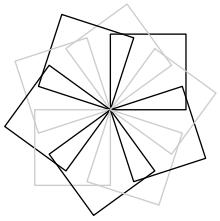
Scratch et METAPOST



mp-scratch

Version 0.7

C.Poulain
chrpoulain@gmail.com

Juillet 2017

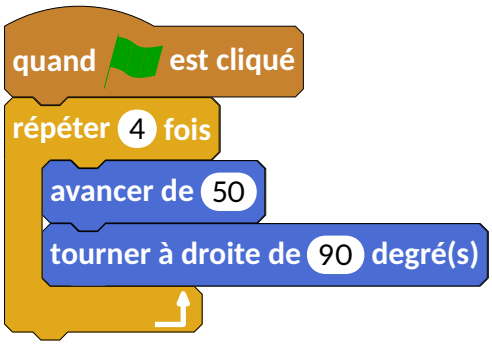
Résumé

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

Avant propos

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 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 « 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[2].




```
input mp-scratch;

beginfig(1);
  draw Drapeau;
  draw Repeter("4");
  draw Avancer("50");
  draw Tournerd("90");
  draw FinBlocRepeter;
endfig;

end
```

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].
Pour l'utilisation de mp-scratch, il sera nécessaire de veiller à installer, si ce n'est pas fait, la fonte Carlito[5]. Et c'est tout ![6]

1 Utilisation

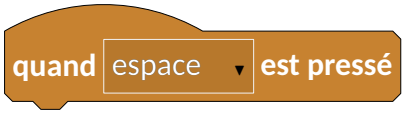
Afin de faciliter la « transcription » des algorithmes créés sous Scratch sous METAPOST, j'ai fait le choix de respecter *au maximum et au mieux* la syntaxe des briques Scratch. Cela ne facilitera pas

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 bclogo : le drapeau vert a été créé à partir des sources de son package et notamment la construction, en METAPOST, de ses drapeaux.
3. Tous les contributeurs sont donc les bienvenus pour développer le package.
4. Arborescence locale de préférence, par exemple dans /home/christophe/texmf/metapost/ sous Linux.
5. `http://www.tug.dk/FontCatalogue/carlito/` pour un exemple. Le choix reste personnalisable évidemment mais Thierry Pasquier, à juste titre, m'a préconisé d'utiliser une fonte sans serif.
6. C'est une grande amélioration par rapport à une version précédente de mp-scratch où il était nécessaire de modifier les fichiers sources fournis…

l'internationalisation du package – Soyons fous ! – mais permet de « produire » très rapidement les figures correspondantes aux différents algorithmes.

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 :

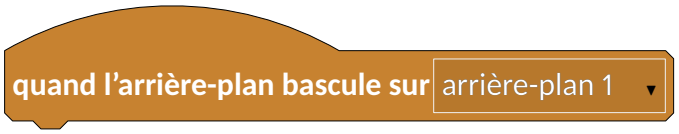

```
beginfig(1);
  draw Drapeau;
endfig;
```




```
beginfig(2);
  draw QPresse("espace");
endfig;
```


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


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




```
beginfig(5);
  draw QBasculeAR("arrière-plan 1");
endfig;
```


```
beginfig(6);
  draw QVolumeSup("volume sonore","10");
endfig;
```

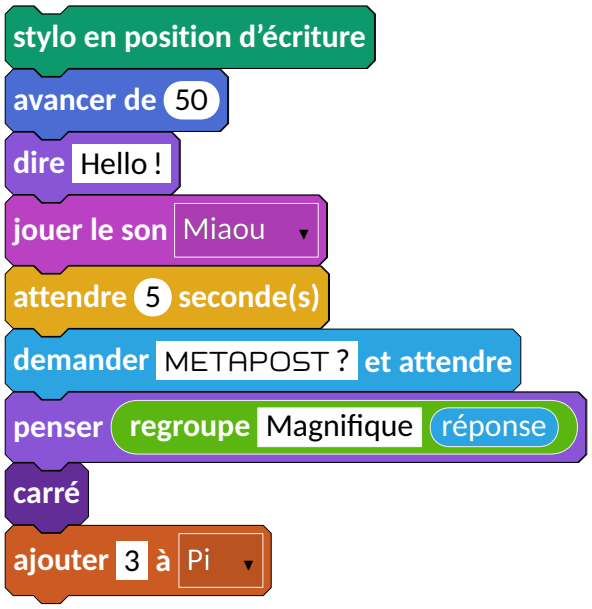

```
beginfig(7);
  draw QRecevoirMessage("message1");
endfig;
```

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


```
beginfig(8);
  draw EnvoyerMessage("message1");
endfig;
```


```
beginfig(9);
  draw EnvoyerMessageA("message1");
endfig;
```

Une fois le script démarré, voici un exemple des blocs produits :



