

TD1
Git, C#
Et introduction à Unity

Chapitre 1 : Git/Github

Git est un système de contrôle de version. Son but initial était d'aider des groupes de développeurs à travailler en collaboration sur de grands projets logiciels. Git gère l'évolution d'un ensemble de fichiers - appelé référentiel (repository) - de manière saine et très structurée.

GitHub propose une plateforme pour vos projets basés sur Git sur Internet.

Démarrer avec github

Cette section fournit une vue d'ensemble de ce qu'est github et pourquoi un développeur peut vouloir l'utiliser.

Il devrait également mentionner tous les grands sujets dans github, et établir un lien avec les sujets connexes. La documentation de github étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Exemples

Installation ou configuration

GitHub est une énorme collection de dépôts Git. En d'autres termes, vous pouvez considérer GitHub comme un ensemble de projets!

Créer un compte

- Visitez la page principale de GitHub <https://github.com/>
- Choisissez un nom d'utilisateur, entrez votre adresse e-mail, puis choisissez un mot de passe sécurisé et vous êtes prêt à partir!

Outils utiles

Pour les débutants Git / GitHub, comprendre comment fonctionne le contrôle de version peut être déroutant au début. Il existe une version graphique de GitHub que vous pouvez télécharger et utiliser. <https://desktop.github.com/> est juste cet outil.

Créer votre premier référentiel

Vous pouvez considérer un référentiel comme un projet. Vous pouvez créer un référentiel en ligne ou hors ligne. Suivez les étapes ci-dessous:

En ligne

1. Connectez-vous d'abord et accédez à votre profil.
2. Accédez à l'onglet "Référentiels (Repositories)" en haut de la page.



Overview Repositories 2 Projects Packages Stars

3. Appuyez sur le bouton vert "Nouveau (New)" et vous êtes prêt!



Overview Repositories 2 Projects Packages Stars



Find a repository...

Type ▾

Language ▾

Sort ▾

New


4. Nommez votre référentiel (Repository)


Repository name *

YourFirstGame



5. Ajoutez un fichier ReadMe et créez ensuite votre référentiel

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.


☒ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)


Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None ▼

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

License: None ▼

This will set  **main** as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

[Create repository](#)

Hors ligne

1. Téléchargez et installez [git](#) (choisissez le système d'exploitation que vous utilisez)
2. Après le téléchargement et l'installation, vous pouvez utiliser l'outil de ligne de commande ou télécharger un client d'interface graphique.
3. Après l'installation, créez un compte sur [github](#)
4. En haut à droite, cliquez sur + et choisissez soit de créer un nouveau référentiel, soit d'importer un existant.
5. Si vous en choisissez un nouveau, entrez le nom du référentiel et choisissez de le rendre public ou privé.
6. Cliquez sur: Créer un référentiel

NB Les dépôts privés ne sont pas disponibles pour les utilisateurs gratuits.

Fichier README

Si votre projet ne contient pas README.md, GitHub peut analyser README.rdoc pour afficher les détails. S'il a les deux, il utilisera README.md, ignorant silencieusement rdoc.

Un fichier README peut inclure-

Titre du projet

Décrivez brièvement votre projet. Vous pouvez également fournir le lien vers le site Web du projet, les badges, la communauté et les informations de contact (par exemple, email, site social).

Télécharger

Lien exécutable (exécutable ou minifié ou fichier d'installation). Il peut également y avoir des liens vers les versions précédentes.

Installation

Comment votre travail peut être utilisé. Cela peut inclure les conditions préalables, les paramètres, les bibliothèques tierces, l'utilisation, les mises en garde, etc.

Manifestation

Il peut s'agir d'un échantillon de code, d'un fichier gif, d'un lien vidéo ou même de captures d'écran.

Auteurs

Noms d'auteur, coordonnées, etc.

Remerciements

Liste de personnes ou de communautés aidées et inspirées tout au long du projet




Contributeur


Instructions pour contribuer (c.-à-d. Ajouter une fonctionnalité, signaler un bogue, soumettre un correctif) au projet. Peut également inclure un lien de documentation.

License

Donnez une courte introduction sur votre licence. Vous pouvez également donner un lien vers le site de licence.

[main](#) [1 branch](#) [0 tags](#) [Go to file](#) [Add file](#) [Code](#)

 romado-workshop Initial commit	8f0e4ed now	 1 commit
 README.md	Initial commit	now

README.md 

YourFirstGame

Vous pouvez maintenant créer/sauvegarder/mettre à jour vos fichiers de travail.

Chapitre 2 : Programmation orientée objet

La programmation orientée objet (POO) est un paradigme de programmation informatique. Elle consiste en la définition et l'interaction de briques logicielles appelées objets ; un objet représente un concept, une idée ou toute entité du monde physique, comme une voiture, une personne ou encore une page d'un livre.

La programmation par objet consiste à utiliser des techniques de programmation pour mettre en œuvre une conception basée sur les objets. Celle-ci peut être élaborée en utilisant des méthodologies de développement logiciel objet, dont la plus connue est le processus unifié (« Unified Software Development Process » en anglais), et exprimée à l'aide de langages de modélisation tels que le Unified Modeling Language (UML).

L'un des langages de programmation de POO est le C#. C# est un langage de programmation moderne, polyvalent et orienté objet, développé par Microsoft et approuvé par l'ECMA et l'ISO. C# est conçu pour la Common Language Infrastructure (CLI), qui consiste en un code exécutable et un environnement d'exécution permettant l'utilisation de divers langages de haut niveau sur différentes plates-formes et architectures informatiques.

