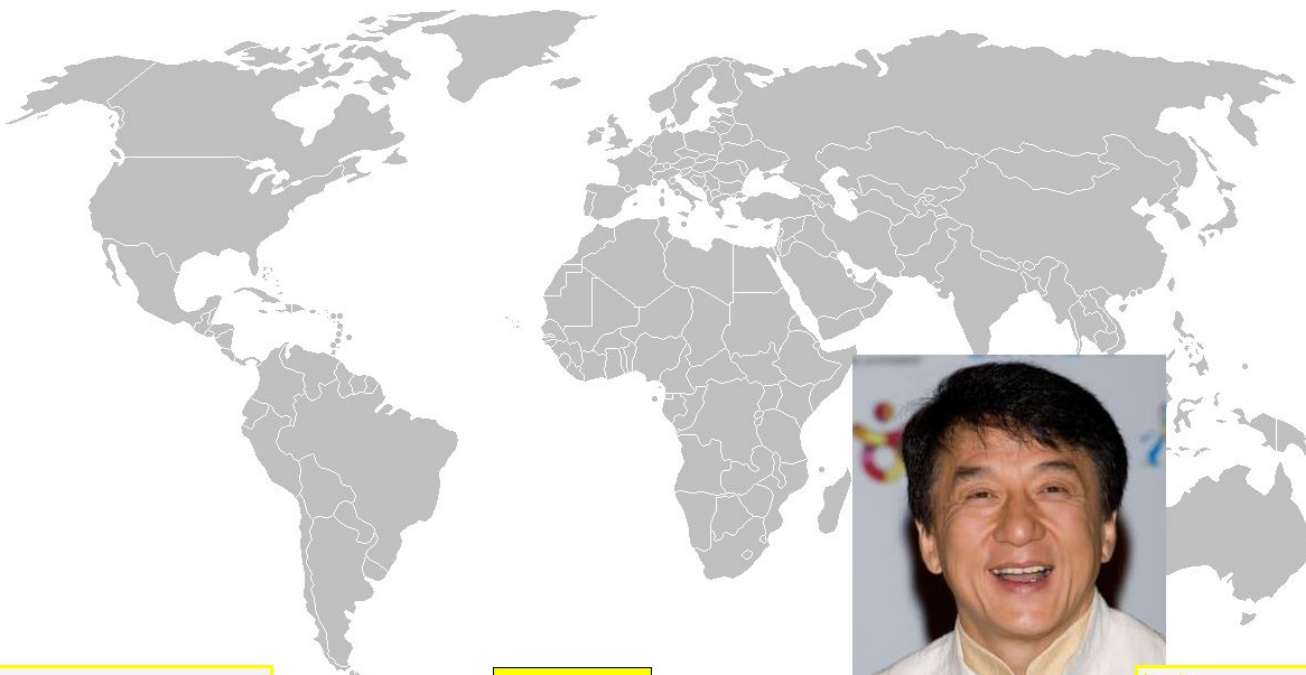


# JAUNE

JAUNE



AI  
Score : 6  
❤️ ❤️ ❤️

Not a person

kevin  
Score : 7  
❤️ ❤️ ❤️ ❤️

Raphaël OLIVIER  
Mathieu GOURICHON  
EISE4

## Introduction

---

L'intelligence artificielle est un secteur en pleine expansion, qui suscite depuis plusieurs années l'intérêt du plus grand nombre. C'est un domaine soumis à de nombreuses expériences plus ou moins justifiées, ce qui en fait (à juste titre), un secteur souvent confronté à la question de l'éthique.

Ce projet a pour but, de dénoncer certaines expériences et recherches mises en place à l'aide de l'intelligence artificielle, et plus que questionnable.

L'intelligence artificielle est fréquemment utilisée pour faire de la reconnaissance faciale. Le but est d'arriver à faire reconnaître à un programme une personne, en lui montrant simplement une photo de celle-ci. Le programme fonctionne grâce au « machine learning » de la manière suivante ; lorsque l'on lui donne une photo, il repère certains « critères », et les comparent aux photos qu'on lui a déjà montrés possédant ces mêmes critères. Jusque-là, rien d'offensant. Ce qui pose question sont des projets basés sur cette reconnaissance faciale dont le but est de reconnaître l'origine, la nationalité d'un individu. Car dans ce cas, cela pose la question suivante, quelles sont ces fameux « critères » qui permettraient de différencier différentes populations et que sont-ils si ce n'est des critères racistes. C'est ce que tente de dénoncer ce projet, intitulé « Jaune », ou une (fausse) intelligence artificielle tente de reconnaître différentes personnalités pour la plupart d'origine asiatique, à grand coup d'amalgames et de racisme.

