

Mercredi 25 novembre 2015
Journée régionale de l'APMEP

Introduction à l'algorithmique dans le
cas général et dans le cadre de la
réforme du collège. Exercices et
applications.

François PETIT
Enseignant au collège Gérard Philipe (PESSAC)
Membre du groupe IREM Algorithmique

Plan de la présentation

- **Introduction**
- **Notion algorithmique de base**
- **Les notions du programme de collège**
- **Exemple de mise en œuvre à l'aide d'outils logiciels (algorithmique, programmation)**
- **Quelques illustrations**

Première partie

Introduction

Qu'est-ce qu'un algorithme ?

Un algorithme décrit un *enchaînement* d'opérations permettant, en un temps *fini*, de *résoudre* un problème donné.

3 8 4	6 1 9	← données
x 2 3	x 3 2	
<hr/>	<hr/>	
1 1 5 2	1 2 3 8	
7 6 8	1 8 5 7	← résultat
8 8 3 2	1 9 8 0 8	

Un algorithme produit ainsi un résultat dépendant des données du problème.

Un premier exemple

Résolution d'une équation du 2nd degré de la forme

$$ax^2 + bx + c = 0$$

(données du problème : a, b et c)

1. je calcule le discriminant : $\Delta = b^2 - 4ac$

2. trois cas à considérer :

α . $\Delta < 0$: pas de solution

β . $\Delta = 0$: une solution double, $x = -b / 2a$

χ . $\Delta > 0$: deux solutions,

$$x_1 = (-b + \text{racine}(\Delta)) / 2a$$

$$x_2 = (-b - \text{racine}(\Delta)) / 2a$$

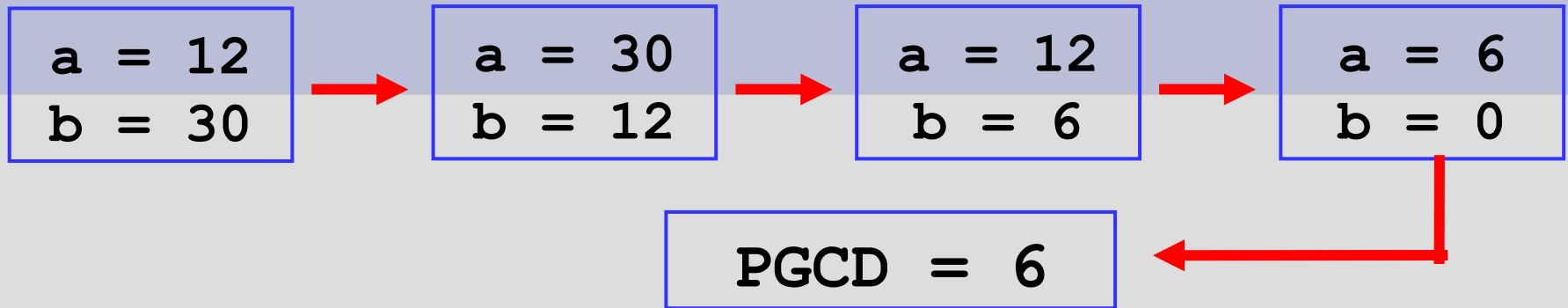
Un deuxième exemple

Algorithme d'Euclide : déterminer le PGCD de deux entiers positifs **a** et **b** (données du problème)

Basé sur les propriétés suivantes :

- $\text{PGCD}(a, 0) = a$
- si $b \neq 0$, $\text{PGCD}(a, b) = \text{PGCD}(b, a \bmod b)$

Un deuxième exemple



1. si $b = 0$, le PGCD vaut a
2. dans le cas contraire, remplacer a par b , et b par $(a \bmod b)$, puis recommencer en 1.

Cette séquence se répète tant que l'égalité $b = 0$ ne se produit pas...

Caractéristiques d'un algorithme

L'expression des algorithmes nécessite l'utilisation d'un **langage** (convention d'écriture) rigoureux, structuré, compréhensible et non ambigu.

Il n'existe pas de langage normalisé à ce niveau, mais les conventions adoptées sont suffisamment similaires pour être comprises de tous...

Caractéristiques d'un algorithme

On s'attend à ce qu'un algorithme soit **correct** (le résultat obtenu est correct) et se termine **en un temps fini**.

*Ces deux propriétés, **correction** et **terminaison**, sont souvent « prouvables » mathématiquement...*

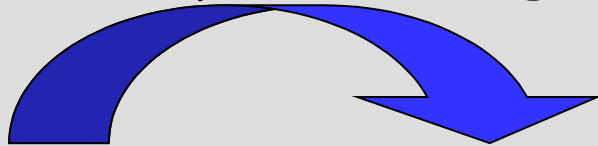
... sans négliger le **temps de calcul** nécessaire...

*On parle de **complexité** de l'algorithme...*

Algorithmique et programmation

Un algorithme peut être *transcrit* dans un langage de programmation, afin d'être *exécuté* par une machine.

