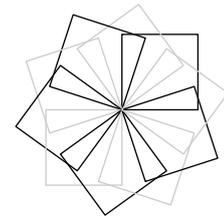




Scratch et METAPOST



mp-scratch

Version 0.9

C.Poulain
chrpoulain@gmail.com *

Février 2020

Résumé

... ou comment utiliser METAPOST pour produire des algorithmes « papier » avec les conventions de Scratch.

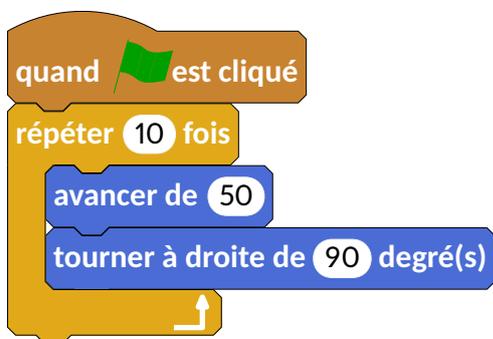
*Pour toute question, remarque, demande... n'hésitez pas à me contacter.

Avant propos

Avec les nouveaux programmes 2016 du Cycle 4 (Classes de 5^e à 3^e 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 présenté 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.

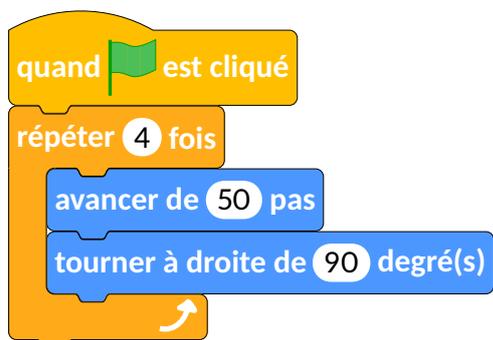
La première solution envisagée a été, bien évidemment, la capture d'écran. Simple, facile, rapide... ses avantages sont nombreux. Cependant, la qualité d'impression est parfois plus que « moyenne »...

Soucieux de proposer quelque chose de plus *cohérent* avec le « monde » L^AT_EX, 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 Scratch².



```
input mp-scratch;  
  
beginfig(1);  
  draw Drapeau;  
  draw Repeter("4");  
  draw Avancer("50");  
  draw Tournerd("90");  
  draw FinBlocRepeter;  
endfig;  
  
end
```

Mais depuis les premiers mois de 2019, la nouvelle version de Scratch est apparue... Comment faire la mise à jour de mp-scratch? Je me suis donc lancé dans quelques lignes de programmation supplémentaires...



```
input mp-scratch;  
  
Scratchversion:=3;  
  
beginfig(1);  
  draw Drapeau;  
  draw Repeter("4");  
  draw Avancer("50");  
  draw Tournerd("90");  
  draw FinBlocRepeter;  
endfig;  
  
end
```

1. Sans aucun lien avec le langage informatique. Il s'agit de questions rapides posées en début de séance.

2. Cet exemple me permet de remercier Maxime CHUPIN à double titre : pour m'avoir fait découvrir gcolor2, un petit utilitaire permettant de récupérer le code RGB de différentes couleurs; et son package bcolor : le drapeau vert a été créé à partir des sources de son package et notamment la construction, en METAPOST, de ses drapeaux.

Pour une utilisation plus pratique, `mp-scratch` est indépendant des autres packages personnels déjà produits tels `geometriesyr16`, `mp-geo` ou `mp-solid`. Au travers d'un dépôt `git`³, on trouvera l'archive à l'adresse

<http://melusine.eu.org/syracuse/G/mp-scratch/>

et l'ensemble des fichiers sera à placer correctement dans une arborescence `TEX`⁴. Et c'est tout!^{5 6}.

Utilisation

Afin de faciliter la « transcription » des algorithmes créés sous Scratch sous `METAPOST`, j'ai fait le choix de respecter *au maximum et au mieux* la syntaxe des briques Scratch. Cela ne facilitera pas l'internationalisation du package – Soyons fous! – mais permet de « produire » très rapidement les figures correspondantes aux différents algorithmes.

De plus, toujours dans un souci de « facilité⁷ », un nouveau paramètre a fait son apparition avec la version 0.8 : `Scratchversion`. En le positionnant à 3, tous les blocs qui suivent la déclaration de ce paramètre⁸ sont adaptés à la version 3 de Scratch.



Pour les codes `METAPOST` suivants, on omettra volontairement : `input mp-scratch` et `end`.

Même si on peut lancer un algorithme en double-cliquant sur le premier bloc, il reste néanmoins pratique de placer un élément de « début » d'algorithme :

```
beginfig(1);
  %Scratchversion:=3;
  draw Drapeau;
endfig;
```



```
beginfig(1);
  Scratchversion:=3;
  draw Drapeau;
endfig;
```



```
beginfig(2);
  %Scratchversion:=3;
  draw QPresse("espace");
endfig;
```



```
beginfig(2);
  Scratchversion:=3;
  draw QPresse("espace");
endfig;
```



3. Tous les contributeurs sont donc les bienvenus pour développer le package.

4. Arborescence locale de préférence et fortement conseillée! Par exemple, sous Linux, dans

```
home ▶ <utilisateur> ▶ texmf ▶ metapost
```

5. C'est une grande amélioration par rapport aux versions antérieures à la version 0.7 de `mp-scratch` où il était nécessaire de modifier les fichiers sources fournis...

6. Pour le texte contenu dans les briques, `mp-scratch` utilise le package `carlito` disponible (et probablement déjà installé) dans les distributions `TEX Live` et `MiKTEX`. Ce choix reste personnalisable évidemment (en modifiant le fichier `LATEXScratch.mp` du package `mp-scratch`) mais Thierry PASQUIER, à juste titre, m'a préconisé d'utiliser une fonte sans serif.

7. Compatibilité ascendante, gain de temps...

8. Le choix de Scratch – version 2 – par défaut est un choix purement arbitraire...

```
beginfig(3);
  %Scratchversion:=3;
  draw QLutinPresse;
endfig;
```

quand ce lutin est cliqué

```
beginfig(3);
  Scratchversion:=3;
  draw QLutinPresse;
endfig;
```

quand ce sprite est cliqué

```
beginfig(4);
  %Scratchversion:=3;
  draw QScenePressee;
endfig;
```

quand la Scène est cliquée

```
beginfig(4);
  Scratchversion:=3;
  draw QScenePressee;
endfig;
```

quand la scène est cliquée

```
beginfig(5);
  %Scratchversion:=3;
  draw QBasculeAR("arrière-plan 1");
endfig;
```

quand l'arrière-plan bascule sur arrière-plan 1

```
beginfig(5);
  Scratchversion:=3;
  draw QBasculeAR("arrière-plan 1");
endfig;
```

quand l'arrière-plan bascule sur arrière-plan 1

```
beginfig(6);
  %Scratchversion:=3;
  draw QVolumeSup("volume sonore","10"
  );
endfig;
```

quand le volume sonore > 10

```
beginfig(6);
  Scratchversion:=3;
  draw QVolumeSup("volume sonore","10"
  );
endfig;
```

quand le volume sonore > 10

```
beginfig(7);
  %Scratchversion:=3;
  draw QRecevoirMessage("message1");
endfig;
```

quand je reçois message1

```
beginfig(7);
  Scratchversion:=3;
  draw QRecevoirMessage("message1");
endfig;
```

quand je reçois message1

Il ne reste donc que deux blocs disponibles dans la catégorie **Evènements** ou **Evènements** :

- * envoyer à tous message1
- * ou envoyer à tous message1 ;
- * et envoyer à tous message1 et attendre
- * ou envoyer à tous message1 et attendre .

```
beginfig(8);
  %Scratchversion:=3;
  draw EnvoyerMessage("message1");
endfig;
```



```
beginfig(9);
  %Scratchversion:=3;
  draw EnvoyerMessageA("message1");
endfig;
```



```
beginfig(8);
  Scratchversion:=3;
  draw EnvoyerMessage("message1");
endfig;
```

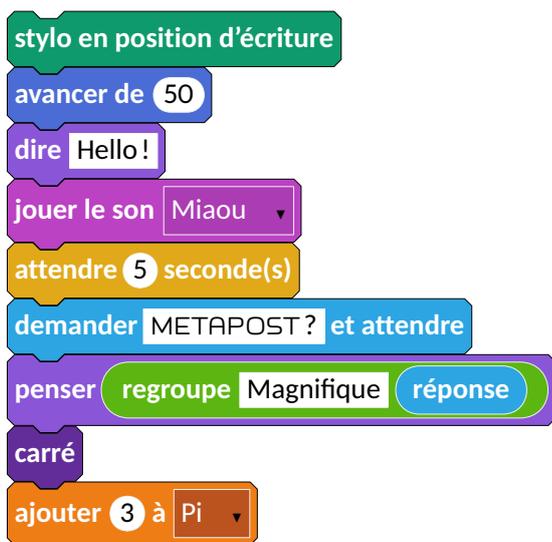


```
beginfig(9);
  Scratchversion:=3;
  draw EnvoyerMessageA("message1");
endfig;
```

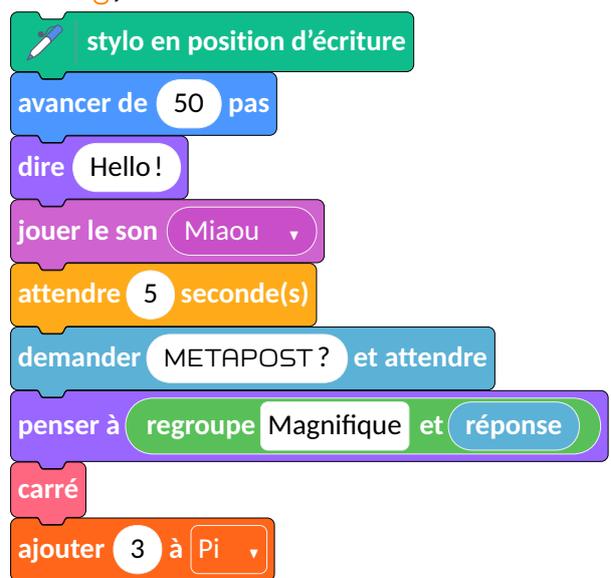


Une fois le script démarré, voici un exemple des blocs produits⁹. On remarque que la syntaxe est très proche du vocabulaire utilisé par Scratch (donc très peu de nouvelles commandes à apprendre) et que les couleurs¹⁰ sont celles utilisées par Scratch¹¹.

```
beginfig(1);
  draw PoserStylo;
  draw Avancer("50");
  draw Dire("Hello !");
  draw Jouer("Miaou");
  draw Attendre("5");
  draw Demander("\MP\ ?");
  draw Penser(OvalOp("regroupe",
    RecText("Magnifique"),OvalCap("
    réponse"))));
  draw Bloc("carré");
  draw AjouterListe("3","Pi");
endfig;
```



```
beginfig(1);
  Scratchversion:=3;
  draw PoserStylo;
  draw Avancer("50");
  draw Dire("Hello !");
  draw Jouer("Miaou");
  draw Attendre("5");
  draw Demander("\MP\ ?");
  draw Penser(OvalOp("regroupe",
    RecText("Magnifique"),OvalCap("
    réponse"))));
  draw Bloc("carré");
  draw AjouterListe("3","Pi");
endfig;
```



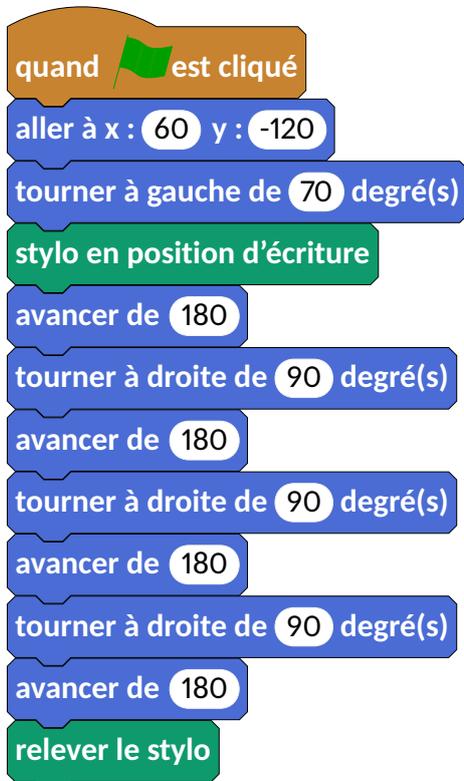
9. On remarque le stylo de Scratch... Obtenu grâce à l'extension Scratch3 de Christian TELLECHEA.

10. Cela reste, bien évidemment, paramétrable. Les paramètres disponibles pour personnaliser les couleurs sont colMouv, colAp, colSon, colStylo, colEvenements, colControle, colCapteur, colBloc, colVar, colList.

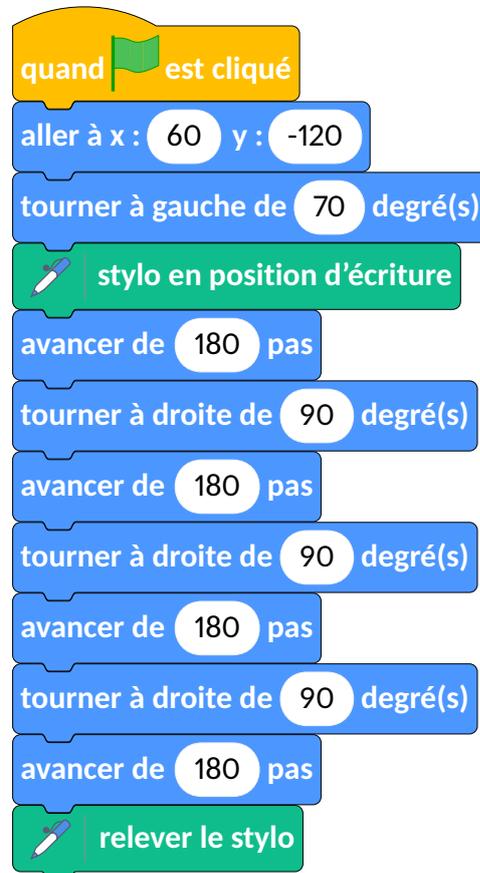
11. grâce à gcolor2. On peut utiliser également grabc en ligne de commande.

Premier pas

Scratch permet d'effectuer des constructions géométriques à l'aide, principalement, des catégories **Mouvement** ou **Mouvement** et **Stylo** ou **Stylo**. Par exemple, en proposant l'algorithme ci-dessous pour que le lutin effectue la figure 1 (page 7).



```
beginfig(1);
  %Scratchversion:=3;
  draw Drapeau;
  draw Aller("60", "-120");
  draw Tournerg("70");
  draw PoserStylo;
  draw Avancer("180");
  draw Tournerd("90");
  draw Avancer("180");
  draw Tournerd("90");
  draw Avancer("180");
  draw Tournerd("90");
  draw Avancer("180");
  draw ReleverStylo;
endfig;
```



```
beginfig(1);
  Scratchversion:=3;
  draw Drapeau;
  draw Aller("60", "-120");
  draw Tournerg("70");
  draw PoserStylo;
  draw Avancer("180");
  draw Tournerd("90");
  draw Avancer("180");
  draw Tournerd("90");
  draw Avancer("180");
  draw Tournerd("90");
  draw Avancer("180");
  draw ReleverStylo;
endfig;
```

L'écriture des blocs **tourner à droite de 15 degré(s)** et **tourner à gauche de 15 degré(s)** est un choix personnel et pédagogique. Néanmoins, si on souhaite retrouver les briques *originales*, on utilise le booléen **symbole** qui est positionné à **false** par défaut.

