MPSI 2017-2018

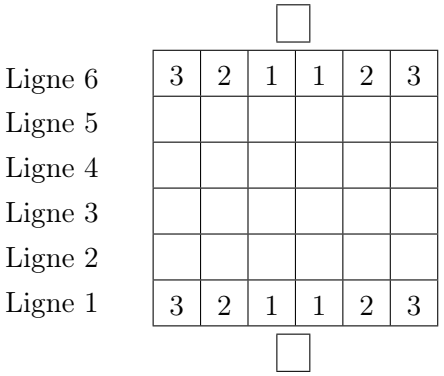
# Informatique

**Mini-projet Gygès**

Le but du mini-projet est de programmer un jeu du nom de Gygès. Les règles sont en annexe. Le joueur sud sera joué par un joueur humain tandis que le joueur nord devra être géré par l'ordinateur. Quelques consignes pour l'implémentation :

1. Chaque case sera représentée soit par
   - le caractère 'A' pour votre but
   - le caractère 'B' pour le but de l'ordinateur
   - une chaine de 2 caractères '11', '12', '13', '14', ... '66' pour les cases centrales. Ce sont les coordonnées de ces cases.

2. Pour modéliser le plateau de jeu, on utilisera un dictionnaire. Les clés seront les cases et les valeurs seront soit 0, si la case est vide, soit 1, 2 ou 3 si la case est occupée par un pion simple, double ou triple. Ainsi le plateau :




   sera codé (en enlevant les guillemets des chaines pour plus de clarté) par :

$$\{A : 0,\ 11 : 3,\ 12 : 2,\ 13 : 1,\ 14 : 1,\ 15 : 2,\ 16 : 3,\ 21 : 0, \ldots,\ 66 : 3,\ B : 0\}$$

3. Un coup sans saut sera modélisé par une liste de cases ; la première case étant l'emplacement du pion. Ainsi dans la configuration de départ donnée dans le point précédent La liste [66, 56, 55, 54] sera le coup possible du triple situé en 66. Pour qu'un coup soit valable, il faut que :
   - les cases qui se suivent dans le tableau soient adjacentes sur le plateau,
   - seule la première case et la dernière case peuvent-être occupées (la première , toujours),
   - il n'y a pas deux fois le même intervalle. On rappelle qu'un intervalle est l'espace situé entre deux cases.

4. A chaque pion, on associera également sa portée, c'est-à-dire la liste des coups qu'il pourra parcourir sans saut. Ainsi la portée du pion en 12 est :

$$[[12, 22, 32], [12, 22, 23], [12, 22, 21]]$$

## Partie I. Préliminaires

1. Créer la fonction *Adjacentes(Case1,Case2)*. Elle prend en argument 2 cases codées comme des chaînes de caractères *'LC'* (avec $L$ et $C$ des entiers compris entre 1 et 6 indices de ligne/colonne pour les cases centrales) ou *'A'* ou *'B'* (pour les buts) et retourne un booléen : *True* si les deux cases sont adjacentes et *False* sinon.
   On rappelle que deux cases centrales sont adjacentes ssi elles partagent exactement un côté, le but *'A'* est adjacent uniquement aux cases de la ligne 1 et *'B'* à celles de la ligne 6.

2. Créer la fonction *Voisins(Case)* qui retourne la liste (dans l'ordre de votre choix) des cases adjacentes à celle donnée en argument.
   La plupart des cases ont 4 voisins, sauf celles situées aux bords gauche et droit du plateau qui n'en ont que 3 et les buts qui en ont 6.

3. Créer une classe *Jeu* avec son constructeur (c'est-à-dire la méthode *__init__*). Elle aura comme paramètre, une liste représentant les positions des pièces au départ. Cette liste sera de longueur 12, les éléments de 0 à 3 donneront la position des triples, les éléments de 4 à 7 donneront la position des doubles et le reste la position des simples. Le constructeur devra créer un attribut *plateau* représentant la position des pièces du plateau comme expliquée plus haut.
   On pourra éventuellement accepter aussi des listes ne positionnant pas les anneaux sur les lignes 1 et 6 afin de pouvoir créer facilement des situations de test, par exemple celles données dans les illustrations des règles.
   Si on procède ainsi, il peut être judicieux d'initialiser toutes les cases à zéro avant de modifier celles occupées.

