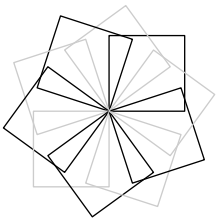
Scratch et METAPOST



C.Poulain

Janvier 2017

Résumé

Utiliser METAPOST pour produire des algorithmes « papier » avec les conventions de Scratch.

Table des matières

Avec les nouveaux programmes 2016 du Cycle 4 (Classes de 5e à 3e de collège) est apparu l'enseignement de l'algorithmique et l'utilisation de Scratch. Développé par le laboratoire Média du MIT, il permet de mettre en œuvre des algorithmes sous forme *ludique*. Sans rentrer dans un débat « pour ou contre », son emploi doit donc être enseigné aux élèves aux travers de différentes activités : questions *flash* [1], questions de compréhension, modification, correction d'algorithmes… Il fallait donc trouver une solution me permettant de proposer des algorithmes Scratch dans mes devoirs.

La première solution envisagée a été, bien évidemment, la capture d'écran. Simple, facile, rapide… ses avantages sont nombreux. Cependant, la qualité d'impression est parfois « moyenne »… Soucieux de proposer quelque chose de plus *cohérent* avec le « monde » LaTeX, je me suis lancé dans la création de mp-scratch avec pour objectif principal de proposer une syntaxe et une présentation très proche de celles utilisées par Scratch.

1. Sans aucun lien avec le langage informatique. Il s'agit de questions rapides posées en début de séance.
2. Ce premier algorithme me permet de remercier Maximum Chupin et son package bclogo : le drapeau vert a été créé à partir des sources de son package et notamment la construction, en METAPOST, de ses drapeaux.

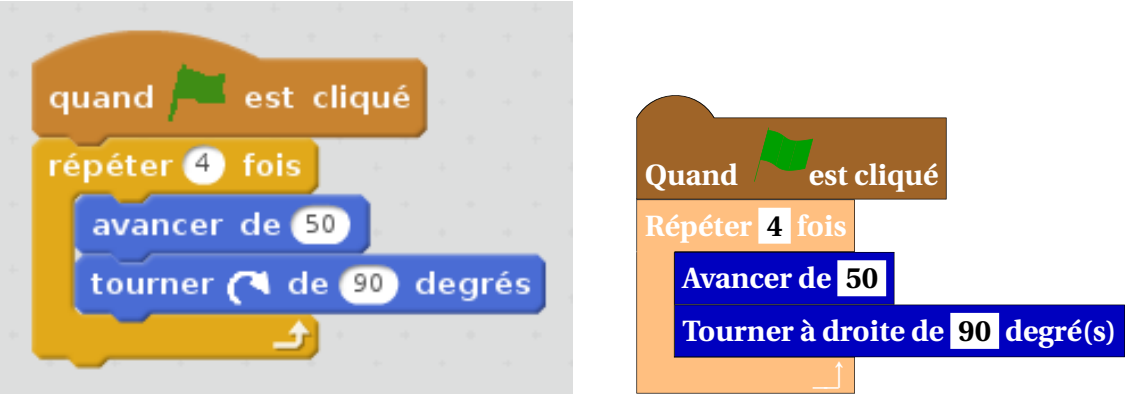


FIGURE 1 – Algorithme de création d'un carré - Versions Scratch et METAPOST [2]

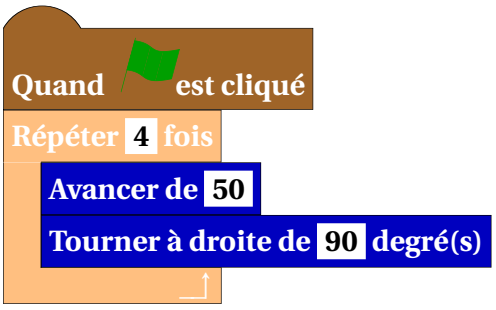
1 Installation

Le package mp-scratch est indépendant d'autres packages personnels. On trouvera l'archive à l'adresse

`melusine.eu.org/`

et l'ensemble des fichiers sera à placer correctement dans une arborescence TEX [3]

2 Utilisation




```
input mp-scratch;

beginfig(1);
  draw Drapeau;
  draw Repeter1(4);
  draw Avancer(50);
  draw Tournerd(90);
  draw FinBlocRepeter1(10);
endfig;
end
```

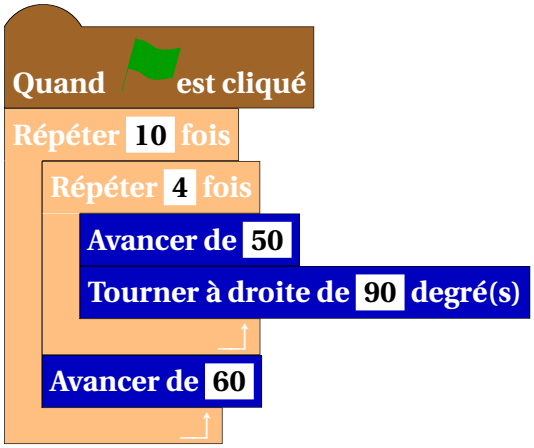
Reprenons l'exemple de l'algorithme du carré pour faire les premières constatations suivantes :

1.► la syntaxe est très proche du vocabulaire utilisé par Scratch ;

2.► les couleurs [4] sont proches de celles utilisées par Scratch ;

3.► le bloc Répéter est un peu particulier car il nécessite l'ajout *manuel* d'un nombre (ou chiffre) [5] qui permettra à METAPOST de faire la liaison correcte lors de la présence de multiples blocs Répéter comme le montre l'exemple ci-dessous.

