Projet Draw++: Resume des Travaux

Votre equipe

Introduction

Le projet Draw++ est un langage de programmation graphique concu pour interpreter et compiler des commandes graphiques simples. Ce projet inclut la creation d'un lexer, d'un parser, d'une grammaire formelle, ainsi que des tests pour valider les composants du langage. En outre, nous avons configure un environnement de developpement sur Ubuntu avec un fichier `requirements.txt` pour gerer les dependances.

Configuration de l'environnement

Pour configurer l'environnement virtuel et installer les dependances sur Ubuntu, suivez les etapes suivantes:

1. Assurez-vous que Python 3.8 ou une version superieure est installee.
2. Creez un environnement virtuel:

   ```
   python3 -m venv env
   ```

3. Activez l'environnement virtuel:

   ```
   source env/bin/activate
   ```

4. Installez les dependances depuis le fichier `requirements.txt`:

   ```
   pip install -r requirements.txt
   ```

5. Pour lancer les tests, utilisez:

   ```
   python3 -m unittest discover
   ```

Structure du Projet

1. Grammaire du Langage

La grammaire formelle de Draw++ a ete definie pour inclure des constructions comme les declarations de curseurs, les methodes de dessin, les boucles (for, while) et les structures conditionnelles (if-else). Voici un extrait de la grammaire:

```
programme ::= instruction*
instruction ::= declaration_curseur | positionnement_curseur
              | dessin_ligne | conditionnelle | boucle_for | ...
```

Cette grammaire a permis de structurer le langage et de guider le developpement du lexer et du parser.

2. Lexer

Le lexer analyse le code source et genere une liste de tokens. Chaque token represente une unite lexicale (mot-cle, identifiant, nombre, symbole, etc.). Le lexer a ete implemente en Python avec des expressions regulieres. Par exemple:

```
token_specs = [
    ("PLUS_PLUS", r'\+\+'),
    ("NUMBER", r'-?\d+(\.\d*)?'),
    ("IDENTIFIER", r'[a-zA-Z_]\w*'),
    ...
]
```

Le lexer inclut la gestion des erreurs pour detecter les caracteres invalides.

3. Parser

Le parser construit un arbre syntaxique abstrait (AST) a partir de la liste de tokens generes par le lexer. Les principales methodes incluent:

- `parse_cursor_declaration()`: Analyse les declarations de curseurs.
- `parse_if_statement()`: Analyse les structures conditionnelles.
- `parse_for_loop()`: Analyse les boucles for.

Voici un exemple de noeud d'AST pour une declaration de curseur:

```
{
    "type": "CURSOR_DECLARATION",
    "name": "myCursor"
}
```

4. Tests

Des tests ont ete implementes pour valider les composants:

- **Tests du Lexer:** Verifient que les tokens sont correctement identifies. Exemple:

```
def test_cursor_declaration(self):
    tokens = self.tokenize_code("cursor myCursor;")
    self.assertEqual(tokens[0].type, TokenType.CURSOR)
```

- **Tests du Parser:** Valident que l'AST est correctement genere. Exemple:

```
def test_cursor_declaration(self):
    syntax_tree = self.parse_code("cursor myCursor;")
    self.assertEqual(syntax_tree[0]["type"], "CURSOR_DECLARATION")
```

Conclusion

Ce projet a permis de concevoir un langage de programmation graphique de A a Z, integrant un lexer, un parser et des tests pour valider les composants. Les etapes realisees constituent une base solide pour le developpement ulterieur de Draw++.