AV-César – Aide à l'amélioration

I/ Ajout d'une nouvelle caractéristique des figures : (testé, méthode précise et fonctionnelle)

- Variable *config_ui.OUTPUTS* : ajouter le nom de la caractéristique voulue

- Méthode *main_ui.Drawing.draw* : ajouter le traitement pour l'affichage
  - `i` est l'indice de la frame que l'on veut afficher
  - `out` est le nom de la caractéristique que l'on veut mettre à jour
  - `state` est le dictionnaire nom_de_la_caracteristique:valeur_frame_i

  Étapes :
  - Rajouter dans le 'else' un « `elif out in [str nom_carac]:` » (souvent on ajoutera des caractéristiques 2 par 2 si l'on souhaite qu'elles soient disponibles tant pour l'ellipse que pour le rectangle)
  - Agrémenter la condition du 'elif' avec un « `type(state[out]).___name___ == 'nom_type_var'` » si l'on utilise un type spécial de valeur (par exemple un tuple ou une liste pour les couleurs).
    - `state[out]` sera la valeur souhaitée à cet instant pour ce paramètre
  - Le second paragraphe du 'else' est responsable de la remise à zéro des figures quand elles sont censées être cachées.
    - ajouter un « `for x in [str nom_carac]:` » suivi du « `if x not in state.keys():` » et remplir avec des fonctions semblables à celles ajoutées dans le paragraphe précédent mais annulant la caractéristique choisie.

- Il existe des variables qui peuvent être utiles :
  - `main.SCALE` donne un facteur multiplicatif pour que les tailles d'objets soient cohérentes et correctement affichées dans la scène
  - `main.WIDTH_SCALE` fait le même travail mais est adaptée aux bordures des objets

- Erreurs récurrentes :
  - si rien ne s'affiche, vérifiez que le deuxième paragraphe du 'else' est correct (il fait un nettoyage très restrictif).
  - si vous changez une caractéristique préexistante, vérifiez que les fichiers de configuration ne contiennent pas d'erreur, sans quoi cela peut poser souci après lancement de l'application (normalement ça n'est pas le cas mais l'on n'est jamais trop prudent).

II/ Ajout de nouveaux paramètres son : (indications)

- config.SOUND_INPUTS : ajouter le nom du nouveau paramètre son

- main.extractFeature : ajouter le traitement nécessaire.
  - ui est l'interface principale (utilisée pour stocker ui.frame_duration_ms)
  - sound est un objet main.analyzedSound
  - le traitement effectué doit retourner un tableau numpy unidimensionnel (peut peut-être fonctionner avec un tableau d'autres dimensions, mais FonctionAnalyse.ListeNormalisée peut poser problème) et il doit être ajouté dans le dictionnaire de listes créé en return, indexé par le même string qu'ajouté dans config.SOUND_INPUTS.

III/ Ajout de plus de figures : (indications)

- main_ui.UI_mainWindow.drawScene : changer les valeurs dans les 'range' des deux boucles 'for' et ajuster les calculs des coordonnées x et y de chacune des figures dans la ligne « figure = Drawing([…], self.scene) ». Le facteur multiplicatif factor a été ajusté pour une répartition correcte des figures, mais il est empirique.

- main_ui.UI_mainWindow.setupUI : ajouter les lignes composées d'un label et une comboBox inclus dans un QhorizontalLayout dans l'interface. Remplacer « x » par le numéro de la ligne de configuration + 4 (pour ajouter une cinquième configuration, x = 9).

```
self.horizontalLayout_x = QtWidgets.QHBoxLayout()
self.horizontalLayout_x.setContentsMargins(-1, -1, -1, 0)
self.horizontalLayout_x.setObjectName("horizontalLayout_x")
self.label_x = QtWidgets.QLabel(mainWindow)
sizePolicy = QtWidgets.QsizePolicy(QtWidgets.QsizePolicy.Preferred,
                                   QtWidgets.QSizePolicy.Maximum)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)
sizePolicy.setHeightForWidth(self.label_x.sizePolicy().hasHeightForWidth())
self.label_x.setSizePolicy(sizePolicy)
self.label_x.setObjectName("label_x")
self.horizontalLayout_x.addWidget(self.label_x)
self.comboBox_x = QtWidgets.QComboBox(mainWindow)
self.comboBox_x.setObjectName("comboBox_x")
self.horizontalLayout_x.addWidget(self.comboBox_x)
self.verticalLayout_2.addLayout(self.horizontalLayout_x)
```

- main_ui.UI_mainWindow.retranslateUI : ajouter la ligne « self.label_x.setText(_translate("mainWindow", "Configuration y")) » avec x ayant toujours la même valeur et y étant le numéro de la ligne de configuration comme ci-dessus.

- main : dans le « if ___name___ == '___main___' », rajouter comboBox_x dans le tableau ui.configCombos

IV/ Ajout d'une image à la place des dessins : (indications et suppositions de notre part)

Dans `main_ui.Drawing.___init___`, changer la commande `addRect` et/ou `addEllipse` afin de dessiner des images que l'on importe devrait être la seule modification nécessaire pour peu que ces objets image supportent les méthodes `resize`, `setPen` (changement du contour) et `setBrush` (changement de la couleur de fond).

V/ Création de nouvelles fonctions : (indications, exemple : `config_interpreter.grad`)

Il suffit de créer une fonction dans le module config_interpreter
Cette fonction pourra recevoir en argument, au choix :

- soit des variables (objets de type `config.Value`, `config.Color` ou `config.Gradation`) ;
- soit des tableaux numpy contenant les valeurs d'un paramètre sonore (parmi la liste `config.SOUND_INPUTS`) pour toutes les frames calculées du fichier son.