PoGER

A Bachelor's project

Privacy policy

This is a confidential document and should not be distributed under any circumstance. Please click and read.

Abstract

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Contents

1 Inception

1.1 A Bachelor's project's journey

This is intended to document my journey through my Bachelor's project. Some information may be omitted for brevity or because it is present in accompanying scripts.

There are 3 major steps : inception, research and implementation.

1.2 Brainsorm

This is the **first step** of a Bachelor's project : choosing what to do. It took me significantly more time that I anticipated before I converged towards an answer.

When hunting for a project idea, I listed a few concepts that I would like them to feature :

- Meta-programming
- Creating a domain specific language (descriptive or interpreted)
- Multi-platform app
- A game
- Something that would be useful to many people, developers or users
- An open-source project, free software if possible

Here were my requirements :

- I didn't want to work on web-related stuff. That means no servlets, microservices, HTML or CSS, etc
- I wanted to work with few programming languages, my preference going to Python.
- I didn't want to just do a research work. I wanted to deliver an actual piece of software.
- The project should be some sort of program with an user interface.

I had a hard time trying to find actual ideas that both matched my requirements and made any sense as an academic work.

1.3 Quest for an Original Idea

In my quest for an original idea for a project, I discussed with friends and professors. Some professors had project proposition in their fields, none of which I found particularly attractive.

One of my friends worked on studying a piece of simulation software and its physics engine, which was inspiring but not quite what I was looking for.

I explored a few concepts that went nowhere, but eventually I settled on the following : I wanted to try studying a piece of software called **Pokemon Essentials**, a game engine of sorts, and the games made with it. Moreover, I wanted to do something with it, come up with an idea for a project that branched from it.

This is the beginning of the transition to the **research stage** of the project.

1.4 Small aside

My concept choice need an explanation : I remembered a game I played years ago. It was a Pokemon game developped on RPG Maker XP. I was curious at the time and wondered if I could contribute in some way to this kind of projects. My curiosity led me to research how they were developed.

What I discovered was a clustered universe of fan-made games :

- ROM hacks : games that build upon the code commercial products, typically GameBoy games.
  - \+ They can be played on common emulatory on any platform.
  - \- Programming for them is very technical.
- Games made with RPG Maker XP : They build on top of the *Pokemon Essentials* project.
  - \+ Easier to program for than ROM hacks, has less limitations and RPG Maker XP offers convenient tools.
  - \- RPG Maker XP is an aging RPG engine, programming for them is technical and they are only compatible with Windows (even running them on compatibility layers such as Wine on Linux isn't always functional). Also, performance is typically poor and RPG Maker XP is a closed-source software, making porting attempts extremely complex. Few projects were completed, typically closed-source.
- Original games : Original software, typically built for Windows or the web.
  - \+ Can have great performance, new game mechanics (eg. MMORPG style)
  - \- Typically closed-source, few projects exist.

Quickly, I got discouraged by the apparent complexity and the sheer amount of failed/incomplete similar games (examples here). But I never let go the idea of contributing that development scene.

1.5 Defining the project through research

During the research stage of the project, I experimented with RPG Maker XP, Pokemon Essentials and projects based on it. This was done to obtain an understanding of how they work and which direction to choose.

What I rapidly determined was that these tools weren't adapted to creating a game in a modern way, and suffered from their age and design decision, to the point that they were basically abandoned and deprecated. It was clear to me that I should not try to build upon them, but rather bring them up somehow.

I won't detail every design decision here, but each one was made toward the goal of addressing the identified shortcomings of the studied software. Please read the *vision document* for more information.

At first I looked into implementing a similar engine that would be decoupled from RPG Maker XP and would run games as programs written in an interpreted domain-specific language I would create. A complementary piece of software would automate the port of games written for the original Pokemon Essentials engine.

The concept of game engine recreation seemed fitting for an academic work, so I validated that idea.

As I looked more into it, the project concept got more refined. As a placeholder name, **PoGER** (***Po****kemon Essentials* ***G****ame* ***E****ngine* ***R****ecreation*) was chosen.

