2

Prise en main de XNA

Connaissez-vous réellement les possibilités qu'offre le framework XNA ? Ce chapitre les présente afin que vous vous rendiez compte de quoi vous serez capable après quelques heures de pratique avec XNA.

Après avoir lu ce premier chapitre, vous serez en mesure de créer votre premier projet, de comprendre les différents éléments qui le composent et de le déployer sur une Xbox 360.

Télécharger l'EDI et XNA

Si l'EDI Microsoft Visual C# Express 2008 et le framework ne sont pas déjà sur votre ordinateur, voici la procédure à suivre pour vous en équiper :

1. Téléchargez Microsoft Visual C# Express 2008 en vous rendant à cette adresse :

   `http://www.microsoft.com/express/download/`

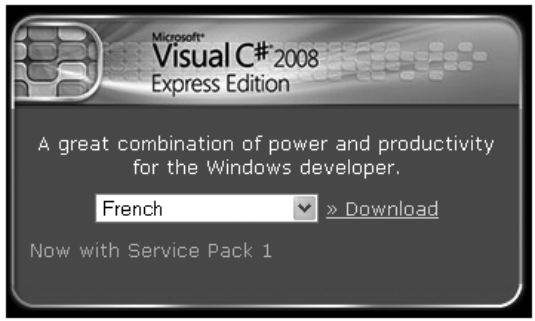

Figure 2-1

Téléchargement de Microsoft Visual C# Express 2008

2. Une fois le fichier téléchargé, exécutez-le pour démarrer l'installation.
3. Il ne reste plus qu'à télécharger et installer XNA en vous rendant sur le site officiel à cette adresse :

   `http://creators.xna.com/en-us/xnags_islive`

Partir d'un starter kit

En général, les *starter kits* sont des projets de jeux prêts à l'emploi et à compiler. Ils sont faciles à modifier et constituent une bonne base pour vos projets : vous pouvez analyser leur code source, en utiliser une partie dans vos jeux et le modifier.

Penchons-nous sur celui livré avec XNA 3.0 :

1. Ouvrez Visual Studio.
2. Cliquez sur Fichier puis Nouveau projet.
3. Dans la section XNA Game Studio 3.0, sélectionnez Platformer Starter Kit (3.0).
4. Validez.

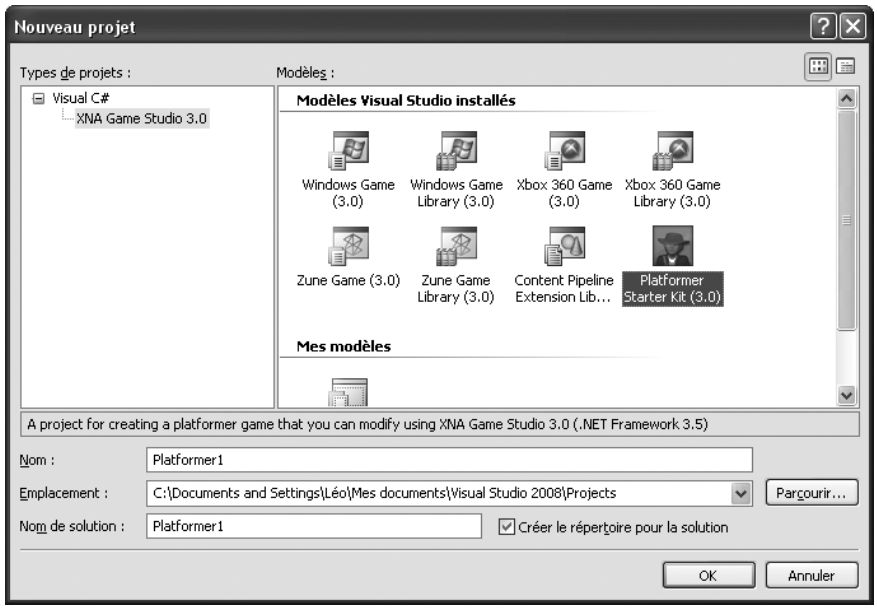

Figure 2-2

Création d'un projet basé sur un starter kit

Dans l'explorateur de solutions situé à droite de l'écran, vous trouvez trois projets : un pour Windows, un pour la Xbox 360 et le dernier pour Zune.