3. Arborescence locale de préférence, par exemple dans /home/christophe/texmf/metapost/ sous linux.
4. Peut-être, sont-elles un peu ternes. Cela reste, bien évidemment, paramétrable.
5. Repeter2, Repeter5...




```
  input mp-scratch;

beginfig(1);
  draw Drapeau;
  draw Repeter2(10);
  draw Repeter1(4);
  draw Avancer(50);
  draw Tournerd(90);
  draw FinBlocRepeter1(10);
  draw Avancer(60);
  draw FinBlocRepeter2(10);
endfig;
end
```

4.► Très peu de nouvelles commandes à apprendre...

2.1 Commandes Scratch

Groupe Mouvement

- **draw** Avancer(10);

  Avancer de 10

- **draw** Tournerd(90);

  Tourner à droite de 15 degré(s)

- **draw** Tournerg(90)

  Tourner à gauche de 15 degré(s)

- **draw** Orienter(90);

  S'orienter à 90

- **draw** Orienterdirection("pointeur␣de␣souris");

  S'orienter vers pointeur de souris

- **draw** Aller(50,100);

  Aller à x : 50 y : 100

- **draw** Allera("pointeur␣de␣souris");

  Aller à : pointeur de souris

- **draw** Glisser(2,50,100);

  Glisser en 2 seconde(s) à x : 50 y : 100

- **draw** Ajouter(10,"x");

  Ajouter 10 à x

- **draw** Mettre(10,"x");

  Donner la valeur 10 à x

- **draw** Ajouter(50,"y");

  Ajouter 50 à y

- **draw** Mettre(10,"y");

- **draw** Rebondir;

- **draw** FixerSensRotation("position␣\‘a␣gauche␣ou␣\‘a␣droite"’;

Les « opérateurs [6] »

s’obtiennent avec la commande LaTeX [7] \opMouv{} comme le montre les exemples ci-dessous :

- **draw** Avancer("\opOp{$\opMouv{abscisse␣x}\bm{+}\opSimple{10}$}");

- **draw** Mettre("\opMouv{abscisse␣x}","y");

- **draw** Ajouter("\opMouv{direction}","y");

Groupe Apparence

- **draw** DireT("Hello",2);

- **draw** Dire("Hello");

- **draw** PenserT("Hmm...",2);

- **draw** Penser("Hmm...");

- **draw** Montrer;

- **draw** Cacher;

- **draw** Basculer("\opAp{costume2}");

- **draw** CostumeSuivant;

6. Je nomme « opérateurs » les variables ou commandes Scratch pouvant s’inclure dans les blocs-commande. Je me demande si je suis assez clair…

7. Oui, une commande LaTeX. Cela signifie donc que cette commande sera passée comme un élément de type string de METAPOST.

- **draw** BasculerAR("\opAp{arriere−plan2}");

- **draw** AjouterEffet("\opAp{couleur}",10);

- **draw** MettreEffet("\opAp{couleur}",10);

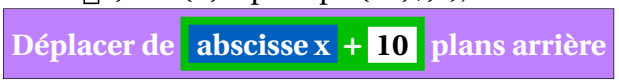
- **draw** AnnulerEffet;

- **draw** AjouterTaille(10);

- **draw** MettreA("\opOp{$\opSimple{10}\bm{+}\opSimple{5}$}");

- **draw** AllerPPlan;

- **draw** DeplacerAP("\opOp{$\opMouv{abscisse x}\bm{+}\opSimple{10}$}");



Les « opérateurs »

s'obtiennent avec la commande LaTeX \opAp{}.

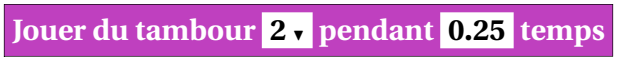
Groupe Son

- **draw** Jouer("miaou");

- **draw** JouerT("miaou");

- **draw** ArreterSon;

- **draw** Tambour(2,0.25);



- **draw** Pause(0.25);

- **draw** JouerNote(50,0.25);

- **draw** ChoisirInstrument(17);

- **draw** AjouterVol(−10);

Ajouter -10 au volume

- **draw** MettreVol(15);

Mettre le volume au niveau 15 %

- **draw** AjouterTempo(20);

Ajouter 20 au tempo

- **draw** MettreTempo(15);

Mettre le tempo à 15 bpm

Les « opérateurs » s'obtiennent par la commande LaTeX \opSon{}.

volume tempo

Groupe Stylo

- **draw** Effacer

Effacer tout

- **draw** Estampiller

Estampiller

- **draw** PoserStylo

Stylo en position d'écriture

- **draw** releverStylo

Relever le stylo

- **draw** MettreCouleur("Magenta",1,0,1);[8]

Mettre la couleur du stylo à

- **draw** AjouterCS("\opOp{$\opSimple{15}\bm{+}\opSimple{10}$}");

Ajouter 15 + 10 à la couleur du stylo

- **draw** MettreCS(25);

Mettre la couleur du stylo à 25

- **draw** AjouterIS("\opOp{$\opSimple{25}\bm{−}\opSimple{10}$}");

Ajouter 15 − 10 à l'intensité du stylo

- **draw** MettreIS(15);

Mettre l'intensité du stylo à 15

- **draw** AjouterTS(12);

Ajouter 12 à la taille du stylo

- **draw** MettreTS("\opOp{$\opSimple{15}\bm{\times}\opSimple{10}$}");

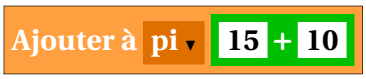
Mettre la taille du stylo à 15 × 10

8. La couleur est définie par son tripler rgb. Le nom de la couleur est libre mais ne doit pas rentrer en conflit avec les couleurs déjà définies dans le sous-package LATEXScratch.mp.