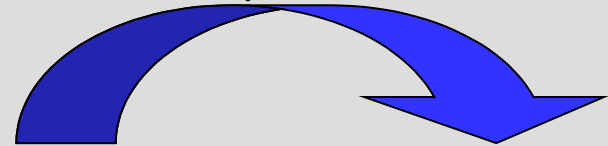
transcription (codage)



Algorithme

Programme

compilation



Programme
exécutable

Algorithmique et programmation

Tout algorithme doit être *universel*, c'est-à-dire être indépendant du langage de programmation dans lequel il sera transcrit.

(cf. l'algorithme d'Euclide, III^e s. av. J.C. !...)

Remarque fondamentale...

On ne peut concevoir un algorithme pour résoudre un problème donné que si l'on connaît soi-même une méthode permettant de le résoudre !...

Ainsi, au collège, l'enseignement de l'algorithmique se doit d'accompagner l'enseignement des mathématiques...

Deuxième partie

Les notions du programme
du collège

Contributions essentielles au DOMAINE 4 du socle commun

Mathématiques Physique-Chimie Sciences de la
vie et de la Terre Technologie

S'exercer au raisonnement inductif et déductif par la résolution de problèmes, les démarches d'essais-erreurs, de conjecture et de validation ; s'exercer au raisonnement logique par le calcul numérique ou littéral, la géométrie et **l'algorithmique**. S'initier à la démonstration mathématique.

Contributions essentielles au DOMAINE 3 du socle commun

- organisation et gestion de données , fonctions ;
- nombres et calculs ;
- géométrie ;
- grandeurs et mesures ;
- **algorithmique et programmation.**

Contributions essentielles au DOMAINE 3 du socle commun

**Thème A -: ORGANISATION ET GESTION DE DONNÉES,
FONCTIONS**

Attendu de fin de cycle

- Programmer des algorithmes simples en lien avec les fonctions
- Notion de variable, en lien avec l'algorithmique et le calcul littéral

Contributions essentielles au DOMAINE 3 du socle commun

Thème B – NOMBRES ET CALCULS

Attendu de fin de cycle

pour illustrer la notion de variable mathématique

- Modéliser un problème en vue de sa résolution
- Utiliser un tableur,
- programmer un algorithme,

Contributions essentielles au DOMAINE 3 du socle commun

Thème E – ALGORITHMIQUE ET PROGRAMMATION

Attendu de fin de cycle

- Analyser un problème complexe,
- Définir des sous-problèmes,
- Définir des étapes de résolution
- Reconnaître des configurations récurrentes,
- Mettre en évidence des interactions
- Traduire un algorithme dans un langage de programmation

Contributions essentielles au DOMAINE 3 du socle commun

Thème E – ALGORITHMIQUE ET PROGRAMMATION

Attendu de fin de cycle

- Notion de variable informatique
- Séquences d'instructions, boucles, instructions conditionnelles
- Gestion d'événements déclenchés par le clavier, la souris, etc.
- Déclenchement de plusieurs séries d'instructions en parallèle
- [4e-3e] Échange de messages entre objets, événements liés au déplacement d'un objet, clonage d'un objet

Troisième partie

Éléments de base de l'algorithmique

- La notion de variable, l'opération d'affectation
manipulation des données (calculs)
- Les opérations d'entrée-sortie
*récupération des données,
affichage des résultats*

Éléments de base de l'algorithmique

- Les structures conditionnelles

traitements alternatifs selon une condition

- Les structures répétitives

répétition d'une séquence d'opérations

La notion de variable

Ces objets sont appelés des **variables**. Chaque variable possède :

- un nom (**identificateur**), permettant de la désigner sans ambiguïté dans l'algorithme,
- une **valeur** (on parle de contenu de la variable),
- un **type** (entier, caractère, ...), déterminant l'ensemble des valeurs que peut prendre la variable et l'ensemble des opérations qui lui sont applicables.

La notion de variable

Représentation graphique usuelle :

y

384

Une variable *entière* de nom *Y* ayant pour valeur *384*...

Une variable ne peut avoir qu'une seule valeur.

Ainsi, si l'on modifie la valeur d'une variable, son ancienne valeur est perdue (« écrasée » par la nouvelle).

La notion de variable

Toutes les variables manipulées par un algorithme doivent être **déclarées**, ce qui permet de préciser leur **nom** et leur **type**.

```
Algorithme Exemple1
```

```
# cet algorithme illustre la déclaration de variables  
variables
```

```
    Y : entier naturel
```

```
    réponse : caractère
```

```
début
```

```
    ...
```

```
fin
```


L'opération d'affectation

L'opération d'affectation permet de ***donner une valeur*** à une variable (l'ancienne valeur, si elle existait, est remplacée par la nouvelle valeur).