Zune

Zune est le lecteur multimédia de Microsoft, concurrent de l'iPod d'Apple. Depuis la version 3.0 d'XNA, il est possible de développer des jeux sur cette plate-forme.

Toutefois, sachez qu'à l'heure où nous écrivons ces lignes, Zune est commercialisé uniquement aux États-Unis et qu'aucune date officielle n'a été annoncée pour une éventuelle apparition sur le marché français.

Appuyez sur F5 pour lancer la compilation du projet sélectionné par défaut (ici, le projet Platformer).

Figure 2-3

Le jeu est agréable à jouer.

Le petit jeu qui s'ouvre alors est un bon exemple de ce que vous pouvez facilement réaliser avec XNA : afficher des graphismes, déclencher des sons et enchaîner plusieurs niveaux.

D'autres *starter kits* sont disponibles et téléchargeables sur le site Internet de la communauté d'XNA : *http://creators.xna.com/en-US/education/starterkits/*

Jeux de rôles, *shoot'em up*, jeux de course et puzzles : il suffit de jeter un œil à la liste des kits pour voir que les possibilités de créations avec XNA sont presque illimitées !

Nous vous conseillons de prendre le temps d'explorer plus en détail les kits, et notamment de regarder leur code. Vous reconnaîtrez sûrement les facettes du langage abordées dans le chapitre précédent, mais certaines parties du code vous sembleront au contraire obscures : ceci est tout à fait normal pour le moment. Cependant, revenir sur les kits plus tard peut

être intéressant et très instructif, ne serait-ce que pour comparer vos méthodes de programmation avec celle d'un autre développeur, ou encore pour savoir comment telle ou telle partie du jeu a été réalisée.

Partager ses projets

Les *starter kits* restent des projets assez simples et leur vocation est essentiellement didactique. La plupart d'entre eux sont basés sur des jeux en 2D. Mais rassurez-vous, il est tout à fait possible de réaliser des jeux en 3D avec XNA.

Pour vous en convaincre, il vous suffit de faire un petit tour sur le site de la communauté (*http://creators.xna.com/en-US/*) et de vous intéresser aux projets des autres membres. En effet, sur ce site, vous pourrez parcourir le catalogue des jeux développés par des amateurs, des développeurs patentés, voire des studios indépendants. Vous pourrez également visualiser des vidéos de présentation et même en acheter certains.

