RPG Maker XP documentation

David Rodriguez Soares

August 3, 2020

Privacy policy

Contents

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*

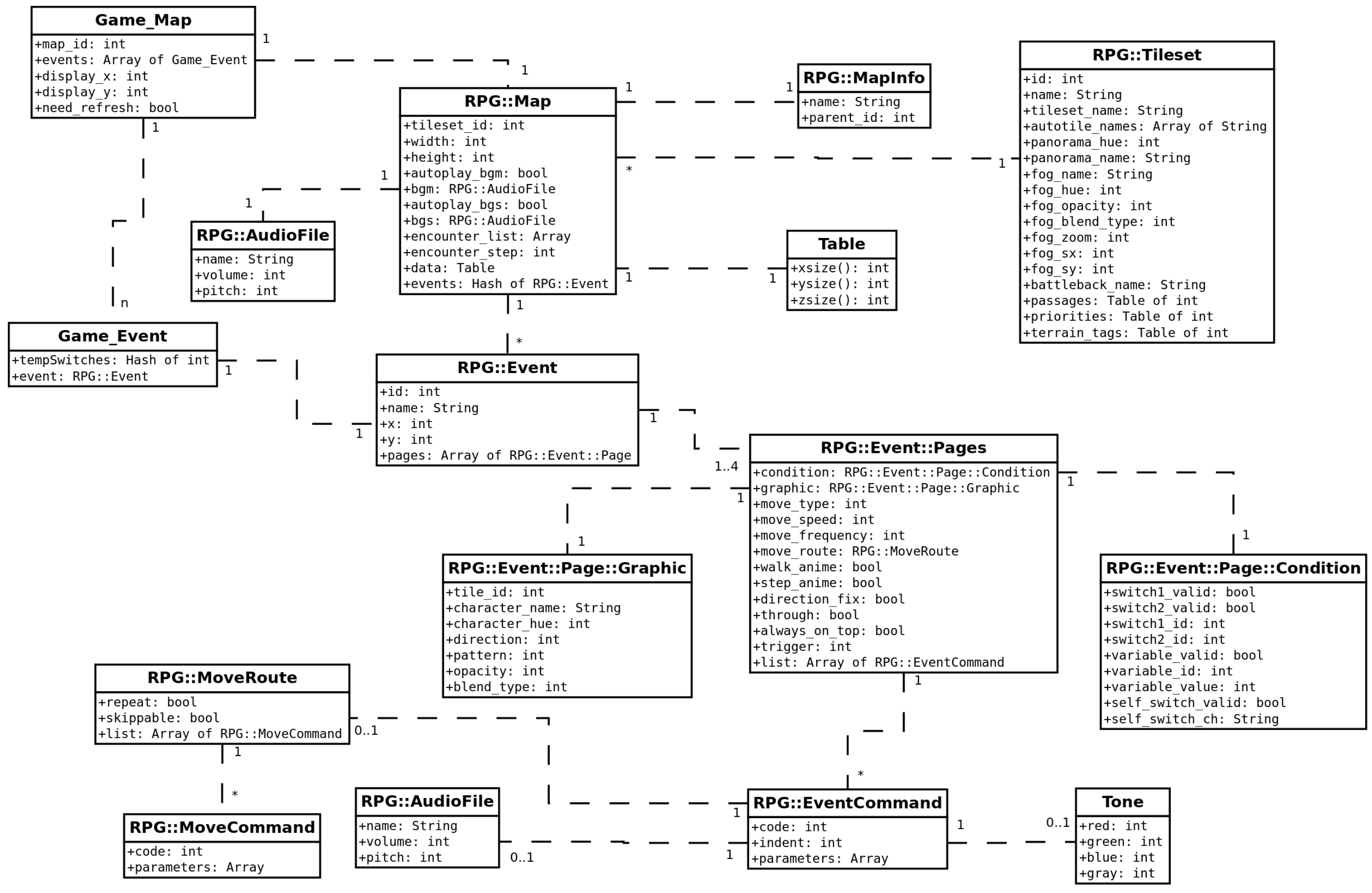


Figure 1: Simplified class map representation for Map/Event

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

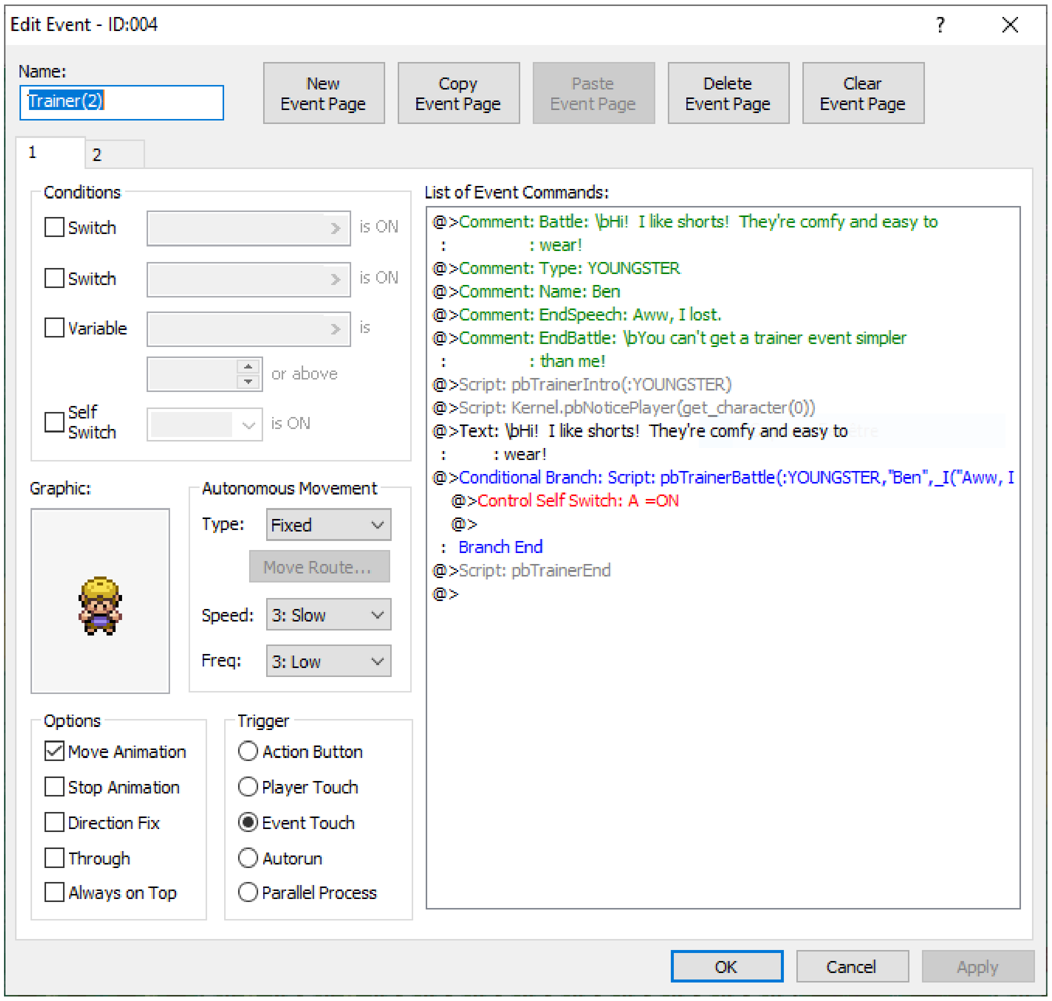
Note : There is no inheritance relationship between any two classes represented. Arities are logically deduced and may not be exact depending in proprietary implementation details. Class `RPG::AudioFile` was duplicated for ease of association routing.

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



Name: Trainer(2)

Conditions: Switch is ON, Switch is ON, Variable is ... or above, Self Switch is ON (none checked)

