Nombre de participant·e·s maximal pour l'atelier	20	Nombre de sessions de 45mn que vous pouvez réaliser	3
Matériel nécessaire (vidéo-projection, grande salle, autre ...)	Vidéo projection, tableau noir + craies (ou tableau blanc + feutres).		

Structuration ADG

Abstraction Découpage Généralisation

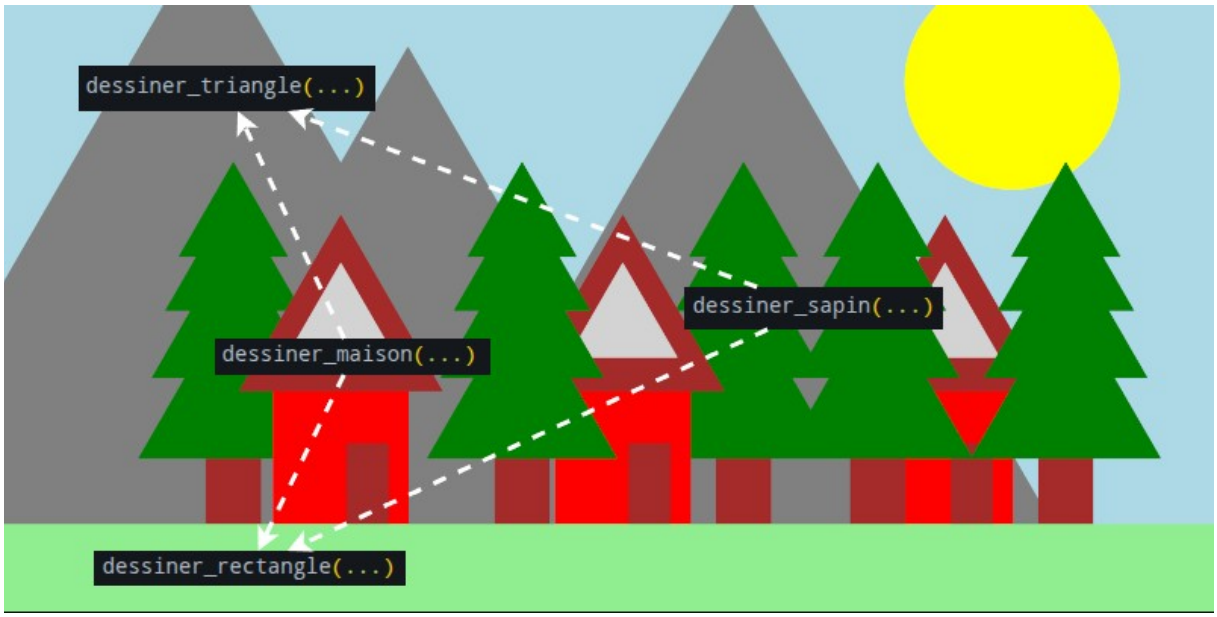
BEFFARA Emmanuel, BERTONI Félix

Université Grenoble Alpes, Éducation Nationale

RÉSUMÉ *Cet atelier propose une discussion et une exploration des principes de base de la structuration du code, en particulier à l'aide de fonctions, de leurs enjeux pédagogiques et didactiques pour les débutants, et de la manière de les leur enseigner. Il ouvre la voie à de futurs réflexions et développements sur le sujet.*

Les participants pourront aussi y trouver une occasion de réfléchir sur leur propre pratique de la programmation et conscientiser ces principes qu'ils appliquent peut-être de manière intuitive.

MOTS-CLÉS *• Programmation, fonctions, structure du code.*



Structuration ADG d'un dessin avec Turtle et Python

1. Déroulement de l'atelier

Cet atelier propose une réflexion autour d'une approche de la programmation et de son enseignement que nous nommons « Abstraction Découpage Généralisation », abrégé en ADG. Cette structuration se retrouve dans tout projet de plus de quelques lignes, elle consiste à découper le code en blocs identifiés, dont les interfaces font abstraction des détails et où chaque bloc peut éventuellement être généralisé au-delà du besoin initial. Cela prend différentes formes concrètes, indépendamment d'un paradigme de programmation particulier, et permet de gérer la complexité du code en travaillant chaque bloc de manière séparée.

La chronologie suivante est proposée pour l'atelier :

1. **ADG : quoi ? pourquoi ? malgré quoi ? (10 minutes).** Définition et exemples, intérêt en programmation (simplification, collaboration, maintenabilité), intérêt pour l'apprentissage (confiance, autonomie, créativité, démarche de projet), complexité de la notion, question de l'institutionnalisation.
2. **Exemple de séquence pédagogique (10 minutes).** Démonstration d'une activité utilisant Python et le module Turtle, pour la classe de 1ère. L'activité développe l'usage de fonctions pour créer des dessins de complexité croissante, en mettant en avant comment l'emploi de fonctions bien définies simplifie progressivement les tâches.
3. **Échange et discussion (25 minutes).** Partage d'expériences et de points de vue sur l'approche présentée : quelle pertinence dans la pratique et l'enseignement, quelles activités pour la travailler en classe, quelle institutionnalisation, quelle évaluation des compétences associées.

2. Description du projet présenté dans l'atelier

L'approche ADG décrit une structuration de l'activité de logicielle qui se retrouve dans tout projet de plus de quelques lignes. Elle n'est pas dépendante d'un paradigme de programmation particulier et peut de fait prendre différentes formes concrètes, en s'appuyant ou pas sur des outils spécifiques à chaque langage (fonctions, classes, modules, etc.) et sur des conventions de programmation plus ou moins explicites (nommage des fonctions et des variables, documentation et emploi commentaires, formatage du code, etc.). En permettant de travailler sur chaque bloc de manière séparée, elle permet de s'affranchir en partie de la complexité inhérente du code en tant qu'ensemble, tout en facilitant le travail collaboratif.