4. Créer la méthode *__repr__(self)*. Elle devra afficher le plateau de jeu avec la position des pièces. On se contentera d'afficher un tableau 6x6 de caractères représentant les pièces avec deux lignes en plus les deux buts. Par la suite (quand le reste tournera !), on pourra éventuellement créer un affichage plus élaboré avec par exemple le module *tkinter*.

## Partie II. Vérifier la validité d'un coup et l'exécuter

À ce stade, on ignore la règle indiquant quels anneaux sont jouables. Pour simplifier encore (car l'action *déloger* complique les choses), on considère ici qu'un coup est la donnée d'une liste de cases adjacentes. Les cases intermédiaires doivent être libres sauf celles correspondant à un rebond, la valeur de l'anneau s'y trouvant indiquant alors le nombre de cases qu'on peut à nouveau parcourir. La case initiale est initialement occupée mais peut-être à nouveau traversée (puisque c'est l'anneau qui s'y trouvait qu'on déplace). La case finale peut être occupée ou non (selon qu'on utilise ou non l'action *déloger* dont on ne se préoccupera que plus tard car elle nécessite des informations supplémentaires).

1. Créer la méthode *SembleValide(self, LeCoup)* Elle devra seulement vérifier si deux cases consécutives sont adjactentes et si le coup ne parcourt pas 2 fois le même intervalle.

2. Créer la méthode *CoupPossible(self, LeCoup)*. Elle renvoie *True* si le coup est possible et *False* sinon. Elle devra vérifier que la méthode ou fonction de la question précédente renvoie *True* mais aussi que lorsqu'il y a un saut, il est de la bonne longueur, et que les cases intermédiaires sont libres (sauf quand on repasse par la case initiale).

3. Créer la méthode *ExécuteCoup(self, CoupJoué)*. Cette méthode devra mettre à jour le plateau de jeu, en exécutant le coup *CoupJoué*.
   Attention, ce travail n'est pas entièrement satisfaisant et ne sera finalisé qu'avec la méthode *EstValide* car on a pour l'instant ignoré la règle indiquant que les seuls anneaux jouables sont les

plus proches du but du joueur. En effet cette règle est délicate car elle admet une exception (cf illustration 10 des règles) et la méthode *MAJCoupsValides* qu'on créera entre temps facilitera la vérification de cette condition.

## Partie III. Générer la liste des coups jouables

On ne se préoccupe toujours pas de l'action déloger (et en particulier de la case où sera envoyé l'anneau délogé), on se contente de considérer qu'un déplacement [avec rebond(s) ou saut(s), on emploie les deux termes indistinctement] peut se terminer sur une case vide ou occupée.

1. Créer la méthode *MAJCoupsSansSaut(self)*. Elle devra construire un dictionnaire représentant la portée de chaque pièce et le mettre dans un attribut appelé *CoupsSansSaut*. Les clés seront les cases. La valeur associée à une clé sera la portée de la pièce se trouvant sur la case c'est-à-dire la liste de ses coups sans saut possibles. Si la case est vide, la valeur sera la liste vide.

2. Créer la méthode *ConcatèneCoups(self, Coup1, Coup2)*. Cette méthode renvoie la concaténation des 2 coups si le coup obtenu est valide et *False* sinon. On supposera les deux coups jouables, et que la case d'arrivée du premier correspond à la case de départ du second (elle sera donc occupée). Dans la concaténation, cette case ne sera bien entendu pas dupliquée.

3. Créer la méthode *MAJCoupsAvecSaut(self)* qui met dans un attribut appelé *CoupsAvecSaut* la liste des coups possibles avec sauts. On vérifiera également si on peut bouger la pièce c'est-à-dire si aucune autre pièce ne se trouve plus près de votre but. Attention, ici on demande une liste, pas un dictionnaire. Le coup peut finir sur une case occupée, ce qui permettra d'utiliser l'action déloger.

4. Créer la méthode *LigneJouable(self,Joueur)* dont l'argument *Joueur* vaudra soit *'Nord'* soit *'Sud'*, et qui retourne un entier représentant le numéro de la première ligne jouable par ce joueur, c'est-à-dire la ligne (au moins partiellement) occupée la plus proche du but de ce joueur.

