battle system skip a lot of unnecessary processing.

You can decide the height of the boxes (or pictures) that represent each turn.

You can decide whether to use pictures or not for the actors and for the enemies.

If you use pictures, the system will look for them in the `Graphics/Pictures/Turns/` folder, and each picture must be named exactly as the character or monster you want it to be used for. If you have set `TURNS_WINDOW_VERTICAL = false`, the system will add the postfix “ H” to the names of the pictures to look for. When a picture cannot be found the system will automatically use text and display the name of the battler.

When displaying text, the system uses three background pictures: they are `Graphics/Pictures/Turn-1.png` , `Turn-2.png` and `Turn-3.png`. Of course you can change the pictures used in the demo and use yours.

Note that you will obtain better results if you use images with the exact size they will appear on the screen, so that RMXP will not perform any resize.

You can decide the position of the window, the name of an optional background image (to be placed in the `Graphics/Pictures/` folder), and the window skin opacity.

The Turn Window can display the turns both in a vertical and a horizontal disposition. When using a horizontal disposition, by setting `TURNS_WINDOW_VERTICAL = false`, you can set the width of the turns, the position of the window and its height.

Finally, the `UPDATE_ONLY_CHANGES` value controls whether the Turns Window will be updated every time the selected command is changed or only when the change produces a variation in the order of the displayed turns.

## Report window.

```

#-----
# REPORT WINDOW #
#-----
# Put "" to use the system skin
BATTLE_REPORT_WINDOW_WINDOWSKIN_NAME = ""
BATTLE_REPORT_WINDOW_WINDOWSKIN_OPACITY = 255
BATTLE_TREASURES_WINDOW_WINDOWSKIN_OPACITY = 255
# Put "" if unused
BATTLE_REPORT_BG_PICTURE_NAME = ""
BATTLE_REPORT_BG_PICTURE_OPACITY = 255
# Put "" if unused
BATTLE_REPORT_BACKGROUND_NAME = ""
BATTLE_REPORT_BACKGROUND_OPACITY = 200

```

The battle Report Window displays information about the experience gained and the levels reached by the characters at the end of a battle.

The window skin name and the window skin opacity can be configured.

A box is used for each character, and you can configure the name of an optional background picture that can be displayed behind each box, with a configurable opacity. You can also configure the name of an optional background image for the entire screen, as well as its opacity.

## Status window.

```

#-----
# STATUS WINDOW #
#-----

```

```
# Adjust ACTOR_COMMAND_WINDOW_WIDTH and STATUS_WINDOW_X accordingly
STATUS_WINDOW_STYLE = 3
```

There are three styles available for the Status Window. The first one can display up to 6 battlers, each one in a row; the second one can display up to 8 battlers in a 2 rows x 4 columns table; the third one can display up to 4 battlers.

```
STATUS_FACE_VISIBLE = true
USE_BATTLE_STATUS_FACES = false
```

These options have been discussed before.

```
USE_STATES_ICONS = true
STATUS_WINDOW_X_OFFSET = 0
# 220 recommended with status window style 1, 112 recommended with
# status window style 2
STATUS_WINDOW_X = (STATUS_WINDOW_STYLE==1)?158:112
# Colors for the names
STATUS_WINDOW_TEXT_COLOR = Color.new(255,240,200)
STATUS_WINDOW_TEXT_BG_COLOR = Color.new(0,0,0)
# Colors for the "HP" text in the HP bar
STATUS_WINDOW_HP_COLOR = Color.new(245,220,130)
STATUS_WINDOW_HP_BG_COLOR = Color.new(0,0,0)
# Colors for the HP value in the HP bar
STATUS_WINDOW_NORMAL_TEXT_HP_COLOR = Color.new(220,240,250)
STATUS_WINDOW_CRISIS_TEXT_HP_COLOR = Color.new(160,180,190)
STATUS_WINDOW_KNOCKOUT_TEXT_HP_COLOR = Color.new(80,100,110)
# Colors for the "SP" text in the SP bar
STATUS_WINDOW_SP_COLOR = Color.new(160,230,240)
STATUS_WINDOW_SP_BG_COLOR = Color.new(0,0,0)
# Colors for the SP value in the SP bar
STATUS_WINDOW_NORMAL_TEXT_SP_COLOR = Color.new(255,175,175)
STATUS_WINDOW_CRISIS_TEXT_SP_COLOR = Color.new(195,115,115)
STATUS_WINDOW_KNOCKOUT_TEXT_SP_COLOR = Color.new(155,55,55)
STATUS_WINDOW_TEXT_SIZE_OFFSET = -4
STATUS_WINDOW_HPSP_BAR_OPACITY = 255
STATUS_WINDOW_HPSP_BAR_BG_OPACITY = 220
STATUS_WINDOW_HPSP_BAR_HEIGHT = 8
STATUS_WINDOW_V_SPACING = 4
# Put "" to use the system skin
STATUS_WINDOW_WINDOWSKIN_NAME = ""
STATUS_WINDOW_WINDOWSKIN_OPACITY = 0
STATUS_WINDOW_VISIBLE_DURING_REPORT = false
# Put "" if unused
STATUS_WINDOW_ACTOR_BG_NAME = "style 3 actor bg"
# Put "" if unused
STATUS_WINDOW_ACTOR_BG_SEL_NAME = "style 3 actor bg selected"
STATUS_WINDOW_ACTOR_BG_OPACITY = 192
STATUS_WINDOW_ACTOR_BG_SEL_OPACITY = 192
STATUS_WINDOW_CURSOR_OPACITY = 0
```

