

# Rapport de projet

## Pixel war

AI30 - Systèmes Multi-Agents

# Sommaire

01

Reddit et  
le r/place

02

Formalisation  
du sujet

03

Modélisation  
des agents

04

API REST

# 01

Reddit et  
le r/Place

# Qu'est ce que reddit

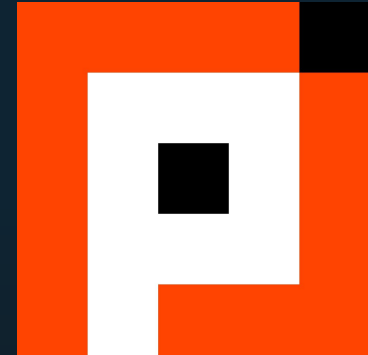
- Site communautaire américain de discussion et d'actualités sociales
- Site divisés en sous-parties ("subreddits") consacrées à des thèmes spécifiques
- Subreddit : r/Subreddit
- Chaque post peut être "upvoté" ou "downvoté"



reddit

## Et le r/Place ?

- Projet collaboratif apparu lors du 1er avril 2017
- Toile numérique où chaque utilisateur peut changer la couleur d'un pixel
- Deux éditions : 2017 (1000x1000 pixels) et 2022 (2000x2000 pixels)
- Créations artistiques et collaboratives
- Pourquoi une guerre ?  $\Rightarrow$  France vs. coalition Espagne - Etats-Unis - Canada
- Défense drapeau et attaque territoire ennemi









# Problématique

**Comment modéliser le comportement des agents ayant participé à la Pixel War ?**

A decorative background on the left side of the slide consisting of a grid of blue squares in various shades of blue, arranged in a pattern that tapers off to the right.

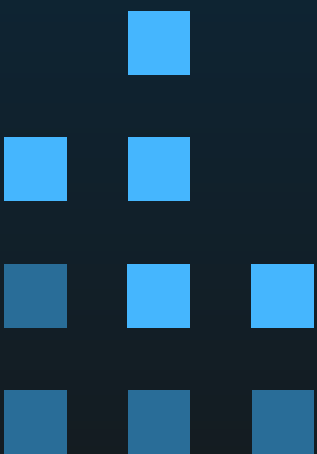
# 02

## Formalisation du sujet



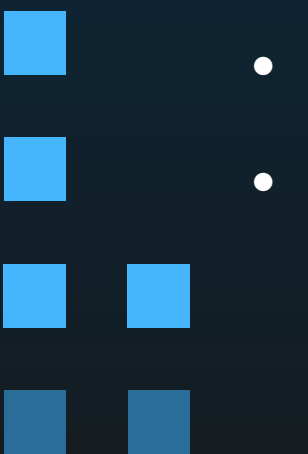


# Idées

- Plusieurs types d'agents : placer des pixels, lire un calque, sécuriser une zone
  - Agents avec un ou plusieurs centres d'intérêt
  - Stratégies pour créer/défendre un motif plus efficacement
- 



# Contraintes

- Communication avec des channels (pour les agents)
  - Passage de paramètres via le front
  - Communication en REST (avec le front)
- 

# Le cahier des charges

- **Côté front :**
  - Afficher la grille de pixels en temps réel
  - Exécuter des requêtes REST
- **Côté back :**
  - Faire communiquer les agents entre eux avec des channels
  - Pouvoir réaliser des motifs à partir d'un calque de couleurs en hexadécimal
  - Imposer un délai entre la pose de chaque pixel
  - Traiter des calculs en parallèle (=goroutines) au lieu d'avoir un traitement purement itératif

A decorative background on the right side of the slide, consisting of a grid of blue squares of varying shades (light blue, medium blue, and dark blue) arranged in a pattern that tapers off to the right.