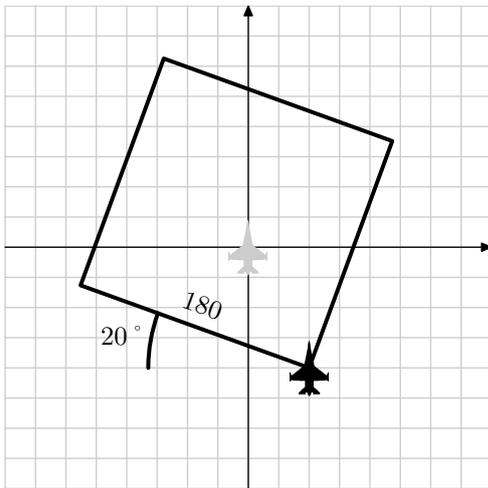


FIGURE 1 – Le lutin trace un carré

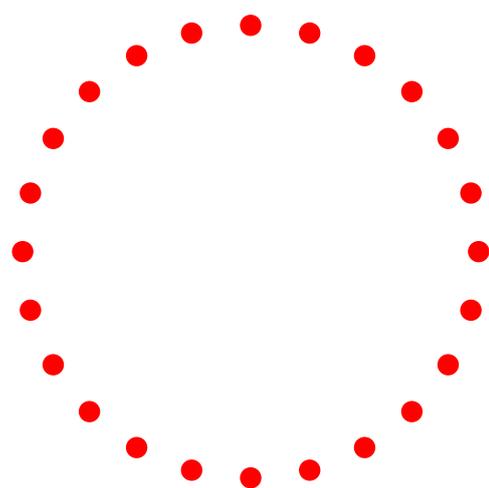


FIGURE 2 – Des points et un cercle

tourner à droite de 15 degré(s)

```
beginfig(1);
  %symbole:=true;
  draw Tournerd("15");
endfig;
```

tourner  de 15 degré(s)

```
beginfig(1);
  symbole:=true;
  draw Tournerd("15");
endfig;
```

tourner à gauche de 15 degré(s)

```
beginfig(1);
  %symbole:=true;
  draw Tournerg("15");
endfig;
```

tourner  de 15 degré(s)

```
beginfig(1);
  symbole:=true;
  draw Tournerg("15");
endfig;
```

Évidemment, on peut avoir besoin des catégories **Contrôle** ou **Contrôle** et **Ajouter blocs** ou **Mes blocs** pour obtenir des figures telles la figure 2 (page 7). Il faut pour cela définir le bloc

Point ou **Point**

définir **Point**

stylo en position d'écriture

relever le stylo

```
beginfig(1);
  %Scratchversion:=3;
  draw NouveauBloc("Point");
  draw PoserStylo;
  draw ReleverStylo;
endfig;
```

définir **Point**

 stylo en position d'écriture

 relever le stylo

```
beginfig(1);
  Scratchversion:=3;
  draw NouveauBloc("Point");
  draw PoserStylo;
  draw ReleverStylo;
endfig;
```

Une fois cette définition faite, on pourra écrire :
— dans une version Scratch 2 :

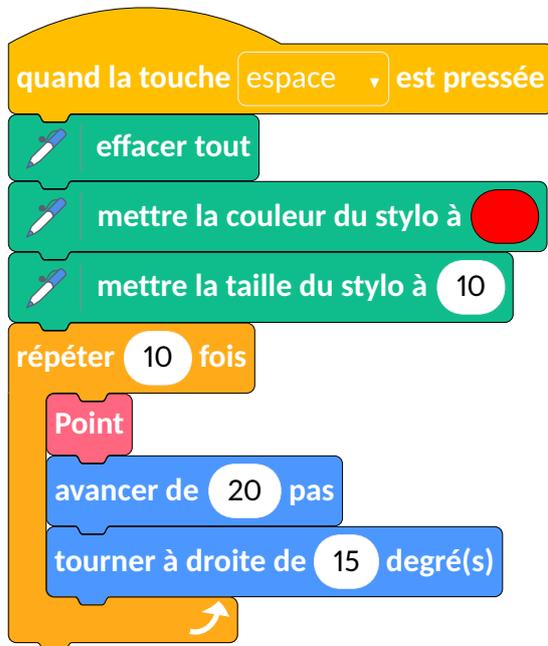


```

beginfig(1);
  %Scratchversion:=3;
  draw QPresse("espace");
  draw Effacer;
  draw MettreCouleur(1,0,0);
  draw MettreTS("10");
  draw Repeter("24");
  draw Bloc("Point");
  draw Avancer("20");
  draw Tournerd("15");
  draw FinBlocRepeteter;
endfig;

```

— dans une version Scratch 3 :



```

beginfig(1);
  Scratchversion:=3;
  draw QPresse("espace");
  draw Effacer;
  draw MettreCouleur(1,0,0);
  draw MettreTS("10");
  draw Repeter("24");
  draw Bloc("Point");
  draw Avancer("20");
  draw Tournerd("15");
  draw FinBlocRepeteter;
endfig;

```

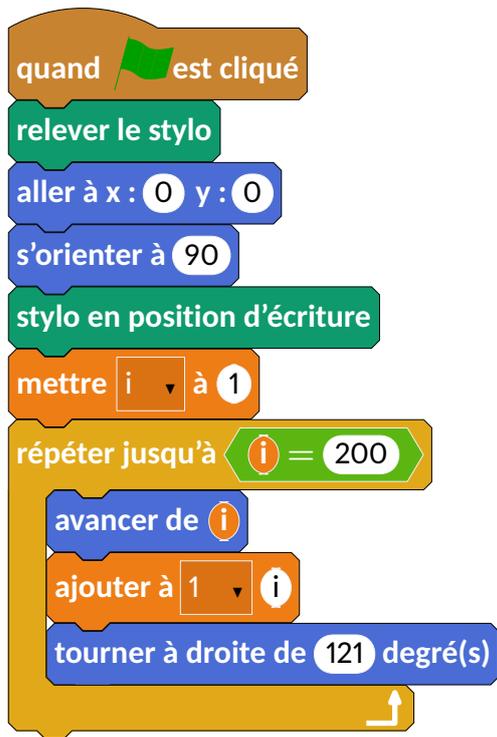
Dans des constructions un peu plus particulières, on peut utiliser des éléments des catégories

Données ou **Données** et **Opérateurs** ou **Opérateurs**.

Ainsi, pour obtenir la figure 3 (page 10)¹², on peut utiliser :

— en version Scratch 2 :

12. <http://www.ac-grenoble.fr/tice74/spip.php?article1219>

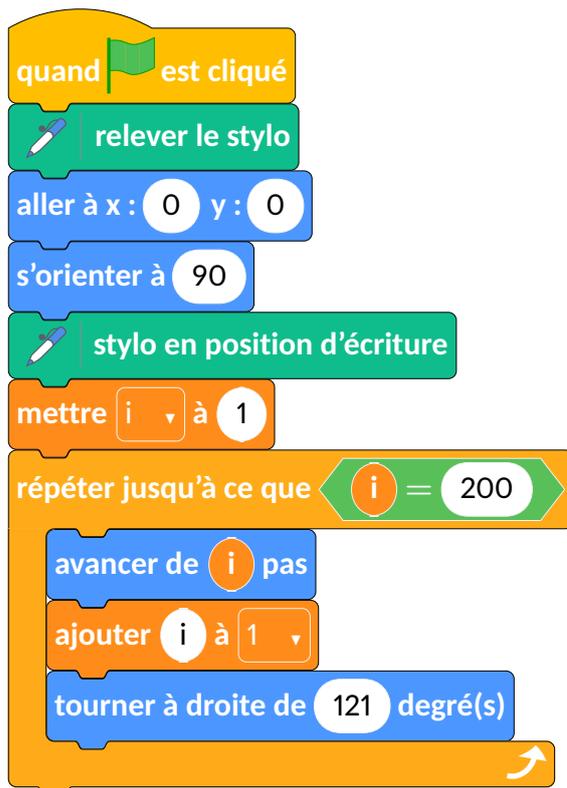


```

beginfig(1);
  %Scratchversion:=3;
  draw Drapeau;
  draw ReleverStylo;
  draw Aller("0","0");
  draw Orienter("90");
  draw PoserStylo;
  draw MettreVar("i","1");
  draw RepeterJ(TestOp(OvalVar("i")
    , "$\bm{=} $" ,OvalNb("200")));
  draw Avancer(OvalVar("i"));
  draw AjouterVar("i","1");
  draw Tournerd("121");
  draw FinBlocRepeter;
endfig;

```

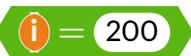
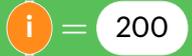
— en version Scratch 3 :



```

beginfig(1);
  Scratchversion:=3;
  draw Drapeau;
  draw ReleverStylo;
  draw Aller("0","0");
  draw Orienter("90");
  draw PoserStylo;
  draw MettreVar("i","1");
  draw RepeterJ(TestOp(OvalVar("i")
    , "$\bm{=} $" ,OvalNb("200")));
  draw Avancer(OvalVar("i"));
  draw AjouterVar("i","1");
  draw Tournerd("121");
  draw FinBlocRepeter;
endfig;

```

On remarque ici l'utilisation du test  ou  ainsi que sa création `TestOp(OvalVar("i"), "$\bm{=} $" ,OvalNb("200"))`.

Les éléments de la catégorie **Opérateurs** ou **Opérateurs** sont particuliers car aucun ne se présente sous la forme d'un bloc « puzzle »  ou  mais sous forme :

- d'ovale `regroupe Hello world` ou `regroupe Hello et world`
`OvalOp("regroupe",RecText("Hello"),RecText("world"))`
`OvalOp("regroupe",RecText("Hello")," et ",RecText("world"))` (Scratch 3)
- ou de « diamant » `i < 10` ou `i < 10`
`TestOp(OvalVar("i"),"$\bm{<}$",OvalNb("10"))`

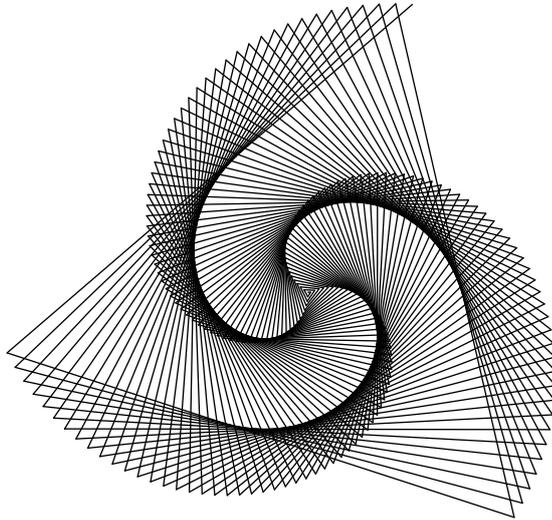


FIGURE 3 – Un segment tournant

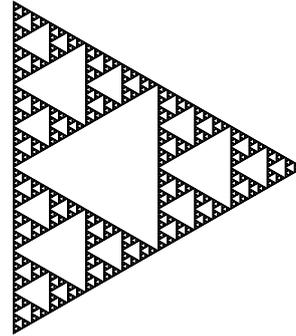
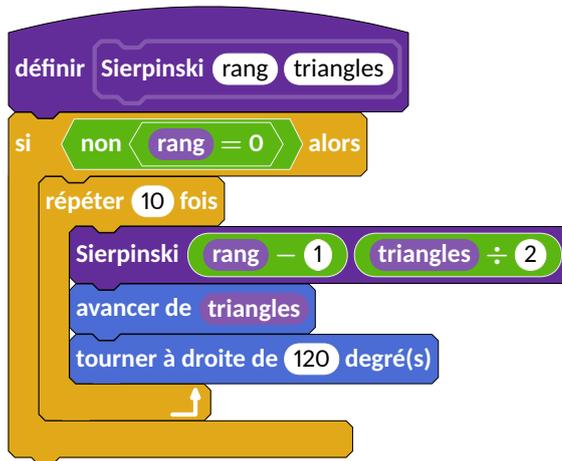


FIGURE 4 – Triangle de Sierpinski

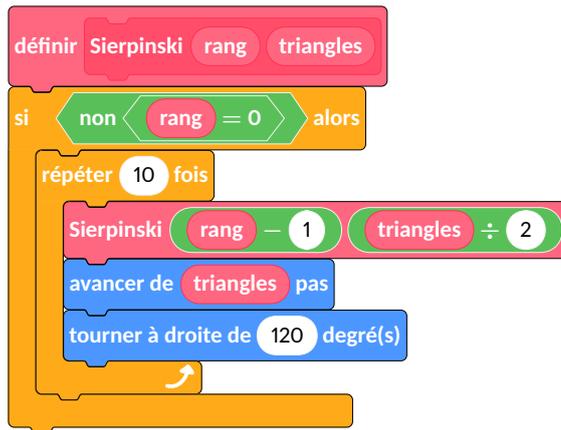
Si nous continuons d'explorer les constructions géométriques, on peut se pencher sur le triangle de Sierpinski (figure 4) qui nécessite la déclaration de deux « branches » d'algorithme :

- La première pour la définition de la récursivité du triangle :
- en version Scratch 2 :



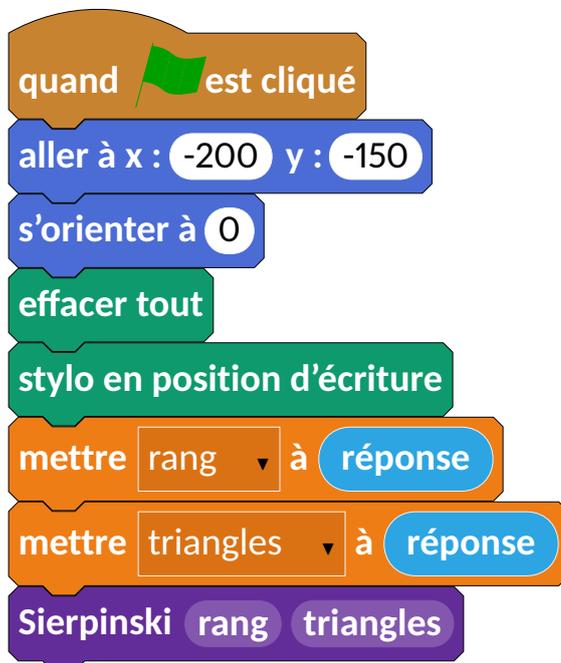
```
beginfig(1);
  %Scratchversion:=3;
  draw NouveauBloc("Sierpinski",
    OvalNb("rang"),OvalNb("
    triangles"));
  draw Si(TestOp("non",TestOp(
    OvalBloc("rang"),"$\bm{=} $" ,
    "0")));
  draw Repeter("3");
  draw Bloc("Sierpinski",OvalOp(
    OvalBloc("rang"),"$\bm{-} $" ,
    OvalNb("1")),OvalOp(OvalBloc
    ("triangles"),"$\bm{\div} $" ,
    OvalNb("2")));
  draw Avancer(OvalBloc("triangles
  "));
  draw Tournerd("120");
  draw FinBlocRepeter;
  draw FinBlocSi;
endfig;
```

