REPERES - IREM. N° 109 - octobre 2017

TORTUE ET GEOMETRIE DYNAMIQUE

Yves MARTIN
Université de La Réunion

Cet article est également consultable en ligne sur le portail des IREM (onglet : Repères IREM) : http://www.univ-irem.fr/

La nouvelle rubrique « algorithmique et programmation » des programmes scolaires (cycles 3 et 4) installe désormais la programmation visuelle au collège. Même si plusieurs environnements seraient utilisables (dont Snap de l'université de Berkley), compte tenu des contraintes de formation continue et de la richesse des ressources d'autoformation disponibles sur ce logiciel (formations canadiennes du RECIT, nombreuses chaines YouTube de collègues, en français), il était logique, et assurément réaliste, que l'institution oriente les enseignants vers une utilisation systématique de Scratch pour cette première année de mise en œuvre de la réforme.

Pourtant d'autres outils se mettent en place et leurs usages laissent entrevoir que certains pourraient non seulement être un complément significatif de ce que propose actuellement l'institution, voire inspirer des inflexions sur l'évolution curriculaire aussi bien de la géométrie dans l'espace que de la programmation en elle-même au collège.

C'est le cas, pensons nous, des dernières avancées du logiciel de géométrie dynamique DGPad, avec son intégration de Blockly (autre environnement complet de programmation visuelle), et en particulier de la mise en œuvre de sa tortue dynamique qui ouvre des perspectives si nouvelles pour pratiquer la modélisation 3D qu'elle mériterait une réflexion plus approfondie pour une éventuelle utilisation scolaire systématique.

1. — Rapide présentation de DGPad

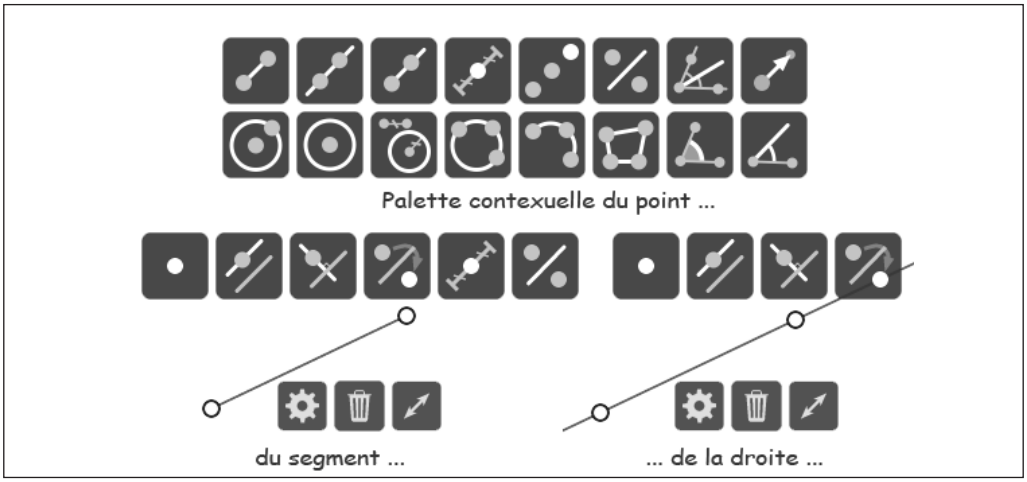
DGPad a d'abord été écrit pour tablettes (Androïd et iOS). Rédigé en JavaScript, l'application est utilisable aussi comme application en ligne, et donc sur ordinateur, mais a priori en ligne (il existe néanmoins des versions locales pour Linux et Mac OS X). Même

si les outils sont utilisables à la souris, ils sont d'abord conçus pour tablette, en particulier pour un pointeur non continu (quand le doigt n'est plus au contact de la tablette), avec l'objectif de conserver la même ergonomie d'engagement direct que sur ordinateur, voire de l'améliorer parfois. Par ailleurs il n'y a pas de roll-over sur tablette (réaction du logiciel au survol d'une partie de l'écran à la souris), donc si on veut réaliser une application de géométrie dynamique moderne, qui accompagne la démarche cognitive de l'élève en anticipant ses constructions, sur tablette, il faut réinventer une interface adaptée. L'auteur de DGPad est aussi auteur de CaRMetal et pourtant DGPad n'est pas – pas du tout – une transcription du logiciel précédent plus ou moins adapté à la tablette, c'est un logiciel et une interface entièrement repensés.

1.a. *Les palettes contextuelles*

Chacun a déjà pratiqué GeoGebra, ou Cabri pour les plus anciens, ou encore CaRMetal ou autres : tous ces logiciels ont une interface classique pour leurs outils. On dit que leurs outils sont **préfixés**, c'est-à-dire que l'on choisit l'outil puis les objets associés. On ne s'en aperçoit pas toujours car on a l'habitude et on va assez vite mais cette interface est particulièrement « gourmande en clic », surtout s'il y a des menus déroulants.

Eric Hakenholz – l'auteur de DGPad – a réfléchi à une interface plus efficace nécessitant moins de contact (de « touché ») sur tablette, ou clic de souris sur ordinateur. Il a choisi une approche radicalement nouvelle, celle des *palettes contextuelles à la situation*. Informatiquement, c'est le choix d'une programmation objet *centrée sur les objets géométriques* et non plus sur les outils. Une autre raison de cette réflexion est l'impossibilité de l'anticipation de constructions aussi simples que les segments ou les droites – comme on le constate dans les versions tablettes d'autres logiciels – si on conserve la logique des outils préfixés. Avec cette interface, c'est comme si on venait réveiller un objet (point, droite, segment, polygone, cercle) et qu'il nous proposait tout ce qu'on peut faire avec lui. Les palettes sont alors différentes pour un point, une droite, un segment, un cercle …



Conceptuellement, on peut dire que l'on passe d'une interface où les outils étaient préfixés (tous les autres logiciels de géométrie dynamique), à une interface où ils sont **infixés**. C'est une nouvelle habitude à prendre surtout pour les outils à trois entrées ou plus (cercle circonscrit, angle, polygone). En effet, le système ne peut pas anticiper où le pointeur tactile – le doigt – va réapparaître pour le troisième point du cercle circonscrit. L'interface doit en tenir compte pour forcer le lieu où l'on attend le pointeur pour conserver des constructions en anticipation.

1.b. *Le mode standard et le mode consultation*

Cette interface qui donne aux objets l'option de nous présenter tout ce qu'il est possible de faire (16 actions depuis un point) a vite été évaluée en classe comme trop ouverte. En pratique les élèves de seconde qui ont été les premiers [1] à tester le logiciel régulièrement créaient trop de points alors qu'ils voulaient juste manipuler des points ou des curseurs. L'option de rajouter un mode consultation a été rapidement mise en place (dès fin 2013). Le logiciel dispose de deux modes : un **mode standard**, quand un des outils du tableau de bord est sélectionné, et le **mode consultation** quand aucun outil n'est sélectionné (on dit aussi le mode *« sans outils »*). Pour manipuler avec Blockly (et la tortue de DGPad) il faut être en mode standard.

Dans le mode consultation, en particulier le repère 3D est plus rapide et il ne se déplace qu'à un doigt (tablette-trackpad) ou clic-gauche-glisser (souris) au lieu de deux doigts (respectivement clic-droit-glisser). Quand on charge une figure, par défaut elle est en mode consultation. Un « touché» (un clic sur ordinateur) sur un item du tableau de bord fait passer l'application en **mode standard**, de création d'objets.

En classe – surtout sur tablette – on peut apprendre aux élèves à passer d'un mode à l'autre rapidement pour manipuler la figure « en consultation ».

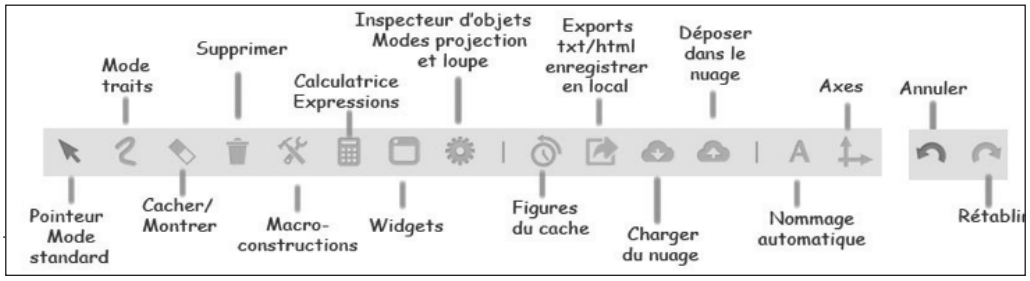
Exemple :

Pliage des 11 patrons du cube en mode consultation :
http://goo.gl/dtawaZ

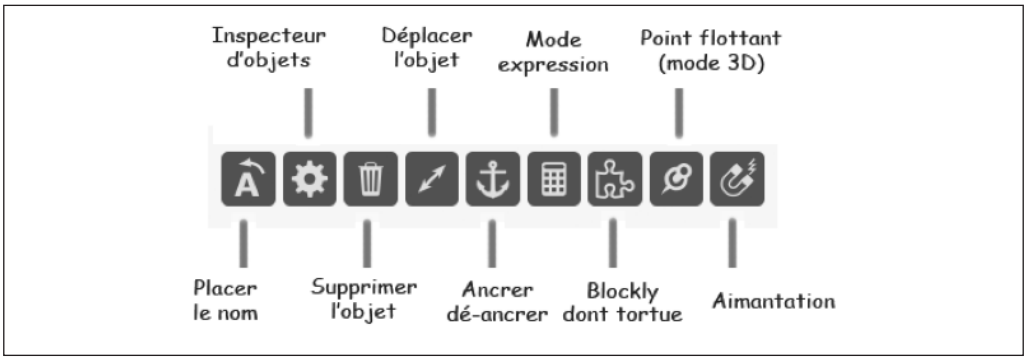
Le tableau de bord de DGPad, est donné ci-dessous sans plus de détail...

1.c. *La palette des comportements (elle aussi contextuelle)*

Elle apparaît sous chaque objet pour enrichir soit le comportement, soit l'aspect des objets (voir l'encadré correspondant en haut de la page suivante).



1 Au lycée professionnel de Saint Joseph dans les classes de David Ethève. Voir ses comptes-rendus sur le site de l'Irem de La Réunion.



Cette palette de comportement contient :

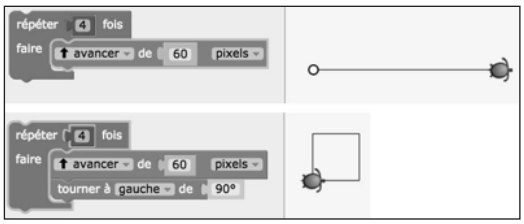
- des comportements spécifiques (d'où son nom) : ancrage et aimantation en 2D, point flottant en 3D ;
- des raccourcis du tableau de bord (inspecteur, suppression, les expressions) ;
- des réglages qui ne se font que sur la fenêtre elle-même (la position du nom des points et le déplacement d'objets superposés) ;
- enfin, et pour nous ici, surtout, l'accès à l'interface Blockly, et en particulier la tortue dynamique : **l'interface Blockly est un comportement** d'objet.