Groupe Données

- **draw** MettreVar("pi",0);

- **draw** AjouterVar("pi","\opOp{$\opSimple{15}\bm{+}\opSimple{10}$}");



- **draw** MontrerVar("pi");

- **draw** CacherVar("pi");

- **draw** AjouterList("\opSimple{\LaTeX}","Listepi");

- **draw** SupprimerList("\opSimple{\LaTeX}","Listepi");

- **draw** InsererList("\opSimple{\MP}",1,"Listepi");

- **draw** RemplacerList(3,"Listepi","\opOp{$\opSimple{4}\bm{+}\opSimple{5}$}");

- **draw** MontrerList("Listepi");

- **draw** CacherList("Listepi");

Les « opérateurs »

s'obtiennent par les commandes LaTeX \opVar{}, \opList{} et \opSousList.

Groupe Évènement

- **draw** Drapeau

- **draw** Presse("espace");

- **draw** LutinPresse;

- **draw** BasculeAR("arriere−plan1");

- **draw** VolumeSup("Volume␣sonore",10);

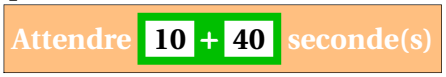
- **draw** RecevoirMessage("message1");

- **draw** EnvoyerMessage("message1");

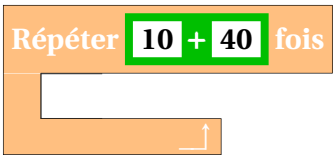
- **draw** EnvoyerMessageA("message1");

Groupe Contrôle

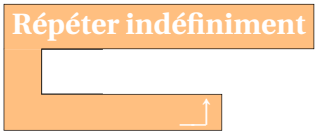
- **draw** Attendre("\opOp{$\opSimple{10}\bm{+}\opSimple{40}$}");



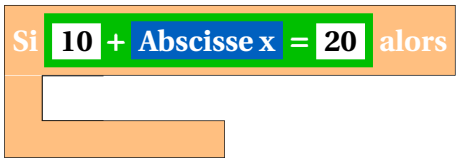

- ```
  draw Repeter1("\opOp{$\opSimple{10}\bm{+}\opSimple{40}$}");
  draw LigneVide(10);
  draw FinBlocRepeter1(10);
  ```



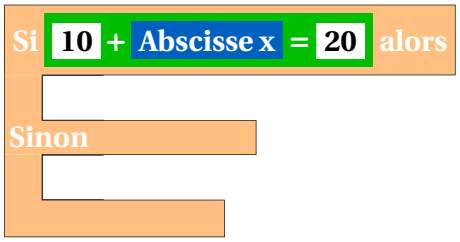

- ```
  draw RepeterI1;
  draw LigneVide(10);
  draw FinBlocRepeter1(10);
  ```



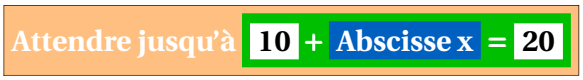

- ```
  draw Si1("\opOp{$\opSimple{10}\bm{+}\opMouv{Abscisse␣x}%
  \bm{=}\opSimple{20}$}");
  draw LigneVide(10);
  draw FinBlocSi1;
  ```
  ```



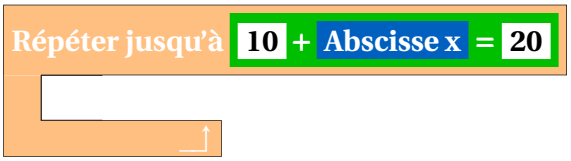

- ```
  draw Si1("\opOp{$\opSimple{10}\bm{+}\opMouv{Abscisse␣x}%
  \bm{=}\opSimple{20}$}");
  draw LigneVide(10);
  draw Sinon1;
  draw LigneVide(10);
  draw FinBlocSi1;
  ```



- **draw** AttendreJ("\opOp{$\opSimple{10}\bm{+}\opMouv{Abscisse␣x}\bm{=}\opSimple{20}$}");

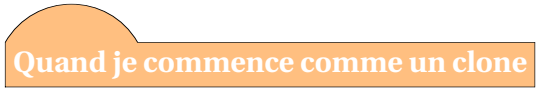



