MASTER EN SCIENCES INFORMATIQUES

Projet d'enseignement assisté par ordinateur sur la découverte des transformations du plan

Bonhomme, Simone

Award date:
1985

Awarding institution:
Universite de Namur

FNDP Namur

Année Académique 1984 - 1985

PROJET D'ENSEIGNEMENT ASSISTE PAR ORDINATEUR SUR LA DECOUVERTE DES TRANSFORMATIONS DU PLAN

Promoteur : Monsieur Claude CHERTON

Mémoire présenté par Simone BONHOMME
en vue de l'obtention du grade de
licencié et maître en informatique.

Je tiens à remercier Monsieur Claude CHERTON pour les conseils et les encouragements sans cesse répétés, pour la disponibilité et la compréhension témoignées tout au long de l'année.

Je remercie également Monsieur Jean-Marie PONCELET de m'avoir partagé son expérience.

TABLE DES MATIERES

CHAPITRE I : INTRODUCTION

1. Choix du sujet 1
2. Vous avez dit EAO ? 2
   2.1 Objectifs 2
   2.2 Situation actuelle 2
   2.3 Que nous réserve l'avenir ? 4
3. Le sujet 5
   3.1 Introduction du sujet 5
   3.2 L'archipel des isométries 6
      3.2.1 Qu'est ce que le GEM ? 6
      3.2.2 Orientation pédagogique 6

CHAPITRE II : MISE AU POINT D'UN SCENARIO

1. Les sous-objectifs 8
2. Découverte de différents mouvements 9
   2.1 Démarche proposée 9
   2.2 Quelles figures proposer ? 9
   2.3 Comment offrir un calque ? 17
   2.4 Classement des différents cas étudiés 19
3. Caractérisation des différents mouvements 20
   3.1 La translation 20
      3.1.1 Une approche possible 20
      3.1.2 Remarques sur cette approche 22
   3.2 La rotation 24
      3.2.1 Une approche possible 24
      3.2.2 Le centre d'une rotation 24
   3.3 L'homothétie 26
   3.4 Remarques 28
4. Comment construire ? 29
   4.1 Première proposition : des primitives 29
   4.2 Seconde proposition : des outils 31
   4.3 Gestion des erreurs 33
5. Nécessité de points de rappel 35

CHAPITRE III : EDITEUR GRAPHIQUE

1. Introduction 36
2. L'écran et le curseur 37
   2.1 Description de l'écran 37
   2.2 Description du curseur 37
3. L'outil GOMME 39
4. L'outil POINT 40
5. L'outil REGLE 41
6. L'outil COMPAS 45
7. Configuration 48

CHAPITRE IV : IMPLEMENTATION

1. Généralités 49
   1.1 TURTLEGRAPHICS 49
   1.2 Découpe en modules 50
2. Gestion du curseur 51
3. Le module GOMME 52
4. Le module CONFIGURATION 53
5. Le module POINTMODE 54
6. Le module LATTE 56
7. Le module COMPAS 57

CONCLUSIONS 58

ANNEXES

CHAPITRE I : INTRODUCTION

1. Choix du mémoire
2. Vous avez dit EAO ?
   - 2.1 Objectifs
   - 2.2 Situation actuelle
   - 2.3 Que nous réserve l'avenir ?
3. Le sujet
   - 3.1 Introduction du sujet
   - 3.2 L'archipel des isométries
     - 3.2.1 Qu'est ce que le GEM ?
     - 3.2.2 Orientation pédagogique

1. Choix du sujet

Etait venu le temps de choisir un sujet de mémoire. Divers sujets, dans des domaines tout aussi variés, se présentaient à moi. Une seule exigence : trouver un sujet qui marierait ma formation toute nouvelle en informatique, avec celle, un rien plus ancienne, en mathématiques.

A cette époque, je commençais à nouveau à envisager un avenir professionnel dans l'enseignement.

Or, l'ordinateur, l'enseignement de et par l'informatique prenaient sans cesse plus de place dans les écoles. Quoi de plus tentant alors que de travailler en EAO (Enseignement Assisté par Ordinateur).

Sans trop savoir ce qui se cachait sous ce sigle, j'avais pourtant l'assurance qu'il me permettrait de travailler en informatique, sans exclure ni les mathématiques, ni l'enseignement.

Lors d'une réunion rassemblant plusieurs membres du corps enseignant intéressés par la mise au point de programmes d'EAO, j'ai rencontré Monsieur Poncelet, régent en mathématiques. Nous avons choisi d'envisager les problèmes posés par l'apprentissage de la géométrie en troisième année du cycle secondaire.

Avant de développer le sujet de ce travail, il me semble important de préciser ce qu'est l'EAO, ses objectifs, sa situation et ses limites actuelles.

2. Vous avez-dit EAO ?

2.1. Objectifs

On a assistè et on assiste encore aujourd'hui à une vèritable "explosion" de l'intèrêt pour l'informatique. Bien sûr, l'informatique est "dans l'air". Elle s'insinue partout, aussi bien dans les entreprises que dans les foyers. La diminution du coût du matèriel renforce ce mouvement. Face à ce phènomène, les écoles se devaient de rèagir! Elles sont concernées, et par l'enseignement de l'informatique comme une discipline autonome, et par l'usage des possibilitès pèdagogiques offertes par cet outil.

L'ordinateur semble en effet être un outil possible, pour dèvelopper un enseignement plus individualisè et individuel. Un professeur devant une classe de 25 èlèves se doit d'adresser son discours à l'ensemble de la classe. Il ne peut se permettre ni d'avancer plus avant avec les intèressès , ni de consacrer tout son temps à provoquer l'èveil de ceux qui ne portent encore aucun intèrêt à la matière. L'introduction de l'outil informatique rend possible l'apprentissage individualisè de certaines matières ou parties de matière. Le rôle de l'enseignant est alors dèplacè : il était le centre du savoir, il sera centrè sur les èlèves. Le maître devra s'assurer que les enseignès ont en main tous les èlèments qui permettront l'appropriation du savoir proposè.

Nous sommes loin de rencontrer souvent de tels logiciels aujourd'hui.

2.2. Situation actuelle

Pour prendre connaissance de quelques didactitiels existant dèjà, je me suis rendue au dèbut de cette annèe acadèmique à un centre OSE (Ordinateur au Service de l'Enseignement). Là, j'ai pu observer le dèroulement d'une vingtaine de programmes pouvant servir à l' enseignement des mathèmatiques. La majoritè de ceux-ci est exclusivement dèmonstrative ou laisse une toute petite part d'initiative à l'enseignè. La question se pose alors du pourquoi d'une telle somme de travail et d'un tel rèsultat ?

On peut constater, même si cela commence à changer, le petit nombre de personnes ayant la formation nècessaire à la fois en pèdagogie, en informatique et dans la matière enseignèe. Des organisations telles que le CEFIS (CEntre pour la Formation à l'Informatique dans le Secondaire) tentent d'y remèdier en

donnant aux enseignants la possibilitè de se former en informatique.

De plus, pour tenter de mettre au point un didactitiel vraiment efficace (permettant l'apprentissage sans l'appui constant de l'enseignant), il faut fournir beaucoup de temps et de travail. J'en ai moi-même pris conscience tout au long de cette annèe et les divers auteurs consultès avancent des chiffres de 100 et même 150 heures de recherche et mise au point pour une heure de cours effectif.

Toutes ces heures de travail ont donnè jusqu'à prèsent des outils dont le coût peut être impressionnant et, qui plus est, ne sont guère transfèrables d'un enseignant à un autre. Je m'explique. Rares (je l'espère) sont les professeurs qui, pour donner leur cours, se rèfèrent à un seul manuel, le lisent tel quel à leurs èlèves et le referment en fin d'annèe en ètant persuadès d'avoir accompli leur tâche du mieux qu'ils le pouvaient.

Non, ils consultent divers auteurs , puisant dans l'un et l'autre les diffèrents èlèments qui leur permettront de donner leur cours tout en y apportant une touche personnelle. Or, les didactitiels existants, semblent tellement impènètrables aux enseignants que ceux-ci ne peuvent y apposer leur griffe. Ils doivent tout prendre ou tout laisser, avec comme consèquence immèdiate, la sous-exploitation de programmes qui ont nècessitè tant et tant d'heures de travail.

Trop souvent, jusqu'à prèsent, les notions d'enseignement assistè par ordinateur et d'enseignement programmè ont ètè confondues. Certains ([1] et [4]) font la distinction entre trois composants de l'EAO :

a) la matière enseignée ou le contenu

c'est-à-dire une masse de points de repère, de connaissances, de documents, ... C'est le temps de l'analyse du problème et de la fixation des objectifs. Une fois de plus, on peut douter de ce que, si cette ètape n'est pas rèalisèe, le programme mis au point puisse avoir l'efficacitè attendue.

b) les èlèments de dialogues constitutifs de l'EAO

c'est-à-dire les questions prèvues, les rèponses attendues, les rèactions à ces rèponses. C'est l'ensemble de ce que l'on appelle ècran, èlèment de scènario,...

c) le déroulement proprement dit d'une session d'EAO

c'est-à-dire une succession d'éléments de scénario, choisis dynamiquement, au fur et à mesure du déroulement de la session, en fonction des connaissances déjà acquises et à acquérir. Cette dernière idée suppose donc l'analyse des réponses et leur traitement spécifique. C'est en fonction des réponses correctes ou incorrectes que le déroulement de la session s'effectuera. C'est ici qu'apparaît l'individualisation du processus d'apprentissage.

Certes, l'enseignement programmé est réalisable par ordinateur, mais il ne représente qu'une petite partie des possibilités d'assistance à l'enseignement. C'est réduire l'ordinateur à être un "tourneur de pages".

Cela étant dit, je ne veux pas en rester à une noire description de la place actuelle de l'ordinateur dans les écoles. Cet outil apporte déjà beaucoup :

- par la démystification de l'ordinateur durant l'apprentissage scolaire, ces futurs adultes ne seront plus habités de la crainte du monde d'aujourd'hui, face à un outil qu'ils ne comprennent ni ne maîtrisent (ou si peu).
- le bon usage de programmes de simulation de phénomènes physiques semble de nature à combler le fossé qui existe aux yeux des élèves, entre les mathématiques enseignées au cours de mathématique et celles utilisées au cours de physique.
- ...

2.3. Que nous réserve l'avenir ?

Sans être ni Madame soleil, ni une pédagogue avertie, mais au vu du travail effectué cette année, je crains que si l'on ne se dirige pas vers une collaboration de plus en plus riche entre pédagogues, enseignants et informaticiens, le risque soit grand de passer à côté de l'apport pédagogique que peut offrir l'ordinateur.

Des progrès "techniques" restent à faire (facilités de gestion d'écran, traitement des réponses, ...). Mais surtout, alors que les unités de pédagogie semblent ouvrir leurs recherches à l'informatique, n'est-il pas temps de réveiller l'attention des informaticiens ? L'EAO n'est-il pas injustement le parent pauvre et délaissé dans la formation de ceux-ci ?

3. Le sujet

3.1. Introduction du sujet

Monsieur Poncelet, enseignant en troisième année du cycle secondaire, se trouve à chaque rentrée scolaire confronté au même problème :

> une classe formée d'élèves en provenance de diverses écoles, de différentes formations. Bien souvent, même la base fait défaut chez certains.

Il ne peut bien sûr reprendre toute la matière en détail avec toute la classe, ni travailler en particulier avec chaque élève pour combler les vides ou lacunes. C'est pourquoi, il est notamment intéressé par la mise au point d'un programme qui permettrait à chaque enfant de se mettre à jour en géométrie. La matière concernée par ce projet est donc l'identification des transformations du plan citées ci-dessous :

> translation,
> rotation,
> symétrie orthogonale,
> symétrie centrale,
> et homothétie.

Cette découverte suppose tout d'abord la mise au point des définitions des différentes transformations et la mise en évidence de leurs caractéristiques. Ensuite, l'élève doit être capable, non seulement de reconnaître le mouvement qui a transformé une figure en son image, mais aussi de construire lui-même l'image d'une figure donnée par une transformation donnée.

Monsieur Poncelet désire s'inspirer pour cet enseignement des recherches poursuivies par le GEM (Groupe d'Enseignement Mathématiques) et du rapport publié par celui-ci : " L'Archipel des Isométries ".
Qu'en est-il exactement ?

3.2. L'Archipel des Isométries.

3.2.1. Qu'est ce que le GEM ?

Le GEM est composé de régents et licenciés enseignant dans le secondaire, d'étudiants de dernière année de licence en mathématiques ainsi que d'enseignants et chercheurs universitaires. Ce groupe siège à Louvain-la-Neuve. Son activité principale est d'enseigner les mathématiques dans le secondaire. (Il en résulte que de nombreux élèves sont associés à ces travaux).

Son deuxième objectif est d'essayer de rendre service aux enseignants en publiant des dossiers variés sur l'enseignement mathématique.

