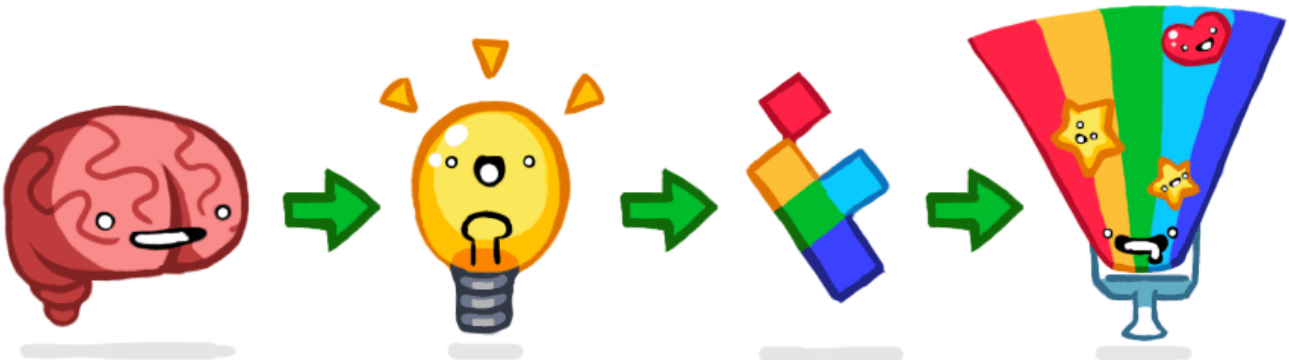
TP - Breakout

Simon Chauvin

LPJV 2012 – 2013



Etapes de développement

1 - Récupération du squelette d'un jeu Flixel

- Récupérez le dossier nommé "Flixel game skeleton" sur le réseau dans le dossier Simon Chauvin.

- Ouvrez le fichier "YourProjectName.as3proj" avec FlashDevelop, si vous appuyez sur F5 (compile et lance le jeu) une fenêtre Flash devrait se lancer confirmant que tout est bien installé.

2 - Modification du nom de votre projet

- Vous allez pouvoir modifier le nom du projet, fermez FlashDevelop et modifiez le nom du fichier "YourProjectName.as3proj".

- Ouvrez FlashDevelop et vous pouvez maintenant changer le nom du projet dans les fichiers "Preloader.as" et "YourProjectName.as", pensez à renommer le fichier .as également. Remplacez toutes les occurences de "YourProjectName" par le véritable nom de votre projet.

- Maintenant cliquez sur le menu "Projet" et sélectionnez "Propriétés", modifiez le nom du fichier "swf" qui sera généré par le compilateur Flash.

3 - Personnalisation du menu

- Dans le fichier "MenuState.as" contenant la classe MenuState vous pouvez apercevoir la méthode "create" qui permet d'initialiser l'état ainsi que la méthode "update" qui permet de le mettre à jour et d'éventuellement récupérer les entrées clavier par exemple.

- Modifiez le titre ainsi que la taille de sa police et sa couleur en modifiant les arguments de l'appel du constructeur de "FlxText" et de l'appel de la méthode "setFormat" présents dans la méthode

"create" de "MenuState".

- Ajoutez également votre nom dans le bas droit de l'écran, vous devrez créer un nouvel objet "FlxText" pour cela.

- Ajoutez la possibilité d'utiliser la touche "entrée" pour passer au "FlxState" suivant. Pour cela utilisez l'opérateur logique "||" (ou) dans l'instruction conditionnelle "if".

4 - Création de la classe Paddle

- Sélectionnez l'onglet "Projet" dans le panneau de droite de FlashDevelop et cliquez droit sur le dossier "src" puis sélectionnez "Ajouter" et enfin cliquez sur "Nouvelle Classe". Une fenêtre s'affiche où vous pouvez précisez le nom de la classe : "Paddle" puis sélectionnez sa classe mère dans le champ "Classe de base" : "FlxSprite".

- Nous allons maintenant déclarer l'image qui nous servira de sprite pour la raquette. Pour cela ajoutez cet attribut à la classe "Paddle" :

```
[Embed(source = '../assets/gfx/paddle.png')] protected var ImgPaddle:Class;
```

Cet attribut permet de spécifier le chemin de l'image à utiliser. Préférez l'utilisation de fichiers "png" dans la mesure où ils ne font pas de compressions irréversibles et qu'ils supportent la transparence.

- Dans le constructeur vous allez pouvoir utiliser l'instruction "super" pour appeler le constructeur de la classe mère "FlxSprite" et précisez la position "x" et "y" de votre sprite ainsi que l'image source.

- Vous pouvez à tout moment modifier la position d'un objet du jeu en modifiant ses attributs "x" et "y".