2. — Les spécificités d'une tortue dynamique

Quand on ouvre Blockly *sur un point*, l'interface propose 5 onglets de comportements possibles :

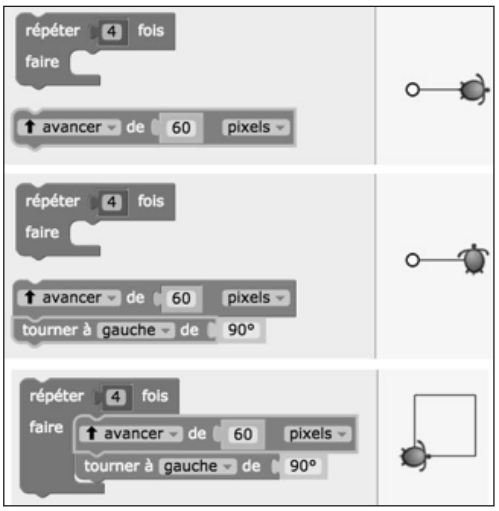
Par défaut, on est sur un comportement activé par un déplacement du point. La tortue est donc un des cinq comportements. Quand on l'active, deux nouveaux items Blockly apparaissent, un item *Tortue* bien entendu mais aussi un item *Texte* car la tortue peut écrire et surtout écrire en LaTeX, ce qui est une nouvelle façon de mettre du TeX dynamique un peu partout dans une figure.

La tortue de DGPad est **dynamique**. Cela signifie plusieurs choses et en particulier que le logiciel construit la trace de la tortue *en temps réel*. Par exemple pour faire un carré, on peut, séquentiellement, avoir différentes approches. En voici deux sur un simple carré :



Dans une approche (ci-dessus), on place un « avancer » *dans la boucle de répétition*, aussitôt la tortue avance de *4 fois* la longueur souhaitée, puis dès que l'on ajoute un *tourner gauche*, le carré se trace aussitôt.

Dans l'autre approche (en haut de la première colonne de la page suivante), on place sur l'environnement de travail un *avancer*, la tortue avan-



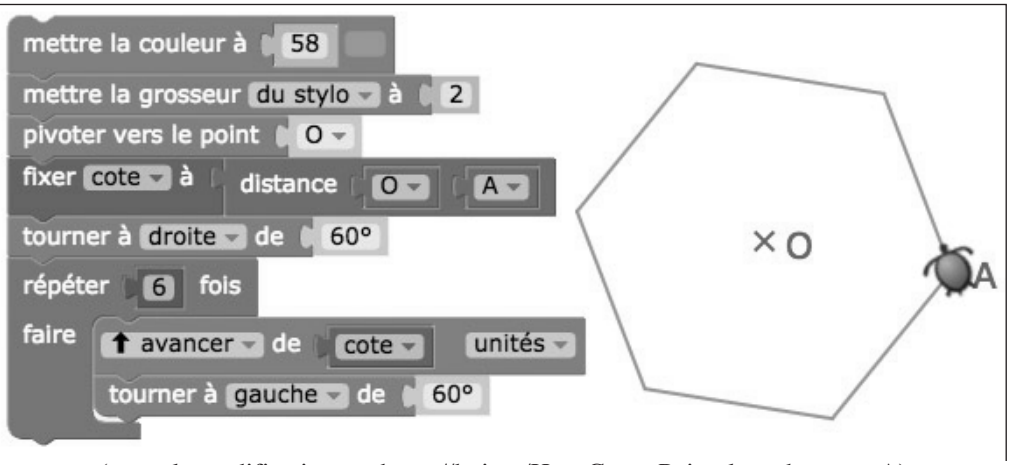
ce aussitôt de la longueur voulue, puis on place le *tourner gauche*, et on voit aussitôt la tortue tourner, puis on place l'ensemble des deux blocs dans la boucle, et le carré se trace aussitôt.

C'est donc une logique très différente de la démarche classique d'autres logiciels qui permet de ralentir la tortue et la voir faire son parcours. Cet aspect dynamique de la tortue déplace les problématiques et ouvre des perspectives.

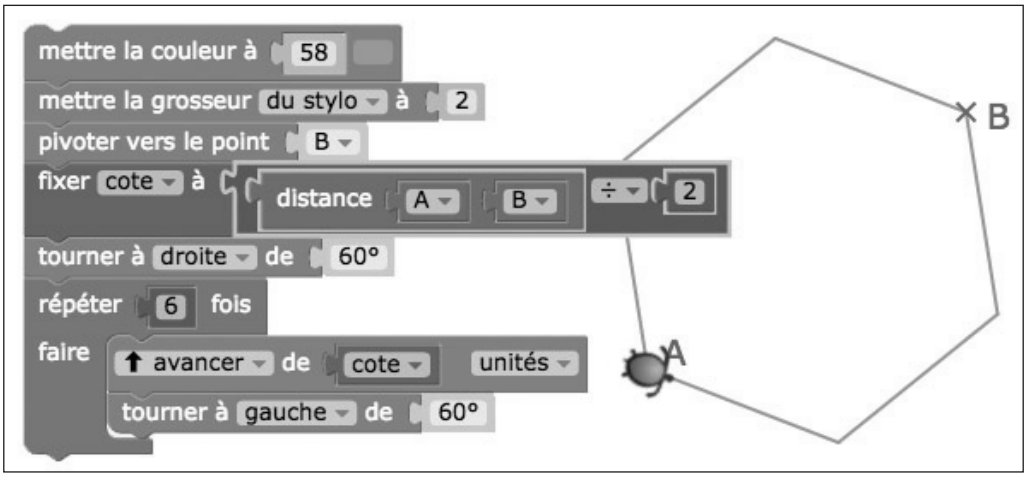
Elle déplace les problématiques car on peut raisonner en terme d'anticipation et construire des activités dans cette optique.

Elle ouvre de nouvelles perspectives car on peut aussi travailler avec cette instantanéité, sur des modifications en temps réel des instructions, et tester, explorer en direct, spontanément, des angles, des longueurs : là aussi, c'est une autre façon de concevoir des activités. On peut aussi jouer sur les modifications d'ordre d'instructions, voir ce qui est invariant, voir ce qui change.

Par exemple, il est facile de faire observer que l'algorithme qui construit un hexagone régulier par centre et point est exactement le même que celui qui le construit à partir d'un diamètre, il y a juste une ligne à changer, la longueur du côté :



(tester la modification sur https://huit.re/HexaCentrePoint, le code est en A)



Sur figure précédente, déjà, beaucoup d'élèves ne mettent pas le « tourner droite » avant la boucle, et construisent un hexagone régulier de côté [OA] : travailler sur des points nommés dans la figure permet un apprentissage plus précis, plus rapide.

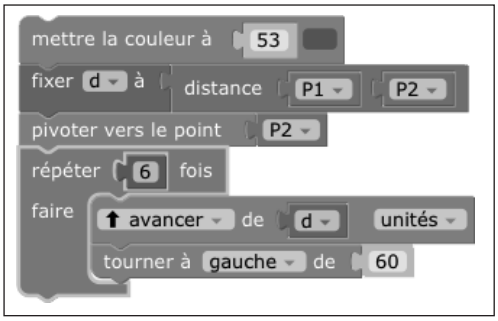
Un autre aspect du dynamisme de la tortue, est que, justement, elle s'inscrit dans un environnement dynamique, et donc que l'on peut utiliser ces points préalablement construits, *et les déplacer ensuite* : la trace n'est pas statique, elle est bien sûr dynamique. Ainsi, cette question du même algorithme pour deux constructions est parlante dans un environnement dynamique, avec des points de référence, car on voit que l'on utilise explicitement ces points, que l'on peut les déplacer, alors que la question ne se pose pas vraiment dans une version non dynamique de la tortue.

La tortue dynamique de DGPad propose un nouveau paradigme d'exploration qui se construit dans une rencontre entre la géométrie dynamique et ce que l'on pourrait appeler la *programmation visuelle dynamique* (PVD). Nous allons en donner quelques exemples.

3. — Exemple d'activités 2D

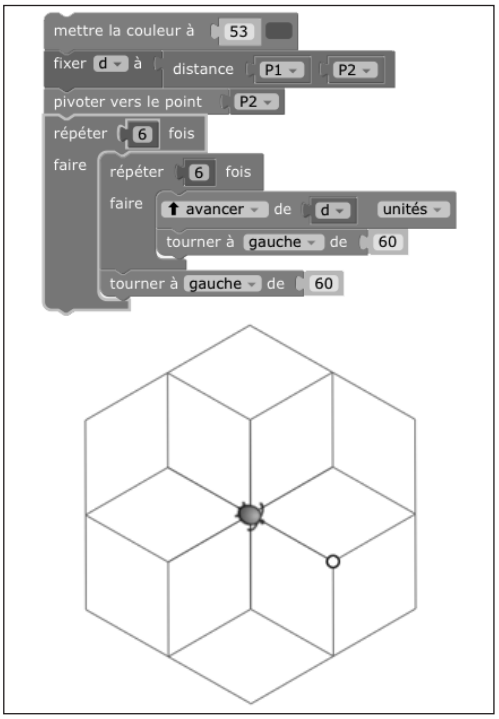
3.a. *Un nouveau changement de cadre entre l'arithmétique et l'algèbre*

L'idée de cette séquence – qui s'inscrit dans une continuité avec un travail préalable – est de commencer par une activité « arithmétique », avec des nombres bien précis, puis ensuite d'étendre les constructions à des « nombres généralisés » que peuvent être les curseurs d'entiers. Le fait que l'on produise des figures intéressantes, probablement inédites pour les élèves, participe de la dévolution de l'activité.



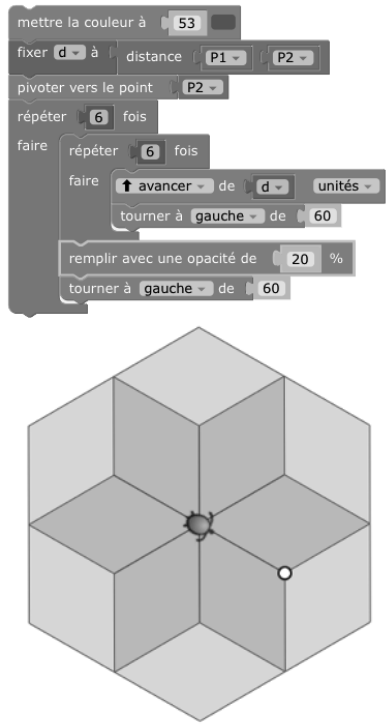
Dans une première phase (encadré précédent), on commence par prendre deux points (par défaut ils s'appellent P_1 et P_2) et sur le premier point, on construit un hexagone régulier de côté $[P_1P_2]$…

Dans une deuxième phase, on se propose de l'itérer 6 fois – comme ci-dessous – en tournant de 60° à chaque itération.



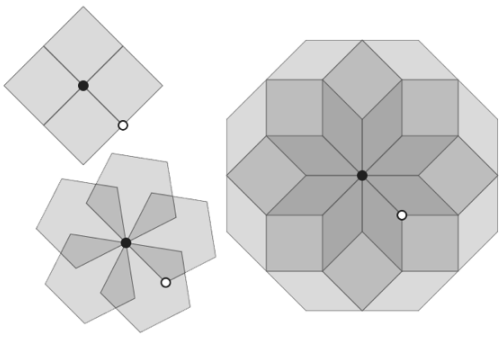
Dans une troisième phase, on explore « manuellement » la situation, en modifiant les angles et le nombre d'itérations pour travailler, essentiellement, sur un carré et un octogone (éventuellement un pentagone). C'est l'occasion – selon la classe – de simples observations ou de conjectures qui peuvent être interrogées : pour les polygones avec un nombre pair de côtés, on reproduit, finalement, le même polygone dans une taille double (simple constatation en collège). Les premiers questionnements peuvent porter

Concrétisation de la visualisation des rotations de l'hexagone : La figure obtenue est trop complexe pour être analysée de manière pertinente. Pour mieux voir ce qui est fait, on décide de colorier le polygone construit, en ajoutant seulement une instruction de **remplissage** dans la première boucle (la plus profonde) après le dessin du polygone.