Format général (syntaxe) :

<nom_variable> ← <expression>

L'opération d'affectation

- l'expression est évaluée, puis sa valeur est rangée dans la variable
- la variable et l'expression doivent avoir des types compatibles

A ← 14

la variable A reçoit la valeur 14

B ← A+5

la variable B reçoit la valeur 19

Les opérations d'entrée-sortie

Les opérations d'entrée-sortie permettent de communiquer avec l'utilisateur : ***recupérer la valeur*** de variables (e. g. les données du problème) ou ***afficher la valeur*** d'expressions (e. g. les résultats de l'algorithme).

Les opérations d'entrée-sortie

Format général (syntaxe) :

`Entrer (<liste_de_variables>)`

`Afficher (<liste_d_expressions>)`

`Entrer (A, B)`

`Afficher ("La somme vaut", A+B)`

Les structures conditionnelles

La structure ***Si-Alors-Sinon*** permet de réaliser telle ou telle séquence d'opérations ***selon la valeur d'une condition*** (expression logique, dont l'évaluation donne la valeur VRAI ou FAUX).

Les structures conditionnelles

Format général (syntaxe) :

Si (<condition>)

Alors ...

Sinon ...

Fin_Si

ou

Si (<condition>)

Alors ...

Fin_Si

Les structures conditionnelles

Exemple.

Calcul de la valeur absolue d'un entier relatif

```
Algorithme ValeurAbsolue
# cet algorithme calcule la valeur absolue d'un
# entier relatif
variables
  X : entier relatif
  absX : entier naturel
début
  Entrer (X)
  Si ( X ≥ 0 )
  Alors absX ← X
  Sinon absX ← -X
  Fin_Si
  Afficher (absX)
fin
```

Les structures répétitives (boucle Pour)

La boucle *Pour* permet de ***répéter*** une séquence d'opérations en faisant ***varier automatiquement*** une variable (appelée ***variable de boucle***). Cette variable de boucle prendra successivement toutes les valeurs d'un ***intervalle*** fixé.

Les structures répétitives (boucle Pour)

Format général (syntaxe) :

```
Pour <variable> de <expr1> à <expr2>
```

```
Faire ...
```

```
Fin_Pour
```

Les structures répétitives (boucle Pour)

Exemple.

Affichage de la « table de 8 »

Affichage obtenu

```
Algorithme TableDe8
# cet algorithme affiche la table de
# multiplication par 8
variables
    I : entier naturel
début
    Pour I de 1 à 10
        Faire
            Afficher ( I, " x ", 8, " = ", 8*I )
        Fin_Pour
    fin
```

```
1 x 8 = 8
2 x 8 = 16
3 x 8 = 24
4 x 8 = 32
5 x 8 = 40
6 x 8 = 48
7 x 8 = 56
8 x 8 = 64
9 x 8 = 72
10 x 8 = 80
```

Les structures répétitives (boucle Tantque)

La boucle ***Tantque*** permet de ***répéter*** une séquence d'opérations tant qu'une condition est vérifiée (c'est-à-dire tant que son évaluation retourne la valeur VRAI).

Les structures répétitives (boucle Tantque)

Format général (syntaxe) :

Tantque (<condition>)

Faire ...

Fin_Tantque

Attention : mal maîtrisée, cette structure peut conduire à un algorithme qui « boucle » indéfiniment (si la condition ne retourne jamais la valeur FAUX).

Les structures répétitives (boucle Tantque)

Exemple.

Calcul du premier multiple de 11 supérieur à 134

```
Algorithme PremierMultipleDe11SupérieurA134
# cet algorithme calcule le premier multiple de 11
# supérieur à 134
variables
    multiple : entier naturel
début
    multiple = 0
    Tantque ( multiple ≤ 134 )
        Faire multiple ← multiple + 11
    Fin_Tantque
    Afficher ( multiple )
fin
```

« Faire tourner » un algorithme à la main

Intérêt.

- comprendre ce que fait un algorithme
- vérifier qu'un algorithme conçu est correct, et éventuellement le corriger...

Exemple.

Déterminer si un entier N est parfait
(égal à la somme de ses diviseurs stricts)

Ainsi...

- 6 est parfait : $1 + 2 + 3 = 6$
- 14 n'est pas parfait : $1 + 2 + 7 = 10 \neq 14$
- 28 est parfait : $1 + 2 + 4 + 7 + 14 = 28$

« Faire tourner » un algorithme à la main

```
Algorithme EntierParfait
```

```
# cet algorithme détermine si un entier N est parfait
```

```
variables
```

```
    N, diviseur, somme : entiers naturels
```

```
début
```

```
    Entrer (N)
```

```
    somme ← 0
```

```
    Pour diviseur de 1 à (N div 2)
```

```
    Faire
```

```
        Si (N mod diviseur = 0)
```

```
            Alors Somme ← Somme + diviseur
```

```
        Fin_Si
```

```
    Fin_Pour
```

```
    Si (N = Somme)
```

```
        Alors Afficher (N, " est parfait ")
```

```
    Sinon Afficher (N, " n'est pas parfait ")
```

```
    Fin_Si
```

```
fin
```

« Fair

main

```
somme ← 0
Pour diviseur de 1 à (N div 2)
Faire
    Si (N mod diviseur = 0)
    Alors Somme ← Somme + diviseur
    Fin_Si
Fin_Pour
```

opération

N

diviseur

somme

Entrer (N)

8

?