Pour plus d'informations sur le sujet :

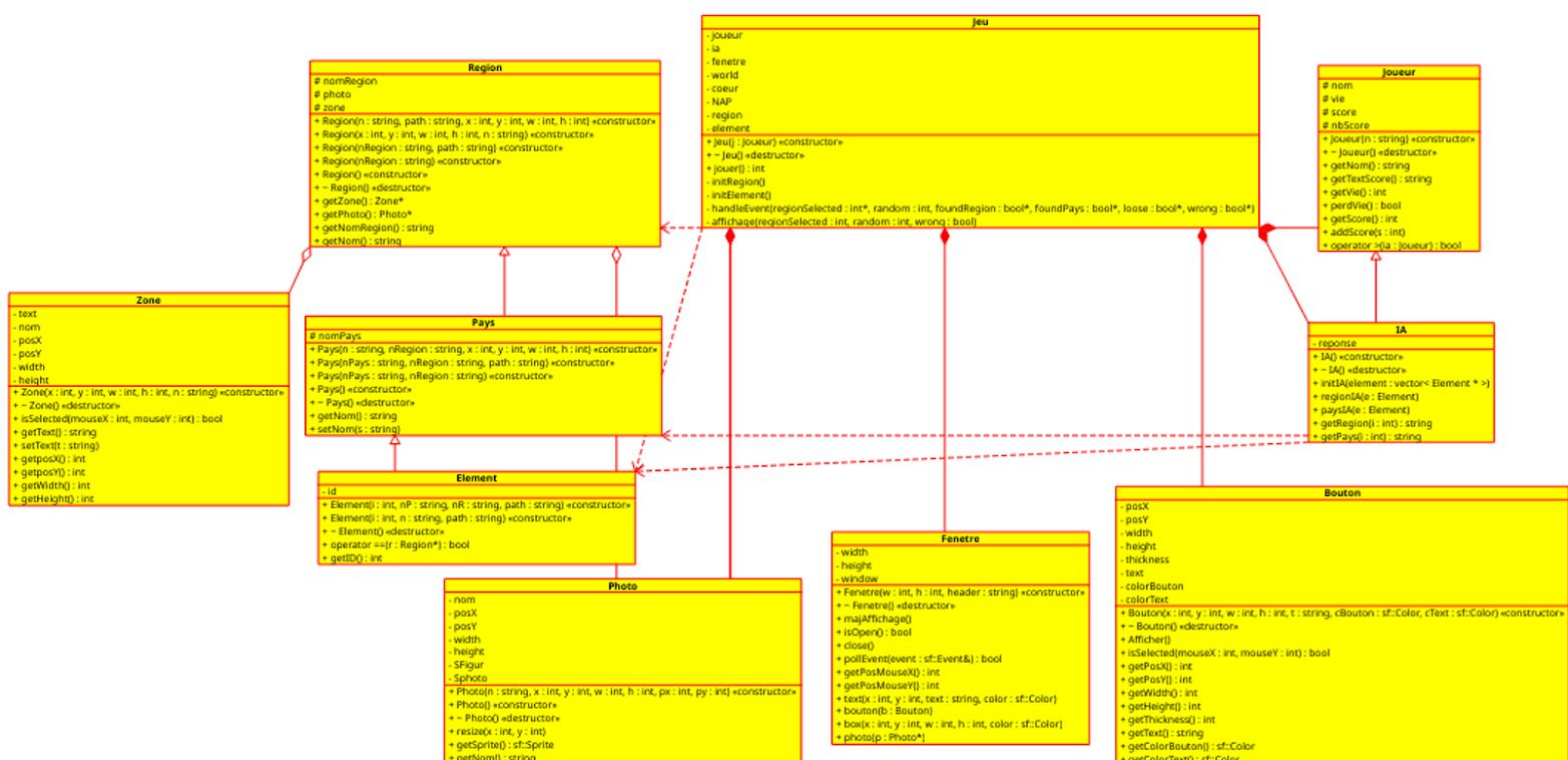
<https://towardsdatascience.com/is-artificial-intelligence-racist-and-other-concerns-817fa60d75e9>

<https://theoutline.com/post/4635/a-new-ethnicity-recognition-tool-is-just-automated-racial-profiling?zd=2&zi=54jf6ico>

## Déroulement

Au lancement de l'application, on retrouve une carte du monde divisée en plusieurs zones représentant les différentes régions du monde (Amérique du Nord, Europe, ...), ainsi qu'une photo d'une personne dont on est censé trouver la nationalité. Dans un premier temps, on doit déjà trouver la région du monde dont est originaire cette personne, et en cas de succès, le pays (ou une zone géographique plus précise). Trouver la région rapporte un point, le pays deux points, et chaque erreur coûte une vie.

## UML



- Classe Jeu :

C'est la classe principale du projet, elle gère le déroulement du programme en récupérant les événements et en commandant l'affichage (grâce à la classe fenêtre).

- Classe Fenêtre :

Elle permet de créer une fenêtre et plusieurs types graphiques, et de les afficher.

- Classe Bouton :

Permet de créer un bouton, possède une méthode afin de savoir si le bouton est appuyé.

- Classe Zone :

Permet de créer une « zone », possède une méthode afin de savoir si la zone est sélectionnée.

- Classe Photo :

Permet de créer un objet lié à une photo dont on peut définir la taille et la position, facilite l'affichage via la classe fenêtre

- Classe Joueur :

Créer un joueur avec un nom, un score, et des points de vie.