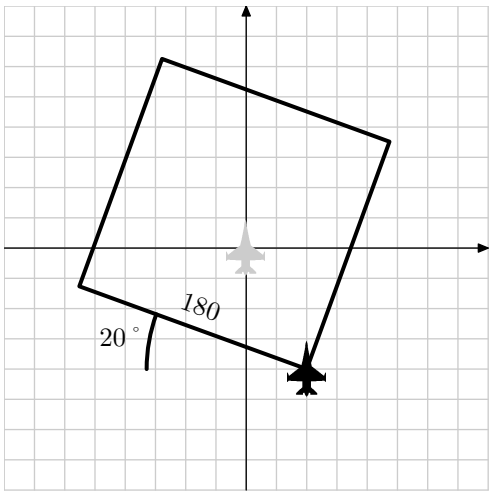

```
beginfig(1);
  draw PoserStylo;
  draw Avancer("50");
  draw Dire("Hello␣!");
  draw Jouer("Miaou");
  draw Attendre("5");
  draw Demander("\MP\␣?");
  draw Penser(OvalOp("regroupe",
      RecText("Magnifique"),OvalCap("
      réponse")));
  draw Bloc("carré");
  draw AjouterListe("3","Pi");
endfig;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

On remarque que la syntaxe est très proche du vocabulaire utilisé par Scratch ; donc très peu de nouvelles commandes à apprendre et que les couleurs [7] sont celles utilisées par Scratch [8].

Premier pas

Pour les codes METAPOST suivants, on omettra volontairement **input** mp−scratch et **end**.

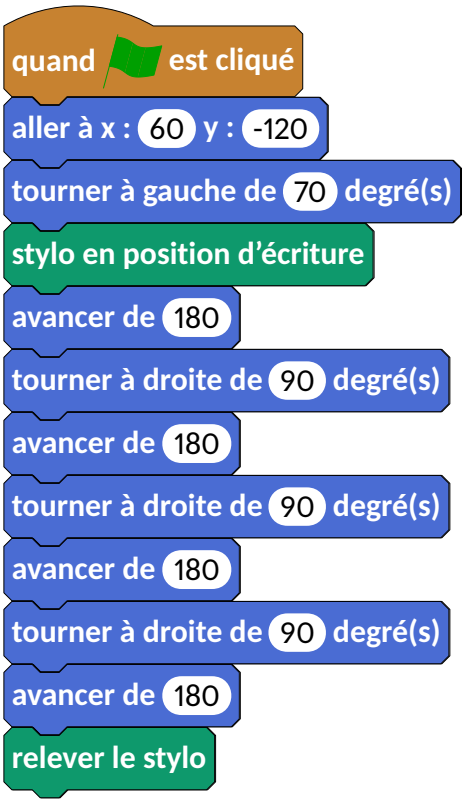
Scratch permet d'effectuer des constructions géométriques à l'aide, principalement, des catégories Mouvement et Stylo. Par exemple, pour que le lutin puisse effectuer le tracé ci-dessous,



on peut proposer l'algorithme suivant :

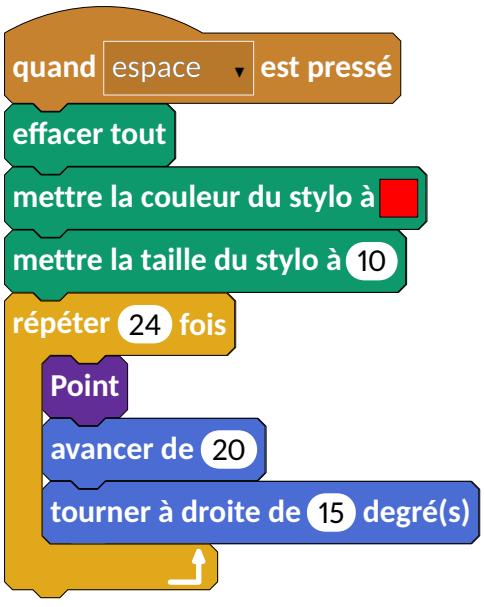
7. Cela reste, bien évidemment, paramétrable. Les paramètres disponibles pour personnaliser les couleurs sont colMouv, colAp, colSon, colStylo, colEvenements, colControle, colCapteur, colBloc, colVar, colList.

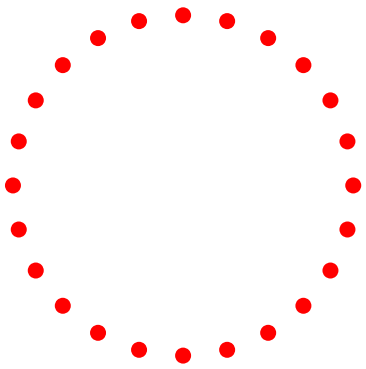
8. grâce à gcolor2.




