Enabling the Damage Graphic

Soryu (@soryu_rpmakermv)*

1 Introduction

When damaged value for the battler is about to be displayed In RPG Maker MV, an image source as Fig. 1 is loaded and processed.

Figure 1: A source "Damage.png" located **./img/system/** in RPGMV

In RPGMZ, this style has been replaced with a simple text image generated by the game font. This change is supposed to be one of an optimization due to slowness of loading images in the RPGMaker engine (javascript). Games developed by using RPGMaker MV are infamous for its poor performance especially in terms of its large memory consumption during its game play. Therefore, RPGMZ engine is constructed to reduce it throughout the game by implementing a system to immediately discard images which are no longer used.

*http://dragonflare.blue/dcave/index_e.php

Loading image sources in RPGMaker is an asynchronized process that following procedures are conducted once the loading is started without waiting until it is finished. When the damaged values are about to be processed in RPGMV/MZ, the value images are prepared just before displaying them. In RPGMZ, the value images which finished its duration on the game screen are immediately deleted to reduce memory consumption. Hence, "Damage.png" must be loaded whenever the damaged value is to be displayed, which sometimes causes a problem that the value images are not ready to be displayed.

This plugin reappears the process of damaged values displaying on the game screen based on the system of RPG Maker MV. Though Fig. 2 is a game screen of RPGMZ, the damaged value is composed by loading a image "Damage.png" as RPGMV.

Figure 2: Damaged values drawing by a source "Damage.png"

During a battle scene, a damaged value source are not deleted every time so that a situation the image sources are not ready is eliminated.

In addition to the alternation of damage font sprites, behavior of value sprites are modified in Fig. 2. To do

this, additional script modification is required. In this plugin, the behavior while updating the position of damaged value sprites are extracted from main script to independent file to relieve the difficulty of scripting. For this scripting, an additional instruction is presented later.

Anyway, this plugin contributes to reuse of "Damage.png" materials in RPG Maker MZ. Many resources of damage sprites targeted on RPG Maker MV never become obsoleted legacies. Furthermore, we can avoid a quiet damage sprites pattern by using the game font in default.

2 Usage

This plugin consists of following two parts. If you still use the default damaged value behavior (animation), **just install the former**.

- SoR_AlternativeDamagePopup_MZ.js
- SoR_ADP_UpdateModifier.js

After installation of this plugin, a image source must locate **./img/system/Damage.png** as RPG Maker MV.

2.1 Plugin Parameters

Plugin parameters of this plugin (a base file `SoR_AlternativeDamagePopup_MZ.js`) are to configure the initial states of damaged sprites (such as initial position). For the animation while damage sprites is displayed, we have to write a script as shown later.

2.2 Fundamental Settings

- Damage_Duration
  - Duration while the sprite is displayed (Number of frames)
  - Total time between a moment the damage is started to display and disappeared
- Initial_DamagePositionY
  - Initial position of damage sprites for y-coordinate
  - Relative to the battler position
- CriticalFlash_Duration
  - Number of frames the damage sprite is flashed in critical attack
- CriticalFlash_Color
  - Sprite color in critical attack, 4 elements of the array ($[R, G, B, A]$)
- DamagePosX_Multiple
  - Padding of the damage sprite for x-coodinate for multiple damages
  - Assume a situation that a new damage is displayed before the previous one is still alive.
  - Position of every damage sprites are shifted by this value.
- DamagePosY_Multiple
  - Padding of the damage sprite for y-coodinate for multiple damages
  - Position of every damage sprites are shifted by this value.
- DamagePosX_Random
  - **Randomly** adjust for x-coodinate for multiple damages
  - The initial point is shifted randomly based on this value.
  - To scatter the value sprites, use this one (compatible with `DamagePosX_Multiple`).
- DamageFadeOut_Frame
  - Number of left frames when sprites begin to fade out
  - Current number of frames goes down from `Damage_Duration`.