— En version Scratch 3 :



```
beginfig(1);
Scratchversion:=3;
draw NouveauBloc("Sierpinski",
OvalNb("rang"),OvalNb("
triangles"));
draw Si(TestOp("non",TestOp(
OvalBloc("rang"),"$\bm{=} $" ,
"0")));
draw Repeter("3");
draw Bloc("Sierpinski",OvalOp(
OvalBloc("rang"),"$\bm{-} $" ,
OvalNb("1")),OvalOp(OvalBloc
("triangles"),"$\bm{\div} $" ,
OvalNb("2")));
draw Avancer(OvalBloc("triangles
"));
draw Tournerd("120");
draw FinBlocRepeter;
draw FinBlocSi;
endfig;
```

— et la deuxième pour le tracé en lui-même :
— en version Scratch 2 :



```
beginfig(2);
draw Drapeau;
draw Aller("-200","-150");
draw Orienter("0");
draw Effacer;
draw PoserStylo;
draw Demander("rang ?");
draw MettreVar("rang",OvalCap("
réponse"));
draw Demander("dimension du
motif ?");
draw MettreVar("triangles",
OvalCap("réponse"));
draw Bloc("Sierpinski",OvalBloc(
"rang"),OvalBloc("triangles"
));
endfig;
```

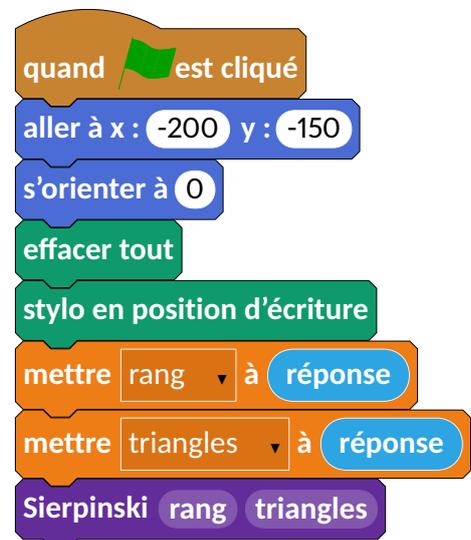
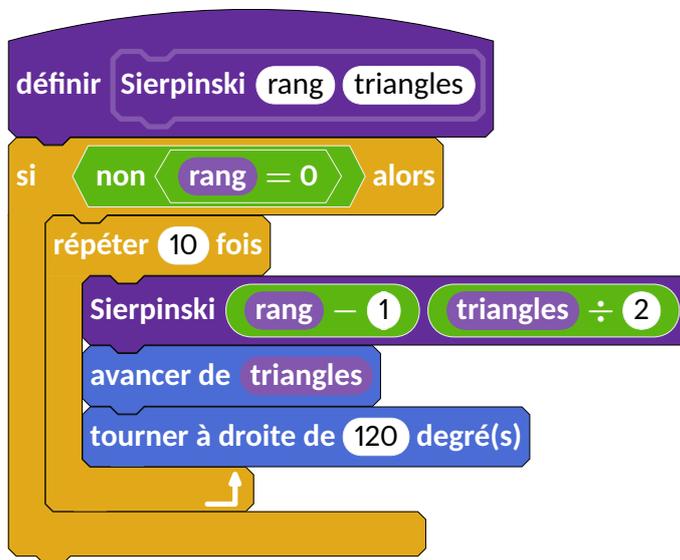
— en version Scratch 3 :



```
beginfig(2);
Scratchversion:=3;
draw Drapeau;
draw Aller("-200","-150");
draw Orienter("0");
draw Effacer;
draw PoserStylo;
draw Demander("rang ?");
draw MettreVar("rang",OvalCap("
réponse"));
draw Demander("dimension du
motif ?");
draw MettreVar("triangles",
OvalCap("réponse"));
draw Bloc("Sierpinski",OvalBloc(
"rang"),OvalBloc("triangles"
));
endfig;
```

Mais on peut également les placer côte à côte directement avec les commandes Deplacera et Deplacerde qui permettent de placer plusieurs algorithmes à l'intérieur d'une même image METAPOST :

— Dans sa version Scratch 2 :



```
beginfig(1);
draw NouveauBloc("Sierpinski",OvalNb("rang"),OvalNb("triangles"));
draw Si(TestOp("non",TestOp(OvalBloc("rang"),"$\bm{=} $" ,"0")));
draw Repeter("3");
draw Bloc("Sierpinski",OvalOp(OvalBloc("rang"),"$\bm{-} $" ,OvalNb("1"))
,OvalOp(OvalBloc("triangles"),"$\bm{\div} $" ,OvalNb("2")));
draw Avancer(OvalBloc("triangles"));
draw Tournerd("120");
draw FinBlocRepeter;
```

```

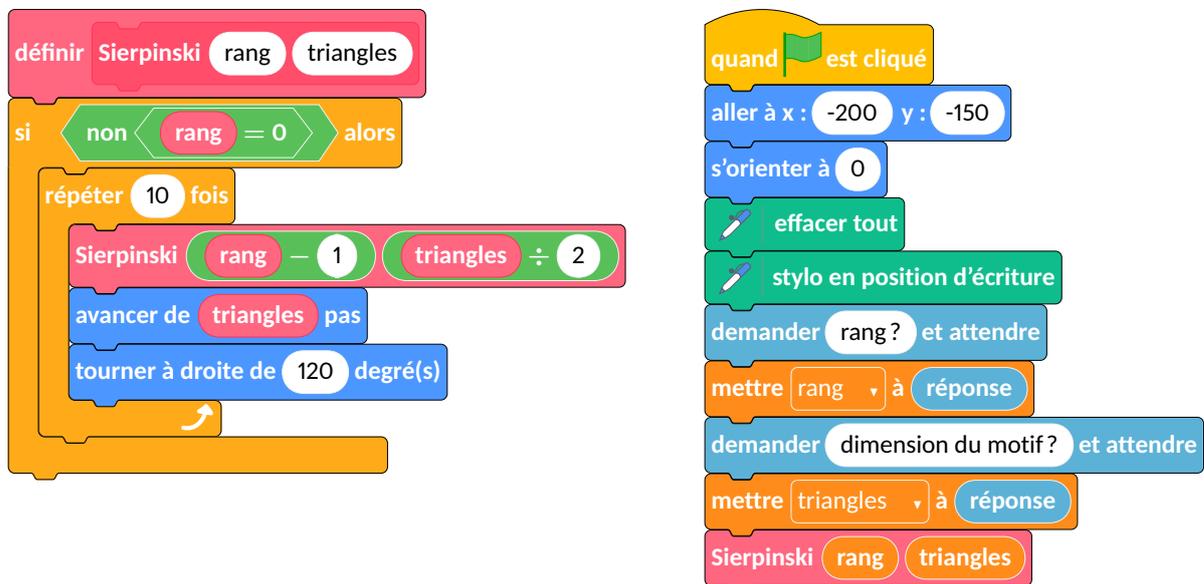
draw FinBlocSi;

Deplacera(10cm,0);

draw Drapeau;
draw Aller("-200","-150");
draw Orienter("0");
draw Effacer;
draw PoserStylo;
draw Demander("rang ?");
draw MettreVar("rang",OvalCap("réponse"));
draw Demander("dimension du motif ?");
draw MettreVar("triangles",OvalCap("réponse"));
draw Bloc("Sierpinski",OvalBloc("rang"),OvalBloc("triangles"));
endfig;

```

— Dans sa version Scratch 3 :



```

beginfig(1);
Scratchversion:=3;
draw NouveauBloc("Sierpinski",OvalNb("rang"),OvalNb("triangles"));
draw Si(TestOp("non",TestOp(OvalBloc("rang"),"$\bm{=} $" ,"0")));
draw Repeter("3");
draw Bloc("Sierpinski",OvalOp(OvalBloc("rang"),"$\bm{-} $" ,OvalNb("1"))
,OvalOp(OvalBloc("triangles"),"$\bm{\div} $" ,OvalNb("2")));
draw Avancer(OvalBloc("triangles"));
draw Tournerd("120");
draw FinBlocRepeter;
draw FinBlocSi;

Deplacera(11.5cm,0);

draw Drapeau;
draw Aller("-200","-150");
draw Orienter("0");
draw Effacer;

```

```

draw PoserStylo;
draw Demander("rang ?");
draw MettreVar("rang",OvalCap("réponse"));
draw Demander("dimension du motif ?");
draw MettreVar("triangles",OvalCap("réponse"));
draw Bloc("Sierpinski",OvalBloc("rang"),OvalBloc("triangles"));
endfig;

```

Intéressons nous maintenant à la possibilité qu'offre Scratch de travailler sur des listes, en construisant un tableau du peintre français MORELLET ou en étudiant la suite de Syracuse.

Tableau de MORELLET C'est un tableau tracé à partir des trente-huit (38) premiers chiffres constituant le nombre π :

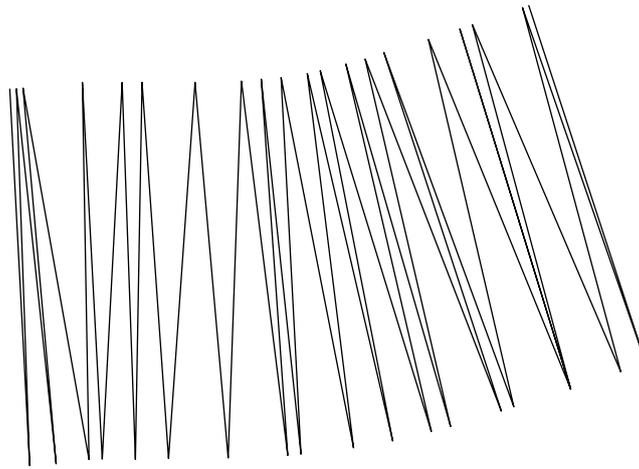


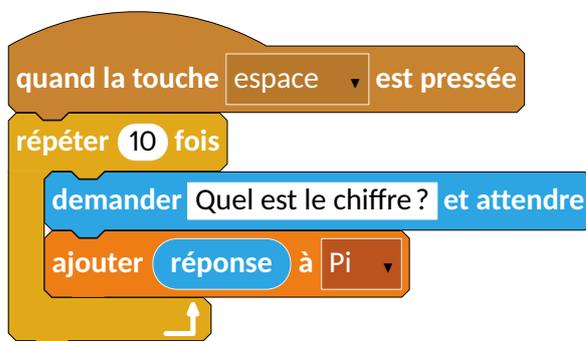
FIGURE 5 – Tableau de MORELLET

Il peut être nécessaire d'indiquer d'utiliser :

- pour créer la variable **varpi** ou **varpi** ;
- pour créer la liste **Pi** ou **Pi** .

Pour compléter la liste **Pi** ou **Pi** , on peut utiliser « le script » suivant :

- en version Scratch 2 :

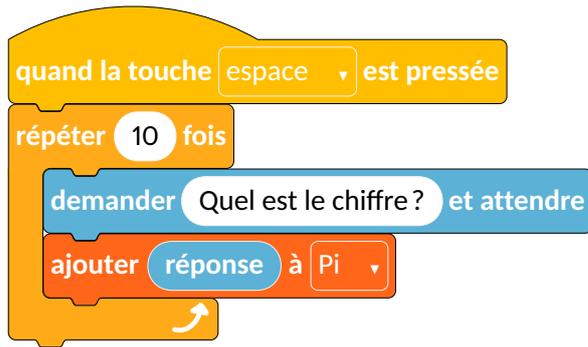


```

beginfig(1);
  %Scratchversion:=3;
  draw QPresse("espace");
  draw Repeter("38");
  draw Demander("Quel est le
    chiffre ?");
  draw AjouterListe(OvalCap("
    réponse"),"Pi");
  draw FinBlocRepeter;
endfig;

```

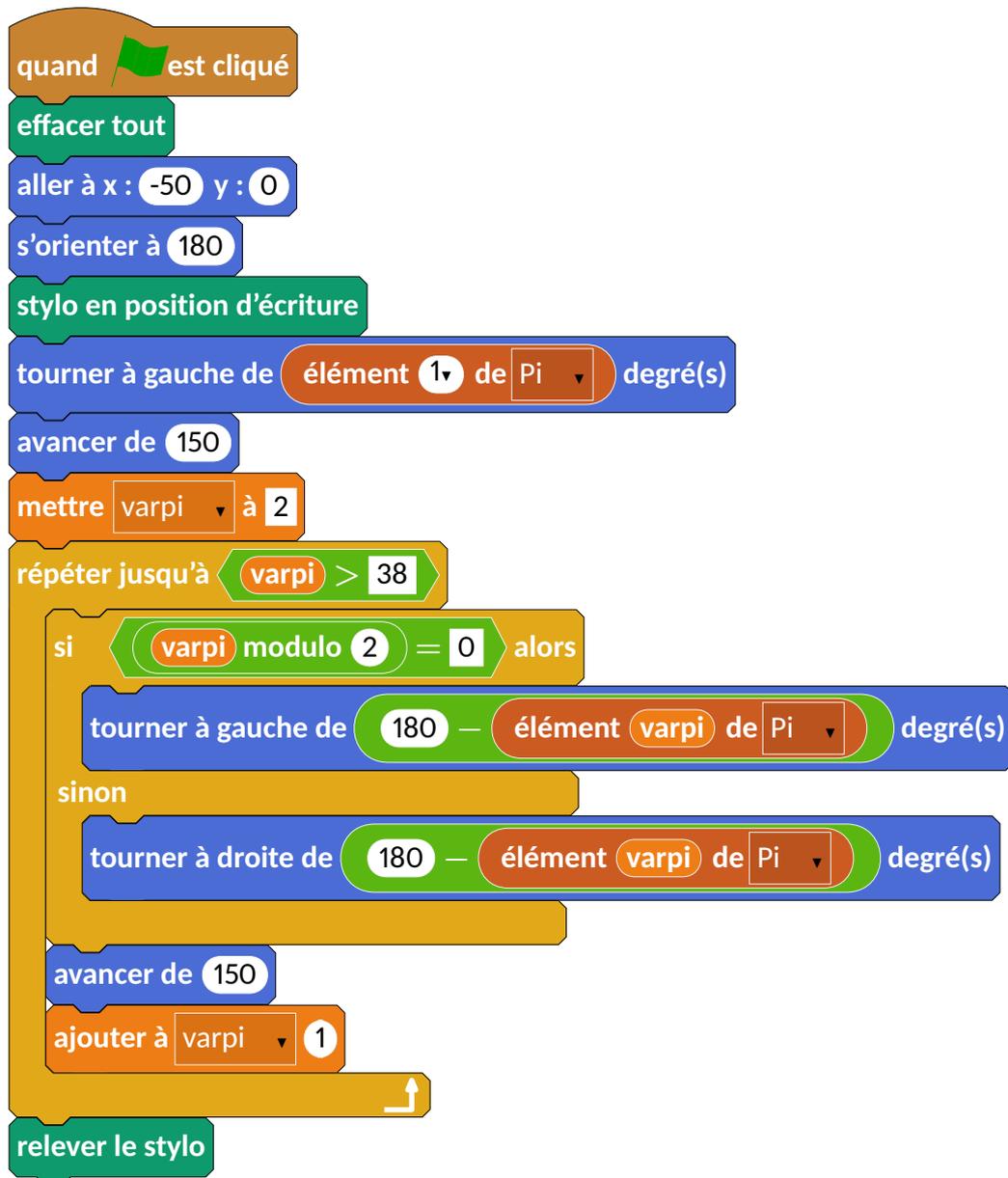
— en version Scratch 3 :



```
beginfig(1);  
Scratchversion:=3;  
draw QPresse("espace");  
draw Repeter("38");  
draw Demander("Quel est le  
chiffre ?");  
draw AjouterListe(OvalCap("réponse"), "Pi");  
draw FinBlocRepeter;  
endfig;
```