"L'Archipel des Isométries (essai de redécouverte)" est l'un de ces dossiers.

3.2.2. Orientation pédagogique du GEM.

Pour situer l'orientation choisie par ce groupe, je vous livre quelques extraits de l'introduction du dossier "L'Archipel des isométries".

" Pourquoi le sous-titre : Essai de redécouverte ?
Nous avons essayé que chaque élève passe, par un effort personnel, du champ de ses appréhensions familières à la structure théorique."

" La façon d'enseigner décrite dans ce livre consiste à proposer à la classe une suite de situations problématiques ouvertes, conduisant par palier à la théorie mathématique. Le professeur introduit dans le travail de l'élève le moins possible de contraintes scolaires extrinsèques. ... Il est important que le contrôle de la situation passe en partie et très tôt à l'élève : l'apprentissage s'en trouve renforcé."

" Le professeur est présent tout de même, mais différemment. Il répond le plus souvent aux questions par d'autres questions. Il provoque aux bons moments des synthèses au cours desquelles le savoir est dûment rangé dans les cahiers et la mémoire."

" On dit aussi que " Ca prend beaucoup trop de temps " et que " C'est si difficile déja quand l'exposé est clair et bien ordonné " ... Pourtant, c'est seulement quand l'élève s'investit dans la recherche qu'il acquiert des connaissances profondes."

Telle est l'optique à la base de toute recherche de ce groupe.

Le chapitre suivant décrit la mise au point d'un scénario possible d'exécution du programme recherché, inspiré de la démarche proposée par le GEM.

CHAPITRE II : MISE AU POINT D'UN SCENARIO

1. Les sous-objectifs
2. Découverte de différents mouvements
   - 2.1 Démarche proposée
   - 2.2 Quelles figures proposer ?
   - 2.3 Comment offrir un calque ?
   - 2.4 Classement des différents cas étudiés
3. Caractérisation des différents mouvements
   - 3.1 La translation
     - 3.1.1 Une approche possible
     - 3.1.2 Remarques sur cette approche
   - 3.2 La rotation
     - 3.2.1 Une approche possible
     - 3.2.2 Le centre d'une rotation
   - 3.3 L'homothétie
   - 3.4 Remarques
4. Comment construire ?
   - 4.1 Première proposition : des primitives
   - 4.2 Seconde proposition : des outils
   - 4.3 Gestion des erreurs
5. Nécessité de points de rappel

1. Scénario proposé : Les sous-objectifs

La première étape de cette découverte des translations, rotations, symétries orthogonales, symétries centrales et homothéties est basée sur l'observation. Les élèves ont à découvrir, à travers des exemples , différents types de mouvements dans le plan, à les identifier et à les grouper.

Ensuite, après avoir donné un nom à chaque classe, il s'agit de mettre en évidence les éléments caractéristiques de chacune d'elles et de découvrir les renseignements nécessaires et suffisants pour identifier un seul de ces mouvements. Et ce pour obtenir comme résultat une définition de chaque transformation du plan étudiée.

Constater ne suffit pas ! L'étape suivante est celle de la construction par l'élève! "Etant donné une figure, en tracer l'image par une transformation donnée" tel est le but de la troisième partie.

Enfin, l'étude des figures symétriques, la découverte des éventuels centres et axes de symétrie d'une figure devrait clôturer ce travail. Cette étape apparaît comme une extension possible, mais elle n'a finalement pas été réalisée.

Tâchons maintenant de voir comment chaque étape a été abordée.

2. Découverte de différents mouvements

2.1. Démarche proposée

Il s'agit d'identifier les translations, rotations, symétries orthogonales, symétries centrales et homothéties. Pour ce faire, il est demandé aux élèves d'analyser quelques cas de transformation d'une figure en trait plein en son image en trait interrompu. En fait d'analyse, l'enseigné doit découvrir comment on peut obtenir l'image à partir de la figure. Ensuite, il doit grouper les cas qui se ressemblent.

Les outils pédagogiques disponibles pour ce travail sont le calque, le papier (pliage), le compas, la latte, ... Bien sûr, l'écran de l'ordinateur ne peut être plié. De même, on ne peut y tracer ni avec la règle, ni avec le compas. Nous verrons plus tard comment offrir ces outils dans le cadre d'un enseignement par ordinateur.

2.2. Quelles figures proposer ?

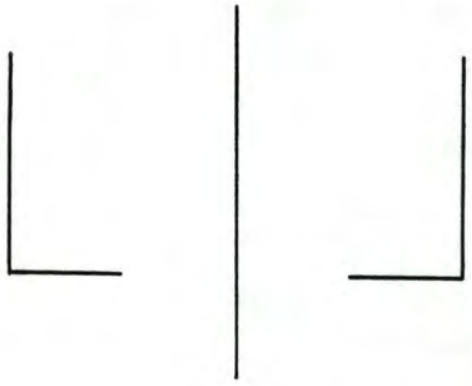
Pour ce qui est du graphisme, l'imagination du programmeur garde son entière liberté. Je voudrais plutôt dans ce paragraphe mettre en évidence différents points importants. Pour ce faire, je prends l'alphabet comme exemple de famille de figures et je m'inspire des travaux et expériences réalisés par le GEM.

Eviter la symétrie.
- - - - - - - - -

Parmi tous les couples (figure, image) proposés, on veillera à insérer des exemples de chaque transformation du plan étudiée dans ce projet.

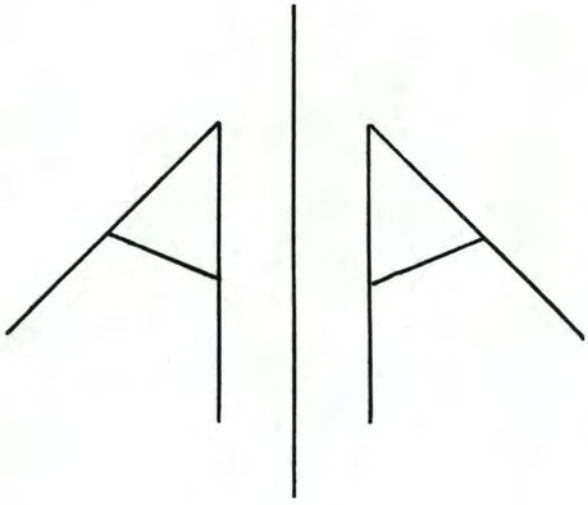
Dans un premier temps, il est bon d'éviter les lettres possédant des propriétés de symétrie :

- Un premier effet de ce choix est qu'une telle lettre perd son identité de lettre si on lui applique une symétrie orthogonale.



Cette perte d'identité facilite la distinction entre déplacement et retournement.

- De plus, l'unicité de la transformation du plan à découvrir n'est pas assurée si l'on travaille avec des lettres possédant des propriétés de symétrie.



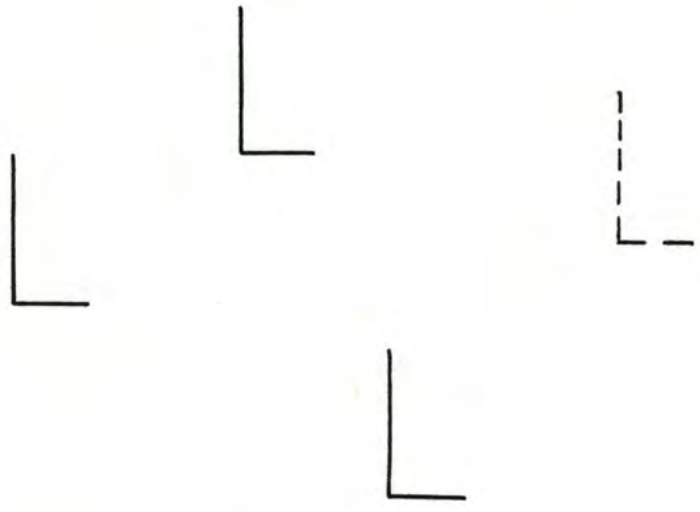
Principe d'économie.
- - - - - - - - - - -

Après l'étape du tâtonnement, très souvent, un seul et même mouvement est évoqué par tous les élèves. Il s'agit du mouvement "le plus simple" qui porte la lettre sur son image. Illustrons cette affirmation par un exemple. Soit le couple image-figure suivant :

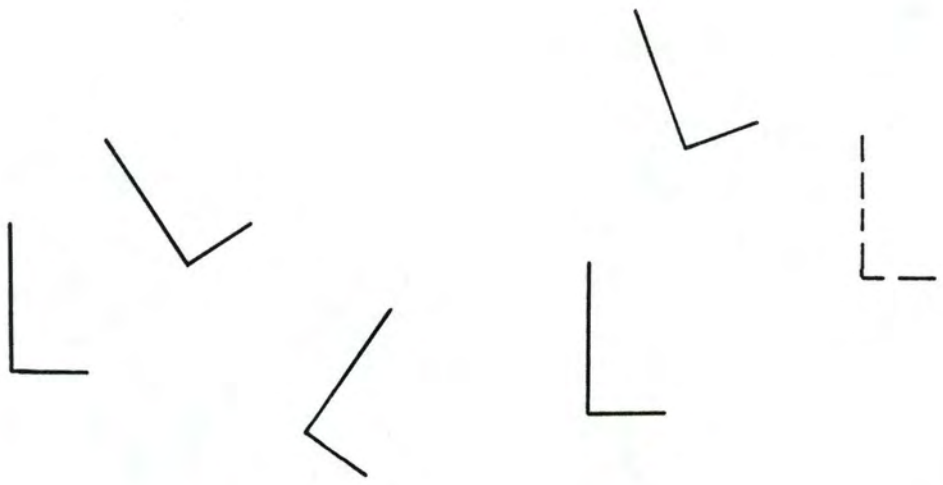
C'est, dans la plupart des cas, un mouvement de translation rectiligne envoyant la lettre en trait plein sur celle en trait interrompu qui est proposée comme solution.

Pourtant, au lieu de voir le mouvement de cette façon, les élèves auraient pu proposer :

soit un mouvement de translation non rectiligne



soit un mouvement qui n'est même pas une translation.



En effet, seules les positions initiale et finale importent. A priori, rien n'empêche un élève à l'imagination galopante de concevoir de tels mouvements. Pourtant, pour tous les exemples simples, on peut vérifier que

PARMI TOUS LES MOUVEMENTS POSSIBLES ,
C'EST LE PLUS IMMEDIAT,
LE PLUS COURT QUI EST PERCU.

Le paragraphe suivant donne des exemples où cette perception du mouvement le plus simple est contrariée.

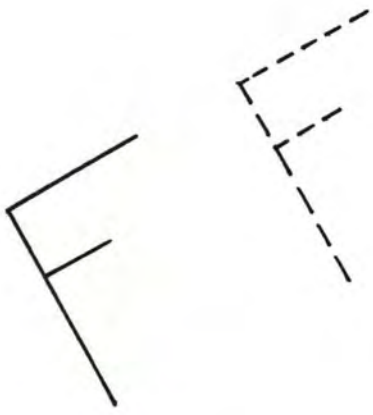


fig 1.

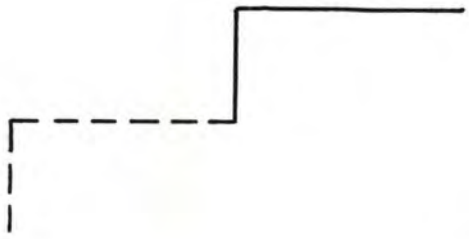


fig 2.

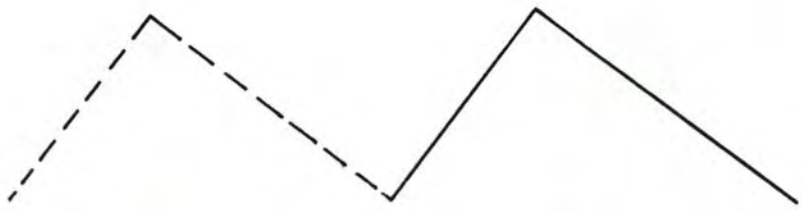
Alors que les élèves réalisent le cas (1) par une translation rectiligne, dans un premier temps, beaucoup résolvent le second en composant deux translations orthogonales dans les directions fournies par les branches du L.

Ces exemples mettent en évidence l'importance de la forme de la figure et du regard de l'utilisateur :

- d'une part les deux branches parallèles du F conduisent le regard dans la direction-même de la translation cherchée,
- d'autre part le L est formé de segments qui dévient le regard dans des directions dont aucune n'est celle de la translation.

Remarque :
Avant de poursuivre, notons que dans le deuxième cas, il est important d'intervenir dans la recherche de l'élève, en lui demandant de résoudre le cas proposé par un seul mouvement.

Pourtant, si l'on dessine les deux L dans l'orientation suivante, un seul mouvement de translation est évoqué.