5. Créer la méthode *MAJCoupsValides(self,Joueur)* qui met dans un attribut appelé *CoupsValides* la liste des coups valides pour le joueur en question. Par rapport à la précédente, cette méthode vérifie de plus que l'anneau déplacé est jouable par le joueur. On rappelle qu'un joueur ne peut déplacer qu'un anneau parmi ceux se trouvant la ligne occupée la plus proche de son but, en tenant compte de l'exception indiquée avec l'illustration 10. En pratique, cette méthode sera utilisée pour Nord joué par l'ordinateur.

6. Créer une méthode *EstValide(self,LeCoup)* retournant un booléen et testant si un coup donné est valide (améliorant *CoupPossible* en effectuant aussi les vérifications de la question précédente ici).

## Partie IV. Gérer une partie

1. Créer la méthode *Joue(self)*. Elle devra demander au joueur (donc Sud) de saisir son coup et vérifier s'il est possible (et sinon, redemander !). Le coup donné sera de la forme $c = [x_1, x_2, x_3, ..., x_n]$. Dans un premier temps $c$ donnera toutes les positions intermédiaires. En amélioration, on peut penser à ne donner que quelques postions qui suffisent à décrire le coup.
   La saisie par l'utilisateur pourra se faire selon l'exemple suivant : *"41 51 52 53 63 B"* (l'utilisateur ne saisit pas les guillemets et sépare les cases par des espaces), et c'est à votre code qu'il revient de convertir cette chaîne unique en la liste de chaînes *['41','51','53','63','B']*. La méthode *split* sur les chaînes peut grandement vous faciliter la tâche.

2. Modifier la méthode *Joue(self)* afin de lui permettre de spécifier un coup avec l'option déloger, par exemple sous la forme : $c = [case_1, case_2, case_3, ..., case_n, 'D', case_{n+1}]$, où l'anneau placé sur la $case_n$ est délogé vers la case $case_{n+1}$. Adapter de même les méthodes *MAJCoupsvalides(self,Joueur)* et *ExécuteCoup(self, CoupJoué)*

3. Créer la méthode *TesteFin(self)*. Elle devra renvoyer *True* si l'un joueur a gagné, *False* sinon.

4. Créer la méthode *JoueOrdi(self)*. Elle devra parcourir la liste des coups possibles. S'il y en a un gagnant, il le joue, sinon il en joue un au hasard.

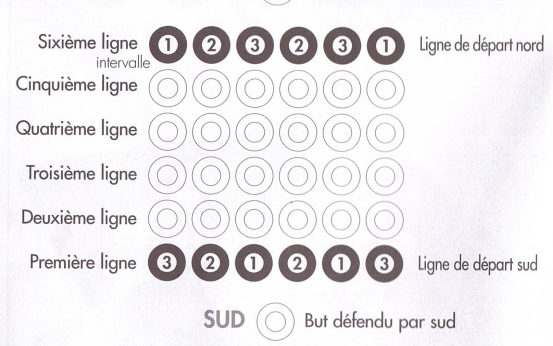
5. Créer la méthode *Début(self)*. Elle devra gérer la partie.

6. Chercher des idées pour améliorer la méthode *JoueOrdi(self)*.

# Gygès®

2 joueurs à partir de 8 ans
**Auteur: Claude LEROY**

***illustration n°1***




**Afin de faciliter la compréhension des règles:**

Nous vous recommandons de lire entièrement les règles du jeu. Placez le plateau devant vous et reconstituez les exemples illustrés:
Vous comprendrez ainsi très rapidement les règles du jeu.
Nous appellerons les deux joueurs «Nord» et «Sud» ***Illustration No 1***

**Contenu:**

1 plateau de jeu de 38 cases,
soit 36 cases centrales et 2 cases de but
12 anneaux:
4 simples, 4 doubles, 4 triples
(tous les anneaux sont de même couleur)

## DISPOSITION DE DEPART

Les joueurs tirent au sort le camp qu'ils utiliseront. Le joueur qui obtient le camp Sud (celui pour qui la marque GYGES est à sa gauche) commence.

***Illustration 1***: exemple de disposition de départ. Chaque joueur prend six anneaux: 2 simples, 2 doubles, et 2 triples.

Le joueur Sud commence la partie et place tous ses anneaux dans l'ordre de son choix sur la première ligne. A son tour, le joueur Nord place tous ses anneaux dans l'ordre de son choix sur la sixième ligne.





## BUT DU JEU

Le but du jeu est d'amener un anneau (simple, double ou triple) jusqu'au but adverse.