On peut ainsi passer à la création de l'algorithme permettant d'effectuer le tracé du tableau :

— Dans sa version Scratch 2 :



```
beginfig(1)
```

```
%François Morellet - Oeuvre Pi piquant, 1=1°, 38 premiers chiffres
```

```
%Scratchversion:=3;
```

```
draw Drapeau;
```

```
draw Effacer;
```

```
draw Aller("-50","0");
```

```
draw Orienter("180");
```

```
draw PoserStylo;
```

```
draw Tournerg(OvalList("élément",OvalMenuNb("1")," de ",RecMenuList("Pi")));
```

```
draw Avancer("150");
```

```
draw MettreVar("varpi",RecText("2"));
```

```
draw RepeterJ(TestOp(OvalVar("varpi"), "$\b{>}$",RecText("38")));
```

```
picture BB[];
```

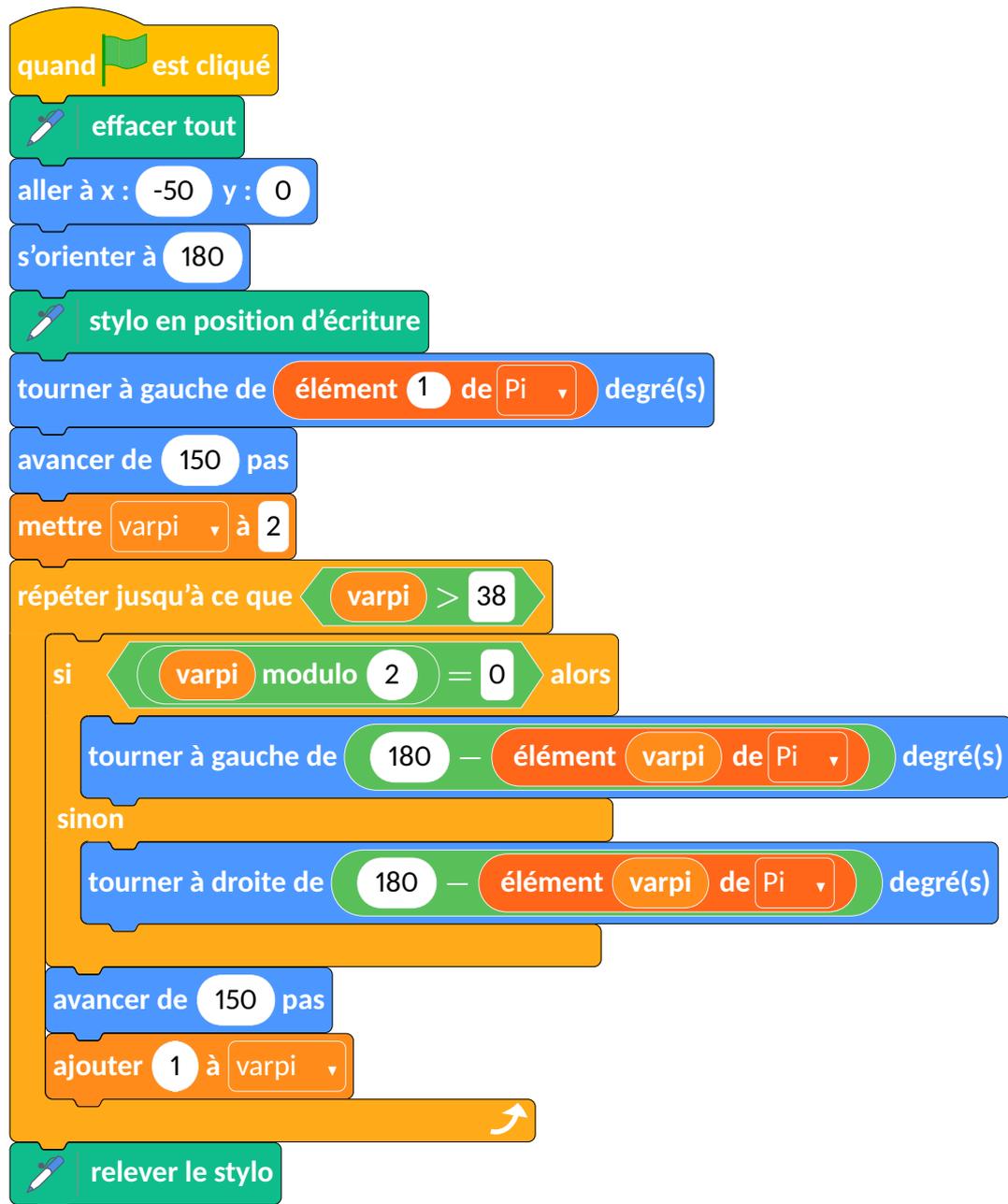
```
BB1=OvalOp(OvalVar("varpi"),"modulo",OvalNb("2"));
```

```

draw Si(TestOp(BB1,"$\bm{=} $" ,RecText("0")));
BB2=OvalList("élément",OvalVar("varpi")," de ",RecMenuList("Pi"));
draw Tournerg(OvalOp(OvalNb("180"),"$\bm{-} $" ,BB2));
draw Sinon;
draw Tournerd(OvalOp(OvalNb("180"),"$\bm{-} $" ,BB2));
draw FinBlocSi;
draw Avancer("150");
draw AjouterVar("varpi","1");
draw FinBlocRepeter;
draw ReleverStylo;
endfig;

```

— Dans sa version Scratch 3 :



```

beginfig(1)
  %François Morellet - Oeuvre Pi piquant, 1=1°, 38 premiers chiffres

  Scratchversion:=3;
  draw Drapeau;
  draw Effacer;
  draw Aller("-50","0");
  draw Orienter("180");
  draw PoserStylo;
  draw Tournerg(OvalList("élément",OvalMenuNb("1")," de ",RecMenuList("
    Pi")));
  draw Avancer("150");
  draw MettreVar("varpi",RecText("2"));
  draw RepeterJ(TestOp(OvalVar("varpi"),"$\bm{>}$",RecText("38")));
  picture BB[];
  BB1=OvalOp(OvalVar("varpi"),"modulo",OvalNb("2"));
  draw Si(TestOp(BB1,"$\bm{=}$",RecText("0")));
  BB2=OvalList("élément",OvalVar("varpi")," de ",RecMenuList("Pi"));
  draw Tournerg(OvalOp(OvalNb("180"),"$\bm{-}$",BB2));
  draw Sinon;
  draw Tournerd(OvalOp(OvalNb("180"),"$\bm{-}$",BB2));
  draw FinBlocSi;
  draw Avancer("150");
  draw AjouterVar("1","varpi");
  draw FinBlocRepeter;
  draw ReleverStylo;
endfig;

```



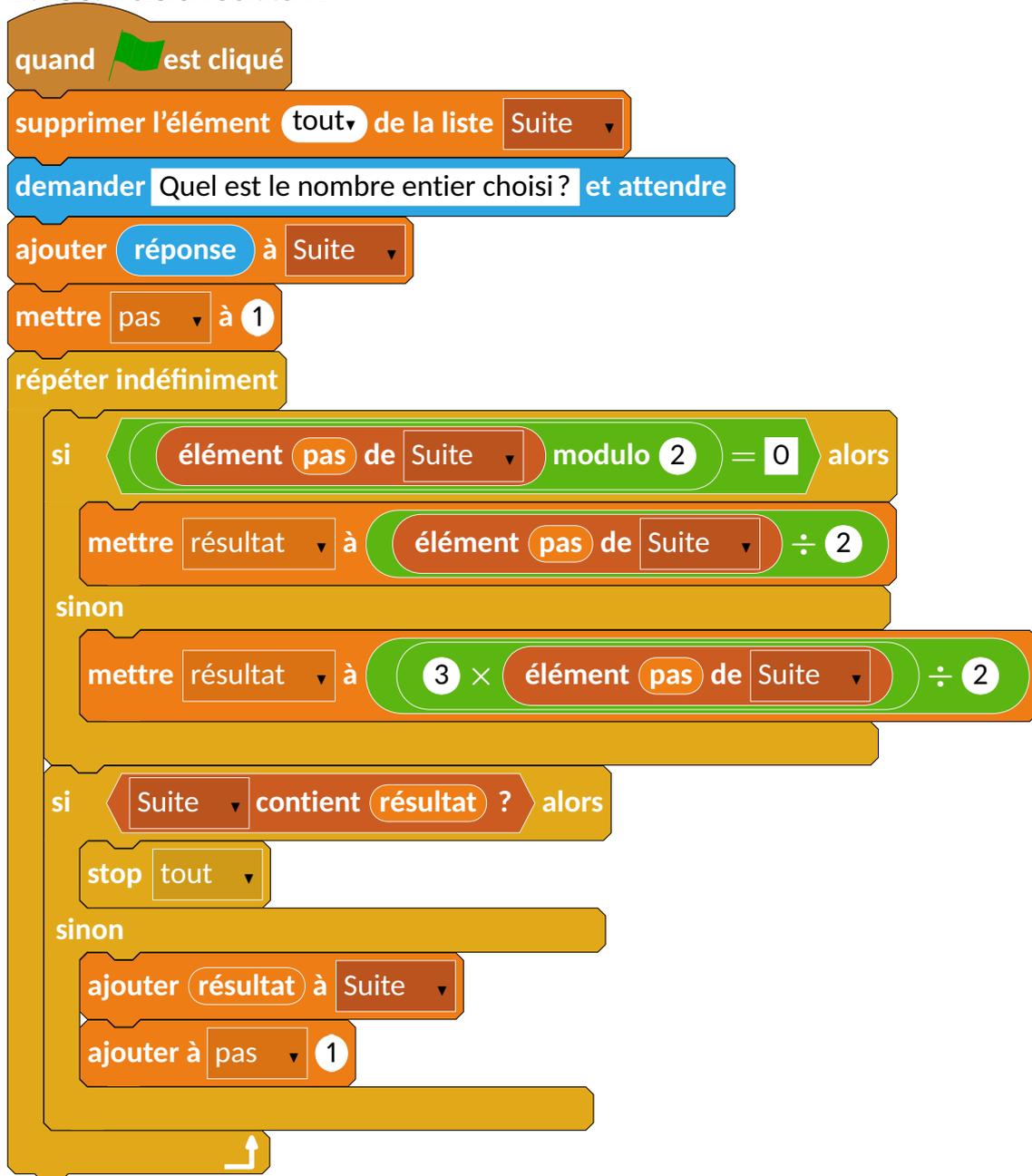
Le choix d'utiliser les `picture BB[]` a été fait pour gagner un peu de temps de compilation. Mais ce n'est pas nécessaire : on peut tout grouper !

Suite de Syracuse C'est une suite mathématique bien connue qui, partant d'un nombre entier, le divise par 2 s'il est pair, le multiplie par 3 et ajoute 1 s'il est impair ; le nombre ainsi obtenu remplaçant le nombre entier de départ...

Indépendamment de la conjecture associée à cette suite, nous pouvons étudier « la longueur » de la suite associée à un nombre entier donné. Voici un exemple de son implantation sous Scratch. Pour cela, on a créé :

- deux variables **résultat** ou **résultat** et **pas** ou **pas** ;
- une liste **Suite** ou **Suite** qui contiendra les éléments de la suite de Syracuse associée au nombre entier choisi.

— Dans sa version Scratch 2 :



```

beginfig(1);
  %Scratchversion:=3;
  draw Drapeau;
  draw SupprimerListe("tout","Suite");
  draw Demander("Quel est le nombre
    entier choisi ?");
  draw AjouterListe(OvalCap("réponse"),"
    Suite");
  draw MettreVar("pas","1");
  draw RepeterI;
  picture BB[];
  BB1=OvalList("élément",OvalVar("pas"),"
    de",RecMenuList("Suite"));
  BB2=OvalOp(BB1,"modulo",OvalNb("2"));
  draw Si(TestOp(BB2,"$\\bm{=} $" ,RecText("
    0")));
  draw MettreVar("résultat",OvalOp(BB1,"
    $\\bm{\\div} $" ,OvalNb("2")));

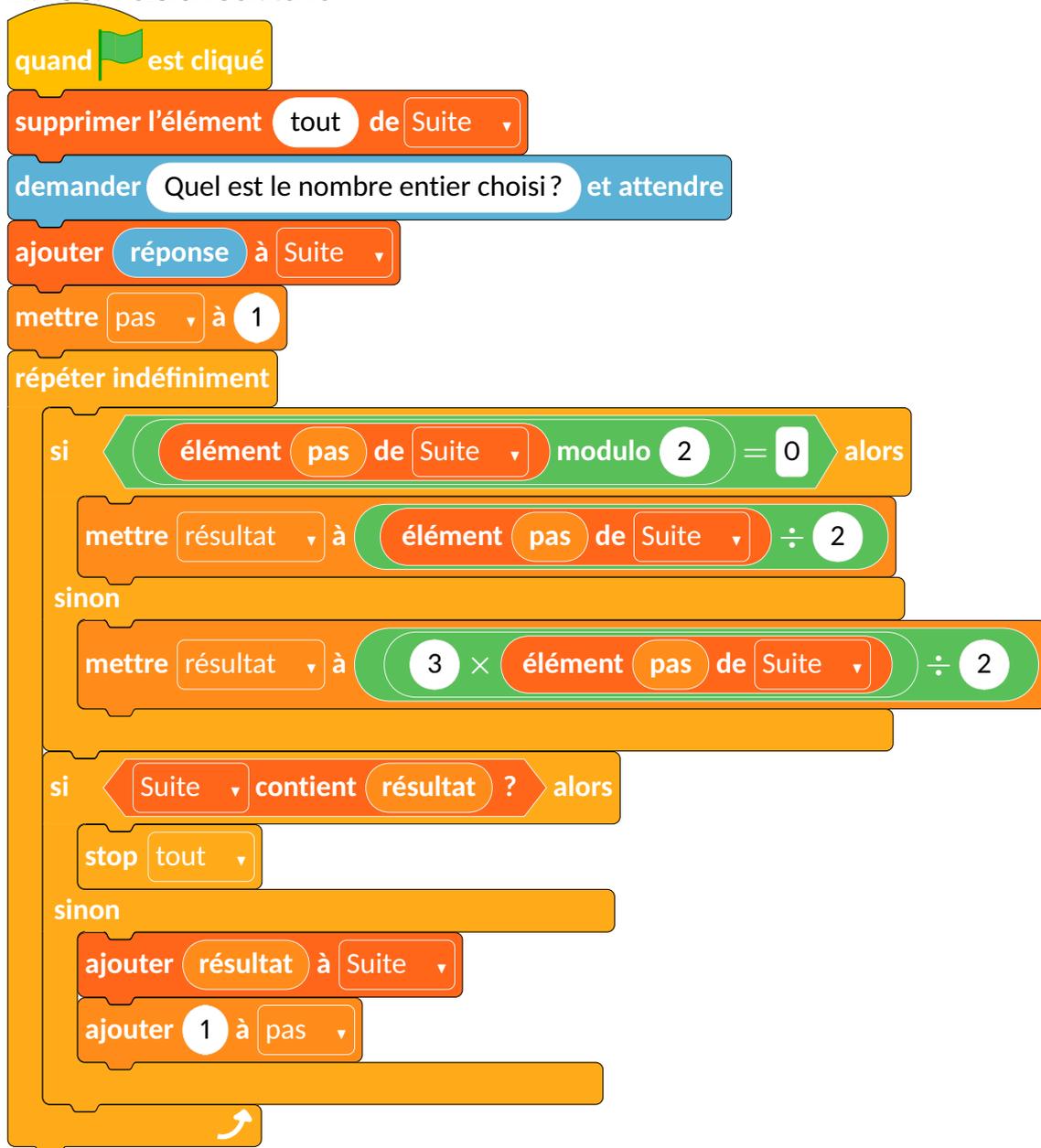
```

```

draw Sinon;
BB3=OvalOp(OvalNb("3"),"$\\bm{\\times} $" ,
  BB1);
draw MettreVar("résultat",OvalOp(BB3,"
  $\\bm{\\div} $" ,OvalNb("2")));
draw FinBlocSi;
draw Si(TestList(RecMenuList("Suite"),"
  contient",OvalVar("résultat")," ?"))
  ;
draw Stop("tout");
draw Sinon;
draw AjouterListe(OvalVar("résultat"),"
  Suite");
draw AjouterVar("pas","1");
draw FinBlocSi;
draw FinBlocRepeter;
endfig;