```
beginfig(1);
  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;
```

Evidemment, on peut avoir besoin des catégories Contrôle et Ajouter blocs pour obtenir des figures telles que celle ci-dessous.



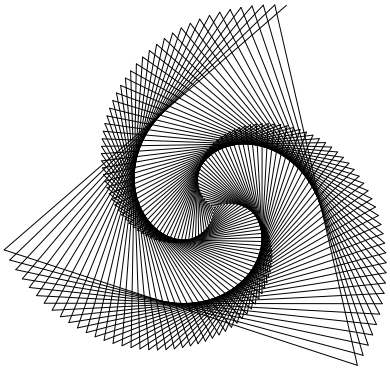


```
beginfig(1);
  draw QPresse("espace");
  draw Effacer;
  draw MettreCouleur(1,0,0);
  draw MettreTS("10");
  draw Repeter("24");
  draw Bloc("Point");
  draw Avancer("20");
  draw Tournerd("15");
  draw FinBlocRepeter;
endfig;
```

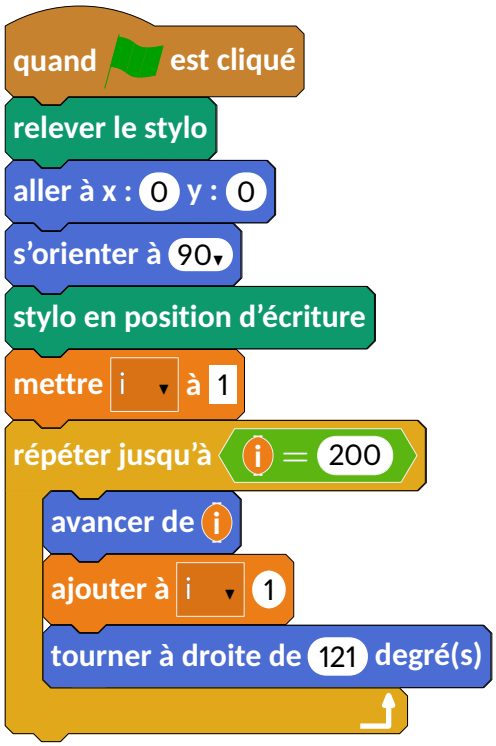
On voit apparaître le bloc Point dont la définition est la suivante :


```
beginfig(1);
  draw NouveauBloc("Point");
  draw PoserStylo;
  draw releverStylo;
endfig;
```

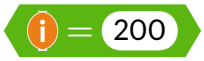
Dans des constructions un peu plus particulières, on peut utiliser des élements des catégories Données et Opérateurs :



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 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 

ainsi que sa création

TestOp(OvalVar("i"),"$\bm{=}$",OvalNb("200")).

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

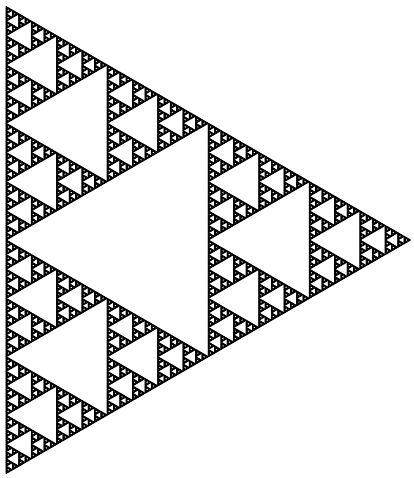
— d'ovale 

OvalOp("regroupe",RecText("Hello"),RecText("world"))

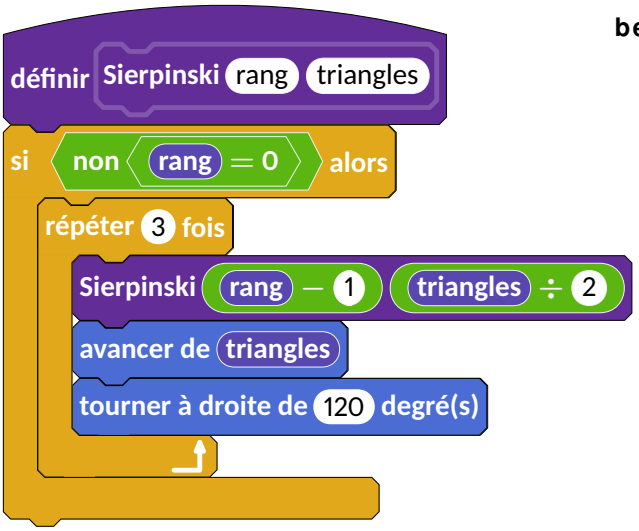
— ou de « diamant » 

TestOp("i","$\bm{<}$",OvalNb("10"))

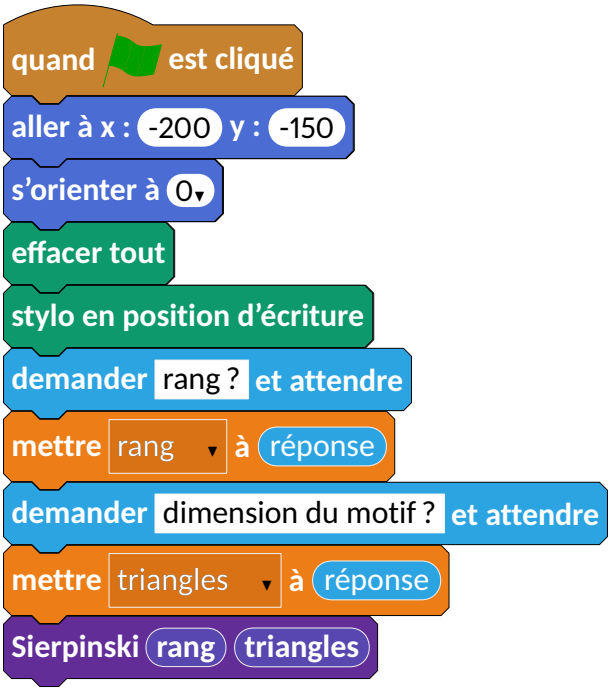
Si nous continuons d'explorer les constructions géométriques, on peut se pencher sur le triangle de Sierpinski :



qui nécessite la déclaration de deux « branches » d'algorithme :

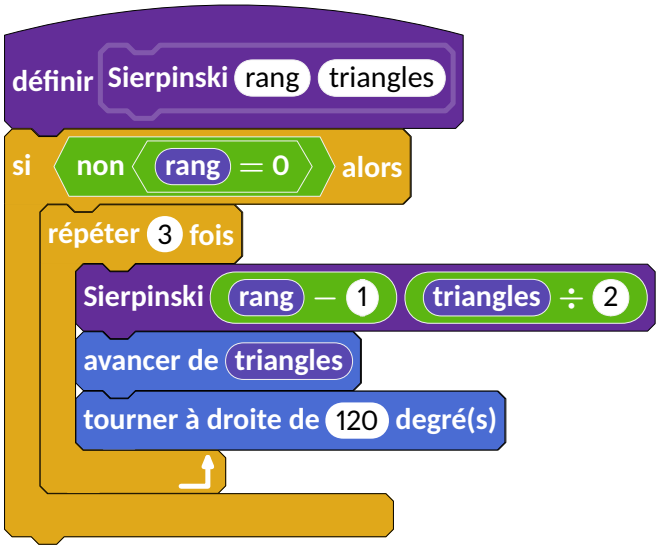