Ainsi, il existe d'autres éléments perturbateurs que la figure elle-même. Il semblerait que ce soient les bords de la feuille (ou de l'écran). Remarquons que cette influence se retrouve également dans le troisième cas, où le mouvement de translation est parallèle au bord inférieur de la feuille, ce qui en facilite la découverte.

En résumé, l'application du principe d'économie est facilitée ou perturbée :

- par la forme de la figure

et/ou

- par la relation de la figure à son environnement immédiat.

Le paragraphe suivant précise l'influence de la forme de la figure.

Influence de la figure.

Prenons à présent l'exemple de rotations.

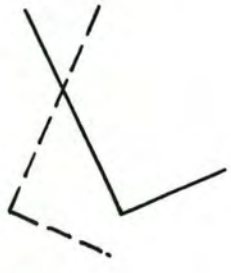


fig 1.

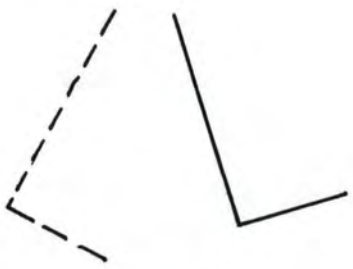


fig 2.

Une rotation comme celle du cas (1) est facilement perçue, parce que son centre est sur la figure ou alors, comme dans (2), parce que celui-ci est à l'intersection du prolongement d'un segment de la figure et de son image.

Par contre, la situation est nettement moins évidente dans les cas où le centre n'occupe aucune position privilégiée par rapport à la figure et à son image.

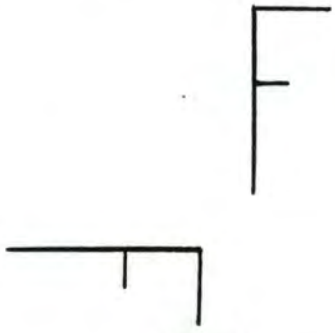
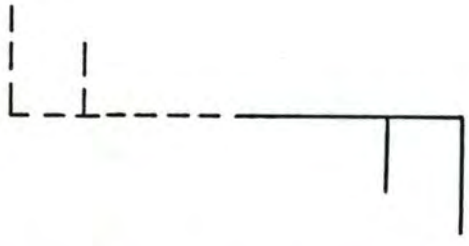


fig 3.

Le mouvement de rotation qui amène les figures l'une sur l'autre est alors difficilement perçu. Une fois encore, il est temps de rappeler à l'élève, s'il décompose cette rotation en plusieurs mouvements, qu'il doit résoudre le cas en un seul mouvement. Se pose alors le problème de déterminer "ce point", "ce centre".

Or, sur des exemples simples (centre sur la figure,...), l'élève perçoit cette notion de "point privilégié". Personnellement, je préfère me limiter maintenant à des cas simples, postposer la découverte de cette construction et résoudre ce problème après avoir mis en évidence les caractéristiques des différents mouvements.

De même, une symétrie centrale est plus facilement découverte quand son centre appartient à la figure (cas 4).



En conclusion, l'influence de la forme de la figure peut se résumer comme suit :

soit elle attire l'attention vers les éléments canoniques de l'isométrie et donc en facilite l'identification (1)

soit au contraire elle en détourne le regard et contrarie la perception de l'isométrie.

(1) Les éléments canoniques d'une isométrie désignent les données qui la déterminent.
Pour une translation : direction, sens et mesure
rotation : centre et angle
sym. orthogonale : axe
sym. centrale : centre
homothétie : centre et rapport

Pour être vraiment complète à ce sujet, je voudrais dire quelques mots sur le principe d'économie élargi.

Le principe d'économie élargi

Dans le cas d'une symétrie centrale, on peut percevoir le mouvement soit comme une rotation d'un demi-tour dans un sens ou dans l'autre, soit comme la composée de deux symétries orthogonales d'axes perpendiculaires,... Ainsi, la symétrie centrale est réalisable par plusieurs mouvements simples et aucun d'eux n'est, de manière évidente, plus simple que les autres. Encore une fois, c'est presque toujours l'un de ces mouvements simples qui sera perçu par l'élève.

De la même manière, si maintenant nous introduisons des figures symétriques (que j'avais rejetées au début de cette approche), les cas proposés ne sont plus résolus de manière unique. A nouveau, plusieurs mouvements aussi simples les uns que les autres sont possibles.

Ainsi, d'une part la nature de l'isométrie (cfr. symétrie centrale) et d'autre part la forme de la figure (cfr. figure symétrique) amènent à élargir le principe d'économie.

LE MOUVEMENT PERCU EST UN MOUVEMENT SIMPLE, MAIS VARIABLE D'UN ELEVE A L'AUTRE ET D'UN MOUVEMENT A L'AUTRE.

Choix des figures

Ainsi, pour en revenir à l'éventail des couples (figure, image) à proposer, il est bon de tenir compte de toutes ces

remarques et de proposer les cas à résoudre selon un crescendo de difficultés.

2.3. Comment offrir un calque ?

L'objectif de l'outil calque est de permettre la découverte du passage d'une figure à son image.

Sur les bancs de l'écolier, le calque est une feuille qui permet de voir par transparence une figure A et son image A'. Sur cette feuille, l'élève recopie la figure A. Dans la position initiale, par superposition, la figure du calque coïncide donc avec la figure A du cas à analyser. Lorsque le calque est déplacé, la figure se déplace également. Ainsi, par superposition, on voit 3 figures. La situation finale est atteinte lorsque la figure du calque coïncide avec l'image A'.

Comment offrir cette possibilité de visualiser le mouvement à l'aide de l'ordinateur ?

1ère possibilité : offrir différentes primitives telles que

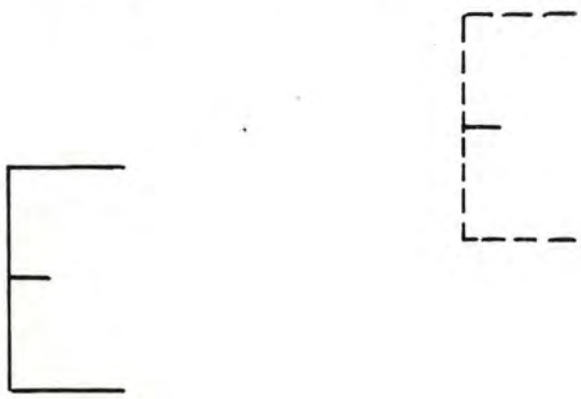
- monter
- descendre
- à gauche
- à droite
- tourner
- retourner
- grandir
- diminuer
- ...

Les primitives seraient statiques. i.e. si j'appelle la primitive "à gauche", le dessin se déplace de N points vers la gauche, N étant fixé.

Inconvénients : Il est demandé à l'élève de résoudre les cas qui lui sont soumis par un seul mouvement. Or, ce système l'oblige à décomposer tout mouvement en une répétition de mêmes petits mouvements.

Pire encore, ce système ne permet pas toujours de passer de la figure à l'image par un seul type de petits mouvements et renforce donc l'élève dans cette tendance à décomposer le mouvement dans certains cas (cfr. principe d'économie ....).

Il suffit de prendre l'exemple suivant :



Même si l'élève perçoit un seul mouvement,(la translation rectiligne) il est obligé de décomposer en une translation horizontale et une translation verticale.

De façon plus évidente encore, il n'est guère significatif de toujours tourner d'un même angle et autour d'un seul point fixé une fois pour toutes.

Cette idée est donc à transformer.

2ème possibilité : offrir ces mêmes primitives, mais avec la possibilité pour l'utilisateur de paramétrer.

Inconvénients : à ce stade de leur voyage dans le pays des transformations du plan, les élèves découvrent par l'observation et ils n'ont pas encore mis en évidence de façon précise et correcte les éléments canoniques de chaque transformation.
Il est donc trop tôt pour introduire les paramètres numériquement.

Le mieux est, je pense, la recherche d'une solution médiane, solution qui rendrait à l'écran le mouvement continu réalisé par l'élève avec un calque.

Prenons le cas de la translation. L'usage d'un JOYSTICK permet de déplacer la figure dans toutes les directions souhaitées et ce, sans qu'il soit nécessaire pour l'utilisateur de préciser ni la direction, ni le sens, ni la longueur du vecteur de translation.

Pour l'homothétie, une solution est d'offrir la possibilité de se positionner sur un point de la figure et de "tirer" pour faire grandir ou de "pousser" pour réduire. Cette possibilité n'exige pas la prise de conscience de l'existence d'un point fixe mais restreint de

beaucoup les homothéties réalisables.

Avec ce système, pour la rotation, il est nécessaire d'avoir déjà (et par un autre moyen) la perception de l'existence d'un point statique. Ayant déterminé ce point, un PADDLE offre la possibilité de faire tourner la figure dans un mouvement continu et sans devoir introduire la notion d'angle de la rotation.

Quant à la symétrie orthogonale, elle ne peut être réalisée de façon acceptable que par rapport à un axe non fixé dans le programme. Cette approche nécessite donc la connaissance de l'existence de cette droite particulière et le moyen de la découvrir, ce qui n'est pas du tout le reflet de la situation que nous supposons être.

Ainsi, cette notion de calque doit encore être approfondie et améliorée si l'on veut offrir à celui qui apprend un outil rendant possible une découverte des transformations du plan.

Remarque sur le calque

Offrir la possibilité de "tramer" le calque est une première introduction au fait que les transformations du plan déplacent non seulement les points appartenant à la figure, mais aussi tous les points du plan. En effet, lors de l'application d'une des primitives définissant le calque, s'il y a une trame, on voit des points, n'appartenant pas au motif étudié, participer au mouvement.

2.4. Classement des divers cas étudiés.

Je n'ai pas étudié vraiment ce problème. Tout reste à faire. Je voudrais juste relever le cas de la symétrie centrale. Il faut faire découvrir à l'élève que c'est un cas particulier des rotations, et un cas particulier des homothéties.

3. Caractérisation des différents mouvements

Après avoir observé, classé, nommé différentes transformations du plan, nous partons maintenant à la recherche de définitions. Il s'agit de dégager la même règle de transformation pour toutes les translations, puis la même règle de transformation pour toutes les rotations, ... Ce problème est abordé classe par classe.

L'ensemble de ce paragraphe se présente comme suit :

Je commence par la présentation de l'ébauche d'une approche possible de la translation, suivie d'une définition rassemblant les éléments caractéristiques. Ensuite, les différents approfondissements à réaliser sont répertoriés.

Le même schéma est appliqué à la rotation, la symétrie orthogonale, la symétrie centrale et l'homothétie. Est ajoutée à ce schéma de base la recherche du centre d'une rotation.

Finalement, une mise en évidence des différences existant dans la nature-même des éléments canoniques est proposée.

3.1. La translation

3.1.1. Une approche possible

Le but est de faire découvrir les éléments canoniques de la translation. Le principe proposé est, après la découverte de l'un de ces éléments, de demander si cela est suffisant. Si la réponse est oui, alors que les renseignements sont en fait insuffisants, un contre-exemple est proposé afin d'amener l'élève à prendre conscience de son erreur.

En tout premier lieu, il est bon d'offrir la possibilité de revoir à l'écran les différents cas placés par l'élève lui-même, dans cette classe.

Ensuite, on peut poser les questions suivantes :

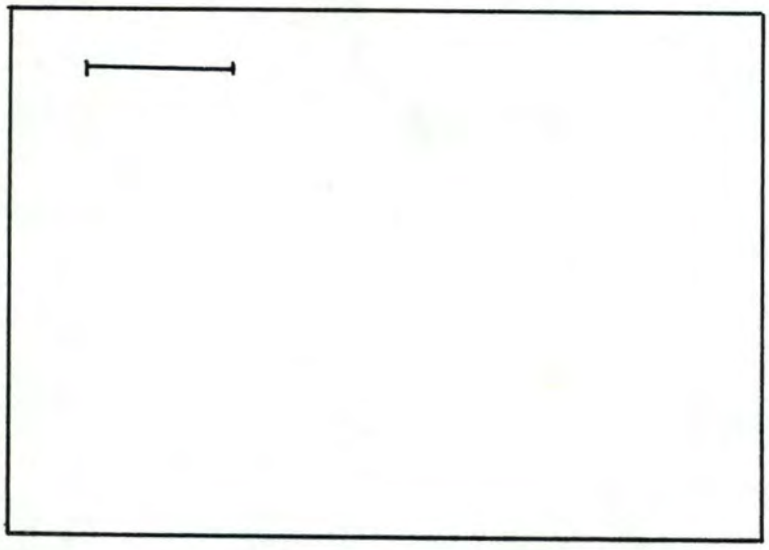
Quelles sont les indications minimales à donner pour caractériser une translation?

