

RPG Maker XP documentation

David Rodriguez Soares

August 3, 2020

Privacy policy

This is a confidential document and should not be distributed under any circumstance. [Please click and read.](#)

Contents

1	What this document is about	3
2	How RMXP stores data	4
3	Events	5
3.1	Basic functionalities	6
3.2	Advanced functionalities	6
4	Commands	7
4.1	Methodology	7
4.2	Miscellaneous information	7
4.3	List of commands	8
4.4	Complex commands	16
5	Command Representation decisions	19
5.1	Commands	20
5.2	Formal grammar	23
6	Maps	26
6.1	Associated classes	26
7	Remarks	28
7.1	Contact	28
7.2	Privacy Policy	28

1 What this document is about

This document holds information about how RPG Maker XP implements *Maps* and *Events*, which is relevant in project PoGER's map/feature extraction effort.

Please read this document's [Privacy Policy](#).

As a result of the limited scope of PoGER and the limited time and information available to the author, the following documentation isn't complete and may not be accurate.

The information was obtained through the official RPG Maker XP built-in documentation, user content found on the internet (forum posts, videos) and the author's reverse-engineering work.

The following abbreviations may be present :

- **RMXP** - RPG Maker XP
- **PE** - Pokemon Essentials

Please note that the author is not a native English speaker.

2 How RMXP stores data

A crucial first step in any reverse-engineering effort in data extraction is to understand used data structures.

As RMXP games run on a *Ruby interpreter*, every element we encounter is either of a *primitive type* or an *object* (class instance).

Ruby primitive types :

- Arrays
- Hashes
- Boolean
- Symbols
- Numbers
- Strings

For the task at end, let's focus on the classes that are associated with maps and events, most of which are part of the RMXP library (other are defined in PE scripts). See *RMXP_full.png*

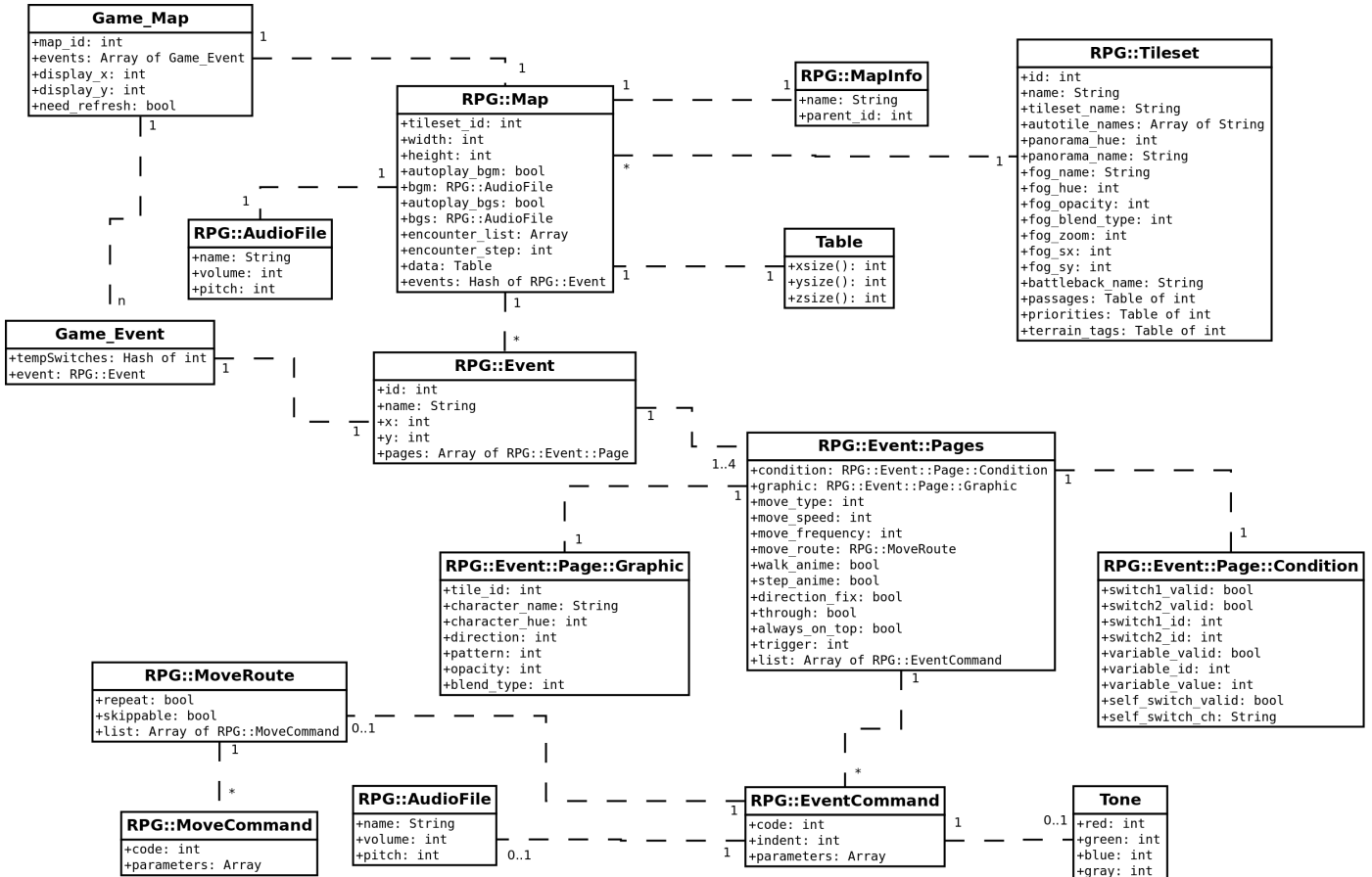


Figure 1: Simplified class map representation for Map/Event

Semantic/Syntax : Linked classes (with arity) display an **associative relationship**.

Note : There is no inheritance relationship between any two classes represented. Arities are logically deduced and may not be exact depending in proprietary implementation details. Class `RPG::AudioFile` was duplicated for ease of association routing.

3 Events

An event, or more precisely a *map event*, is a way to introduce elements with behavior, therefore bringing flexibility and dynamism into the game world.

