RPG Maker XP documentation

David Rodriguez Soares

July 28, 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 7
4.4 Complex commands 16

5 Command Representation decisions **19**

6 Maps **20**

7 Remarks **21**
7.1 Contact 21
7.2 Privacy Policy 21

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).

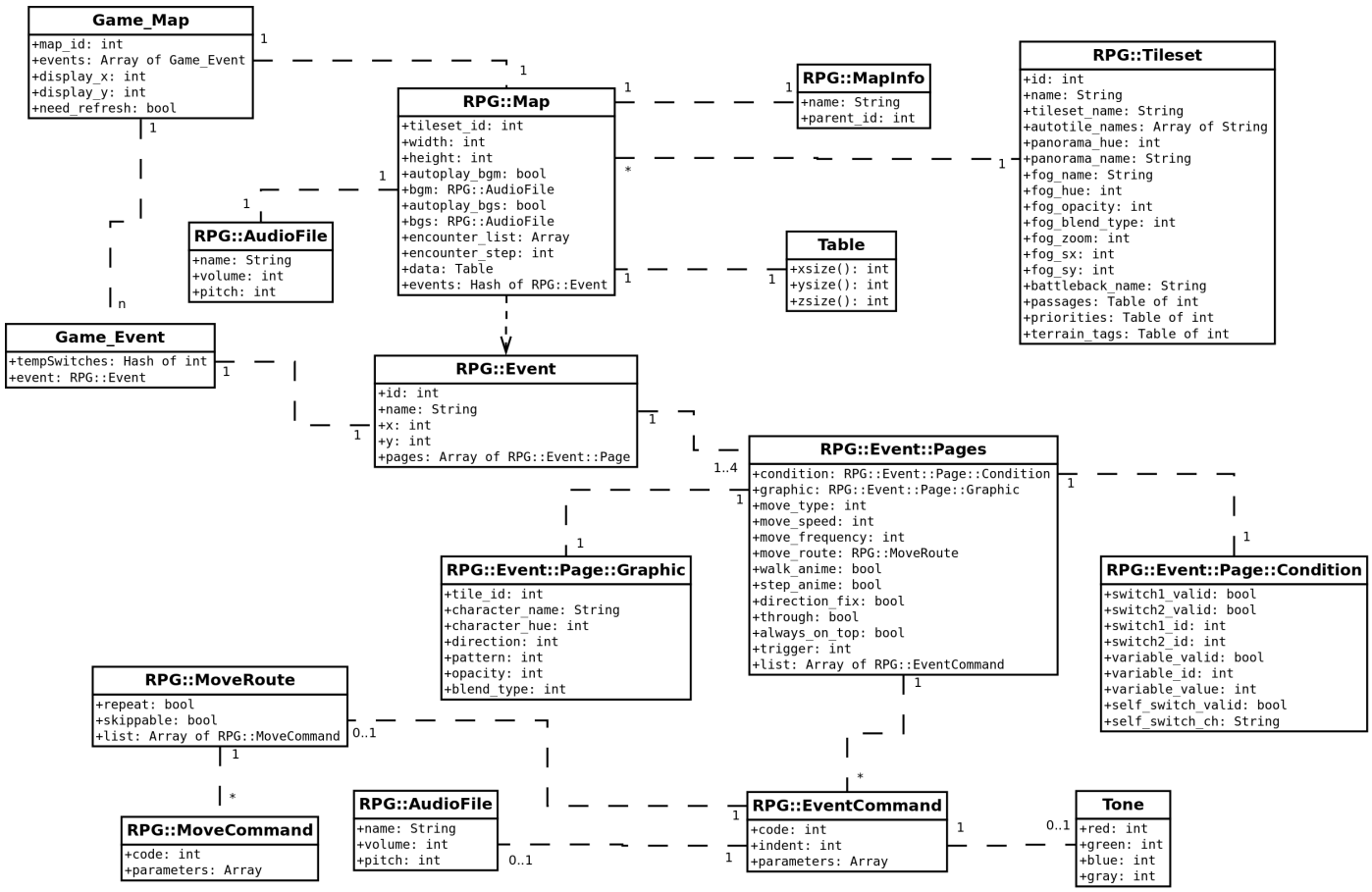