- ```
  draw RepeterJ1("\opOp{$\opSimple{10}\bm{+}\opMouv{Abscisse␣x}%
  ␣\bm{=}\opSimple{20}$}");
  draw LigneVide(10);
  draw FinBlocRepeter1(10);
  ```



- **draw Stop**("ce␣script");

- **draw** CommencerClone;



- **draw** CreerClone("Lutin1");

- **draw** SupprimerClone;

Groupe Capteurs

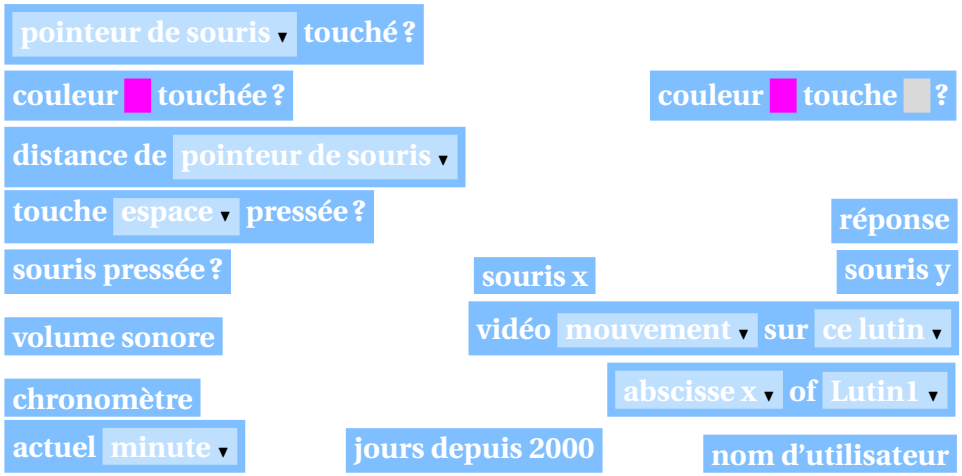
- **draw** Demander("Quel␣est␣votre␣prénom␣?");

- **draw** ActiverVideo("active");

- **draw** TransparenceVideo("\opOp{$\opSimple{17}\bm{+}\opSimple{25}$}");

- **draw** ReinitChrono;

Les « opérateurs » s'obtiennent par les commandes LaTeX \opCap{} et \opCapCap.



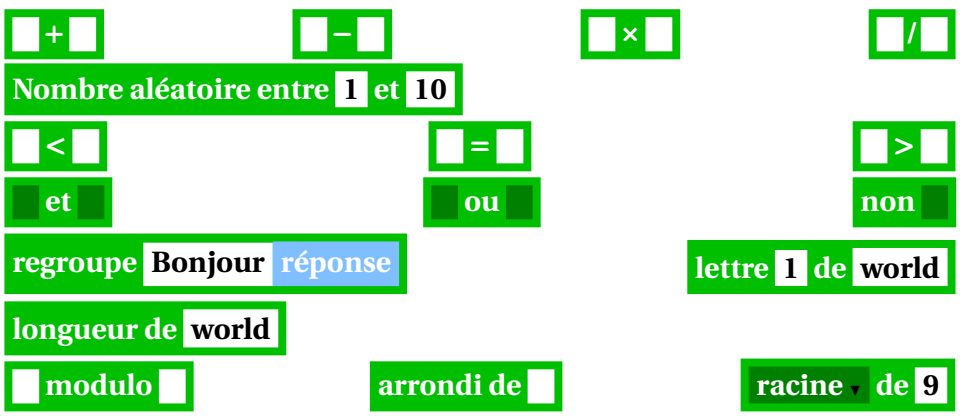
Néanmoins, il faut parfois un codage conséquent. Par exemple, voici un capteur et son code.


```
\opCap{couleur %
  \definecolor{Magenta}{rgb}{1,0,1}%
  \colorbox{Magenta}{\textcolor{white}{\phantom{t}}}%
  touche%
  \definecolor{LGray}{gray}{0.85}%
  \colorbox{LGray}{\textcolor{white}{\phantom{t}}} ?%
}
```

Un peu barbare, non ? Mais, cela ne nécessitera qu'un simple copier-coller pour les autres utilisations…

Groupe Opérateurs

Les éléments de ce groupe s'obtiennent par les commandes LaTeX \opOp{} et \opSousOp{}.



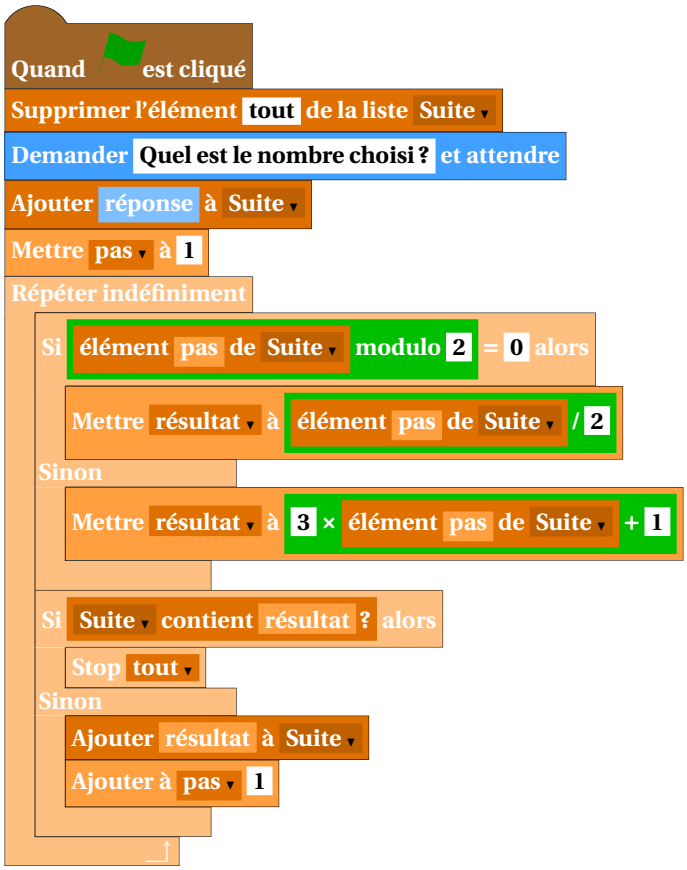
Groupe Ajouter blocs

- **draw** NouveauBloc("Pentagone");

- **draw** NouveauBloc("Pentagone␣\opBloc{cote}");

- **draw** Bloc("Pentagone");

Terminons cette liste de commandes par un algorithme associé à la suite de Syracuse :)



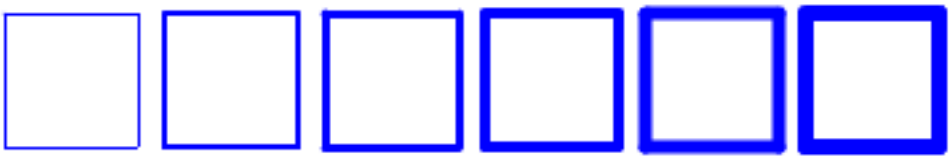
```
beginfig(1);
draw Drapeau;
draw SupprimerList("\opSimple{tout}","Suite");
draw Demander("\opSimple{Quel␣est␣le␣nombre␣choisi␣?}");
draw AjouterList("\opCap{réponse}","Suite");
draw MettreVar("pas",1);
draw RepeterI1;
draw Si2("\opOp{\opList{élément␣\opVar{pas}␣de␣\opSousList{Suite}}␣modulo␣\opSimple{2}%
\,$\bm{=}$\,\opSimple{0}}");
draw MettreVar("résultat","\opOp{\opList{élément␣\opVar{pas}␣de␣\opSousList{Suite}}%
\,$\bm{/}$\,\opSimple{2}}");
draw Sinon2;
draw MettreVar("résultat","\opOp{\opSimple{3}\,$\bm{\times}$\,\opList{élément␣\opVar{pas}%
␣de␣\opSousList{Suite}}\,$\bm{+}$\,\opSimple{1}}");
draw FinBlocSi2;
draw Si3("\opList{\opSousList{Suite}␣contient␣\opVar{résultat}␣?}");
```

