RPGMaker et la gestion de projets

Xavier Van de Woestyne

Juillet 2015

RPGMaker est un logiciel que j'apprécie beaucoup pour plusieurs raisons. Premièrement, il permet à des enfants de réaliser leurs rèves… créer des jeux-vidéos (et ça… je trouve ça foncièrement cool!). Malgré une accessibilité incroyable, il n'en reste pas moins customizable. Certains projets sont bluffants tant ils sont personnalisés (à tel point que le moteur initial en devienne invisible). Et l'outil est extensible au moyen du langage Ruby. Cependant, bien que permettant de créer des projets sérieux, au delà de l'usage d'un langage moderne, le flux de travail est souvent archaïque. L'idée de cet article (très aimablement suggéré par **Pierre Ruyter**) est de proposer formellement quelques pistes sur ce qui a été mis en place lors du développement de projets conséquents avec RPGMaker et effleurer l'agilité au sein d'une équipe utilisant ce logiciel. Cette série de petite présentation ne donnera pas de tutoriel explicatif détaillé par outils mais tâchera d'offrir des liens plus exhaustifs !

Note préalable

Il est évident que pour beaucoup, RPGMaker n'est rien de plus qu'un jouet. Pour ma part, j'ai décidé d'y voir un outil fiable de prototypage de jeux >d'aventures, me permettant de ne pas devoir créer d'éditeur.
De plus, son extensibilité par Ruby permet d'appréhender les bases du designs de jeux (d'aventures) en terme d'architecture de code, les bonnes pratiques et introduit aussi un lot de challenges assez amusant. Pour ma part, RPGMaker à été un vecteur de progression indiscutable et c'est pour ce fait que j'en parle sans aucune honte sur mon blog.

Mise en contexe

Historiquement, RPGMaker ne permet pas l'extension au moyen de scripts Ruby, ce qui fait que les gens ayant de l'expériences dans l'usage du logiciel

n'ont pas spécialement d'expérience en tant que programmeurs. Et que les programmeurs ayant joint le projet plus tard, n'ont pas spécialement trouvé de public adéquat pour la mise en place de *workflow's* typiquement programmeurs. De ce fait, alors que l'éco-système du logiciel s'enrichit de jours en jours, le flux de travail mis en oeuvre par la communauté (francophone, car je ne connais qu'elle) est généralement archaïque. Au cours de cet article, je vais proposer une collection d'outils et de méthodologies que j'ai eu l'occasion de mettre partiellement en place lors du développement de RME et de son éco-système et qui nous a relativement beaucoup aidé. J'évoquerai aussi des stratégies que nous n'avons pas mise en oeuvre dans RME.

Les problématiques

Il existe plusieurs problématiques liées à la construction de projets en collectivité. En voici une liste (surement non exhaustive, je m'en excuse) :

- Partage de documents;
- partage du projet;
- séquence de travail sur des composantes du projet.

Par exemple, on peut imaginer que certaines personnes décident de se servir de courriels pour transmettre aux autres membres de l'équipe la nouvelle mise à jour du projet.

Les fausses bonnes idées chronophages

Dropbox, ou encore un serveur FTP peut sembler être une bonne approche pour partager aux membres du projet facilement. Cependant alors que Dropbox introduit des soucis sur le partages de projets (lorsque deux personnes modifient un projet simultanément, par exemple) le FTP ralentit énormément le cycle de partage d'une application. En effet, imaginons un projet de plus de 200MB, chaque partage entrainerait des upload's de plusieurs minutes (parfois proche d'une heure complète).
De plus, ces solutions occultent totalement la notion de versionnement ou encore de journalisation des modifications. Une idée serait de créer des *changeslog*, cependant, je vous laisse imaginer le temps perdu à notifier dans la bonne sémantique, les modifications apportées au projet. De plus, si le changelog n'est pas exhaustif, il peut arriver très souvent de ne pas prendre la décision de télécharger la dernière version du projet (qui aurait tout de même été modifié).

Des services à la rescousse

Depuis quelques années, il existe une collection variée de services répondant à beaucoup de problématiques. Ces services peuvent être utilisés gratuitement (et

généralement, étendu en services payants pour répondre à certaines contraintes). Dans cet article, nous survolerons quelques un de ces services en vue de créer un cycle de travail efficace!

Petit interlude sur l'open-source

Généralement, une des plus grosse limite des services qui seront proposés ici porte sur la visibilité des actions et des offres. En général, pour garantir la privatisation, il est impératif de payer.
De mon point de vue, concevoir en équipe, un projet ouvert est un gain d'argent indéniable et permet à autrui de voir quels sont les mécanismes/tactiques mises en oeuvre lors du développement d'un jeu. Je suis donc assez favorable à la diffusion massive de son travail, laissant à tout le monde le loisir de constater ce qui est fait ! Après je comprendrai totalement le sceptisissme de certains et ceux désireux de rester dans le confort de la non diffusion de leur projet avant sa fin, n'auront sans aucun doûtes, aucun scrupules à payer quelques dollars pour rendre leur service fermé !

Git (et Github.com)

Il serait impossible de commencer cette présentation sans parler de Git et de Github, qui auront été pour RME, un bienfait impressionnant.
Git est un logiciel (inventé par Linus Torvalds et concrétisé par lui plus une grande collection de contributeurs) permettant de gérer (sous forme de versions) l'historisation de fichiers. Même si Git est pensé pour la gestion du code source, il est tout à fait possible de s'en servir pour versionner, en plus, des fichiers binaires. Il a été pensé pour le travail collaboratif et permet, en plus de manipuler des branches de développement pour réduire au mieux les collisions. Couplé avec Github.com, (ou Bitbucket, des interfaces web pour Git, qui offrent la possibilité de faire des projets privés gratuitement), vous bénéficiez d'un pannel d'outil adéquat pour diffuser vos ressources et offrir un BugTracker ainsi qu'une liste de tâche à effectuer.

Liens utiles

- Le tutoriel de OpenClassRoom, assez complet;
- Un site très complet.

La publication de projets RPGMaker

Git se dote de fonctionnalités, comme par exemple le fichier `.gitignore` permettant d'occulter certains fichiers de la liste des fichiers à versionner. Une bonne pratique serait d'ignorer les fichiers statiques du projets. Par exemple le

`Game.exe`, le `projet.rvproj` et éventuellement les données de `Data/` changeant peu et n'ayant pas d'incidence sur le bon déroulement du projet.

Manipuler ses tâches

Même si nous verrons plus bas qu'il existe des outils agiles pour manipuler ses tâches, en utilisant Git avec le site Github, au moyen d'issues et de milestones (un regroupement d'issues articulées autours d'une étape dans le projet), il est possible de construire une véritable *todoo liste*, lié à des personnes.