- Classe IA (hérite de joueur) :

Créer une IA avec les mêmes paramètres qu'un joueur et possède en plus des réponses prédéfinies.

- Classe Région :

Créer une « région », c'est-à-dire une zone avec une photo d'une région du monde (exemple : Europe).

- Classe Pays (hérite de Région) :

Créer un pays avec une zone et est lié à une région.

- Classe Élément (hérite de Pays) :

Créer un « élément », possédant une image et est lié à un pays et donc une région. Possède également un « ID » afin de pouvoir choisir chaque élément facilement.

## Contraintes de développement

---

- 8 classes : notre application en possède 10 : Jeu, Région, Pays, Eléments, Fenêtre, Bouton, Photo, Zone, Joueur, IA.
- 3 niveaux de hiérarchie :

Région -> Pays -> Eléments

Pays dérive de Région ce qui permet d'associer chaque pays avec sa région et facilite la création et la manipulation des objets dans la classe jeu puisqu'un seul vecteur possède toutes les régions et pays.

Eléments dérive de Pays ce qui permet de lier chaque élément à son pays et sa région respective. De plus, la classe région possède un objet photo indispensable pour l'affichage ce qui évite d'avoir à en recréer un dans la classe élément.

- 2 fonctions virtuelles :

La classe Région possède une fonction virtuelle « `getNom()` » redéfinie dans sa classe fille pays. Cela permet au vecteur région créée dans la classe jeu (composé en réalité de région et de pays) d'accéder facilement au nom souhaité sans avoir à se soucier si c'est un pays ou une région. (Exemple, pour accéder au nom du pays stocké dans le vecteur région en `i`, on aura juste à demander : `region[i].getNom()` ).

- 2 surcharges d'opérateurs :

Dans la classe Elément : `bool operator == (region *r).`

Il permet de comparer facilement le nom de la région appartenant à un élément à une région et simplifie donc le code du déroulement du jeu.

Dans la classe Joueur : `bool operator > (Joueur ia).`

Il permet de comparer facilement si le joueur à battu l'IA et allège donc le code du déroulement du jeu.

- 2 conteneurs différents :

Dans la classe IA : `map<int, Pays *>` reponse.

Cette map permet d'associer pour chaque élément (grâce à son ID stocké dans l'int) la réponse que donnera l'IA. En effet le pays créé ici correspond à la réponse de l'IA pour tel élément.

Le choix d'utiliser une map revient dans le fait qu'elle permet de lier facilement deux objets (ici un int et un pays) et que, de plus, il est très facile d'accéder à un élément en particulier d'une map.

Dans la classe Jeu : `vector< vector<Region*> >` region.

Ce vecteur de vecteur (comme un vecteur à deux dimensions) permet de créer l'équivalent d'un tableau à deux dimensions où la première colonne correspond au nom de la région et où, pour chaque ligne on retrouve les différents pays de cette région. L'avantage par rapport à un tableau classique est que chaque ligne a une taille différente.

Le choix d'utiliser un vecteur revient dans le fait qu'il est beaucoup plus facile d'accéder un élément en particulier, contrairement par exemple à une liste. Il se présente donc sous la forme suivante :

Amérique du Nord	Canada	USA	Mexique		
Afrique	Nord	Centre	Moyen-Orient	Sud	Madagascar
Océanie	Australie	Nouvelle-Zélande			

## Procédure d'installation

L'application requiert seulement la bibliothèque SFML, utilisé pour la partie graphique.

## Implémentation particulière

---

La fonction « `isZoneSelected()` » de la classe `Zone`, car outre le fait qu'elle renvoie si la zone a été sélectionnée ou non, elle change le nom de celle-ci dès que la souris est dans la zone, ce qui, à l'affichage, affiche le nom des pays ou des régions dès que la souris se trouve dans leur zone. Elle facilite de plus le code du déroulement du jeu car on a donc juste à faire appel à cette fonction, on se passe donc d'une fonction du type « `isMouseInZone()` » et cela allège fortement le code.

Le vecteur deux dimensions de la classe `Jeu` (présenté ci-dessus), car il permet de parcourir facilement les régions/pays sans se soucier des problèmes de taille.

L'initialisation des éléments et de l'IA via des fichiers externe, car cela permet de facilement modifier les éléments du jeu, d'en rajouter, ou de modifier les paramètres de l'IA.

La méthode `perdVie()` de la classe `Joueur` qui enlève une vie au joueur et retourne vraie si le joueur n'a plus de vie. Cela permet d'alléger le code du déroulement du jeu dans la fonction `handleEvent()` (voir classe `Jeu`).

Le traitement aléatoire des éléments dans la classe `Jeu`. On choisit un élément arbitrairement dans le vecteur "element" afin de ne pas avoir tout le temps le même ordre. Une fois qu'on a "joué" avec cet élément, il est supprimé du vecteur et on a juste à choisir à nouveau arbitrairement dans le vecteur sans risque de tomber deux fois sur le même. Cela facilite vraiment le traitement aléatoire des éléments. De plus il est facile de savoir si l'on a utilisé tous les éléments à notre disposition car il nous suffit d'interroger la taille du vecteur.