Events have two aspects :

- A GUI element

Edit Event - ID:004

Name:

1 2

Conditions


☐ Switch is ON

☐ Switch is ON

☐ Variable is or above

☐ Self Switch is ON

Graphic:



Autonomous Movement

Type:

Speed:

Freq:

Options

☒ Move Animation

☐ Stop Animation

☐ Direction Fix

☐ Through

☐ Always on Top

Trigger

☐ Action Button

☐ Player Touch

☒ Event Touch

☐ Autorun

☐ Parallel Process

List of Event Commands:

```
@>Comment: Battle: \bHi! I like shorts! They're comfy and easy to wear!
@>Comment: Type: YOUNGSTER
@>Comment: Name: Ben
@>Comment: EndSpeech: Aww, I lost.
@>Comment: EndBattle: \bYou can't get a trainer event simpler than me!
@>Script: pbTrainerIntro(:YOUNGSTER)
@>Script: Kernel.pbNoticePlayer(get_character(0))
@>Text: \bHi! I like shorts! They're comfy and easy to wear!
@>Conditional Branch: Script: pbTrainerBattle(:YOUNGSTER,"Ben",_I("Aww, I lost."))
@>Control Self Switch: A =ON
@>Branch End
@>Script: pbTrainerEnd
@>
```

- Its data class instance counterpart `RPG::Event`

```

Map031_events > {} Map031_eventTrainer(2)_json > {} pages > {} 0
1 {
2   "class": "Event",
3   "name": "Trainer(2)",
4   "x": 5,
5   "y": 14,
6   "pages": [
7     {
8       "class": "Page",
9       "condition": {
10        "class": "Page::Condition",
11        "switch1": null,
12        "switch2": null,
13        "self_switch": null,
14        "variable": null,
15        "variable_value": null
16      },
17      "graphic": {
18        "class": "Page::Graphic",
19        "tile_id": 0,
20        "character_name": "trchar037",
21        "character_hue": 0,
22        "direction": "Down",
23        "pattern": 0,
24        "opacity": 255,
25        "blend_type": "Normal"
26      },
27      "move": "Fixed",
28      "move_speed": 3,
29      "move_frequency": 3,
30      "walk_anime": true,
31      "step_anime": false,
32      "direction_fix": false,
33      "through": false,
34      "always_on_top": false,
35      "trigger": "onEventTouch",
36      "list": [
37        {
38          "class": "EventCommand",
39          "code": 108,
40          "indent": 0,
41          "parameters": "Battle: \\bHi! I like shorts! They're
42        },
43        {
44          "class": "EventCommand",
45          "code": 408,
46          "indent": 0,
47          "parameters": "wear!"
48      ],
49    },
50    {
51      "class": "EventCommand",
52      "code": 108,
53      "indent": 0,
54      "parameters": "Type: YOUNGSTER"
55    },
56    {
57      "class": "EventCommand",
58      "code": 108,
59      "indent": 0,
60      "parameters": "Name: Ben"
61    },
62    {
63      "class": "EventCommand",
64      "code": 108,
65      "indent": 0,
66      "parameters": "EndSpeech: Aww, I lost."
67    },
68    {
69      "class": "EventCommand",
70      "code": 108,
71      "indent": 0,
72      "parameters": "EndBattle: \\bYou can't get a trainer e
73    },
74    {
75      "class": "EventCommand",
76      "code": 408,
77      "indent": 0,
78      "parameters": "than me!"
79    },
80    {
81      "class": "EventCommand",
82      "code": 355,
83      "indent": 0,
84      "parameters": "pbTrainerIntro(:YOUNGSTER)"
85    },
86    {
87      "class": "EventCommand",
88      "code": 355,
89      "indent": 0,
90      "parameters": "Kernel.pbNoticePlayer(get_character(0))
91    },
92    {
93      "class": "EventCommand",
94      "code": 101,
95      "indent": 0,
96      "parameters": "\\bHi! I like shorts! They're comfy a

```

3.1 Basic functionalities

These are the easiest and most straightforward behavior to implement into an event :

- Giving an element a *sprite* (texture) : This is useful for objects capable of movement, NPCs, etc.
- *Movement* : Select how the element moves with presets (speed, frequency, pattern, etc).
- *Event commands* : Select the trigger for behavior and what the element does when triggered (movement, dialogue, etc) within the extensive command list.

3.2 Advanced functionalities

These require an understanding of conditional execution and scripting :

- *Conditional execution* : branching instructions based on the value of : global variables, global switches, self switches, script return, etc.
- *Pages* : Allow to give an element different behavior depending on conditions.
- *Move routes* : Define a sequence of movement commands to be executed.
- *Script calls* : Call a script to be executed for more complex behavior, launching mini-games, retrieving data, etc.

4 Commands

Commands are a mechanism, through which most of an `RPG::Event`'s behavior is defined.

Although they are very similar in structure and use, a distinction is made between `RPG::EventCommand` and `RPG::MoveCommand`.

EventCommands are the representation of elements present in the "List of Event Commands" in the GUI. They are the building block of event's behavior.

MoveCommands are the representation of an individual movement the event is capable of, typically found in sequences `RPG::MoveRoute` associated with a dedicated *EventCommand*.

They both have, at least :

- A *code* : An integer that uniquely identifies the particular command.
- *Parameters* : Depend on the particular command, can be empty, a variable, an object, or a list of objects.

Additionally, *EventCommands* have an *indent* integer value, tied to the layout visible in the "List of Event Commands" in the GUI.

4.1 Methodology

In order to successfully *extract semantic from events*, it was decided that *documenting* every command used in Pokemon Essentials and finding an appropriate (human-readable) *representation* was the way forward.

The objective is to formalize a **DSL** (Domain Specific Language) into which events will be translated to, which exhibit desirable properties ([See Command Representation decisions section](#)).

