# BO - HUB

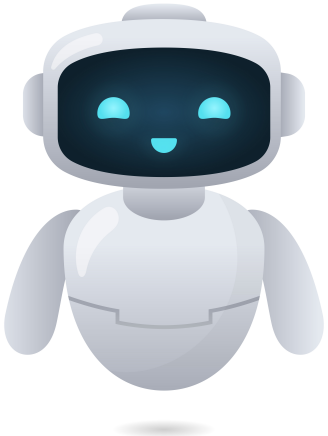
B-INN-000

# MIA

Minecraft IA

## - Minecraft -

**Minecraft** est un jeu vidéo sandbox développé par Mojang Studios et initialement créé par *Markus "Notch" Persson*. Il a été officiellement publié le **18 novembre 2011**. Le jeu offre une liberté quasi totale aux joueurs pour explorer, construire et interagir avec un monde généré procéduralement composé de blocs cubiques représentant différents matériaux comme la terre, la pierre, les minerais, l'eau, et bien plus.

# M-IA

Aujourd'hui, vous allez créer une **Intelligence Artificielle** (IA), c'est-à-dire une machine reproduisant l'intelligence humaine.
Pour cela, vous allez avoir besoin de **Minecraft** (et de votre compte Microsoft), **Node JS** (langage de programmation) et un **environnement de développement** (IDE tel que VSCode).
Pour programmer notre IA, nous allons utiliser une **API**.

*Une **API**, pour "Application Programming Interface" en anglais, est un ensemble normalisé de classes, de méthodes, de fonctions et de constantes qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels. Elle est offerte par une bibliothèque logicielle ou un service web, le plus souvent accompagnée d'une description qui spécifie comment des programmes « consommateurs » peuvent se servir des fonctionnalités du programme « fournisseur ».*

## Setup

Avant de commencer à programmer, vous devez installer **Node JS** et **Visual Studio Code** si ce n'est pas déjà fait !

- Lien de téléchargement de Node JS
- Lien de téléchargement de VSCode

Maintenant vous pouvez créer un **dossier** où vous voulez en le nommant `MIA`. Ouvrez-le avec **Visual Studio Code**.
Il va falloir initialiser votre fichier avec **Node JS** en utilisant la commande ci-dessous dans un **terminal**.

```
Terminal
~/B-INN-000> npm init
```

Si vous rencontrez un problème durant les téléchargements, veuillez demander à un **Cobra**

# Tutoriels Minecraft

## Tuto n°1 - Minecraft

Le Tutoriel n°1 est réservé pour les participants possédants **Minecraft**. Si vous n'avez pas **Minecraft**, suivez le Tuto n°2 - Web Client.

Pour réaliser ce **Coding Club**, vous allez avoir besoin de **Minecraft**. Vous devez donc **lancer le jeu** en utilisant la version **1.20**.
**Comment rejoindre le serveur ?**

- Lancez une partie en **Multijoueur** (ou "***Multiplayer***" en anglais).
- **Rejoignez un nouveau serveur** en cliquant sur **Nouveau serveur**.
- Ajoutez **l'adresse IP du serveur** donnée par les **Cobras**.
- Connectez-vous.

Et voilà, vous êtes connecté !

## Tuto n°2 - Web Client

Le Tutoriel n°2 est réservé aux participants ne possèdants pas **Minecraft**. Si vous avez déjà le jeu et réalisé le Tuto n°1 - Minecraft, alors passez à l'étape suivante.

Si vous ne possedez pas **Minecraft**, ce n'est pas grave. Vous allez à la place utiliser un **Web Client** !
*Un **Web Client** est tout simplement un ordinateur ou une application qui envoie des requêtes à un serveur depuis Internet directement. Donc pas d'installation de Minecraft, n'est-ce pas génial ?*

Cliquez ici pour accéder au Web Client

Pouf ! Vous êtes sur Minecraft !
Maintenant, vous pouvez vous **connecter à un serveur**. Cliquez sur **Connect to server** puis **Add** afin d'ajouter un nouveau serveur.