You can decide whether to use pictures or text for the states.

The STATUS\_WINDOW\_X\_OFFSET value is used with style 1 and makes the rows shift a little bit to one another.

You can set the X coordinate of the window.

Colors can be configured for the text and text border displaying the names of the actors.

Colors can be configured for the text and text border displaying the text "HP" in the hp bar.

Colors can be configured for the numbers displayed in the hp bar, differentiating between the three conditions Normal, Crisis, Knockout.

Colors can be configured for the text and text border displaying the text "SP" in the sp bar.

Colors can be configured for the numbers displayed in the sp bar, differentiating between the three conditions Normal, Crisis, Knockout.

The value of STATUS\_WINDOW\_TEXT\_SIZE\_OFFSET controls the size of the text.

You can configure the opacity of the bars and their backgrounds, as well as their height.

With STATUS\_WINDOW\_V\_SPACING you can add some pixels of space between each row (this only affects style 1).

You can set the name and the opacity of the window skin.

You can decide whether to show or not the Status Window during the report phase at the end of the battle.

Background images can be used behind every actor information box, one for the selected actor, one for the non-selected actors. Opacity for these images can be set.

Finally the opacity of the cursor can be set.

## Battle Formulas Basics.

### Physical Attacks

The resulting damage of physical attacks depends linearly on the attack value of the used weapon, and on one parameter of the user: the strength for close combat weapons, and the dexterity for ranged weapons. In formulas:

$$\text{attack} = \text{ATK} * \text{STR} \text{ or } \text{attack} = \text{ATK} * \text{DEX}.$$

At this point, the algorithm uses the value configured as `ATTACK_DAMAGE_RANGE_CORRECTION` which gets multiplied by the attack power, obtaining the final attack power:

$$\text{attack} = \text{attack} * \text{ATTACK\_DAMAGE\_RANGE\_CORRECTION}$$

The target's defense value will be subtracted from the attack to get the damage:

$$\text{damage} = \text{attack} - \text{defense}.$$

The *defense* value equals the PDEF parameter (or, when `SQUARED_DEFENSE` is true, its square  $\text{PDEF} * \text{PDEF}$ ) multiplied by `DEFENSE_RANGE_CORRECTION`.

### Skills

The power of the skill is used in combination with the caster's parameters, using the -F modifiers (`ATK-F`, `STR-F`, `INT-F`, `DEX-F`, `AGI-F`) in the database skills tab.

For example, for a skill with `power=40`, `ATK-F=35`, `INT-F=80`, the attack power will be:

$$\text{attack} = 40 * (0.35 * \text{caster's ATK value} + 0.80 * \text{caster's INT value}).$$

If all the modifiers are null, the skill's power directly represents the attack power:

$$\text{attack} = 40.$$

At this point, the algorithm uses the value configured as `SKILLS_DAMAGE_RANGE_CORRECTION` which gets multiplied to the attack power, obtaining the new attack power:

$$\text{attack} = \text{attack} * \text{SKILLS\_DAMAGE\_RANGE\_CORRECTION}$$

To get the damage, a *defense* value is calculated and subtracted to the *attack* value.