4.2 Miscellaneous information

Codes used in Pokemon Essentials 17.2 (*81 total*) :

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 33, 34, 37, 38, 39, 40, 41, 42, 44, 101, 102, 104, 106, 108, 111, 112, 113, 115, 118, 119, 121, 122, 123, 125, 201, 202, 208, 209, 210, 221, 222, 223, 225, 231, 232, 235, 236, 241, 242, 247, 248, 249, 250, 314, 354, 355, 401, 402, 404, 408, 411, 412, 413, 655

Implementation details :

- `RPG::MoveCommand` use range [1-45]
- `RPG::EventCommand` use range [101- x], $x \geq 655$
- A "frame" is defined as $\frac{1}{20}$ second \Rightarrow change into milliseconds $m = n * 1000 / 20 \equiv n * 50$.
- Every event has an ID (integer > 0). Actions that can affect other events can target the player using id -1 and the current event using id 0.
- Special variables : MapID, PartyMembers, Gold, Steps, PlayTime, *Timer*, SaveCount.

They should all be read accessible. Underlined ones should also be write accessible. *Italic ones are probably not used*.

4.3 List of commands

0	Description	Nothing, empty command or end of the event command list
	Parameters	None
	Notes	Will not be represented
	Representation	None
1	Description	<code>RPG::MoveCommand</code> - Move to the South
	Parameters	None
	Notes	See footnote ¹
	Representation	"Move, S"
2	Description	<code>RPG::MoveCommand</code> - Move to the West
	Parameters	None
	Notes	See footnote ¹
	Representation	"Move, W"
3	Description	<code>RPG::MoveCommand</code> - Move to the East
	Parameters	None
	Notes	See footnote ¹
	Representation	"Move, E"
4	Description	<code>RPG::MoveCommand</code> - Move to the North
	Parameters	None
	Notes	See footnote ¹
	Representation	"Move, N"
5	Description	<code>RPG::MoveCommand</code> - Move to the SouthWest
	Parameters	None
	Notes	See footnote ¹
	Representation	"Move, SW"
6	Description	<code>RPG::MoveCommand</code> - Move to the SouthEast
	Parameters	None
	Notes	See footnote ¹
	Representation	"Move, SE"
7	Description	<code>RPG::MoveCommand</code> - Move to the NorthWest
	Parameters	None
	Notes	See footnote ¹
	Representation	"Move, NW"
8	Description	<code>RPG::MoveCommand</code> - Move to the NorthEast
	Parameters	None
	Notes	See footnote ¹
	Representation	"Move, NE"
9	Description	<code>RPG::MoveCommand</code> - Move at random (N,E,S,W)
	Parameters	None
	Notes	See footnote ¹
	Representation	"Move, R"

10	Description Parameters Notes Representation	RPG::MoveCommand - Move towards player None See footnotes ^{1,3} "Move, TODO"
11	Description Parameters Notes Representation	RPG::MoveCommand - Move away from player None See footnotes ^{1,3} "Move, TODO"
12	Description Parameters Notes Representation	RPG::MoveCommand - Take 1 step forward None See footnote ¹ "Move, TODO"
13	Description Parameters Notes Representation	RPG::MoveCommand - Take 1 step backward None See footnote ¹ "Move, TODO"
14	Description Parameters Notes Representation	RPG::MoveCommand - Jump to relative coordinates on the same map [2] - 0:deltaX [signed integer], 1:deltaY [signed integer] None "Jump, TODO"
15	Description Parameters Notes Representation	RPG::MoveCommand - Wait n seconds [1] - 0:number of seconds to wait n [integer $\in \mathbb{N}^*$] Typically $n == 2$, but values up to 15 were found in PE. "Wait seconds, n "
16	Description Parameters Notes Representation	RPG::MoveCommand - Turn towards South None See footnote ² "Turn, S"
17	Description Parameters Notes Representation	RPG::MoveCommand - Turn towards West None See footnote ² "Turn, W"
18	Description Parameters Notes Representation	RPG::MoveCommand - Turn towards East None See footnote ² "Turn, E"
19	Description Parameters Notes Representation	RPG::MoveCommand - Turn towards North None See footnote ² "Turn, N"
20	Description Parameters Notes Representation	RPG::MoveCommand - Turn 90° right, relative to current position None See footnote ² "Turn, R"

21	Description Parameters Notes Representation	RPG::MoveCommand - Turn 90° left, relative to current position None See footnote ² "Turn, L"
22	Description Parameters Notes Representation	RPG::MoveCommand - Turn 180° None See footnote ² "Turn, 180"
23	Description Parameters Notes Representation	RPG::MoveCommand - Turn 90° to the left or right, at random None See footnote ² "Turn, 90random"
24	Description Parameters Notes Representation	RPG::MoveCommand - Turn at random (90° or 180°) None See footnote ² "Turn, random"
25	Description Parameters Notes Representation	RPG::MoveCommand - Turn towards player None See footnotes ^{2,3} "Turn, TODO"
26	Description Parameters Notes Representation	RPG::MoveCommand - Turn away from player None See footnotes ^{2,3} "Turn, TODO"
33	Description Parameters Notes Representation	RPG::MoveCommand - Turn ON walking animation None "Animation, ON"
34	Description Parameters Notes Representation	RPG::MoveCommand - Turn OFF walking animation None "Animation, OFF"
37	Description Parameters Notes Representation	RPG::MoveCommand - Turn ON "through" None Equivalent to activating "walk through walls", making it possible to walk through impassable tiles/characters. "WTW, ON"
38	Description Parameters Notes Representation	RPG::MoveCommand - Turn OFF "through" None Equivalent to deactivating "walk through walls". "WTW, OFF"
39	Description Parameters Notes Representation	RPG::MoveCommand - Always on top ON None Elevate the display priority, therefore bringing the event graphic to the forefront (above any tile/character) "AOT, ON"