```

— Dans sa version Scratch 3 :



```

beginfig(1);
Scratchversion:=3;
draw Drapeau;
draw SupprimerListe("tout","Suite");
draw Demander("Quel est le nombre
entier choisi ?");
draw AjouterListe(OvalCap("réponse"),"
Suite");
draw MettreVar("pas","1");
draw RepeterI;
picture BB[];
BB1=OvalListMulti("élément",OvalVar("
pas"),"de",RecMenuList("Suite"));
BB2=OvalOp(BB1,"modulo",OvalNb("2"));
draw Si(TestOp(BB2,"$\bm{=} $" ,RecText("
0")));
draw MettreVar("résultat",OvalOp(BB1,"
$\bm{\div} $" ,OvalNb("2")));

draw Sinon;
BB3=OvalOp(OvalNb("3"),"$\bm{\times} $" ,
BB1);
draw MettreVar("résultat",OvalOp(BB3,"
$\bm{\div} $" ,OvalNb("2")));
draw FinBlocSi;
draw Si(TestList(RecMenuList("Suite"),"
contient",OvalVar("résultat")," ?"));
;
draw Stop("tout");
draw Sinon;
draw AjouterListe(OvalVar("résultat"),"
Suite");
draw AjouterVar("pas","1");
draw FinBlocSi;
draw FinBlocRepeter;
endfig;

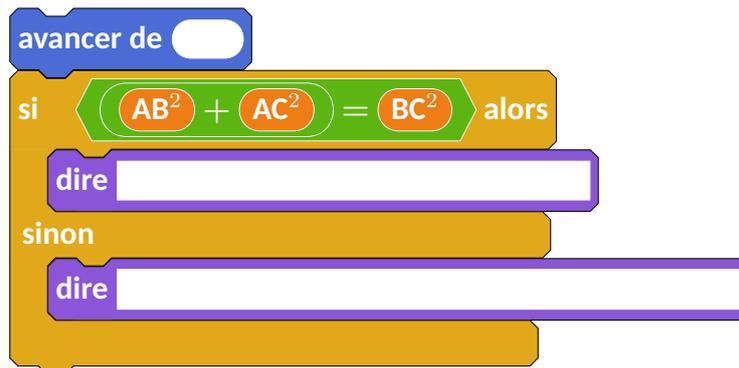
```



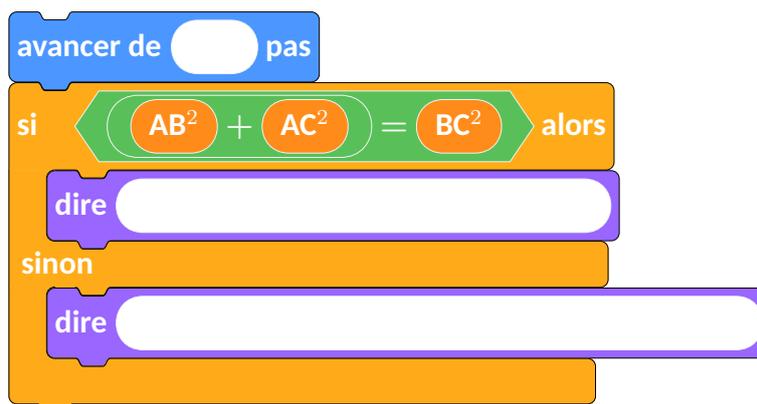
En scratch 3, on remarquera le changement de style à la fin des blocs Repeter ou Si. En effet, le bloc **Stop** étant spécial, il a fallu faire des adaptations. Ceci est rendu possible par le booléen **BlocStopAvant**.

Le coin du prof!

Cette partie recense des possibilités dont ne dispose pas, à ma connaissance, Scratch. Elles ont été rendues nécessaires par l'enseignement « débranché » de l'algorithmique. En plus de pouvoir produire des blocs avec des arguments vides,



```
beginfig(1);
  %Scratchversion:=3;
  draw Avancer("\phantom{100}");
  draw Si(TestOp(OvalOp(OvalVar("AB$^2$"), "$\bm{+}$", OvalVar("AC$^2$")), "$\bm{=} $" , OvalVar("BC$^2$")));
  draw Dire("\phantom{le triangle $ABC$ est rectangle en $A$}");
  draw Sinon;
  draw Dire("\phantom{le triangle $ABC$ n'est pas un triangle rectangle}");
  draw FinBlocSi;
endfig;
```



```
beginfig(1);
  Scratchversion:=3;
  draw Avancer("\phantom{100}");
  draw Si(TestOp(OvalOp(OvalVar("AB$^2$"), "$\bm{+}$", OvalVar("AC$^2$")), "$\bm{=} $" , OvalVar("BC$^2$")));
  draw Dire("\phantom{le triangle $ABC$ est rectangle en $A$}");
  draw Sinon;
```

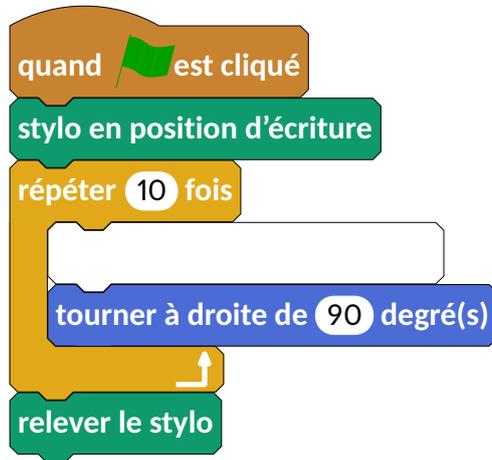
```

draw Dire("\phantom{le triangle $ABC$ n'est pas un triangle rectangle}");
draw FinBlocSi;
endfig;

```

les dispositifs supplémentaires sont les suivants :

- Une boîte puzzle « vide » afin de fournir des algorithmes à compléter :
- Dans sa version Scratch 2 :

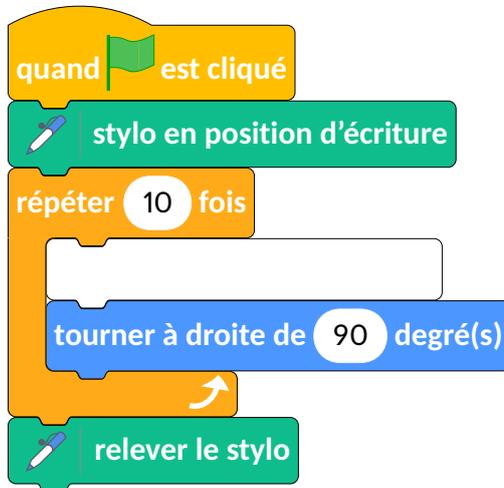


```

beginfig(1);
  %Scratchversion:=3;
  draw Drapeau;
  draw PoserStylo;
  draw Repeter("4");
  draw CommandeVide("5");%5cm
  draw Tournerd("90");
  draw FinBlocRepeater;
  draw ReleverStylo;
endfig;

```

- Dans sa version Scratch 3 :



```

beginfig(1);
  Scratchversion:=3;
  draw Drapeau;
  draw PoserStylo;
  draw Repeter("4");
  draw CommandeVide("5");%5cm
  draw Tournerd("90");
  draw FinBlocRepeater;
  draw ReleverStylo;
endfig;

```

- Une ligne « en pointillés » pour alléger l'écriture de certaines parties d'algorithmes :
- Dans sa version Scratch 2 :



```

beginfig(1);
  %Scratchversion:=3;
  draw Avancer("50");
  draw LignePointilles;
  draw Jouer("miaou");
endfig;

```

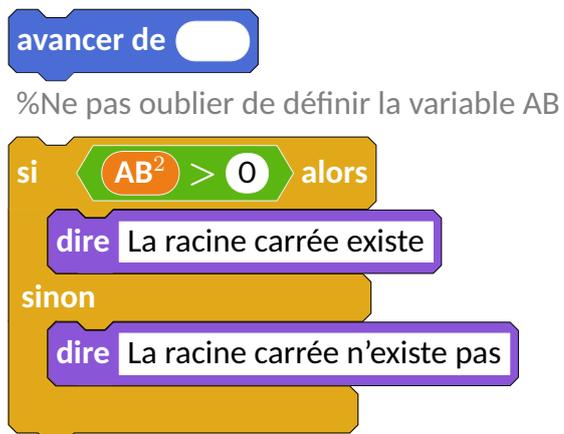
— Dans sa version Scratch 3 :



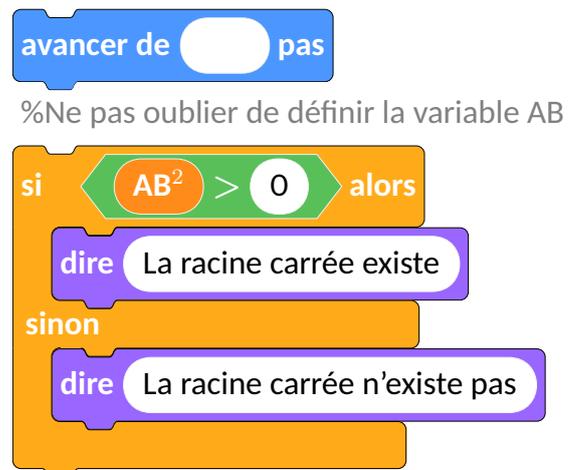
```
beginfig(1);  
Scratchversion:=3;  
draw Avancer("50");  
draw LignePointilles;  
draw Jouer("miaou");  
endfig;
```

— L'inclusion de commentaires :

— en ligne, en lieu et place des blocs « puzzle » :



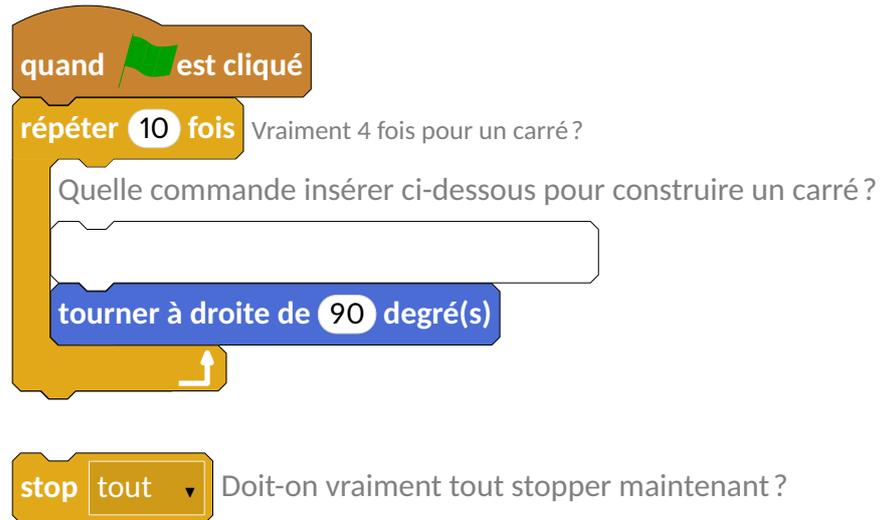
```
beginfig(1);  
%Scratchversion:=3;  
draw Avancer("\phantom{100}");  
draw Commentaires("\%Ne pas  
oublier de définir la  
variable AB");  
draw Si(TestOp(OvalVar("AB$^2$")  
, "$\bm{>}$", OvalNb("0")));  
draw Dire("La racine carrée  
existe");  
draw Sinon;  
draw Dire("La racine carrée n'  
existe pas");  
draw FinBlocSi;  
endfig;
```



```
beginfig(1);  
Scratchversion:=3;  
draw Avancer("\phantom{100}");  
draw Commentaires("\%Ne pas  
oublier de définir la  
variable AB");  
draw Si(TestOp(OvalVar("AB$^2$")  
, "$\bm{>}$", OvalNb("0")));  
draw Dire("La racine carrée  
existe");  
draw Sinon;  
draw Dire("La racine carrée n'  
existe pas");  
draw FinBlocSi;  
endfig;
```

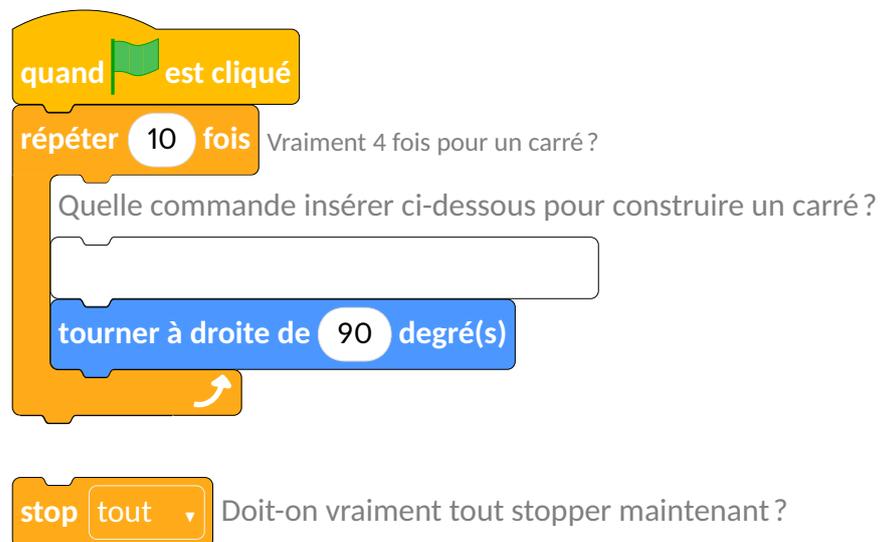
— en bout de bloc « puzzle » :

— Dans sa version Scratch 2 :



```
beginfig(1);  
  %Scratchversion:=3;  
  draw Drapeau;  
  draw Repeter("4");  
  draw CommentairesLigne("\footnotesize Vraiment 4 fois pour un  
    carré ?");  
  draw Commentaires("Quelle commande insérer ci-dessous pour  
    construire un carré ?");  
  draw CommandeVide("7");  
  draw Tournerd("90");  
  draw FinBlocRepeter;  
  draw LigneVide;  
  draw Stop("tout");  
  draw CommentairesLigne("Doit-on vraiment tout stopper maintenant  
    ?");  
endfig;
```