```
beginfig(1);
  draw NouveauBloc("Sierpinski",OvalNb("
      rang"),OvalNb("triangles"));
  picture Tee;
  Tee=TestOp(OvalBloc("rang"),"$\bm{=}$",
      "0");
  draw Si(TestOp("non",Tee));
  draw Repeter("3");
  picture BB[];
  BB1=OvalOp(OvalBloc("rang"),"$\bm{-}$",
      OvalNb("1"));
  BB2=OvalOp(OvalBloc("triangles"),"$\bm
      {\div}$",OvalNb("2"));
  draw Bloc("Sierpinski",BB1,BB2);
  draw Avancer(OvalBloc("triangles"));
  draw Tournerd("120");
  draw FinBlocRepeter;
  draw FinBlocSi;
endfig;
```




```
beginfig(2);
  draw Drapeau;
  draw Aller("-200","-150");
  draw Orienter("0");
  draw Effacer;
  draw PoserStylo;
  draw Demander(RecText("rang ?"));
  draw MettreVar("rang",OvalCap("réponse"
      ));
  draw Demander(RecText("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.




```
beginfig(1);
  draw NouveauBloc("Sierpinski",OvalNb("rang"),OvalNb("triangles"));
  picture Tee;
  Tee=TestOp(OvalBloc("rang"),"$\bm{=}$","0");
  draw Si(TestOp("non",Tee));
  draw Repeter("3");
  picture BB[];
```

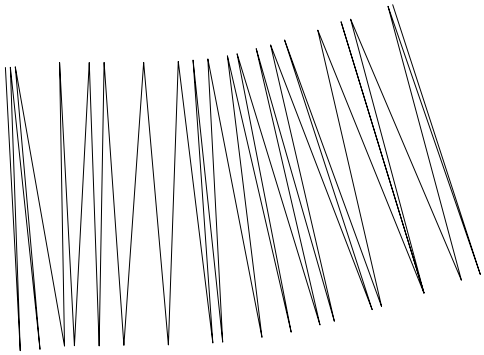
```
  BB1=OvalOp(OvalBloc("rang"),"$\bm{-}$",OvalNb("1"));
  BB2=OvalOp(OvalBloc("triangles"),"$\bm{\div}$",OvalNb("2"));
  draw Bloc("Sierpinski",BB1,BB2);
  draw Avancer(OvalBloc("triangles"));
  draw Tournerd("120");
  draw FinBlocRepeter;
  draw FinBlocSi;

  Deplacera(10cm,0);

  draw Drapeau;
  draw Aller("-200","-150");
  draw Orienter("0");
  draw Effacer;
  draw PoserStylo;
  draw Demander(RecText("rang ?"));
  draw MettreVar("rang",OvalCap("réponse"));
  draw Demander(RecText("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.

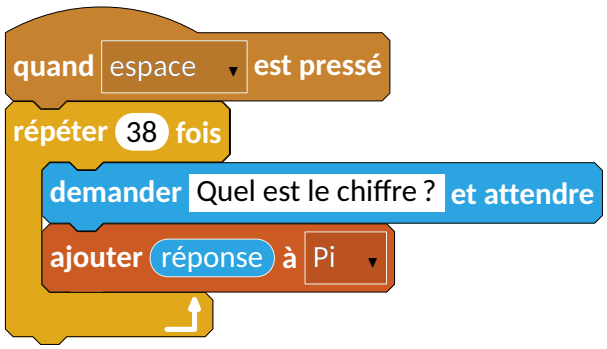
Tableau de Morellet C'est un tableau tracé à partir des trente-huit (38) premiers chiffres constituant le nombre π. Il peut être nécessaire d'indiquer d'utiliser :

— Créer une variable pour créer la variable varpi ;

— Créer une liste pour créer la liste Pi.

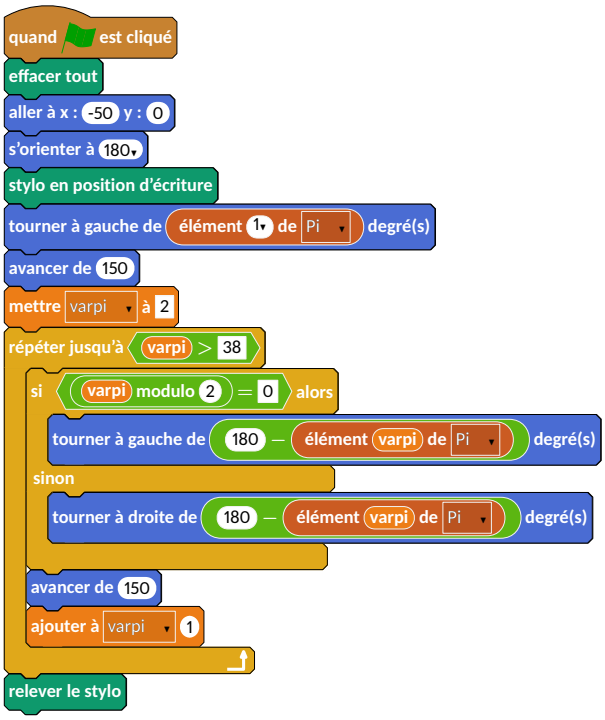


Pour compléter la liste Pi, on peut utiliser « le script » suivant :