2.3 Adjust the Base Position of Damaged Values

- Actor_damageOffsetX
- Actor_damageOffsetY
  - Padding for the damage relative to the actor position

- Enemy_damageOffsetX
- Enemy_damageOffsetY
  - Padding for the damage relative to the enemy position

3 Modification of the Animation for Damage Sprites

To modify the behavior of damage sprites during its duration on the game screen, it is the best to write a script to control them. When we want to modify it, we can use additional file `SoR_ADP_UpdateModifier.js`. There is just one function named `calcDamageSpritePosition` to modify the animation. This function is extracted so that it works independently. Then we just learn a minimum coding rule without understanding all of RPGMaker. However, **we must have fundamental mathematical and physics knowledge to draw natural and beautiful animation for the game**.

3.1 Looking at the Default Style in RPGMZ

A file `SoR_ADP_UpdateModifier.js` is just for writing the behavior of damage sprites in every frame as shown in Code 1. There is just one fuction `calcDamageSpritePosition`. Initially, the default behavior in RPGMZ is described.

Code 1: 'Default behavior'

```
1 Sprite_Damage.prototype.calcDamageSpritePosition
      = function(sprite,i,len) {
2 sprite.dy += 0.5;
3 sprite.ry += sprite.dy;
4 if (sprite.ry >= 0) {
5 sprite.ry = 0;
6 sprite.dy *= −0.6;
7 }
8 }
```

In default style of RPGMZ, the damage sprite is originally placed near the center of target battler. It then begins to fall toward the below of the game screen. After that, it bounds several times and settles on below the target. This behavior is expressed just 6 lines in the code.

To understand the description in this function, some variables which can be used are listed in Table 3.1.

Except for `this.battler` to specify the target bettler, they are all defined in the default RPGMZ engine.

Variables	Uses
sprite	Damage Sprite
sprite.rx	x-coordinate of the Sprite
sprite.ry	y-coordinate of the Sprite
sprite.dx	Derivation for x-direction
sprite.dy	Derivation for y-direction
sprite.scale.x	Scale of the Sprite for x
sprite.scale.y	Scale of the Sprite for y
this._duration	Current animation time (frames)
this.battler	Target battler of this damage sprite

Table 1: Variables should be used to describe the damage animation

We must pay attention to that a derivation (dx, dy) is used to determine the position of sprites (rx, ry). The function `calcDamageSpritePosition` is called every frame while the sprite is displayed on the battler (Number of frames designated by a plugin parameter `Damage_Duration`). Seriously explaining, The position (y) of the damage sprite is sentenced by Eq (1).

$$\begin{aligned} y &= \int_0^{duration} f(t)dt \\ &= \sum_{t=0}^{duration} dy \end{aligned} \tag{1}$$

This means that the code is not directly designating the position y but calculating based on a derivation $dy = f(t)$ which denotes how much the sprite moves in one frame. Accumulating dy in y, we can see the damage sprite moving on the screen smoothly. (The basic idea is "The velocity multiplied by duration becomes the total moving length" as elementary arithmetic.)

Never write as $y+ = 1$, or you have **a poor game that the sprite moves monotonically.**

Then, look at the Code 1 again. We now have $dy = 0.5t$ and it is accumulated in the value of y-coordinate. Hence, **the velocity of moving sprite is getting increased**. However, it will fall into the further below of the screen. Then we stop falling by using `if` **branch at Line 4** at a designated point. In that time, dy is multiplied by -0.6. The sprite temporally goes to up. But it soon goes down because we have $dy = 0.5t$ initially, and it oscillates for a while.

3.2 Moving on the Trail of Parabola

Here is an another example to create own damage animation. As shown in Fig. 3, we try to launch the sprite as **trailing on the parabola**.

Figure 3: Damage parabola animating as drawing the parabola

Code 2 is a script to show a behavior previously shown in Fig. 2. Damage sprites are launched with an oblique angle and they gradually fade. The horizontal direction is determined whether the target is an actor or enemy.