Qu'est-ce qui permet de distinguer une translation d'une autre ?

une distance (1) :

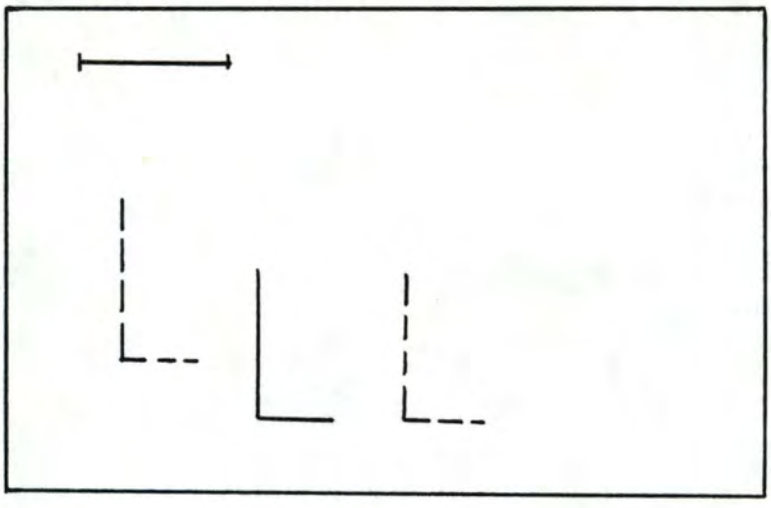
sur l'écran, apparaît alors, dans un endroit privilégié, un segment. L'utilisateur peut faire varier la longueur de ce segment.

Soit l'écran



"Est-ce suffisant ?"

Si la réponse est OUI, donner un contre-exemple, utilisant la longueur du segment.



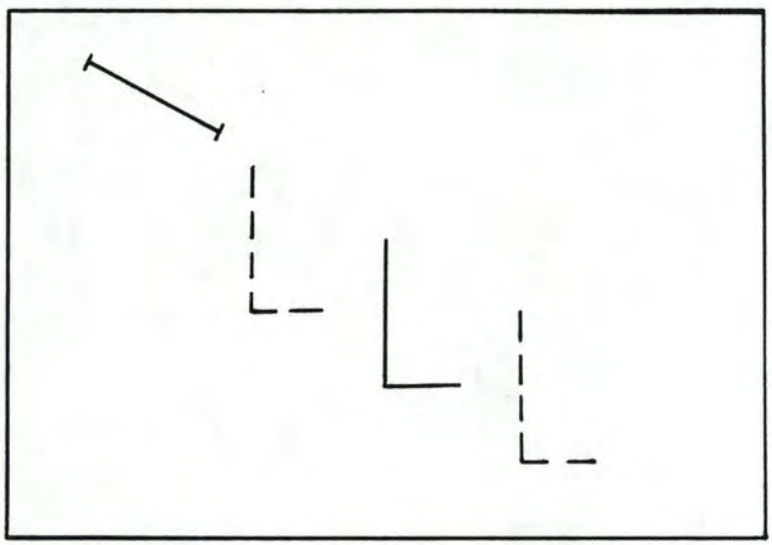
Si la réponse est NON, poser la question

"Que doit-on donner comme renseignement supplémentaire ?"

(1) La notion de distance est une notion théorique connue par l'élève

une direction :

La possibilité de "faire tourner" le segment est alors offerte. Puis, la séquence d'actions présentée pour la distance est répétée, avec bien sûr adéquation du contre-exemple.



un sens :

Le segment devient flèche. Le même processus est à nouveau appliqué.

Une fois que l'élève a découvert que ces trois caractéristiques sont nécessaires et suffisantes pour décrire une seule translation, une définition lui est proposée.

Vu la démarche suivie, il n'y a normalement à ce stade guère de problèmes de mémorisation de ces éléments puisque c'est l'enseigné lui-même qui les a découverts.

3.1.2. Remarques sur cette approche

Le paragraphe précédent décrit l'ébauche d'une approche possible. Il est à noter que beaucoup de recherches doivent encore être réalisées. Je voudrais en recenser quelques-unes ci-dessous :

1. L'ordre dans lequel les différents éléments canoniques de la transformation sont découverts ne peut être statique. Le programme qui guiderait cette découverte doit pouvoir gérer leur introduction dans n'importe quel ordre.

2. La gestion des réponses fournies par l'enseigné n'est pas un problème simple. Il est tout d'abord nécessaire de recenser le vocabulaire acceptable.

   ex : distance, écart, ...

   Il faudra peut-être aussi tenir compte d'éventuelles fautes d'orthographe.

   Si la réponse est acceptée, tout va bien, il suffit de passer à l'étape suivante. Mais si elle est incorrecte, il ne suffit pas de dire NON. Il est nécessaire de justifier ce non, ou mieux encore, d'amener l'élève à la découverte de l'erreur.

3. Il ne faut envisager une gestion des réponses que si réponses il y a. Que faire si l'élève ne répond rien ? Comment le guider ? Combien d'essais-erreurs lui permettre ? ...

Signalons que ces mêmes remarques s'appliquent au schéma général, et non pas particulièrement à l'étude de la translation.

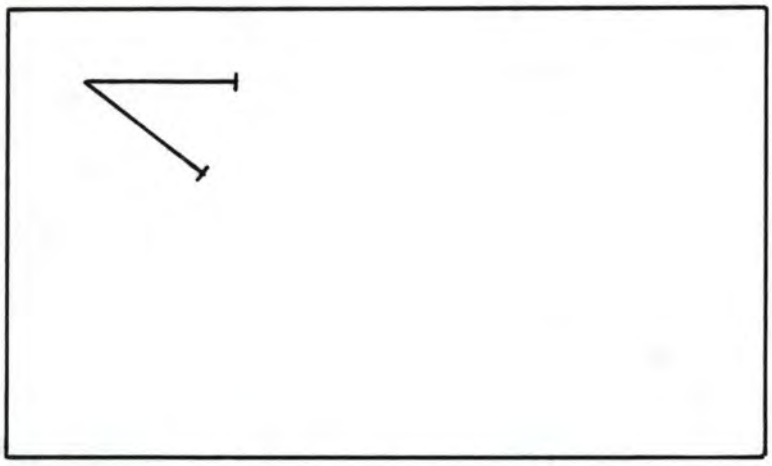
3.2. La rotation

3.2.1. Une approche possible

Les différentes notions à mettre en évidence sont :

- le point fixe
- l'amplitude

Deux segments de droite sont affichés à l'écran et il est possible de les faire tourner.



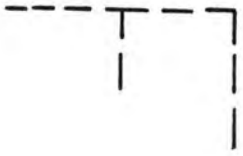
- le sens de la rotation

Jusqu'à présent, dans les différents cas proposés à l'observation de l'apprenant, le centre de la rotation est sur la figure (J'ai expliqué longuement pourquoi dans le paragraphe 2.2). Le point fixe est dans ce cas découvert sans méthode : il apparaît clairement aux yeux de l'observateur. Le moment est venu de présenter des cas de rotations où le centre n'est plus sur la figure et de poser le problème de la recherche de celui-ci.

3.2.2. Le centre d'une rotation

Enoncé du problème

"Soient deux figures. On sait que la seconde a été obtenue à partir de la première par une rotation. Trouve le centre de cette rotation".



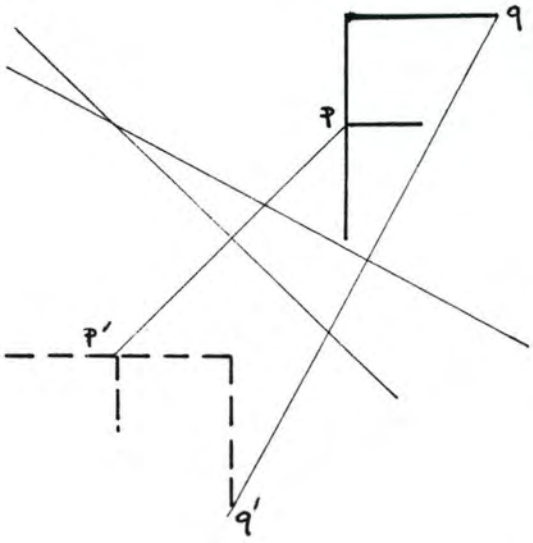
La question n'est pas simple. Pour en faciliter la résolution, un problème préliminaire est posé: "Soient deux points p et p'. Trouve le centre d'une rotation qui envoie p sur p'. Trouve les centres de toutes les rotations qui envoient p sur p'".

Par tâtonnement et avec l'aide de questions telles que

- je t'impose le rayon du cercle qui doit passer par p et p', trouve son centre.
- puis-je imposer n'importe quel rayon ?
- combien de rotations envoient p sur p' ?
- ...

l'élève découvre peu à peu qu'il existe une infinité de centres, que ceux-ci forment une droite perpendiculaire au segment [p,p'] et passant par son milieu. Cette droite est appelée médiatrice de [p,p'].

Le préliminaire étant résolu, un bref raisonnement permet de solutionner le problème initial de recherche du centre d'une rotation. Il suffit de choisir deux couples de points (point original, image), par exemple (p,p') et (q,q'). La médiatrice de [p,p'] est l'ensemble des centres des rotations qui envoient p sur p', la médiatrice de [q,q'] est l'ensemble des centres des rotations qui envoient q sur q'. La rotation cherchée envoie p sur p' et q sur q'. Son centre est donc à l'intersection des médiatrices de [p,p'] et [q,q'].



Il suffit de proposer l'outil CALQUE à l'élève pour vérifier que le point trouvé est effectivement le centre de la rotation appliquant F sur son image. On peut également proposer de tracer une troisième médiatrice.

3.3. L'homothétie

Le même schéma étant appliqué à la symétrie orthogonale, à la symétrie centrale et à l'homothétie, je ne les développe pas. Par contre, je voudrais présenter l'ébauche d'une recherche concernant la découverte du rapport d'une homothétie.

- Présenter des agrandissements ou des réductions

fig 1.

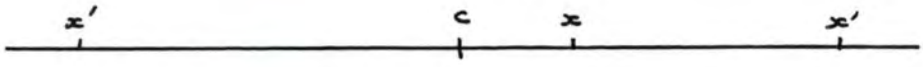
fig 2.

L'image est deux fois plus grande que la figure: on parle de rapport 2 dans le cas (1).

Par contre, dans le cas (2) l'image est deux fois plus petite que la figure: on parle de rapport 1/2.

- Faire découvrir la nécessité de signer ce rapport.

Soit un rapport r, deux points c et x. Où situer x' sur la droite cx pour que d(x',c) = r d(x,c) ?

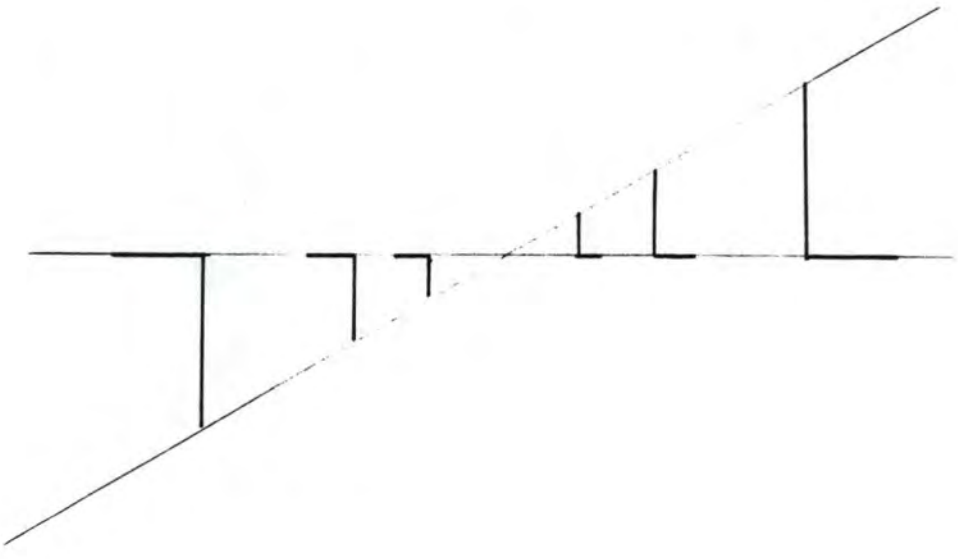


L'élève découvre qu'il y a deux possibilités. La nécessité de donner un renseignement supplémentaire si l'on veut définir l'image de façon unique apparaît : ce sera le signe du rapport.

- Proposer une série d'exercices de recherche de l'image d'un point, pour différentes valeurs du rapport.

  r>1 ; r=1
  0<r<1 ; r=0
  -1<r<0 ;r=-1
  r<-1

- Visualiser le passage du rapport par 0.



3.4. Remarques

1) A ce stade de la découverte des transformations du plan, il est nécessaire d'insister sur certaines différences:

   la translation est définie par un vecteur (longueur-direction- sens) mais est indépendante de la position de ce vecteur dans le plan.

   par contre, la position de l'axe d'une symétrie orthogonale ou du centre d'une rotation est un des éléments de leur caractérisation.