Les élèves peuvent alors décrire ce qu'il se passe, d'abord d'un point de vue opérationnel en terme de variation de couleur. Ensuite, interpréter cette variation, en termes géométriques, avec plus ou moins de précision.

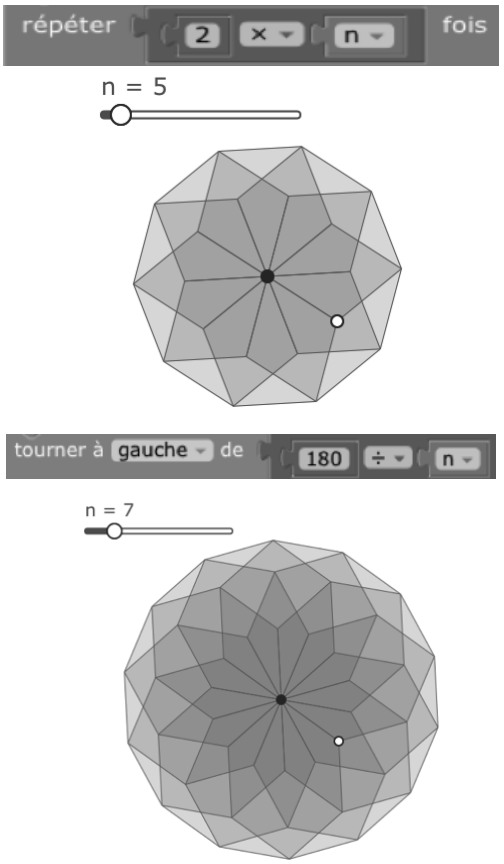
sur le nombre de teintes différentes, et leur régularité dans les cas pairs, et pourquoi il n'y pas cette régularité avec un nombre de côtés impair. Selon la classe, on peut attendre des réponses très simples, de l'ordre de la plausibilité seulement, ce qui est, didactiquement, tout à fait acceptable dans ce contexte, l'essentiel étant la curiosité et le questionnement.



Pourquoi apparaît-il un carré dans la construction avec l'octogone ? Certains étudiants de M2 MEEF en TD de TICE ont trouvé cette question délicate. On voit donc qu'il faut, relativiser les possibilités de questionnement, mais aussi prendre conscience qu'on explore un champ qui n'est pas encore dans l'environnement scolaire et que c'est l'occasion de construire des activités (et une culture) sur les figures partielles extraites.

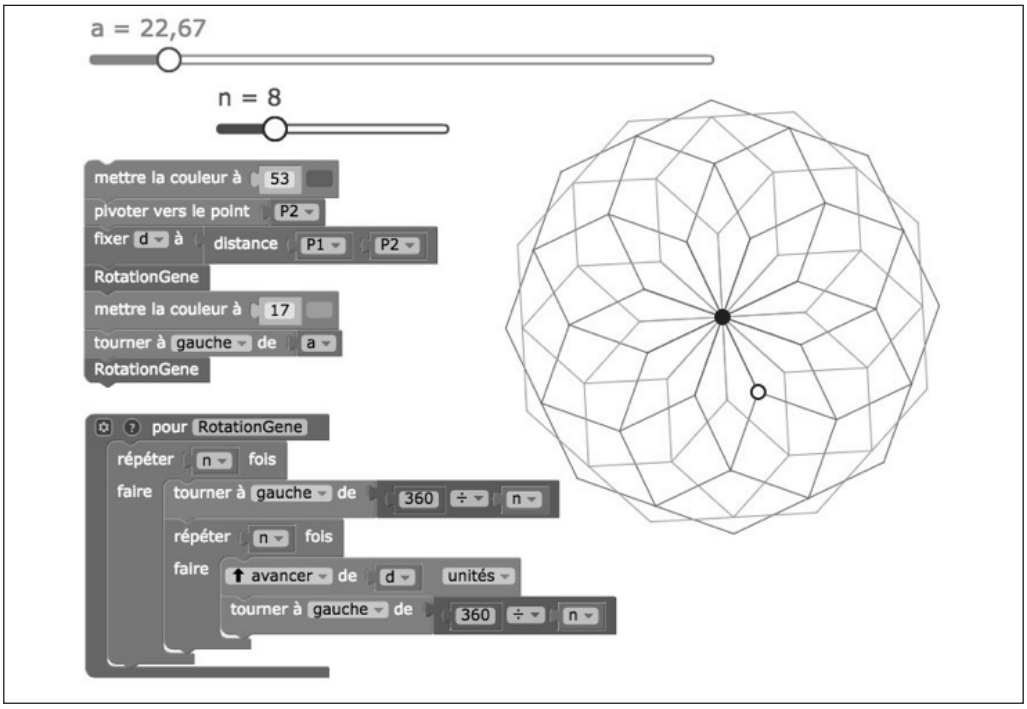
Toutefois l'intérêt de cette séquence n'est pas tellement dans ces questionnements. Il est surtout dans l'introduction, à la phase suivante, d'un curseur entier qui sert de variable du nombre de côtés du polygone. La tâche est alors de remplacer les nombres, en particulier les angles, par des expressions algébriques. (huit.re/RotPoly1)

On peut poursuivre dans des variantes exploratrices, par exemple en doublant le nombre d'itérations de la boucle extérieure et divisant par deux l'angle de la rotation qui passe d'un polygone au suivant : (huit.re/RotPoly2)



On n'est plus dans l'analyse de la figure – trop complexe – mais plutôt dans une démarche de « gamification » (passage à des jeux sérieux) de l'activité géométrique.

Pour montrer au lecteur les possibilités de cette tortue dynamique, poursuivons, hors démarche scolaire : dans l'illustration suivante, on place le code initial précédent (de Poly1)

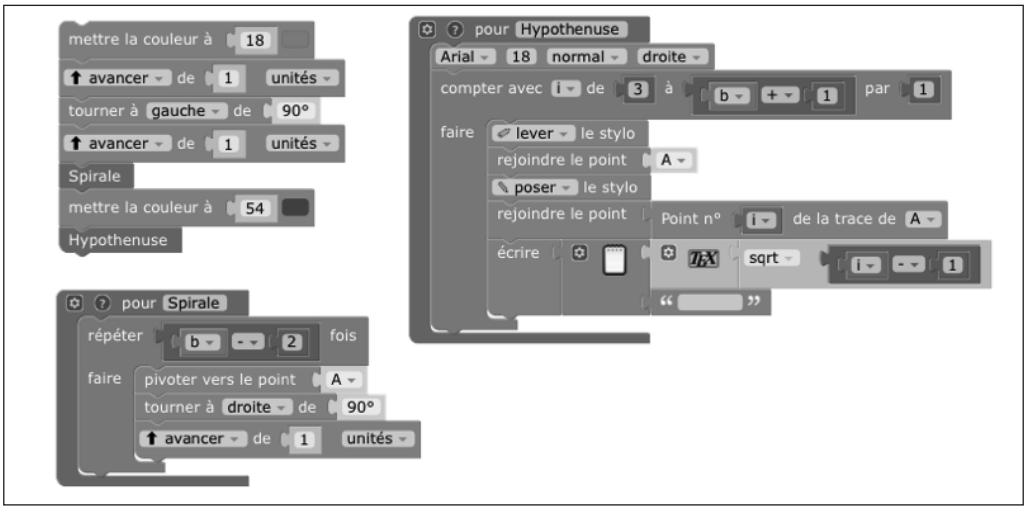


dans une procédure *RotationGene*, et, en ajoutant un curseur d'angle de rotation (*a*), on peut construire la figure et sa rotation dynamique en ajoutant deux lignes, comme ci-dessus (huit.re/RotRotPoly1).

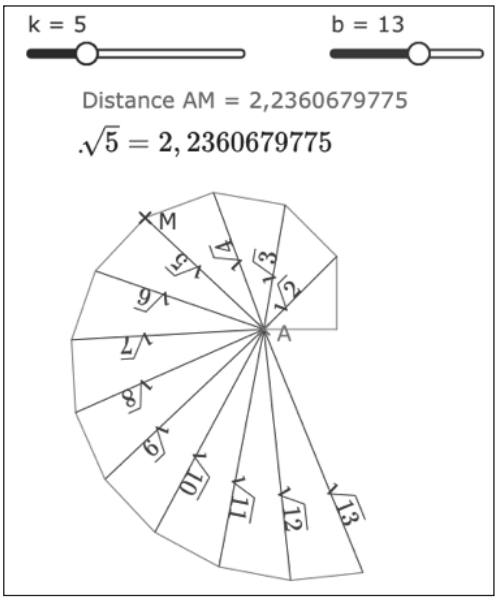
2.b. *Exemple simple d'utilisation de* LaTeX

La spirale de Pythagore est un bel exemple de subtil changement de cadre entre géométrie et programmation. Dans la procédure *Spirale* suivante, si on utilise l'angle droit – celui de l'équerre du dessin papier crayon – on ne trace, dans un premier temps, que la spirale seule, sans l'hypoténuse. C'est donc la direction de la tortue (*pivoter vers A*) qui fait office de trait virtuel d'appui de l'angle droit. En ce sens cet exercice est particulièrement intéressant.

La procédure *Hypoténuse* est ici présentée (encadré page suivante) dans une version sophistiquée pour montrer l'utilisation du TeX. En classe on peut simplement chercher à la construire sans faire coïncider l'indice avec la racine carrée. Le principe retenu est de calquer la version papier-crayon, et donc de n'utiliser que des procédés géométriques élémentaires pour, ensuite – dans une version enseignant d'illustration – vérifier de résultat de Pythagore. Dans cette procédure on retourne en A et on rejoint un point de la trace précédente. La difficulté du côté de l'élève est de bien choisir les indices. On voit aussi qu'une trace peut se définir à partir des éléments précédents d'elle-même, à la manière d'une suite récursive, sans que le concept ne soit évoqué, il est seulement « en acte » ici.



