Journées de l'APMEP Bordeaux 2018

Atelier : La géométrie dynamique et la tortue

Présentation des concepts

Historiquement, on a appelé « tortue » un robot piloté par un programme écrit en langage Logo. Dans le langage Logo, il y avait en particulier 2 instructions (emblématique de la tortue) :

- Avancer …
- TournerGauche …

On retrouve cette approche (francisée) dans le langage GéoTortue.

Le langage Python propose également une tortue (le module turtle) avec la même finalité que dans le langage Logo : être un support d'apprentissage du langage (dont la tortue est un ambassadeur).

Fondamentalement, la tortue est un paradigme de déplacement (on avance et/ou on pivote) qui constitue une alternative au paradigme de déplacement repéré (la téléportation) et il peut être intéressant d'associer les deux paradigmes.
La tortue peut être implémenté dans n'importe quel langage de programmation (dès lors que l'on dispose d'une interface de sortie graphique).
On peut aussi implémenter la tortue en géométrie dynamique, et implémenter une tortue dynamique.
Qu'est-ce que l'on entend par là ?

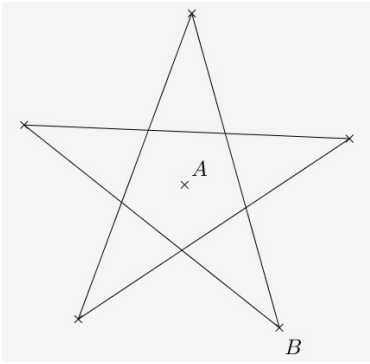
Activités

1) Le pentagramme

Pentagramme droit :
l'Homme positif

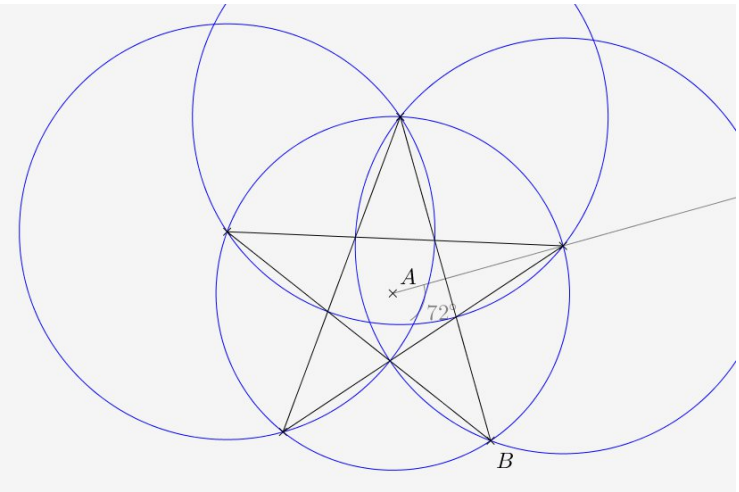
Pentagramme inversé :
l'Homme négatif

On veut construire ce pentagramme dynamique en A et B :



Méthodes :

- via l'interface :



traits de construction

- par script :

  Il serait compliqué de mimer la construction via l'interface car on serait confronté au problème d'intersections multiples.

  Méthode classique : faire de la géométrie repérée = chaque point du pentagramme sera construit à partir de ses coordonnées.

Avec CaRMetal :

```
sommets :=[];
sommets[0]:="B";
pour i allant de 1 à 4 {
   sommets[i]:=Point("x(A)+cos(_i*72)*(x(B)-x(A))-sin(_i*72)*(y(B)-y(A))",
                     "y(A)+cos(_i*72)*(y(B)-y(A))+sin(_i*72)*(x(B)-x(A))");
}
pour i allant de 0 à 4 {
   Segment(sommets[i],sommets[(i+2)%5]);
}
```

On obtient un pentagramme dynamique en A et B.
Avec une tortue dynamique, on obtiendra la même figure dynamique.

TP

Avec CaRMetal :

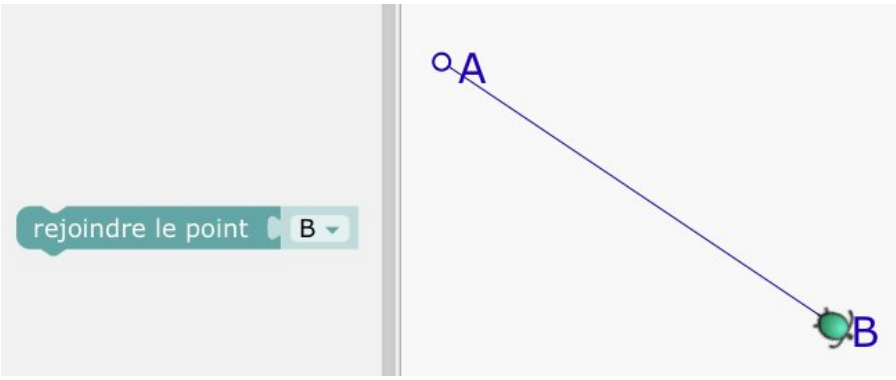
```
AttacherTortue("A");
Viser("B");
LeverStylo();
Avancer("d(A,B)");
BaisserStylo();
TournerGauche(180-18);

```

A compléter/tester

Avec DGPad :

Le script est une « propriété » (au sens informatique) du point A.
Le script crée un seul objet (une liste de points améliorée).
On a un script en Programmation Visuelle par Blocs, avec une surcouche Blockly.



A compléter/tester

2) Le patron dynamique du cube

Si on passe en 3D, on pourra facilement construire un patron de cube. L'élément dynamique important sera l'angle de la pliure, qui sera donné par un curseur.
En 3D, la tortue peut pivoter de trois façons différentes :
* TournerGauche (comme en 2D) en restant dans le même plan.
* PivoterHaut (avion qui décolle)
* PivoterGauche (avion qui vrille)

On peut commencer par construire le patron à plat.

TP : construction d'un (des 11) patron dynamique du cube.

Avec CaRMetal et/ou DGPad

facultatif : ajouter la sixième face.

3) Courbe donnée par son équation intrinsèque (rayon de courbure)

L'équation intrinsèque d'une courbe est de la forme R=f(s)

R est le rayon de courbure, s l'abscisse curviligne.

On a : R=ds/dΘ où Θ est l'angle du vecteur tangent avec la droite (Ox).
Cela permet de tracer la courbe à partir de son équation intrinsèque avec un élément dynamique ds (un curseur ds).

Le script a la forme générale suivante :

```
pour i allant de 1 à ... {
    Avancer(ds);
    TournerGauche(...);
}
```

TP : construction des « courbes à rayon sinusoïdal » définies par R=a(b+cos(s/a))