2) Pour chaque transformation du plan étudiée, il faut s'assurer que l'apprenant a découvert comment construire l'image d'une figure donnée.

4. Comment construire?

Ayant découvert les éléments canoniques des diverses transformations du plan étudiées et formulé leur définition, il s'agit maintenant de "mettre en pratique" toutes ces découvertes et de construire l'image d'une figure donnée par une transformation du plan donnée.

Dans un enseignement traditionnel, les outils à la disposition de l'enseignant et de l'élève sont la craie et le tableau, le papier, le crayon et la gomme, le compas, la latte ou la règle, le rapporteur...

J'ai cherché comment offrir cette possibilité de construction d'images de figures, dans le cadre d'un enseignement par ordinateur.

4.1. Première proposition: des primitives

La première idée développée est celle d'offrir des primitives que l'utilisateur puisse faire exécuter. J'ai alors passé en revue les différentes transformations afin de recenser l'ensemble minimum des fonctions à offrir pour permettre à l'utilisateur de construire l'image d'une figure.

L'utilisation de primitives permettant de dessiner sur un écran nécessite l'existence d'un curseur, afin de déterminer les éléments concernés par la primitive.

Une liste possible de telles primitives pourrait être celle-ci :

pour la translation :

- tracer la droite passant par 2 points déterminés (au moyen du curseur)
- tracer la droite parallèle à une droite donnée, passant par un point donné
- effacer le segment de droite compris entre deux points
- tracer le segment de droite compris entre deux points
- ...

pour la rotation :

- en un point d'une droite donnée, tracer la droite formant un angle d'amplitude donnée avec cette droite
- possibilité de reporter une longueur
- ...

pour l'homothètie :

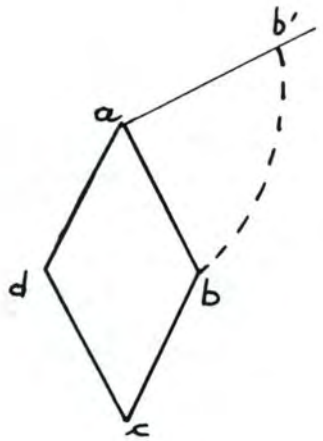
- tracer la droite passant par deux points
- tracer le segment déterminé par deux points
- tracer la droite parallèle à une droite déterminée et passant par un point déterminé
- ...

On procède de même pour les symétries orthogonales ou centrales.

Très vite il apparaît deux gros inconvénients:

- beaucoup de primitives (se ressemblant tout en étant différentes) sont nécessaires.

- ces primitives nécessitent parfois l'introduction de beaucoup de renseignements, et ce de façon répétitive.
  En effet :
  Soit le losange abcd. Construire son image par la rotation de centre a et d'angle égal à +90 degrés.



Pour rechercher l'image du point b, il faut utiliser la primitive :
"Tracer la droite formant avec une droite donnée un angle d'amplitude donnée"

Elle nécessite de préciser
- le point a
- la droite ab
- l'angle +90 degrés.

De même pour rechercher l'image de c, il faut introduire les données suivantes :

- le point a
- la droite ac
- l'angle +90 degrès.

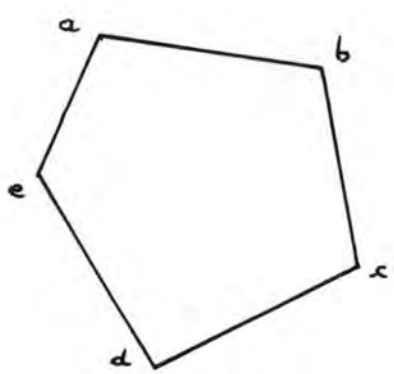
Idem pour d.

Il apparaît immédiatement qu'il y a une répétition très grande dans l'introduction des données. Nous imaginons sans peine la lassitude grandissante de celui qui a à construire l'image d'un polygone irrégulier à 12 ou 20 côtés !

Pour éviter cette répétition, il faudrait donner une mémoire aux primitives. C'est ainsi qu'est née la seconde (et dernière) proposition.

4.2. Seconde proposition : des outils

Si on offre à l'utilisateur un outil règle et un outil compas, il est aisé d'imaginer comment va se résoudre le problème de la répétition. Ainsi la règle est caractérisée par une direction, qu'elle conserve jusqu'à ce qu'on la modifie. Donc, pour construire l'image d'un polygone (abcde) par la translation de vecteur ab,



il suffit de donner à la règle, la direction ab, puis de se positionner en a, d'y tracer une droite ayant cette direction, puis de se rendre en b, puis en c, etc...

Je vous propose, à vous lecteur, de dénombrer les opérations qu'il aurait fallu fournir pour arriver au même résultat en utilisant la primitive : "tracer une droite parallèle à une droite donnée passant par un point donné"

De même,si nous reprenons l'exemple de la rotation, un compas caractérisé par la position de "la pointe sèche" et par un écartement permet de résoudre le cas du losange avec nettement moins d'introductions de données. En effet, les données "point a" et "angle +90" sont conservées par le compas.

J'arrête ici ma description des outils fournis. En effet, à ce stade du travail (i.e. une première approche de toutes les étapes de la découverte des isométries), je me suis aperçue qu'il ne serait pas possible de tout développer dans le cadre de ce mémoire. J'ai préféré choisir une partie précise et l'implémenter. Or, c'est justement l'éditeur graphique que j'ai choisi de réaliser parce que, tout en étant au coeur du problème, cette partie peut aussi faire un tout en devenant un outil permettant par exemple de se familiariser avec des constructions géométriques telles que la recherche du point milieu d'un segment ou le tracé d'une droite perpendiculaire à un segment, à l'aide d'un compas, ...

Mais avant de passer au chapitre suivant qui étudie ce problème en détail, je voudrais reparler de la gestion des erreurs et du scénario proposé!!

4.3. Gestion des erreurs

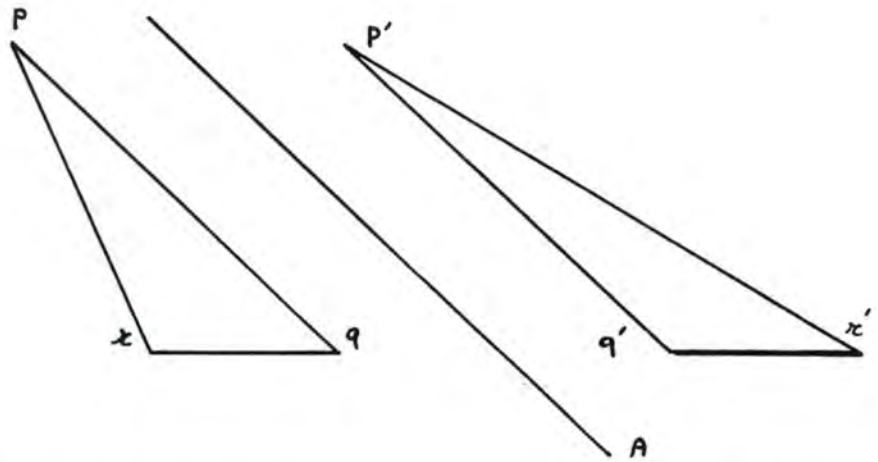
Encore une fois, toute la gestion des erreurs n'a pas été abordée ici.

La première idée, la plus classique, serait de mettre au point un ensemble de triples (énoncé-solution-explication si erreur). Le principal inconvénient de cette proposition est que la solution étant mémorisée, le programme compare la réponse donnée par l'utilisateur avec la solution proposée dans le triple. S'il y a équivalence, l'exercice "suivant" est proposé. Sinon, l'explication associée est proposée à l'élève.

Ainsi, quelque soit l'erreur, l'explication est la même puisque le seul renseignement est la coïncidence ou non des solutions et que l'explication est fonction du triple et pas de l'erreur.

Prenons l'exemple suivant:

Soit le triangle pqr. Tracer son image par la symétrie orthogonale d'axe A.



L'erreur commise est très fréquente. Pourtant, tout n'est pas incorrect. Les distances sont bien reportées de part et d'autre de l'axe.

Or dans l'optique des triples, l'explication proposée va reprendre toutes les caractéristiques de la symétrie orthogonale, avec le risque de ne pas aider l'élève : La définition de la symétrie orthogonale, il la connait!

Par contre, si l'erreur est détectée, une explication correspondante pourra être proposée. Apparaît ainsi la notion de type d'erreurs.

Les classes d'erreurs étant mises en évidence, il est possible d'envisager une analyse de la solution proposée pour

mettre en évidence le type de l'erreur commise, si erreur il y a. Ensuite, une explication appropriée et peut-être des exercices, permettant à l'enseigné de se familiariser avec le concept mal acquis, peuvent être proposés.

La recherche des types d'erreurs et leur analyse est un gros travail qui ne peut être réalisé qu'en collaboration avec plusieurs enseignants et en examinant beaucoup de travaux d'élèves. Cet objectif est cependant une extention possible et même souhaitée.

5. Nécessité de points de rappel

Le scénario tel qu'il a été proposé

- découverte des différents mouvements
- mise en évidence de leurs caractéristiques
- construction d'images

n'est que séquentiel. Rarement on a vu pareil processus d'apprentissage atteindre ses objectifs.

En effet lors d'un enseignement, en fonction des réponses fournies et des erreurs commises, sans cesse, des rappels de matières précédentes, ou de bases supposées connues sont nécessaires.

Ainsi, des points de rappel sont à ajouter au scénario proposé. Ces rappels doivent être fonction de l'erreur commise dans les réponses.

On peut même envisager d'aller plus loin encore dans la gestion du déroulement d'une session:

> L'enchaînement des diverses étapes, des rappels éventuels, serait lui-même confié à un programme, un coordinateur qui gérerait l'évolution de la découverte de la matière enseignée. Pour qu'un tel programme soit réalisable, il faut exprimer sous forme de structure de données la sémantique des connaissances à acquérir.

CHAPITRE III : EDITEUR GRAPHIQUE

1. Introduction
2. L'écran et le curseur
   - 2.1 Description de l'écran
   - 2.2 Description du curseur
3. L'outil GOMME
4. L'outil POINT
5. L'outil REGLE
6. L'outil COMPAS
7. Configuration

1. Introduction

Ce chapitre comprend l'analyse du problème soulevé en fin du chapitre précédent :

> Offrir la possibilité à l'utilisateur de construire graphiquement l'image d'une figure donnée par une transformation du plan donnée.

J'ai choisi de réaliser cet objectif sous la forme d'outils. Rapidement, la liste des constructions nécessaires à chaque transformation a mis en évidence trois outils :

- l'outil POINT permettant la gestion d'un curseur et le dessin du point désigné par celui-ci.
- l'outil REGLE caractérisé essentiellement par une direction et permettant le tracé de droites et demi-droites, ainsi que la gestion du curseur.
- l'outil COMPAS décrit par un point (centre) et un écartement. Il dessine des cercles, travaille avec des angles et gère le curseur.

Ces trois outils permettent de tracer l'image de toute figure par l'une des transformations étudiées. Pourtant, très vite, l'écran devient encombré. La nécessité d'un outil GOMME apparaît clairement.

Les outils sont définis par les primitives qu'ils offrent et les données qu'ils mémorisent. Pour respecter le désir de l'enseignant d'apposer sa griffe sur un didacticiel, j'offre la possibilité de restreindre ou d'étendre l'ensemble des primitives utilisables. Ainsi, au gré du professeur et/ou au fil de la matière, l'élève a à sa disposition une panoplie plus ou moins vaste et plus ou moins perfectionnée d'instruments.

Avant d'étudier chaque outil, il me faut définir l'écran sur lequel ces outils dessinent et les caractéristiques du curseur utilisé.

2. L'écran et le curseur

2.1. Description de l'écran

La page graphique est affichée en permanence.
Comment l'écran se présente-t-il ?

Un cadre définit les limites du déplacement du curseur. Ce cadre permet en outre de réserver une zone de l'écran pour les messages adressés à l'utilisateur et pour le mot-clé lui rappelant quel outil il utilise.

2.2. Description du curseur

Le curseur, qui dans la version actuelle est une flèche, identifie un et un seul point de l'écran, appelé PTCOURANT (point courant). Le mouvement de celui-ci est commandé soit par quatre touches clavier, soit par un JOYSTICK.

Par pitié pour les nerfs de l'utilisateur qui doit faire traverser tout l'écran au curseur, le déplacement de celui-ci ne peut pas toujours être point à point. Se pose alors un problème de précision !

Lorsque la gestion du mouvement se fait à partir du clavier, le curseur peut être déplacé vers la gauche, vers la droite, vers le haut ou vers le bas. La solution à la demande de précision est d'offrir le choix entre deux grandeurs de déplacement du curseur (grand-pas <-> petit-pas) ou plus exactement la possibilité de passer de l'un à l'autre.