5 - Ajout de l'objet Paddle dans le jeu

- Ouvrez maintenant le fichier "PlayState.as" dans lequel nous allons décrire la plus grosse partie de notre gameplay. Nous retrouvons ici les mêmes méthodes "create" et "update" que dans n'importe quelle classe héritant de "FlxState".

- Vous pouvez changer la couleur d'arrière plan du jeu grâce à l'attribut "bgColor" de la classe "FlxG". Le format couleur est de ce type : 0xAARRGGBB et chaque double est un nombre hexadécimal représentant respectivement l'alpha, le rouge, le vert et le bleu. Un chiffre hexadécimal peut prendre les valeurs 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. C'est un système d'encodage de l'information au même titre que le système binaire.

- Dans la méthode "create" vous allez ajouter un objet "Paddle" nommé "paddle", vous devez pouvoir y accèder dans la méthode "create" et dans la méthode "update" donc pensez à le déclarez en tant qu'attribut de la classe. Vous devez ensuite instancier la raquette en appelant son constructeur et en y ajoutant le mot clé "new" puis utiliser la méthode "add" de "FlxState" pour le rajouter dans l'état de jeu.

- Si vous compilez et lancez votre application l'image de votre paddle devrait normalement s'afficher à l'écran.

6 - Création de la classe Ball et ajout de la balle

- Exécutez les mêmes étapes que pour la raquette mais dans le cas de l'objet "Ball".

- Compilez et lancez le jeu, la balle devrait s'afficher.

7 - Création de la classe Block et ajout des blocs

- Exécutez les mêmes étapes que pour la raquette mais dans le cas de l'objet "Block".

- Vous allez devoir créer plusieurs blocs à des positions différentes et pour cela vous devez passer en paramètres du constructeur de votre classe "Block" les positions "x" et "y" voulues. Pour cela, ajoutez en paramètres de votre constructeur deux entiers "xPosition" et "yPosition" que vous utiliserez dans l'appel du constructeur de "FlxSprite". Il vous suffira ensuite lors de l'appel du constructeur de "Block" de préciser une valeur de "x" et de "y" qui seront répercutées dans le constructeur de "FlxSprite".

- Pour ajoutez plusieurs blocs vous pouvez utilisez une boucle. Grâce à la taille du bloc et à l'indice courant vous pouvez placer des séries de blocs les uns à côtés des autres.

- Compilez et lancez le jeu vous devriez maintenant voir les blocs s'afficher à l'écran.

8 - Mise en mouvement de la balle

- Nous allons faire avancer la balle à une vitesse de 80 pixels par secondes grâce à l'objet "velocity" présent dans "FlxObject" (don't hérite "FlxSprite").

- Dans le constructeur de l'objet balle affectez à l'attribut "x" de l'objet "velocity" la valeur 80 et faites de même pour l'attribut "y".

- Compilez et lancez le jeu vous devriez voir bouger la balle.

9 - Gestion des commandes de la raquette

- Les commandes claviers ou souris effectuées par le joueur doivent être gérées à chaque frame, donc dans la méthode "update" de l'objet "Paddle".

- Créez la méthode "update" dans la classe "Paddle", n'oubliez pas qu'elle redéfinie la méthode "update" de sa classe mère "FlxSprite".

- Importez la classe FlxG, qui est une classe statique (non instanciée et non instaciable).

- L'objet "keys" accessible de cette manière "FlxG.keys" permet d’accéder aux méthodes "pressed(key)" et "justReleased(key)" permettant de savoir quand une touche à été enfoncée ou relâchée.

- Grâce a ces méthodes et à l'instruction conditionnelle "if" vous pouvez isoler le moment où le joueur appui sur une touche et appliquer le comportement souhaité.

- Nous voulons être en mesure de contrôler la raquette avec les flèches gauche et droite, à chacune de ces touches seront associées un mouvement.

- Lorsque la touche flèche gauche est enfoncée, modifier l'attribut "x" de l'objet "velocity" de la même façon que pour la balle. Même chose dans l'autre sens avec la flèche droite.

10 - Gestion des collisions entre la balle et la raquette

- Afin de gérer les collisions nous allons utiliser les fonctions fournis par Flixel.

- Des deux éléments la balle est celui qui doit rebondir et bouger, la raquette ne doit pas subir la collision. Nous ne voudrions pas que la raquette recule lorsque la balle rebondit dessus.

- Nous allons donc dans le constructeur de la raquette initialiser à "true" le booléen "immovable" tandis que nous initialisons l'attribut "elasticity" de la balle à "1" (élasticité maximum) dans le constructeur de la balle.

- Nous allons maintenant dire à Flixel d'effectuer la collision entre la balle et la raquette, à chaque

frame. Pour cela appelez la méthode "collide(object1:FlxBasic, object2:FlxBasic)" de l'objet "FlxG" avec comme arguments les objets "ball" et "paddle".

- Compilez et lancez le jeu, faites en sorte que les deux objets se rencontrent vous devriez voir la balle rebondir sur la raquette.

