EVEEX - Guide d'utilisation du prototype python

> Guide d'utilisation du prototype d'encodeur réalisé en python
> Version du logiciel associé : 0.1.0
> Dernière édition : le *mercredi 7 octobre 2020*

Introduction

Le prototype d'encodeur que nous vos proposons sous le nom de EVEEX (Encodeur Vidéo Expérimental ENSTA Bretagne) peut être un peu effrayant au premier abord. Nous vous proposons ce guide qui vous permettra d'utiliser les différents outils composant le programme de compression vidéo.

À quoi sert ce programme ?

Les images, et plus particulièrement les vidéos sont au coeur de notre quotidien moderne. Les vidéos, par exemple, à elles seules représentent près de *80% du trafic internet mondial* !
Il devient alors intéressant de comprendre comment ces images ou ces vidéos sont compressées afin d'être stockées sur les serveurs, ou bien envoyées sur d'autre appareils.

EVEEX est un projet expérimental développé par 5 étudiants en 2ème année en SNS et SOIA de l'ENSTA Bretagne :

- Alexandre FROEHLICH
- Guillaume LEINEN
- Hugo QUESTROY
- Jean-Noël CLINK
- Hussein SAAD

Quels sont les outils qui composent le programme

À l'heure où nous rédigeons ces lignes, le prototype EVEEX comporte assez peu de fonctionnalités.

Tout d'abord le coeur du projet : la **compression d'image**. En se basant sur un principe similaire à la compression JPG, nous compressons une image afin de la transmettre sous forme de bitstream à fin de stockage ou de transmission.

Un autre outil consiste en la **génération d'images** notamment pour les essais de compression. Nous pourrons notamment retrouver un générateur d'image blanches ou "vides", un générateur d'images mosaïques ou encore la génération d'images à partir d'un fichier de configuration au format JSON.

Enfin, il comporte un outil permettant de **simuler la connexion à distance entre deux appareils**, en lançant un serveur d'écoute, puis un client lui envoyant des bitstreams, correspondant à une image encodée.

Principales commandes

Cette partie va traiter de la documentation des différentes commandes avec les différentes options disponibles.

Génération d'images

Génération d'une image blanche

```
python eveex.py -blank_img sizeX sizeY outputfile
```

Permet de générer une image blanche ou "vide" de taille `sizeX * sizeY` et de l'enregistrer dans le fichier `outputfile`.

Exemple :

```
# Pour créer une image blanche de taille 100x100 pixels
python eveex.py -blank_img 100 100 blank.png
```

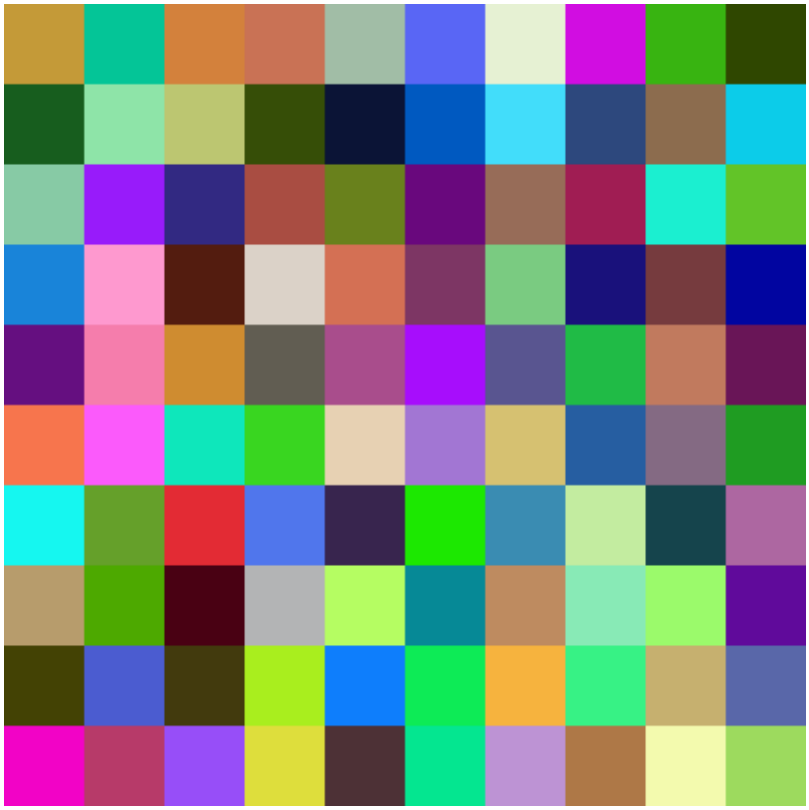
Génération d'une image mosaïque

```
python eveex.py -mosaic_img sizeX sizeY outputfile
```

Permet de générer une image en mosaïque de taille `sizeX * sizeY` et de l'enregistrer dans le fichier `outputfile`.

Exemple :

```
# Pour créer une image mosaïque de taille 100x100 pixels
python eveex.py -mosaic_img 100 100 mosaic.png
```



Génération d'une image à partir d'un fichier de configuration

```
python eveex.py -json_img conf_file.json image.png
```

Permet de générer une image à partir d'un fichier de configuration en JSON `conf_file.json` qui décrit l'image (voir exemple ci-dessous) et qui enregistre le résultat dans `image.png`.

Exemple : Le fichier ci-dessous crée une image de 100x100 pixels de couleur par défaut noire avec deux cercles et de rectangles. Le code devrait être assez explicite.

```
{
  "header" : {
    "size" : [100, 100],
    "background_color": "black"
  },
  "content" : [
    {
      "type" : "circle",
      "position" : [50, 50],
      "size" : 20,
      "color" : "red"
    },
    {
      "type" : "rectangle",
      "position" : [10, 20],
      "size" : [20, 20],
      "color" : "blue"
    },
    {
      "type" : "circle",
      "position" : [80, 10],
      "size" : 4,
      "color" : "green"
    },
```

```
    {
      "type" : "rectangle",
      "position" : [75, 75],
      "size" : [20, 10],
      "color" : "white"
    }
  ]
}
```

Encodage d'une image

```
python eveex.py -encode input.png output.txt
```

Permet d'encoder un fichier image `input.png` et d'enregistrer les bitstreams dans un fichier `output.txt`.

Serveur d'écoute de bitstreams

```
python eveex.py -server port
```

Permet de créer un serveur d'écoute sur le port `port` qui sera en attente de connexions TCP pour récupérer des bitstreams correspondant aux images encodées.

Exemple:

```
# Lancer un serveur d'écoute sur le port 5000
python eveex.py -server 5000
```

Client et envoi de bitstreams

```
python eveex.py -client ip_serveur port_serveur bitstreams.txt
```

Permet de créer un client qui se connecte au serveur localisé par l'adresse `ip_serveur` sur le port `port_serveur` et de lui envoyer les bitstreams contenus dans le fichier `bitstreams.txt`.

Exemple:

```
# Permet de se connecter au serveur sur la même machien au port 5000
# et de lui envoyer les bitstreams contenus dans bitstreams.txt
python eveex.py -client 127.0.0.1 5000 bitstreams.txt
```