```
  draw Stop("tout");
  draw Sinon3;
  draw AjouterList("\opVar{résultat}","Suite");
  draw AjouterVar("pas",1);
  draw FinBlocSi3;
  draw FinBlocRepeter1(10);
endfig;

end
```

3 Quelques exemples

3.1 Sujet de Brevet des collèges



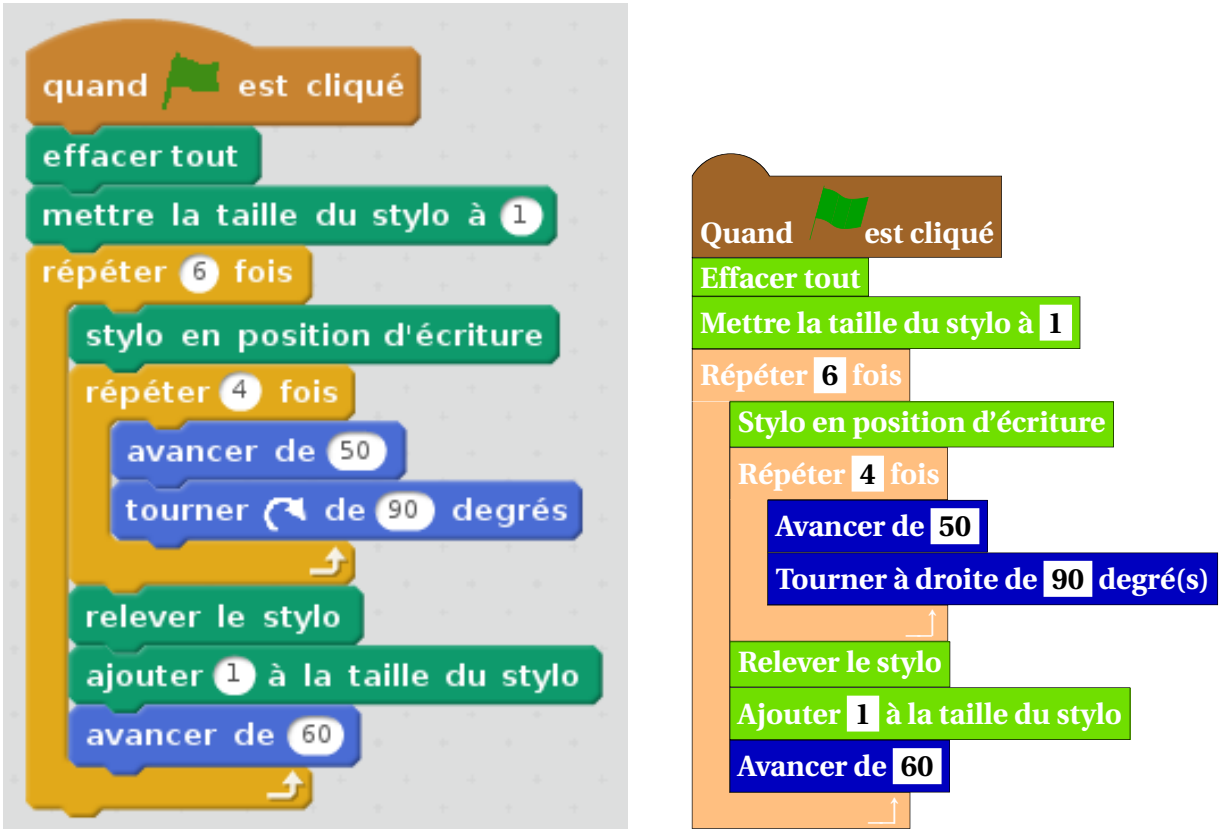


FIGURE 2 – Figure du Sujet 0 - versions Scratch et METAPOST