```
beginfig(1);
  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 :




```
beginfig(1)%François Morellet – Oeuvre Pi
    piquant, 1=1°, 38 premiers chiffres
  draw Drapeau;
  draw Effacer;
  draw Aller("-50","0");
  draw Orienter("180");
  draw PoserStylo;
  draw Tournerg(OvalListMulti("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=OvalListMulti("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;
```

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,

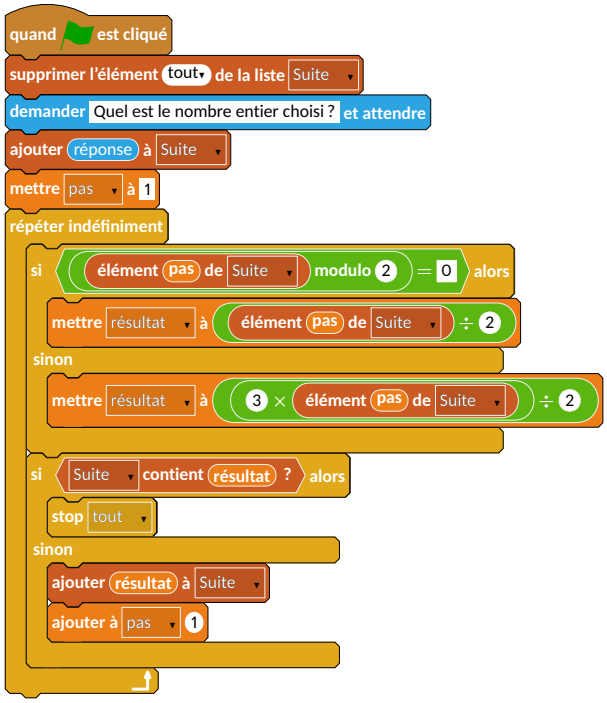
le nombre ainsi obtenu remplaçant le nombre entier de départ…

Indépendemment de la conjecture associée à cette suite, nous pouvons étudier « la longueur » de la suite associé à un nombre entier donné.

Voici un exemple de son implantation sous Scratch. Pour cela, on a créé :

— deux variables résultat et pas ;

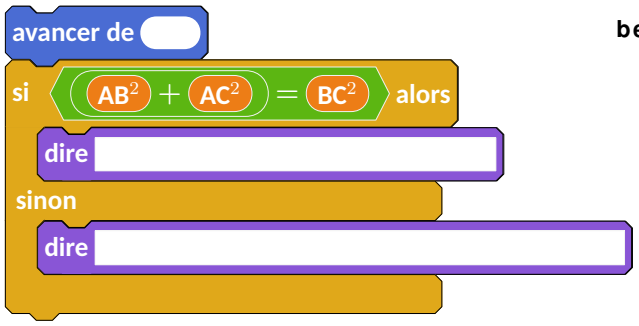
— une liste Suite qui contiendra les éléments de la suite de Syracuse associée au nombre entier choisi.




```
beginfig(1);
  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;
```

Le coin du prof !