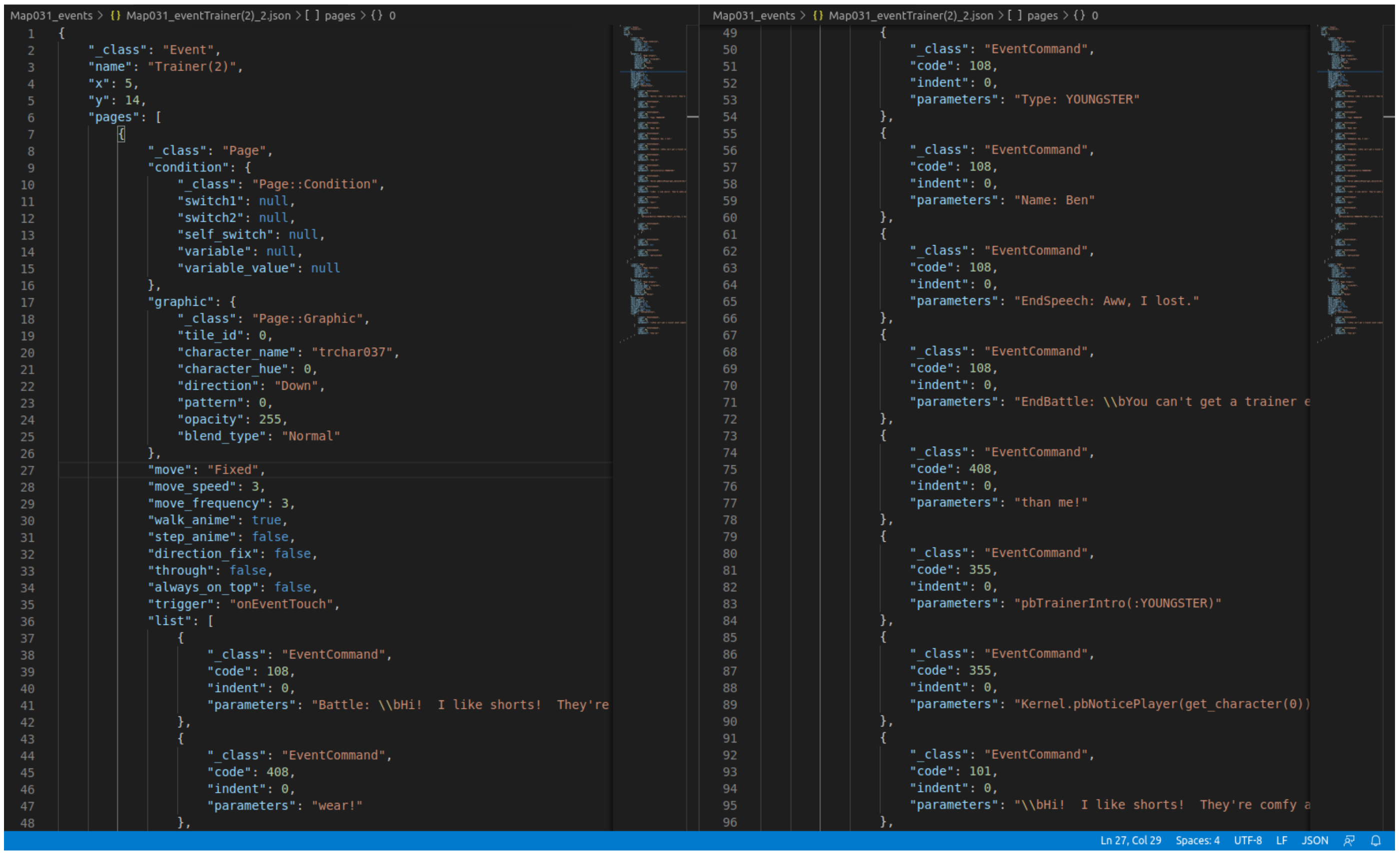
Autonomous Movement — Type: Fixed; Speed: 3: Slow; Freq: 3: Low

Options: [✓ Move Animation] Stop Animation, Direction Fix, Through, Always on Top

Trigger: Action Button, Player Touch, [✓ Event Touch], Autorun, Parallel Process

```
@>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
  @>Control Self Switch: A =ON
  @>
 :  Branch End
@>Script: pbTrainerEnd
@>
```

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



Map031_events > {} Map031_eventTrainer(2)_2.json > [ ] pages > {} 0

```json
{
    "_class": "Event",
    "name": "Trainer(2)",
    "x": 5,
    "y": 14,
    "pages": [
        {
            "_class": "Page",
            "condition": {
                "_class": "Page::Condition",
                "switch1": null,
                "switch2": null,
                "self_switch": null,
                "variable": null,
                "variable_value": null
            },
            "graphic": {
                "_class": "Page::Graphic",
                "tile_id": 0,
                "character_name": "trchar037",
                "character_hue": 0,
                "direction": "Down",
                "pattern": 0,
                "opacity": 255,
                "blend_type": "Normal"
            },
            "move": "Fixed",
            "move_speed": 3,
            "move_frequency": 3,
            "walk_anime": true,
            "step_anime": false,
            "direction_fix": false,
            "through": false,
            "always_on_top": false,
            "trigger": "onEventTouch",
            "list": [
                {
                    "_class": "EventCommand",
                    "code": 108,
                    "indent": 0,
                    "parameters": "Battle: \\bHi!  I like shorts!  They're
                },
                {
                    "_class": "EventCommand",
                    "code": 408,
                    "indent": 0,
                    "parameters": "wear!"
                },
```