## DEPLACEMENT DES PIECES

1. Les joueurs jouent à tour de rôle. Chacun leur tour, ils doivent déplacer un anneau de la manière suivante: ***Illustrations 2a, 2b, 2c***
   - un anneau simple se déplace uniquement d'une case, ***Illustration 2a***
   - un anneau double se déplace obligatoirement de deux cases, ***Illustration 2b***
   - un anneau triple se déplace obligatoirement de trois cases. ***Illustration 2c***

2. On peut déplacer un anneau dans quatre directions: à droite, à gauche, en avant et en arrière.

3. Un anneau double ou triple peut changer de direction en route en tournant à angle droit. ***Illustration 2b et 2c***: quelques exemples de déplacement

4. Il est interdit de déplacer un anneau en diagonale sur les 36 cases centrales du plateau. ***Illustration 2b et 2c***

5. Seul le but adverse est accessible en diagonale. Voir notamment les ***illustrations 5 et 6b*** (flèches pointillées)

6. Vous pouvez jouer uniquement les anneaux situés sur la ligne occupée la plus proche de vous.

Tant que Sud a des anneaux en première ligne, il doit jouer un anneau situé sur la première ligne. Si la première ligne est vide, il doit jouer un anneau situé sur la deuxième ligne. Si la première et la deuxième ligne sont vides, il doit jouer un anneau situé sur la troisième ligne, et ainsi de suite.

***Illustration 3***

Nord doit jouer un des anneaux se trouvant sur sa ligne de départ.

Sud doit jouer depuis la troisième ligne. Les anneaux situés entre les deux lignes jouables peuvent servir à augmenter la capacité de déplacement d'autres anneaux. La suite vous montrera comment utiliser cette capacité.

Un anneau ne peut passer en cours de route par dessus un autre anneau. L'anneau rencontré constitue dans ce cas un obstacle. Par contre, si votre anneau termine son coup sur une case occupée par un autre anneau, vous pouvez décider:

soit de rebondir (Voir ci-après point I)
soit de déloger (Voir ci-après point II)

***Illustration 4a***

Le triple ne peut pas se rendre au point Z.

***Illustration 4b***

Le triple peut se rendre sur les anneaux simple et double...

## I) REBONDIR

Si un anneau accède à une case déjà occupée par un autre anneau, il peut décider de rebondir:
- uniquement d'une case s'il arrive sur un anneau simple
- obligatoirement de deux cases s'il arrive sur un anneau double
- obligatoirement de trois cases s'il arrive sur un anneau triple.

L'anneau sur lequel il rebondit ne change pas de place. Il constitue à ce moment un remplin.

Un anneau peut rebondir plusieurs fois de suite et obtenir ainsi un rayon d'action très important. En effet, la capacité initiale de déplacement de l'anneau est ainsi considérablement augmentée.

Les parcours à rebonds multiples permettent d'atteindre le but défendu par l'adversaire, et donc de gagner la partie!

***Illustration 5***

Sud joue son simple et rebondit sur un double, puis sur un triple, puis sur un double, puis sur un triple, puis finalement sur un simple et accède au but. Il remporte la partie.

**Remarques importantes**

1) Lors d'un parcours, un anneau ne peut passer deux fois par le même intervalle. (On appelle intervalle l'espace compris entre deux cases. ***Illustration 1***) Par contre, un anneau peut passer deux fois par la même case.





***Illustration 6a***

Sud peut, avec son double, atteindre le triple mais il ne peut pas repartir en sens inverse car il repasse par le même intervalle.

***Illustration 6b***

Sud joue son triple en arrière, rebondit sur son double, atteint le simple en repassant par la même case, rebondit sur le triple et remporte la partie.

2) Lorsqu'un anneau traverse le plateau et atteint la ligne jouable de l'adversaire, celui-ci pourra le rejouer dès le coup suivant. Plus fort encore, si son dernier rebond lui fait dépasser cette ligne, il deviendra même le seul anneau que l'adversaire pourra jouer.

***Illustration 7a***

Sud joue un simple, rebondit sur un double, puis sur un simple, puis sur un double, puis sur un simple, et s'arrête sur la case vide Z. Nord pourra le rejouer dès son tour suivant

***Illustration 7b***

Sud joue un simple, rebondit sur un double, puis sur un autre double et s'arrête sur la sixième ligne. Nord sera obligé de jouer cet anneau au prochain coup.