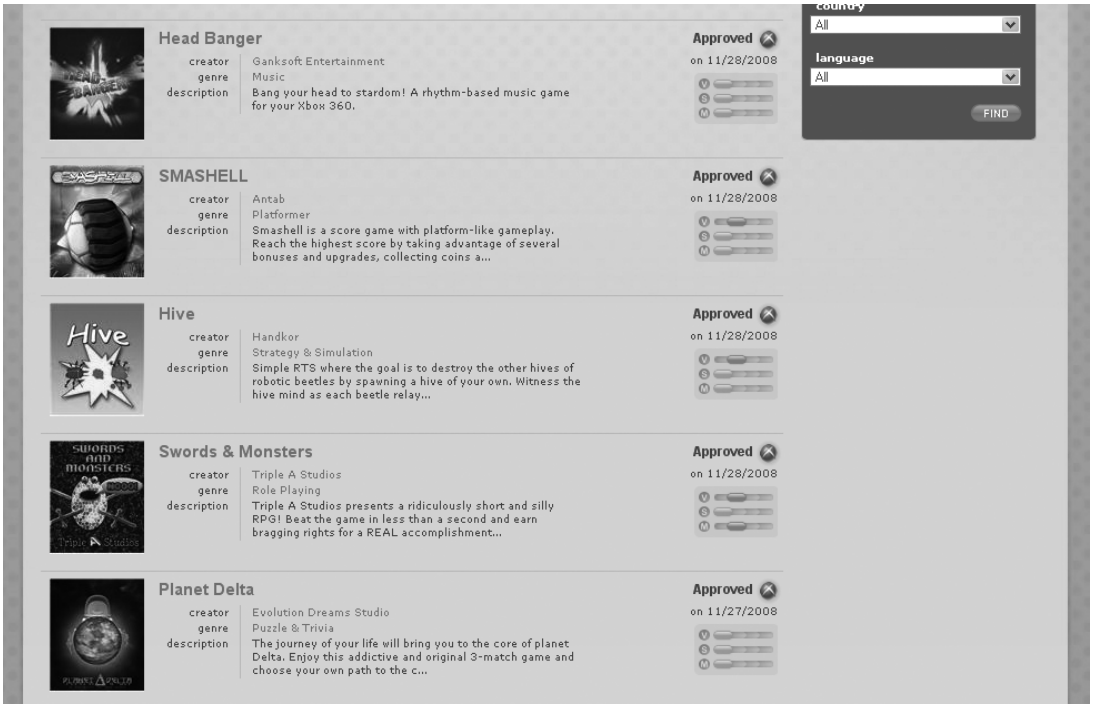

Figure 2-4

Le catalogue de jeux disponible sur le site de la communauté

Nous conseillons vivement de vous y inscrire et de participer aux forums de discussion, car c'est le meilleur endroit pour obtenir de l'aide sur XNA et, d'une manière générale, sur la programmation de jeux.

Encouragement

Au moment où nous rédigeons ce livre, il n'existe pas de lieu de rassemblement pour la communauté francophone. Cependant, nous espérons que l'anglais n'est pas une barrière infranchissable pour vous. En effet, vous ne pourrez pas y échapper (même si vous réussissez à vous procurer des ouvrages en français tels que celui-ci), et il y a de fortes chances qu'à un moment ou un autre vous deviez échanger avec un interlocuteur étranger.

Vous aurez sûrement envie de partager vos projets. Ceci est intéressant à plusieurs titres : vous pourrez ainsi présenter vos créations à vos amis, mais surtout, vous récolterez par ce biais les avis et conseils des autres développeurs.

Avec la sortie de XNA 3.0, la possibilité de vendre ses jeux sur le Xbox Live est apparue. Votre jeu est alors disponible sur le Xbox Live Market pour quelques centaines de points Microsoft. Les détails sont disponibles sur le site Internet de la communauté (*http://creators.xna.com/*).

Une autre solution pour faire connaître vos talents de développeur et vous frotter aux autres afin de progresser est de participer aux concours de programmation XNA. Citons par exemple l'Imagine Cup, organisé chaque année par Microsoft et possédant une catégorie intitulée *Game Development*, dont les finalistes gagnent plusieurs milliers de dollars (*http://imaginecup.com*).

L'architecture d'un projet XNA

Dans cette partie, nous allons d'abord nous intéresser aux différents éléments du framework, puis nous nous pencherons sur les différentes méthodes qui composent le cycle de vie d'un jeu vidéo sous XNA.

Structure du framework

Le framework XNA comporte essentiellement trois parties, chacune correspondant à une DLL (*Dynamic Link Library*, c'est-à-dire une bibliothèque de fonctions) :

- le moteur graphique (`Microsoft.XNA.Framework.dll`), qui contient tout qu'il faut pour gérer l'affichage dans votre jeu ;
- le modèle d'application d'un jeu (`Microsoft.XNA.Framework.Game.dll`), que nous détaillerons dans la section « Structure du code » ;
- et le *content pipeline* (`Microsoft.XNA.Framework.Content.Pipeline.dll`), utile à la gestion des ressources (texture, son, etc.) du jeu.

Les fonctions contenues dans ces bibliothèques font appel à des fonctions de DirectX de plus bas niveau, c'est-à-dire qu'à une ligne de code utilisant le framework XNA correspondent plusieurs lignes de code utilisant directement DirectX.