40	Description	RPG::MoveCommand - Always on top OFF
	Parameters	None
	Notes	
	Representation	"AOT, OFF"
41	Description	RPG::MoveCommand - Change event's graphic
	Parameters	TODO
	Notes	
	Representation	"TODO"
42	Description	RPG::MoveCommand - Change event's graphic opacity
	Parameters	[1] - 0:new opacity value n [integer 0-255]
	Notes	
	Representation	"Opacity, n "
44	Description	RPG::MoveCommand - Play a sound effect
	Parameters	TODO
	Notes	
	Representation	"Play SE, TODO"
101	Description	RPG::EventCommand - Show text
	Parameters	[1] - 0:text s [String]
	Notes	s must be properly double-quoted and formatted (inner double-quotes and backslashes must be escaped).
	Representation	"Show Text, s "
401	Description	RPG::EventCommand - Show text (continued)
	Parameters	[1] - 0:text s [String]
	Notes	Continuation of 101.
	Representation	See footnote ⁴
102	Description	RPG::EventCommand - Show choices
	Parameters	[2] - 0:array of size n [Array of Strings], 1:cancel behaviour [integer 0-4]
	Notes	Displays up to 4 selectable options in a message window. Cancel behaviour : 0 disallow canceling, $1-4 \leq n$ selects choice by default.
	Representation	"Choose, {0}, default={1}"
104	Description	RPG::EventCommand - Change text options
	Parameters	[2] - 0:position p [integer 0-2], 1>window border b [integer 0-1]
	Notes	Sets message window position and border. p follows "common relation 1", b follows "common relation 2"
	Representation	"Change text options, position={0}.toString(), border={1}.toString()"
106	Description	RPG::EventCommand - Wait
	Parameters	[1] - 0:number of frames to wait n [integer $\in \mathbb{N}^*$]
	Notes	Conversion to milliseconds chosen for its more precise and general use : $m = n * 1000/20 \equiv n * 50$, TODO:research its use
	Representation	"Wait ms, m "
108	Description	RPG::EventCommand - Comment
	Parameters	[1] - 0:comment text s [String]
	Notes	Has no effect. TODO:research link to particle effects.
	Representation	"# s "

408	Description	RPG::EventCommand - Comment (continued)
	Parameters	[1] - 0:comment text <i>s</i> [String]
	Notes	Happens after a 108.
	Representation	"# <i>s</i> "
111	Description	RPG::EventCommand - Conditional branch
	Parameters	See " Conditional branch " section.
	Notes	Complex but essential command.
	Representation	"If, {condition}"
112	Description	RPG::EventCommand - Loop
	Parameters	None
	Notes	Loops over commands until broken. TODO:research usage
	Representation	"Loop"
113	Description	RPG::EventCommand - Break loop
	Parameters	None
	Notes	Escape innermost loop. TODO:research usage
	Representation	"Break"
115	Description	RPG::EventCommand - Exit Event Processing
	Parameters	None
	Notes	TODO:research usage
	Representation	TODO
118	Description	RPG::EventCommand - Label
	Parameters	[1] - 0:label name <i>s</i> [String]
	Notes	Sets a label to allow jumping to.
	Representation	"Label, <i>s</i> "
119	Description	RPG::EventCommand - Jump to Label
	Parameters	[1] - 0:label name <i>s</i> [String]
	Notes	Jumps to a label.
	Representation	"Goto, <i>s</i> "
121	Description	RPG::EventCommand - Control switches
	Parameters	[3] - 0:starting switch <i>ssa</i> [integer], 0:starting switch <i>ssz</i> [integer], 0:new state <i>n</i> [integer]
	Notes	Batch control is unused in PE, therefore deprecated. <i>n</i> follows "common relation 3".
	Representation	"Control Switch, <i>ssa.toString()</i> , <i>n.toString()</i> "
122	Description	RPG::EventCommand - Control variables
	Parameters	See " Control variables " section.
	Notes	Batch control is unused in PE, therefore deprecated.
	Representation	"Control Variable, TODO"
123	Description	RPG::EventCommand - Control Self Switch
	Parameters	[2] - 0:SS character <i>s</i> [String of length 1], 1:new state <i>n</i> [integer 0-1]
	Notes	<i>n</i> follows "common relation 3".
	Representation	"Control SS, <i>s</i> , <i>n.toString()</i> "

125	Description	RPG::EventCommand - Change Gold
	Parameters	[3] - 0:operation <i>o</i> [integer 0-1], 1:operand <i>n</i> [integer 0-1], 2:value <i>v</i> [integer]
	Notes	Values of <i>n</i> : 0: <i>v</i> is a constant, 1: <i>v</i> is a variable(id). <i>o</i> follows "common relation 4"
	Representation	"Control Variable, :Money <i>o</i> .toString() <i>v</i> .toString()"
201	Description	RPG::EventCommand - Transfer Player
	Parameters	[6] - 1:map <i>m</i> [integer], 2:coordinate <i>x</i> [integer], 3:coordinate <i>y</i> [integer], 4:player direction <i>d</i> [integer], 5:fading <i>f</i> [integer].
	Notes	{0} must be 0, 1 unused in PE. <i>d</i> follows "common relation 5". <i>f</i> follows "common relation 3".
	Representation	"Transfer Player, destination=(<i>m</i> , <i>x</i> , <i>y</i>), direction= <i>d</i> .toString(), fading= <i>f</i> .toString()"
202	Description	RPG::EventCommand - Set Event Location
	Parameters	[5] - 0:event id <i>e</i> [integer], 2:coordinate <i>x</i> [integer], 3:coordinate <i>y</i> [integer], 4:direction <i>d</i> [integer]
	Notes	Change an event's location on the current map. {1} must be 0, other values unused in PE. <i>d</i> follows "common relation 5".
	Representation	"Move Event, <i>e</i> .toString(), (<i>x</i> , <i>y</i>), direction= <i>d</i> .toString()"
208	Description	RPG::EventCommand - Change Transparency Flag
	Parameters	[2] - 0:flag <i>d</i> [integer 0-1]
	Notes	When transparency is set, the graphic isn't displayed. <i>d</i> follows "common relation 3".
	Representation	"Set Transparency, <i>d</i> .toString()"
209	Description	RPG::EventCommand - Set Move Route
	Parameters	[2] - 0:target id <i>d</i> [integer], 1:RPG::MoveRoute
	Notes	
	Representation	"Set Move Route, <i>d</i> .toString()"
210	Description	RPG::EventCommand - Wait for Move's Completion
	Parameters	None
	Notes	To be put after a Set Move Route. Without it, further commands can be executed before the end of the walking animation.
	Representation	"Wait for move route completion"
221	Description	RPG::EventCommand - Prepare for transition
	Parameters	None
	Notes	Freezes the screen, so there's nothing moving during the transition. To be fused with Execute Transition.
	Representation	
222	Description	RPG::EventCommand - Execute Transition
	Parameters	[1] - 0:transition file name <i>s</i> [String]
	Notes	Plays the animation. TODO:research how transition work.
	Representation	"Transition, <i>s</i> , freeze={True/False}"
223	Description	RPG::EventCommand - Change Screen Color Tone
	Parameters	[2] - 0:RPG::Tone, 1:duration(frames) <i>d</i> [integer]
	Notes	Typically used in fade out (to black/white)/fade in cycles. <i>d</i> to be changed into ms.
	Representation	"Change Screen Color Tone, <i>d</i> , {0}.toString()", "Fadein, <i>d</i> ", "Fadeout, <i>d</i> , {color}"
225	Description	RPG::EventCommand - Screen Shake
	Parameters	[3] - 0:shake power [integer], 1:shake speed [integer], 2:duration(frames) <i>d</i> [integer]
	Notes	Scarcely used in PE, {0} and {1} are not well defined so they can be deprecated. <i>d</i> to be changed into ms.
	Representation	"Shake Screen, <i>d</i> "

