PoGER*

Vision Document

Version *0.1*

Revisions

Date	Version	Description
June 28th 2020	0.1	Initial draft
N/A	1.0	First Release Placeholder

*pending name change

Contents

1 Introduction

1.1 Purpose

This document outlines the vision for the PoGER[1] project. This includes, but is not limited to :

- State the general idea
- Define high-level features
- Identify user categories and their respective needs
- Explore risks and constraints

1.2 Scope

The scope of this document is limited to this project only. Said project has the following dependencies :

- RPG Maker XP - The platform on which both following projects run
- The Pokemon Essentials project - The Pokemon "game engine" on RPG maker XP
- The Pokemon Uranium project - Game built on top of the Pokemon Essentials project

  This was chosen as the final test subject for PoGER.

1.3 Definitions, acronyms and abbreviations

- PoGER[1] - **Po**kemon Essentials **G**ame **E**ngine **R**ecreation
- Game Engine Recreation - In this instance, the process of creating a game engine, in order to play on it games that were developed for another one.
- RPGXP - RPG Maker XP
- PE - Pokemon Essentials
- Fangame - Fan-made game, or a game that was made by one or multiple people not being employed for that purpose.
- Dev-time - Quantity of time available to developer(s) to implement this project.

1.4 Preamble

The first part of this project was a research stage, as a result of which the decision to launch the development of PoGER was made. Most choices about its structure, content and implementation can be traced back to that stage.

At the time of writing, a document detailing the initial research stage doesn't exist yet, but one is planned to be produced shortly.

[1] pending name change

2 Positioning

2.1 Problem Statement

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

2.2 Related problem

This is specific to fangames made using PE.

The problem of	fangames using PE typically infringe on the Pokemon IP
affects	anyone willing to code/coding a fan-game (possibly also players)
the impact of which is	fangame creator typically face legal threats and/or actions
A successful solution would be	to keep a strict position of academic game engine recreation project, with no aim to infringe on IP or provide direction to do so

2.3 Product position statement

The PoGER project aims to provide a PE implementation that solves the identified problems. The author states that he hasn't been able to identify any similar project.

A secondary objective would be to code a tool capable of helping to port an existing PE-based project.

3 User Descriptions

3.1 User summary

This may be too straightforward.

Name	Description	Role
Fangame maker	Primary end user	Uses PoGER as engine for their game
Fangame player	Secondary end user	Plays a game running on PoGER
Maintainer	Contributes code to the project	Add functionality, fix bugs, code optimization, etc

3.2 User environment

End users use a number of OS platforms. Targeted platforms are :

- Windows (x86)
- GNU/Linux (x86)
- Android (ARM)

Other platfroms may be implemented, like iOS, OSX, PSVita, etc

Name	Typical platforms
Fangame maker	Windows, Linux
Fangame player	Windows, Linux, Android, etc
Maintainer	Windows, Linux

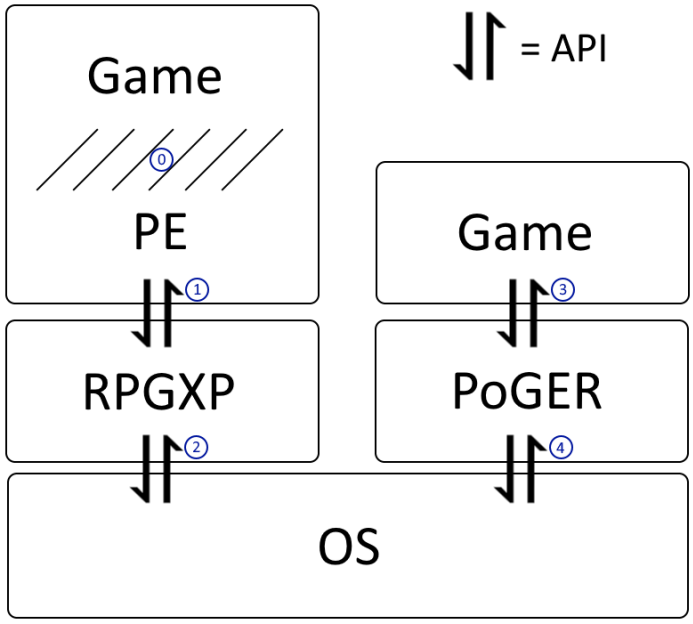
All targeted platfroms feature online support, which should be leveraged by "nice to have" features.

Touch input (mouse, finger) support looks like a must-have feature.