Dans Visual Studio, vous pouvez voir à quelles bibliothèques votre projet est lié dans la section References de l'explorateur de solutions.

Structure du code

Le déroulement d'un jeu sous XNA est le suivant : les méthodes `Initialize()` et `LoadContent()` sont appelées en premier puis, tant que le joueur ne quitte pas le jeu, les méthodes `Update()` et `Draw()` sont exécutées en boucle ; enfin, lorsque le joueur quitte le jeu, la méthode `UnloadContent()` est appelée. La liste ci-dessous détaille les différentes actions à effectuer dans chacune de ces cinq méthodes.

- **`Initialize`** – Comme son nom l'indique, c'est dans cette méthode que se font tous les réglages de base : instanciation d'un objet, chargement de paramètres, etc.
- **`LoadContent`** et **`UnloadContent`** – Si vous suivez le modèle d'application proposé par Microsoft, c'est ici que vous chargerez ou déchargerez vos ressources. Cependant, certains programmeurs ont tendance à réaliser ce travail avec la méthode `Initialize()`.
- **`Update`** – Cette méthode fait partie de la boucle de jeu. D'après le modèle d'application proposé par Microsoft, c'est ici que vous devrez effectuer toutes les opérations dites logiques, c'est-à-dire tout ce qui ne concerne pas l'affichage à l'écran.
- **`Draw`** – Cette dernière méthode, qui fait également partie de la boucle de jeu, est appelée à chaque fois que l'écran de jeu est mis à jour. Vous devrez donc y écrire uniquement du code utile à l'affichage.

Souvenez-vous que la séparation du code entre les méthodes `Update()` et `Draw()` n'est absolument pas obligatoire. Il s'agit d'une proposition de design faite par les créateurs d'XNA afin que les développeurs utilisant XNA puissent facilement récupérer des composants créés par d'autres et mettre les leurs à disposition (les composants seront abordés au chapitre 4). Nous suivrons cette recommandation dans ce livre, le modèle étant très simple à comprendre et le code créé très bien organisé de cette manière.

Créer un projet

Le temps est maintenant venu de débuter notre premier projet. Dans Visual Studio cliquez sur Fichier puis sur Nouveau projet. Sélectionnez Windows Game (3.0), puis validez.

Ouvrez le fichier `Game1.cs`. Vous reconnaissez l'architecture que nous venons de voir et, grâce aux commentaires, vous comprenez ce que fait notre programme à chaque appel de `Update()` et de `Draw()`.

Dans le fichier `Program.cs`, vous retrouvez la fonction `Main()` de notre programme. C'est ici que notre objet `Game1` s'instancie, et que le lancement du jeu a lieu via la méthode `Run()`.