?

somme ← 0

0

(Pour) diviseur ← 1

1

N mod diviseur = 0 ? oui

Somme ← Somme + diviseur

1

« Fai

main

```
somme ← 0
Pour diviseur de 1 à (N div 2)
Faire
    Si (N mod diviseur = 0)
    Alors Somme ← Somme + diviseur
    Fin_Si
Fin_Pour
```

opération

N

diviseur

somme

8

1

1

(Pour) diviseur ← 2

2

N mod diviseur = 0 ? oui

Somme ← Somme + diviseur

3

(Pour) diviseur ← 3

3

« Faire

a main

```
somme ← 0
Pour diviseur de 1 à (N div 2)
Faire
    Si (N mod diviseur = 0)
    Alors Somme ← Somme + diviseur
    Fin_Si
Fin_Pour
```

opération

N

diviseur

somme

8

3

3

N mod diviseur = 0 ? non

(Pour) diviseur ← 4

4

N mod diviseur = 0 ? oui

Somme ← Somme + diviseur

7

```
Si (N = Somme)
Alors  Afficher (N, " est parfait ")
Sinon  Afficher (N, " n'est pas parfait ")
Fin_Si
```

opération

N

diviseur

somme

8

4

7

(Pour) fin de boucle

N = Somme ? non

Affichage : non parfait !

Application Algo Box

Algorithme d'Euclide

On souhaite produire un algorithme permettant de calculer le PGCD de deux nombres entiers positifs non nuls a et b , avec $a > b$, à l'aide de l'algorithme d'Euclide.

Approximation de $\sqrt{2}$ par dichotomie

Balle rebondissante

Conversion de durées

Deviner un nombre

Multiplication russe

...

Troisième partie

Exemple de mise en œuvre à l'aide d'outils logiciels (Scratch ou Metabot Blocs)

Scratch

Scratch 2 Offline Editor

Fichier Edition Conseils A propos

Scripts Costumes Sons

Mouvement Evénements
Apparence Contrôle
Sons Capteurs
Style Opérateurs
Données Ajouter blocs

avancer de 10
tourner de 15 degrés
tourner de 15 degrés
s'orienter à 90
s'orienter vers
aller à x: 0 y: 0
aller à pointeur de souris
glisser en 1 secondes à x: 0
ajouter 10 à x
donner la valeur 0 à x
ajouter 10 à y
donner la valeur 0 à y
rebondir si le bord est atteint
fixer le sens de rotation position
abscisse x