The Problem Statement is a crucial : it defines the observed issues with the current situation and what the objectives for the proposed solution are. The following is taken from the *vision document* for PoGER :

The problem of	• PE being based on RPGXP, therefore being bound to Windows • Performance being poor • PE being a convoluted solution to make fangames • The need to being fluent in RPGXP-specific Ruby • Having no separation between game code and PE code • General lack of standards and documentation • PE being fan-made and not well maintained • Each fangame is isolated, and typically lacks multi-player or online features.
affects	• Anyone willing to code/coding a fan-game • People willing to play fangames on other platforms
the impact of which is	a variety of unnecessary technical problems
A successful solution would be	An implementation of PE that is : • performant • available (multi-platform) • straightforward and simpler to develop for • allows to discover fangames • has online capabilities • is separate from the fangames • provides documentation, sets standards • open-source to allow contributors to maintain it • has a way to port games from the original PE

As you can see, I believe there are sufficient issues with the current solution to warrant an alternative.

The following sections focus on documenting what the studied software are and some of the useful information I learned about them.

2 Research subjects

2.1 RPG Maker XP

According to its Wikipedia article and its official website, RPG Maker XP :

- was developped by Enterbrain and released on July of 2004.
- runs on Microsoft Windows.
- uses the Ruby programming language.
- allows its users to create their own role-playing game with powerful dedicated tools.

This is only one instance among the long line of RPG Maker releases, some of which were released on other platforms, including consoles. Probably due to its early popularity, new projects continued to use this version way after it was superseded by newer ones.

2.2 The Pokemon Essentials project

According to its wiki fandom page and another wiki article, Pokemon Essentials :

- "a collection of gameplay-altering original code designed for use in an RPG Maker XP game".
- is a RMXP project, made to be the basis for a game.
- was forked from Flameguru's Pokémon Starter Kit.
- was developped by Peter O. (2007-2010) and Maruno (2011-2017)
- was last updated in October of 2017, when it was hit by DMCA takedown notice.
- is the source of a fair proportion of Pokemon fan-made games.

2.2.1 Comments

Programming with RPG Maker XP and Pokemon Essentials is complex, and is typically done in very small teams of amateurs, not using collaborative source code management tools like *git*. This results in ambitious project requiring years before they can be released and are hard to maintain.

I believe this is the reason most projects never see a release, and the ones that do are buggy.

2.3 The Pokemon Uranium project

According to its Wikipedia page, its official website and its Fan-made Wiki website, Pokemon Uranium :

- is a fan-made game
- uses the RPG Maker XP engine and Pokemon Essentials
- has unique region, creatures, plot and quests
- was developped by a small team (2 main devs) for about 9 years and released in August 2016.
  - Game designer and developer : JV12345 (aka. ∼JV∼)
  - Game developer and creative director : Nageki (aka. Involuntary Twitch)
- The development team was hit by DMCA takedown notices shortly after release. They stopped development in September 2016.
- Fan-made website and following game updates are community efforts.

3 Research

3.1 Methodology

Here are the high-level steps I took to study and experiment with RMXP, PE and PU :

1. Installing a working copy of RMXP
2. Opening the PE project

   This helped me understand how the project is structured, where the assets are located and how the scripting language works. Most of my experimentation started from it.
3. Opening the PU project

   This helped me understand how developers implemented new features and edited PE to suit their needs.

The objective was to understand how PE and its derivative works function and how features are implemented. I also needed to identify issues and come up with solutions to them, determine the specifics of the software I would implement in the next step of this project and document my findings.

3.2 Technical findings

3.2.1 Structure

PE and PU have a similar structure :

○ Root	Contains compiled game files and libraries, plus a few useful programs, some probably made by Pokemon Essentials developers.
↳ Audio	Contains the musical assets of the game, typically MIDI and OGG files.
↳ Data	Contains the logic of the game and most of its data.
↳ Fonts	Contain fonts used for displaying text.
↳ Graphics	Contains the graphical assets of the game, typically tile sets, in a well organized folder structure.
↳ (*PBS*)	(Facultative) Contains human-readable data to be compiled, like items, species, types, dialogue (and translations), etc.