# 03

## Modélisation des agents

# Les types d'agents



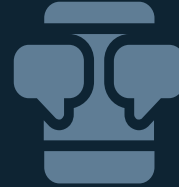
## Manager

L'agent manager lit la grille, utilise des calques et donne des ordres aux workers



## Worker

L'agent worker reçoit des ordres des managers et s'occupe de placer des pixels



## Chat

L'agent chat fait office de serveur afin de permettre aux agents d'avoir connaissance les uns des autres

# Structure des agents



Manager



Worker



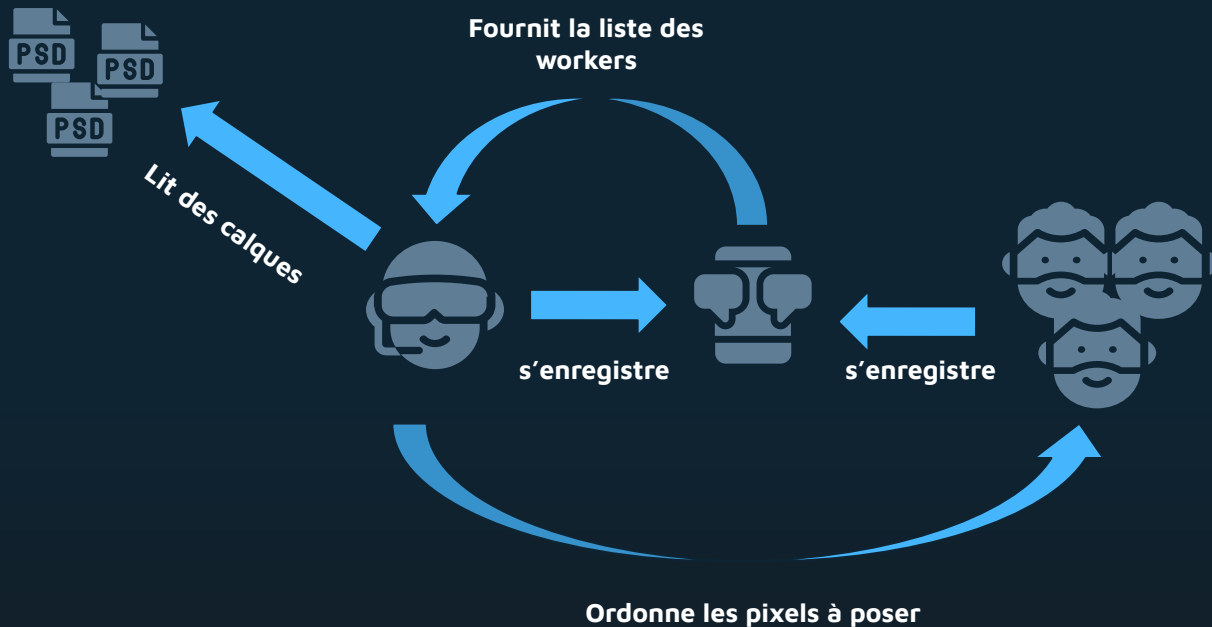
Chat

```
type AgentManager struct {  
    id          string  
    workers    []*AgentWorker  
    hobby      string  
    conquestValue float64  
    Chat       *Chat  
    Painting    painting.ManagerPainting  
    ImgLayout   [][]painting.HexColor  
    pixelsToPlace []painting.HexPixel  
    Cout        chan interface{}  
    Cin         chan FindWorkersResponse  
}
```

```
type AgentWorker struct {  
    id          string  
    tab         []painting.HexPixel  
    Hobbies     []string  
    Cin         chan interface{}  
    Chat       *Chat  
    mu         sync.Mutex  
}
```

```
type Chat struct {  
    Ams    []*AgentManager  
    Aws    []*AgentWorker  
    Cin    chan interface{}  
    srvUrl string  
    placeId string  
    cooldown int  
    height   int  
    width    int  
}
```

# Fonctionnement général





# Fonctionnement détaillé



## Worker

1. S'enregistre sur le chat

**Goroutine** : attend qu'on lui donne des pixels à placer

**Goroutine** : place les pixels qu'il a à placer en respectant le *cooldown*

-> synchronisation de la liste de pixels à placer entre les 2 goroutines



## Manager

1. S'enregistre sur le chat
2. Prend connaissance de ses workers
3. Charge son calque

**Goroutine** : regarde les éléments de son calques qui ne sont pas placé, envoie des instructions au workers en conséquence

Peut décider de dessiner son calque à plusieurs endroit différents si les pixels déjà pris ne sont pas volés