231	Description Parameters Notes Representation	RPG::EventCommand - Show Picture See " Show Picture " section.
232	Description Parameters Notes Representation	RPG::EventCommand - Move Picture See " Move Picture " section.
235	Description Parameters Notes Representation	RPG::EventCommand - Erase Picture [1] - 0:picture id [integer] TODO
236	Description Parameters Notes Representation	RPG::EventCommand - Set Weather effect [3] - 0:weather id [integer], 1:power [integer], 2:duration (frames) [integer] <i>power</i> to be removed. TODO:research how weather is generated. TODO
241	Description Parameters Notes Representation	RPG::EventCommand - Play BGM [1] - 0:audio <i>a</i> [AudioFile] "Play BGM, <i>a.toString()</i> "
242	Description Parameters Notes Representation	RPG::EventCommand - Fade Out BGM [1] - 0:duration (seconds) <i>n</i> [integer] "Fade Out BGM, <i>n</i> "
247	Description Parameters Notes Representation	RPG::EventCommand - Memorize BGM/BGS None "Memorize BGM/BGS"
248	Description Parameters Notes Representation	RPG::EventCommand - Restore BGM/BGS None "Restore BGM/BGS"
249	Description Parameters Notes Representation	RPG::EventCommand - Play ME [1] - 0:audio <i>a</i> [AudioFile] "Play ME, <i>a.toString()</i> "
250	Description Parameters Notes Representation	RPG::EventCommand - Play SE [1] - 0:audio <i>a</i> [AudioFile] "Play SE, <i>a.toString()</i> "

314	Description	<code>RPG::EventCommand</code> - Restore All
	Parameters	[1] - <code>0</code> :actor id [<code>integer</code>]
	Notes	Equivalent to healing and restoring PPs. Ignore parameter.
	Representation	"Restore All"
354	Description	<code>RPG::EventCommand</code> - Return to Title Screen
	Parameters	None
	Notes	
	Representation	"Return to Title Screen"
355	Description	<code>RPG::EventCommand</code> - Script
	Parameters	[1] - <code>0</code> :script string [<code>String</code>]
	Notes	To be overhauled.
	Representation	TODO
655	Description	<code>RPG::EventCommand</code> - Script (continued)
	Parameters	[1] - <code>0</code> :script string [<code>String</code>]
	Notes	To be overhauled.
	Representation	TODO
402	Description	<code>RPG::EventCommand</code> - When
	Parameters	[1] - <code>0</code> :choice id [<code>integer</code>], <code>1</code> :choice string equivalent [<code>integer</code>]
	Notes	Used with choices and conditional branches, has code block per choice.
	Representation	TODO
404	Description	<code>RPG::EventCommand</code> - End of When
	Parameters	None
	Notes	
	Representation	None
411	Description	<code>RPG::EventCommand</code> - Else
	Parameters	None
	Notes	Used with conditional branch 111.
	Representation	TODO
412	Description	<code>RPG::EventCommand</code> - Branch End
	Parameters	None
	Notes	End of a code block (as result of branching). TODO:investigate whether it is present in every code block and if it should be represented (is indentation sufficient?).
	Representation	TODO
411	Description	<code>RPG::EventCommand</code> - Repeat above
	Parameters	None
	Notes	Marks end of Loop 112 code block.
	Representation	TODO

Current representation has ≈ 39 instructions ($> 50\%$ reduction !).

¹Movements consolidated with new *Move* command with argument.

²Turs consolidated with new *Turn* command with argument.

³Unknown algorithm to determine direction "towards player" and "away from player."

⁴Is part of a command sequence that should be merged in a sensible way.

4.3.1 Common relations

In parenthesis are the proposed representation or information :

1. 0:Top, 1:Middle, 2:Bottom
2. 0:Show, 1:Hide
3. 0:ON, 1:OFF
4. 0:Increase, 1:Decrease (+, -)
5. 0:Keep same, 2:Down, 4:Left, 6:Right, 8:Up (K,S,W,E,N)
6. 0:'==', 1:'>=', 2:'<=', 3:'>', 4:'<', 5:'!='
7. 0:constant, 1:variable
8. 0:'>=', 1:'<='
9. 0:'=', 1:'+=', 2:'-=', 3:'*=', 4:'/', 5:'%' (affectation, increment, decrement, multiplication, division, modulo)
10. 0:coordinate X, 1:coordinate Y, 2:direction (3-5 unused)
11. 0:NW, 1:Centered (picture coordinate origin)
12. 0:Normal, 1:Additive, 2:Subtractive (blending type)

TODO:determine if division is always rounded to an integer (and how) or not.

4.4 Complex commands

Some commands have complex behaviour that doesn't fit in the table above, therefore detailed explanation were put here instead.

4.4.1 Conditional branch - 111

This command is RMXP's equivalent of an 'if' instruction, and therefore hinges on expressing a condition. Given the expansive list of conditions that can be expressed, its syntax is quite complex.

The *first parameter* is crucial : it defines the type of condition. **integer** 0-12 :

- 0 Check *Switch* state.

Parameters	[3] - 1:switch id <i>n</i> [integer], 2:switch state <i>d</i> [integer 0-1]
Notes	<i>d</i> follows "common relation 3".
Representation	"If, <i>n.toString()</i> , <i>d.toString()</i> "

- 1 Check *Variable* value.