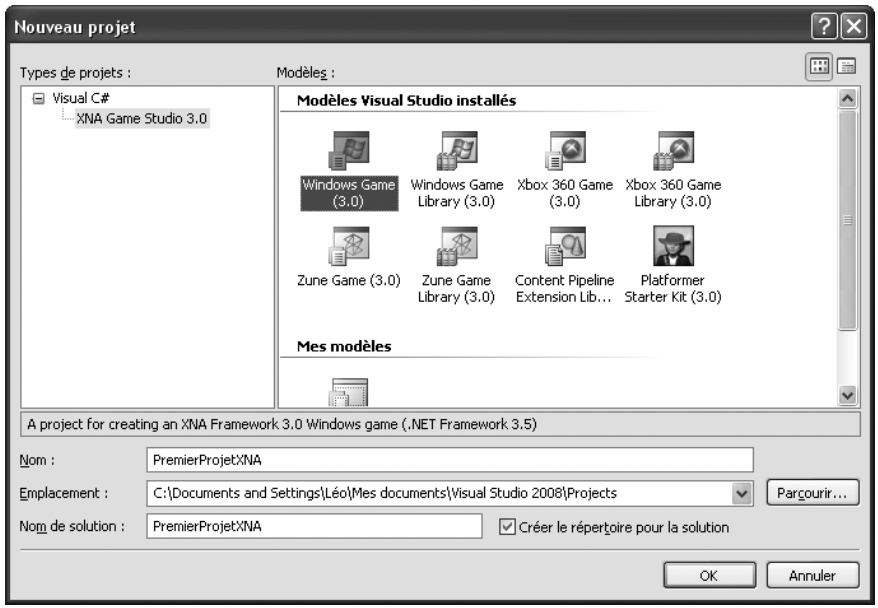

Figure 2-5

Création d'un projet pour Windows

En compilant notre programme, vous constaterez qu'il ne s'agit que d'une simple fenêtre avec un fond bleu. Si vous le lisez sur Xbox, vous aurez également la possibilité d'utiliser le bouton Back de la manette pour quitter.

À chaque appel de la méthode `Draw()`, la ligne de code suivante va se charger d'effacer puis de coloriser l'écran :

```
GraphicsDevice.Clear(Color.CornflowerBlue);
```

Attention ! `Color` n'est pas une classe mais une structure. La différence entre structure et classe est la suivante :

- une classe se manipule par une référence ;
- une structure se manipule par sa valeur.

Nous avons déjà rencontré d'autres structures lors du chapitre sur C#. Ainsi, les types `int` et `double`, pour ne citer que deux exemples, font partie des structures.

Voyons à présent comment modifier ce programme de base et changeons la couleur de la fenêtre. Pour cela, effacez l'argument passé à la méthode `Clear`, puis récrivez « `Color.` ». Lorsque vous tapez « . », une petite fenêtre s'ouvre, affichant tout ce que contient la structure `Color` (figure 2-6). Choisissez alors la couleur que vous désirez.

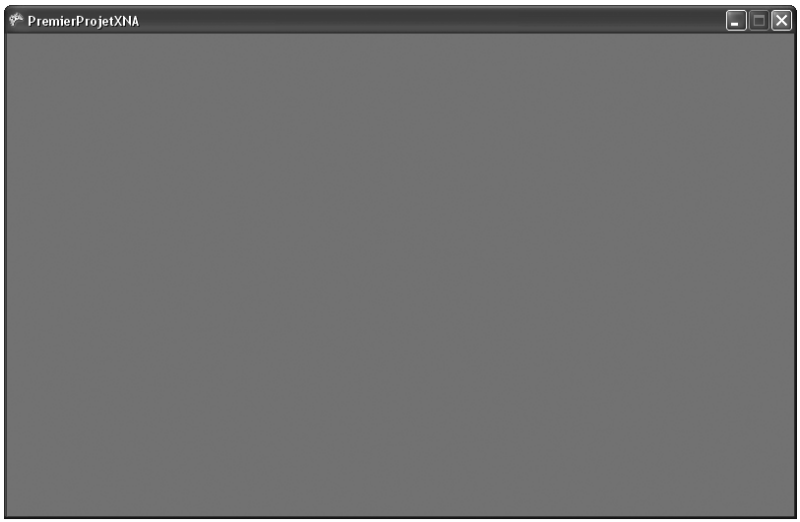

Figure 2-6

Une première fenêtre avec XNA

> **En pratique**
>
> Ce mécanisme s'appelle IntelliSense. Il s'agit du système d'autocomplétion de Microsoft qui, en plus de vous aider dans l'écriture du code, vous fournit de la documentation sur les classes, fonctions, etc. Son fonctionnement est repris plus en détail dans l'annexe A.