# Méta-paramètres

hobby	Détermine l'image à placer
nWorkers	Fixe, aléatoire, ou proportionnel à la taille de l'image
cooldown	Intervalle de temps entre chaque pixel posé par un même agent
size	Taille du <i>canvas</i> (carré)
conquestValue	<p>Valeur influent l'audace des managers à vouloir conquérir de nouveau territoire</p> <p>0 -&gt; sécurise ses pixels 100 -&gt; ne vas pas hésiter à peindre son image à plusieurs endroits différents</p>

A decorative graphic on the left side of the slide, consisting of a grid of blue squares of varying shades (dark blue, medium blue, and light blue) arranged in a pattern that tapers to the right.

# 04

## API REST

# Commandes de l'API

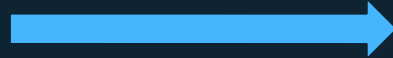
/new_place	Envoi une hauteur (en pixels), une largeur (en pixels) et un délai (en secondes) <b>Retourne l'ID du canvas</b>
/paint_pixel	Envoie des coordonnées x et y, une couleur, l'ID d'un canvas et l'ID du poseur <b>Ne renvoie rien</b>
/get_pixel	Envoie des coordonnées x et y et l'ID d'un canvas <b>Retourne la couleur du pixel (x,y)</b>
/get_canvas	Envoie l'ID d'un canvas <b>Retourne la grille en entière</b>
/get_diff	Envoie l'ID d'un canvas <b>Retourne les pixels modifiés depuis la dernière requête /get_diff</b>

# Utilisation de l'API par les agents



Manager

`/get_canvas`



Connaître l'état actuel de la grille afin d'envoyer les bonnes instructions au worker



Worker

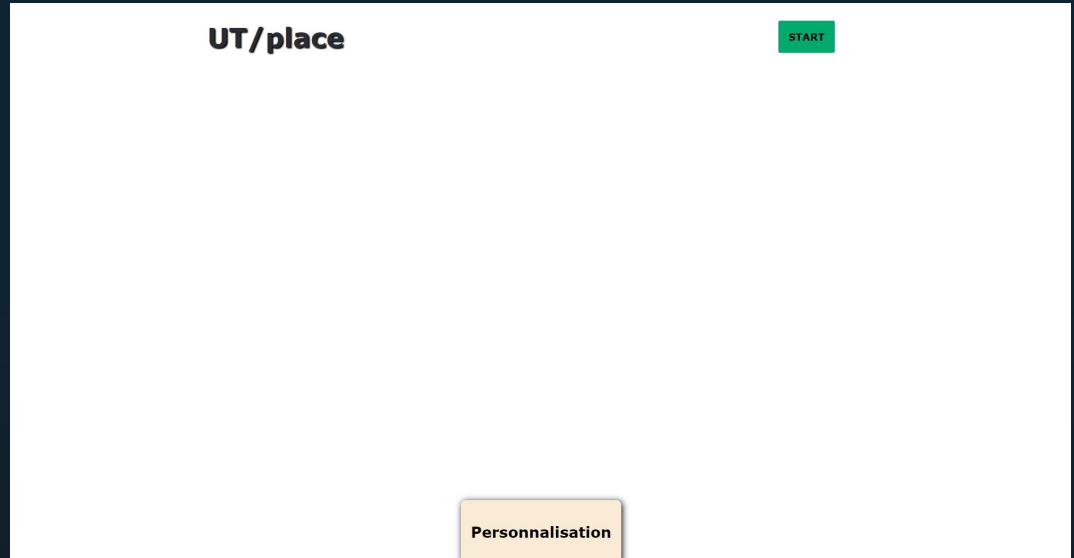
`/paint_pixel`



Peindre les pixels données en instructions par le manager

# Le front-end

- **Technologie :**
  - JavaScript (framework React)
- **Structure**
  - Affichage
  - Interaction