Lorsque le mouvement est commandé par JOYSTICK, la position centrale du JOYSTICK correspond à la position du curseur. Le déplacement de la manette détermine la direction et la grandeur du mouvement de celui-ci. Ainsi, la vitesse de déplacement du curseur est fonction du mouvement de la manette. Une légère variation de celle-ci permet un positionnement précis du curseur sur l'écran.

Lorsque le curseur est au bord du cadre, si l'utilisateur commande un mouvement en direction du cadre, le curseur réapparaît de l'autre côté (ex.: disparition à gauche - apparition à droite, disparition en haut - apparition en bas et vice versa).

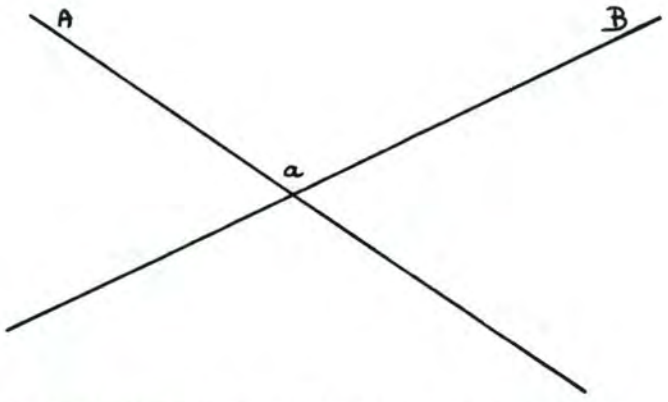
Dernière caractéristique du curseur : il peut écrire soit au crayon, soit à l'encre. Nous verrons dans le paragraphe décrivant l'outil GOMME, pourquoi cette distinction est faite. L'utilisateur peut à tout moment changer le mode d'écriture du curseur. Ce mode est, lui aussi, rappelé à l'utilisateur par un mot-clé écrit à l'écran.

En résumé, la gestion du curseur comprend :

- la gestion du déplacement de celui-ci soit par JOYSTICK soit par le clavier, avec possibilité de choisir la grandeur du déplacement,
- le choix du mode d'écriture (encre ou crayon).

3. L'outil GOMME

Une gomme assez rudimentaire serait réalisée en permettant à chaque primitive de tracer soit en blanc pour dessiner, soit en noir, couleur du fond pour effacer. Les problèmes surviennent dès qu'il y a intersection de traits.



Si je trace la droite A en couleur du fond afin de l'effacer, j'élimine également le point 'a' de l'intersection. Or, ce point appartient à B et doit être maintenu.

Un exemple de gomme très sophistiquée serait de retenir la liste des N dernières actions effectuées (N entier fixé) et de permettre soit d'annuler l'une de ces N actions, soit d'annuler tout ce qui a été exécuté après l'une d'elle.

Pour concevoir l'outil GOMME, je me suis à nouveau inspirée des moyens traditionnels : un crayon, une plume ou un rotring et une gomme.

Quelles sont les caractéristiques de travaux réalisés à l'aide de ces outils ?

Le crayon s'efface, l'encre pas. Alors, lorsque je dessine, pour éviter de devoir tout recommencer à la moindre erreur, je construis au crayon puis je recopie à l'encre.

L'outil GOMME offre donc trois possibilités :

- faire apparaître par clignotement les parties au crayon. Sur papier un trait au crayon est différent d'un trait à l'encre, sur l'écran pas. Je fais donc appel au clignotement,
- effacer ce qui est au crayon,
- mettre tout le dessin à l'encre.

Lorsque l'utilisateur fait appel à cet outil, le mot-clè GOMME apparaît à l'écran.

C'est pour permettre la distinction entre les traits au crayon et les traits à l'encre qu'il est nécessaire d'introduire deux modes d'écriture.

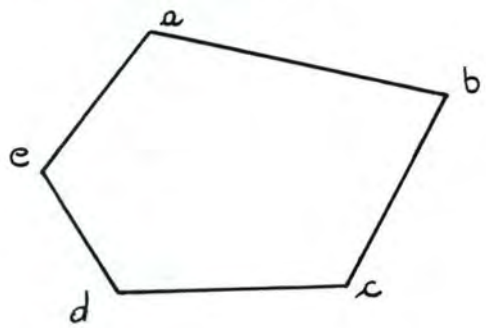
4. L'outil POINT

Cet outil , tout comme les outils règle et compas, permet la gestion du curseur (au sens où elle a été définie dans le deuxième paragraphe de ce chapitre). La particularité de ce mode est de permettre le dessin de PTCOURANT. Lorsque cet outil est utilisé, le mot-clè POINT apparaît à l'écran.

5. L'outil REGLE

Le premier but de l'outil est de tracer des segments de droite déterminés par deux points. La gestion du curseur est réalisable à partir de l'outil et il est donc possible d'identifier ces points.

Soit à tracer l'hexagone abcde



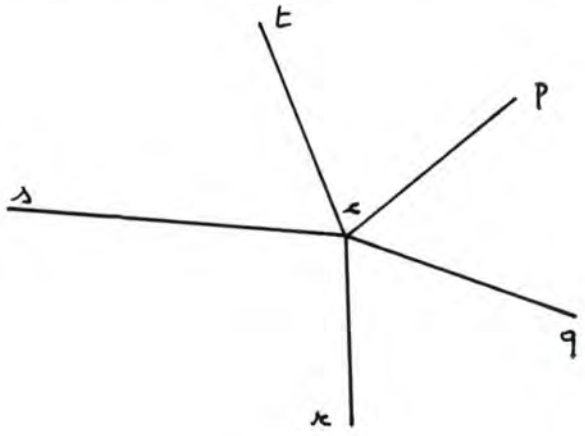
La suite d'actions pourrait être :

```
introduire (a,b)
tracer
introduire (b,c)
tracer
etc ...
```

Pour éviter de devoir répéter l'introduction des points, celui à partir duquel on a tracé la dernière droite est mémorisé. Appelons le PTCOUREG (point courant de la règle). Le segment est alors déterminé par PTCOURANT et PTCOUREG.

Reprenons l'exemple :
Le point a est introduit ! PTCOUREG=a. Ensuite, le curseur est positionné en b et la primitive est appelée. Le segment ab est tracé et PTCOUREG=b. Il suffit de se positionner en c, de faire appel à cette même primitive, etc...

Grâce à la mémorisation d'un point, tout polygone, toute séquence de segments de droite est aisément tracé. Par contre, considérons une figure telle que l'étoile par exemple :



Si nous voulons tracer ce dessin avec la primitive introduite, le plus avantageux est de procéder comme suit :

```
introduire p (PTCOUREG = p)
se positionner en c
tracer (PTCOUREG = c)
se positionner en q
tracer (PTCOUREG = q)
introduire r
se positionner en c
tracer (PTCOUREG = c)
etc...
```

Cette séquence n'est ni très logique, ni très agréable. Des manipulations pas du tout naturelles doivent être introduites. Une solution possible est de distinguer un autre point en le "marquant" PTMAREG (point marqué de la règle) à l'aide d'une primitive et d'offrir la possibilité de tracer le segment déterminé par PTCOURANT et PTMAREG.

L'étoile est alors tracée par la séquence d'instructions suivante :

```
marquer c
se positionner en p
tracer
se positionner en q
tracer
etc...
```

De plus, pour faciliter la construction de l'image d'une figure par une translation, il est possible de tracer une droite parallèle a une droite donnée passant par un point donné. Nous avons vu qu'un des principaux avantages apportés par les outils est leur mémoire. L'outil REGLE que l'on pourrait appeller "REGLE-RAPPORTEUR" ou "SUPER-REGLE" est caractérisé par une direction : celle de la dernière droite tracée.

Ainsi une primitive trace, par PTCOURANT, une droite de même direction que la règle. Remarquons aussi que cette primitive modifie PTCOUREG (puisqu'une droite est tracée).

Bien souvent, dans ce genre de construction, l'utilisateur n'a besoin que d'une demi-droite. Alors, plutôt que de tracer la droite entière, il peut à son gré, à l'aide de deux touches, choisir de dessiner la demi-droite droite, la demi-droite gauche ou les deux. Ceci permet d'alléger fortement l'écran de droites inutiles.

Enfin, pour que cet éditeur puisse servir pour d'autres constructions en géométrie et pour donner une certaine liberté à l'enseignant, cette primitive n'est accessible que si la configuration déterminée par l'utilisateur privilégié en donne l'autorisation (ceci permet, par exemple, d'utiliser aussi ce programme pour apprendre à tracer des droites parallèles à l'aide d'un compas).

Dans les mêmes conditions, une primitive traçant une droite de direction orthogonale à celle de la règle et passant par PTCOURANT est offerte. J'ai fait ici le choix, bien qu'une droite soit tracée, de ne pas modifier la direction de la règle. En effet, cette dernière primitive est introduite surtout pour la construction de l'image d'une figure par une symétrie orthogonale. Dans ce cas, l'utilisateur désire tracer plusieurs droites perpendiculaires à l'axe. Il lui suffit de donner la direction de l'axe à la règle et d'appeler la primitive autant de fois qu'il le faut. Ceci ne serait pas possible si la direction de la règle était modifiée. Par contre PTCOUREG est modifié.

Enfin, si la configuration le permet, l'outil REGLE peut dessiner PTCOURANT. Cette proposition évite de devoir retourner à l'outil POINT pour dessiner un point. (Notons également que lorsque cette primitive est offerte, l'outil POINT n'a plus de raison d'être.)

Il est dès lors nécessaire de compléter la définition de PTCOUREG :

PTCOUREG est le dernier point

soit qui a été dessiné,
soit à partir duquel on a tracé une droite.

En conclusion, l'outil REGLE est caractérisé par deux points (PTCOUREG et PTMAREG) et une direction. Il permet de

1) gérer le curseur;
2) marquer un point;
3) tracer le segment de droite joignant PTCOURANT à PTCOUREG;
4) tracer le segment de droite joignant PTCOURANT à PTMAREG;
5) tracer la demi-droite droite, la demi-droite gauche ou les deux, passant par PTCOURANT et de direction égale à celle de la règle;
6) tracer la demi-droite droite, la demi-droite gauche ou les deux passant par PTCOURANT et de direction orthogonale à celle de la règle;
7) dessiner PTCOURANT.

Les primitives 5,6 et 7 sont optionnelles. Rappelons également que lorsque cet outil est utilisé, le mot-clé REGLE apparaît à l'écran.

6. L'outil COMPAS

Tout comme les outils POINT et REGLE, la gestion du curseur est nécessaire afin d'identifier des points de l'écran.

Quelles sont les caractéristiques d'un compas ?

- tracer des circonférences,
- reporter des longueurs.

Appliquant le même principe que pour l'outil REGLE, en vue de minimiser les introductions de données, l'outil COMPAS a une "mémoire". Il est caractérisé par un centre et un écartement. Il est nécessaire d'offrir la possibilité de marquer un point. Appelons PTMARCOMP (point marqué du compas) le dernier point marqué dans l'outil COMPAS.

La question suivante est de savoir comment donner un écartement (nommé ECART) au compas. Il m'a semblé intéressant de pouvoir le faire

- numériquement, par introduction au clavier d'un entier positif;
- géométriquement, par une distance entre deux points, PTMARCOMP et PTCOURANT.

Le compas ainsi défini, permet de dessiner le cercle de centre PTMARCOMP et de rayon égal à ECART. Quant au report de distances, il est réalisé par la primitive suivante :

> "Renvoit en inverse (blanc si le point était noir et vice-versa) les deux points appartenant à la droite passant par PTCOURANT et de direction égale à celle de la règle, situés à une distance de PTCOURANT égale à ECART."

Dans le but d'alléger le dessin de constructions certes intéressantes, mais un peu encombrantes, il est également possible de demander le point milieu du segment [PTMARCOMP,PTCOURANT]. Cette primitive est optionnelle et peut n'être offerte à l'utilisateur que lorsqu'il sait comment découvrir le milieu d'un segment à l'aide d'un compas uniquement.

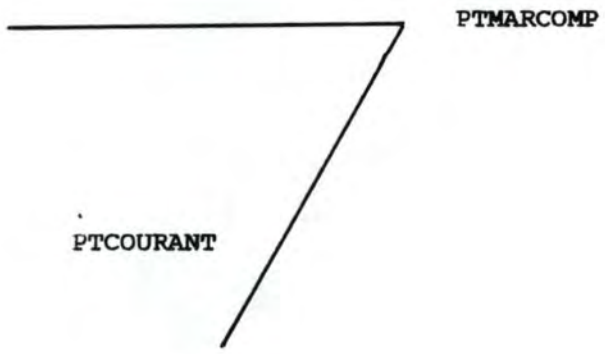
Dans le cadre de l'étude des transformations du plan, il est également nécessaire de pouvoir facilement reporter des angles. La mémoire du compas s'amplifie et devient capable de retenir un angle. Le compas se transforme en "COMPAS-RAPPORTEUR". Dans la version actuelle, cet angle est introduit numériquement au clavier.