```
beginfig(1);
  draw Drapeau;
  draw Effacer;
  draw MettreTS(1);
  draw Repeter2(6);
  draw PoserStylo;
  draw Repeter1(4);
  draw Avancer(50);
  draw Tournerd(90);
  draw FinBlocRepeter1(10);
  draw ReleverStylo;
  draw AjouterTS(1);
  draw Avancer(60);
  draw FinBlocRepeter2(10);
endfig;
```

3.2 Œuvre d'art

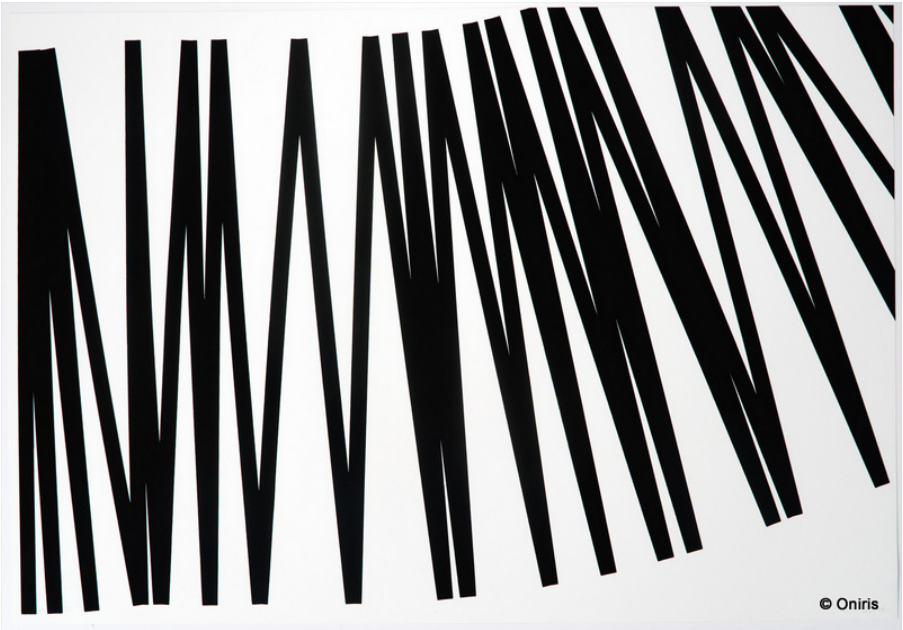
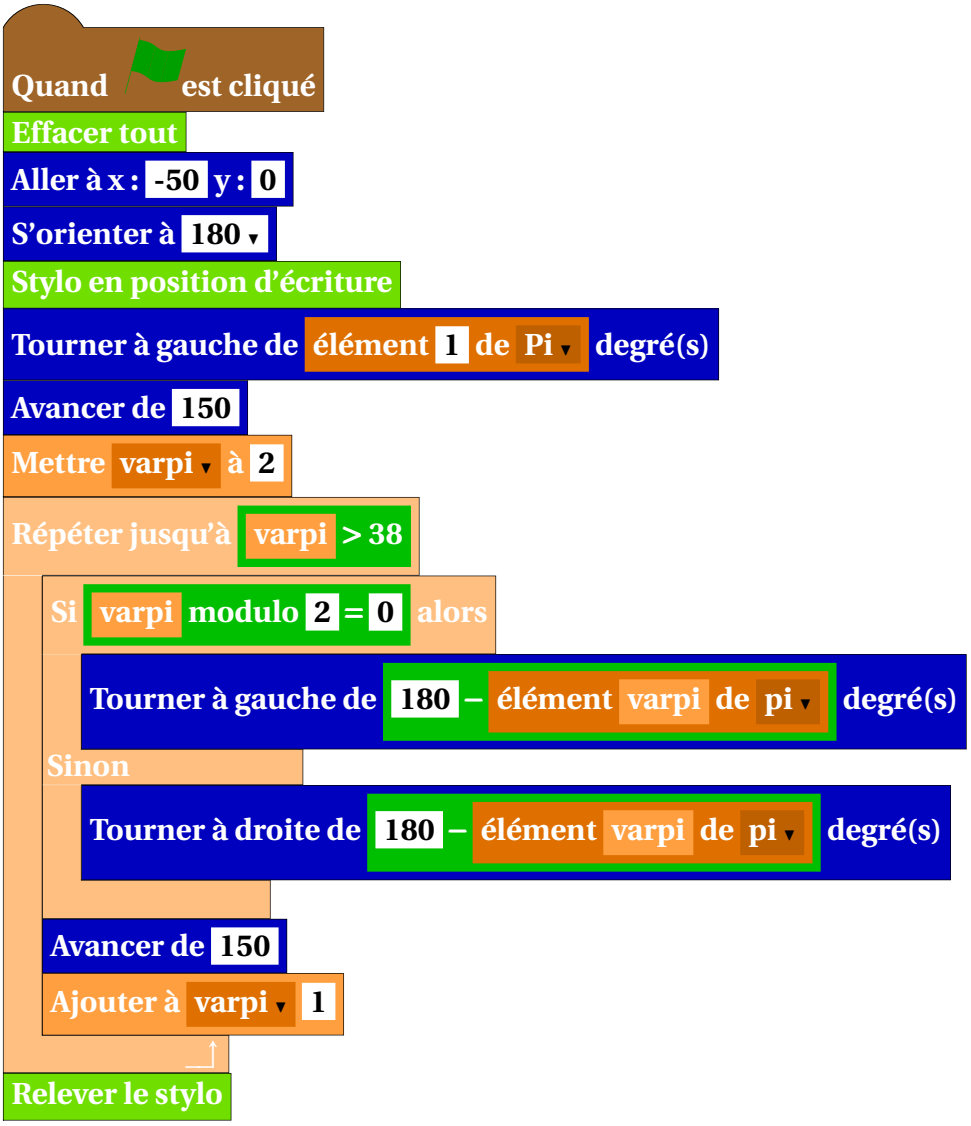
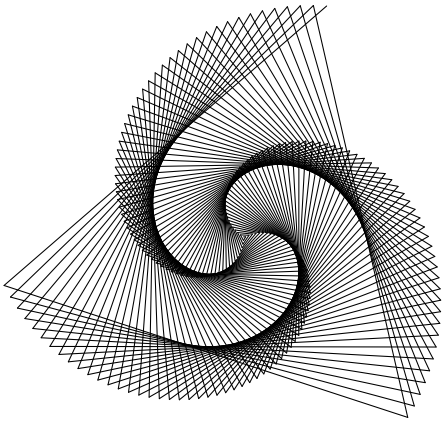


FIGURE 3 – François Morellet - Oeuvre Pi piquant, 1=1°, 38 décimales