Map031_events > {} Map031_eventTrainer(2)_2.json > [ ] pages > {} 0

```json
                {
                    "_class": "EventCommand",
                    "code": 108,
                    "indent": 0,
                    "parameters": "Type: YOUNGSTER"
                },
                {
                    "_class": "EventCommand",
                    "code": 108,
                    "indent": 0,
                    "parameters": "Name: Ben"
                },
                {
                    "_class": "EventCommand",
                    "code": 108,
                    "indent": 0,
                    "parameters": "EndSpeech: Aww, I lost."
                },
                {
                    "_class": "EventCommand",
                    "code": 108,
                    "indent": 0,
                    "parameters": "EndBattle: \\bYou can't get a trainer e
                },
                {
                    "_class": "EventCommand",
                    "code": 408,
                    "indent": 0,
                    "parameters": "than me!"
                },
                {
                    "_class": "EventCommand",
                    "code": 355,
                    "indent": 0,
                    "parameters": "pbTrainerIntro(:YOUNGSTER)"
                },
                {
                    "_class": "EventCommand",
                    "code": 355,
                    "indent": 0,
                    "parameters": "Kernel.pbNoticePlayer(get_character(0))
                },
                {
                    "_class": "EventCommand",
                    "code": 101,
                    "indent": 0,
                    "parameters": "\\bHi!  I like shorts!  They're comfy a
                },
```

Ln 27, Col 29   Spaces: 4   UTF-8   LF   JSON

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, <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
1	Description	`RPG::MoveCommand` - Move to the South
	Parameters	None
	Notes	See footnote[1]
2	Description	`RPG::MoveCommand` - Move to the West
	Parameters	None
	Notes	See footnote[1]
3	Description	`RPG::MoveCommand` - Move to the East
	Parameters	None
	Notes	See footnote[1]
4	Description	`RPG::MoveCommand` - Move to the North
	Parameters	None
	Notes	See footnote[1]
5	Description	`RPG::MoveCommand` - Move to the SouthWest
	Parameters	None
	Notes	See footnote[1]
6	Description	`RPG::MoveCommand` - Move to the SouthEast
	Parameters	None
	Notes	See footnote[1]
7	Description	`RPG::MoveCommand` - Move to the NorthWest
	Parameters	None
	Notes	See footnote[1]
8	Description	`RPG::MoveCommand` - Move to the NorthEast
	Parameters	None
	Notes	See footnote[1]
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]
11	Description	`RPG::MoveCommand` - Move away from player
	Parameters	None
	Notes	See footnotes[1,3]
12	Description	`RPG::MoveCommand` - Take 1 step forward
	Parameters	None
	Notes	See footnote[1]

13	Description	`RPG::MoveCommand` - Take 1 step backward
	Parameters	None
	Notes	See footnote[1]
14	Description	`RPG::MoveCommand` - Jump to relative coordinates on the same map
	Parameters	[2] - **0**:deltaX `[signed integer]`, **1**:deltaY `[signed integer]`
	Notes	
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.
16	Description	`RPG::MoveCommand` - Turn towards South
	Parameters	None
	Notes	See footnote[2]
17	Description	`RPG::MoveCommand` - Turn towards West
	Parameters	None
	Notes	See footnote[2]
18	Description	`RPG::MoveCommand` - Turn towards East
	Parameters	None
	Notes	See footnote[2]
19	Description	`RPG::MoveCommand` - Turn towards North
	Parameters	None
	Notes	See footnote[2]
20	Description	`RPG::MoveCommand` - Turn 90° right, relative to current position
	Parameters	None
	Notes	See footnote[2]
21	Description	`RPG::MoveCommand` - Turn 90° left, relative to current position
	Parameters	None
	Notes	See footnote[2]
22	Description	`RPG::MoveCommand` - Turn 180°
	Parameters	None
	Notes	See footnote[2]
23	Description	`RPG::MoveCommand` - Turn 90° to the left or right, at random
	Parameters	None
	Notes	See footnote[2]
24	Description	`RPG::MoveCommand` - Turn at random (90° or 180°)
	Parameters	None
	Notes	See footnote[2]
25	Description	`RPG::MoveCommand` - Turn towards player
	Parameters	None
	Notes	See footnotes[2,3]
26	Description	`RPG::MoveCommand` - Turn away from player
	Parameters	None
	Notes	See footnotes[2,3]