Retenir une valeur numérique représentant un angle ne suffit pas pour nous faciliter la construction de l'image d'une figure par une rotation par exemple.

Encore faut-il pouvoir,

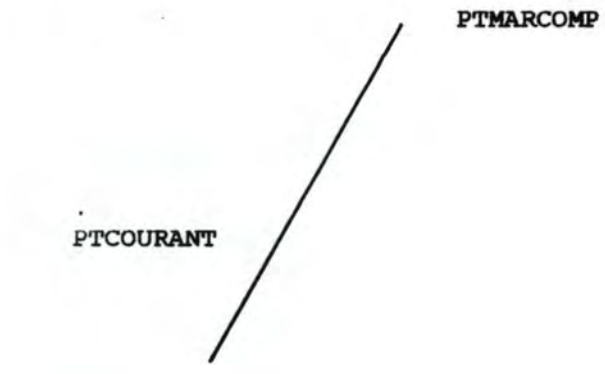
soit tracer les deux demi-droites formant en PTMARCOMP, un angle égal en valeur absolue à l'angle du compas, avec la demi-droite déterminée par PTMARCOMP et PTCOURANT,

soit ANGLE := 30



soit tracer la demi-droite formant en PTMARCOMP, un angle égal à l'angle du compas, avec la demi-droite déterminée par PTMARCOMP et PTCOURANT.

soit ANGLE := 30



et

soit ANGLE := -30

PTMARCOMP

PTCOURANT

Ces deux primitives sont elles aussi optionnelles. La première a été introduite en vue de ne pas résoudre le problème du signe de l'angle à la place de l'élève. Par contre lorsque cette notion est maîtrisée, la seconde permet un allègement du dessin.

En conclusion, l'outil COMPAS est caractérisé par un point, PTMARCOMP, un écartement, ECART, et un angle. Il permet de

1) gérer le curseur;
2) marquer un point;
3) donner numériquement une valeur à ECART;
4) donner la valeur d(PTCOURANT,PTMARCOMP) à ECART;
5) donner numériquement une valeur à ANGLE;
6) tracer le cercle de centre PTMARCOMP et de rayon égal à ECART;
7) renvoyer en inverse les deux points, appartenant à la droite passant par PTCOURANT et de direction égale à celle de la règle, situés à une distance de PTCOURANT égale à ECART;
8) tracer les deux demi-droites formant en PTMARCOMP, un angle égal en valeur absolue à l'angle du compas, avec la demi-droite déterminée par PTMARCOMP et PTCOURANT;
9) tracer la demi-droite formant en PTMARCOMP, un angle égal à l'angle du compas, avec la demi-droite déterminée par PTMARCOMP et PTCOURANT;
10) renvoyer en inverse le point milieu du segment [PTCOURANT,PTMARCOMP];

Les primitives 8,9 et 10 sont optionnelles. Rappelons également que lorsque cet outil est utilisé, le mot-clé COMPAS apparaît à l'écran.

Remarques :
Il serait bon d'envisager comme une extension la recherche d'une introduction de l'angle par une méthode qui ne serait pas numérique.

7. Configuration

Afin de permettre à l'enseignant de modifier l'outil en fonction de ses souhaits pédagogiques, de l'avancement de la matière ou de l'étudiant, certaines primitives sont optionnelles. La notion d'utilisateur privilégié apparaît immédiatement comme une nécessité. Sans cette notion de privilège, un élève un peu futé, pourrait s'octroyer l'usage de toutes les facilités.

Cet utilisateur privilégié a la possibilité de configurer le système, c'est-à-dire de permettre ou non l'usage de certaines primitives. Pour ne pas que chaque session doive être initialisée par le maître, il est nécessaire de mémoriser la dernière configuration introduite. A tout moment, même au milieu d'une session, cette configuration est modifiable.

Dans la version actuelle, six primitives sont optionnelles :

- tracer une droite parallèle à une droite donnée passant par un point donné;
- tracer une droite perpendiculaire à une droite donnée passant par un point donné;
- dessiner un point en mode règle;
- demander les deux demi-droites formant en PTMARCOMP, un angle donné en valeur absolue, avec la demi-droite déterminée par PTCOURANT et PTMARCOMP;
- demander la demi-droite formant en PTMARCOMP, un angle signé donné, avec la demi-droite déterminée par PTCOURANT et PTMARCOMP;
- demander le milieu d'un segment.

La notion d'utilisateur privilégié est réalisée par l'existence d'un mot de passe. Celui-ci est mémorisé et peut lui aussi être changé à tout moment.

CHAPITRE IV : IMPLEMENTATION

1. Généralités
   1.1 TURTLEGRAPHICS
   1.2 Découpe en modules
2. Gestion du curseur
3. Le module GOMME
4. Le module CONFIGURATION
5. Le module POINTMODE
6. Le module LATTE
7. Le module COMPAS

1. Généralités

L'école de Monsieur Poncelet est en possession d'APPLE II. J'ai donc travaillé moi aussi avec cette machine. J'ai programmé en Pascal (plutôt qu'en FORTH par exemple), afin d'élargir le cercle des personnes susceptibles de modifier le produit final.

Dans ce chapitre, je ne veux pas entrer dans tous les détails de l'implémentation. Je voudrais donner une vue d'ensemble de la découpe et mettre l'accent sur certains choix de réalisation.

Pour le lecteur désireux de connaître l'implémentation, j'ai inséré dans les annexes, une liste alphabétique des modules et leur spécification, le texte du programme.

1.1. TURTLEGRAPHICS

Le système Pascal sur APPLE II offre un outil graphique assez puissant : TURTLEGRAPHICS. Le mode graphique est essentiellement caractérisé par une "tortue" que l'on peut décrire comme suit :

- elle a une position donnée par les coordonnées cartésiennes du point où elle se situe;
- sa tête est dirigée dans une certaine direction;
- elle se déplace en laissant derrière elle une trace.

On peut demander à cette tortue

- de se rendre en tel point;
- d'avancer de telle distance dans la direction donnée par sa tête;
- de tourner la tête d'autant de degrés;
- de changer la couleur de sa trace;
- ...

Je ne peux pas énumérer ici toutes les possibilités offertes. Sachez seulement que j'utilise TURTLEGRAPHICS et que tout renseignement complémentaire se trouve dans les manuels du Pascal sur APPLE.

1.2. Découpe en modules

L'éditeur graphique est découpé en les modules suivant :

- INITIALISATION
- POINTMODE
- LATTE
- COMPAS
- GOMME
- CONFIGURATION

Il faut ajouter à ces six modules, le module "gestion du curseur" (déplacement par clavier - déplacement par JOYSTICK - mode d'écriture).

2. Gestion du curseur

Au-delà de la taille, de la forme du curseur, j'avais le choix suivant :

soit dessiner le curseur sur l'écran indépendamment de l'état de l'écran (la flèche apparaît alors en blanc quelque soit le dessin, avec le risque de ne pas la voir si l'écran est fort encombré);

soit dessiner le curseur grâce à un OU logique (la flèche apparaît noire là où l'écran était blanc et vice-versa). L'avantage de ce choix est qu'il suffit d'utiliser à nouveau un OU logique pour restaurer l'état initial de l'écran.

Pourtant, après avoir essayé l'un et l'autre, j'ai préféré dessiner la flèche en blanc sur noir. Ce choix m'impose de sauver l'état de l'écran là où je dessine le curseur (puisque celui-ci vient écraser le dessin) afin de pouvoir restaurer le dessin lorsque le curseur sera déplacé.

La gestion du curseur est assurée par les procédures :

- SAUVER
- DESSINCURSEUR
- MODULO (fonction)
- GAUCHE
- DROITE
- HAUT
- BAS
- CHANGEPAS
- ENCRECRAYON
- IMMOBJOY
- JOYCURS

3. Le module GOMME

Moyennant quelques manipulations (cfr annexes), deux pages graphiques sont à notre disposition avec possibilité d'écrire dans l'une ou l'autre, d'afficher la page 1 ou la page 2, copier la page 1 dans la page 2 et réciproquement. Ainsi, une réalisation possible est de dire que :

- à tout moment, la page 1 est affichée et représente le dessin tel que le veut l'utilisateur. Elle contient donc tous les traits au crayon et à l'encre.

- par contre, la page 2 ne contient que les traits à l'encre.

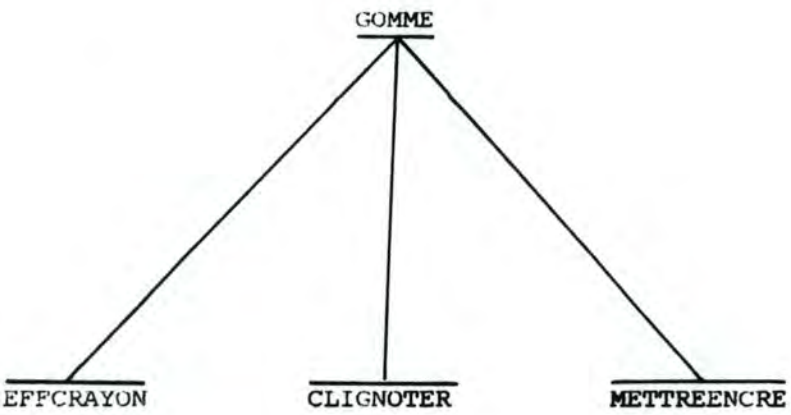
L'usage de ces deux pages implique notamment, lorsqu'il y a trace et que le mode d'écriture choisi par l'utilisateur est "encre", la nécessité d'écrire dans la page 1 ET dans la page 2

Ainsi, pour faire apparaître par clignotement ce qui est au crayon, il suffit d'afficher, par la procédure AFFPAGE, la page 1 en alternance avec la page 2. Ceci est réalisé par la procédure CLIGNOTER.

La procédure METTREENCRE en recopiant la page 1 dans la page 2 réalise la mise à l'encre du dessin, alors que c'est en copiant la page 2 dans la page 1 que la procédure EFFCRAYON efface tous les traits qui sont au crayon dans le dessin.

L'initialisation de l'outil gomme consiste à réserver la place mémoire pour la deuxième page graphique. Cette deuxième page est mise à blanc et ne contient que le cadre ainsi que le mot clé "GOMME".

Découpe de ce module



4. Le module CONFIGURATION

Comme nous l'avons vu, dans la version actuelle, six primitives sont optionnelles. J'ai choisi de configurer par des booléens.

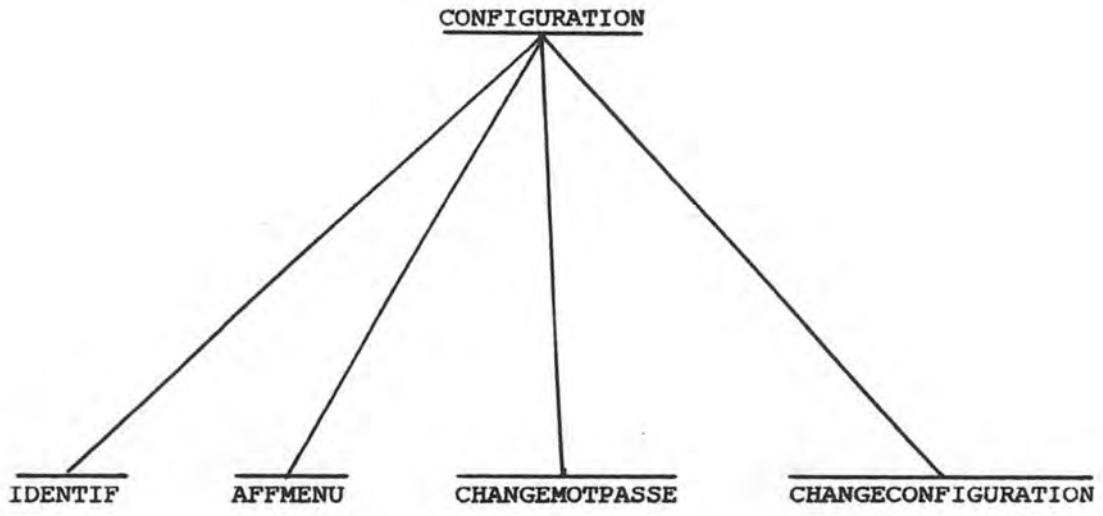
En début de programme, la procédure LIRECONF lit la configuration telle qu'elle a été mémorisée sur fichier et donne une valeur aux variables booléennes.

A tout moment de la session, cette configuration peut être modifiée. Pour cela, l'utilisateur doit être identifié comme privilégié. La fonction IDENTIF lui demande son mot de passe ! S'il ne correspond pas à celui qui est mémorisé, le message "Identification Incorrecte" est affiché, par contre s'ils sont identiques l'utilisateur a la possibilité de changer la configuration. Les variables booléennes sont mises à jour et la nouvelle configuration est mémorisée sur fichier.