```
beginfig(1)%François Morellet – Oeuvre Pi piquant, 1=1°, 38 décimales
  draw Drapeau;
  draw Effacer;
  draw Aller(-50,0);
  draw Orienter(180);
  draw PoserStylo;
  draw Tournerg("\opList{élément␣\opSimple{1}␣de␣\opSousList{Pi}}");
  draw Avancer(150);
  draw MettreVar("varpi","\opSimple{2}");
  draw RepeterJ1("\opOp{\opVar{varpi}\,$\bm{>}$\,38}");
  draw Si2("$\opOp{\opVar{varpi}\mbox{␣modulo␣}\opSimple{2}\,=\,\opSimple{0}}$");
  draw Tournerg("$\opOp{\opSimple{180}}\bm{-}\opList{élément␣\opVar{varpi}␣de␣\opSousList{pi}}$");
  draw Sinon2;
  draw Tournerd("$\opOp{\opSimple{180}}-\opList{élément␣\opVar{varpi}␣de␣\opSousList{pi}}$");
  draw FinBlocSi2;
  draw Avancer(150);
  draw AjouterVar("varpi","\opSimple{1}");
  draw FinBlocRepeter1(10);
  draw ReleverStylo;
endfig;
```

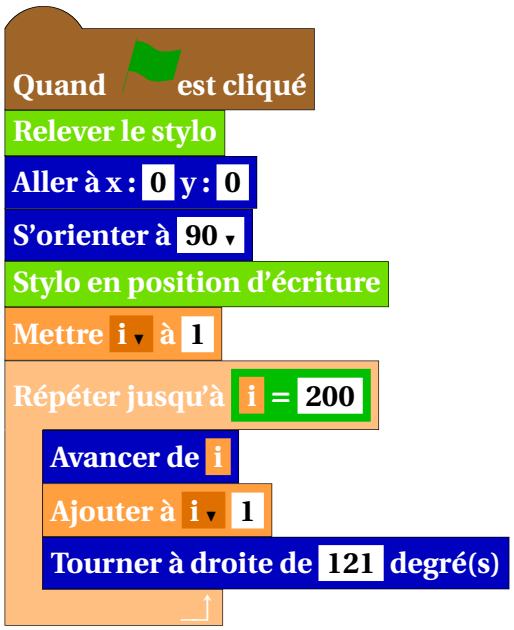
3.3 Une « spirale »



Source : http://www.ac-grenoble.fr/tice74/spip.php?article1219

