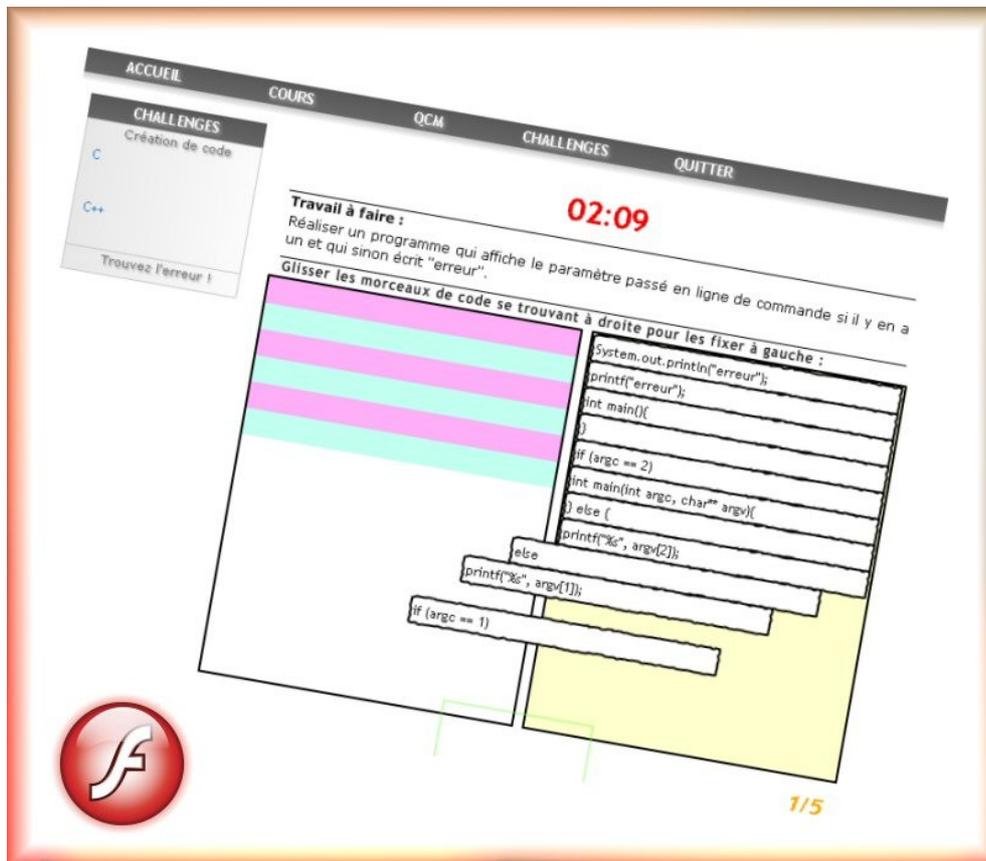


IN42

Création d'un programme interactif d'apprentissage de la programmation



SOMMAIRE

<u>I. PRÉSENTATION DU PROGRAMME.....</u>	<u>3</u>
1. PARTIE NAVIGATION.....	3
2. PARTIE APPRENTISSAGE.....	4
<i>a. Partie cours.....</i>	<i>4</i>
<i>b. Partie QCM.....</i>	<i>4</i>
3. PARTIE CHALLENGE.....	5
<i>a. Challenge « Création de code ».....</i>	<i>5</i>
<i>b. Challenge « Trouver l'erreur ! ».....</i>	<i>6</i>
<u>II. PRISE EN COMPTE DE L'UTILISATEUR.....</u>	<u>7</u>
1. ERGONOMIE DU PROGRAMME.....	7
<i>a. Cible du programme.....</i>	<i>7</i>
<i>b. Sobriété et intérêt.....</i>	<i>7</i>
2. INTÉRÊT DU PROGRAMME.....	8
<i>a. Apprentissage ludique.....</i>	<i>8</i>
<i>b. QCM et Challenges stimulants.....</i>	<i>8</i>
<u>III. EVOLUTIVITÉ.....</u>	<u>10</u>
1. EVOLUTIVITÉ DE LA BASE DOCUMENTAIRE.....	10
2. AJOUTS DE FONCTIONNALITÉS.....	10

I. Présentation du programme

Le but de ce projet était de créer un programme sous Flash 8/AS2 permettant de présenter de manière interactive du contenu tout en permettant à l'utilisateur de consolider ses acquis en effectuant des tests.

