Avant-propos

Si vous lisez ce livre, c'est que votre objectif est sûrement de créer un jeu vidéo, c'est-à-dire d'ordonner à l'ordinateur ou à la console d'effectuer un certains nombres de tâches.

La programmation de jeu vidéo

Lors d'une utilisation quotidienne d'un ordinateur ou de votre console, vous n'avez nul besoin de programmer. Si vous devez faire une recherche sur l'Internet ou que vous voulez jouer à un jeu, vous vous contenterez d'utiliser un programme écrit par quelqu'un d'autre ; et ceci est tout à fait normal, nul besoin d'être plombier pour prendre un bain !

Définition

Un programme informatique a pour but d'indiquer à un ordinateur la liste des étapes nécessaires à la réalisation d'une tâche. La programmation est le nom donné au processus de création d'un programme.

Pour certains, la programmation constitue une véritable passion, pour d'autres, c'est un moyen pratique de donner une solution à un problème… Dans tous les cas, force est de constater que la programmation devient un hobby et pénètre dans l'univers du grand public. Pierre angulaire de la science informatique, c'est une activé fascinante qui attire et motive de nombreux étudiants vers de réelles opportunités de travail, qu'il s'agisse de l'univers du jeu ou non. Toutefois, elle n'en reste pas moins un domaine complexe et de surcroît en constante évolution.

Mais la passion n'est pas le seul ingrédient requis pour réussir ses programmes… On ne s'improvise pas spécialiste en informatique ! En effet, la création d'un jeu n'est pas seulement affaire de programmation : il faut aller au-delà et s'attaquer à la partie graphique, audio et bien évidemment au *gameplay.*

Les concepts qui seront abordés dans ce livre vous donneront de solides bases, mais ne soyez pas déçu si vos premiers jeux n'égalent pas les réalisations sophistiquées auxquelles vous êtes habitué. C'est une expérience incroyable que de voir une de ses créations prendre forme, et même si le challenge est parfois difficile, la récompense est toujours très gratifiante.

Code intelligible, code machine

Un ordinateur ne comprend que des instructions très simples :

1. Récupérer le contenu d'un emplacement mémoire.
2. Lui appliquer une opération mathématique basique.
3. Déplacer le résultat vers un autre emplacement mémoire.

- En plus de diviser à l'extrême chaque tâche, pour être compris directement par l'ordinateur, vous devez lui parler en binaire, c'est-à-dire en une succession de 0 et 1. Imaginez donc la complexité du code machine qui se cache derrière le démineur de Microsoft…
- Ce type de code n'étant pas du tout intelligible par un humain, il a donc fallu créer des langages possédant une syntaxe plus proche de notre langue ainsi que les outils nécessaires à la traduction du code écrit dans ces langages vers le code machine correspondant. Ces derniers sont généralement appelés compilateurs.
- On distingue plusieurs types de langages : ceux dits de bas niveau et ceux de haut niveau. Plus un langage est de bas niveau, plus il se rapproche de la machine, c'est-à-dire que sa syntaxe est moins innée, que la gestion de la mémoire est plus difficile, etc. Prenons deux exemples. L'assembleur étant un langage de bas niveau, il faut traiter directement avec les registres du processeur, et il implique une bonne connaissance de l'architecture système. À l'inverse, le Visual Basic est un langage plus abordable qui n'est pas soumis aux mêmes contraintes que celles que nous venons de citer.
- Il faut surtout garder en tête qu'un langage qui pourrait être classé de plus haut niveau n'est pas forcément plus facile à maîtriser qu'un autre. Tout dépend du programmeur, bien sûr, mais aussi du besoin : à cause de sa simplicité, le Visual Basic n'offre pas les mêmes possibilités d'optimisation que le C, par contre, il s'avère très pratique pour développer rapidement une application.

Les algorithmes

Un algorithme est l'énoncé d'une suite d'opérations constituant une solution à un problème donné. On peut présenter toutes les actions de notre quotidien sous la forme algorithmique. Par exemple, pour la cuisson des pâtes :

1. Saler l'eau.
2. Porter à ébullition.
3. Plonger les pâtes.
4. Mélanger pour éviter qu'elles ne collent au fond.
5. Égoutter.
6. Rincer.

Grâce à cet algorithme, vous pouvez aisément expliquer à quelqu'un la façon de cuire des pâtes, si besoin est.

Le langage algorithmique est un compromis entre notre langage courant et un langage de programmation. Ainsi, la compréhension d'une fonction d'un programme est plus aisée qu'en se plongeant directement dans le code.

XNA et son environnement

Il existe une multitude de langage de programmation et de bibliothèques qui peuvent être utilisés pour programmer un jeu vidéo. Comment faire le bon choix ?

Pourquoi choisir XNA ?

L'un des principaux critères qui peut motiver votre choix est la plate-forme cible. En effet, vous n'utiliserez pas forcément les mêmes outils pour créer un jeu pour Xbox 360 ou téléphone mobile. D'une manière générale, pour développer un jeu pour console, vous devrez utiliser un kit de développement adapté : la PSP possède son SDK utilisable en C++, celui de la Nintendo DS repose quant à lui sur le C.