Figure 1: Simplified class map representation for Map/Event

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

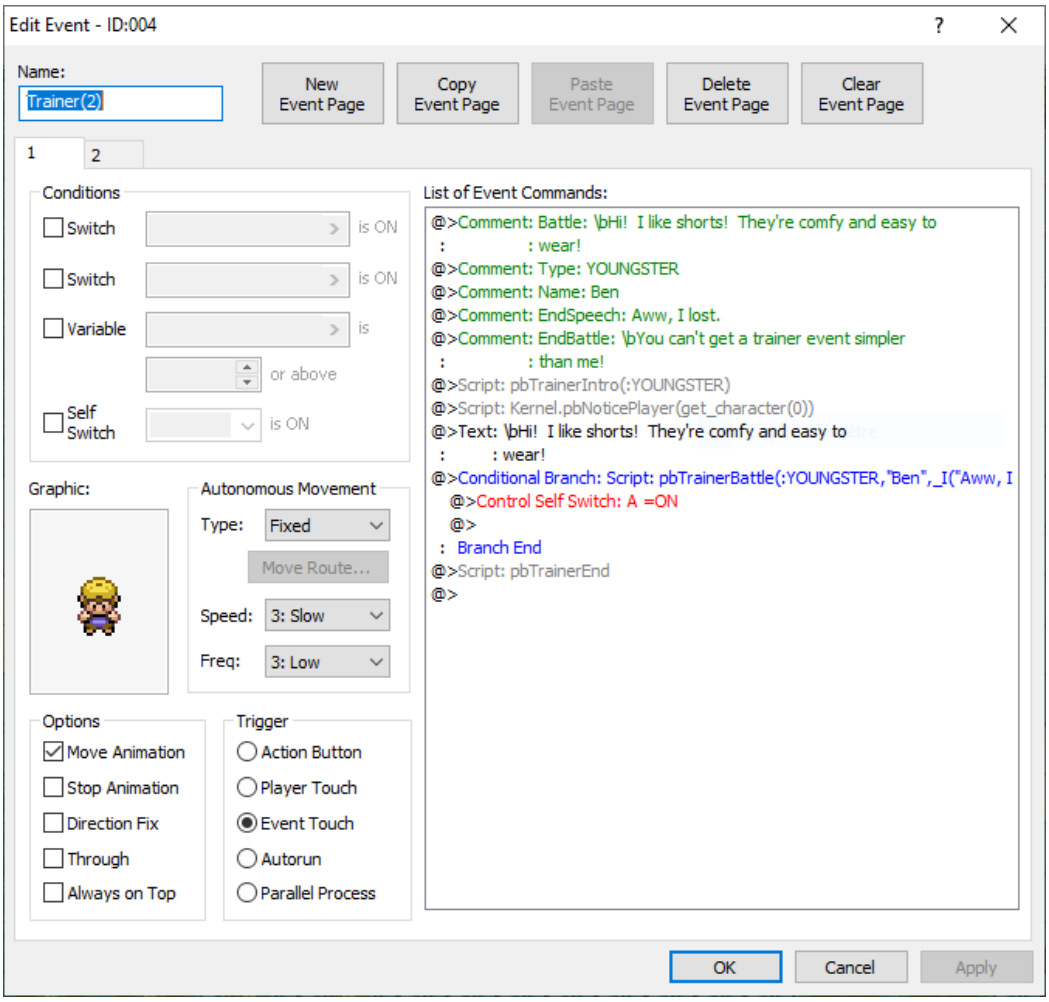
Note : There is no inheritance relationship between any two classes represented. Arities are logically deduced and may not be exact depending in proprietary implementation details. Class `RPG::AudioFile` was duplicated for ease of association routing.