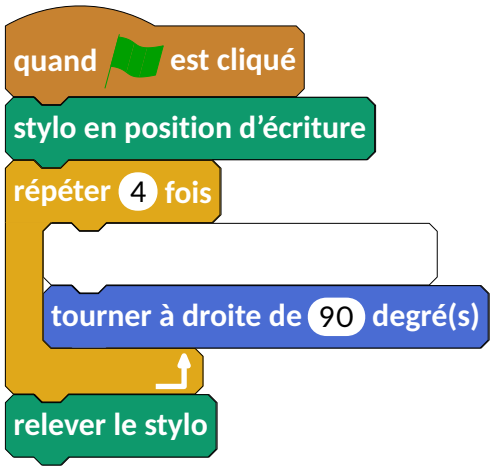
Cette partie recense des possibilités dont ne dispose pas, à ma connaissance, Scratch. Elles ont été rendues nécessaires par l'enseignement « débranché » de l'algorithmique. En plus de pouvoir produire des blocs avec des arguments vides,




```
beginfig(1);
  draw Avancer("\phantom{100}");
  picture BB[];
  BB1=OvalOp(OvalVar("AB$^2$"),"$\bm{+}$",
      OvalVar("AC$^2$"));
  draw Si(TestOp(BB1,"$\bm{=}$",OvalVar("BC
      $^2$")));
  draw Dire(RecText("\phantom{le␣triangle␣$
      ABC$␣est␣rectangle␣en␣$A$}"));
  draw Sinon;
  draw Dire(RecText("\phantom{le␣triangle␣$
      ABC$␣n'est␣pas␣un␣triangle␣rectangle}")
      );
  draw FinBlocSi;
endfig;
```

les dispositifs supplémentaires sont les suivants :

— Une boite puzzle « vide » afin de fournir des algorithmes à compléter :



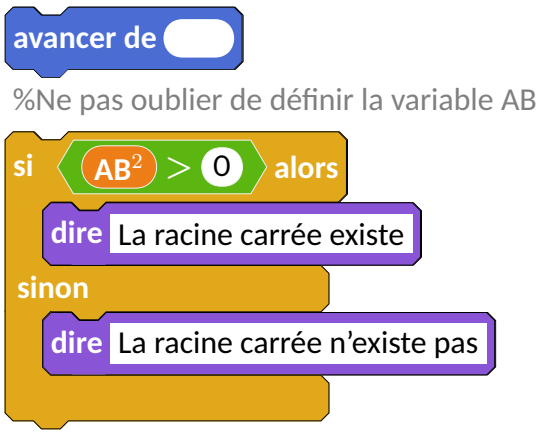

```
beginfig(1);
  draw Drapeau;
  draw PoserStylo;
  draw Repeter("4");
  draw CommandeVide("5");%5cm
  draw Tournerd("90");
  draw FinBlocRepeter;
  draw ReleverStylo;
endfig;
```

— Une ligne « en pointillés » permettant d'alléger l'écriture de certaines parties d'algorithmes :


```
beginfig(1);
  draw Avancer("50");
  draw LignePointilles;
  draw Jouer("miaou");
endfig;
```

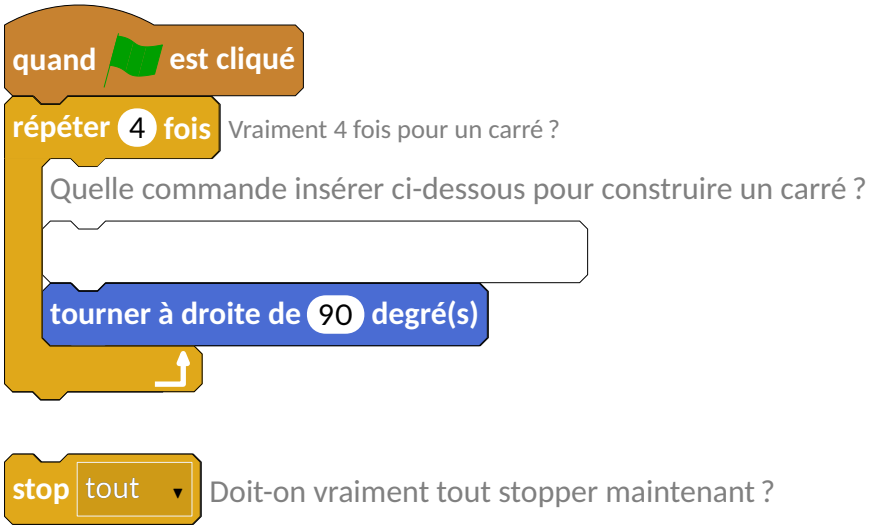
— L'inclusion de commentaires :