# Le front-end

- **Interaction**

- Start

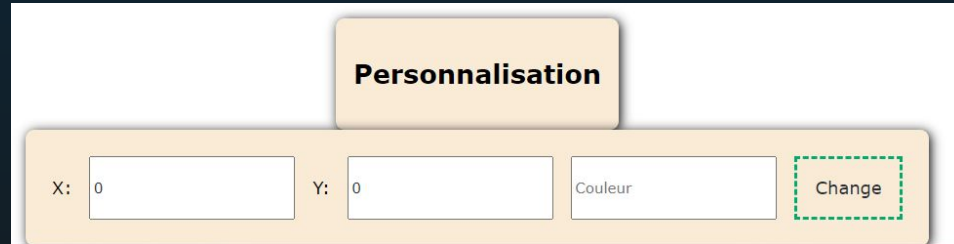
- Lance les requetes:
      - /get\_canvas 1 fois
      - /get\_diff toutes les 0.1s

- Personnalisation

- Change couleur d'un pixel



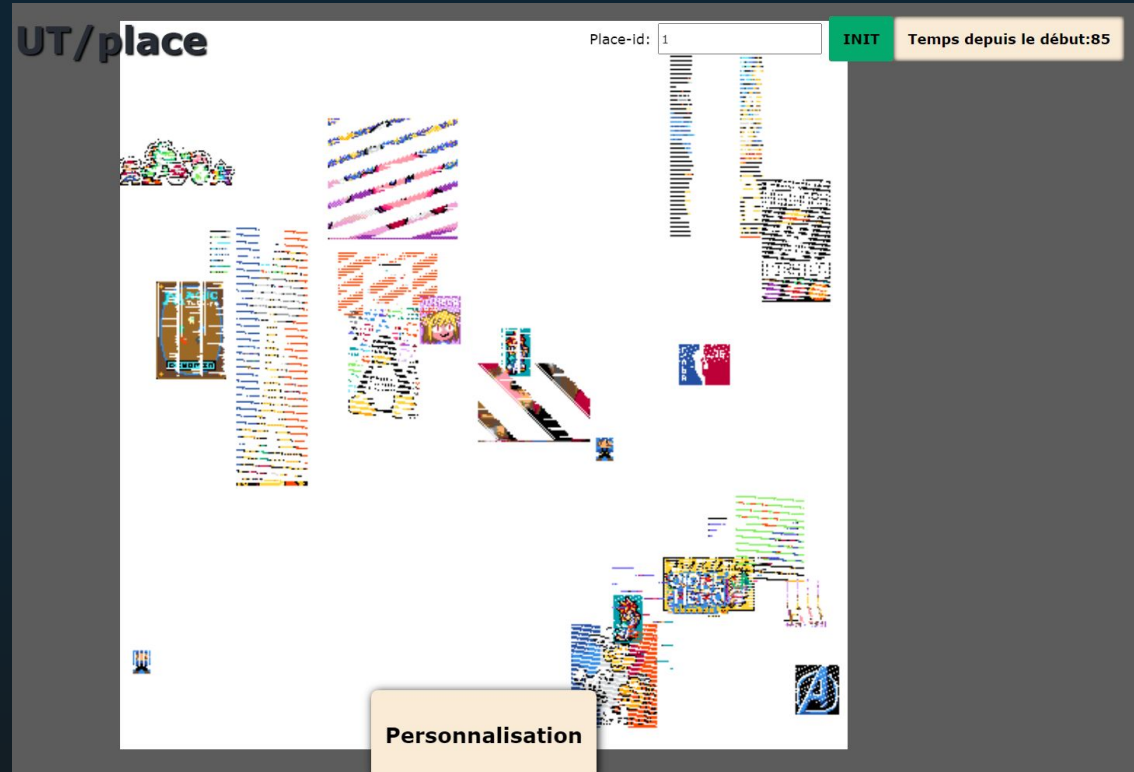
A UI element for starting the application. It consists of a text input field labeled "Place-id:" containing the value "0", and a green button labeled "START".



A UI element for customizing a pixel. It features a title bar labeled "Personnalisation". Below the title bar, there are three input fields: "X:" with the value "0", "Y:" with the value "0", and "Couleur". To the right of these fields is a button labeled "Change" which is highlighted with a dashed green border.

# Le front-end

- **Affichage**
  - Canvas
    - Affiche les pixels reçus (réponse de /getcanvas & /get\_diff)



# Démonstration