3 Events

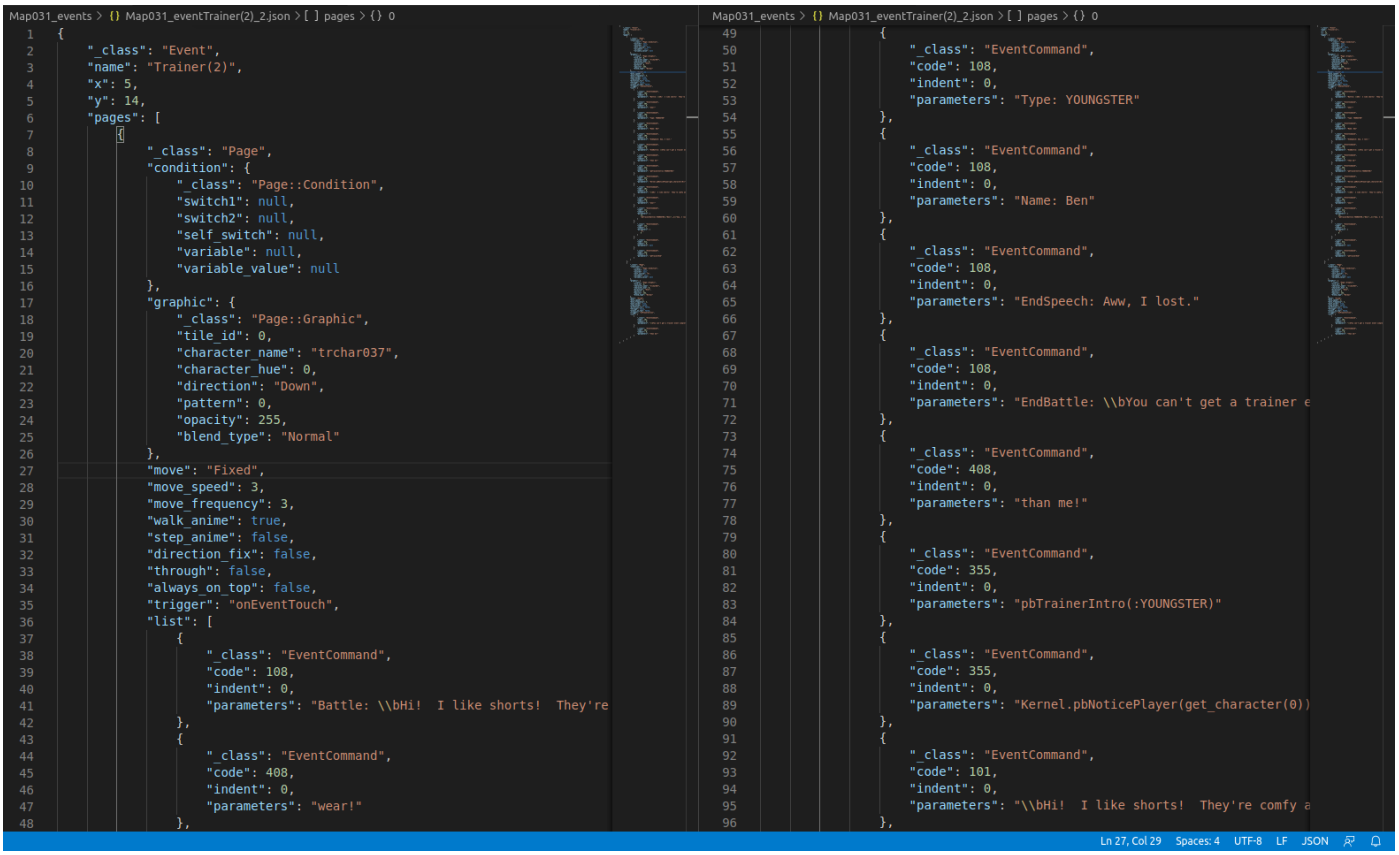
An event, or more precisely a *map event*, is a way to introduce elements with behavior, therefore bringing flexibility and dynamism into the game world.

Events have two aspects :

- A GUI element



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



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.

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, <u>Gold</u>, Steps, PlayTime, *Timer*, SaveCount.