Les raisons suivantes font de C# un langage professionnel largement utilisé :

- C'est un langage de programmation moderne et polyvalent.
- Il est orienté objet.
- Il est orienté composants.
- Il est facile à apprendre.
- C'est un langage structuré.
- Il produit des programmes efficaces.
- Il peut être compilé sur une variété de plateformes informatiques.
- Il fait partie du cadre .Net.

Caractéristiques de programmation fortes de C#

Voici une liste de quelques caractéristiques importantes de C# :

- Conditions booléennes
- Collecte automatique des déchets
- Bibliothèque standard
- Versionnage d'assemblage
- Propriétés et événements
- Gestion des délégués et des événements
- Génériques faciles à utiliser
- Indexeurs
- Compilation conditionnelle
- Multithreading simple
- LINQ et Expressions Lambda
- Intégration avec Windows

Environnement de développement intégré pour C#

Microsoft fournit les outils de développement suivants pour la programmation C# :

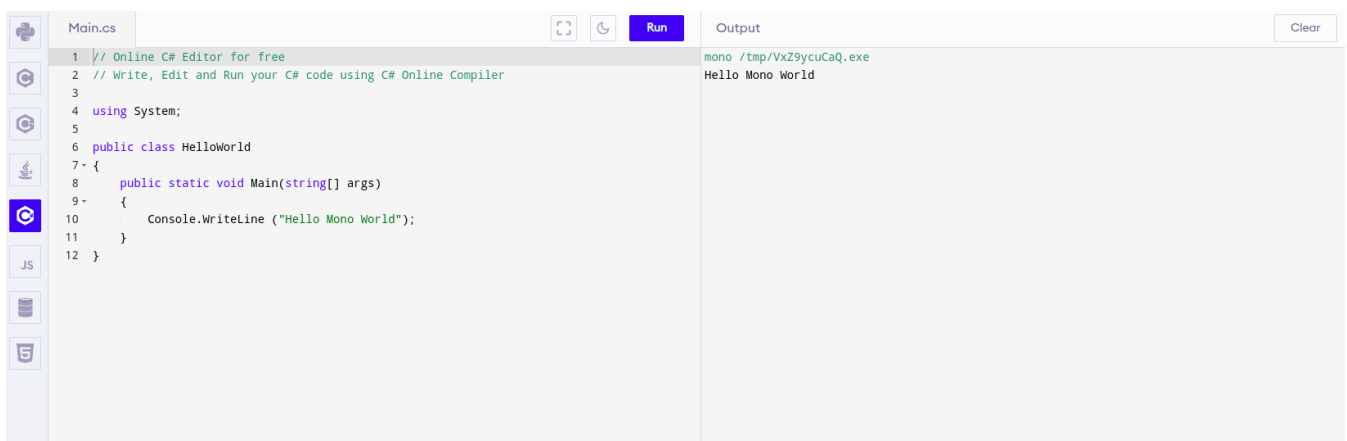
- Visual Studio 2010 (VS)
- Visual C# 2010 Express (VCE)
- Visual Web Developer

Les deux derniers sont disponibles gratuitement sur le site officiel de Microsoft. Grâce à ces outils, vous pouvez écrire toutes sortes de programmes C#, des simples applications en ligne de commande aux applications plus complexes.

Vous pouvez également écrire des fichiers de code source C# à l'aide d'un éditeur de texte de base comme Notepad, et compiler le code en assemblages à l'aide du compilateur en ligne de commande qui fait également partie du .NET Framework.

Création d'un code simple

Dans ce TD, nous allons utiliser le compilateur en ligne "rogramiz - <https://www.programiz.com/csharp-programming/online-compiler/>".



The screenshot shows the Programiz online C# compiler interface. On the left, there is a sidebar with icons for different programming languages: C#, JavaScript, Python, Java, C++, PHP, and Ruby. The main editor area displays a C# code file named 'Main.cs' with the following content:

```
1 // Online C# Editor for free
2 // Write, Edit and Run your C# code using C# Online Compiler
3
4 using System;
5
6 public class HelloWorld
7 {
8     public static void Main(string[] args)
9     {
10         Console.WriteLine ("Hello Mono World");
11     }
12 }
```

At the top right of the editor, there are icons for undo, redo, and a 'Run' button. To the right of the editor is an 'Output' window showing the result of the compilation and execution:

```
mono /tmp/VxZ9ycuCaQ.exe
Hello Mono World
```

A 'Clear' button is located at the top right of the output window.

Vous pouvez remarquer qu'il y a déjà un programme simple, appelé "Hello World".

Comprenons le code ligne par ligne :

- La première ligne du programme `using System` ; - le mot-clé `using` est utilisé pour inclure l'espace de nom `System` dans le programme. Un programme comporte généralement plusieurs instructions `using`.
- La ligne suivante comporte une déclaration de classe, la classe `HelloWorld` contient les données et les définitions de méthodes que votre programme utilise. Les classes contiennent généralement plusieurs méthodes. Les méthodes définissent le comportement de la classe. Toutefois, la classe `HelloWorld` ne possède qu'une seule méthode `Main`.
- La ligne suivante définit la méthode `Main`, qui est le point d'entrée de tous les programmes C#. La méthode `Main` indique ce que fait la classe lorsqu'elle est exécutée.
- La méthode `Main` spécifie son comportement avec l'instruction `Console.WriteLine("Hello World")` ; `WriteLine` est une méthode de la classe `Console` définie dans l'espace de noms `System`. Cette instruction provoque l'affichage du message "Hello, World !" à l'écran.