## II) DELOGER

Si lors d'un déplacement, un joueur accède à une case déjà occupée par un anneau, il peut décider au lieu de rebondir de le déloger. L'anneau avec lequel on déloge prend simplement la place de l'anneau délogé. L'anneau délogé est relogé sur une des cases centrales vides du plateau.

Attention: Un anneau délogé ne peut pas être relogé au-delà de la ligne jouable par l'adversaire.

***Illustrations 8a***

Sud joue son double, déloge le triple de la troisième ligne et le reloge en quatrième ligne. Le double a pris la place du triple en troisième ligne.

Etant donné que Nord n'a plus d'anneau sur sa ligne de départ (sixième ligne),ses anneaux jouables se trouvent en cinquième ligne.
Sud ne peut donc reloger un anneau que sur les cases vides des cinq premières lignes. (voir cadre fléché) En aucun cas un anneau ne peut être relogé entre le but de l'adversaire et la ligne où se trouvent ses anneaux jouables.

***Illustration 8b***

Suite au coup qu'il a joué au point 8a, Sud a maintenant deux accès possibles pour atteindre le but.

Un joueur peut déloger un anneau après avoir rebondi une ou plusieurs fois sur d'autres anneaux. Voir notamment ***illustration 12a***

Attention: On ne peut en aucun cas jouer un coup qui reproduirait par rebond ou délogeage la situation antérieure. On devra alors choisir un autre coup.

## FIN DE LA PARTIE

Le premier joueur qui amène un anneau sur le but adverse remporte la partie.

## PRECISIONS IMPORTANTES

Un but n'est pas un lieu de passage. On ne peut utiliser la case du but comme case de déplacement. Seul l'adversaire a le droit d'accéder au but et il ne peut le faire qu'une fois, puisque ce coup met fin à la partie.

***Illustrations 9a***

Exemple d'un anneau qui finit son parcours en rebondissant sur un triple en cinquième ligne et par suite d'encombrement ne peut accéder au but.

***Illustration 9b***

Exemple d'un anneau qui a trois voies d'accès au but.

## CAS PARTICULIER

Il peut arriver, bien que ce cas soit rare, que tous vos anneaux jouables soient bloqués. Dans ce cas, exceptionnellement, vous pourrez jouer un anneau de la ligne suivante.

***Illustration 10***

Tous les anneaux de la première ligne de Sud sont bloqués, il doit donc jouer un anneau de la deuxième ligne.





## APERÇUS STRATEGIQUES

**1. L'attaque**

Vous l'avez compris, pour accéder au but adverse, un anneau a besoin de rebondir sur d'autres anneaux. Lorsque vous déplacez un anneau, gardez à l'esprit que vous ne pourrez généralement pas le rejouer au coup suivant. Il doit donc être placé de manière à aider les pions restés en arrière à amplifier leur mouvement initial. Votre souci sera de vous créer un «réseau» vous permettant d'accéder au but adverse, en évitant que votre adversaire l'utilise en sens inverse.

***Illustration 11***

Sud joue son double, déloge le triple de la troisième ligne et le reloge au point Z. Sud construit ainsi pour le coup suivant un réseau d'attaque partant de son anneau simple aboutissant au but adverse.

**2. Défense et contre-attaque**

Là encore, déloger et reloger constitue le coup tactique idéal.

***Illustration 12a***

Supposons que votre adversaire Nord vous menace de l'attaque «triple-triple-but» Vous pouvez par exemple vous défendre en déplaçant "votre" double pour déloger un simple et le reloger entre les deux triples attaquant.

***Illustration 12b***

Du même coup, vous menacez votre adversaire par «triple-double-simple-double-but»

## POUR PROGRESSER RAPIDEMENT

Une bonne méthode consiste à rejouer le dernier mauvais coup. En effet, si par exemple un joueur crée lors de son déplacement une voie qui peut mener son adversaire directement au but, on peut considérer ce coup comme une «erreur d'inattention», et le joueur devra le rejouer.

Ainsi, en rejouant un coup inadapté, on découvre rapidement l'envergure du jeu, et ses innombrables possibilités.

Dailleurs, en tournoi, un joueur ne pourra pas gagner suite à une erreur d'inattention de l'adversaire, mais uniquement lorsqu'il crée une menace que l'adversaire ne peut d'aucune manière parer.