33	Description	`RPG::MoveCommand` - Turn ON walking animation
	Parameters	None
	Notes	
34	Description	`RPG::MoveCommand` - Turn OFF walking animation
	Parameters	None
	Notes	
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.
38	Description	`RPG::MoveCommand` - Turn OFF ”through”
	Parameters	None
	Notes	Equivalent to deactivating ”walk through walls”.
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)
40	Description	`RPG::MoveCommand` - Always on top OFF
	Parameters	None
	Notes	
41	Description	`RPG::MoveCommand` - Change event’s graphic
	Parameters	TODO
	Notes	
42	Description	`RPG::MoveCommand` - Change event’s graphic opacity
	Parameters	[1] - **0**:new opacity value n `[integer 0-255]`
	Notes	
44	Description	`RPG::MoveCommand` - Play a sound effect
	Parameters	TODO
	Notes	
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).
401	Description	`RPG::EventCommand` - Show text (continued)
	Parameters	[1] - **0**:text s `[String]`
	Notes	Continuation of 101.
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.
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”
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

108	Description	`RPG::EventCommand` - Comment
	Parameters	[1] - **0**:comment text s `[String]`
	Notes	Has no effect. TODO:research link to particle effects.
408	Description	`RPG::EventCommand` - Comment (continued)
	Parameters	[1] - **0**:comment text s `[String]`
	Notes	Happens after a 108.
111	Description	`RPG::EventCommand` - Conditional branch
	Parameters	See "Conditional branch" section.
	Notes	Complex but essential command.
112	Description	`RPG::EventCommand` - Loop
	Parameters	None
	Notes	Loops over commands until broken. TODO:research usage
113	Description	`RPG::EventCommand` - Break loop
	Parameters	None
	Notes	Escape innermost loop. TODO:research usage
115	Description	`RPG::EventCommand` - Exit Event Processing
	Parameters	None
	Notes	TODO:research usage
118	Description	`RPG::EventCommand` - Label
	Parameters	[1] - **0**:label name s `[String]`
	Notes	Sets a label to allow jumping to.
119	Description	`RPG::EventCommand` - Jump to Label
	Parameters	[1] - **0**:label name s `[String]`
	Notes	Jumps to a label.
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".
122	Description	`RPG::EventCommand` - Control variables
	Parameters	See "Control variables" section.
	Notes	Batch control is unused in PE, therefore deprecated.
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".
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"
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".

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".
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".
209	Description	`RPG::EventCommand` - Set Move Route
	Parameters	[2] - **0**:target id d `[integer]`, **1**:`RPG::MoveRoute`
	Notes	
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.
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.
222	Description	`RPG::EventCommand` - Execute Transition
	Parameters	[1] - **0**:transition file name s `[String]`
	Notes	Plays the animation. TODO:research how transition work.
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.
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.
231	Description	`RPG::EventCommand` - Show Picture
	Parameters	See "Show Picture" section.
	Notes	
232	Description	`RPG::EventCommand` - Move Picture
	Parameters	See "Move Picture" section.
	Notes	
235	Description	`RPG::EventCommand` - Erase Picture
	Parameters	[1] - **0**:picture id `[integer]`
	Notes	
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.
241	Description	`RPG::EventCommand` - Play BGM
	Parameters	[1] - **0**:audio a `[AudioFile]`
	Notes	
242	Description	`RPG::EventCommand` - Fade Out BGM
	Parameters	[1] - **0**:duration (seconds) n `[integer]`