Une version finale « enseignant » – i.e. non construite par des élèves – permet ensuite de vérifier que la distance d'un point de la spirale au point A d'origine est bien égale à la racine carrée annoncée :



Ceci est possible car la figure initiale est construite dans le repère des unités du logiciel (la tortue avance d'une unité), indépendant d'un zoom à la souris par exemple. En pratique on se donne un curseur entier (k) et un point M préconstruit. Et sur M on place le comportement qui le positionne à la (k + 2)-ème position de la tortue. C'est encore un autre aspect, plus technique, du dynamisme de la tortue.

huit.re/BalladeSurPytha

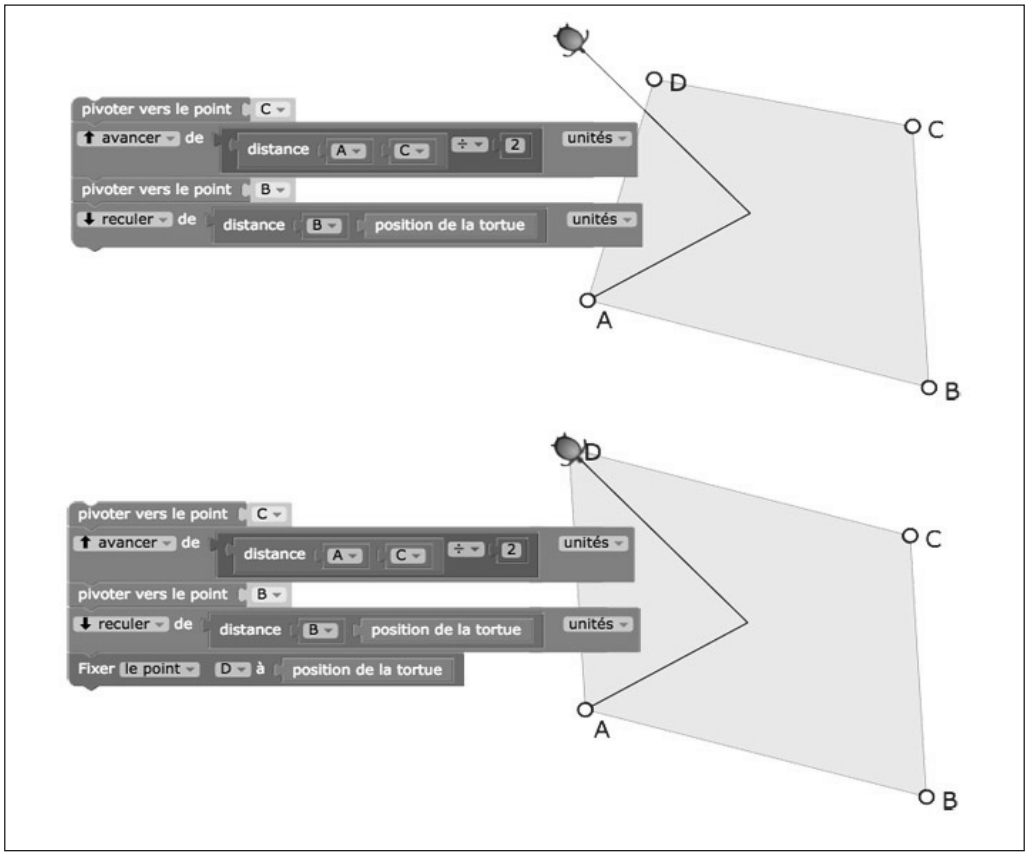
2.c. *Quand la tortue devient elle-même algébrique*

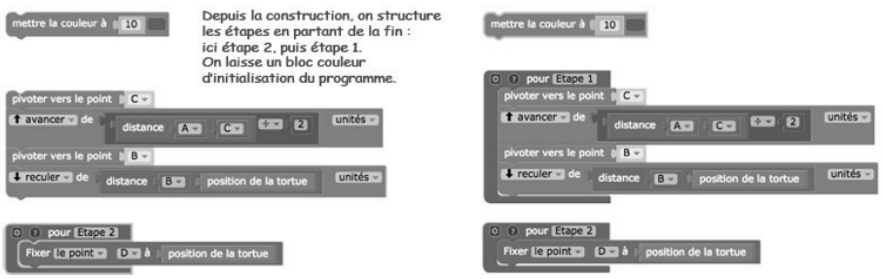
Une autre approche tout à fait nouvelle, et didactiquement très prometteuse, de la tortue dynamique réside *dans l'utilisation de la position de la tortue*. Le fait que l'on soit dans un environnement dynamique peut donner à cette position, selon les activités, soit une dimension arithmétique, soit une dimension algébrique. On peut même utiliser ce type de programmation comme travail d'approche de la notion de variable.

Voyons un exemple de base sur le parallélogramme. Comme la tortue ne crée pas d'objets DGPad, on se propose de modifier – par un comportement – le sommet D d'un quadrilatère ABCD pour qu'il devienne un parallélogramme. Le principe est le suivant : la tortue part de A, elle parcourt la moitié de AC, puis pivote vers B et recule de la distance entre elle et le point C. Dans la dernière instruction, la position de la tortue est, de fait, une variable qui est utilisée dans un programme de construction géométrique. Ensuite, le point D est positionné, par la tortue, à sa position.

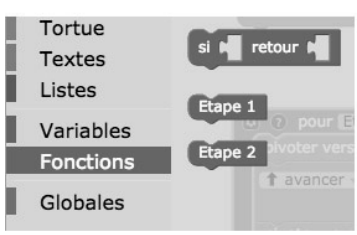
Dans le cadre d'un travail sur la communication, on peut introduire la notion de procédure qui ne sert ici qu'à rendre compte des différentes étapes à effectuer. Voir le déroulement résumé dans l'encadré de la page suivante…

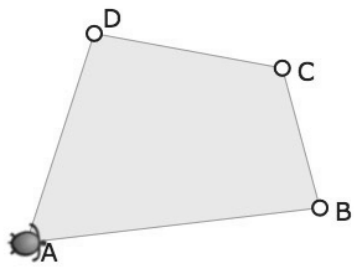
On peut envisager des exercices plus subtils, avec des calculs intermédiaires. Par exemple regardons cette mission de la tortue : partant de A la tortue doit transformer les sommets B et D du quadrilatère ABCD pour qu'il devienne un losange de diagonale [AC] et de côté AB. (figure en ligne : huit.re/QDvsLosange)

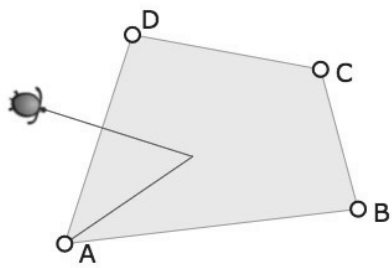




Ce qui peut s'appliquer ainsi :







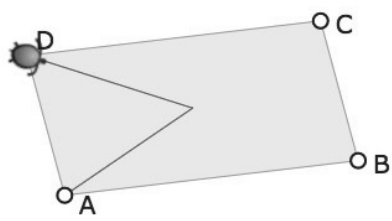
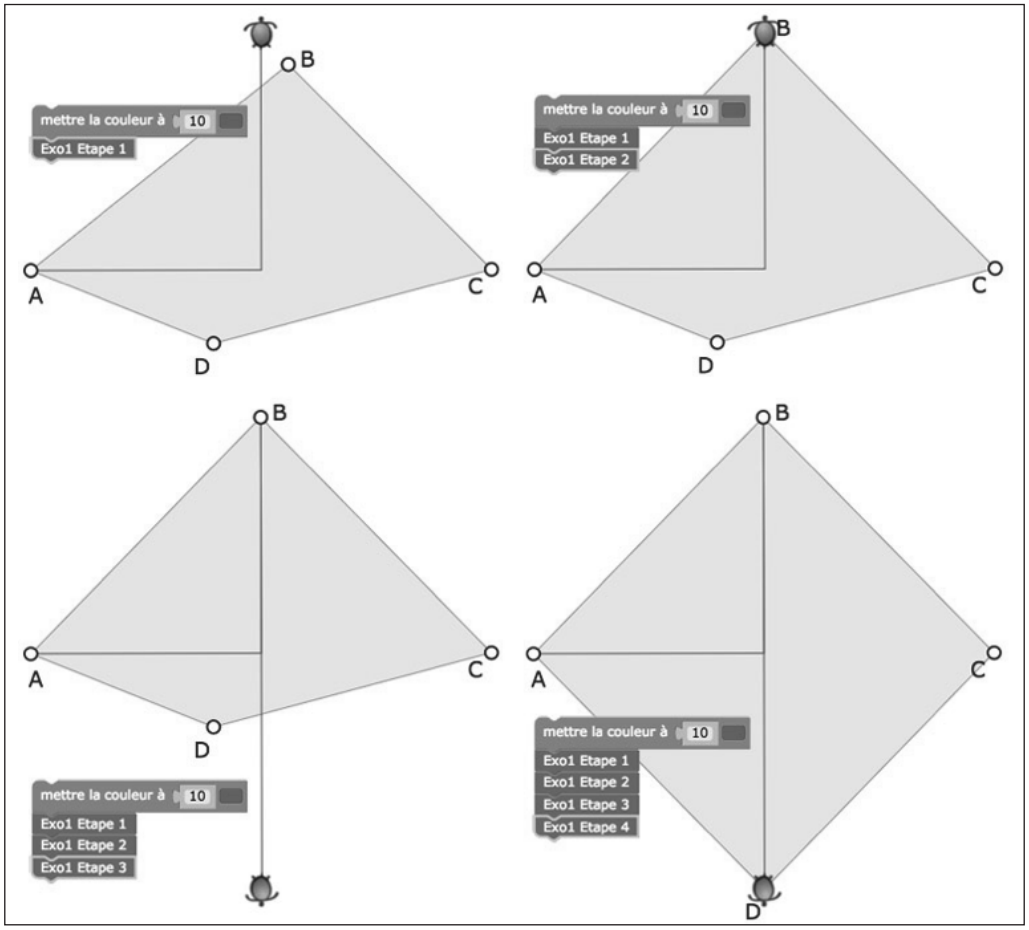
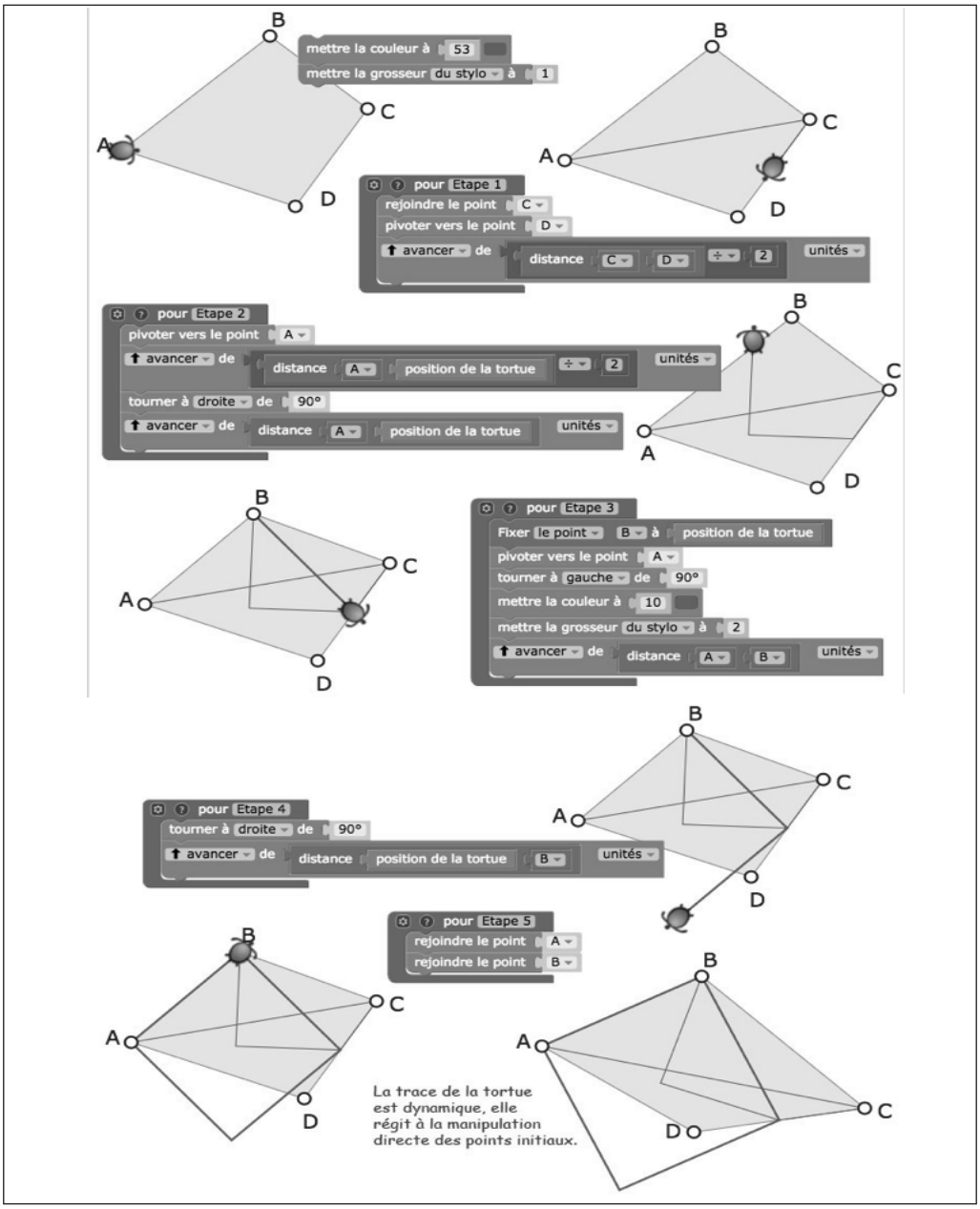