```
beginfig(1);
  draw Drapeau;
  draw releverStylo;
  draw Aller(0,0);
  draw Orienter(90);
  draw PoserStylo;
  draw MettreVar("i",1);
  draw RepeterJ1("\opOp{$\opVar{i}\bm{=}%
␣␣␣␣\opSimple{200}$}");
  draw Avancer("\opVar{i}");
  draw AjouterVar("i",1);
  draw Tournerd(121);
  draw FinBlocRepeter1(10);
endfig;
```

FIGURE 4 – Figure géométrique - Code Scratch



3.4 Triangle de Sierpinski

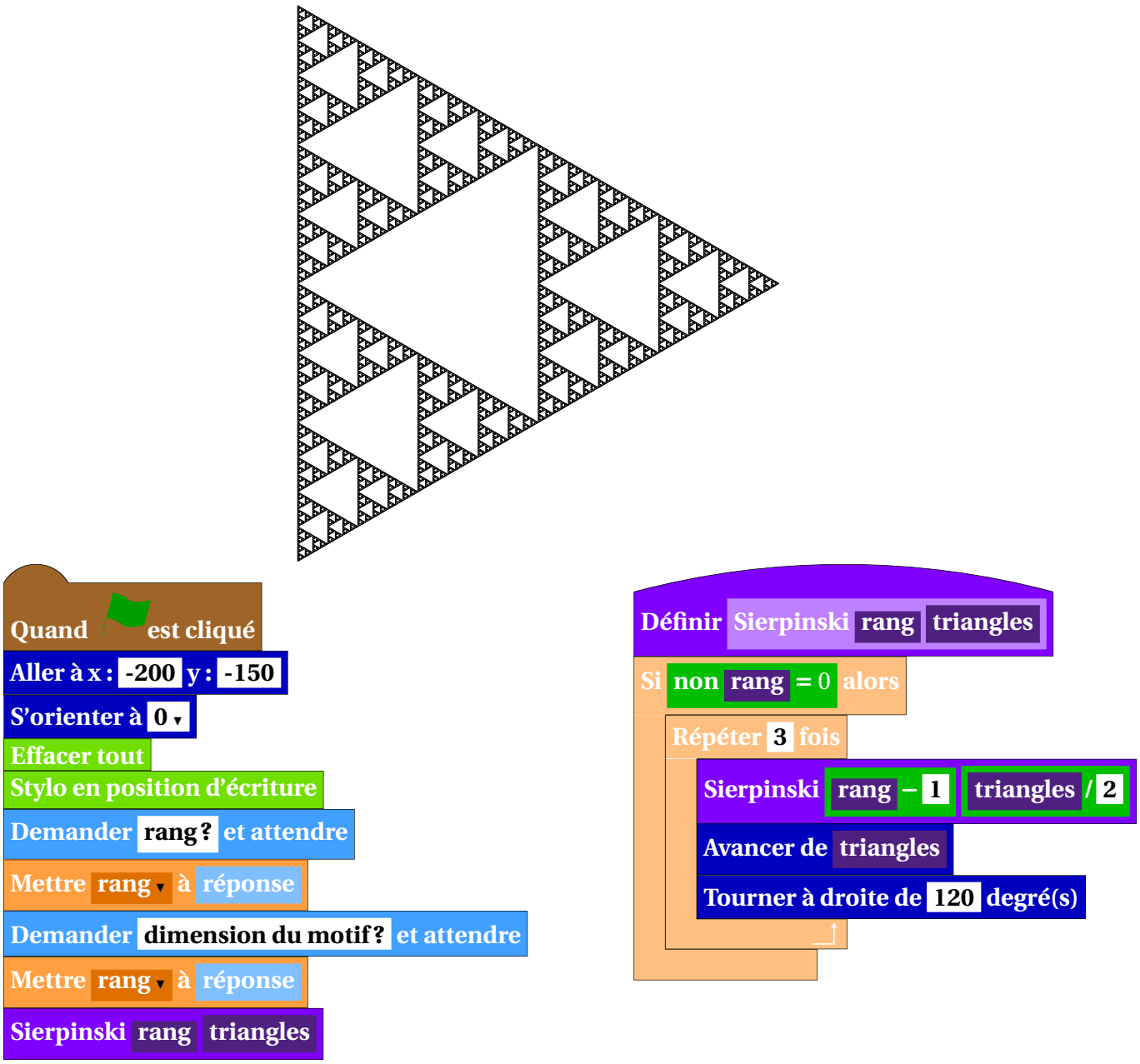


FIGURE 5 – Triangle de Sierpinski - Code Scratch et METAPOST

```
beginfig(1);%https://sites.google.com/site/stjomaths/scratch
  draw Drapeau;
  draw Aller(-200,-150);
  draw Orienter(0);
  draw Effacer;
  draw PoserStylo;
  draw Demander("\opSimple{rang␣?}");
  draw MettreVar("rang","\opCap{réponse}");
  draw Demander("\opSimple{dimension␣du␣motif␣?}");
  draw MettreVar("rang","\opCap{réponse}");
  draw Bloc("Sierpinski␣\opBloc{rang}␣\opBloc{triangles}");
  _coinprec:=(10cm,0);
```

```
  draw NouveauBloc("Sierpinski␣\opBloc{rang}␣\opBloc{triangles}");
  draw Si1("\opOp{non␣$\opBloc{rang}\bm{=}0$}");
  draw Repeter2(3);
  draw Bloc("Sierpinski␣\opOp{$\opBloc{rang}\bm{-}\opSimple{1}$}%
␣\opOp{$\opBloc{triangles}\,\bm{/}\,\opSimple{2}$}");
  draw Avancer("\opBloc{triangles}");
  draw Tournerd(120);
  draw FinBlocRepeter2(10);
  draw FinBlocSi1;
endfig;
```

4 Historique

21/01/2017 Version 0.5 - Publication sur `www.melusine.eu.org/syracuse/`

19/01/2017 Version 0.32 - Ajout d'éléments de présentation (▼).

18/01/2017 Version 0.31 - Ajout du groupe Son.

15/01/2017 Version 0.3 - Modification du code. Conception de la documentation.

08/01/2017 Version 0.2 - Ajout des commandes des groupes Données et Capteurs.

06/01/2017 Version 0.15 - Ajout des commandes du groupe Ajouter blocs.

05/01/2017 Version 0.1 - Sont disponibles les commandes des groupes Mouvement, Apparence, Stylo, Evènements, Contrôle.