Il convient de noter les points suivants :

- C# est sensible à la casse.
 - Toutes les déclarations et expressions doivent se terminer par un point-virgule (;).
 - L'exécution du programme commence à la méthode `Main`.
-

Variables C#

Les variables en C# sont les mêmes que les variables en mathématiques.

En C#, une variable stocke une valeur d'un type de données spécifique. Elle peut stocker une valeur numérique, un caractère, une chaîne de caractères ou d'autres types de valeurs. Vous pouvez déclarer et affecter une valeur à une variable (comme `int x = 5`) ; où `int` est le type de données, `x` est le nom d'une variable, `=` est un opérateur qui affecte la valeur à une variable, et `5` est la valeur entière affectée à une variable `x`.

exemple de déclaration de variables :

```
int num = 100;
float rate = 10.2f;
decimal amount = 100.50M;
char code = 'C';
bool isValid = true;
string name = "Steve";
```

Déclarations conditionnelles

C# comprend les types d'instructions if suivants :

- if statement
- else-if statement
- else statement
- Switch

Essayons le code suivant :

```
int i = 10, j = 20;

if (i == j)
{
    Console.WriteLine("i is equal to j");
}
else if (i > j)
{
    Console.WriteLine("i is greater than j");
}
else if (i < j)
{
    Console.WriteLine("i is less than j");
}
```

```

int x = 10;

switch (x)
{
    case 5:
        Console.WriteLine("Value of x is 5");
        break;
    case 10:
        Console.WriteLine("Value of x is 10");
        break;
    case 15:
        Console.WriteLine("Value of x is 15");
        break;
    default:
        Console.WriteLine("Unknown value");
        break;
}

```

Boucle en C#

C# comprend les instructions de boucle suivantes :

- For Loop
- While Loop
- Do-while

Essayons les codes suivant :

```

for(int i = 0; i < 10; i++)
{
    Console.WriteLine("Value of i: {0}", i);
}

```

```

int i = 0; // initialization

while (i < 10) // condition
{
    Console.WriteLine("i = {0}", i);

    i++; // increment
}

```

```

int i = 0;

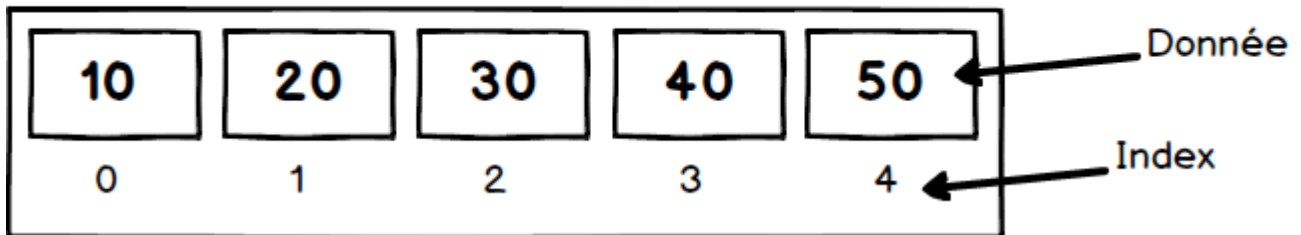
do
{
    Console.WriteLine("i = {0}", i);
    i++;
} while (i < 5);

```

Tableaux en C#

Une variable est utilisée pour stocker une valeur littérale, tandis qu'un tableau est utilisé pour stocker plusieurs valeurs littérales.

```
int[] tab = new int[] {10, 20, 30, 40, 50};
```



Exercice

- Ouvrir "<https://www.programiz.com/csharp-programming/online-compiler/>"
- Pratiquer C# :
 - Écrivez un programme pour afficher les premiers n nombres naturels et leur somme.
 - Écrivez un programme pour calculer la factorielle d'un entier.

Chapitre 3 : Unity

Unity est un moteur de jeu multiplateforme (smartphone, ordinateur, consoles de jeux vidéo et Web) développé par Unity Technologies. Il est l'un des plus répandus dans l'industrie du jeu vidéo, aussi bien pour les grands studios que pour les indépendants du fait de sa rapidité aux prototypages et qu'il permet de sortir les jeux sur tous les supports.

Il a la particularité de proposer une licence gratuite dite « Personal » avec quelques limitations de technologie avancée au niveau de l'éditeur, mais sans limitation au niveau du moteur. Le logiciel a la particularité d'utiliser du code (C#) sur la plateforme « .NET » avec l'implémentation Mono.

En quoi Unity est-il excellent ?

Les forces et les avantages d'Unity :