Nous avons choisi de développer un programme interactif d'apprentissage de la programmation qui se décompose en trois parties :

- **la partie navigation** permettant à l'utilisateur d'accéder à toutes les informations du programme
- **une partie apprentissage** permettant de présenter des notions de programmation à l'utilisateur sous la forme de **cours** et de **QCM**. Ces derniers permettant à l'utilisateur d'approfondir les notions vues dans le cours tout en lui fournissant **des explications permettant de progresser**.
- **une partie Challenge** permettant à l'utilisateur de tester si les connaissances qu'il a acquises sont solides.

1. Partie navigation

La partie navigation a été conçue pour **permettre à l'utilisateur d'accéder rapidement à ce qu'il veut**. L'utilisateur peut ainsi naviguer dans les principales parties du programme à l'aide d'une barre de menu se situant en haut de l'écran :



Cette barre de menu est bien entendue **paramétrable à l'aide du fichier XML « menu.xml »** qui a permis ici d'ajouter les rubriques COURS, QCM ainsi que CHALLENGES.

Chaque rubrique ajoutée dans le menu principal donne lieu à **un sous-menu déroulable** qui apparaîtra dans la partie gauche de l'écran. Ce sous-menu est aussi paramétrable à l'aide d'un fichier XML dont l'adresse doit être donnée dans « **menu.xml** ». Chaque sous-menu est accompagné d'une **page d'indications** qui est elle aussi configurable en XML.



Pour plus d'ergonomie, **les deux menus sont réactifs au passage de la souris**.

2. Partie apprentissage

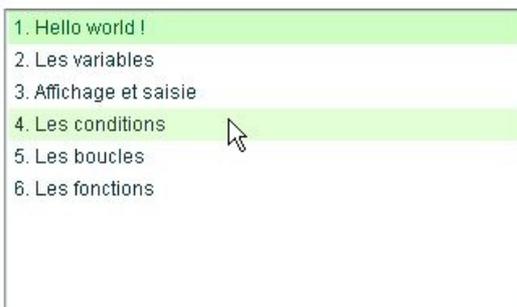
a. Partie cours

La partie cours permet à l'utilisateur de se former dans les domaines qu'il souhaite. A l'aide du menu, il peut ainsi rapidement accéder à la partie qui l'intéresse. **Les cours sont entièrement paramétrables à l'aide de fichiers XML** qui partitionnent le cours en un certain nombre de **chapitres au format HTML**. Il est ainsi très facile de créer ses propres cours puisqu'il suffit de créer une page au format HTML et le cours est créé.

L'utilisateur est dans un premier temps invité à choisir le chapitre qu'il souhaite consulter dans le plan du cours. Il lui est tout à fait possible lors de la consultation de revenir à un chapitre antérieur par l'intermédiaire des **boutons de navigation** (précédent, sommaire, suivant).

Mon premier programme

Plan du cours



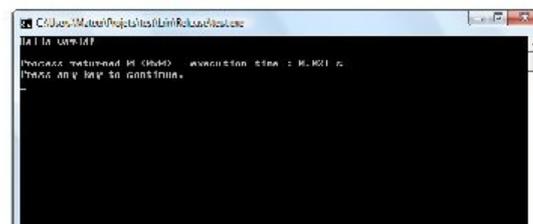
Commencer

3. Affichage d'un texte

Rien de plus simple, il suffit de faire appel à la fonction **printf** prendre 1 ou plusieurs paramètres. Le premier est une chaîne de caractères, les autres sont des noms de variables (on y reviendra plus tard).
Voici une ligne d'exemple :

```
printf("Hello world !\n");
```

Voici ce qui s'affiche dans votre console :



Remarque : le caractère \n fait passer à la ligne lors de l'affichage.



b. Partie QCM

La partie QCM fait partie intégrante de l'apprentissage de l'utilisateur. Les QCM peuvent en effet regrouper une ou plusieurs parties du cours. L'utilisateur peut ainsi se tester en ayant un réel retour sur ses acquis puisque, **à la fin de chaque test, des explications permettent à l'utilisateur de progresser.**

Les QCM sont bien entendus **entièrement configurables par l'intermédiaire de fichiers XML**.

