PlayN une librairie multi-plateforme pour développer des jeux

Quel développeur n'a jamais rêvé de développer son propre jeu vidéo ? Je pense même que la majorité des développeurs s'y sont risqués, que ce soit à leurs instants perdus, ou lors de projets d'étudiants. Toujours est-il que ce rêve d'enfant hante sûrement une majorité d'entre nous.
Nous avons aujourd'hui la chance de vivre une véritable révolution du monde vidéo ludique grâce à l'explosion du marché des smartphones et des tablettes. Le marché du jeu vidéo est en train de changer du tout au tout. Là où il y a quelques années il fallait débourser plusieurs dizaines d'euros pour acheter le moindre jeu vous pouvez maintenant trouver de véritables perles à moins de 1 euro. Les raisons de cette baisse de prix sont multiples, tout à d'abord le nombre de client potentiels à explosé, les temps de développement ont été réduits et de nouveaux mécanismes de paiement sont apparus (publicité, micro paiement in-app, …). Seul point noire à ce nouvel écosystème : la multitude de périphériques existants sur le marché et du coup la segmentation du marché. Cette diversité contribue bien sûr à dynamiser et à étendre le marché mais il rend le travail des développeurs très compliqué. Il leur faut adapter leurs développements au plus grand nombre d'appareils (même s'ils se contentent en général de l'adapter aux plus vendus). Heureusement il existe aujourd'hui plusieurs frameworks qui permettent de développer facilement un jeu qui tournera sur un grand nombre de plateformes. Je vous propose aujourd'hui de découvrire rapidement l'une d'entre elle : PlayN [1]
Présentation
Ce framework est issu du travail de Google, elle permet l'écriture d'application directement compatibles avec Java, Android, HTML5, Chrome, iOS et Flash. L'avantage d'une telle librairie est évidemment la simplicité de portage du code d'une plateforme à une autre. Si vous avez bien travaillé c'est en général transparent.
Afin de travailler avec PlayN il vous faudra :
Eclipse
Java 1.6+
Un client Git
Maven 3+
Idealement un environnement de développement Android et/ou iOs si vous voulez développer pour mobiles.
L'architecture standard d'un projet PlayN est assez simple. Elle repose sur Maven. Vous avez un projet principal composé de modules. Dans ces modules vous avez un module "core", qui contient 90% de votre code, et un module par plateforme cible qui contiendra le code spécifique à chacune. En règle général vous n'aurez pas trop à y toucher, PlayN s'occupe de (presque) tout.
L'avantage de l'environnement de développement de PlayN est son côté multi-plateforme. En effet comme PlayN est capable de faire tourner votre code dans un environnement Java vous pourrez tester/debugger votre code directement sur votre ordinateur de développement. Fini les galères de déploiement sur un appareil connecté en USB, la lenteur du remote débuggeur. Bien entendu cela ne vous dispensera pas totalement de tests sur les appareils cibles, mais au moins le gros du travail est effectué sur une machine rapide. Autre avantage non moins important : vous pourrez utiliser tous vos outils habituels pour tester/debugger/profiler vos développements.
Le code
Le développement d'un jeu avec PlayN commence par une classe implémentant l'interface Game. Cette classe contiendra toute la logique de votre jeu dans seulement deux methodes.
En effet lorsqu'un jeu PlayN est demarré une sorte de boucle infinie se lance :
1
Vous l'aurez sûrement compris, la méthode update s'occupera de mettre à jour l'état des objets de votre jeu, et la méthode paint dessinera à l'écran ce qu'il y a dessiner.
Une troisième méthode doit être implémentée : <code>public int updateRate();</code>. Cette méthode défini avec quelle période (en ms) vous voulez appeler la méthode update. Attention cela ne définit pas la fréquence des appels à paint. Paint sera toujours appelée aussi souvent que possible. paint et update prennent toutes deux en paramètre un float contenant le temps exact écoulé depuis le dernier appel. Dans paint ceci vous

permettra d’interpoler l’état de vos objets entre deux appels à update, et dans update à savoir combien de temps s’est exactement écoulé depuis le dernier appel. Si vous ne faites pas d’interpolation dans paint vous risque de ne pas avoir d’animations fluides. En effet si vous faites du code simulant la chute d’une balle, sans interpolation cette dernière ne sera dessinée qu’aux positions calculées dans la méthode update. Imaginez un balle de 20 pixels de large qui se déplace de 50 pixels à chaque appel à update toutes les 50ms. Dans ce cas vous aurez une balle qui “saute” de 50 pixels en 50 pixels.

Architecture graphique

PlayN étant une libraire destinée à développer des jeux son API sert essentiellement à gérer des graphismes. Pour cela elle découpe l’affichage en couches (Layers). L’API définit un layer de base (root) qui contient une hiérarchie de layers. Les layers sont dessinés de manière automatique par l’API. Les layers peuvent être manipulés de manière simple par des transformations affines (rotations, déformations, déplacements, …).

Il existe plusieurs type de Layers dans PlayN :

ImageLayer : un layer représentant une image chargée à partir d’un fichier (PNG, BMP, …)

Canvas : un layer représentant une image où vous dessinez à la main (ou juste dessiner à la main)

ImmediateLayer : un layer dont le rendu est fait directement dans le framebuffer

SurfaceLayer : un layer dont le rendu est fait off-screen puis copié dans le framebuffer

Les contrôles

Tout jeu se doit d’interagir avec l’utilisateur. Pour cela PlayN propose plusieurs mécanismes de contrôles. L’API sait parfaitement gérer un clavier, une souris ou un écran tactile. Pour cela rien de plus simple, il suffit d’utiliser les objets de la classe PlayN correspondants à ce que l’on veut faire : PlayN.pointer(), PlayN.mouse(), PlayN.keyboard() et de jouer avec. L’API se veut assez simple et vous trouverez facilement ce que vous cherchez. Sachez juste que les points d’entrées aux différentes fonctions de l’API sont en général dans la classe PlayN qui n’est grosso-modo qu’un gros réservoir de méthodes statiques.

Conclusion

Si vous voulez réaliser votre vieux rêve d’enfant je vous conseille vivement d’aller jeter un coup d’oeil à cette librairie vraiment bien faite. En quelques heures vous pourrez obtenir vos premiers résultats et peut être même un mini jeu.