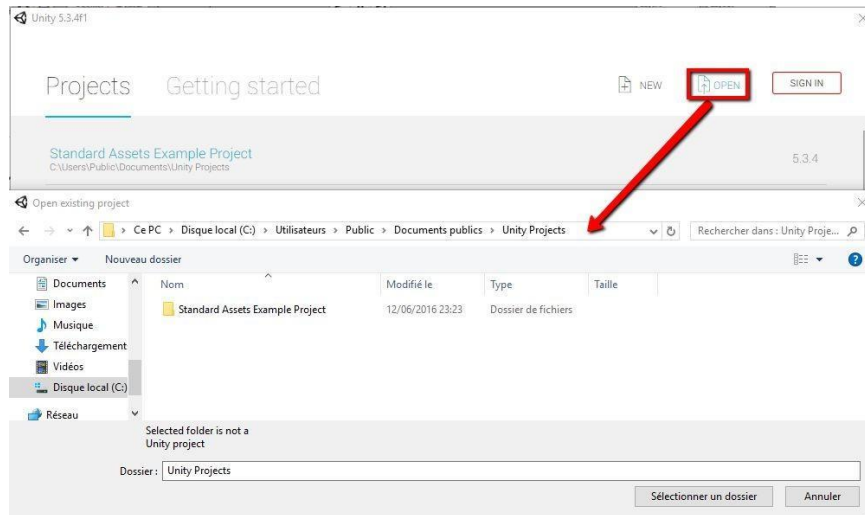
- C'est un outil professionnel de développement de jeux vidéos
- 30+ plateformes supportées
- Utiliser par une communauté vibrante de + 1 Million de développeurs
- Un outil accessible aux débutants
- Des milliers de fonctionnalités dont vos jeux bénéficieront : simulation physique, ombres dynamiques ...
- Un flux de travail visuel puissant supporté par un éditeur sophistiqué
- Un gain de productivité sans concession sur les dernières technologies de création de jeux vidéos
- Un éditeur customisable
- Basé sur un système de composant plutôt que d'héritage
- L'interaction avec les éditeurs pendant que le jeu s'exécute
- 100 % des plateformes VR/AR sont compatibles Unity

Les faiblesses :

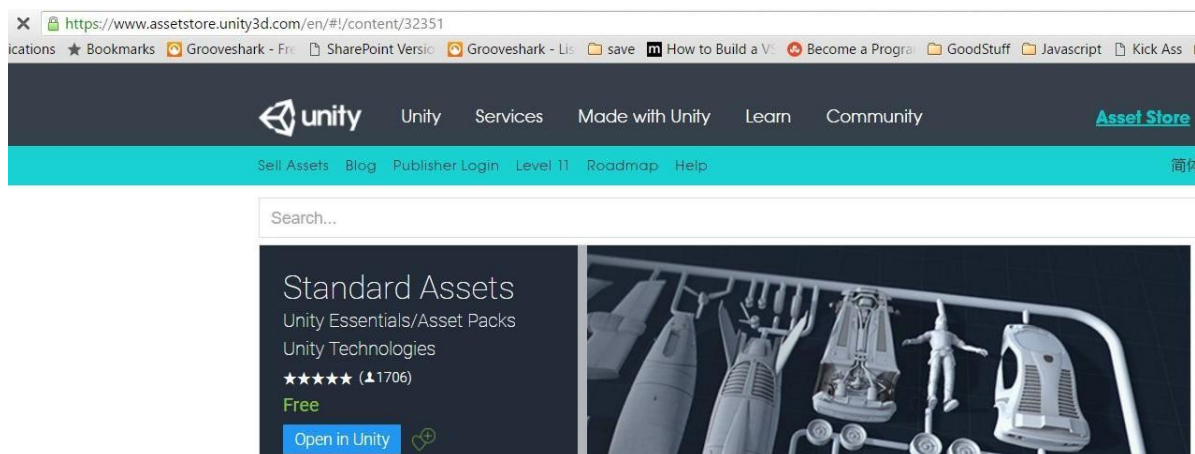
- Dans des scènes complexes, les interactions entre l'éditeur Unity et l'éditeur de code peut induire un problème de maintenance.
- Débogage visuel de l'éditeur
- Pas de support pour les bibliothèques dynamiques
- Support modéré pour les gestionnaires de sources

Comment utiliser Unity ?

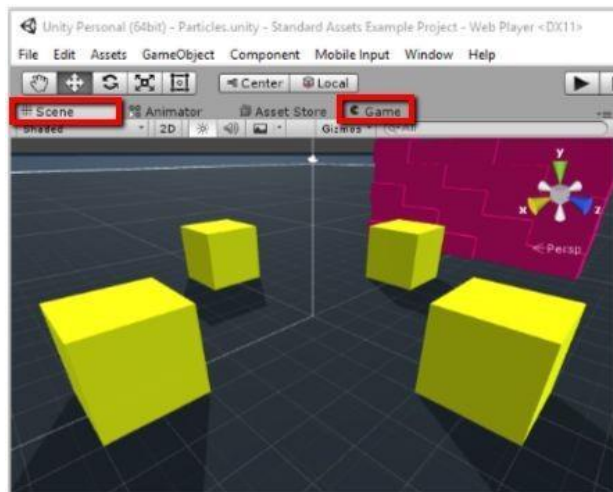
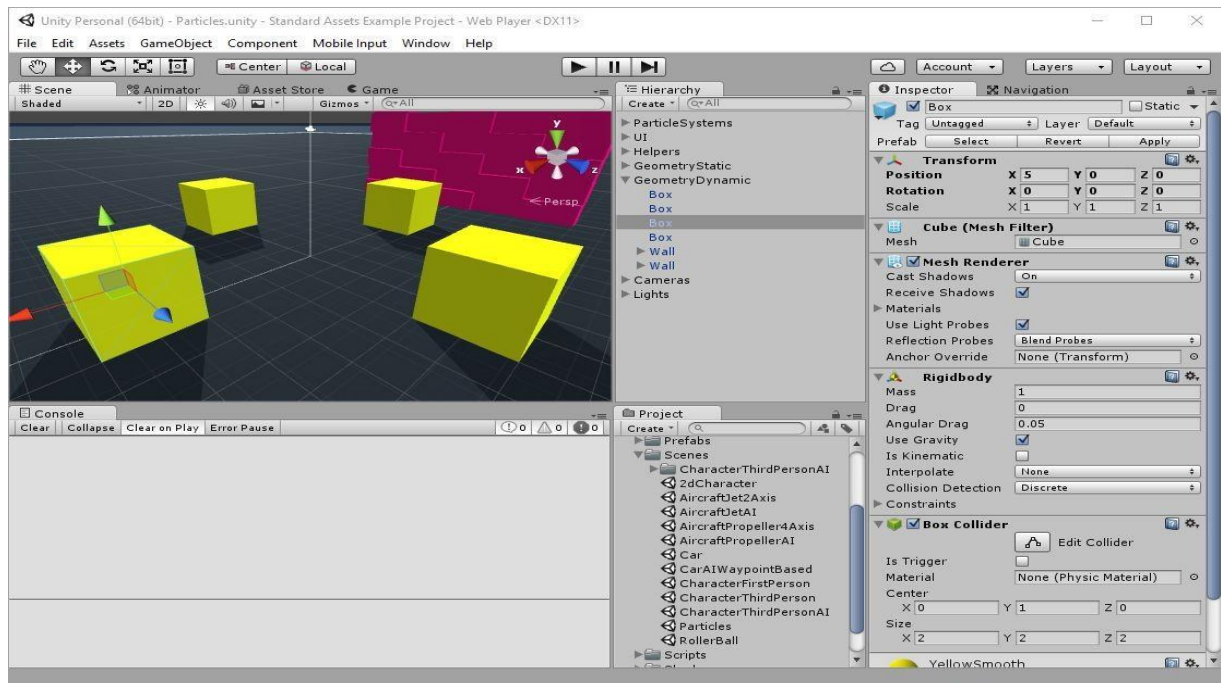
Ouvrir Unity et charger un projet depuis les exemples fournis avec l'éditeur



