Algorithmiques avec C# et Unity

1.1 Variables, structures conditionnelles et boucles

Dans cette section, nous allons explorer les concepts fondamentaux de la programmation en C#, notamment les variables, les structures conditionnelles et les boucles. Ces éléments sont essentiels pour écrire des programmes efficaces et fonctionnels. Vous avez déjà une certaine expérience en programmation, le langage JavaScript étant similaire à C# dans plusieurs aspects. Nous allons donc nous concentrer sur les particularités de C# tout en renforçant vos compétences en programmation.

1.2 Types de données

C# est un langage fortement typé, ce qui signifie que chaque variable doit être déclarée avec un type de données spécifique. Voici quelques types de données couramment utilisés en C# :

- `string` : pour les chaînes de caractères.
- `int` : pour les nombres entiers.
- `float` : pour les nombres à virgule flottante.
- `bool` : pour les valeurs booléennes (vrai ou faux).
- `GameObject` : pour représenter des objets dans un environnement de jeu (utilisé dans des moteurs de jeu comme Unity).
- `Vector2` : pour représenter des vecteurs en 2D (utilisé dans des contextes graphiques ou de jeu).
- `Vector3` : pour représenter des vecteurs en 3D (utilisé dans des contextes graphiques ou de jeu).

Nous verrons d'autres types de données au fur et à mesure de notre progression.

1.2.1 Variables

Les variables sont des conteneurs utilisés pour stocker des données. En C#, chaque variable doit être déclarée avec un type spécifique, ce qui permet au compilateur de vérifier la validité des opérations effectuées sur ces variables. Les nombres entiers sont déclarés avec le type `int`, les nombres à virgule flottante avec le type `float`, les chaînes de caractères avec le type `string` et les valeurs booléennes avec le type `bool`.

À noter : En C#, les variables de type `float` doivent être suivies de la lettre `f` pour indiquer qu'il s'agit d'un nombre à virgule flottante.

```
string nom = "Alice"; // Déclaration d'une variable chaîne de caractères
int score = 4; // Déclaration d'une variable entière
float taille = 1.75f; // Déclaration d'une variable flottante
```



```
bool estMort = true; // Déclaration d'une variable booléenne
```

1.3 Concaténation de chaînes

La concaténation de chaînes en C# peut être réalisée de plusieurs façons. Voici les méthodes les plus courantes :

1.3.1 Utilisation de l'opérateur +

```
string message = "Joueur: " + nom + ", Score: " + score;
Debug.Log(message);
```

1.3.2 Utilisation d'interpolation de chaînes

Similairement à JavaScript, C# permet l'interpolation de chaînes en utilisant le symbole $ avant la chaîne. Cela facilite l'insertion de variables directement dans la chaîne. Ensuite, les variables sont placées entre accolades {}.

```
string message = $"Joueur: {nom}, Score: {score}, Vie: {vie}%";
Debug.Log(message);
```

1.3.3 Structures conditionnelles

Comme en JavaScript, les structures conditionnelles en C# permettent d'exécuter du code en fonction de certaines conditions. Les principales structures conditionnelles sont `if`, `else if` et `else`.

On place la condition entre parenthèses ( ) et le bloc de code à exécuter entre accolades {}.

```
if (score > 10) {
    // Code à exécuter si le score est supérieur à 10
} else if (score == 10) {
    // Code à exécuter si le score est égal à 10
} else {
    // Code à exécuter si le score est inférieur à 10
}
```

1.4 Opérateurs logiques et de comparaison

Les opérateurs de comparaison en C# sont similaires à ceux utilisés en JavaScript. Ils permettent de comparer des valeurs et de retourner un résultat booléen (`true` ou `false`). Voici les principaux opérateurs de comparaison :

```
-   `==` : égal à
-   `\≠` : différent de
-   `>` : supérieur à
-   `<` : inférieur à
```

```
-   `≥` : supérieur ou égal à
-   `≤` : inférieur ou égal à
```

Les opérateurs logiques en C# sont également similaires à ceux de JavaScript. Ils permettent de combiner plusieurs conditions logiques. Voici les principaux opérateurs logiques :

```
&& : et logique (les deux conditions doivent être vraies)
|| : ou logique (au moins une des conditions doit être vraie)
! : non logique (Si la condition est vraie, retourne faux et vice versa)
```

```
if (score > 10 && estMort == false) {
    // Code à exécuter si le score est supérieur à 10 ET que le joueur n'est pas mort
}
```

1.5 Fin d'une ligne

À la différence de JavaScript, chaque instruction en C# doit se terminer par un point-virgule `;`. Cela indique la fin de l'instruction au compilateur.

```
int score = 4 // Cette ligne est incorrecte en C# et générera une erreur de compilation
score = score + 1; // Incrémentation du score
```