Figure en ligne : huit.re/QDvsParallelo

L'exercice est décrit symboliquement ci-dessous en quatre étapes. Seule l'étape 1 est un peu délicate car il faut utiliser le résultat de Pythagore pour redéfinir la position de B. L'activité est algorithmiquement subtile, même si cela peut passer inaperçu en classe, car on commence par positionner la tortue à la future place B, calculée avec Pythagore et la longueur AB (Etape 1), puis on déplace B sur la position en cours de la tortue (Etape 2). Non seulement cela fonctionne (pas de boucle logique) mais de plus c'est tout à fait dynamique : si on déplace C, la position de B est recalculée et remise à jour. Cette subtilité de l'implémentation de Blockly dans ce contexte dynamique sera développé plus conceptuellement dans la dernière partie de cet article.

Entre ces deux extrêmes de nombreuses variantes sont possibles, surtout si, finalement,



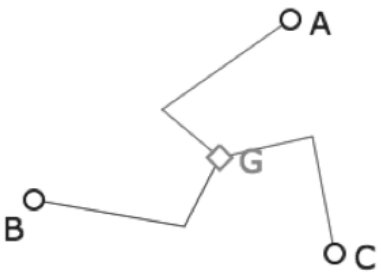

mettre la couleur à 53
mettre la grosseur du stylo à 1
pour Etape 1
rejoindre le point C
pivoter vers le point D
avancer de distance C D ÷ 2 unités
pour Etape 2
pivoter vers le point A
avancer de distance A position de la tortue ÷ 2 unités
tourner à droite de 90°
avancer de distance A position de la tortue unités
pour Etape 3
Fixer le point B à position de la tortue
pivoter vers le point A
tourner à gauche de 90°
mettre la couleur à 10
mettre la grosseur du stylo à 2
avancer de distance A B unités
pour Etape 4
tourner à droite de 90°
avancer de distance position de la tortue B unités
pour Etape 5
rejoindre le point A
rejoindre le point B
A
B
C
D
La trace de la tortue est dynamique, elle régit à la manipulation directe des points initiaux.

on ne cherche pas à déplacer les points (il faut faire attention à la récursivité qui guette, on est précédemment — page 35 — à la frontière du possible) mais plus à définir une figure, sous forme de trace, à partir d'une autre.

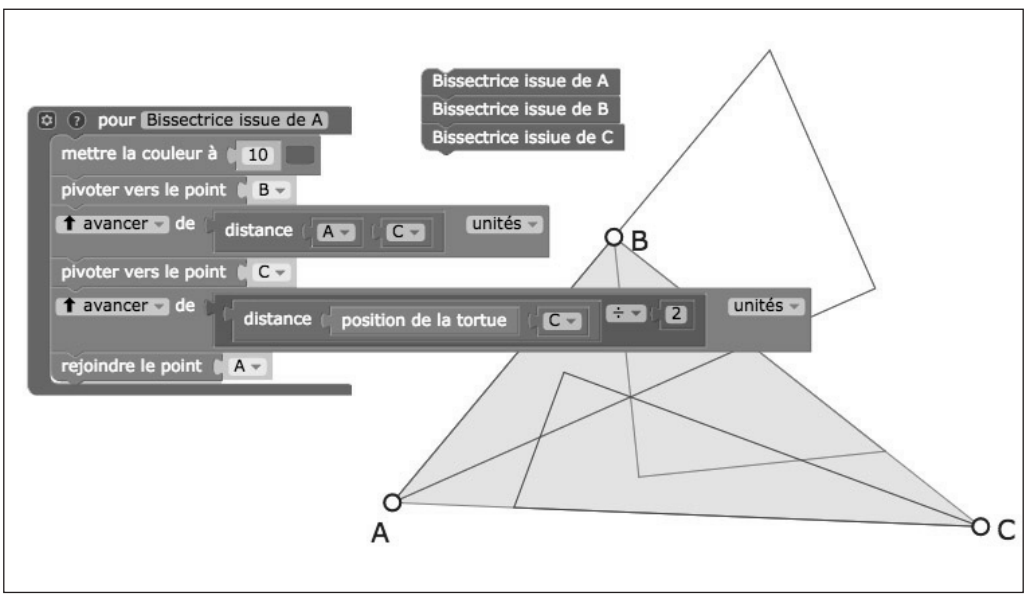
Voici un exemple de ce type d'activité : *Soit ABCD un quadrilatère. La nouvelle mission de la tortue est de réaliser un carré de sommet A et de sommet opposé à A le milieu de [CD]. Elle doit redéfinir B comme sommet du carré, et terminer le carré. La tortue laissera visible toutes ses traces préliminaires ou intermédiaires en bleu, le carré final sera en rouge, avec ses 4 côtés.* (Encadré page 36 et : huit.re/QD_carre1)

L'activité peut même être structurée autour d'un défi qui proposerait de construire la figure en un minimum de lignes de programme.

L'outil « position de la tortue » n'algébrise pas nécessairement une situation. Parfois elle reste dans un registre arithmétique comme par exemple cette nouvelle représentation de l'intersection des médianes au tiers depuis le milieu du côté opposé :



Voici, dans le cadre ci-dessous, la construction à la tortue, dans une version élève, des bissectrices d'un triangle à partir de la propriété de médiane d'un triangle isocèle… On voit sur ces exemples élémentaires combien



cette rencontre entre la géométrie dynamique munie d'une tortue et la programmation visuelle, quand elle est, elle aussi, dynamique comme ici, active une réflexion algorithmique originale et féconde qui invite à une exploration didactique plus approfondie.

4. — Tortue 3D dynamique

Il existe déjà une tortue 3D dans GéoTortue. Le site propose la construction générique des polyèdres réguliers et de plusieurs polyèdres archimédiens. Toutefois le code des figures s'inscrit dans le contexte d'un repère absolu, avec un retour systématique à l'origine entre les faces. Dans un contexte dynamique, la géométrie de la tortue 3D prend une tout autre dimension. Elle simplifie énormément les constructions et offre, comme en 2D, un regard neuf, innovant, extraordinaire à explorer et à enseigner. Nous allons en donner quelques aperçus élémentaires, en restant dans une utilisation scolaire.

4.a. *Une première approche, le « pivoter vers le haut »*

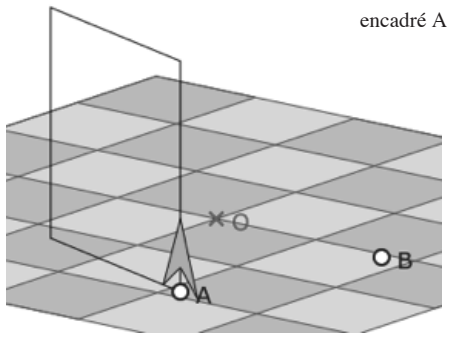
La tortue 3D est une flèche, verte sur sa carapace (le haut), noire sur le ventre (le dessous). Si l'on travaille des constructions pas à pas, par essais-erreurs, voir la tortue 3D pendant la construction est un atout important. Par ailleurs un outil de réinitialisation des angles permet aussi, parfois, de simplifier grandement la réflexion et d'aller plus vite dans les réalisations.

L'utilisation la plus simple est celle qui utilise des pavés droits, en pratique des cubes. En effet l'outil 3D le plus immédiat, car *perceptivement* le moins orienté, est le *pivoter vers le haut*. La tortue monte dans le sens de sa direction. Il n'est utile que si la tortue est orthogonale à l'axe de rotation, ce qui est naturellement le cas sur les faces carrées, même si ce point là reste une donnée non perçue explicitement par les élèves. C'est seulement quand on utilisera un autre outil, le *pivoter vers la gauche*, que cet implicite s'explicitera plus facilement.

Voici (cadre A ci-dessous) un exemple de construction d'un cube – et donc de ses 12 arêtes – par 4 carrés. Soient 2 points A et B sur le plan du sol, on se propose de construire un cube de côté [AB]. Pour cela on commence par construire une face, disons la « face avant » si la face contenant [AB] est une face à droite.

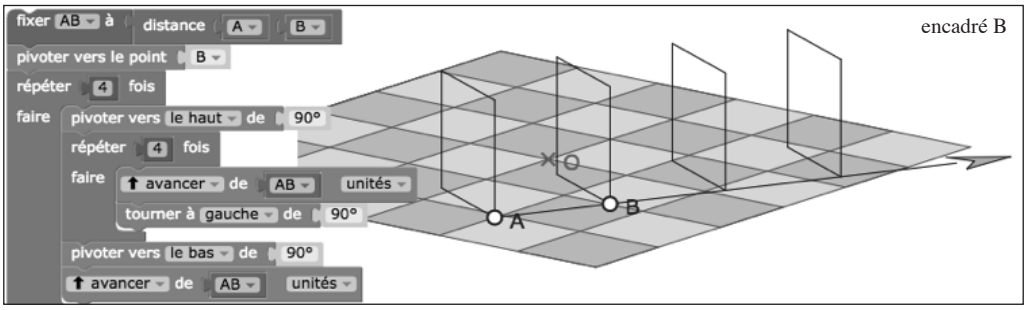
On voit bien les implicites opérationnels : pointant vers B, le *pivoter vers le haut de 90°*

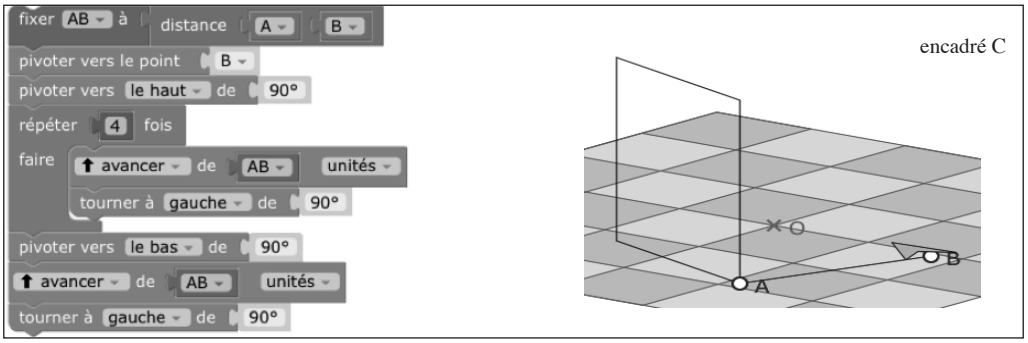
encadré A

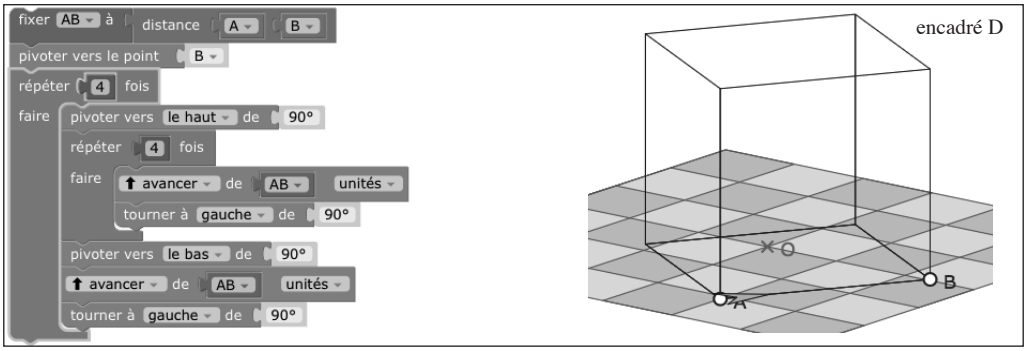


place la tortue dans cette face orthogonale à [AB]. L'utilisation didactique de cette opérationnalité implicite n'est pas un problème, c'est une simplification qui éventuellement, permet de rentrer dans une phase de recherche a-didactique. Ces implicites seront verbalisés plus tard, quand il faudra en rendre compte, ou quand ce sera plus efficace de faire autrement.