Q.C.M. : Le langage C

Vous obtenez un score de **5 bonnes réponses sur 5**.

Vous gagnez un niveau, Bravo ! Votre nouveau niveau est le 1.

Détails de votre QCM :

Quel est le nom de la fonction principale d'un programme ?
Vous avez répondu : main
Vrai !

Lequel de ces éléments ne fait pas partie d'un IDE ?
Vous avez répondu : Le freezer
Vrai !

Pour stimuler l'utilisateur, un système de points a été mis en place, ce qui lui permet de voir son évolution. Un tableau récapitulatif des différents tests passés est d'ailleurs disponible :

Résultats des Q.C.M.

Ici, vous retrouverez tous vos résultats aux Q.C.M.

Q.C.M.	Niveau	Score	Nb de Questions	Date
Le langage C++	2	3	5	03/05/2009 16h35
Le langage C	1	5	5	03/05/2009 16h35
Le langage C	0	5	5	03/05/2009 16h34

3. Partie Challenge

Afin de tester la solidité des acquis de l'utilisateur, deux challenge ont été mis en place, un challenge de type drag & drop : « Création de code » et un challenge plus classique « Trouver l'erreur ! ».

a. Challenge « Création de code »

Ce challenge, de type **drag & drop** consiste à soumettre l'utilisateur à une série d'exercices dont le but est de **créer, en temps limité, selon un objectif et avec un code**

fourni le code d'un programme fonctionnel. Avant de commencer, l'utilisateur pourra choisir le niveau de difficulté qui lui correspond le mieux. (Voir II)

01:38

Travail à faire :

Réaliser un programme qui affiche le paramètre passé en ligne de commande si il y en a un et qui sinon écrit "erreur".

Glissez les morceaux de code se trouvant à droite pour les fixer à gauche :

```
int main(int argc, char** argv){
if (argc == 2)
printf("erreur");
}

printf("%s", argv[2]);
System.out.println(argv[1]);
printf("%s", argv[1]);
else
```

1/5

b. Challenge « Trouver l'erreur ! »

Ce challenge consiste à soumettre l'utilisateur à une série d'exercices dont le but est de trouver, en temps limité, les erreurs présentes dans le code d'un programme. Avant de commencer, l'utilisateur pourra aussi choisir le niveau de difficulté qui lui correspond le mieux. (Voir II)

00:02

Rapport d'erreur : Erreur de syntaxe.

Travail à faire :

Sélectionnez la ou les lignes qui posent problèmes.

Sélectionnez la ou les lignes demandées (sélection multiple avec CTRL) et appuyez sur Valider :

```
switch (variable)
{
case > 5:
printf("Traitement 1");
break;
case default:
printf("Traitement par defaut");
};
```

Valider

5/5

II. Prise en compte de l'utilisateur

1. Ergonomie du programme

a. Cible du programme

Le programme cible **les utilisateurs sensibles de vouloir apprendre la programmation**. Etant donné que ce type d'utilisateur est habitué à utiliser des logiciels, **des composants standard ont été utilisés** pour créer la majorité du programme (Listes, champs de textes, cases à cocher).

Les 7 règles d'or de Joëlle Coutaz ont été appliquées : **l'ensemble est cohérent** (la partie navigation est claire), **concis, il n'y a pas de charges cognitives inutiles** (des animations, mais pas trop), **l'utilisateur décide à tout moment de ce qu'il veut faire** (en encourageant éventuellement une perte de points si il quitte précipitamment un challenge), **l'ensemble est souple et peu très facilement évoluer**, **les cours sont structurés** et pendant les tests **l'utilisateur sait à tout moment où il en est**.