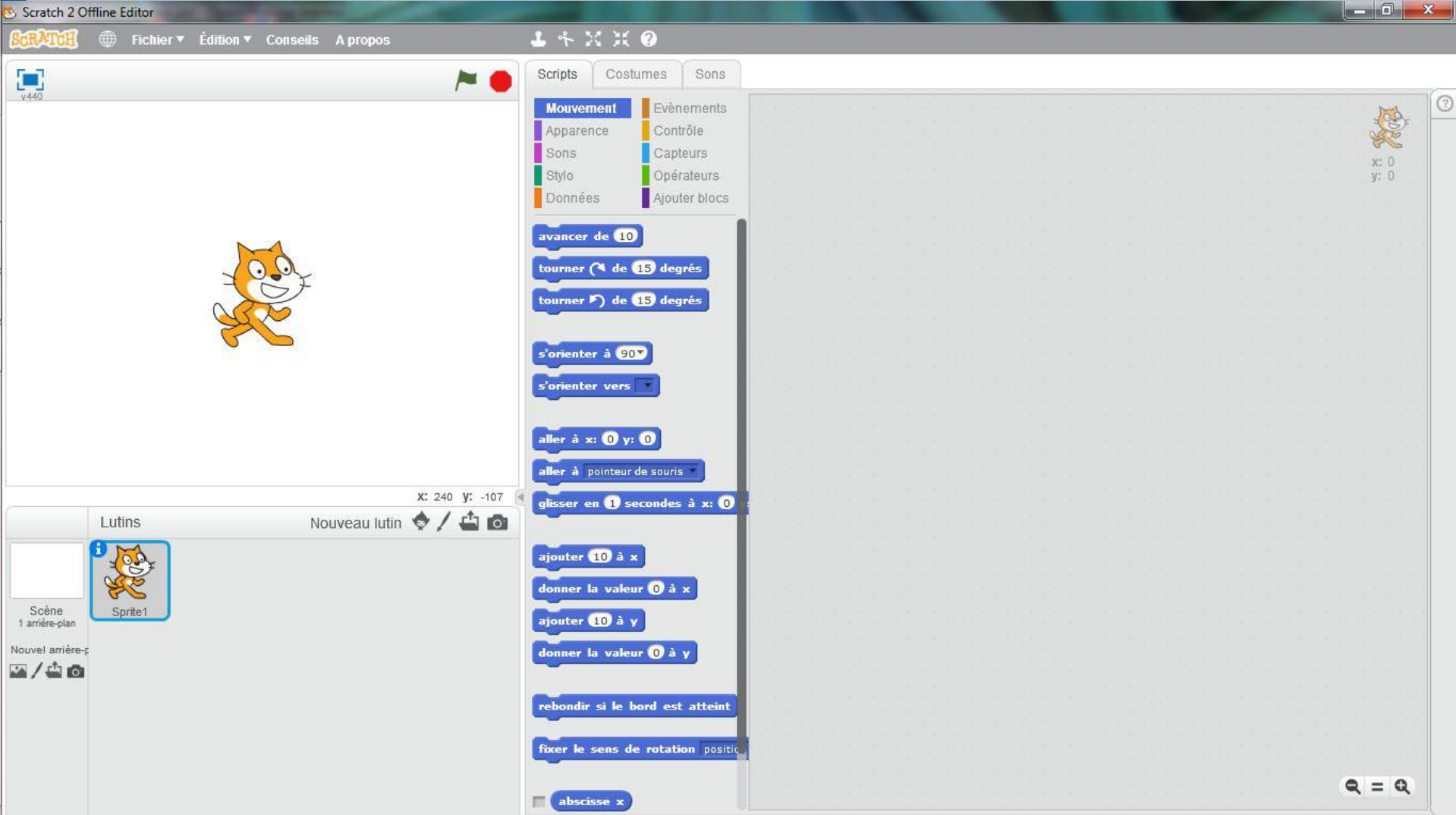
x: 240 y: -107

Lutins Nouveau lutin

Scène 1 arrière-plan
Nouvel arrière-plan

Sprite1

x: 0 y: 0

The image shows the Scratch 2 Offline Editor interface. At the top, the title bar reads "Scratch 2 Offline Editor". Below it is a menu bar with "Fichier", "Edition", "Conseils", and "A propos". The main workspace is divided into three panels. The top-left panel is the stage, showing a white background with a small cat sprite in the center. The bottom-left panel is the "Lutins" (Sprites) area, showing a "Nouveau lutin" button and a list of sprites, including "Sprite1" which is the cat. The right panel is the "Scripts" area, showing a list of categories (Mouvement, Evénements, Apparence, Contrôle, Sons, Capteurs, Style, Opérateurs, Données, Ajouter blocs) and a script area containing several blue blocks: "avancer de 10", "tourner de 15 degrés" (two instances), "s'orienter à 90", "s'orienter vers", "aller à x: 0 y: 0", "aller à pointeur de souris", "glisser en 1 secondes à x: 0", "ajouter 10 à x", "donner la valeur 0 à x", "ajouter 10 à y", "donner la valeur 0 à y", "rebondir si le bord est atteint", "fixer le sens de rotation position", and "abscisse x". The stage coordinates are shown as "x: 240 y: -107" and the cat sprite's coordinates as "x: 0 y: 0".

Scratch



Scratch

avancer de 10

tourner de 15 degrés

tourner de 15 degrés

s'orienter à 90

s'orienter vers

aller à x: 0 y: 0

aller à pointeur de souris

glisser en 1 secondes à x: 0

ajouter 10 à x

donner la valeur 0 à x

ajouter 10 à y

donner la valeur 0 à y

rebondir si le bord est atteint

fixer le sens de rotation position

abscisse x

quand flag pressé

quand espace est pressé

quand ce lutin est cliqué

quand l'arrière-plan bascule sur

quand volume sonore > 10

quand je reçois message1

envoyer à tous message1

envoyer à tous message1 et

effacer tout

estampiller

stylo en position d'écriture

relever le stylo

choisir la couleur pour le stylo

ajouter 10 à couleur du stylo

mettre la couleur du stylo à 0

ajouter 10 à l'intensité du stylo

choisir l'intensité 50 pour le stylo

ajouter 1 à la taille du stylo

choisir la taille 1 pour le stylo

attendre 1 secondes

répéter 10 fois

répéter indéfiniment

si alors

si alors

sinon

attendre jusqu'à

répéter jusqu'à

stop tout

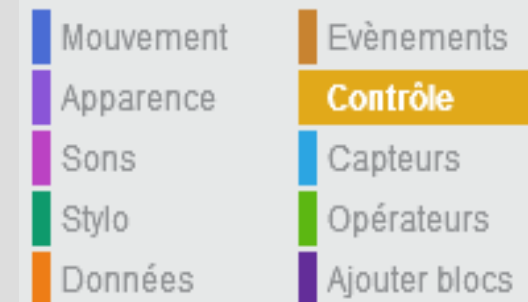
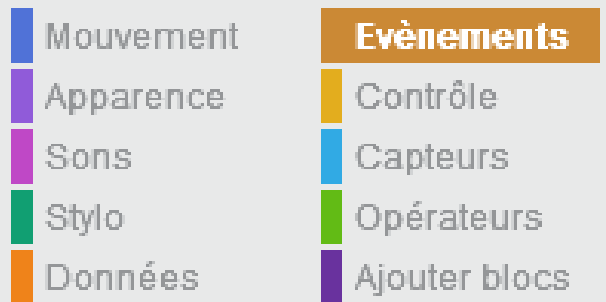
quand je commence comme