La position de la tortue permet d'anticiper sur la suite. Elle montre qu'il faut abaisser la tortue sur le sol, avancer d'une arête, puis tourner à droite, avant d'itérer ce qui précède et ces trois instructions, le tout quatre fois. Comme les constructions se font en temps réel, il faut choisir ce que l'on place en premier, soit l'itération (la boucle) soit les instructions nouvelles.







Dans le cas de la boucle placée d'abord (illustration cadre B), il faut un peu d'habitude, pour voir qu'il manque, dans l'état de la construction ci-dessus, seulement une instruction. C'est une nouvelle culture, celle de la tortue dynamique, qui se construit dans le temps. Ceci étant, il est plus simple, et didactiquement bien plus parlant, de placer (comme dans le cadre C) les trois instructions avant d'itérer le tout dans une boucle.

On voit mieux, compte tenu de la position de la tortue, orthogonale à l'arête [AB], qu'un simple « répéter 4 fois » construit un cube, à partir de 4 carrés : cadre D.

La construction étant terminée, on remarquera que la construction de 4 carrés permet de visualiser les 6 faces, mais que l'on n'a pas construit les 6 faces. Pour s'en rendre compte, il suffirait de remplir les carrés juste après la boucle interne.

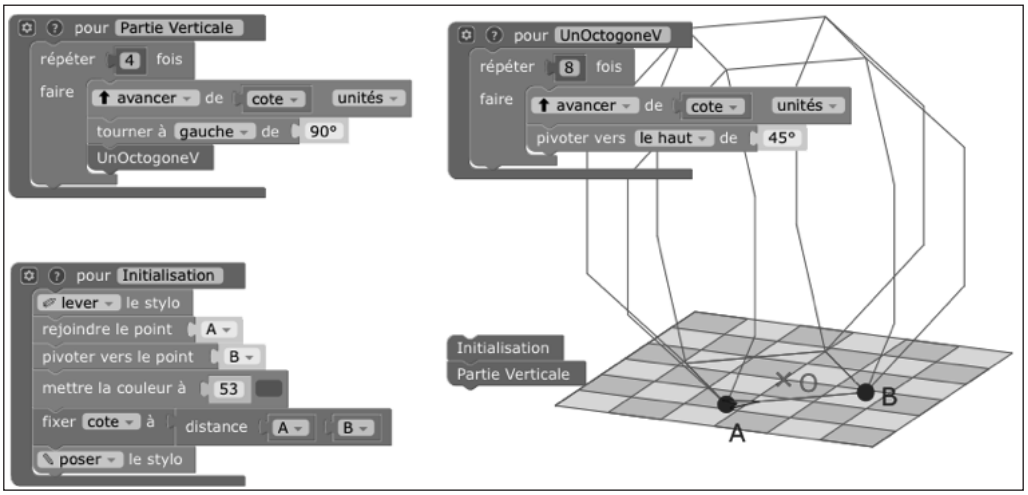
Figure en ligne : https://huit.re/Cube4Carres

En classe, on peut entamer une réflexion sur ce qui a été fait : combien a-t-on construit de segment ? Quelles sont les arêtes construites en double ?

Sur cette base-là, beaucoup de choses sont réalisables sans trop d'effort : le cuboctaèdre en ne traçant que 6 carrés, et même le rhombicuboctaèdre (ci-dessous) en ne traçant que 6 octogones réguliers, sans avoir à tracer de carrés.

Un objectif spécifique à cette dernière activité pourrait être de chercher une programmation compacte – on dira factorisée – afin de proposer aux élèves une décontextualisation de la factorisation (de l'algèbre) en l'appliquant à des procédures de construction. D'une certaine façon on peut dire que c'est, de la part d'un enseignant de mathématique, une instrumentalisation « algébrique » d'une réflexion naturelle sur l'optimisation du code d'une procédure.

Dans cette optique, voici (ci-dessous) une première partie de la construction, dont une factorisation est mise en œuvre dans la procédure *Partie Verticale* : le fait d'avancer et de tourner avant de tracer l'octogone vertical permet, en plaçant ces trois blocs dans une boucle,



de construire l'ossature verticale du polyèdre en quelques lignes.

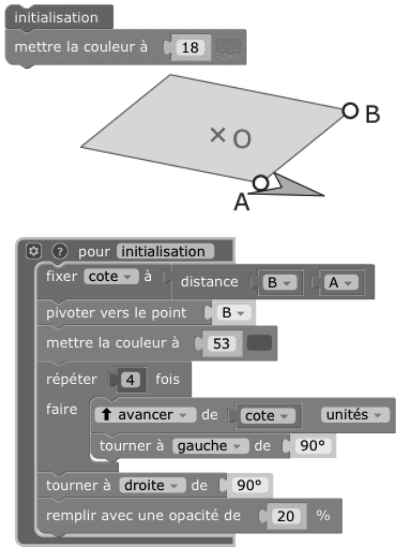
À la fin de la procédure, la tortue est en A, orientée vers B. Pour terminer la construction, il suffit de la placer au bon endroit, de réaliser une procédure *UnOctogoneH* et de l'appliquer deux fois.

Figure : huit.re/RhombiCuboAB (écrire OA à la place de AB pour *centré en O*)

4.b. *Un patron de cube pliable, à la tortue, en trois étapes*

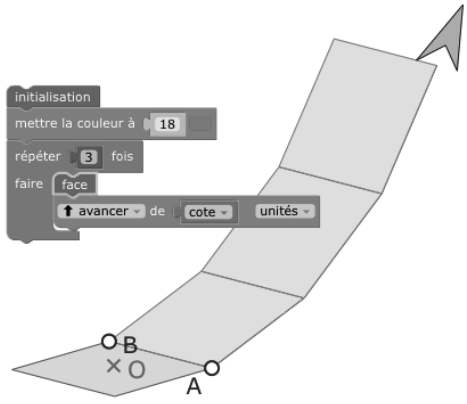
Toujours avec cette simple instruction « *pivoter vers le haut* », on poursuit vers la programmation de figures tout à fait surprenantes par leur simplicité technique.

Étape 1 : une procédure d'initialisation construit un carré de côté [AB] dans le plan du sol. On place la tortue orthogonalement à l'arête [AB] (le *tourner à droite* ci-dessous) :



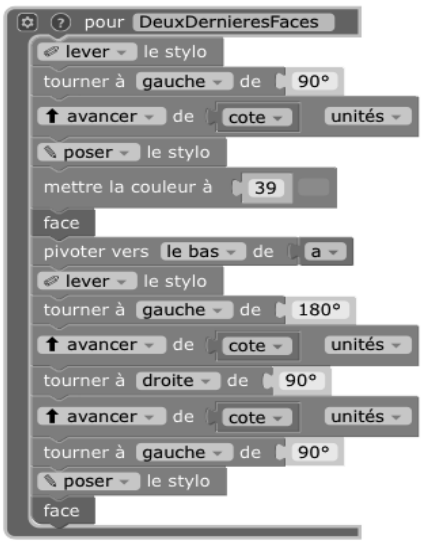
Étape 2 : un curseur **a** de 0 à 90° représente l'angle de pliage du patron. Dans cette étape on construit une procédure **Face** qui permet de factoriser la partie latérale du patron standard dans une boucle. Cette procédure **Face** n'est rien d'autre que la construction d'un carré précédée d'une rotation vers le haut de la tortue de l'angle **a** du curseur :

Il suffit (ci-dessous) de répéter la procédure puis d'avancer d'une arête : localement on pivote de l'angle **a**, sur l'arête qui convient.

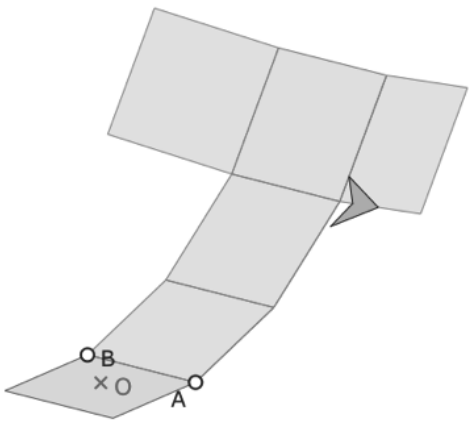


Le lecteur percevra sur cet exemple la simplification apportée par un travail en local, c'est-à-dire dans le repère propre de la tortue, par rapport à une démarche basée sur des produits de matrices 3x3 dans un repère absolu.

Étape 3 : ajout des deux dernières faces.



Il est facile de construire la fin du patron en regardant, pas à pas, où est la tortue : la démarche est donc opérationnelle. Chacun peut suivre la construction de la face gauche depuis la position précédente de la tortue.



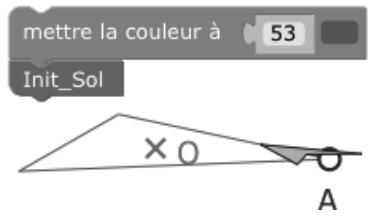
On arrive donc, avec une seule instruction 3D de la tortue, *pivoter vers le haut* (ou *le bas* une fois), sans aucun calcul, ni d'angle, ni de longueur, à construire un patron pliable de cube en quelques procédures élémentaires à réaliser.