  They should all be read accessible. <u>Underlined ones should also be write accessible</u>. *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	`RPG::MoveCommand` - Move to the South
	Parameters	None
	Notes	See footnote[1]
	Representation	"Move, S"
2	Description	`RPG::MoveCommand` - Move to the West
	Parameters	None
	Notes	See footnote[1]
	Representation	"Move, W"
3	Description	`RPG::MoveCommand` - Move to the East
	Parameters	None
	Notes	See footnote[1]
	Representation	"Move, E"
4	Description	`RPG::MoveCommand` - Move to the North
	Parameters	None
	Notes	See footnote[1]
	Representation	"Move, N"
5	Description	`RPG::MoveCommand` - Move to the SouthWest
	Parameters	None
	Notes	See footnote[1]
	Representation	"Move, SW"
6	Description	`RPG::MoveCommand` - Move to the SouthEast
	Parameters	None
	Notes	See footnote[1]
	Representation	"Move, SE"
7	Description	`RPG::MoveCommand` - Move to the NorthWest
	Parameters	None
	Notes	See footnote[1]
	Representation	"Move, NW"
8	Description	`RPG::MoveCommand` - Move to the NorthEast
	Parameters	None
	Notes	See footnote[1]
	Representation	"Move, NE"
9	Description	`RPG::MoveCommand` - Move at random (N,E,S,W)
	Parameters	None
	Notes	See footnote[1]
	Representation	"Move, R"
10	Description	`RPG::MoveCommand` - Move towards player
	Parameters	None
	Notes	See footnotes[1,3]
	Representation	"Move, TODO"

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

22	Description	`RPG::MoveCommand` - Turn 180°
	Parameters	None
	Notes	See footnote[2]
	Representation	"Turn, 180"
23	Description	`RPG::MoveCommand` - Turn 90° to the left or right, at random
	Parameters	None
	Notes	See footnote[2]
	Representation	"Turn, 90random"
24	Description	`RPG::MoveCommand` - Turn at random (90° or 180°)
	Parameters	None
	Notes	See footnote[2]
	Representation	"Turn, random"
25	Description	`RPG::MoveCommand` - Turn towards player
	Parameters	None
	Notes	See footnotes[2,3]
	Representation	"Turn, TODO"
26	Description	`RPG::MoveCommand` - Turn away from player
	Parameters	None
	Notes	See footnotes[2,3]
	Representation	"Turn, TODO"
33	Description	`RPG::MoveCommand` - Turn ON walking animation
	Parameters	None
	Notes	
	Representation	"Animation, ON"
34	Description	`RPG::MoveCommand` - Turn OFF walking animation
	Parameters	None
	Notes	
	Representation	"Animation, OFF"
37	Description	`RPG::MoveCommand` - Turn ON "through"
	Parameters	None
	Notes	Equivalent to activating "walk through walls", making it possible to walk through impassable tiles/characters.
	Representation	"WTW, ON"
38	Description	`RPG::MoveCommand` - Turn OFF "through"
	Parameters	None
	Notes	Equivalent to deactivating "walk through walls".
	Representation	"WTW, OFF"
39	Description	`RPG::MoveCommand` - Always on top ON
	Parameters	None
	Notes	Elevate the display priority, therefore bringing the event graphic to the forefront (above any tile/character)
	Representation	"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 back-slashes 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 s `[String]`
	Notes	Happens after a 108.
	Representation	"# s"

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 *s* `[String]`
	Notes	Sets a label to allow jumping to.
	Representation	"Label, *s*"
119	Description	`RPG::EventCommand` - Jump to Label
	Parameters	[1] - **0**:label name *s* `[String]`
	Notes	Jumps to a label.
	Representation	"Goto, *s*"
121	Description	`RPG::EventCommand` - Control switches
	Parameters	[3] - **0**:starting switch *ssa* `[integer]`, **0**:starting switch *ssz* `[integer]`, **0**:new state *n* `[integer]`
	Notes	Batch control is unused in PE, therefore deprecated. *n* follows "common relation 3".
	Representation	"Control Switch, *ssa*.toString(), *n*.toString()"
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 *s* `[String of length 1]`, **1**:new state *n* `[integer 0-1]`
	Notes	*n* follows "common relation 3".
	Representation	"Control SS, *s*, *n*.toString()"
125	Description	`RPG::EventCommand` - Change Gold
	Parameters	[3] - **0**:operation *o* `[integer 0-1]`, **1**:operand *n* `[integer 0-1]`, **2**:value *v* `[integer]`
	Notes	Values of *n*: 0:*v* is a constant, 1:*v* is a variable(id). *o* follows "common relation 4"
	Representation	"Control Variable, :Money *o*.toString() *v*.toString()"