Parameters	[5] - 1:variable id <i>n</i> [integer], 2:what it is compared to <i>m</i> [integer] 3:constant or variable id <i>x</i> [integer], 4:comparator <i>c</i> [integer]
Notes	<i>c</i> follows "common relation 6", <i>m</i> follows "common relation 7".
Representation	<i>m</i> =='constant' : "If, <i>n.toString()</i> , <i>c.toString()</i> , <i>x</i> " <i>m</i> =='variable' : "If, <i>n.toString()</i> , <i>c.toString()</i> , <i>x.toString()</i> "

- 2 Check *Self-Switch* state.

Parameters	[3] - 1 :self switch character n [String of size 1], 2 :switch state d [integer 0-1]
Notes	d follows "common relation 3".
Representation	"If, n , $d.toString()$ "

6 Check *Event* direction.

Parameters	[3] - 1 :event id n [integer], 2 :direction d [integer 0-1]
Notes	d follows "common relation 5".
Representation	"If, $n.toString()$, Facing, $d.toString()$ "

7 Check *Player's money*.

Parameters	[3] - 1 :amount n [integer], 2 :comparator d [integer 0-1]
Notes	d follows "common relation 8".
Representation	"If, Money, $d.toString()$, n "

12 Check *Script's return*.

Parameters	[2] - 1 :Script s [String]
Notes	Script must return a boolean (prehaps returning nothing is OK?)
Representation	"If, Script, s "

Values 3,4,5,8,9,10,11 were not found in PE, therefore not researched.

4.4.2 Control variables - 122

Parameters **0** and **1** [integer] are indexes for the range of variables that will be affected. Variable is s

As batch control of variables is unused in PE, it is deprecated in the representation (parameter **1** is ignored).

Parameter **2** o [integer 0-5] sets the **operation** to be performed on the variable, and follows "common relation 9".

Parameter **3** defines the **operand type** [integer 0-7] :

0 Constant.

Parameters	[5] - 4 :constant n [integer]
Notes	
Representation	"Control, $s.toString()$, $o.toString()$, n "

2 Random integer.

Parameters	[6] - 4 :constant a [integer], 5 :constant z [integer]
Notes	Will choose a number $x \in [a, z]$. TODO:check if a and z are included.
Representation	"Control, $s.toString()$, $o.toString()$, $[a, z]$ "

6 Event's attribute.

Parameters	[6] - 4 :event id n [integer], 5 :attribute id d [integer 0-2]
Notes	d follows "common relation 10".
Representation	"Control, $s.toString()$, $o.toString()$, Event, $n.toString()$, $d.toString()$ "

7 Only used once, to put the "Money"/"Gold" special variable in a temporary variable to be used in a condition, therefore isn't really needed.

Values 1,3,4,5 were not found in PE, therefore not researched.

4.4.3 Show Picture - 231

This command is only used in the intro.

Description	Display a picture.
Parameters	[10] - 0 :picture priority number p [integer], 1 :picture name s [String], 2 :coordinate origin c [integer 0-1], 3 :unused, 4 :relative position x [integer], 5 :relative position y [integer], 6 :horizontal zoom zx [integer], 7 :vertical zoom yz [integer] 8 :opacity o [integer 0-255], 9 :blending type b [integer 0-2]
Notes	c follows "common relation 11", b follows "common relation 12".
Representation	"Show Picture, s , priority= p , coordinates=($c.toString()$, x , y), zoom=(zx , yz), opacity= o , blending= $b.toString()$ "

Picture priority number p is used when multiple pictures are on display, because overlapping textures need to have an unambiguous drawing order.

Here, let there be pictures p_1, p_2 with priorities 2,4 respectively. Therefore, p_1 is drawn first, then p_2 . The result is that, if they are overlapping, p_2 will be drawn **over** p_1 , removing parts of p_1 from being displayed.

Typically, $x = y = 0$

4.4.4 Move Picture - 232

Parameters are mostly identical to "Show Picture". This is mostly used to animate intro's pictures (movement and opacity).

Description	Move a picture.
Parameters	[10] - 0 :picture priority number p [integer], 1 :duration in frames f [integer], 2 :coordinate origin c [integer 0-1], 3 :unused, 4 :relative position x [integer], 5 :relative position y [integer], 6 :horizontal zoom zx [integer], 7 :vertical zoom yz [integer] 8 :opacity o [integer 0-255], 9 :blending type b [integer 0-2]
Notes	c follows "common relation 11", b follows "common relation 12".
Representation	"Move Picture, priority= p , coordinates=($c.toString()$, x , y), zoom=(zx , yz), opacity= o , blending= $b.toString()$ "

5 Command Representation decisions

The representation chosen is a result of careful consideration of its future usage requirements (including but not limited to) :

- Readability : It is destined to be read and written by humans, therefore it should be as straightforward and non-cryptic as possible.
- Brevity : In the interest of anyone (human or software) reading/writing it, the *less is more* approach is to be applied : instructions should not be longer than what is necessary.
- Unambiguity : As any formal language, its use and syntax should be unambiguous.
- Simplicity : Limiting the amount of available instructions by combining related ones is good practice.
- Expandability : There should be room left for additional behavior to be implemented.

At the time of writing these lines, representations in this document are still not final, it's a work-in-progress.

Particular decisions :

- Python syntax style : Reduces explicit syntax (like semicolons and curly braces), therefore reducing syntax errors.
- Case insensitive : Simplification allowing any program to simply make everything lowercase when reading an *Event*, and users to Use any casing style they prefer. This also makes it harder to have variable/switch name collisions by forcing users to explicitly name their variables.
- Switches, Self-switches and variables : Should be all represented as **symbols**.

Proposed representation : `":s"` [**String**] (*string beginning with colon*)

Let **s** be the string representation (name) of the Switch/Self-switch/Variable. **s** of length 1 is to be reserved to self-switches.

Note that merging variables and switches may allow greater flexibility for users.

- Symbol length limit : TODO

Ideas

- Timers could be implemented as integers : Let `:PlayTime` be a read-only integer variable that counts the seconds of play time (an *epoch* of sorts).