Figure : huit.re/Cube1PatronAB
(OA à la place de AB pour *centré en O*)
Quatre patrons à la fois :
huit.re/Cube4PatronsAB
(ou OA à la place de AB)

4.c. *« Pivoter vers la gauche », l'instruction générique de la 3D des polyèdres*

L'exemple précédent est simple car le *pivoter vers le haut* s'applique toujours. En effet, la tortue étant orthogonale à l'arête à chaque sortie de « Face », cela correspond à une rotation selon un axe orthogonal à la direction de la tortue. Sur des exemples plus généraux, on utilisera plutôt un **« pivoter vers la gauche »** qui correspond à une rotation *dans l'axe de la tortue*. En pratique, pour toutes les autres situations que le cube et le pavé, c'est cette instruction qui est à privilégier. À titre d'illustration, montrons – hors cadre du collège, plutôt pour le lycée – comment réaliser le patron standard pliable d'un tétraèdre régulier :

On suppose avoir déjà fait une procédure d'initialisation avec un triangle équilatéral dans le plan du sol :



La tortue est sur une arête de la face du sol. Pour tracer une face (2 côtés seulement, sans finir la face, pour pouvoir itérer plus facilement), il suffit de :

- **pivoter** sur l'arête de l'angle voulu (ligne 1),
- tracer les deux côtés du triangle équilatéral (les lignes 2 à 5 ci-contre),
- remettre la tortue dans le plan de l'arête suivante, dans la bonne direction, pour itérer la procédure (les 3 dernières lignes ci-dessous).

Visualisons ces trois dernières lignes :

Illustration 1 : on vient de finir la face par le tracé de la seconde arête. La tortue est donc dans le plan de la face, dans la direction de cette seconde arête.

Illustration 2 : par une rotation (**tourner**, dans le plan de la face) de 30°, on place la tortue orthogonalement à la face tracée et donc à l'arête de cette face au sol.

Illustration 3 : un **pivoter vers le haut** de l'angle du curseur replace alors la tortue dans le plan du sol (car elle est déjà orthogonale à l'arête de la face, au sol).

Illustration 4 : par une rotation (**tourner**, dans le plan du sol) on positionne la tortue sur l'arête suivante, ce qui permet ensuite d'itérer la procédure.

Illustration 1

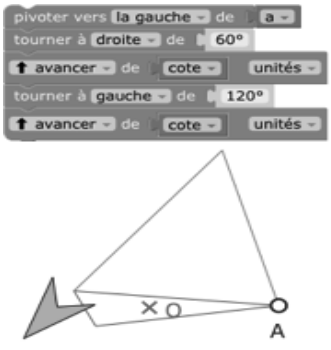


Illustration 2

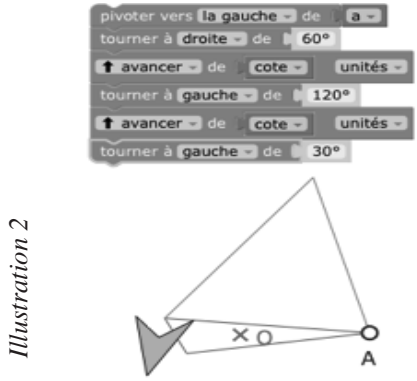


Illustration 3

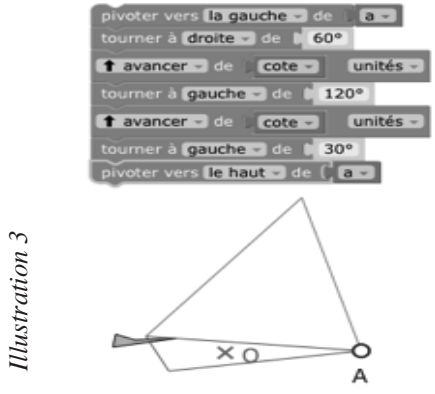
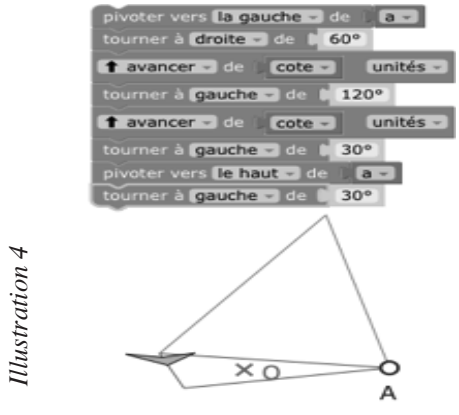


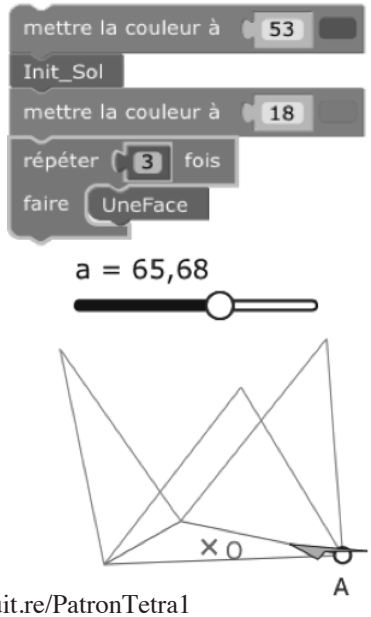
Illustration 4



Ces illustrations montrent à quel point la visualisation de la tortue en temps réel est un apport didactique essentiel et comment elle permet (et oblige à) une appropriation fine de l'espace, au moins, par exemple, en formation initiale des enseignants.

Mathématiquement, ces trois opérations ne sont, bien entendu, pas commutatives. Cette procédure **UneFace** illustre aussi que la composée de rotations dans l'espace n'est pas commutative, et la présence de la tortue montre clairement l'ordre dans lequel il faut effectuer les opérations : cela peut même être utilisé comme outil d'illustration en formation initiale des étudiants en licence sur le produit de matrices.

Pour terminer ce patron, il suffit d'itérer ce qui vient d'être fait :



huit.re/PatronTetra1

On peut voir aussi cet autre patron :

huit.re/PatronTetra2

Cette technique s'applique sans problème à des figures plus complexes (hors contexte scolaire ou de formation), comme par exemple :

Trois patrons du dodécaèdre : huit.re/Dodeca_3patrons.

Cinq patrons du ballon de football (32 faces) : huit.re/Ballon_5patrons.

Même figure en version animée : huit.re/Ballon_5Panim.

Patron du snubdodécaèdre (92 faces) : huit.re/Snub_Dodeca.

Patron du rhombicosidodécaèdre (62 faces) : huit.re/Rhombicosidodeca_Patron

Même figure en version animée : huit.re/Rhombicosi_Anime.

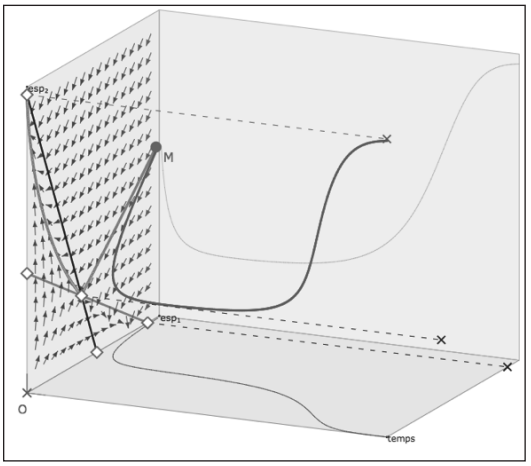
5. — Les possibilités du Blockly de DGPad hors contexte tortue

L'adaptation de Blockly dans un logiciel de géométrie dynamique permet un enrichissement surprenant du logiciel initial, voyons quelques exemples.

2.a. *Exemple d'utilisation en cours à l'université, en L1 GéoSciences*

Dans cet exemple, il s'agit d'illustrer de manière dynamique, en manipulation directe, une simulation du modèle proie/prédateur dans une variante dite « de compétition » (avec deux lois logistiques) du modèle initial de Volterra. Le contenu de ce cours n'est pas particulièrement technique mathématiquement, la maquette insiste sur la dimension «descriptive » des mathématiques abordées. Il s'agit donc surtout de donner à voir et à mani-

puler en TD autour de ces phénomènes. L'intérêt ici est d'observer, dans le cas d'un point répulsif, le comportement temporel du modèle : sur la trajectoire (verte - du champ de vecteur), on ne voit pas réellement que le point mobile M frôle longtemps le point répulsif avant la répulsion, c'est-à-dire pendant de nombreuses itérations – donnée manipulable avec un curseur. Plonger la trajectoire classique dans sa temporalité de courbe intégrale est une illustration très enrichissante. On voit réellement le point proche du point d'équilibre répulsif, sur la durée. Cette figure est faite avec Blockly, sur des listes de segments, l'interface générale de la géométrie dynamique permettant de faire varier tous les paramètres de la situation.

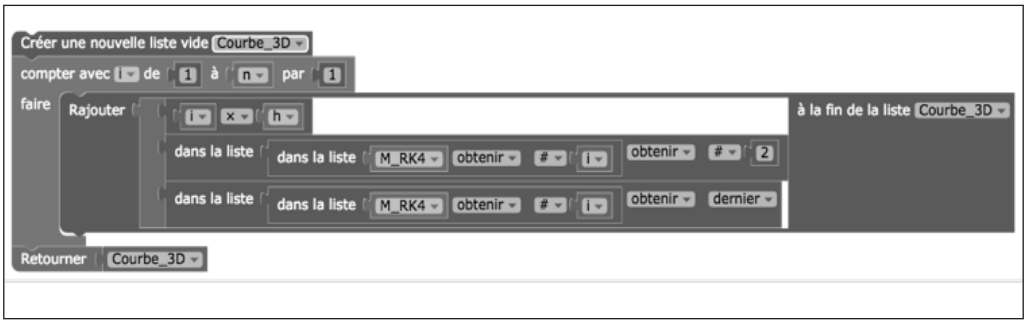


Dans l'illustration ci-contre :

- le champ de vecteurs de l'espace de phase (du plan de la trajectoire) est produit par un aspect des listes de segments qui n'existe que dans l'environnement Blockly (ajouté amicalement par l'auteur pour ce cours) ;
- la trajectoire verte est produite par une programmation de Runge Kutta du système, en Blockly (la seule chose non triviale de cette figure) ;
- La courbe intégrale rouge n'est qu'un jeu d'écriture avec la liste précédente.

La courbe intégrale permet de voir que, pour certaines positions initiales de M, le point répulsif, intersection des deux isoclines, est longuement approché avant de devenir brusquement répulsif. Sur la figure on a fait apparaître un segment pointillé pour voir le point d'équilibre dans le temps.

En termes de programmation, la courbe rouge (3D) se construit à partir de la trajectoire verte de la façon suivante : on introduit le temps (discrétisé par le compteur *i*) en première coordonnée, et on déploie les coordonnées de la trajectoire (liste M_RK4) dans le temps :



Remarque : c'est le comportement d'une liste *qui existe*. Le terme *créer* de la première ligne signifie essentiellement « initialiser ». On ne crée pas, on modifie les objets.

La figure standard (ci dessus) :
https://huit.re/Competition3D1

La même avec animation temporelle :

https://huit.re/Competition3Dt1

Voltera simple :

https://huit.re/Volterra3D
(ou 3Dt pour animation)

Volterra logistique :

https://huit.re/LVLogistique5

[la manipulation directe – du point M, des curseurs – est beaucoup plus rapide sous Chrome que sous Firefox par exemple]

5.b. *Auto-référence et rupture du déterminisme*

L'utilisation de Blockly, dans un logiciel de géométrie dynamique peut être bien plus conceptuelle. En effet celui-ci autorise l'autoréférence des objets (essentiellement des points et des listes), et cela permet de rompre le déterminisme des logiciels de géométrie dynamique (GD dans la suite), et donc d'accéder à d'autres champs d'investigation, en particulier dans des réalisations plus didactiques par exemple. Voyons cela en détail.