créer un clone de moi-même

supprimer ce clone

Scratch

Activité 1 : Mouvements et Actions sur lutin



Exercice 1 :

Pour commencer, visualisons les dimensions de la scène, en pixels.

Choisir une scène différente, on cliquant sur le dossier : Sélectionner la scène nommée **XY-grid**.

Placer le lutin à gauche de l'écran et écrire un script afin qu'il traverse de gauche à droite en une seule fois quand le drapeau vert est pressé.

Scratch

Activité 2 : Données et Opérateurs, Capteurs

Exercice 4 :

Ecrire un programme qui affiche la table de 7.

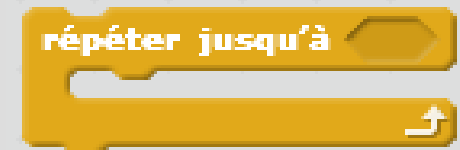
Aide : Utiliser les blocs suivants :



Exercice 5 :

Ecrire un programme qui affiche la table de 7 en utilisant une **variable** dans **une boucle**.

Aide : Utiliser les blocs suivants :



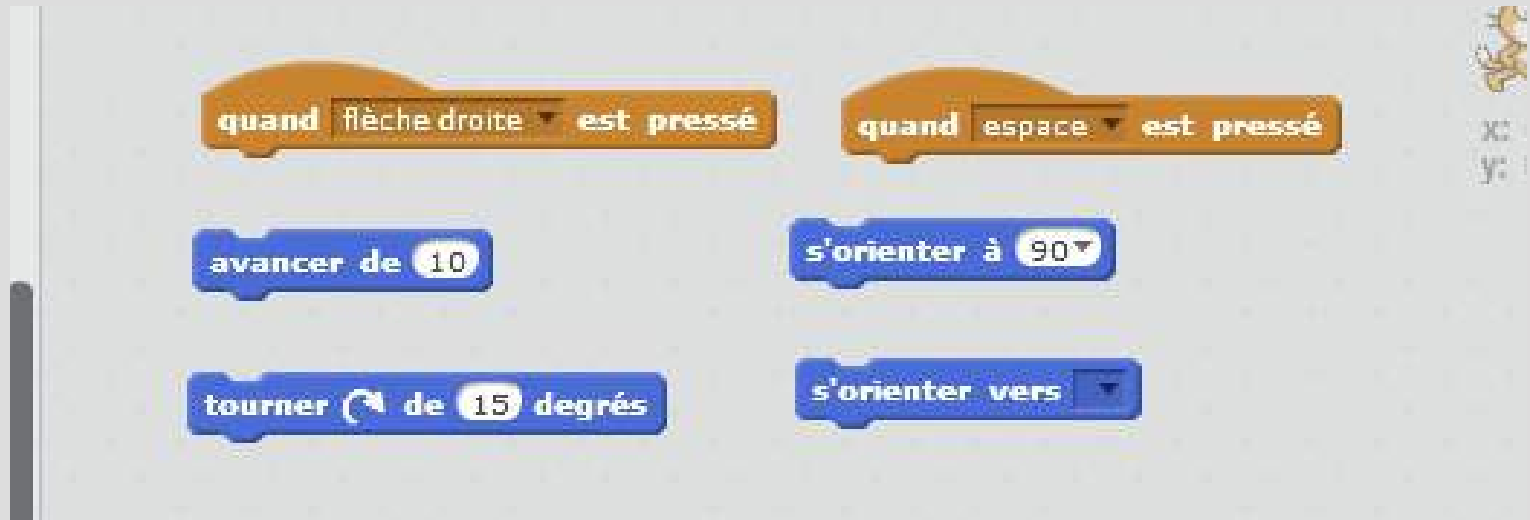
Scratch

Activité 1 : Mouvements et Actions sur lutin

Exercice 2 :

Ecrire un script permettant de faire avancer avec le clavier le lutin et de le faire rebondir chaque fois qu'il atteint un bord.

Aide : Utiliser les blocs suivants :



Scratch

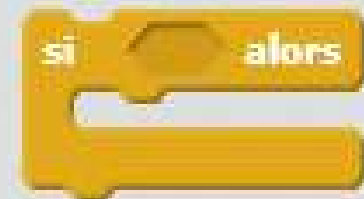
Activité 1 : Mouvements et Actions sur lutin

Exercice 3 :

Ajouter un second lutin au centre de la scène.