Si les exemples ne sont pas disponible, il est possible de les charger via la recherche Standard Assets dans l'Asset Store



Présentation des composants : éditeur :



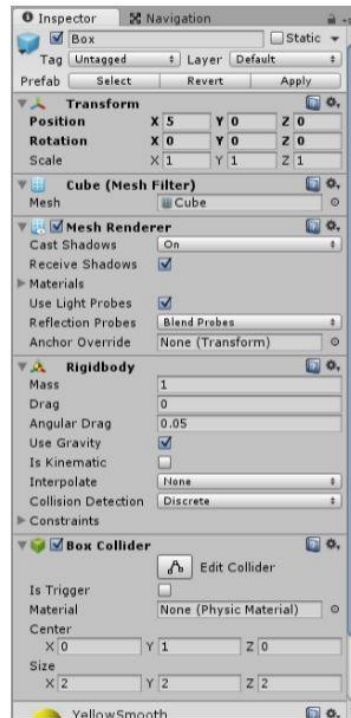
Les onglets Scene et Game permettent de visualiser la scène 3D et de lancer la simulation



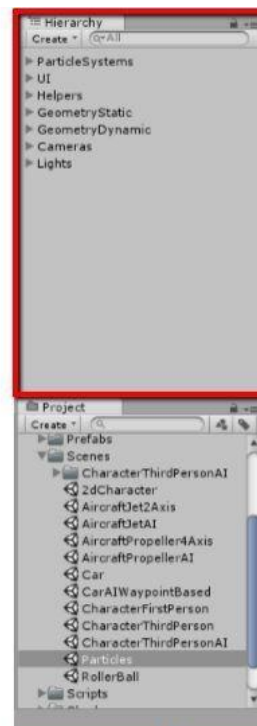
La barre d'outils à gauche sert à manipuler et à observer les objets de la scène

Les boutons du milieu contrôlent l'exécution de la scène

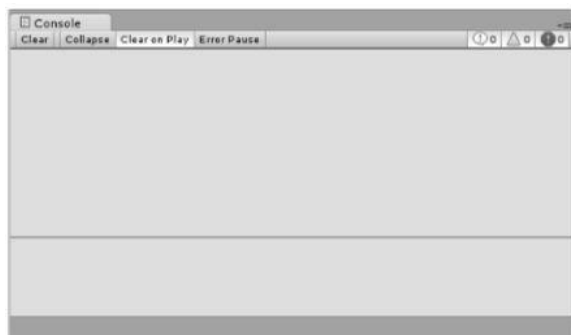
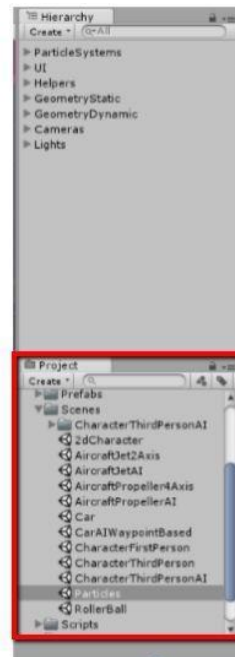
L'inspecteur affiche les informations concernant l'objet sélectionné. Les propriétés des composants attachés à l'objet sont également représentés.