Suivez les étapes suivantes :

- Ajoutez l'**adresse IP du serveur** donnée par les **Cobras**.
- Entrez la version **1.20** dans "**Version Override**".
- Entrez un pseudonyme (ou pseudo) dans "**Username Override**".
- Laissez les autres options vides.
- **Connectez-vous** !

Et voilà, vous êtes connecté !

## CREATION DU BOT.

Très bien, nous pouvons commencer à **programmer** notre IA :

- Retournez sur **Visual Studio Code**.
- Créez un fichier nommé `index.js`.

Avant d'écrire du code, vous allez devoir installer un **module** c'est-à-dire l'**API** que l'on va utiliser qui va intéragir avec Minecraft.
Exécutez la commande suivante dans le **Terminal** de VSCode.

```
▽                          Terminal                          - + x
~/B-INN-000> npm install mineflayer
```

Très bien, nous pouvons enfin programmer dans le fichier `index.js`.
Tout d'abord, nous devons importer notre module `mineflayer`.
Vous allez donc pouvoir écrire la ligne ci-dessous dans votre fichier `index.js`.

```
const mineflayer = require('mineflayer');
```

La syntaxe `const` permet de définir une **constante**, c'est-à-dire **une variable** qui ne peut pas être modifiée. La variable est nommée `mineflayer` afin d'utiliser le module.

Pour créer le Bot, on va également utiliser une **constante**.

```
const bot = mineflayer.createBot({
    host: 'IP DU SERVEUR',
    username: 'NOM DE VOTRE BOT'
});
```

Ne mettez pas tous le même nom du Bot sinon il ne pourra pas se connecter. **Trouvez des noms originaux !**

# Evenements

Pour que le Bot effectue une action, il faut le contraindre à un **événement**.
*Un **événement** est une action ou une occurrence qui est reconnue par un système logiciel*

## Evenement n°1 - Messages

Notre premier événement sera un **message**, c'est-à-dire que lorsqu'on va envoyer un message dans le chat du jeu, le Bot va exécuter une action.

```
bot.on('chat', (username, msg) => {
    if (username === bot.username) return;
    if (msg === "Bonjour")
        return bot.chat(`Bonjour ${username} !`);
});
```

Dans ce programme, le Bot répond à un utilisateur, autre que lui, qui lui envoi "*Bonjour*".
Vous savez désormais comment envoyer un message avec le Bot en créant un événement.
Maintenant on va complexifier les événements.

## Evenement n°2 - Le Temps

Maintenant, on va utiliser un événement appelé `physicsTick`. Cet événement consiste à exécuter une action du bot à chaque Tick (c'est-à-dire toutes les 0.05 secondes).
Avec cet événement, le bot pourra regarder le joueur le plus proche.

*Vous devez remplir les "..." grâce aux indications :*

```
bot.on('...', () => {
    // Creation du filtre pour chercher uniquement un joueur (player).
    const filter_entity = (entity) => entity.type === '...';
    // On recupere le joueur le plus proche.
    const player = bot.nearestEntity(filter_entity);

    if (player) {
        // On recupere la position de la tete du joueur
        const pos = player.position.offset(0, player.height, 0);
        // On demande au bot de regarder a une position.
        bot.lookAt(...);
    }
});
```

*Le listener de cet événement ne possède pas de paramètres d'où le fait qu'on ne mette rien entre les parenthèses.*

## Le saviez-vous ?

Parmi les autres créatures incontournables de Minecraft, on trouve les **Endermen**, ces grands hommes noirs aux yeux violets. Si vous les avez déjà rencontrés en jeu, vous avez dû être surpris par leur langage incompréhensible.
Et pourtant, derrière cette langue mystérieuse, quelques éléments vous sont familiers si vous parlez anglais. Certaines de leurs exclamations comme "*here*", "*what's up*" et "*this way*" sont en réalité des mots anglais mais prononcés à l'envers.
Vous pouvez donc maintenant parler avec les **Endermen**, mais pour comprendre le reste de leur langue, un dictionnaire serait le bienvenu.

## MIA-TTAQUE

Maintenant que vous avez à-peu-près compris comment fonctionne votre Bot, il est temps de révéler quel est notre véritable but avec celui-ci : un Bot PvP (on va se battre !!!).