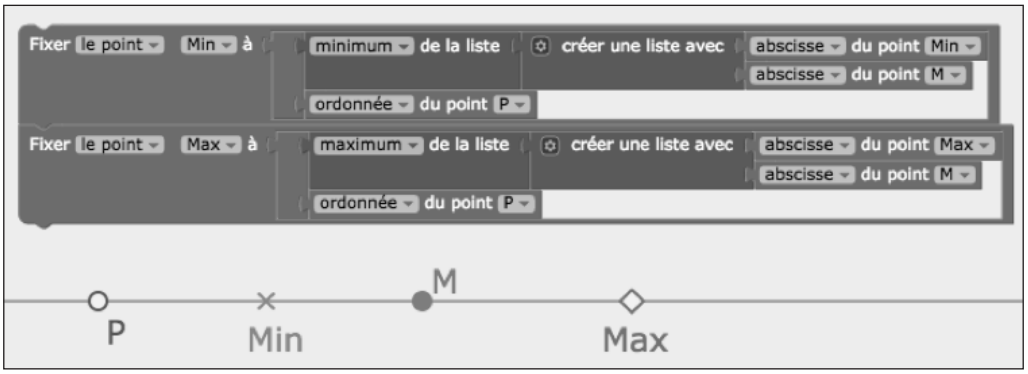
Déterminisme et continuité

En GD, on parle de *déterminisme* pour signifier qu'une figure reprend sa position initiale quand les objets initiaux reprennent leurs positions initiales. Cela paraît être un minimum du cahier des charges d'un logiciel de géométrie dynamique (avec d'autres propriétés) mais cela ne va pas de soi car cette propriété est, en général, en contradiction avec la demande de *continuité* – et en particulier de continuité des intersections. Par exemple la version 1 de Cinderella, le logiciel de Ulrich Kortenkamp, était totalement continu (en particulier les intersections de coniques étaient continues !) et donc non déterministe, ce qui n'était pas un problème pour l'utilisation au niveau d'enseignement où il était utilisé (Bac +4). La version 2 de Cinderella – celle actuellement disponible en ligne – est déterministe, pour une utilisation scolaire.

Une des tâches des pionniers de la géométrie dynamique a donc été de concilier la contrainte du déterminisme et des choix pertinents de continuité. Pour cela l'auteur de Cabri Géomètre, Jean Marie Laborde, avait institué un engagement direct très subtil sur les intersections : quand on prend dans Cabri, une intersection droite-cercle « à la volée » (ie sans créer les deux intersections par l'outil d'intersection), il y a continuité de l'intersection retenue même quand le rayon du cercle (par exemple défini par 3 points) passe par l'infini et que le cercle change d'orientation. Cette technique a depuis été reprise par tous les logiciels, et parfois systématisée (au moins par GeoGebra).

Exemple de rupture de déterminisme

Avant DGPad, seul CaRMetal avait déjà une potentialité d'autoréférence des points qui permet de briser le déterminisme d'une figure d'une manière ciblée, choisie par l'utilisateur, tout en permettant un retour au déterminisme si nécessaire, dans la même figure. Cette fonctionnalité était présente dans son noyau CaR magistralement écrit par René Grothmann. La figure dite « du thermomètre à mémoire », archétypique de cette situation, a été déjà présentée sous CaRMetal. Voici une version DGPad, réalisée en Blockly : sur une droite horizontale, pilotée par un point *P*, on place un



point *M* et des points *Min* et *Max* qui peuvent être créés au préalable n'importe où sur la page. Ils vont représenter les abscisses minimales et maximales que va prendre le point *M* dans une manipulation de l'utilisateur. Le point *Min*, restant au minimum du déplacement de *M*, y compris quand *M* revient à une position initiale, il y a rupture de déterminisme et la figure ci-dessus contient en mémoire (une partie de) l'action de l'utilisateur. Ce n'est donc plus de la géométrie que l'on fait, mais de la cinématique à mémoire. L'écriture est toute simple, car elle est auto-référente, le point *Min* est défini à partir du point *Min*. Le tour de force est du côté du programmeur qui a fait en sorte que ceci fonctionne (alors que Blockly justement a priori ne le permet pas, voire même pas du tout, au contraire).

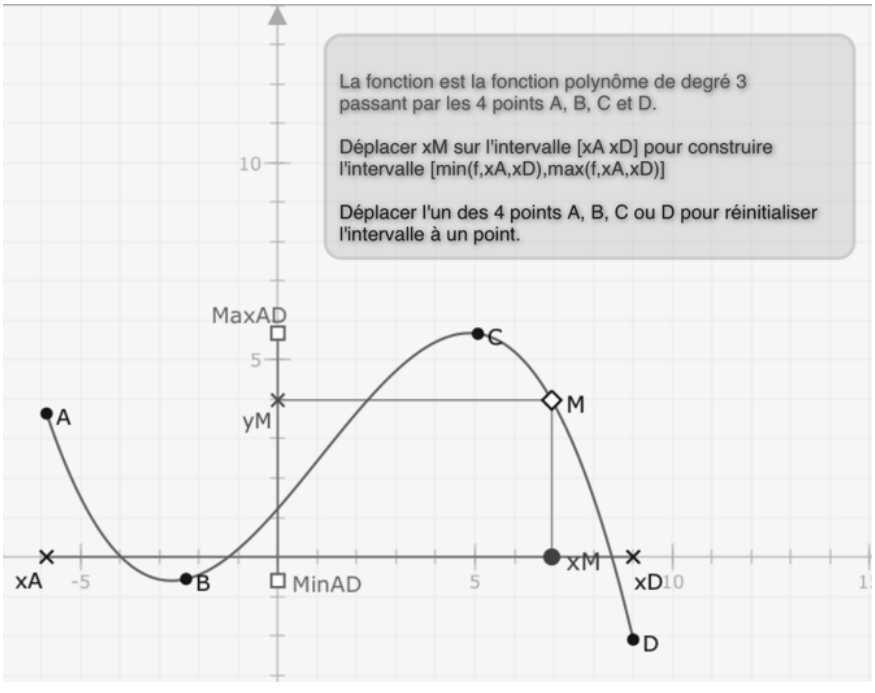
On peut retrouver une position initiale (ou en fixer une) en activant, sur le point P, le comportement de déplacement suivant :

Utilisation en ligne : huit.re/Thermo1

Exemple d'utilisation scolaire : l'entrée dans le champ fonctionnel en début de lycée confronte l'élève à un nouveau paradigme, avec des propriétés qui ont un sens global (sur un intervalle, comme le sens de variation) alors que, avec l'apprentissage de l'algèbre, les problématiques des années précédentes étaient uniquement locales. La difficulté est renforcée dans la mesure où, pour enseigner le *global*, on va se centrer sur du *local* (dérivation en un point) et du *local universel* : une propriété locale vraie en tout point d'un intervalle (dérivable sur un intervalle), concept que les élèves identifient souvent à la définition d'une propriété globale.

La méthode précédente permet de travailler cette question essentielle de différentiation entre le global et le local universel. Voici une visualisation explicite des extrema locaux (en construction) d'une fonction par prégnance kinesthésique des positions extrêmes des ordonnées.

Bien entendu nous sommes ici loin, très loin, des problématiques fondamentales sur les questions de réification des concepts mathématiques, et de cette époque où les mathématiciens (Frege entre autres) ont fait sortir l'étude des fonctions de la cinématique du point, y com-



Manipuler la figure en ligne : huit.re/Extrema (on peut placer A, B, C, D pour illustrer qu'un extremum local n'est pas nécessairement global). De nombreuses variantes sont possibles.

pris dans les programmes scolaires, puisque c'est bien cela que cette figure réintroduit, comme support *temporel* de représentation d'un concept *global*.

Cette figure est alors une illustration concrète des propos de Michel Serre, dans « Petite Poucette[2] », quand il explique que « le cognitif procédural »… « celui que j'appelle Petit Poucet… l'emporte sur Socrate ! Retournement plus que millénaire dans la présomption de compétence ! »

Conclusion

Après avoir exploré rapidement la pertinence de la notion de *tortue dynamique* en 2D, nous venons de voir la puissance que ce concept déploie en 3D, avec simplicité, sans calculs, en installant un rapport direct avec les problématiques de la géométrie dans l'espace, y compris au lycée et au-delà si on le souhaite.

Il serait intéressant de faire connaître plus avant cette *tortue dynamique* construite au cœur même du nouveau programme de collège, dans une rencontre entre deux outils en manipulation directe que sont la géométrie dynamique et la programmation par bloc, rendant, aussi cette dernière dynamique.

L'existence d'un seul outil ne peut probablement pas modifier les orientations curriculaires générales d'un programme. Néanmoins, force est de constater qu'une *tortue 3D dynamique* comme celle de DGPad modifie en profondeur le rapport à la pratique de la géométrie dans l'espace chez l'utilisateur, qu'il soit étudiant, enseignant ou élève, et ouvre des perspectives si innovantes, dans l'esprit des programmes actuels, qu'il serait intéressant, d'une façon ou d'une autre d'en tenir compte institutionnellement.

Compléments

Cet article n'a pas abordé la prise en main de l'interface, ces points sont détaillés dans les deux articles suivants :

- pour le Blockly de DGPad : http://revue.sesamath.net/spip.php?article863 ;
- spécifiquement pour la tortue: http://revue.sesamath.net/spip.php?article875 (très long article dont celui-ci est inspiré).

Ces deux articles détaillent la construction d'environ 160 figures, environ 40 en Blockly et environ 120 avec la tortue.

- Les spirolatères, avec une utilisation en classe de 4° de la tortue de DGPad

http://revue.sesamath.net/spip.php?article934

irem.univ-reunion.fr/IMG/pptx/spirolateres_irem_fev2017.pptx

La tortue de DGPad ayant été implémentée en mai-juin 2016, au moment de la rédaction de cet article il y avait encore peu de sites qui en traitait. Outre les précédents, citons l'Irem de Toulouse, dont la communication à l'APMEP en octobre 2016 :

http://www.ires-tlse-mathsetnumerique.fr/AtelierLyon/index.html
Et cet atelier à Limoges (2017)

http://www.ires-tlse-mathsetnumerique.fr/AtelierLimoges/index.html

L'auteur de DGPad, Eric Hakenholz, pour ses cours en collège, a une utilisation très différente de sa tortue, avec une utilisation systé-

2 « Petite Poucette » 2012 – Editions du Pommier - chapitre sur l'algorithmique ; pages 73 à 76.

matique du TeX. Il propose de superbes exercices sur les fractions et les nombres décimaux.

On consultera avec intérêt ses pages de cours qui contiennent des liens sur toutes ses figures :

* en 5° géométrie
  huit.re/CoursEric_Angles_TR ;
* en 5° sur les relatifs
  huit.re/CoursEric_Relatifs ;
* en 4° sur le triangle rectangle
  huit.re/CoursEric_pytha ;
* en 5° sur les fractions
  huit.re/CoursEric_Fractions

Deux iBooks dynamiques (figures manipulables dans le livre), en téléchargement gratuit pour faire connaître ces possibilités, sont disponibles sur AppStore pour Mac OS X ou iOS (iPad)

Cours L1 sur Volterra :

http://itunes.apple.com/fr/book/id1223516709

Ce cours (50 pages – 25 figures – 50 Mo) sera complété début 2018 d'un chapitre qui détaillera, pour les collègues, la façon dont ces figures sont réalisées.

Blockly Tortue et DGPad :

http://itunes.apple.com/fr/book/id1231386212

Livre de 186 pages dont 82 figures – 240 Mo – détaille et complète cet article.