— Dans sa version Scratch 3 :



```
beginfig(1);  
  Scratchversion:=3;
```

```

draw Drapeau;
draw Repeter("4");
draw CommentairesLigne("\footnotesize Vraiment 4 fois pour un
    carré ?");
draw Commentaires("Quelle commande insérer ci-dessous pour
    construire un carré ?");
draw CommandeVide("7");
draw Tournerd("90");
draw FinBlocRepeter;
draw LigneVide;
draw Stop("tout");
draw CommentairesLigne("Doit-on vraiment tout stopper maintenant
    ?");
endfig;

```

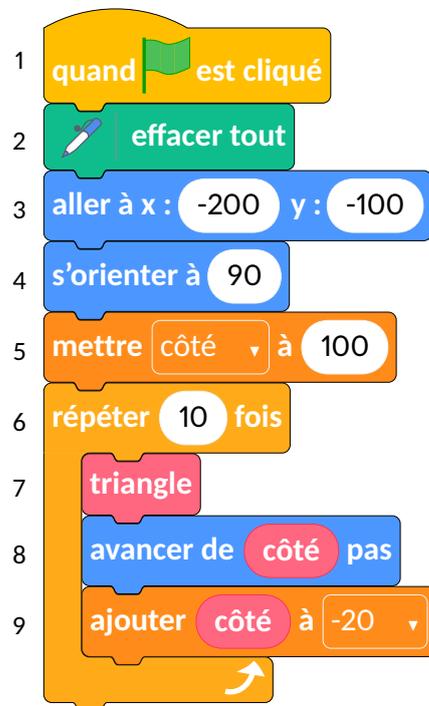
— La numérotation des lignes¹³ d'un algorithme :



```

beginfig(1);
    %Scratchversion:=3;
    NumeroteLignes:=true;
    draw Drapeau;
    draw Effacer;
    draw Aller("-200","-100");
    draw Orienter("90");
    draw MettreVar("côté","100");
    draw Repeter("5");
    draw Bloc("triangle");
    draw Avancer(OvalBloc("côté"));
    draw AjouterVar("côté","-20");
    draw FinBlocRepeter;
endfig;

```



```

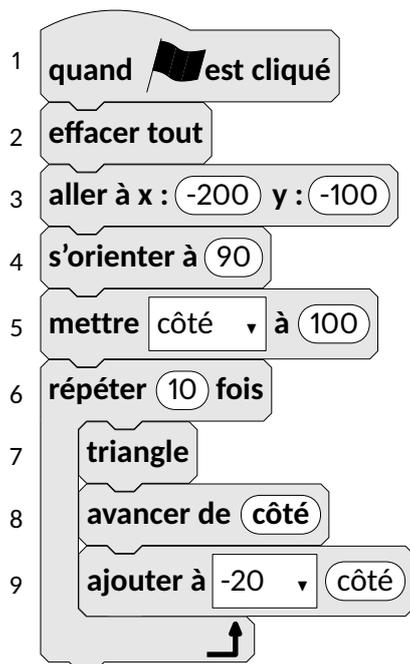
beginfig(1);
    Scratchversion:=3;
    NumeroteLignes:=true;
    draw Drapeau;
    draw Effacer;
    draw Aller("-200","-100");
    draw Orienter("90");
    draw MettreVar("côté","100");
    draw Repeter("5");
    draw Bloc("triangle");
    draw Avancer(OvalBloc("côté"));
    draw AjouterVar("côté","-20");
    draw FinBlocRepeter;
endfig;

```

13. Idée qui est apparue nécessaire à la lecture du sujet de mathématiques du Brevet des Collèges 2017

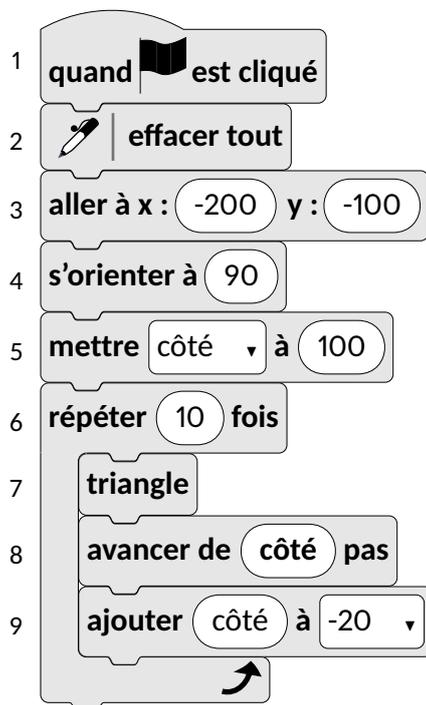
— La disparition des couleurs¹⁴ pour une impression visuellement meilleure pour les élèves :

— Dans sa version Scratch 2 :



```
beginfig(1);
  %Scratchversion:=3;
  coefprint:=0.9;
  print:=true;
  NumeroteLignes:=true;
  draw Drapeau;
  draw Effacer;
  draw Aller("-200","-100");
  draw Orienter("90");
  draw MettreVar("côté","100");
  draw Repeter("5");
  draw Bloc("triangle");
  draw Avancer(OvalBloc("côté"));
  draw AjouterVar("côté","-20");
  draw FinBlocRepeter;
endfig;
```

— Dans sa version Scratch 3 :



```
beginfig(1);
  Scratchversion:=3;
  coefprint:=0.9;
  print:=true;
  NumeroteLignes:=true;
  draw Drapeau;
  draw Effacer;
  draw Aller("-200","-100");
  draw Orienter("90");
  draw MettreVar("côté","100");
  draw Repeter("5");
  draw Bloc("triangle");
  draw Avancer(OvalBloc("côté"));
  draw AjouterVar("côté","-20");
  draw FinBlocRepeter;
endfig;
```



La constante `coefprint` sert à modifier le niveau de gris pour l'impression.

14. Suggestion proposée lors du stage L^AT_EX de Dunkerque (Juin 2017) et par le package `scratch` en TikZ de Christian TELLECHEA.

On peut également être tenter de se détacher de Scratch et de son installation liée à Adobe Air¹⁵...

Ainsi, on peut penser à tester :

- Snap, utilisable en ligne à l'adresse <http://snap.berkeley.edu/snapsource/snap.html#>
- Phratch, utilisable en local et disponible à l'adresse <http://www.phratch.com/>.

Si Snap propose des blocs aux couleurs très proches de celles de Scratch, il est à noter que Phratch a une catégorie Couleurs possédant des blocs comme celui ci-dessous :



```
beginfig(1);
  %Scratchversion:=3;
  draw BlocUser((64/256,64/256,64/256)
    )("fixer ",RecMenuText("Couleur1"
    )," \'a ",RecCouleur
    (144/256,54/256,122/256));
endfig;
```



```
beginfig(1);
  Scratchversion:=3;
  draw BlocUser((64/256,64/256,64/256)
    )("fixer ",RecMenuText("Couleur1"
    )," \'a ",RecCouleur
    (144/256,54/256,122/256));
endfig;
```

ou encore une catégorie Fichiers comme ci-dessous



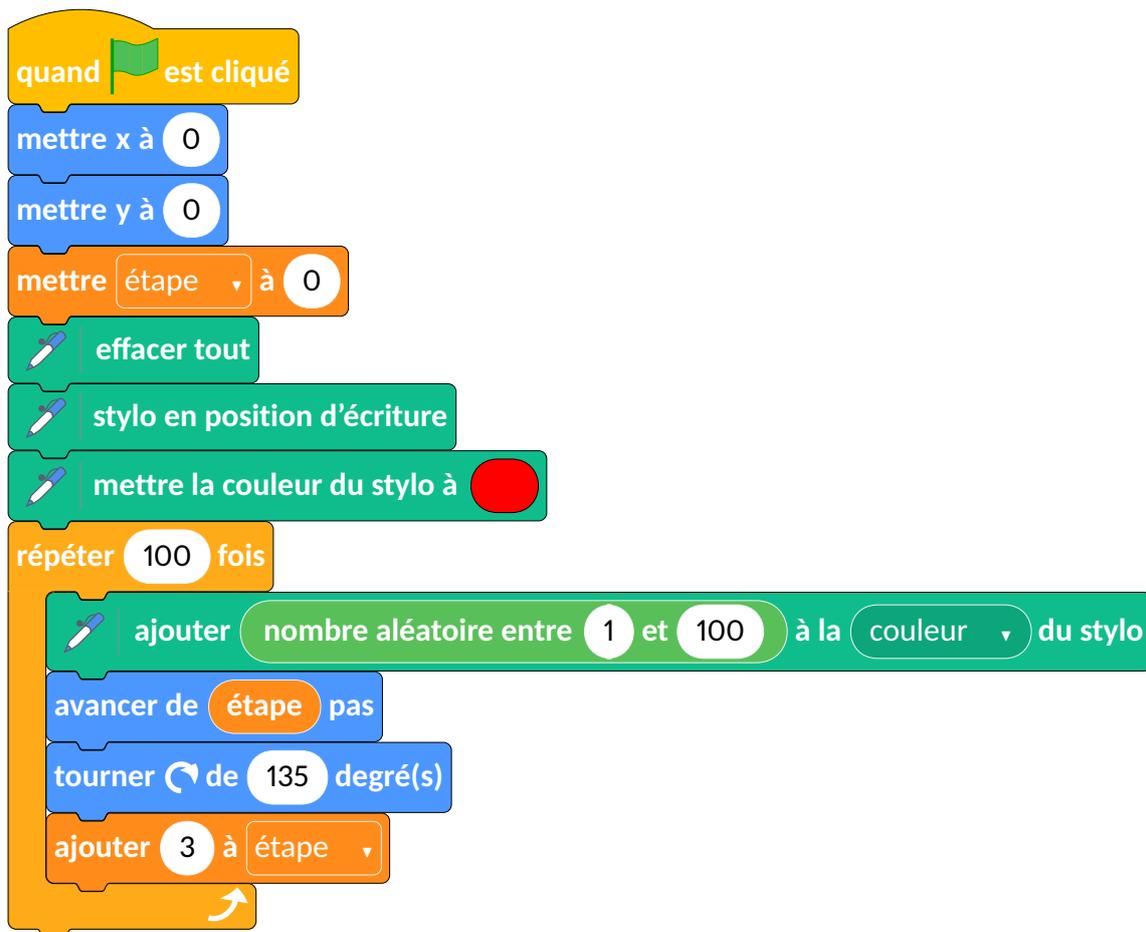
```
beginfig(1);
  %Scratchversion:=3;
  draw BlocUser((64/256,64/256,64/256)
    )("fixer ",RecMenuText("Couleur1"
    )," \'a ",RecCouleur
    (144/256,54/256,122/256));
  draw BlocUser
    ((44/256,120/256,195/256))("
    supprimer le fichier",RecText("
    Exemple.png"));
endfig;
```



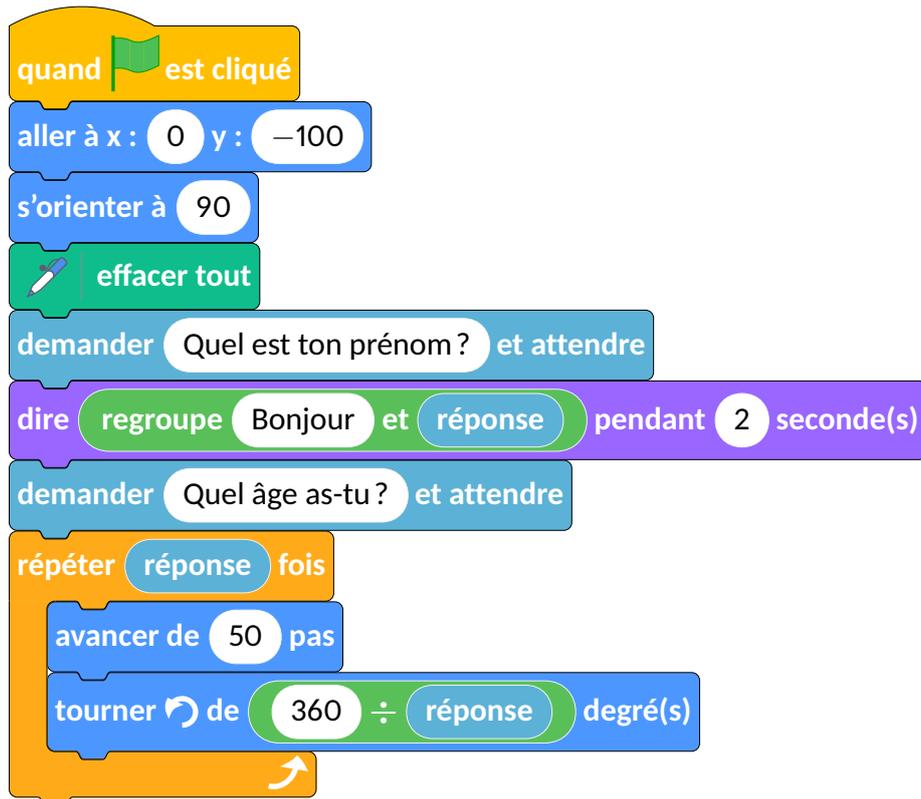
```
beginfig(1);
  Scratchversion:=3;
  draw BlocUser((64/256,64/256,64/256)
    )("fixer ",RecMenuText("Couleur1"
    )," \'a ",RecCouleur
    (144/256,54/256,122/256));
  draw BlocUser
    ((44/256,120/256,195/256))("
    supprimer le fichier",RecText("
    Exemple.png"));
endfig;
```

15. En tout cas pour la version 2.

Galerie d'exemples

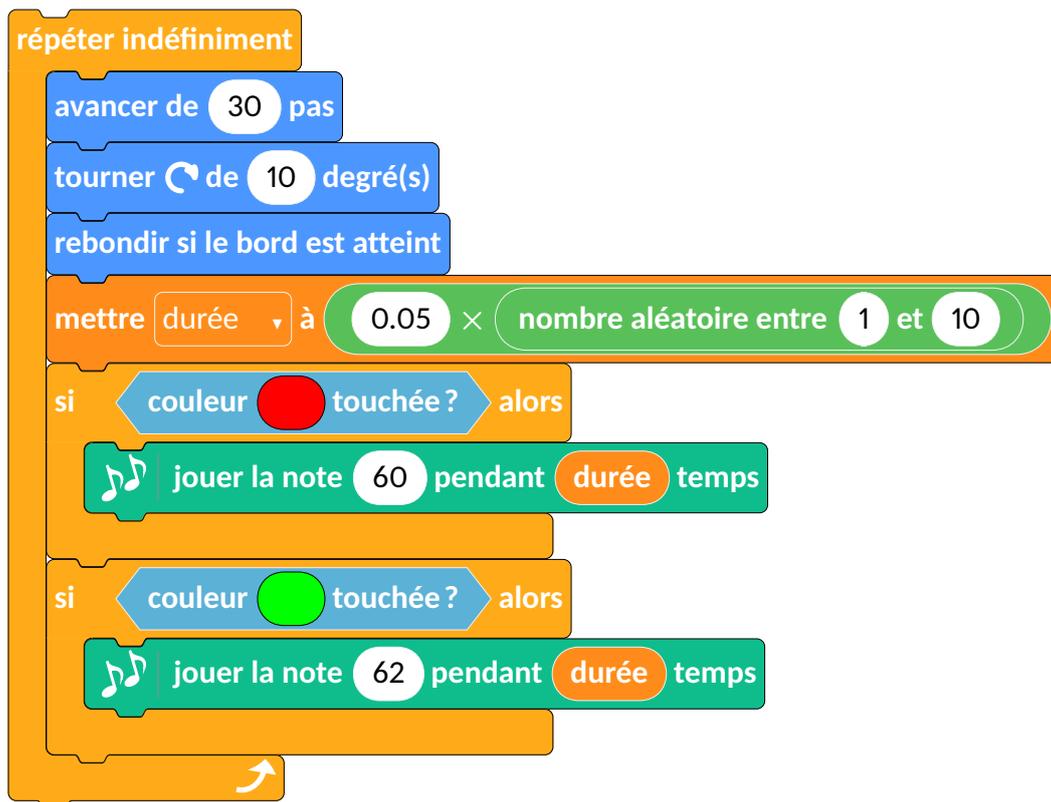


```
beginfig(1);  
  draw Drapeau;  
  draw Mettre("x", "0");  
  draw Mettre("y", "0");  
  draw MettreVar("étape", "0");  
  draw Effacer;  
  draw PoserStylo;  
  draw MettreCouleur(1,0,0);  
  draw Repeter("100");  
  draw AjouterCS(OvalOp("nombre aléatoire entre",OvalNb("1"),"et",OvalNb("100"  
    )), "couleur");  
  draw Avancer(OvalVar("étape"));  
  draw Tournerd("135");  
  draw AjouterVar("3", "étape");  
  draw FinBlocRepeter;  
endfig;
```



```

beginfig(1);
  draw Drapeau;
  draw Aller("0", "$-$100");
  draw Orienter("90");
  draw Effacer;
  draw Demander("Quel est ton prénom ?");
  draw DireT(OvalOp("regroupe",OvalText("Bonjour"),"et",OvalCap("réponse")), "2
  ");
  draw Demander("Quel âge as-tu ?");
  draw Repeter(OvalCap("réponse"));
  draw Avancer("50");
  draw Tournerg(OvalOp(OvalNb("360"), "$\bm{\div}$",OvalCap("réponse")));
  draw FinBlocRepeter;
endfig;
  