3.2.2 Digging for information on PU

Here are some of my findings :

- Fans have been able to patch bugs and update the game for some time. I suppose there is a small group of people with access to the project's source code and files that have been maintaining it and its online services.
- The game was mostly complete when released. Fans were able to add elements to push it farther towards completion but there still are some hanging threads, particularly in the post game (after the main plot is over).
- I wasn't able to identify the used versions of RPG Maker XP or Pokemon Essentials, but I determined this wouldn't significantly impact on working on it.
- Despite efforts to patch it, the game has a long list of unresolved bugs and game breaking/crashing situations.
- GitHub user *acedogblast* launched a re-implementation project on the Godot game engine in early 2019. As of writing, he states that : "Only a limited portion of the game is playable currently."
- Download links are hosted on Reddit. Current version is `1.2.4` and was released on October 29th 2018.

3.2.3 Data representation

The data that constitutes a PE project is diverse :

Category	Description	Storage
Scripts	Logic of the game, classes, ect	`Scripts.rxdata` file
Objects	Files containing sets of class instances, grouped per file	`dat` and `rxdata` files
Maps	Contain map representation, one per file	`MapXXX.rxdata`
Dialogue	Contain dialogue	TODO

Findings :

- Data representation isn't human-readable. I found `rxdata` and `dat` files are either marshalled data or structured binaries. These can contain RGSS code or serialized objects or lists of objects.
- All `rxdata` and most `dat` files begin by the same two bytes, which is probably the magic number/signature for RPG Maker XP files : $\backslash x04$ $\backslash x08$

  Note that usually, signatures are 4 bytes long.
- I haven't been able to find any built-in way to export data, instead resorting to external tools. I have tried a few tools found on github or forums.

  This means I will either have to reverse-engineer them and build my own tool (probably in its native language Ruby to take advantage of the module/class definitions I may be able to find in the code) or find a way to code my own extraction scripts within RMXP and execute them.

3.3 Non-technical findings

Here are some of my research findings that doesn't fit in the previous section.

- The projects I researched have a particular characteristic that I wasn't execting, but makes sense : They are personal side-projects from (mostly) individual developers.

  With this in mind, it is no surprise that they are very long running projects that took years to release and displayed other properties I used to find odd :

  - Closed source projects : Because they were very personal.
  - Don't use code source management : Because there's no need, they are 1-2 developer personal projects that happened to be eventually released. Perhaps developers didn't consider the advantages of using one, or one wasn't available or familiar to them.
  - Buggy projects (years after release) : With so few maintainers and such code complexity, patching bugs is a challenge.

  As if the challenges facing such endeavors weren't great enough already, they were struck with the threat of legal repercussions on their authors, forcing these to abandon their creation. I can't help but having sincere sentiments of admiration for their work and sadness for the tragic destiny of their projects.

Fortunately, we can learn from the failures of the past how to better go forward. It is my firm belief that there is a solution that solves the challenges that brought these projects down.

3.4 Extracting data

In order to decouple PE from its original code base and RGSS, it became obvious that all useful data had to be extracted from the project.

Since `Audio`, `Graphics` and `Fonts` directories contain already-accessible data, they didn't require any extraction efforts.

See section *Data representation* for categories of data that were identified to need extraction.

3.4.1 Scripts

What I found :

- `Scripts.rxdata` is the second largest `rxdata` file of the project, and contains over 120k lines of code.
- The script is unique : it is divided in named sections (that appear in RMXP's editor on the left hand side) that allow to manage its colossal line count.
- Like other compiled files, its content are almost impossible to read outside of RMXP.

I spent quite a lot of time working on a python script able to decrypt `rxdata` files, and this one in particular, but didn't have much success, so I tried looking elsewhere.

After some more digging, I found a functional utility that was created for the purpose of editing/extracting the scripts from this file : Gemini Editor.

Here are links to a forum post about it and a github repository.

With it, I was able to extract Pokemon Essentials scripts (see '`PE_scripts`' directory).

It proved invaluable to be able to explore/search the whole script base using a more powerful code editor, saving lots of time from my attempts to create my own scripts to export data.

4 Remarks

4.1 Contact

4.2 Privacy Policy