4 Product Overview

4.1 Product perspective

RPGXP + PE vs PoGER :



Notes :

0 - Non-standard, no separation between "game engine" and game code

1 - Vendor-specific, Ruby language

2 - Windows-specific, close source

3 - Standard, clear boudary between game engine and game code

4 - Platform adaptable, open source, documented API

The number of APIs remain constant, but the clearer and more focused structure should allow for better performance, wlĥile bringing desireable features like multi-platform support.

4.2 Summary of capabilities

Comparisons are made with regard to RPGXP + PE.

User benefit	**Feature(s)**
Create a game	An end user can create its own game easier, and with less technical knowledge.
Focus on content	A game creator doesn't have to understand PoGER implementation details.
Ease of collaboration	A PoGER game's structure is modular, optimized for collaborative work.
Better performance	With less overhead, an end user experiences less lag and better battery life.
Play anywhere	An end user can play on any device which platform is supported.
Online features	Online features, like multiplayer, open up possibilities for developers and players.
Add functionality	This project is open source. Maintainers and game creators can add functionality they need and everyone benefits from it.
Port a game	It should be possible to port a RPGXP+PE game to PoGER.

5 Product Features

5.1 Similar functionality to PE

As a port of PE, this project should implement similar features and display similar functional behaviour.

5.2 Easier to develop for

A game running on PoGER should've the following properties, interesting for developers :

- Clear structure, improving collaborative workflow
- Separate from game engine
- Online capabilities
- Platform-agnostic
- Documented API
- Doesn't require using proprietary/paid software

5.3 Similar functionality to PE

As a port of PE, this project should implement similar features and display similar functional behaviour.

5.4 Licensing

This is an open-source project, inline with free software values. A licensing standard like GNU GPL should be used.

6 Constraints

6.1 Limited dev-time

This project's initial development is subject to strict time constraints of a few weeks only, with one active developer only.

6.2 Cross-platform

Design decisions are to be made with cross-platforming in mind. For example, the choice of programming language is critical for their ease of use and platform compatibility.

6.3 Original language

PE was written in RPGXP-specific Ruby, which means none of the source code can be reused. A considerable amount of time will be necessary to re-implement everything.

6.4 Limited resources

PoGER targeted platforms may have limited hardware resources (storage, RAM, CPU cores and speed). How much of each is used is to be examined to establish a "*minimum configuration*" and a "*recommended configuration*".

6.5 Usability

An end user should not face great usability difficulties, in terms of setting up PoGER, launching it or normally using it.

6.6 Standards

Decisions about setting standards, documentation, etc should be made in a straightforward, non-obtuse manner, while being strict enough to be useful without being obstructive.

7 Risks

7.1 Risk list

Possible risks to the success of implementation include, but are not limited to :

- Lack of dev-time
- Unclear or changing goals/requirements of the system
- Non-friendly system
- Bugs or similar difficulties, which resolution takes significant dev-time
- Issues setting up an online infrastructure for online capabilities
- Lack of long-term interest. After initial stage, this project could fail to draw interest toward further development.

8 Documentation Requirements

8.1 Release notes, README file

This project is destined to be hosted on a Git-based platform, therefore some form of release notes are nice-to-have's.

Release notes :

- Is to be written for each major and minor release of the system.
- Should contain a list of changes : additions, removals, changes, bugfixes, etc
- Are intended for all end users, therefore should be written *concisely* and with *clear* (but not too complex) language

README :

- Should exist on the root of the file structure
- Should be Markdown-formatted for better readability
- Should contain basic information about the project (including links to additional resources), enough for any user to get started
- Should contain disclaimers about the nature of this project

8.2 Documentation

An offline documentation is to be maintained in the `Documentation` folder of the project, both in an editable form and a PDF form. This documentation should be sufficiently structured and complete for any user to be able to answer most common questions and represent most useful details.

Separating documentation into multiple parts seems appropriate :

- Installation guides
- Primary user documentation
- Secondary user documentation

Effort should be put to avoid duplicated and outdated informations in these docs.

In combination with READMEs and release notes, it represents a sufficient source of information

9 Others

9.1 Buzzwords

- open-source
- free software
- game engine (recreation)
- cooperative
- multi-platform
- integrated
- online capabilities

9.2 Legal disclaimer

The original author of this project asserts that this is a purely academical endeavor, with no ambition to release this work as a commercial product or part of one. He cannot be held responsible for any use of this project's code or resources outside of its intended purpose.

The author does not condone trademark infringement or any other unlawful act that may involve this project in any way.