```



```

beginfig(1);
draw RepeterI;
draw Avancer("30");
draw Tournerd("10");
draw Rebondir;
draw MettreVar("durée",OvalOp(OvalNb("0.05"),"$\bm{\times}$",OvalOp("nombre
    aléatoire entre",OvalNb("1"),"et",OvalNb("10"))));
draw Si(TestCap("couleur",OvalCouleur(1,0,0),"touchée ?"));
draw JouerNote("60",OvalVar("durée"));
draw FinBlocSi;
draw Si(TestCap("couleur",OvalCouleur(0,1,0),"touchée ?"));
draw JouerNote("62",OvalVar("durée"));
draw FinBlocSi;
draw FinBlocRepeter;
endfig;

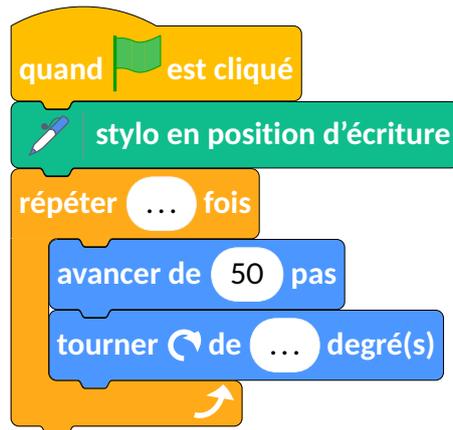
```



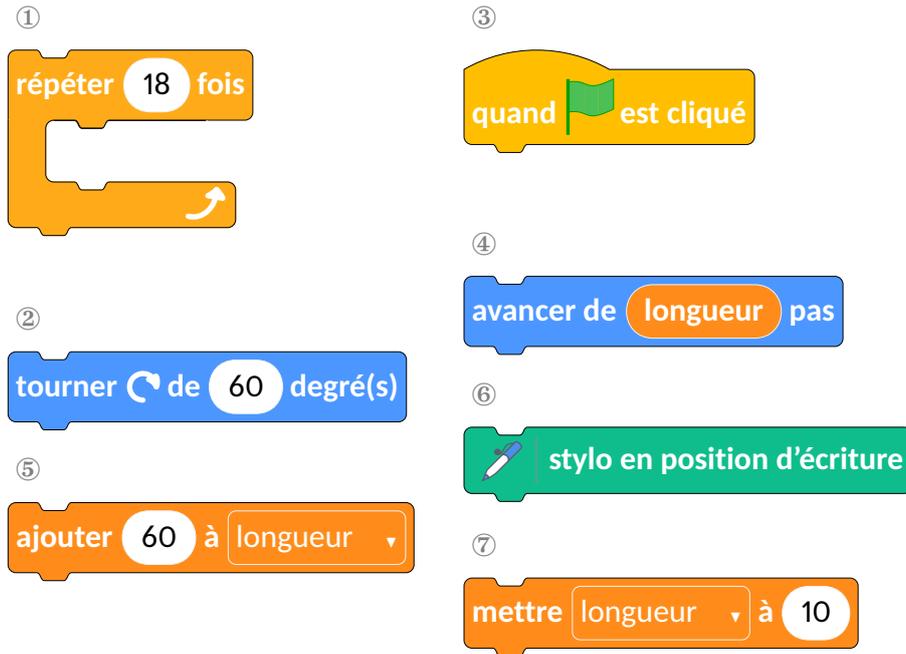
```

beginfig(1);
  draw Drapeau;
  draw SupprimerListeAll("notes");
  draw MettreVar("nombre de notes","3");
  draw Repeter(OvalVar("nombre de notes"));
  draw Demander("Une note ?");
  draw AjouterListe(OvalCap("réponse"),"notes");
  draw FinBlocRepeter;
  draw MettreVar("somme","0");
  draw MettreVar("n","0");
  draw Repeter(OvalVar("nombre de notes"));
  draw AjouterVar("1","n");
  draw AjouterVar(OvalList("élément",OvalVar("n"),"de",RecListMenu("notes")),
    "somme");
  draw FinBlocRepeter;
  draw MettreVar("moyenne",OvalOp(OvalVar("somme"),"$\bm{\div}$",OvalVar("
    nombre de notes")));
  draw Dire(OvalOp("regroupe",OvalText("Moyenne="),"et",OvalVar("moyenne")));
endfig;

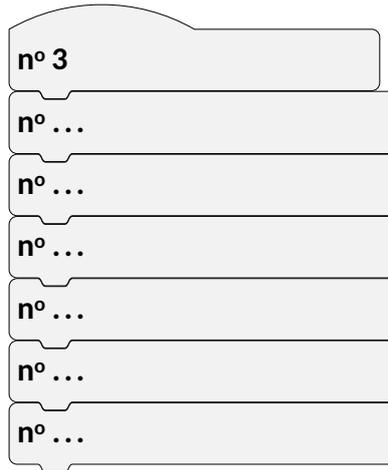
```



```
Scratchversion:=3;
symbole:=true;
beginfig(1);
  draw Drapeau;
  draw PoserStylo;
  draw Repeter("\dots");
  draw Avancer("50");
  draw Tournerd("\dots");
  draw FinBlocRepeter;
endfig;
```



```
Scratchversion:=3;
symbole:=true;
beginfig(2);
  draw Commentaires("\ding{172}");
  draw Repeter("18");
  draw LigneVide;
  draw FinBlocRepeter;
  Deplacera(0,-4cm);
  draw Commentaires("\ding{173}");
  draw Tournerd("60");
  Deplacera(0,-6cm);
  draw Commentaires("\ding{176}");
  draw AjouterVar("60","longueur");
  Deplacera(6cm,0);
  draw Commentaires("\ding{174}");
  draw Drapeau;
  Deplacera(6cm,-3cm);
  draw Commentaires("\ding{175}");
  draw Avancer(OvalVar("longueur"));
  Deplacera(6cm,-5cm);
  draw Commentaires("\ding{177}");
  draw PoserStylo;
  Deplacera(6cm,-7cm);
  draw Commentaires("\ding{178}");
  draw MettreVar("longueur","10");
endfig;
```



```

Scratchversion:=3;
symbole:=true;
print:=true;

beginfig(3);
  BlocE:=true;
  draw BlocUser(0.95*white)("\no3\hbox to4cm{");
  BlocE:=false;
  draw BlocUser(0.95*white)("\no\dots\hbox to4cm{");
  draw BlocUser(0.95*white)("\no\dots\hbox to4cm{");
endfig;

```

Historique

- 29/02/2020 Version 0.9** - Refonte *complète* de mp-scratch, notamment avec un « collage » récursif enfin réussi :). Quelques correctifs. Apparition des « icônes » (notes²², stylo²³ et vidéo²⁴) dans les briques. Mise à jour de la documentation.
- 20/02/2020 Version 0.8** - Refonte *partielle* de mp-scratch. Ajout d'une option de changement de version de Scratch (pour passage à la version 3 (Enfin :)).
- 07/07/2017 Version 0.7** - Refonte de mp-scratch. Ajout d'une option d'impression. Ajout d'une option de numérotation des blocs.
- 15/04/2017 Version 0.63** - Ajout d'une commande Commentaires et mise à jour de la documentation.
- 07/03/2017 Version 0.62** - Ajout d'une commande LignePointilles et mise à jour de la documentation.
- 17/02/2017 Version 0.61** - Grâce à Thomas DEHON, ajout des commandes correspondantes à la sélection de « la scène ». Mise à jour de la documentation.
- 16/02/2017 Version 0.59** - Correction des commandes Dire, DireT, Penser, PenserT. Mise à jour de la documentation (informations sur l'installation du package).
- 15/02/2017 Version 0.57** - Correction de problèmes mineurs d'affichage. Correction de la documentation.
- 14/02/2017 Version 0.55** - Mise à jour de la documentation.
- 13/02/2017 Version 0.53** - Ajout des chanfreins sur les blocs. Correction de « doublons » de commandes. Mise à jour de la documentation.
- 05/02/2017 Version 0.51** - Sur les conseils de Maxime CHUPIN et Thierry PASQUIER, travail sur les couleurs (mise en accord avec celles de Scratch et personnalisation possible). Passage des majuscules aux minuscules pour les blocs.
- 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, Évènements, Contrôle.

22. Grâce au package bclogo de Maxime CHUPIN.

23. Grâce au package scratch3 de Christian TELLECHEA.

24. Par création personnelle :).

Pense-bête

Mouvement

| | |
|--|--|
| avancer de 10 pas | <pre>beginfig(1); draw Avancer("10"); endfig;</pre> |
| le boolean symbole est égal à false tourner à droite de 15 degré(s) | <pre>beginfig(2); symbole:=false; draw Commentaires("le boolean symbole est égal à false"); draw Tournerd("15"); endfig;</pre> |
| le boolean symbole est égal à true tourner ↻ de 15 degré(s) | <pre>beginfig(3); symbole:=true;%valeur par défaut draw Commentaires("le boolean symbole est égal à true"); draw Tournerd("15"); endfig;</pre> |
| le boolean symbole est égal à false tourner à gauche de 15 degré(s) | <pre>beginfig(4); symbole:=false; draw Commentaires("le boolean symbole est égal à false"); draw Tournerg("15"); endfig;</pre> |
| le boolean symbole est égal à true tourner ↻ de 15 degré(s) | <pre>beginfig(5); symbole:=true;%valeur par défaut draw Commentaires("le boolean symbole est égal à true"); draw Tournerg("15"); endfig;</pre> |
| aller à position aléatoire ▾ | <pre>beginfig(6); draw Allera("position aléatoire"); endfig;</pre> |
| aller à x : 10 y : 200 | <pre>beginfig(7); draw Aller("10","200"); endfig;</pre> |
| glisser en 1 seconde(s) à position aléatoire ▾ | <pre>beginfig(8); draw Glissera("1","position aléatoire"); endfig;</pre> |
| glisser en 1 seconde(s) à x : 63 y : 30 | <pre>beginfig(9); draw Glisser("1","63","30"); endfig;</pre> |
| s'orienter à 90 | <pre>beginfig(10); draw Orienter("90"); endfig;</pre> |

| | |
|--|--|
| s'orienter vers <input type="text" value="pointeur de souris"/> | <pre>beginfig(11); draw Orienterdirection("pointeur de souris"); endfig;</pre> |
| ajouter <input type="text" value="10"/> à x | <pre>beginfig(12); draw Ajouter("10","x"); endfig;</pre> |
| mettre x à <input type="text" value="63"/> | <pre>beginfig(13); draw Mettre("x","63"); endfig;</pre> |
| ajouter <input type="text" value="10"/> à y | <pre>beginfig(14); draw Ajouter("10","y"); endfig;</pre> |
| mettre y à <input type="text" value="63"/> | <pre>beginfig(15); draw Mettre("y","63"); endfig;</pre> |
| rebondir si le bord est atteint | <pre>beginfig(16); draw Rebondir; endfig;</pre> |
| fixer le sens de rotation <input type="text" value="gauche-droite"/> | <pre>beginfig(17); draw FixerSensRotation("gauche-droite"); endfig;</pre> |
| abscisse x | <pre>beginfig(18); draw OvalMouv("abscisse x"); endfig;</pre> |
| ordonnée y | <pre>beginfig(19); draw OvalMouv("ordonnée y"); endfig;</pre> |
| direction | <pre>beginfig(20); draw OvalMouv("direction"); endfig;</pre> |

Apparence

dire Bonjour! pendant 2 seconde(s)

```
beginfig(1);  
  draw DireT("Bonjour !", "2");  
endfig;
```

dire Bonjour!

```
beginfig(2);  
  draw Dire("Bonjour !");  
endfig;
```

penser à Hmm... pendant 2 seconde(s)

```
beginfig(3);  
  draw PenserT("Hmm\dots", "2");  
endfig;
```

penser à Hmm...

```
beginfig(4);  
  draw Penser("Hmm\dots");  
endfig;
```

basculer sur le costume guitar-electric2-b

```
beginfig(5);  
  draw Basculer("guitar-electric2-b"  
  );  
endfig;
```

costume suivant

```
beginfig(6);  
  draw CostumeSuivant;  
endfig;
```

basculer sur l'arrière-plan arrière plan 1

```
beginfig(7);  
  draw BasculerAR("arrière plan 1");  
endfig;
```

arrière-plan suivant

```
beginfig(8);  
  draw ARSuivant;  
endfig;
```

ajouter 10 à la taille

```
beginfig(9);  
  draw AjouterTaille("10");  
endfig;
```

mettre la taille à 100 % de la taille initiale

```
beginfig(10);  
  draw MettreA("100");  
endfig;
```

ajouter 25 à l'effet couleur

```
beginfig(11);  
  draw AjouterEffet("25", "couleur");  
endfig;
```

mettre l'effet couleur à 0

```
beginfig(12);  
  draw MettreEffet("couleur", "0");  
endfig;
```

annuler les effets graphiques

```
beginfig(13);  
  draw AnnulerEffet;  
endfig;
```

Montrer

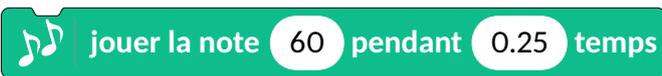
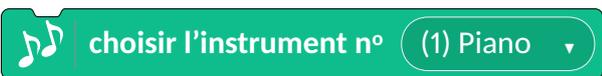
```
beginfig(14);  
  draw Montrer;  
endfig;
```

