Qui est-ce ?
% Les paramètres du titre : titre, auteur, date Projet de programmation

Groupe Z
Frédéric,Laurent,Tony et Romain
git:git@gitlab.etu.umontpellier.fr:e20180001091/qui-est-ce.git
L2 informatique
Faculté des Sciences
Université de Montpellier.

April 16, 2022

Abstract

Description très succinte du problème et des différentes étapes de réalisation

1 Étape 2 : aider à la saisie des personnages

1.1 Description du problème: format des données et du résultat

Premièrement, au début de la programmation, nous avions eu deux choix de concept pour l'attribution des caractéristiques aux personnages depuis l'interface:

- Saisir manuellement (au clavier) chaque attributs, valeurs de chaque personnages
- Créer les attributs et les valeurs associées, les stockers, puis les attribuer aux personnages en les sélectionnant

Comme les deux concepts possédaient des avantages et des inconvénients, nous avions décidé d'établir la liste de celles-ci afin de déterminer la meilleure des deux.

- Pour la saisie manuelle :

L'avantage :
- Développement d'apparence plus simple (peu de fonction à implémenter)

Les inconvénients :
- Malgré la possibilité de vérifier les occurrences par comparaison orthographique, un "non" != "Non" != "Nope" donc possibilité d'occurence
- Dans le cas d'une envie de modifier ou de supprimer un attribut ou une valeur, il faudrait apporter la modification manuellement pour chaque personnage
- Complexifie les méthodes de vérification de la validité des données

- Saisie inconfortable des caractéristiques

  • Pour la saisie des caractéristiques et leur attribution aux personnage par simple selection :

Les avantages :
- Vérification de la validité des données plus simple (unicité des caractéristiques des personnages)
- Modification et de suppression des caractéristiques plus confortable et rapide

(!) Possibilité d'attribuer automatiquement les attributs déjà créés aux personnages

(!) Lors de la modification ou de la suppression d'un attribut ou d'une valeur, chaque personnage ayant cet attribut ou cette valeur recevra une modification automatique au niveau de sa liste de caractéristique

Les inconvénients :
- Malgré la possibilité de vérifier les occurrences par comparaison orthographique, un "non" != "Non" != "Nope" donc possibilité d'occurence
- Complexification du développement

(!) Création d'une class de type fenêtre pour la création et la modification des caractéristiques

(!) Création d'une class de type fenêtre pour l'attribution des valeurs aux attributs des personnages

Après reflexion, nous avions alors opté pour le deuxième concept en raison de fait que malgré la complexité du codage, le générateur serait beaucoup plus agréable à utiliser et de meilleur qualité.

Deuxièmement, après s'être mis d'accord sur la manière de saisie des données, il n'y avait plus qu'à réfléchir sur la manière de récupérer les données de chaque personnage.
Comme les paramètres de configuration de l'interface et le nom du plateau du qui-es-ce sont contenus directement dans les attributs du générateur, il à été facile de les récolter. Cependant, les personnages sont des objets de class et que leurs données ont été placées dans leurs attributs, nous avions alors implementé une fonction parcourant les personnages un à un avec leurs attributs. Au fur et à mesure, la fonction récolter alors leurs données sous la forme d'une String, et lorsque tout les personnages ont été parcouru, les données des personnages et de la configuration de l'interface du qui-es-ce sont écrites dans un fichier Json généré automatiquement et dont le nom est donnée en fonction du nom du dossier contenant les images des différents personnages.

1.2 Scénario des interactions avec l'utilisateur

1. L'utilisateur execution le programme :
   - (sans sauvegarde) il saisit le code d'execution suivit du chemin vers les images
   - (avec sauvegarde) il saisit le code d'execution suivit de l'option -save
2. Le générateur initialise l'interface
3. L'utilisateur configure l'interface avec les entry :
   - Il saisie des dimensions du plateau
   - Il saisie de la taille des images représentant les personnages
   - Il saisie l'espacement entre les images

(!) Le générateur affiche d'un messages d'erreur si les dimensions sont trop grandes, si la taille d'image est trop grande ou si l'espacement entre les images est trop grand

4. L'utilisateur crée les caractéristiques avec le bouton "configuration Attributs"
5. Le générateur affiche une nouvelle fenêtre permettant à l'utilisateur de créer les caractéristiques
   - l'utilisateur créé / modifi ou suppression des attributs et les valeurs correspondant

(!) affichage d'une erreur s'il y a la création d'un attribut déjà existant

(!) affichage d'une erreur si la modification du nom d'un attribut correspond à un autre
(!) affiche une demande de confirmation si vous effectué une modification et affichage d'un avertissement si vous voulez supprimer un attribut ou une valeur

6. L'utilisateur affiche l'aperçu du plateau en cliquant avec le bouton "Aperçu "

7. L'utilisateur caractérise les personnages du plateau :

    - Clique sur l'image d'un des personnage sur l'aperçu du plateau
    - Le générateur affiche l'attribut que le personnage possède
    - Selection des valeurs correspondant aux attributs du personnage et valide la selection

(!) affichage d'une erreur si le set d'attribut du personnage n'est pas unique

    - Répétition du même processus pour tout les personnages du plateau

8. Le générateur verifi la validité du plateau :

    - Il vérifie si toutes les caractéristiques de chaque personnage ont été attribuées d'une ou de plusieurs valeurs

(!) affichage d'un message d'erreur si des personnages ont été mal caractérisés (application d'un filtre rouge sur l'image des personnage mais caractérisés / application d'un filtre vert dans le cas contraire)

    - Si tout a été correctement fait, le générateur affiche un message indiquant la génération du fichier .json