La vue hiérarchique affiche la liste des objets dans la scène et leurs liaisons. Les éléments sont liés entre eux via drag & drop



L'onglet projet de voir les fichiers chargés dans le projet. Les scènes sont indiquées par une icône Unity



L'onglet Console permet la visualisation des messages du code, les avertissements et les erreurs Unity

La Vue :

- La vue Scene est l'objet principal de l'interface qui vous permettra de manipuler vos objets et de visualiser le monde de votre jeu
- Elle comporte également des éléments invisibles en mode jeu les gizmos (ancres de contrôles des objets), les caméras, les lumières et des indicateurs
- Vue Scene == Design Time
- La vue Game == Run time
- Elle représente l'expérience de l'utilisateur final
- En cliquant sur le bouton lecture et stop la vue passe automatiquement de Scene vers Game, vice et versa

La barre d'outils :

- Les boutons du bloc de gauche permettent :
 - La navigation dans la scène
 - Le déplacement des objets sélectionnés
 - La rotation des objets sélectionnés
 - Le redimensionnement des objets sélectionnés
 - La manipulation de propriétés en 2D

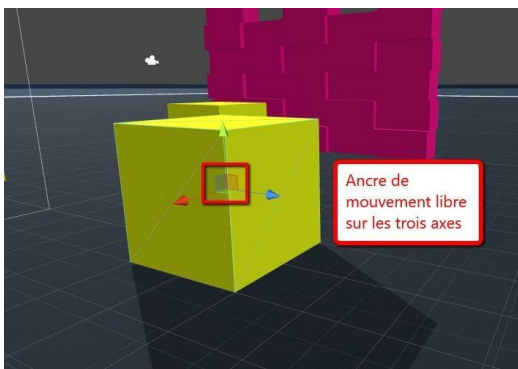


- Les boutons du bloc centrale permettent :
 - Le lancement et l'arrêt du jeu
 - La mise en pause permettant par exemple l'ajustement de variables pendant l'exécution du jeu
 - La mise en pause en repassant par la vue Scene

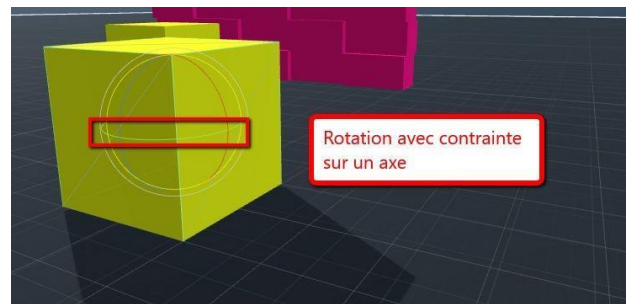
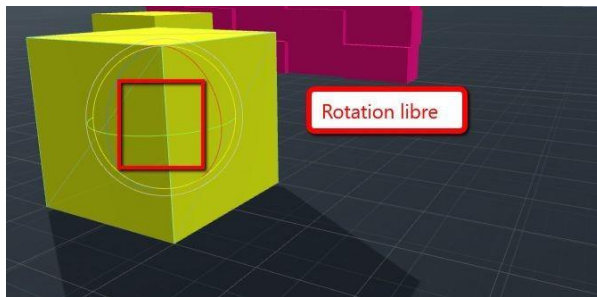


Utiliser la souris et le clavier :

- L'outil de manipulation de scène permet le déplacement du point de vue dans la scène
 - En translation
 - En orbite du point observé
 - En zoom sur le point observé
- Préférer une souris à trois boutons
- Les touches Alt et Ctrl modifient ces comportements
- L'outil Translation permet le déplacement d'un objet en mode libre ou en contraignant le mouvement sur une dimension uniquement



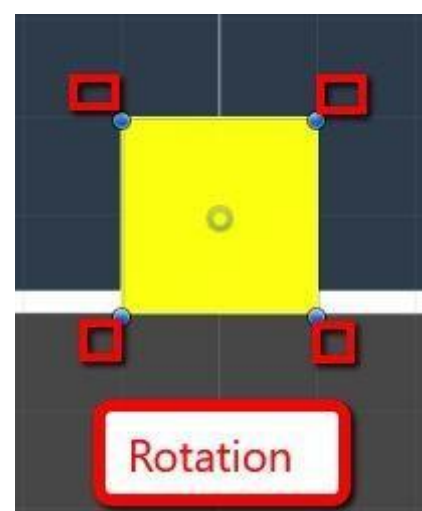
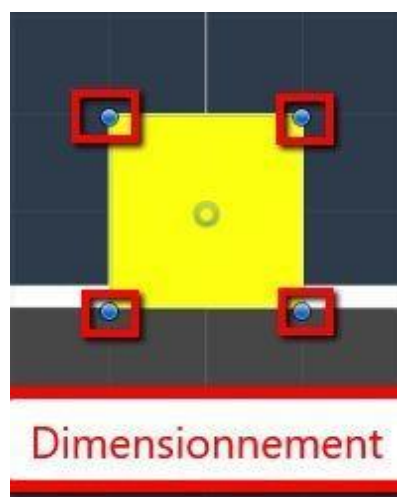
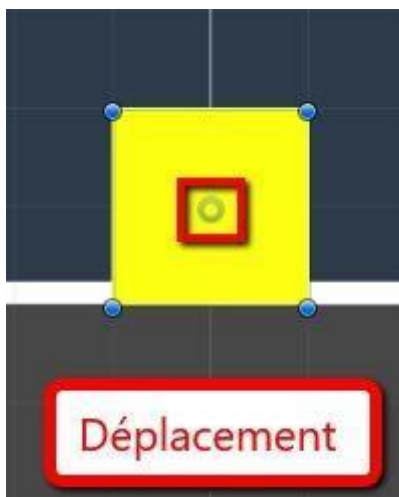
- L'outil Orbite permet la rotation d'un objet en mode libre ou en contraignant le mouvement sur une dimension uniquement



- L'outil de mise à l'échelle permet le redimensionnement d'un objet sur tous les axes ou en contraignant le redimensionnement sur un seul axe uniquement