Quand le premier lutin le rencontre faite les se saluer

Aide : Utiliser les blocs suivants :



Scratch

Objectifs, grandes étapes de réalisation d'un projet

En tout premier il faut penser à son scénario :

- Combien de lutins contient le jeu? Les créer.
- Combien de tableau y aura-t-il? Créer les fonds d'écran.
- Quel est le but de chaque tableau? Comment passer d'un tableau au suivant?
- Est-ce qu'il y a un score? Comment l'augmenter ?
- Y aura-t-il un générique de début et de fin?
- Les messages entre lutins (signaux envoyés à tous) sont-ils clairs?

Scratch

Création pas à pas du jeu « Thésée et le Minotaure »



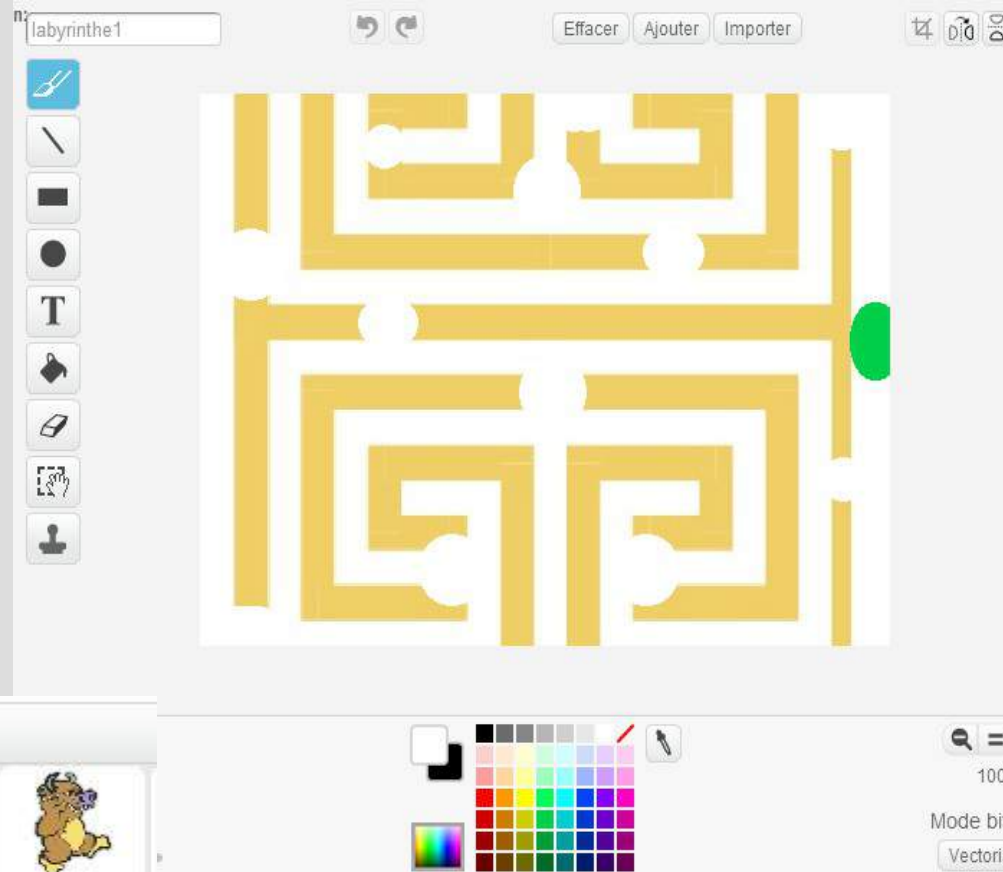
Scratch

Création pas à pas du jeu « Thésée et le Minotaure »

```
quand  pressé  
aller à x: 0 y: 0  
montrer  
dire Salut! pendant 2 secondes
```

```
quand  est pressé  
tourner  de 90 degrés  
avancer de 10
```

```
quand  est pressé  
tourner  de 90 degrés  
avancer de 10
```



Scratch

Création pas à pas du jeu « Thésée et le Minotaure »

```
quand [drapeau cliqué] pressé
répéter indéfiniment
  si [couleur touchée?] alors
    avancer de -4
```

```
quand [drapeau cliqué] pressé
cacher
aller à x: [nombre aléatoire entre -200 et 200] y: [nombre aléatoire entre -100 et 100]
attendre 3 secondes
montrer
```

```
quand [drapeau cliqué] pressé
répéter indéfiniment
  si [couleur touchée?] alors
    dire [J'ai gagné!] pendant 2 secondes
    ajouter à SCORE 10
    envoyer à tous Fin_du_jeu
```

```
 SCORE
mettre SCORE à 0
ajouter à SCORE 1
montrer la variable SCORE
cacher la variable SCORE
```

```
quand [drapeau cliqué] pressé
répéter indéfiniment
  si [Trésor touché?] alors
    dire [A moi le trésor!] pendant 2 secondes
    ajouter à SCORE 10
    envoyer à tous tresor_trouvé
```