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



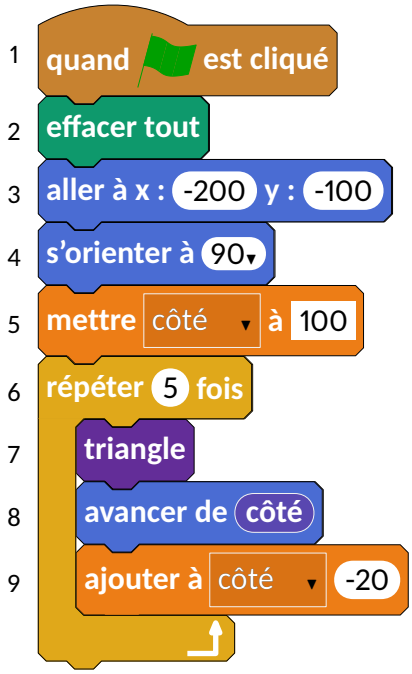

```
beginfig(1);
  draw Avancer("\phantom{100}");
  draw Commentaires("\%Ne␣pas␣
      oublier␣de␣définir␣la␣variable␣
      AB");
  draw Si(TestOp(OvalVar("AB$^2$"),"
      $\bm{>}$",OvalNb("0")));
  draw Dire("La␣racine␣carrée␣existe
      ");
  draw Sinon;
  draw Dire("La␣racine␣carrée␣n'
      existe␣pas");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

— en bout de bloc « puzzle » :




```
beginfig(1);
  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;
```

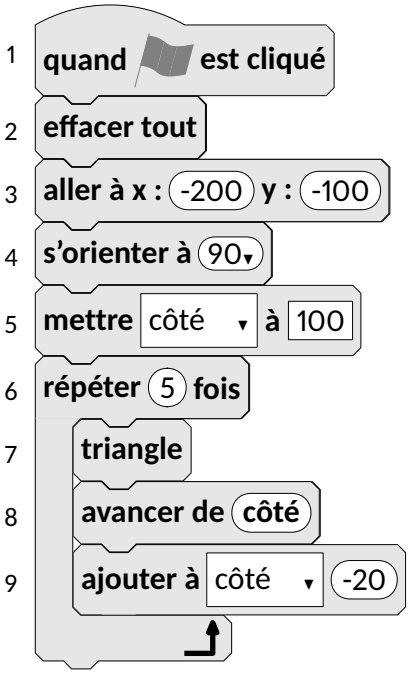
— La numérotation des lignes [9] d'un algorithme :




```
beginfig(1);
  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;
```

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

— La disparition des couleurs [10] pour une impression visuellement meilleure pour les élèves :

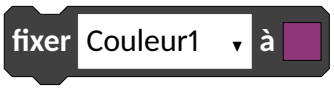



```
beginfig(1);
  coefprint:=0.9;
  print:=true;
  NumeroteLignes:=true;
  draw Drapeau;
  draw Effacer;
  draw Aller("-200","-100");
  draw Orienter("90");
  draw MettreVar("côté","100");
  draw Repeter("5");
  draw Bloc("triangle");
  draw Avancer(OvalBloc("côté"));
  draw AjouterVar("côté","-20");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

On peut également être tenter de se détacher de Scratch et de son installation liée à Adobe Air...
Ainsi, on peut penser à tester :

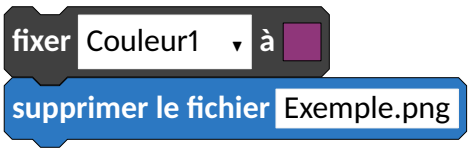
— Snap, utilisable en ligne à l'adresse `http://snap.berkeley.edu/snapsource/snap.html#`
— Phratch, utilisable en local et disponible à l'adresse `http://www.phratch.com/`.

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




```
beginfig(1);
  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




```
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"));
```

10. Suggestion proposée lors du stage LaTeX de Dunkerque (Juin 2017) et par le package Scratch en TikZ de C.Télléchéa

Historique

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.

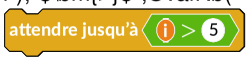
15/01/2017 Version 0.3 - Modification du code. Conception de la documentation.

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.

- **draw** AttendreJ(TestOp(OvalVar("i"),"$\bm{>}$",OvalNb("5")));



- **draw Stop**("ce_script");

- **draw** CommencerClone;

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

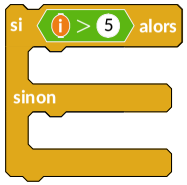
- **draw** SupprimerClone;

-
```
draw Repeter("10");
draw LigneVide;
draw FinBlocRepeter;
```

-
```
draw RepeterI;
draw LigneVide;
draw FinBlocRepeter;
```

-
```
draw Si1(TestOp(OvalVar("i"),"
     $\bm{>}$",OvalNb("5")));
draw LigneVide;
draw FinBlocSi;
```

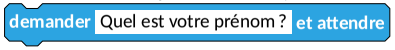
-
```
draw Si(TestOp(OvalVar("i"),"
     $\bm{>}$",OvalNb("5")));
draw LigneVide;
draw Sinon;
draw LigneVide;
draw FinBlocSi;
```