11 - Gestion des collisions de la balle et des blocs

- Flixel nous permet de gérer des collisions entre des objets mais également entre des groupes d'objets. Nous allons créer un groupe de blocs afin de pouvoir faciliter la gestion des collisions entre ceux-ci et la balle.

- Instanciez un nouvel objet "FlxGroup" appelé "blocks" et utilisez la méthode "add(object:FlxBasic)" fournie par "FlxBasic" dont hérite "FlxGroup" (de la même manière que "FlxState") pour ajouter chaque bloc au groupe. Pensez à ajouter le groupe à l'état de jeu.

- Maintenant dans la méthode "update" de votre état principal de jeu vous pouvez appelez la méthode "collide" de la classe "FlxG" en précisant en paramètres l'objet "ball" ainsi que le groupe "blocks".

- N'oubliez pas d'initialiser l'attribut "immovable" à "true" dans le constructeur de "Block" afin que ceux-ci ne subissent pas la collision avec la balle.

- Compilez et lancez le jeu, normalement la balle doit maintenant rebondir sur les blocs.

12 - Ajout d'un compteur de balles

- Lorsque la balle sort de l'écran (vers le bas ou le haut) nous souhaiterions qu'un compteur se décrémente et soit affiché à l'écran. Disons que notre compteur est initialisé à 3 et que nous l'affichons en haut à droite de l'écran.

- Nous allons donc créer un objet "FlxText" nommé "ballsLeftLabel" et une variable de type entier nommé "ballsLeft". Placez l'objet "ballsLeftLabel" en haut à droite.

- Maintenant, faites en sorte que chaque fois que la balle quitte l'écran le compteur de balles soit décrémenté et qu'une nouvelle balle apparaisse au centre de l'écran. Pensez au fait que la position "x" et"y" d'un objet correspond à son coin supérieur droit et non son centre.

- Pensez à mettre à jour le label avec la nouvelle valeur du compteur, pour cela vous devez mettre à jour l'attribut "text" de l'objet "ballsLeftLabel".

- Utilisez la méthode "toString" pour convertir un entier en une chaîne de caractères.

- Compilez et lancez le jeu, testez si cela fonctionne avec les quatre bords de l'écran.

13 - Gestion des rebonds de la balle sur les bords de l'écran

- Lorsque la balle touche les bords droit et gauche de l'écran vous devez inversez la "velocity" de la balle.

14 - Gestion du game over

- Une fois que le compteur atteint zéro vous devez afficher un message à l'écran précisant que la partie est finie. Proposez de recommencer ou de revenir au menu.

- Utilisez les attributs "width" et "height" présents dans "FlxG" qui contiennent la taille de l'écran.

- Pour recommencer une partie vous devez appeler la méthode "switchState" fournie par la classe "FlxG" et lui passer en paramètre l'état de jeu à recommencer.

- Compilez et testez.

15 - Rendre les blocs destructibles

- Détruisez chaque bloc touché par la balle.

- Pour cela nous devons ajouter un paramètre lors de l'appel de la fonction "collide" sur la balle et le groupe de blocs. Ce paramètre doit être le nom de la fonction qui va être exécutée lorsqu'un bloc et la balle se rencontrent.

- Cette fonction peut-être privée puisqu'elle ne sera utilisée que dans la classe. Elle doit prendre en arguments deux "FlxObject". Écrivez là.

- Dans cette fonction vous allez tester si les "FlxObject" passés en paramètres sont du type "Block" avec l'opérateur "is" qui permet de vérifier si un objet est bien d'un certain type.

- Si c'est le cas alors vous devrez "tuer" l'objet en question avec la méthode "kill". De cette manière l'objet disparaîtra de l'écran et ne sera plus considéré par le jeu.

16 - Ajout d'un score

- Ajoutez un score et incrémentez le lorsqu'un bloc est détruit.

17 - Ajout d'un bonus qui élargie la taille de la raquette

- Ajoutez un nouveau type d'objet héritant de "FlxSprite".

- Testez la collision entre cet objet et la balle pour savoir si le bonus a été récupéré ou non. N'utilisez pas la méthode "collide" car les la balle est un objet physique mais pas le bonus qui doit simplement disparaître lorsqu'il est touché. A la place utilisez la fonction "overlap" qui renvoie "true" si les deux objets rentrent en collision mais n'effectue pas d'opérations de physiques, la balle ne rebondira pas sur le bonus.

- Si les deux objets se rencontrent, faites disparaître le bonus et utilisez un nouveau sprite plus grand et chargez le à la place de l'ancien sprite de la raquette. La méthode "loadGraphic" vous permet à tout moment de changer l'image d'un objet héritant de "FlxSprite".

18 - Ajout d'un malus qui rétrécit la balle

- Même chose que précédemment mais dans le cas d'un rétrécissement maintenant.