L'ADG mobilise des compétences primordiales pour réaliser des projets, mais c'est une notion complexe, avec trois composantes interdépendantes (abstraction, découpage, généralisation), et qui est souvent appliquée de manière intuitive et non totalement conscientisée par les professionnels, et enseignée de manière avec aussi une part variable d'implicite. Cela soulève donc un questionnement sur l'ADG en tant qu'objet d'enseignement pour les débutants en programmation :

- Quels acquis viser ?
- Avec quelles méthodes ?
- Comment évaluer ces acquis ?

Une activité exemple est proposée, ainsi qu'un retour d'expérience de sa mise en œuvre. Elle consiste à faire dessiner les élèves avec Python et Turtle, en construisant itérativement des outils de dessin sous forme de fonctions : d'abord des formes, puis des objets, puis des plans, et ainsi de suite. Chaque étape commence par écrire du code dans la séquence principale du programme (ex : une maison à partir de formes), puis à isoler le code dans une fonction et à le généraliser (dessiner une maison étant donné une position passée en paramètres).

Lors de la phase d'échange, si des participants sont volontaires, ce pourra être l'occasion de choisir une ou plusieurs pistes qu'ils pourront explorer dans le but de faire un retour d'expérience plus tard. Le compte-rendu des échanges au sein de l'atelier sera par la suite mis à disposition en accès libre dans un dépôt qui permettra également d'accueillir et de coordonner d'éventuels futurs développements.

3. Expérimentations réalisées ou envisagées

Une première version de l'atelier a déjà été menée avec lors de la Journée des inspecteurs de NSI de l'académie de Grenoble, ce qui a contribué à affiner différents questionnements autour de l'approche et de son applicabilité en classe :

Jusqu'à quel niveau de granularité l'approche ADG est-elle viable ?

L'usage explicite d'une telle approche peut-il être exigible pour des élèves de lycée, notamment dans la mesure où envisager les tâches informatiques dans leur globalité nécessite une certaine maturité (intellectuelle en générale, et dans la discipline informatique en particulier) ?

Quels dispositifs sont employés et dans quelle mesure permettent-ils d'approcher les compétences visées ? Par exemple : fournir des trames de code à compléter, forcer la réflexion loin du clavier et jusqu'à quel point, etc.

Quelle interaction avec la pratique des projets (individuels et groupes) ? Quelles parties des programmes révèlent le mieux le besoin de structuration (on pense en particulier à la conception d'interfaces, même simples) ?

Intégrer ou pas, et de quelle façon, cette approche de la modularité à des éléments d'enseignement explicite (montrer, accompagner, puis laisser faire) ?

4. Liens avec la recherche

L'approche ADG met en avant trois composantes essentielles de l'activité de programmation, en cohérence avec les compétences caractéristiques de la pensée informatique telles que formalisées par Declercq (2021) suivant Selby et Woollard (2013) : *évaluer*, *anticiper*, *décomposer*, *généraliser* et *abstraire*. Si les deux premières sont mobilisées dès les premiers exercices de programmation, où la spécification de l'attendu est prise en charge par l'enseignant (tâches du type « écrire une fonction qui… » ou lecture d'un code donné) et où le code produit est essentiellement à usage unique, les trois dernières sont en jeu dès qu'il est nécessaire de penser la programmation comme moyen de résolution de problème, avec les enjeux et compétences transversales qui s'y rapportent (décomposition de problème, modélisation, etc.).

Ces différentes compétences sont des objectifs d'apprentissage à différents niveaux scolaires. Cependant, la caractérisation précise des compétences et le niveau d'attendu adapté à chaque niveau sont encore à établir, comme en témoignent différents travaux portant sur l'élaboration de référentiels (Projet PIAF, 2020; Chane-Lune et al., 2024; Beffara et al., 2026).

L'atelier sera l'occasion d'enrichire ces réflexions, en particulier sur les aspects de la pratique de la programmation qui ne sont pas liés à des notions particulières mais à des savoir-faire plus transversaux spécifiques à la discipline informatique.

5. Liens vers les ressources et point de contact

- Activité Turtle : https://1nsi.felix-bertoni.fr/programmation_python/fonctions_python/cours/
- Présentation : https://feloxyde.gitlab.io/jdi_structuration_adg/
- Point de contact pour avoir plus de renseignement : felix.bertoni@ac-grenoble.fr

6. Références

Beffara, E., Desmoulins, C., Rasse, A., & Wack, B. (2026). *Création et validation d'un référentiel de compétences en informatique pour le cycle 4* [Poster]. Didapro 11.

Chane-Lune, S., Declercq, C., & Hoarau, S. (2024). Un référentiel de compétences en programmation pour identifier les difficultés des débutants et différencier les activités. In K. Mens & O. Goletti (Éds.), *Actes du colloque Didapro 10 sur la Didactique de l'informatique et des STIC. Volume 1* (p. 53-63). UC Louvain. https://hal.science/hal-04482126

Declercq, C. (2021). Didactique de l'informatique : Une formation nécessaire. *STICEF*, *28*(3). https://doi.org/10.23709/STICEF.28.3.8

Projet PIAF. (2020). *Référentiel des compétences – Pensée Informatique et Algorithmique dans l'enseignement Fondamental*. https://piaf.loria.fr/contributions/referentiel-des-competences/

Selby, C., & Woollard, J. (2013). *Computational thinking : The developing definition* [Monograph]. University of Southampton. https://eprints.soton.ac.uk/356481/