Nous avons par ailleurs **respecté les codes couleurs** pour informer l'utilisateur de sa progression (rouge lors d'une perte de points et vert lors d'un gain).

b. Sobriété et intérêt

Le programme réalisé se veut réellement **sobre sans toutefois tomber dans la platitude**. L'idée était de réaliser un programme motivant pour l'utilisateur sans en faire trop ni pas suffisamment. Ainsi, pour pimenter l'intérêt de l'utilisateur, **des animations ont été introduites avec parcimonies** et **un composant de drag & drop a été travaillé pour qu'il soit réellement ludique, performant et ergonomique**.

Le chronomètre avertit de manière intelligente l'utilisateur du temps restant : **au début le son est calme pour ne pas gêner l'utilisateur** et **le son devient plus puissant** pour avertir l'utilisateur que le challenge va se terminer.

2. Intérêt du programme

a. Apprentissage ludique

Afin de motiver l'utilisateur, **une gestion de niveau a été introduite**. L'utilisateur part au début du programme avec le niveau 0 et il peut arriver au niveau 9 en pratiquant les QCM et les challenges. **Un chronomètre sonore et visuel a été utilisé** pour stimuler l'utilisateur pendant les challenges. Il a ainsi plusieurs défi : **réussir les questions du challenge tout en respectant la contrainte de temps**.

b. QCM et Challenges stimulants

Les QCM et les deux challenges ont été travaillés de manière à ce que l'utilisateur reste constamment motivé.

- Retour informatif à la fin du QCM

Afin que l'utilisateur progresse plus rapidement, **les QCM ont été conçus de manière à ce qu'ils apportent un complément au cours**. Ainsi, l'utilisateur voit à la fin pour chaque question : un message lui indiquant qu'il a réussi et dans le cas contraire la bonne réponse.

- Gestion intelligente des questions

Afin que l'utilisateur éprouve de l'intérêt à tester ses connaissances, **la sélection des questions a été particulièrement travaillée**.

Dans les deux cas, pour les QCM et pour les challenges, **les questions sont sélectionnées aléatoirement**.

Pour les QCM, **les questions sont sélectionnées aléatoirement selon le niveau du candidat**. Des questions de son niveau sont d'abord sélectionnées, puis si il n'y en a pas 5, on autorise une différence de difficulté de 1 points et ainsi de suite.

Afin de pimenter les challenges, une technique est utilisée pour que le candidat puisse sélectionner son niveau de difficulté tout en sélectionnant les questions de manières à ce qu'il puisse **surpasser son niveau actuel**. Ainsi, **toutes les questions supérieures à son niveau sont sélectionnées puis une sélection par ranking est effectuée**. Cela va permettre de tirer de préférence les questions qui se rapprochent du niveau de l'utilisateur tout en permettant à des questions plus difficiles d'être sélectionnées. **Le challenge « Trouver l'erreur ! » sélectionne aléatoirement le nombre de mauvaises lignes qui vont être insérés dans**

l'exercice. Le challenge « Création de code » va quant à lui sélectionner aléatoirement le nombre de ligne inutile rajoutée pour composer un code.

Le candidat pourra aussi à chaque début de challenge procéder à plusieurs tirages d'exercices pour choisir le niveau de difficulté qui lui correspond le mieux. Plus la difficulté est grande, plus le gain maximum est important et plus la pénalité encourue est faible, et inversement. Le tirage aléatoire des exercices s'adapte en fonction du niveau de l'utilisateur et utilise des procédés permettant de varier grandement les questions.

III. Evolutivité

1. Evolutivité de la base documentaire

La partie la plus intéressante de ce programme réside sans doute dans le fait qu'**il serait très facilement possible de faire évoluer la base documentaire**. Pour cela, il est possible de paramétrer presque tout par XML et il est possible de rajouter des cours très facilement en utilisant le format HTML.

2. Ajouts de fonctionnalités

A partir de ce programme, il est très facilement possible d'ajouter des fonctionnalités. En effet, de par la structure utilisée (utilisation de fichiers XML faisant appels à des sous-animations Flash, **utilisations de classes**), il est possible d'ajouter de nouveaux challenges par exemple.

Pour conclure, ce programme a été créé de manière à stimuler l'utilisateur. Pour cela, le programme propose une partie apprentissage et des défis stimulants. Afin de faciliter l'utilisation du programme, les principes principaux d'ergonomie ont été respectés.