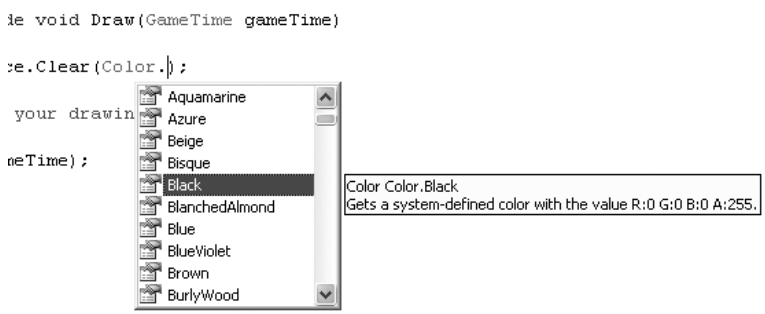

Figure 2-7

Choix d'une couleur grâce à IntelliSense

Si vous avez regardé le code de base d'un peu plus près, vous avez peut-être remarqué la présence d'un objet `graphics` de type `GraphicsDeviceManager`. C'est cet objet qui gère les traitements graphiques du jeu. Ainsi, vous pouvez facilement redimensionner votre application.

```
public Game1()
{
    graphics = new GraphicsDeviceManager(this);
    this.graphics.PreferredBackBufferHeight = 100;
    this.graphics.PreferredBackBufferWidth = 100;
    Content.RootDirectory = "Content";
}
```

Ou encore, la faire démarrer en mode plein écran.

```
public Game1()
{
    graphics = new GraphicsDeviceManager(this);
    this.graphics.ToggleFullScreen();
    Content.RootDirectory = "Content";
}
```

S'outiller pour développer sur Xbox 360

Si vous souhaitez développer pour la Xbox 360, vous devez disposer d'un abonnement Premium au XNA Creators Club. Cet abonnement vous coûtera 49 pour 4 mois ou 99 pour un an. Si vous êtes étudiant, vous avez accès à une version d'essai de l'abonnement premium. Renseignez-vous auprès de votre structure enseignante.

Il faut également configurer votre Xbox pour transférer vos jeux depuis votre PC :

1. Depuis votre console, connectez-vous à Xbox Live, puis téléchargez XNA Game Studio Connect.
2. Rendez-vous ensuite dans votre bibliothèque de jeu, puis dans la section « Jeux de la communauté » et lancez l'application que vous venez de télécharger.
3. La première fois que vous lancez cet utilitaire, vous voyez un code apparaître à l'écran, notez-le.
4. Sur votre ordinateur, lancez l'application XNA Game Studio Device Center, soit en allant la chercher dans le répertoire XNA Game Studio 3.0, soit à partir de Visual Studio (menu Outils).
5. Cliquez sur Add Device, choisissez Xbox 360, entrez le nom que vous voulez donner à votre console, puis insérez le code que vous avez récupéré auparavant.

Vous pouvez maintenant déployer votre projet sur votre Xbox. Pour ce faire, assurez-vous que vous avez bien démarré XNA Game Studio Connect sur la console et compilez le jeu dans Visual Studio. Le reste se fait automatiquement et votre jeu devrait apparaître sur la console.

Si vous avez créé un projet pour Windows et que vous voulez finalement le lire sur votre Xbox, il vous suffit de faire un clic droit sur le projet dans l'explorateur de solutions de

Visual Studio et de choisir Create Copy for Xbox 360. Le projet est alors prêt à être compilé pour la console.

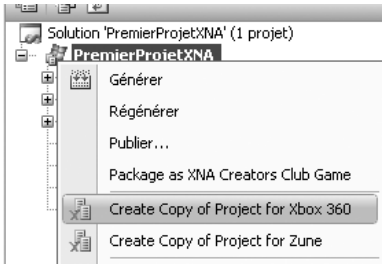

Figure 2-8

Création d'une copie de projet pour la Xbox 360

Développer un jeu pour la console ne se fait pas totalement de la même manière que pour Windows (notamment à cause de la diversité des moniteurs TV). Vous trouverez un guide des bonnes pratiques sur le site de la communauté :

http://creators.xna.com/en-US/education/bestpractices

En résumé

Dans ce chapitre, vous avez découvert :

- les types de jeux que vous pouvez créer avec XNA ;
- comment télécharger des jeux ou partager les siens avec la communauté ;
- la structure du framework et d'un projet avec XNA ;
- comment développer un jeu pour la Xbox 360.