-
```
draw RepeterJ(TestOp(
     OvalVar("i"),"$\bm
     {>}$",OvalNb("5")));
draw LigneVide;
draw FinBlocRepeter;
```

Capteurs

- **draw** Demander("Quel_est_votre_pr\'enom_?");

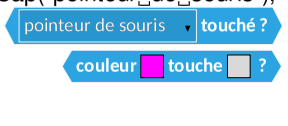


- **draw** ActiverVideo("activ\'e");

- **draw** TransparenceVideo("10");

- **draw** ReinitChrono;

- **draw** TestCapMulti(RecMenuCap("pointeur_de_souris"),"touch\'e_?");

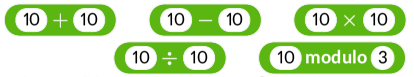


- **draw** OvalCap("r\'eponse");

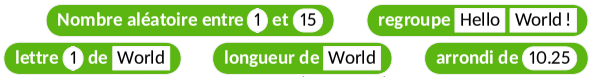
- **draw** OvalCapMulti("vid\'eo",RecMenuCap("mouvement"),"sur",RecMenuCap("ce_lutin"));

Opérateurs

- **draw** OvalOp(OvalNb("10"),"$\bm{+}$",OvalNb("10"));

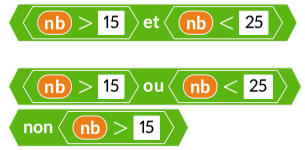


- **draw** OvalOp("Nombre_al\'eatoire_entre",OvalNb("1"),"et",OvalNb("15"));



- **draw** TestOp(OvalVar("nb"),"$\bm{<}$",RecText("15"));

-
```
picture BB[];
BB1=TestOp(OvalVar("nb"),"
     $\bm{>}$",RecText("15
     "));
BB2=TestOp(OvalVar("nb"),"
     $\bm{<}$",RecText("25
     "));
draw TestOp(BB1,"et",BB2);
```



Bloc

- **draw** NouveauBloc("Pentagone",OvalBloc("cote"));

- **draw** Bloc("Pentagone",OvalNb("cote"));

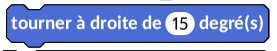
Aide-mémoire mp-scratch

Christophe Poulain

Mouvements

- **draw** Avancer("10");

- **draw** Tournerd("90");



- **draw** Tournerg("90")

- **draw** Orienter("90");

- **draw** Orienterdirection("pointeur_de_souris");

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

- **draw** Allera("pointeur_de_souris");

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

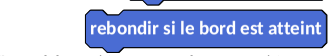
- **draw** Ajouter("10","x");

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

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

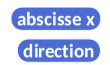
- **draw** Mettre("10","y");

- **draw** Rebondir;



- **draw** FixerSensRotation("position_\'a_gauche_ou_\'a_droite"';

- **draw** OvalMouv("abscisse_x");



Apparence

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

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

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

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

- **draw** Montrer; **draw** Cacher;

- **draw** Basculer("costume2");

- **draw** CostumeSuivant;

- **draw** BasculerAR("arriere−plan2");

- **draw** AjouterEffet("couleur","10");

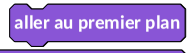
- **draw** MettreEffet("couleur","10");

- **draw** AnnulerEffet;

- **draw** AjouterTaille("10");

- **draw** MettreA("75");

- **draw** AllerPPlan;



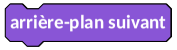
- **draw** DeplacerAP("1");

- **draw** OvalApp("costume␣\#");

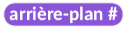
Quand « la scène » est sélectionnée, on dispose de :

- **draw** BasculerARA("\opApMenu{arriere−plan2}");

- **draw** ARSuivant;



- **draw** OvalApp("arri\'ere−plan␣\#");



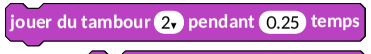
Sons

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

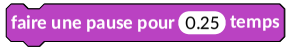
- **draw** JouerT("miaou");

- **draw** ArreterSon;

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



- **draw** Pause("0.25");



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

- **draw** ChoisirInstrument("17");

- **draw** AjouterVol("−10");

- **draw** MettreVol("15");

- **draw** AjouterTempo("20");

- **draw** MettreTempo("15");

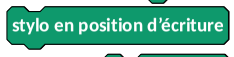
- **draw** OvalSon("volume");

Stylo

- **draw** Effacer;

- **draw** Estampiller;

- **draw** PoserStylo;



- **draw** ReleverStylo;

- **draw** MettreCouleur(1,0,1);

- **draw** AjouterCS("10");

- **draw** MettreCS(25);

- **draw** AjouterIS("10");

- **draw** MettreIS(15);

- **draw** AjouterTS(12);

- **draw** MettreTS("10");

Données

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

- **draw** AjouterVar("pi","10");

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

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

- **draw** AjouterListe("thing","Listepi");

- **draw** SupprimerListe("thing","Listepi");

- **draw** InsererListe("thing",1,"Listepi");

- **draw** RemplacerListe("3","Listepi","4");

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

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

- **draw** OvalVar("c\^ot\'e");

- **draw** OvalList("Pythagore");

- **draw** OvalListMulti("\' el \' ement",OvalNb("1"),"de", RecMenuList("Pythagore"));

- **draw** OvalListMulti("longueur␣de",RecMenuList("Pythagore"));

- **draw** TestList(RecMenuList("Pythagore"),"contient", RecText("thing"),"?");

Évènements

- **draw** Drapeau;

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

- **draw** QLutinPresse;

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

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

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

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

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

Quand « la scène » est sélectionnée, on dispose de :

- **draw** QScenePressee;

Contrôle

- **draw** Attendre("10");