Then, setting a timer for x seconds could be as simple as storing $(:PlayTime + x)$ in a variable and testing it later against the current value of `:PlayTime` !

- Commands have parameters (see examples) :
 - No parameter : command line must contain the command keyword only.
 - 1 parameter : command line must contain the command keyword, plus the expected parameter value (no parameter name; parameter may be facultative)
 - $n > 1$ parameters : command line must contain the command keyword, and the expected parameters `parameter_name = parameter_value` as a comma-separated list (no brackets; parameter name mandatory; parameter may be facultative).
 - Note on *facultative* parameters : marked with a `*`.
 - Strictly equivalent : `:ON \equiv True`, `:OFF \equiv False`

5.1 Commands

Description	Step - Move the event (perform 1 step).
Parameters	[1] - 0: direction - [String]
Notes	direction $\in \{S,W,E,N,SW,NW,NE,SE,R,1F,1B,1A,1T\}$, see "Directions" below.
Examples	"Step NW", "Step 1T"
Description	Turn - Turn the event (change direction).
Parameters	[1] - 0: direction - [String]
Notes	direction $\in \{S,W,E,N\}$, see "Directions" below.
Examples	"Turn N", "Turn W"
Description	Move Event - Move Event to absolute/relative coordinates on the same map.
Parameters	[2] - 0: event* - [String] (name of the event to move) 1: relative_coordinates/absolute_coordinates - [list of 2 int]
Notes	event is optional, defaults to self.
Examples	"Move Event relative_coordinates=[7,-5]", "Move Event event=Jack, absolute_coordinates=[4,12]"
Description	Wait - Pause event behavior execution for a given amount of time.
Parameters	[1] - 0: ms - [int] (time in milliseconds)
Examples	"Wait ms=3000", "Wait ms=20"
Description	Switch - Switch event properties on/off
Parameters	[2] - 0: property - [String], 1: value - [int/String/:ON/OFF]
Notes	property must be a configuration variable, see "Configuration variables" below. value must be a valid value for that property.
Examples	"Switch property=move_animation value=:ON" "Switch property=Animation value=:OFF" "Switch property=graphic value="trchar28"
Description	Play - Play audio.
Parameters	[3] - 0: SE/BGM/ME - [String], 1: volume* - [int], 2: pitch* - [int]
Notes	volume and pitch default to 100, their values are relative to 100 (percentage).
Example	"Play BGM="022-Field05", volume=100, pitch=100"
Description	Show Text
Parameters	[1] - 0: text - [String]
Example	"Show Text "Hello, World !"
Description	Choose - Give player a list of items to choose from.
Parameters	[2] - 0: choices - [list of String], 1: default* - [int] n (behavior on cancel)
Notes	If default not set, the player must choose (no cancel). Otherwise, select n^{th} item on the list.
Examples	"Choose choices=["Yes","No"]", "Choose choices=["One","Two","Three"], default=1"
Description	Change Text Options
Parameters	[2] - 0: position* - [Top/Middle/Bottom], 1: border* - [Show/Hide] (window border))
Example	"Change Text Options position=Middle, border=Show"
Description	End Execution - Ends behavior execution.
Parameters	[0]
Example	"End Execution"

Description	Label - Marks a line as a target for a Goto .
Parameters	[1] - 0: name - [String]
Notes	Please find a good name for the label (not like the example).
Example	" Label " here "
Description	Goto - Change line to be executed next.
Parameters	[1] - 0: label - [String]
Example	" Goto " here "
Description	Transfer Player - Teleport player.
Parameters	[1] - 0: map* - [String], 1: x - [int], 2: y - [int] 3: direction* - [String], 4: fading* - [:ON/:OFF]
Notes	map defaults to the one the player is in. direction $\in \{S,W,E,N,K\}$, defaults to " K ". map defaults to (TODO).
Example	" Transfer Player map="Kurt's house", x=2, y=4"
Description	Set Move Route
Parameters	TODO
Notes	TODO.
Example	TODO
Description	Shake Screen
Parameters	TODO
Notes	TODO.
Example	TODO
Description	Transition
Parameters	TODO
Notes	TODO.
Example	TODO
Description	Show Picture
Parameters	TODO
Notes	TODO.
Example	TODO
Description	Move Picture
Parameters	TODO
Notes	TODO.
Example	TODO
Description	Erase Picture
Parameters	TODO
Notes	TODO.
Example	TODO
Description	Set weather
Parameters	TODO
Notes	TODO.
Example	TODO

Description	Fade out BGM
Parameters	TODO
Notes	TODO.
Example	TODO
Description	Memorize BGx
Parameters	TODO
Notes	TODO.
Example	TODO
Description	Restore BGx
Parameters	TODO
Notes	TODO.
Example	TODO
Description	Restore All
Parameters	TODO
Notes	TODO.
Example	TODO
Description	Return to title screen
Parameters	TODO
Notes	TODO.
Example	TODO

Directions :

- S,W,E,N : South, West, East, North (vertical/horizontal movement)
- SW,NW,NE,SE : South-West, North-West, North-East, South-East (diagonal movement)
- R : random movement (S,W,E,N)
- 1F,1B : one step Forwards/Backwards
- 1A,1T : one step Away from/Towards the player
- K : Keep the same (for teleportation)