## CONTACT

Si vous désirez obtenir des informations concernant:

- les règles du Gygès
- Gygès sur Internet
- les tournois
- Comment noter une partie de Gygès
- les autres jeux édités par Swissgames
- ou encore nous communiquer vos meilleures parties...

**Vous pouvez nous contacter par e-mail:**
Swissgames@swissonline.ch

**par lettre postale:**
Swissgames
Place du Temple 2
1227 Carouge

**par fax:**
++41.22.342.55.59

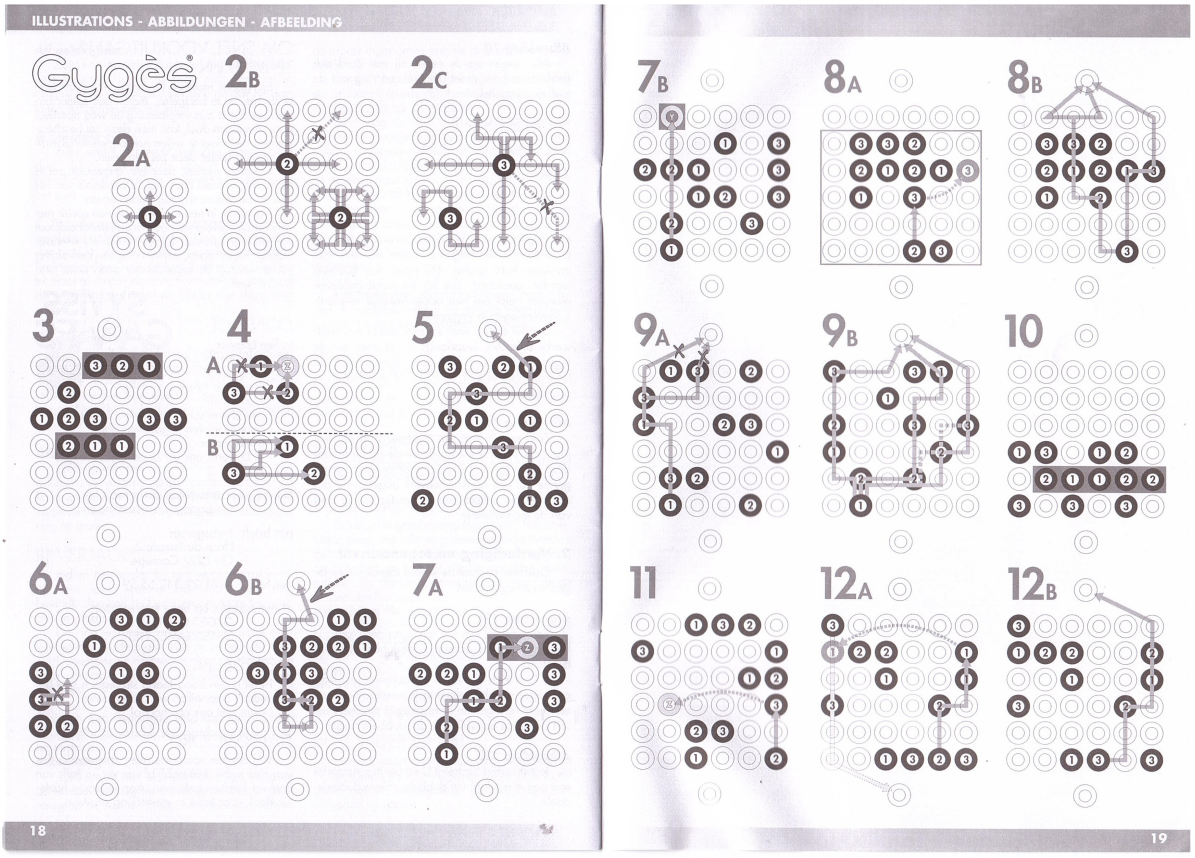
**Ou encore directement sur nos sites internet:**
GYGES.COM
SWISSGAMES.NET

## AVIS

Vous avez inventé un jeu de stratégie de haut niveau?
Nous sommes interessés à vous rencontrer.
Contactez-nous!

## CREDIT

Nous remercions vivement pour leur collaboration et leurs encouragements toutes les personnes qui nous ont aidés, de très près, de près, ou de loin et même de vraiment loin.


ILLUSTRATIONS - ABBILDUNGEN - AFBEELDING
Gygès
2A
2B
2C
3
4
A
B
5
6A
6B
7A
18
7B
8A
8B
9A
9B
10
11
12A
12B
19