Metabot Blocs



Metabot Blocs

The screenshot shows the Metabot Blocks programming environment. At the top, there are three tabs: "Editeur", "Paramètres", and "Aide". On the left side, there is a sidebar menu with the following categories: "Robot" (containing Distance, Vitesse, Paramètres, Leds), "Programme" (containing Temps, Affichage, Logique, Boucles, Maths, Variables, Fonctions), and "Temps" (which is currently selected). The main workspace contains the following blocks:

- A green block: attendre 1000 millisecondes
- A red block: définir item à créer un chronomètre
- A green block: obtenir la valeur du chronomètre item
- A green block: attendre que le chronomètre item atteigne la valeur 1000
- A green loop block: répéter toute les 1000 ms faire

Editeur

Paramètres

Aide

Robot

Distance

Vitesse

Paramètres

Leds

Programme

Temps

Affichage

Logique

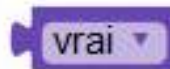
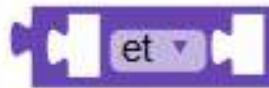
Boucles

Maths

Variables

Fonctions

Execution



Editeur

Paramètres

Aide

Robot

Distance

Vitesse

Paramètres

Leds

Programme

Temps

Affichage

Logique

Boucles

Maths

Variables

Fonctions

Execution

répéter 10 fois

faire

répéter tant que vrai

faire

répéter jusqu'à

faire

répéter avec i de 1 à 10

faire

sortir de la boucle

Metabot Blocs

The image shows a software interface for 'Metabot Blocs'. On the left is a sidebar with a list of categories. The 'Maths' category is currently selected and highlighted in grey. The main area on the right displays several blue blocks from the 'Maths' category. The blocks are: a block with the number '0', a block with a plus sign and a dropdown arrow, a block with the text 'est pair' and a dropdown arrow, a block with the text 'cos' and a dropdown arrow, a block with the text 'atan2' and a dropdown arrow followed by the text 'et' and another dropdown arrow, a block with the Greek letter pi (π) and a dropdown arrow, and a block with the text 'nombre entier' and a dropdown arrow, followed by 'aléatoire entre', a box containing the number '1', the text 'et', and a box containing the number '10'.

Robot

- Distance
- Vitesse
- Paramètres
- Leds

Programme

- Temps
- Affichage
- Logique
- Boucles
- Maths**
- Variables
- Fonctions
- Execution

0

+

est pair

cos

atan2 et

π

nombre entier aléatoire entre 1 et 10

Metabot Blocs

The image shows a software interface for 'Metabot Blocs'. At the top, there are three tabs: 'Editeur', 'Paramètres', and 'Aide'. On the left side, there is a sidebar with two main sections: 'Robot' and 'Programme'. Under 'Robot', there are sub-items: 'Distance', 'Vitesse', 'Paramètres', and 'Leds'. Under 'Programme', there are sub-items: 'Temps', 'Affichage', 'Logique', 'Boucles', 'Maths', 'Variables', and 'Fonctions'. The 'Variables' category is currently selected and highlighted. The main workspace on the right contains several blocks:

- A block labeled 'variable globale: item', with a dropdown menu showing 'item' and a default value of '0'.
- A block labeled 'définir item à', with a dropdown menu showing 'item' and a white square icon.
- A block labeled 'incrémenter item de', with a dropdown menu showing 'item' and a value of '1'.
- A block labeled 'item', with a dropdown menu showing 'item'.
- A block labeled 'var du programme en cours', with a dropdown menu showing 'var' and another dropdown menu showing 'en cours'.
- A block labeled 'définir var du programme en cours à', with a dropdown menu showing 'var', another dropdown menu showing 'en cours', and a white square icon.

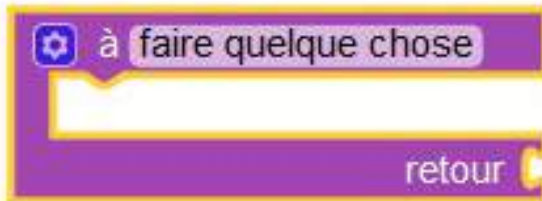
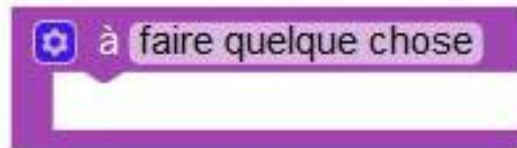
Metabot Blocs

Robot

Distance
Vitesse
Paramètres
Leds

Programme

Temps
Affichage
Logique
Boucles
Maths
Variables
Fonctions
Execution



Crée une fonction avec une sortie.



Quatrième partie

Illustrations diverses

Merci de votre attention...