201	Description	`RPG::EventCommand` - Transfer Player
	Parameters	[6] - **1**:map m `[integer]`, **2**:coordinate x `[integer]`, **3**:coordinate y `[integer]`, **4**:player direction d `[integer]`, **5**:fading f `[integer]`.
	Notes	{0} must be 0, 1 unused in PE. d follows "common relation 5". f follows "common relation 3".
	Representation	"Transfer Player, destination=(m,x,y), direction=d.toString(), fading=f.toString()"
202	Description	`RPG::EventCommand` - Set Event Location
	Parameters	[5] - **0**:event id e `[integer]`, **2**:coordinate x `[integer]`, **3**:coordinate y `[integer]`, **4**:direction d `[integer]`
	Notes	Change an event's location on the current map. {1} must be 0, other values unused in PE. d follows "common relation 5".
	Representation	"Move Event, e.toString(), (x,y), direction=d.toString()"
208	Description	`RPG::EventCommand` - Change Transparency Flag
	Parameters	[2] - **0**:flag d `[integer 0-1]`
	Notes	When transparency is set, the graphic isn't displayed. d follows "common relation 3".
	Representation	"Set Transparency, d.toString()"
209	Description	`RPG::EventCommand` - Set Move Route
	Parameters	[2] - **0**:target id d `[integer]`, **1**:`RPG::MoveRoute`
	Notes	
	Representation	"Set Move Route, d.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 s `[String]`
	Notes	Plays the animation. TODO:research how transition work.
	Representation	"Transition, s, freeze={True/False}"
223	Description	`RPG::EventCommand` - Change Screen Color Tone
	Parameters	[2] - **0**:`RPG::Tone`, **1**:duration(frames) d `[integer]`
	Notes	Typically used in fade out (to black/white)/fade in cycles. d to be changed into ms.
	Representation	"Change Screen Color Tone, d, {0}.toString()", "Fadein, d", "Fadeout, d, {color}"
225	Description	`RPG::EventCommand` - Screen Shake
	Parameters	[3] - **0**:shake power `[integer]`, **1**:shake speed `[integer]`, **2**:duration(frames) d `[integer]`
	Notes	Scarcely used in PE, {0} and {1} are not well defined so they can be deprecated. d to be changed into ms.
	Representation	"Shake Screen, d"
231	Description	`RPG::EventCommand` - Show Picture
	Parameters	See "Show Picture" section.
	Notes	
	Representation	TODO

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

354	Description	`RPG::EventCommand` - Return to Title Screen
	Parameters	None
	Notes	
	Representation	"Return to Title Screen"
355	Description	`RPG::EventCommand` - Script
	Parameters	[1] - **0**:script string `[String]`
	Notes	To be overhauled.
	Representation	TODO
655	Description	`RPG::EventCommand` - Script (continued)
	Parameters	[1] - **0**:script string `[String]`
	Notes	To be overhauled.
	Representation	TODO
402	Description	`RPG::EventCommand` - When
	Parameters	[1] - **0**:choice id `[integer]`, **1**:choice string equivalent `[integer]`
	Notes	Used with choices and conditional branches, has code block per choice.
	Representation	TODO
404	Description	`RPG::EventCommand` - End of When
	Parameters	None
	Notes	
	Representation	None
411	Description	`RPG::EventCommand` - Else
	Parameters	None
	Notes	Used with conditional branch `111`.
	Representation	TODO
412	Description	`RPG::EventCommand` - 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	`RPG::EventCommand` - Repeat above
	Parameters	None
	Notes	Marks end of Loop `112` code block.
	Representation	TODO

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

[1]Movements consolidated with new *Move* command with argument.
[2]Turs consolidated with new *Turn* command with argument.
[3]Unknown algorithm to determine direction "towards player" and "away from player.
[4]Is 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:Substractive (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 n `[integer]`, **2**:switch state d `[integer 0-1]`
Notes	d follows "common relation 3".
Representation	"If, n.toString(), d.toString()"

1 Check *Variable* value.

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

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 yx `[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, zy), 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 yx `[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, zy), 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 :

- **C**omma **S**eparated **V**alues-style : Enable software to leverage CSV libraries when interpreting *Events*, significantly decreasing implementation complications.
- 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 explicitely 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-olny 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` !

6 Maps

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

`tileset_id`	int	Value of a `RPG::Tileset` unique identifier component `id`. The `RPG::Tileset` object can be retrieved through the global hash `$data_tilesets` using the id as the key.
`width,height`	int,int	Attribute euivalent to `data.xsize()` and `data.ysize()`.

7 Remarks

7.1 Contact

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

7.2 Privacy Policy