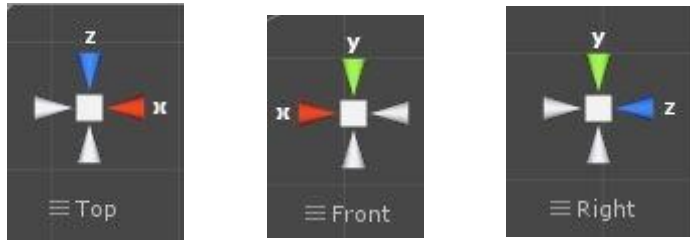
- L'outil Rect combine les outils déplacement, rotation et redimensionnement et facilite ces transformations en mode 2D



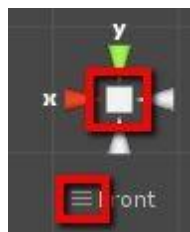
- Les touches W, E, R permettent de passer directement aux outils de déplacement, rotation et redimensionnement.
- Le menu Edit > Preferences > Keys permet la programmation des raccourcis
- L'utilisation de raccourcis permet un gain de productivité
- Les indicateurs de l'interface graphique (axes, poignées, cercles) sont surnommés Gizmos

Le gizmo des axes :

- Permet de tourner la camera pour entrer en mode plan en cliquant sur un axe



- En cliquant sur le centre on rentre/sort en mode orbite autour de l'élément sélectionné



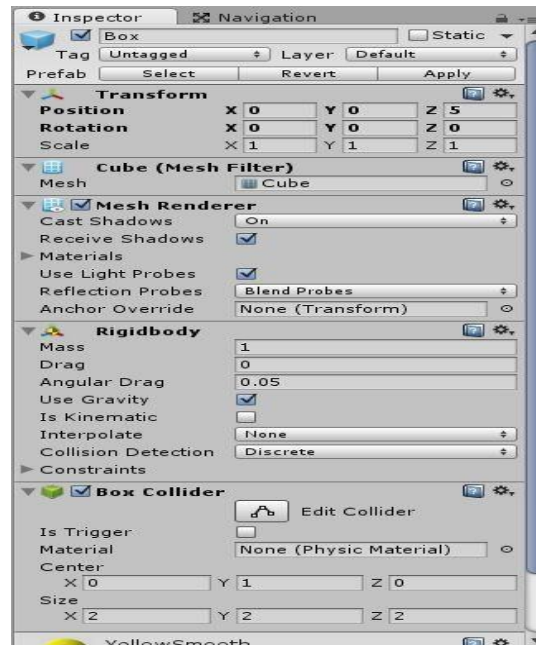
L'onglet Hierarchy :

- L'onglet hiérarchie est une liste des objets présents dans la scène
- Il maintient un référentiel par nom
- Les objets sont groupés en hiérarchies et liés ce qui facilite leur maintenance
Ex : transformations groupées



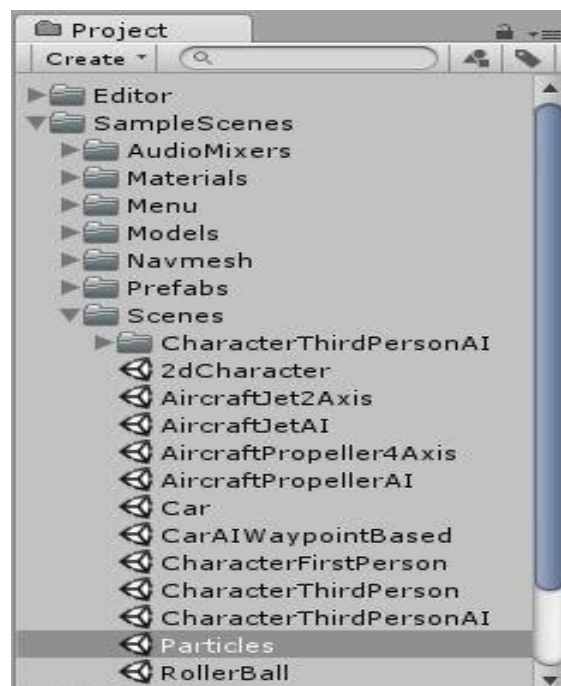
L'onglet Inspector :

- L'onglet inspecteur affiche les informations de l'objet(s) sélectionné(s)
- Les composants affectés à l'objet sont également affichés



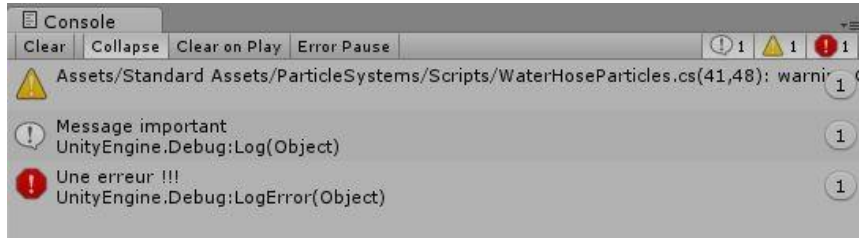
L'onglet Project :

- Affiche les dossiers et les fichiers du projet
- Permet le chargement / navigation entre différentes scènes Unity



L'onglet Console :

- Affiche les messages de débogages
- Affiche les avertissements et les erreurs rencontrées lors de l'exécution du jeu