Du côté des PC, vous pouvez programmer un jeu vidéo dans un peu près n'importe quel langage. En ce qui concerne la partie graphique du jeu, deux solutions s'offrent à vous : la première consiste à utiliser des bibliothèques de très bas niveau telles que DirectX, OpenGL ou encore SDL. La seconde possibilité consiste à utiliser un moteur graphique comme OGRE ou Allegro. Elles est particulièrement intéressante car elle permet de gagner beaucoup de temps.

XNA est une bibliothèque de bas niveau basée sur le framework Compact .Net dans son implémentation pour Xbox 360 (ou le lecteur multimédia Zune de Microsoft) et sur le framework .Net dans son implémentation pour PC.

Comprendre le framework .NET

- Le framework .NET (prononcez « dotNet »), est un composant Windows apparu dans sa version 1.0 en 2002. Depuis, Microsoft a sorti régulièrement de nouvelles versions. Avec le système d'exploitation Windows XP, ce composant était facultatif. Cependant la version 3.0 du framework, .NET est directement intégré à Windows Vista.

En détail

Voici récapitulées les années de sortie des précédentes versions de notre framework :

1.1 en 2003 ;

2.0 en 2005 ;

3.0 en 2006 ;

3.5 en 2007.

- Il dispose de deux atouts majeurs pour simplifier le développement d'applications web ou Windows : le CLR (*Common Language Runtime*) et les bibliothèques de classes.

- Le CLR est une machine virtuelle (bien que Microsoft préfère utiliser le terme *runtime*) utilisée pour exécuter une application .NET. Il possède, entre autres, un composant appelé JIT (*Just In Time*, c'est-à-dire juste à temps), qui compile du code MSIL (*Microsoft Intermediate Language*) vers du code compréhensible par la machine. Ainsi, tout langage disposant d'un compilateur qui produit du code MSIL (les spécifications techniques sont disponibles à cette adresse : `http://www.ecma-international.org/publications/standards/Ecma-335.htm/`) est exécutable par le CLR et bénéficie des possibilités offertes par la plate-forme. Il est donc possible de choisir un langage parmi un grand nombre (C#, C++, VB.NET, J#, etc.), le choix ne dépendant plus forcément des performances mais plutôt d'une affaire de goût. Le CLR comporte également une multitude d'autres technologies dont vous ne saisiriez peut-être pas l'intérêt pour le moment, mais que nous aborderons plus tard dans cet ouvrage.

MSIL

Langage ressemblant à de l'assembleur, MSIL ne comporte aucune instruction propre à un système d'exploitation ou à du matériel.

Le framework .NET met également à la disposition du programmeur plus de 2 000 classes utilitaires, qui lui permettent de gagner un temps précieux lors du développement. Ainsi, manipulation de chaînes de caractères, communication réseau, accès aux données sont choses faciles à réaliser. À chaque nouvelle version du framework, la bibliothèque de classes s'étoffe davantage et les fonctionnalités disponibles sont de plus en plus performantes.

XNA : faciliter le développement de jeu vidéo

Le framework XNA (*XNA's Not Acronymed*) est constitué de plusieurs bibliothèques .NET et permet un développement multi-plate-forme : les classes fournies par XNA permettent au programmeur de développer un jeu pour Windows puis de le porter très facilement pour qu'il soit utilisable sur Xbox 360 ou sur le lecteur multimédia Zune.

L'un des buts de XNA est de simplifier au maximum le développement de jeu vidéo. Par exemple, si vous avez déjà eu une expérience dans le développement avec l'api DirectX ou OpenGL, vous savez certainement qu'écrire l'initialisation de votre programme vous prendrait un certain temps alors qu'avec XNA tout est automatique. C'est précisément là que réside tout l'intérêt du framework : avec XNA, il vous suffit seulement d'écrire quelques lignes de code très facilement compréhensibles pour créer un jeu complet.

Bon à savoir

Soulignons également que le framework XNA est livré avec ce que l'on appelle des *Starter Kit.* Ces petits projets de jeu vidéo montrent les possibilités offertes ainsi que le niveau d'accessibilité du développement.

Officiellement, XNA ne peut être utilisé qu'avec le langage de programmation C#. En pratique, vous pouvez également réaliser un jeu avec XNA en VB.NET, mais vous ne pourrez pas utiliser tous les composants offerts par le framework.

Version

XNA 3.0 est disponible depuis le 30 octobre 2008, c'est sur cette version que ce livre se focalise.

C#, langage de programmation de XNA

Langage de programmation orienté objet à typage fort, C# (prononcez « C-Sharp ») a fait son apparition avec la plate-forme .NET. Il est très proche à la fois du Java et du C++. Ses détracteurs le qualifient souvent de copie propriétaire de Java.

Java

Très répandu dans le monde du logiciel libre, ce langage s'exécute lui aussi sur une machine virtuelle. À l'heure actuelle et selon des sondages qui paraissent régulièrement sur l'Internet, il s'agit du langage le plus populaire parmi les développeurs.

