Fabien Cocheteux – A2 Prog – IIM

Premier programme

I – Objectif :

L'objectif de ce TP est de programmer un premier jeu simple dans une console en mode texte.

Les règles sont les suivantes :

L'application indique au joueur un temps à compter dans sa tête.
Le joueur appuie une première fois sur n'importe quelle touche pour commencer le jeu.
Le joueur appuie une seconde fois sur une touche lorsqu'il estime que le temps écoulé depuis le début du jeu est proche du temps à estimer.
L'application affiche au joueur le temps écoulé entre les deux appuis.

I – Mise en place projet :

Créer un nouveau projet C++ "ChronoSpacer" avec le template "Console Windows".
Tous le code de ce projet sera ajouté dans la fonction main de ChronoSpacer.cpp.

III – Implémentation simple :

Programmer les règles de base du jeu en utilisant les fonctions documentées plus bas.

Par étape :

- Déclarer une variable contenant le temps à compter dans sa tête

```
float timeToFind = 5.0f;
```

- Afficher les règles du jeu aux joueurs avec la fonction “std::cout”
- Mettre une première fois le jeu en pause en utilisant la fonction “system(“pause”)” de manière à attendre que le joueur appuie sur une touche pour lancer le jeu.
- Après la pause, déclarer une variable contenant le temps en début de jeu et l’initialiser avec la valeur de retour de la fonction “clock()”

```
float startTime = (float)clock() / (float)CLOCKS_PER_SEC;
```

- Mettre une seconde fois le jeu en pause en utilisant la fonction “system(“pause”)” pour attendre que le joueur compte le temps à estimer dans sa tête.
- Déclarer une variable contenant le temps en fin de jeu et l’initialiser avec la valeur de retour de la fonction “clock()”

```
float endTime = (float)clock() / (float)CLOCKS_PER_SEC;
```

- Calculer la différence entre le temps de début et le temps de fin. Stocker cette valeur dans une nouvelle variable de type float identifiée “elapsedTime”.
- Afficher la valeur contenue dans “elapsedTime” en console avec la fonction “std::cout”

III - a) std::cout

La fonction « std::cout » affiche un texte en console en C.

Elle est défini dans la bibliothèque « iostream » qui doit être ajouté en début de programme en utilisant l'instruction préprocesseur « include » :

```
#include <iostream>
```

La syntaxe d'appel à la fonction std::cout est particulière et s'écrit : std::cout << valeur << std::endl;

```
std::cout << "Hello world!" << std::endl;
```

Résultat : "Hello world!"

std::endl marque un retour à la ligne. Il est également possible d'écrire un retour à la ligne avec le caractère spécial '\n'.

```
std::cout << "Hello world!\n";
```

Pour afficher plusieurs valeurs à la suite en console, on écrira :

```
std::cout << valeur1 << valeur2 << valeur3 << std::endl;
```

III - b) system

La fonction « system » permet d'invoquer des commandes MSDOS en console.
Elle est défini dans la bibliothèque « windows.h » :

```
#include <windows.h>
```

Cette fonction prend un paramètre obligatoire qui indique la commande à exécuter. Par exemple :

```
system("dir");
```

Résultat : affiche le contenu du dossier d'exécution de l'application

En particulier :

```
system("pause");
```

Résultat : met en pause l'exécution de l'application en attendant qu'une touche du clavier soit appuyée.

III - c) clock

La fonction « clock » permet de récupérer le temps du processeur depuis le début de l'application. Ce temps est compté en ticks processeur.

Elle est défini dans la bibliothèque « time.h » :

```
#include <time.h>
```

La fonction ne prend pas de paramètre. En revanche, elle **retourne** une valeur. Autrement dit lorsqu'elle est évaluée, la fonction est remplacée dans l'expression par la valeur du temps écoulé.

```
long timeInTicks = clock();
```

La constante « CLOCKS_PER_SEC » est une valeur fixe définie dans la bibliothèque « time.h ». Cette valeur permet de convertir le temps en ticks processeur en un temps en secondes par une simple division.

```
float timeInSeconds = (float)clock() / (float)CLOCKS_PER_SEC;
```

IV – Structure conditionnelle « if / else »

Si le joueur arrive à estimer le temps à plus ou moins une seconde prêt, le jeu affiche au joueur qu'il a gagné. Sinon, le jeu affiche au joueur qu'il a perdu.

Créer une variable contenant comme valeur le temps de l'intervalle de tolérance :

```
float winInterval = 1.0f;
```

Implémenter cette règle en fin de programme à l'aide d'une structure conditionnelle « if / else ». L'expression conditionnelle comparera cette valeur d'intervalle à la valeur obtenue par le calcul : "elapsedTime - timeToFind".

V – Boucle « for »

Utiliser une boucle « for » pour permettre de jouer le jeu 10 fois par exécution.

VI – Aléatoire

On souhaite tirer aléatoirement le temps à découvrir entre une valeur minimale et une valeur maximale

Pour le tirage aléatoire, inclure <cstdlib> et <ctime> en début de programme. On utilisera alors les fonctions suivantes :

```
- srand(time(NULL)); // Random seed initialization. Must be called only once !
- rand(); // Random int between 0 and the constant value RAND_MAX (excluded)
```

Ainsi que la valeur constante :

```
- RAND_MAX // Max random value possible returned by rand()
```

On utilisera enfin le calcul suivant pour obtenir une valeur avec le calcul suivant :

```
float timeToFind = ((float)rand() / (float)RAND_MAX) * (maxTime - minTime)) + minTime;
```

A chaque boucle de jeu, un nouveau temps aléatoire à estimer est calculé.

VII – Structure conditionnelle « while » :

Remplacer la boucle de jeu précédemment implémentée en respectant la règle suivante :

Si le joueur gagne la partie, alors l'intervalle de tolérance est réduit de 80% de sa valeur précédente et le jeu recommence. Si le joueur perd, le jeu s'arrête.

Utiliser une boucle « while » pour implémenter cette règle.

VIII – Bonus :

Au delà de 10 secondes après le temps à estimer, le jeu est automatiquement perdu.

Inclure <conio.h> puis copier la fonction ci-dessous **AVANT** la fonction main.

```
// Get Input
bool isKeyPressed()
{
	if (_kbhit())
	{
		while (_kbhit()) // Flush input buffer
		{
			_getch();
		}
		return true; // Key Was Hit
	}
	return false; // No keys were pressed
}
```

La fonction isKeyPressed ne bloque pas l'exécution du programme. Sa valeur de retour est « true » si une touche a été appuyée depuis sa dernière exécution, sinon « false ».

Dans la boucle de jeu, remplacer le second « system("pause") » par une structure conditionnelle while en utilisant la fonction isKeyPressed pour réaliser la règle énoncée ci-dessus.