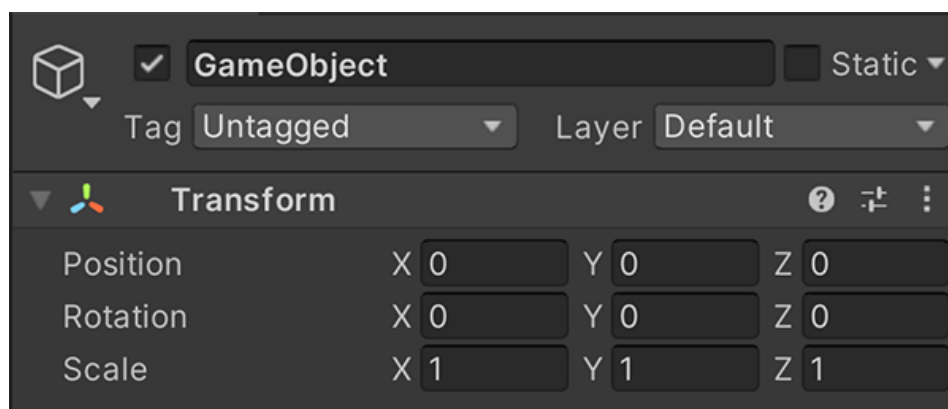


Composants

Les composants sont les pièces fonctionnelles de chaque objet de jeu. Les composants contiennent des propriétés que vous pouvez modifier pour définir le comportement d'un objet de jeu.

Pour afficher une liste des composants attachés à un objet de jeu dans la fenêtre de l'inspecteur, sélectionnez votre objet de jeu. sélectionnez votre GameObject dans la fenêtre Hiérarchie ou la fenêtre Scene ou dans la fenêtre Scène.

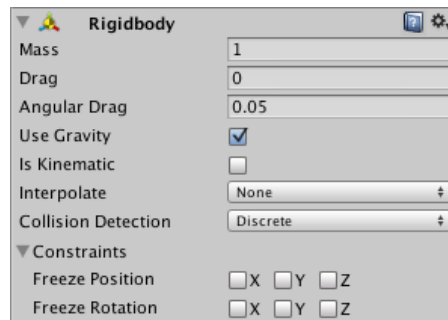
Un GameObject peut avoir de nombreux composants, mais chaque GameObject possède un composant Transform. En effet, la transformation détermine l'emplacement, la rotation et l'échelle du GameObject. Pour créer un GameObject vide, sélectionnez GameObject > Create Empty. Lorsque vous sélectionnez le nouveau GameObject, l'inspecteur affiche le composant Transform avec les valeurs par défaut.



Autres composants

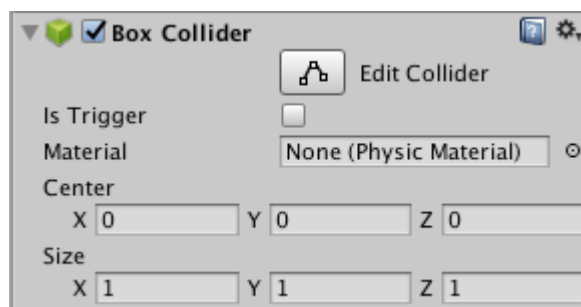
- **Rigidbody**

Un Rigidbody est le composant principal qui permet le comportement physique d'un GameObject. Avec un Rigidbody attaché, l'objet répondra immédiatement à la gravité. Si un ou plusieurs composants Collider sont également ajoutés, le GameObject est déplacé par les collisions entrantes.



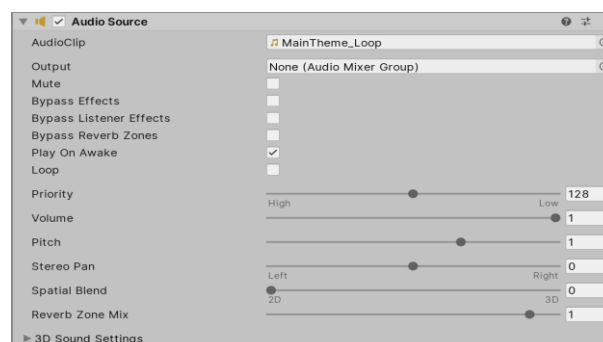
- **Colliders**

Les composants Collider définissent la forme d'un objet à des fins de collisions physiques. Un collider, qui est invisible, ne doit pas nécessairement avoir exactement la même forme que le maillage de l'objet. En fait, une approximation grossière est souvent plus efficace et indiscernable dans le jeu.



- **Audio**

Un jeu serait incomplet sans une certaine forme d'audio, qu'il s'agisse de musique de fond ou d'effets sonores. Le système audio d'Unity est flexible et puissant.



Exercice :

- Ouvrir un projet de test
- Pratiquer l'unité (La Vue, La barre d'outils, Utiliser la souris et le clavier, Les onglets, Le gizmo, composants)

Introduction aux Standard Assets

Dans Unity, les Standard Assets sont des éléments prédéfinis et génériques utilisés pour faciliter le développement d'une application. Ces actifs peuvent inclure des modèles 3D, des textures, des animations, des scripts, des effets visuels, et plus encore. L'utilisation d'actifs standard permet de gagner du temps lors de la création de jeux en offrant des ressources prêtes à l'emploi pour intégrer facilement des éléments dans un projet.

Utilisation des Standard Assets pour les Modèles et Animations Existants

Les Standard Assets peuvent être utilisés pour améliorer les modèles et les animations existants dans un projet. Par exemple, des modèles 3D génériques fournis par Unity peuvent être intégrés à des modèles plus complexes ou personnalisés créés par les développeurs. Cela permet d'ajouter des éléments de décor, des personnages ou des objets d'interface rapidement sans devoir les modéliser ou les animer manuellement. Les animations des actifs standard peuvent également être appliquées aux modèles existants. Cela peut inclure des animations de marche, de course, de saut ou d'interaction qui sont prêtes à l'emploi. Par conséquent, les développeurs peuvent se concentrer sur la logique du jeu plutôt que sur la création d'animations de base.