Code 2: 'Parabola Animation of Sprites'

```
1  Sprite_Damage.prototype.calcDamageSpritePosition
       = function(sprite,i,len) {
2  const t = 120−this._duration;
3
4  sprite.dx = this.battler.isActor()? 1 : −1;
5  sprite.dy = −3.0+0.063*t;
6  sprite.scale.x −= 0.001;
7  sprite.scale.y −= 0.001;
8  sprite.rx += sprite.dx;
9  sprite.ry += sprite.dy;
10 if(sprite.dy < 0) sprite.opacity−=1;
11 }
```

First, a fundamental knowledge of dynamics in physics is required to draw a parabola. This behavior shown in Fig. 3 is known as "Projectile motion".

$$v_x = v_{x0} \tag{2}$$

$$v_y = v_{y0} - gt \tag{3}$$

v_{x0}, v_{y0} are constants to note the inital velocity for x,y-direction. Also, $g = 9.81(m/s^2)$ is a constat for gravitation acceleration. The formula of Projectile motion is split into two equations for each direction. For horizontal, the sprite moves with constant velocity v_{x0}. For vertical, it is a free fall with initial velocity v_{y0}. Actually, we just substitute an arbitrary v_{x0}, v_{y0} into `sprite.dx` and `sprite.dy` in the script.

In Line 4, the velocity for x is set as $v_{x0} = 1$ with inverting the direction by the target (actor or enemy). In Line 5, $v_{y0} = -3.0$ is used. For g in the script, $g = 0.063$ is temporarily applied because actual physics movement is not appropriate to fit the object within the 2D game screen. Then we also have to adjust the constant experimentally.

Therefore, the derivation of y-coordinate is given by $dy = -3.0 + 0.063t$. Note that, -3.0 denotes to launch toward upper of the screen because the coordinate system for y-direction is inverted in the game screen.

Time t is a frame time given by Line 2 (Remember that `this._duration` is left alive time of the sprite). Also, `Damage_Duration` is set as 120 in this example. t gradually increased, the sprite first goes upper and turns to fall toward the lower of the screen because the sign of dy are inverted.

Additionally, `sprite.scale.x` and `sprite.scale.y` are modified to decrease gradually so that the sprite is flying away. In Line 10, the sprite will start to fade out explicitly.

In this way, dx and dy are given a constant velocity and that depends on time t. They are accumulated in (rx, ry) every frame. Then, we get an animation that the damage sprite flies away like drawing a parabola.

3.3 Summary for Damage Animation

The essence of scripting the damage animation is to determine the derivative (dx, dy) of moving damage sprites. Accumulating them in (rx, ry), we obtain a smooth animation. The summary of making script for the animation is following.

1. Get a current frame time as `const t = XXX -this._duration`
   - XXX is a value designated in the plugin parameter `Damage_Duration`
2. Determine `sprite.dx` and `sprite.dy` (Also use current time t)
   - Without t, the sprite moves monotonically (the same velocity).
   - Using t, the velocity is changed referring t.
3. Add `sprite.dx` and `sprite.dy` into `sprite.rx` and `sprite.ry`, respectively.
4. If we need to change the behavior, use conditional branch with `if` statements.

Consequently, if we have ideas to determine dx and dy, any kinds of animation for damage sprites are presented in RPGMaker.

4 Information for Possible Conflict to other plugins

This plugin overwrites most functions to handle damage drawing. Though it is hardly usual to use with other similar plugins, additional modification should be done if needed.

4.1 Overwritten

Alternate the damage style (just for drawing process)

- Overall of `Sprite_Damage` class
- Sprite_Battler.prototype.createDamageSprite

Avoid to destory the damage sprites every time.

- Sprite_Battler.prototype.destroyDamageSprite

4.2 Extension

Load and prepare damage sprite sources.

- `BattleManager.setup`
- `Sprite_Damage.prototype.initialize`

5 Version info.

- ver. 1.00 (Oct. 3rd, 2020) released!