***Qu'est-ce que le PvP ?***
*Le Player versus Player, abrégé en PvP (ou JcJ en français), est un terme lié aux jeux vidéos/de rôles. Il décrit un mode de jeu permettant à des joueurs de s'affronter les uns contre les autres.*

Pour cela nous allons avoir besoin de 2 nouveaux modules à installer donc 2 commandes à exécuter dans votre **Terminal** (trop bien).
*Ces modules sont des **plugins** de **Mineflayer**.*

```
Terminal
~/B-INN-000> npm install mineflayer-pvp
```

```
Terminal
~/B-INN-000> npm install mineflayer-armor-manager
```

Comme au début, on va initaliser 2 **constantes** en dessous de la première.

```
const pvp = require('mineflayer-pvp').plugin;
const armorManager = require('mineflayer-armor-manager');
```

Puis en dessous de l'initialisation du Bot, vous devez charger les **plugins** (pvp et armorManager) :

```
bot.loadPlugin(pvp);
bot.loadPlugin(armorManager);
```

### MODE PVP

Nous allons ici activer le **mode PvP** du bot en utilisant le **plugin** `pvp`.
Nous allons donc créer une fonction appelée `attackPlayer(bot)`. Pour définir une fonction utilisez la syntaxe suivante:

```
function attackPlayer(bot) {
    // Filtrer les entites pour que ce soit uniquement un joueur.
    const filter = ...
    // Recuperer le joueur le plus proche.
    const player = ...

    if (player) {
        // Attaquer le joueur.
        bot.pvp.attack(...)
    }
}
```

*Completez les "..." en utilisant ce que vous avez vu précédemment.*

Maintenant on peut **appeler** la fonction `attackPlayer(bot)` dans l'événement `chat` quand le message `attack` est reçu.

*Compléter en utilisant l'événement `chat` précédente à partir des "...".*

```
bot.on('chat', (username, msg) => {
    if (username === bot.username) return;
    if (msg === "Bonjour")
        return bot.chat(`Bonjour ${username} !`);
    ...
});
```

Pour **désactiver** le **mode PvP**, vous devez créer la fonction `stopAttack(bot)` en utilisant la même syntaxe que précédemment.
Dans cette fonction vous allez utiliser l'**expression** suivante : `bot.pvp.stop()`.
Comme pour l'activation, vous allez utiliser l'événement `chat` afin d'arrêter le **mode PvP**. Ainsi, lorsque le bot recevra le message `stop`, il devra **exécuter** la fonction `stopAttack(bot)` (exactement comme pour l'**activation**).

# Fin - Missions Bonus

Bien. Si vous êtes allé jusque là c'est que vous êtes trop fort ! (ou que je sais pas faire des sujets complets). Mais ne vous inquiétez pas, j'ai toujours quelques petits bonus pour vous.

Psst ! Vous allez sûrement avoir besoin des documentations des différents modules, n'hésitez pas à les utiliser !

**Liens des documentations :**
Mineflayer Documentation
PvP Plugin Documentation
Armor Plugin Documentation

## Mission Bonus n°1 : Equipement

Vous avez sûrement remarqué que le **plugin** nommé `Armor Manager` n'a jamais été utilisé. Alors c'est à vous de l'utiliser et de faire des recherches sur les **Documentations** de ***Mineflayer*** et du plugin `Armor Manager` (liens des documentations ci-dessus).
Vous allez devoir intégrer dans la fonction `attackPlayer(bot)`, les expressions afin d'équiper le Bot comme vous le souhaitez.

## Mission Bonus n°2 : Creation

Si vous êtes ici c'est que vous avez compris comment fonctionnent les API. Mais maintenant c'est à votre tour de continuer votre propre Bot en **créant** et en **innovant**. Il est possible de faire tout et n'importe quoi avec cette API ainsi que ses plugins grâce à la grandeur du monde cubique.

## Le saviez-vous ?

Les chercheurs utilisent **Minecraft** afin de développer des IA car le jeu permet d'exposer les IA à des **tâches complexes** qui imitent les défis du monde réel, comme la **collecte de ressources**, la **construction de structures**, et l'**adaptation à des environnements changeants**.