5.2 Formal grammar

```
EVENT
  : '[event]' LF+ (CONFIG LF)+ LF+ PAGE+

PAGE
  : '[page]' LF+ (CONFIG_OR_COND LF)* LF+ STATEMENTS? '[end]' LF+

CONFIG
  : CONFIG_VAR '=' PARAMETER

CONFIG_OR_COND
  : LOG_EXPR
  | CONFIG

STATEMENTS          // block of lines that define an event's behavior
  : (STATEMENT LF+)+

STATEMENT           // line that define an event's behavior
  : 'if' WHITESPACE LOG_EXPR LF CODE_BLOCK ('else' LF CODE_BLOCK)?
  | 'loop' LF CODE_BLOCK
  | 'when' WHITESPACE VALUE LF CODE_BLOCK
  | 'break'
  | CMD
  | VAR_MANIPULATION
  | SCRIPT

CODE_BLOCK          // block of lines whose execution is subject to flow control
  : INDENT STATEMENTS DEDENT

CMD
  : CMD_ID PARAMETERS?

PARAMETERS
  : PARAMETER (',' WHITESPACE? PARAMETER)*

PARAMETER
  : (PARAMETER_NAME '=')? PARAMETER_VALUE

PARAMETER_VALUE
  : '[' VALUE (WHITESPACE VALUE)* ']'
  | VALUE

VAR_MANIPULATION
  : VAR_ID ASSIGN_OPERATOR EXPRESSION

EXPRESSION          // expression that returns a value
  : LOG_EXPR
  | NUM_EXPR
  | SCRIPT

LOG_EXPR            // expression that returns a logical value
  : COMPARABLE LOG_OPERATOR COMPARABLE
  | SCRIPT

NUM_EXPR            // expression that returns a numerical value
  : TERM (ADD_OP TERM)*

PARAMETER_NAME
  : WORD
```

```

SCRIPT
    : 's:' WORD (WHITESPACE PARAMETERS)?

TERM
    : FACTOR (MUL_OP FACTOR)*

FACTOR
    : NUMBER
    | '(' NUM_EXPR ')'

CMD_ID
    : WORDS

VAR_ID
    : ':' WORD

VALUE
    : NUMBER
    | STRING
    | WORD

STRING
    : '"' ['^"]* '"'

NUMBER
    : -? [0-9]+ ('.' [0-9]+)?

WHITESPACE
    : ' '+

LF
    : '\r\n'
    | '\n'

WORDS
    : WORD (WHITESPACE WORD)*

WORD
    : [a-z\_0-9]+

LOG_OPERATOR
    : '=='
    | '>='
    | '<='
    | '>'
    | '<'
    | '!='

ASSIGN_OPERATOR
    : '='
    | '+='
    | '-='
    | '*='
    | '/='

MATH_OP
    : '+'
    | '-'
    | '*'
    | '/'

```

- CMD_ID is expected to be one of the defined operation. An error should be thrown otherwise.

- Comments must be stripped before lexing. Multi-line comments aren't supported.
- Tokens (terminal values) `INDENT` and `DEDENT` should be generated when reading the event file in order to represent indentation, thus allowing for block of statements to be syntactically represented.

5.2.1 Configuration variables

Mostly contained in the event, but can be overridden by the pages.

CONFIG_VAR	type	Default	Description.
ID	int	N/A	(Deprecated) Identifies the event.
name	String	N/A	(Deprecated:moved to file name) Identifies the event.
xy	[int, int]	N/A*	Position of the event.
graphic	String	None	Texture of the event.
opacity	int	TODO	.
direction	String	S	[N,E,S,W] Initial facing direction (if has <code>graphic</code>).
trigger	String	None	Trigger for the behavior of the event.
move_animation	bool	TODO	.
stop_animation	bool	TODO	.
direction_fix	bool	TODO	.
through	bool	TODO	.
always_on_top	bool	TODO	.
movement	TODO		.
movement_speed	int	TODO	.
movement_frequency	int	TODO	.
preset	String	None	Proposed "preset" for simple, common events (boulder, door, etc).

* : Mandatory configuration, therefore no default.

Note that, like anything in an event file, it should be read in a case-insensitive way.

6 Maps

Here we will focus on `RPG::Map`. Here are its components :

<code>tileset_id</code>	int	Value of a <code>RPG::Tileset</code> unique identifier component <code>id</code> . The <code>RPG::Tileset</code> object can be retrieved through the global hash <code>\$data_tilesets</code> using the <code>id</code> as the key.
<code>width,height</code>	int,int	Attribute equivalent to <code>data.xsize()</code> and <code>data.ysize()</code> .
<code>autoplay_bgX</code>	bool	Indicated whether an audio is to be played as soon as the map is loaded.
<code>bgX</code>	AudioFile	The audio that is to be played.
<code>encounter_list</code>	Array	TODO
<code>encounter_step</code>	int	TODO
<code>data</code>	Table of int	Contain the <i>map</i> representation of the 3 tile layers.
<code>events</code>	Hash	Contain the <i>Event</i> representation (<code>RPG::Event</code>) for this map.

6.1 Associated classes

- The role of the associated `Game_Map` instance is to be studied further, but current understanding indicated that it is an alias derived from its `RPG::Map` instance that is tailored for PE's needs.
- About the associated `RPG::MapInfo` instance : It contains a few useful informations :

<code>name</code>	String	The name of the map.
<code>parent_id</code>	int	In the map tree, the id of the parent map.

This information can be retrieved directly from the compiled *MapInfos* file :

```
mapinfos = pbLoadRxData("Data/MapInfos")
map_name = mapinfos[id].name
parent_map_id = mapinfos[id].parent_id
parent_map_name = mapinfos[parent_map_id].name rescue nil
```

- `RPG::AudioFile` : Basic data container :

<code>name</code>	String	The name of the audio file (no extension).
<code>volume</code>	int	Acts like a volume slider, normalized at 100.
<code>pitch</code>	int	Allows to adjust sound pitch, normalized at 100.

- `RPG::Tileset` : Represents a normal tileset :

<code>id</code>	int	The id of the tileset.
<code>name</code>	String	Its name (no extension).
<code>autotile_names</code>	Array of String	Names of associated autotiles (up to 7).
<code>panorama_X</code>		TODO
<code>fog_X</code>		TODO
<code>battleback_name</code>	String	Name of the texture that appears during combat.
<code>passages</code>	2D Table of int	Properties of individual textures.
<code>priorities</code>	2D Table of int	Properties of individual textures.
<code>terrain_tags</code>	2D Table of int	Properties of individual textures.

- **Table** : Used for 2D/3D arrays, with 3 class methods to retrieve dimensions. x and y correspond to their GUI map representation and z the map layers (background, intermediate, foreground).

7 Remarks

7.1 Contact

Contact the author by email : David.Rodriguez.1@etu.unige.ch

7.2 Privacy Policy

This document and its content are private and confidential. It is only intended for its academic recipient. It is strictly prohibited to copy, print, publish, share or distribute any part of it without written permission from its original author.

If you received this document by mistake, please inform its author and delete it. Thank you for your cooperation and understanding.