

Apprendre l'algorithmique

Rémi HEITZ
Youssef IRHBOULA
Antoine HUGUET

Vendredi 22/11/19

Sommaire

- 1 Présentation
- 2 Structuration du projet
- 3 Répartition des tâches

- 1 Présentation
 - Quel est ce projet ?
 - Quel est son intérêt ?
- 2 Structuration du projet
- 3 Répartition des tâches

Quel est ce projet ?

Le projet

À quoi sert ce projet ?

Ce projet a pour but d'apprendre à l'utilisateur les **bases de l'algorithmique** à l'aide d'une interface utilisateur graphique, à l'aspect ludique.

À qui s'adresse ce projet ?

Ce projet s'adresse aux personnes ayant de **aucune à très peu d'expérience** en algorithmique et en programmation.

Quel est ce projet ?

Nos objectifs

Facilité d'utilisation

Ce programme doit être le plus simple et intuitif possible à utiliser : l'utilisateur doit pouvoir accéder à toutes les fonctionnalités du programme dès sa première utilisation.

User friendly

L'interface doit être agréable d'utilisation, afin d'éviter de dissuader des débutants, peu habitués à l'allure austère des IDE.

Quel est son intérêt ?

Utilisation actuelle

L'objectif

- Apprendre les bases de l'algorithmique à des débutants
- Découvrir le concept de variables ou de boucles

Les fonctionnalités

L'utilisateur a à sa disposition de nombreux blocs graphiques parmi lesquels :

- Des boucles WHILE
- Des test IF et ELSE
- Des variables (pouvant être de type Int ou String)
- Des blocs pour effectuer des calculs
- Des blocs pour afficher des résultats

Quel est son intérêt ?

Utilisation future possible

Objectifs futur

- Apprendre à des utilisateurs plus expérimentés des algorithmes usuels
- Résoudre des défis en utilisant des structures de données plus complexes
- Afficher en *temps réel* l'état des structures de données parcourues

Exemples de structures de données

- Des graphes dont les sommets visités sont affichés au fur et à mesure du parcours
- Des listes dont l'état de tri est affiché en temps réel.

- 1 Présentation
- 2 Structuration du projet
 - Découpage du code
 - Modification de la structure
- 3 Répartition des tâches

Découpage du code

Front end / Back end

Notre projet, de par sa nature, est très fortement découpé entre son *Front end* et son *Back end*, avec des fonctions pour communiquer entre.

Exécution

Le lancement du code est assuré par un fichier `main.py`, qui récupère les fonctions définies dans les autres modules.

Planification et restructuration

Planification

Les dossiers `codeAnalysis/models` et `GUI/ressources` ont été prévus dès le début du projet.

Refactoring

La partie GUI a subi 2 grosses restructuration :

- Pour découper le module en différents sous programme
- Pour faciliter l'exécution du programme depuis `main.py`

Processus de développement

Processus

- Créations des objets
- Incorporation dans les fonctions principales

Créations des objets

- Réflexion commune sur les besoins en attributs et méthodes
- Création des classes correspondantes

Incorporation

- Discussions sur les besoins pour l'interconnexion *Front/Back*.
- Ajout des objets

- 1 Présentation
- 2 Structuration du projet
- 3 Répartition des tâches

Répartition des tâches

Front End

- Création de la fenêtre principale - **Rémi**
- Graphismes - **Antoine**
- Création de la logique d'affichage - **Rémi**
- Ajout de blocs et modules complémentaires - **Rémi & Antoine**
- Refactoring - **Antoine**

Back End

- Lecture et exécution du code - **Youssef**
- Vérification syntaxique et sémantique - **Youssef**
- Tests unitaires - **Rémi**

Annexes

Arborescence

```

└─ codeAnalysis
  └─ models
    ├── Bloc.py
    └─ config.py
  ├── block_to_code.py
  └─ tests.py
└─ GUI
  └─ models
    ├── Blocks.py
    └─ config.py
  ├── Windows.py
  └─ resources
    ├── backgrounds
    ├── blocks
    └─ fonts
  ├── gui.py
  ├── .gitignore
  ├── CODE_OF_CONDUCT.md
  ├── CONTRIBUTING.md
  ├── LICENSE
  ├── main.py
  ├── README.md
  └── REQUIREMENTS.md
```

Test

Back end

Un fichier `tests.py` utilise le module `unittest` afin d'effectuer des tests unitaires.

Front end

Les tests unitaires ne sont pas nécessaires car les fonctions graphiques sont *par définition* testable visuellement.