Cacher

```
beginfig(15);  
  draw Cacher;  
endfig;
```

| | |
|---|--|
| aller à l' <input type="text" value="avant"/> plan | <pre>beginfig(16); draw AllerPlan("avant"); endfig;</pre> |
| déplacer de <input type="text" value="1"/> plan(s) vers l' <input type="text" value="avant"/> | <pre>beginfig(17); draw DeplacerPlan("1","avant"); endfig;</pre> |
| <input type="text" value="numéro"/> du costume | <pre>beginfig(18); draw OvalApp(ColleBoxNew(RecAppMenu("numéro"),"du costume")); endfig;</pre> |
| <input type="text" value="numéro"/> de l'arrière-plan | <pre>beginfig(19); draw OvalApp(ColleBoxNew(RecAppMenu("numéro")," de l'arrière-plan")); endfig;</pre> |
| taille | <pre>beginfig(20); draw OvalApp("taille"); endfig;</pre> |
| basculer sur l'arrière-plan <input type="text" value="1"/> et attendre | <pre>beginfig(21); draw BasculerARA("1"); endfig;</pre> |

Son

| | |
|--|--|
|  | <pre>beginfig(1); draw JouerT("Miaou"); endfig;</pre> |
|  | <pre>beginfig(2); draw Jouer("Miaou"); endfig;</pre> |
|  | <pre>beginfig(3); draw ArreterSon; endfig;</pre> |
|  | <pre>beginfig(4); draw AjouterEffetSon("10","hauteur"); endfig;</pre> |
|  | <pre>beginfig(5); draw MettreEffetSon("hauteur","100"); endfig;</pre> |
|  | <pre>beginfig(6); draw AnnulerEffetSon; endfig;</pre> |
|  | <pre>beginfig(7); draw AjouterVol("\$-\$10"); endfig;</pre> |
|  | <pre>beginfig(8); draw MettreVol("90"); endfig;</pre> |
|  | <pre>beginfig(9); draw OvalSon("volume"); endfig;</pre> |
|  | <pre>beginfig(10); draw Tambour(" (1) Caisse claire","0.25 "); endfig;</pre> |
|  | <pre>beginfig(11); draw Pause("0.25"); endfig;</pre> |
|  | <pre>beginfig(12); draw JouerNote("60","0.25"); endfig;</pre> |
|  | <pre>beginfig(13); draw ChoisirInstrument("(1) Piano"); endfig;</pre> |
|  | <pre>beginfig(14); draw MettreTempo("15"); endfig;</pre> |

 ajouter 20 au tempo

```
beginfig(15);  
  draw AjouterTempo("20");  
endfig;
```

 tempo

```
beginfig(16);  
  draw OvalMusique("tempo");  
endfig;
```

Stylo

 effacer tout

```
beginfig(1);  
  draw Effacer;  
endfig;
```

 estampiller

```
beginfig(2);  
  draw Estampiller;  
endfig;
```

 stylo en position d'écriture

```
beginfig(3);  
  draw PoserStylo;  
endfig;
```

 relever le stylo

```
beginfig(4);  
  draw ReleverStylo;  
endfig;
```

 mettre la couleur du stylo à 

```
beginfig(5);  
  draw MettreCouleur(1,0.5,0.25);  
endfig;
```

 ajouter 10 à la couleur du stylo

```
beginfig(6);  
  draw AjouterCS("10","couleur");  
endfig;
```

 mettre la transparence du stylo à 50

```
beginfig(7);  
  draw MettreCS("transparence",  
    50");  
endfig;
```

 ajouter 10 à la taille du stylo

```
beginfig(10);  
  draw AjouterTS("10");  
endfig;
```

 mettre la taille du stylo à 10

```
beginfig(11);  
  draw MettreTS("10");  
endfig;
```

Évènements

| | |
|--|--|
| quand  est cliqué | <pre>beginfig(1); draw Drapeau; endfig;</pre> |
| quand la touche espace est pressée | <pre>beginfig(2); draw QPresse("espace"); endfig;</pre> |
| quand ce sprite est cliqué | <pre>beginfig(3); draw QLutinPresse; endfig;</pre> |
| quand l'arrière-plan bascule sur arrière-plan 1 | <pre>beginfig(4); draw QBasculeAR("arrière-plan 1"); endfig;</pre> |
| quand le volume sonore > 10 | <pre>beginfig(5); draw QVolumeSup("volume sonore","10"); endfig;</pre> |
| quand je reçois message 1 | <pre>beginfig(6); draw QRecevoirMessage("message 1"); endfig;</pre> |
| envoyer à tous message 1 | <pre>beginfig(7); draw EnvoyerMessage("message 1"); endfig;</pre> |
| envoyer à tous message 1 et attendre | <pre>beginfig(8); draw EnvoyerMessageA("message 1"); endfig;</pre> |
| quand la scène est cliquée | <pre>beginfig(9); draw QScenePressee; endfig;</pre> |

Contrôle

attendre 10 seconde(s)

```
beginfig(1);  
  draw Attendre("10");  
endfig;
```

répéter 10 fois

```
beginfig(2);  
  draw Repeter("10");  
  draw LigneVide;  
  draw FinBlocRepeter;  
endfig;
```

répéter indéfiniment

```
beginfig(3);  
  draw RepeterI;  
  draw LigneVide;  
  draw FinBlocRepeter;  
endfig;
```

si réponse = 10 alors

```
beginfig(4);  
  draw Si(TestOp(OvalCap("réponse"), "$\bm{=} $"  
    ,OvalNb("10")));  
  draw LigneVide;  
  draw FinBlocSi;  
endfig;
```

si réponse = 10 alors
sinon

```
beginfig(5);  
  draw Si(TestOp(OvalCap("réponse"), "$\bm{=} $"  
    ,OvalNb("10")));  
  draw LigneVide;  
  draw Sinon;  
  draw LigneVide;  
  draw FinBlocSi;  
endfig;
```

attendre jusqu'à ce que réponse = 10

```
beginfig(6);  
  draw AttendreJ(TestOp(OvalCap("réponse"), "$\bm{=} $"  
    ,OvalNb("10")));  
endfig;
```

répéter jusqu'à ce que réponse = 10

```
beginfig(7);  
  draw RepeterJ(TestOp(OvalCap("réponse"), "$\bm{=} $"  
    ,OvalNb("10")));  
  draw LigneVide;  
  draw FinBlocRepeter;  
endfig;
```

stop tout

```
beginfig(8);  
  draw Stop("tout");  
endfig;
```

quand je commence comme un clone

```
beginfig(9);  
  draw CommencerClone;  
endfig;
```

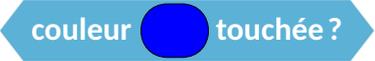
créer un clone de moi-même

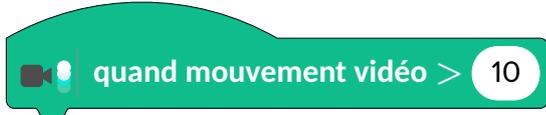
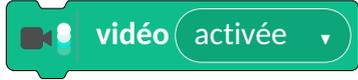
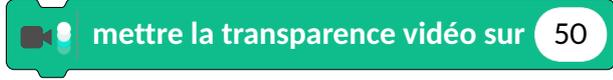
```
beginfig(10);  
  draw CreerClone("moi-même");  
endfig;
```

supprimer ce clone

```
beginfig(11);  
  draw SupprimerClone;  
endfig;
```

Capteurs

| | |
|---|---|
|  | <pre>beginfig(1); draw TestCap("touche le ",OvalCapMenu(" pointeur de souris")," ?"); endfig;</pre> |
|  | <pre>beginfig(2); draw TestCap("couleur",OvalCouleur(0,0,1)," touchée ?"); endfig;</pre> |
|  | <pre>beginfig(3); draw TestCap("couleur",OvalCouleur(1,0,0)," touche ",OvalCouleur(0,1,0)); endfig;</pre> |
|  | <pre>beginfig(4); draw OvalCap("distance de",OvalMenuCap(" pointeur de souris")); endfig;</pre> |
|  | <pre>beginfig(5); draw Demander("What's your name ?"); endfig;</pre> |
|  | <pre>beginfig(6); draw OvalCap("réponse"); endfig;</pre> |
|  | <pre>beginfig(7); draw TestCap("touche",OvalMenuCap("espace")," pressée ?"); endfig;</pre> |
|  | <pre>beginfig(8); draw TestCap("souris pressée ?"); endfig;</pre> |
|  | <pre>beginfig(9); draw OvalCap("souris x"); endfig;</pre> |
|  | <pre>beginfig(10); draw OvalCap("souris y"); endfig;</pre> |
|  | <pre>beginfig(11); draw MettreGlissement("glissable"); endfig;</pre> |
|  | <pre>beginfig(12); draw OvalCap("volume sonore"); endfig;</pre> |
|  | <pre>beginfig(13); draw OvalCap("chronomètre"); endfig;</pre> |
|  | <pre>beginfig(14); draw ReinitChrono; endfig;</pre> |

| | |
|---|--|
|  | <pre>beginfig(15); draw OvalCap(RecMenuCap("numéro de l'arrière-plan")," de ", OvalMenuCap("la scène")); endfig;</pre> |
|  | <pre>beginfig(16); draw OvalCap(RecMenuCap("année"),"actuelle"); endfig;</pre> |
|  | <pre>beginfig(17); draw OvalCap("jours depuis 2000"); endfig;</pre> |
|  | <pre>beginfig(18); draw OvalCap("nom d'utilisateur"); endfig;</pre> |
|  | <pre>beginfig(19); draw QuandMV("10"); endfig;</pre> |
|  | <pre>beginfig(20); draw VideoSur("mouvement","sprite"); endfig;</pre> |
|  | <pre>beginfig(21); draw ActiverVideo("activée"); endfig;</pre> |
|  | <pre>beginfig(22); draw TransparenceVideo("50"); endfig;</pre> |

Opérateurs



```
beginfig(1);
  draw OvalOp(OvalNb("10"), "$\bm{+}$", OvalNb("20"));
endfig;
```



```
beginfig(2);
  draw OvalOp(OvalNb("10"), "$\bm{+}$", OvalNb("20"));
endfig;
```



```
beginfig(3);
  draw OvalOp(OvalNb("10"), "$\bm{+}$", OvalNb("20"));
endfig;
```



```
beginfig(4);
  draw OvalOp(OvalNb("10"), "$\bm{+}$", OvalNb("20"));
endfig;
```



```
beginfig(5);
  draw TestOp("nombre aléatoire entre",
    OvalNb("1"), "et", OvalNb("10"));
endfig;
```



```
beginfig(6);
  draw TestOp(OvalNb("\hbox to0.5em{") , "$\bm{>}$", OvalNb("50"));
endfig;
```



```
beginfig(7);
  draw TestOp(OvalNb("\hbox to0.5em{") , "$\bm{<}$", OvalNb("50"));
endfig;
```



```
beginfig(8);
  draw TestOp(OvalNb("\hbox to0.5em{") , "$\bm{=}$", OvalNb("50"));
endfig;
```



```
beginfig(9);
  draw TestOp(TestOp(
    OvalCap("réponse"), "$\bm{<}$", OvalNb("100"))
    , "ou", TestOp(OvalCap("réponse"), "$\bm{>}$", OvalNb("50")));
endfig;
```



```
beginfig(10);
  draw TestOp(TestOp(
    OvalCap("réponse"), "$\bm{<}$", OvalNb("100"))
    , "ou", TestOp(OvalCap("réponse"), "$\bm{>}$", OvalNb("50")));
endfig;
```



```
beginfig(11);
  draw TestOp("non", TestOp(OvalCap("réponse"), "$\bm{>}$", OvalNb("50")));
endfig;
```

regrouper apple et banana

```
beginfig(12);  
draw OvalOp("regrouper",OvalNb("apple"),"et"  
            ,OvalNb("banana"));  
endfig;
```

lettre 1 de apple

```
beginfig(13);  
draw OvalOp("lettre",OvalNb("1"),"de",OvalNb("apple"  
        ));  
endfig;
```

longueur de apple

```
beginfig(14);  
draw OvalOp("longueur de",OvalNb("apple"));  
endfig;
```

apple contient a

```
beginfig(15);  
draw TestOp(OvalNb("apple"),"contient",OvalNb("a"))  
        ;  
endfig;
```

399 modulo 19

```
beginfig(16);  
draw OvalOp(OvalNb("399"),"modulo",OvalNb("19"));  
endfig;
```

arrondi de $\sqrt{3}$

```
beginfig(17);  
draw OvalOp("arrondi de",OvalNb("$\sqrt{3}$"));  
endfig;
```

abs de $\sqrt{3}$

```
beginfig(18);  
draw OvalOp(RecMenuOp("abs"),"de",OvalNb("$\sqrt{3}$"));  
endfig;
```

Variables

| | |
|---|---|
| ma variable | <pre>beginfig(1); draw OvalVar("ma variable"); endfig;</pre> |
| mettre <input type="text" value="ma variable"/> à <input type="text" value="0"/> | <pre>beginfig(2); draw MettreVar("ma variable",OvalNb("0")); endfig;</pre> |
| ajouter <input type="text" value="1"/> à <input type="text" value="ma variable"/> | <pre>beginfig(3); draw AjouterVar(OvalNb("1"),"ma variable"); endfig;</pre> |
| montrer la variable <input type="text" value="ma variable"/> | <pre>beginfig(4); draw MontrerVar("ma variable"); endfig;</pre> |
| cacher la variable <input type="text" value="ma variable"/> | <pre>beginfig(5); draw CacherVar("ma variable"); endfig;</pre> |

Listes

| | |
|--|---|
| mes numéros | <pre>beginfig(1); draw OvalList("mes numéros"); endfig;</pre> |
| ajouter chose à mes numéros | <pre>beginfig(2); draw AjouterListe("chose","mes numéros"); endfig;</pre> |
| supprimer l'élément 1 de mes numéros | <pre>beginfig(3); draw SupprimerListe("1","mes numéros"); endfig;</pre> |
| supprimer tous les éléments de la liste mes numéros | <pre>beginfig(4); draw SupprimerListeAll(" mes numéros"); endfig;</pre> |
| insérer chose en position 1 de mes numéros | <pre>beginfig(5); draw InsérerListe("chose","1 ","mes numéros"); endfig;</pre> |
| remplacer l'élément 1 de la liste mes numéros par chose | <pre>beginfig(6); draw RemplacerListe("1","mes numéros","chose "); endfig;</pre> |
| élément 1 de mes numéros | <pre>beginfig(7); draw OvalList("élément",OvalNb("1"),"de", RecMenuList("mes numéros")); endfig;</pre> |
| position de chose dans mes numéros | <pre>beginfig(8); draw OvalList("position de",OvalNb("chose"),"dans",RecMenuList(" mes numéros")); endfig;</pre> |
| longueur de mes numéros | <pre>beginfig(9); draw OvalList("longueur de",RecMenuList("mes numéros")); endfig;</pre> |
| mes numéros contient chose ? | <pre>beginfig(10); draw TestList(RecMenuList("mes numéros "),"contient",OvalNb("chose"),"?"); endfig;</pre> |
| montrer la liste mes numéros | <pre>beginfig(11); draw MontrerListe("mes numéros"); endfig;</pre> |
| cacher la liste mes numéros | <pre>beginfig(12); draw CacherListe("mes numéros"); endfig;</pre> |

Bloc

définir Triangle longueur

```
beginfig(1);  
  draw NouveauBloc("Triangle",OvalBloc("longueur"))  
  ;  
endfig;
```

Triangle réponse

```
beginfig(2);  
  draw Bloc("Triangle",OvalCap("réponse"));  
endfig;
```

Personnalisation

supprimer le fichier Exemple.png

```
beginfig(1);  
  draw BlocUser((44/256,120/256,195/256))("supprimer le fichier",RecText("Exemple.png"));  
endfig;
```