247	Description	`RPG::EventCommand` - Memorize BGM/BGS
	Parameters	None
	Notes	
248	Description	`RPG::EventCommand` - Restore BGM/BGS
	Parameters	None
	Notes	
249	Description	`RPG::EventCommand` - Play ME
	Parameters	[1] - **0**:audio *a* `[AudioFile]`
	Notes	
250	Description	`RPG::EventCommand` - Play SE
	Parameters	[1] - **0**:audio *a* `[AudioFile]`
	Notes	
314	Description	`RPG::EventCommand` - Restore All
	Parameters	[1] - **0**:actor id `[integer]`
	Notes	Equivalent to healing and restoring PPs. Ignore parameter.
354	Description	`RPG::EventCommand` - Return to Title Screen
	Parameters	None
	Notes	
355	Description	`RPG::EventCommand` - Script
	Parameters	[1] - **0**:script string `[String]`
	Notes	To be overhauled.
655	Description	`RPG::EventCommand` - Script (continued)
	Parameters	[1] - **0**:script string `[String]`
	Notes	To be overhauled.
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.
404	Description	`RPG::EventCommand` - End of When
	Parameters	None
	Notes	
411	Description	`RPG::EventCommand` - Else
	Parameters	None
	Notes	Used with conditional branch `111`.
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?).
411	Description	`RPG::EventCommand` - Repeat above
	Parameters	None
	Notes	Marks end of Loop `112` code block.

[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 :

- 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` !
- 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 `parameter_name = parameter_value` (parameter name recommended but not mandatory; 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`'≡'`True`', '`:OFF`'≡'`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	[5] - **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` - Set a sequence of commands, to be executed by a set event
Parameters	[1] - **0**: `event* - [String]`
Notes	`event` defaults to self. Must be followed by a *code block*.
	Used to move other events or to semantically indicate a ”move sequence/route”.
Example	See ”Move Route” section below.
Description	`Screen Shake`
Parameters	[1] - **0**: `duration - [int]`
Notes	`duration` is expressed in milliseconds.
Example	”`Screen Shake 600`”
Description	`Transition` - Execute transition visual effect.
Parameters	[2] - **0**: `name - [String]`, **1**: `freeze - [True/False]`
Notes	If `freeze` is enabled, stops every animation.
Example	”`Transition name="battle1", freeze=True`”
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` - Set overworld's weather.
Parameters	[2] - **0**: `name - [String]`, **1**: `duration* - [int]`
Notes	`duration` defaults to infinite duration. The effect scope of `Set weather` is to be determined (for current map, radius on the current map, across maps). Provisionally, it's limited to current map.
Example	”`Set weather name="Rainy", duration=12000`”

Description	`Fade out BGM`
Parameters	[1] - **0**: `duration - [int]`
Notes	`duration` in milliseconds.
Example	"`Fade out BGM 3000`"
Description	`Memorize BGx`
Parameters	[0]
Example	"`Memorize BGx`"
Description	`Restore BGx`
Parameters	[0]
Example	"`Restore BGx`"
Description	`Restore All`
Parameters	[0]
Notes	Restore all stats for player's party.
Example	"`Restore All`"
Description	`Return to title screen`
Parameters	[0]
Notes	Quits current game and returns to title screen (without saving).
Example	`Return to title screen`
Description	`Save`
Parameters	[1] - **0**: `allow_cancel - [True/False]`
Notes	Prompts a "save your progress" dialog to the player.
Example	`Save allow_cancel=True`

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, not recommended)
- R : random movement (S,W,E,N)
- 1F,1B : one step Forwards/Backwards (according to current orientation/direction)
- 1A,1T : one step Away from/Towards the player
- K : Keep the same (for teleportation)

Execution flow control :

- `if code_block [else code_block]?` : For implementing conditional execution of code blocks.
- `loop code_block` : `code_block` must contain a `break` statement for the loop to not be infinite. Infinite loop detection should be implemented.
- `choice ... [when (value) code_block]+` : For implementing behavior on player's choice.

Total :

- 25 (+ 1*new*) commands + 3 forms of flow control !
- vs. 81 commands !
- 3 pages de doc vs. 9 pages de doc

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 `graphic`).
`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 :

`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 equivalent to `data.xsize()` and `data.ysize()`.
`autoplay_bgX`	bool	Indicated whether an audio is to be played as soon as the map is loaded.
`bgX`	AudioFile	The audio that is to be played.
`encounter_list`	Array	TODO
`encounter_step`	int	TODO
`data`	Table of int	Contain the *map* representation of the 3 tile layers.
`events`	Hash	Contain the *Event* representation (`RPG::Event`) 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 :

`name`	String	The name of the map.
`parent_id`	int	In the map tree, the id of the parent map.

  This information can 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 :

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

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

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

- `Table` : Used for 2D/3D arrays, with 3 class mathods 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