Tout comme le framework .NET dont il est indissociable, le langage C# est régulièrement mis à jour et se voit ajouter des améliorations syntaxiques ou de conception.

Choisir son environnement de développement intégré

Pour utiliser XNA ou, d'une manière plus générale, programmer dans un langage compatible .Net, vous aurez besoin d'un EDI (Environnement de Développement Intégré). Microsoft en propose toute une gamme comprenant :

- Visual Studio Express.
- Visual Studio Standard.
- Visual Studio Professional.
- Visual Studio Team System.

Chaque version vise un public différent, les versions Express (il en existe une pour le langage C#, une pour le C++, une pour le VB et une pour le développement web) sont gratuites et s'adressent au développeur amateur tandis que la version Team System est orientée pour le développement professionnel en équipe.

XNA 3.0 est compatible avec les versions de Visual Studio 2008. Dans ce livre, nous utiliserons la version Microsoft Visual C# Express 2008.

Vous connaissez maintenant tous les outils nécessaires pour commencer, alors bonne lecture et bienvenue dans le monde du C# et de XNA !

À qui s'adresse le livre ?

Ce livre s'adresse à tous ceux qui désirent créer des jeux pour PC, pour Xbox 360 ou pour le Zune sans avoir d'expérience préalable dans ce domaine ou même dans celui plus vaste de la programmation. En effet, nous y présentons les notions de bases du C# nécessaires à la compréhension de XNA.

Ainsi, ce livre vous sera utile si, étudiant en programmation, vous souhaitez découvrir l'univers du développement de jeux vidéo ; si vous travaillez au sein d'un studio indépendant ou en tant que freelance et que vous souhaitez vous former aux spécificités de développement pour Xbox ; ou si, tout simplement, vous êtes curieux de vous initier au développement de jeu et que vous avez choisi XNA.

Cependant, nous vous conseillons tout de même de vous munir d'un ouvrage sur le langage de programmation C# : ce livre ne constitue pas un document de référence sur ce langage, nous ne verrons ici que ce qui sera utile à la compréhension du framework XNA, et certaines facettes du langage seront mieux détaillées dans un ouvrage spécialisé.

Structure de l'ouvrage

Le **chapitre 1** présente les notions de base du langage de programmation C#, qui vous seront utiles dès le **chapitre 2** à la création d'une première application avec XNA.

Nous attaquerons les choses sérieuses dans le **chapitre 3** en apprenant à afficher de premières images à l'écran puis, dans le **chapitre 4**, nous apprendrons à récupérer les entrées utilisateur sur le clavier, la souris ou la manette de la Xbox 360. Ces notions seront mises en pratique avec la création d'un clone de Pong dans le **chapitre 5**. Le **chapitre 6** poussera plus loin les fonctions d'affichage d'images dans XNA.

Dans le **chapitre 7**, vous étofferez votre jeu en lui ajoutant un environnement sonore qu'il s'agisse des sons ou de morceaux de musique. Puis, dans le **chapitre 8**, vous découvrirez les techniques de lecture ou d'écriture de fichiers qui entrent en jeu dans les fonctionnalités de sauvegarde.

Dans le **chapitre 9**, vous vous écarterez un peu du monde de XNA pour rejoindre celui des sciences cognitives et plus particulièrement l'implémentation d'un algorithme de recherche de chemin. Le **chapitre 10** abordera également un domaine qui n'est pas propre à XNA : la gestion de la physique. Nous verrons donc comment implémenter un moteur physique.

Dans le **chapitre 11**, le dernier à utiliser des exemples en deux dimensions, vous découvrirez comment créer un jeu multijoueur avec XNA, qu'il s'agisse d'un jeu sur écran splitté ou en réseau.

Le **chapitre 12** propose une introduction à la programmation de jeux en 3D avec XNA. Pour terminer, dans le **chapitre 13**, vous apprendrez à réaliser des effets en HLSL.

Si vous n'avez jamais utilisé l'IDE Visual Studio, ou si vous souhaitez compléter vos connaissances, l'**annexe A** est consacrée à sa prise en main. L'**annexe B** vous donne des

pistes pour que vous puissiez pousser votre exploration de XNA au-delà de ce livre. Elle présente donc différentes sources d'informations disponibles sur le Web, ainsi que des méthodes de génération de documentation pour vos projets.

Remerciements

Je tiens tout d'abord à remercier Aurélie qui partage ma vie depuis un moment déjà et qui sait toujours faire preuve de compréhension lorsque je passe des heures scotché à mon ordinateur à coder encore et encore.

Merci également à mes parents qui ont tout mis en œuvre pour que j'accomplisse mes rêves et sans qui je n'aurais sûrement jamais écrit ce livre.

Enfin je remercie les éditions Eyrolles, et tout particulièrement Sandrine et Muriel qui m'ont accompagné tout au long de la rédaction de cet ouvrage.

Léonard Labat

Developper.xna@gmail.com