- When `F_FACTORS_DEFENSE_SCALING` is false the `PDEF-F` and `MDEF-F` values in the database are used to decide how much of the skill's damage can be absorbed with physical and/or magical defense.  
If a skill deals a damage of 500 and you have `PDEF-F=30` and `MDEF-F=25`, then up to  $500 * 0.30 = 150$  points can be absorbed by the physical defense, and up to  $500 * 0.25 = 125$  points by the magical defense of the target.
  - If `SQUARED_DEFENSE` is true, the system will use the squares of `PDEF` and `MDEF`.
- When `F_FACTORS_DEFENSE_SCALING` is true the `PDEF-F` and `MDEF-F` values are used as multipliers instead of limiters, and the defense value is calculated as  $\text{defense} = \text{MDEF-F} / 100.0 * \text{MDEF} + \text{PDEF-F} / 100.0 * \text{PDEF}$ .
  - If `SQUARED_DEFENSE` is true, the system will use the squares of `PDEF` and `MDEF` instead.

Then the algorithm uses the value configured as `DEFENSE_RANGE_CORRECTION` to get the final value of *defense*:

$$\text{defense} = \text{defense} * \text{DEFENSE\_RANGE\_CORRECTION}$$

And finally the damage is calculated as:

$$\text{damage} = \text{attack} - \text{defense}.$$

### Beyond skills' power limits.

The ExponentialDamage element can be used to change the meaning of the power value of a skill. For a skill tagged with this element the actual power will be  $e^{(\text{power}/10.0)}$ .

Here is a reference table:

10	--> 2
15	--> 4
20	--> 7
25	--> 12
30	--> 20
35	--> 33
40	--> 54

45 --> 90  
50 --> 148  
60 --> 403  
70 --> 1096  
80 --> 2980  
90 --> 8103  
100 --> 22026  
110 --> 59874  
120 --> 162754

As a rule of thumb, each 1 point increment means a 10% increment in the damage (70-->1096, 71-->1211).

The DamageX10 element can be used to multiply the skill's power by 10.

The PropSkPow element can be used to create skills that deal a damage proportional to the damage of a physical attack, given the current stats and equipment of a character. The considered value for the physical attack damage will also take into account the difference between close combat weapons and ranged weapons.

The power of the skill as configured in the database indicates the percentage of damage that skill will deal, with 100 indicating a skill that deals as much damage as a physical attack, 200 indicating a skill that deals twice the damage of a physical attack and so on. Note that if you are using different values for ATTACK\_DAMAGE\_RANGE\_CORRECTION and SKILLS\_DAMAGE\_RANGE\_CORRECTION, that will affect the damage dealt by these skills too. For example, with ATTACK\_DAMAGE\_RANGE\_CORRECTION=1.0 and SKILLS\_DAMAGE\_RANGE\_CORRECTION=2.0, a skill with the PropSkPow element and power=150 will deal  $150/100*(2.0/1.0)=3.0$  times the damage of a physical attack.

## **Blue Magic.**

An element name can be defined with

```
BLUE_MAGIC_NAME = "Blue Magic"
```

that allows skills tagged with that element to be learned from enemies, using the skill named as specified in

```
DRAKOKEN_NAME = "Drakoken"
```

## **ATB Mode.**

By setting

```
ATB_MODE = true
```

in the configuration, you will see little bars indicating the time to the next turn for each battler. This feature does not change the flow of the actions in the battle system, it is only a different way to show the turns.

## Grandia Mode.

By setting

```
GRANDIA_MODE = true
```

in the configuration you will see a bar that shows the portraits of the battlers moving as their next turns approach. This feature does not change the flow of the actions in the battle system, it is only a different way to show the turns.

When using this feature you can set the position of the bar and the picture used for it.

```
GRANDIA_BAR_WINDOW_X = 550
GRANDIA_BAR_WINDOW_Y = 80

# Put "" if unused
GRANDIA_BAR_BG_NAME = "grandia_bar_bg"
```

## Custom States.

These are all the custom states available in the battle system along with a short description.

REFLECT	reflect skills to the character who used them.
RIGENE	recover a little amount of HP.
EXPx2	get doubled experience at the end of battles.
REACTION	lets a character attacked with a physical attack react with a physical attack, a skill, or a selectable action.
PRIORITY	get the first turn in battle.
AUTO_PHOENIX	automatically use a reviving object on a dead character.
THEEVERY	steal rarities.
AUTO_LIFE	the character is revived when his HP reach zero.
STOP_TIME	perform actions without delay (get consecutive turns).
MP1	use only 1 MP for any skill.
HALF_MP	use half the MPs needed for skills.
DOOM	starts a countdown on the target. The target dies when the countdown reaches zero.
ZOMBIE	skills restore HPs and healing skills produce damage.
STONE	the character dies when attacked physically.
HEALER	doubles the effect of healing skills.
SUPER_GUARD	extra reduction of damage when guarding.
GOLDx2	get doubled gold at the end of battles.