Ce système est rudimentaire. En effet, le mot de passe étant conservé dans un fichier de la disquette, il n'est pas très compliqué d'aller le lire ou même de le modifier. Pour rendre cet accès un peu moins tentant, j'ai écrit une procédure de codage et de décodage du mot de passe. Il y a aussi possibilité de modifier le mot de passe.

Remarque :
.....
L'accès à la configuration mémorisée sur fichier est aussi aisé que celui du mot de passe. Il faudrait donc également envisager un codage de cet enregistrement.

Découpe de ce module
- - - - - - - - - - -



5. Le module POINTMODE

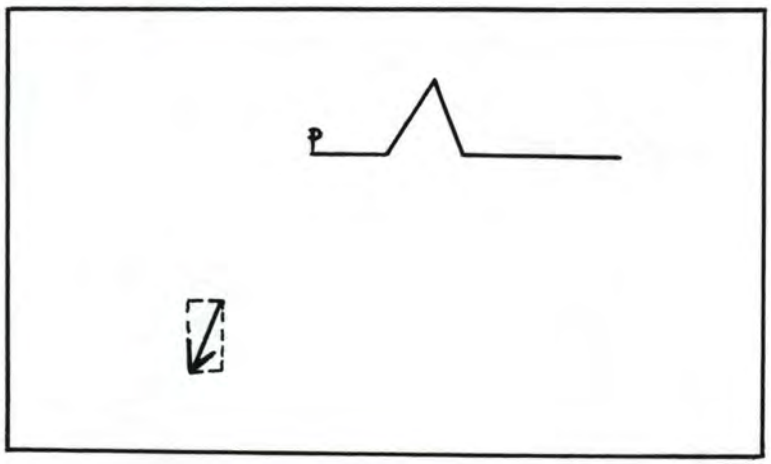
La particularité de ce module est de permettre le dessin de PTCOURANT. Plusieurs remarques sont à formuler quant à ce dessin :

- pour la première fois, le besoin d'écrire dans la seconde page graphique apparaît. En effet, si le mode d'écriture choisi est "encre", il faut non seulement dessiner ce point dans la page 1, mais aussi dans la page 2. La procédure **TRACEPAGE** modifie le paramètre utilisé par **TURTLEGRAPHICS** pour savoir dans quelle page la tortue doit écrire;

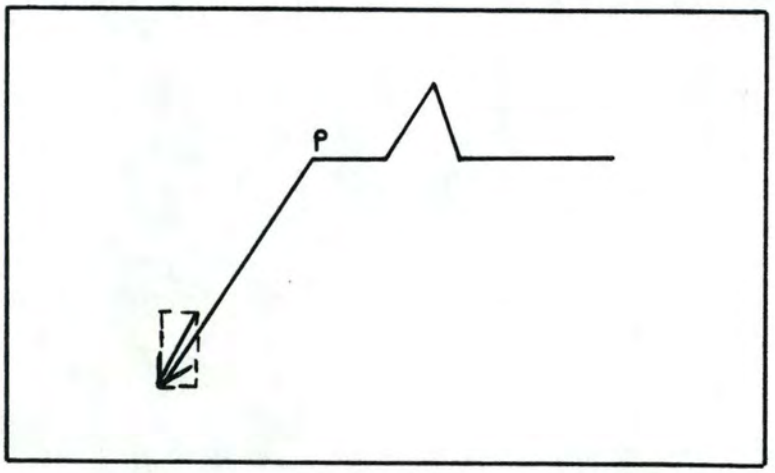
- la nécessité de la mise à jour de la copie de l'écran sous le curseur apparaît. Sans cette modification, lors du prochain déplacement du curseur et de la restauration de l'écran, le point dessiné serait perdu.

Cette dernière remarque est bien sûr également d'application pour n'importe quel tracé (droite - circonférence - point). Considérons en effet l'exemple suivant :

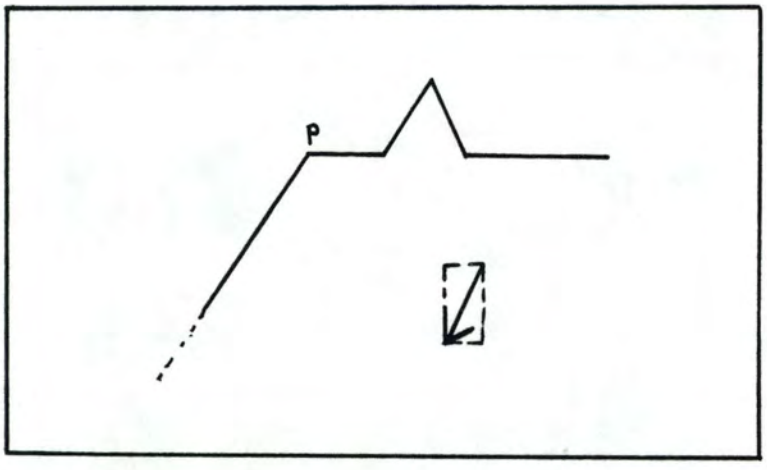
soit l'écran,



Ainsi, la copie de l'écran est un rectangle noir.
Traçons le segment joignant le point courant au point p.



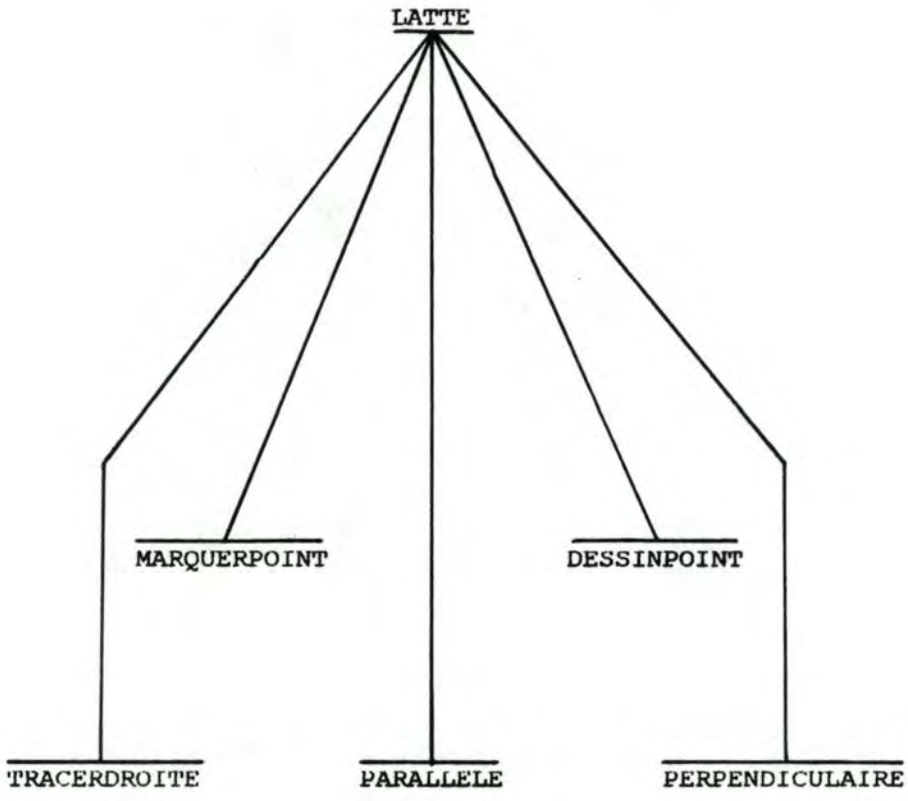
Si la copie n'est pas mise à jour, lors du prochain déplacement du curseur, la restauration de l'état initial de l'écran donnerait la figure suivante, où il y a perte de points.



6. Le module LATTE

Les remarques formulées dans le paragraphe précédent sont aussi valables pour cet outil. Il me reste juste à ajouter que si l'utilisateur fait appel à une primitive dont l'usage ne lui est pas autorisé, le message 'PRIMITIVE PAS PERMISE' est affiché à l'écran le temps nécessaire à la lecture.

Découpe de ce module

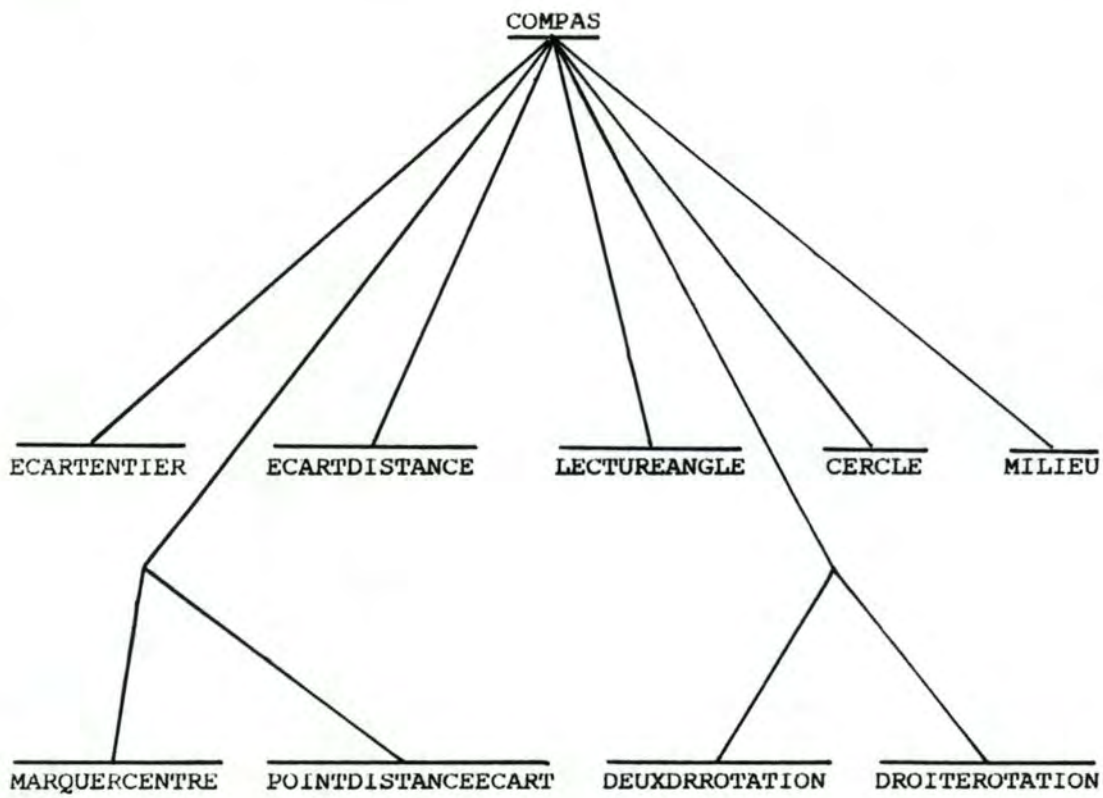


7. Le module COMPAS

Par deux fois dans les primitives proposées par cet outil, l'utilisateur introduit des valeurs numériques au clavier.

Pour gérer cette introduction en mode graphique, j'ai choisi d'utiliser la zone de l'écran réservée aux messages pour l'utilisateur. Sous le terme "gestion de l'introduction" se cache la prise en compte d'un éventuel signe et la possibilité d'effacement. Toute cette gestion est réalisée par la procédure LECTUREENTIER.

Découpe de ce module



CONCLUSIONS

C'est ici que je termine la rédaction de ce mémoire. Comme je l'ai dit au début du chapitre IV, le lecteur désireux de connaître l'implémentation, trouvera en annexe une liste des spécifications des modules et le texte du programme.

A propos de cette implémentation, quelques remarques s'imposent. TURTLEGRAPHICS est un outil puissant, mais il occupe beaucoup de place en mémoire centrale. De plus, je n'utilise pas toutes ses possibilités. Il serait bon d'écrire les routines utilisées en assembleur par exemple, et ce afin de récupérer une place, ô combien, nécessaire. En effet, lorsque j'ai rassemblé les différents outils en un seul programme, j'ai dû mener un rude combat contre les "STACK OVERFLOW".

Dans sa version actuelle, le programme tient tout juste en mémoire. Quelques améliorations sont à faire. J'en ai tentés quelques-unes, mais je me suis à nouveau trouvée face à des problèmes de place mémoire. J'ai renoncé à toute modification dans l'immédiat, faute de temps.

L'éditeur graphique que j'ai implémenté n'est qu'un îlot d'un projet beaucoup plus vaste. L'étude de ce projet, comme je l'ai mentionné tout au long des différents chapitres, doit être reprise, approfondie, ... et ce, en collaboration avec des enseignants et des pédagogues.

Quant à l'éditeur graphique lui-même, il serait bon de le mettre entre les mains d'élèves. Leurs réactions face au produit permettraient de mettre en évidence certaines difficultés de manipulations, d'éventuels manques, ...

Je puis enfin affirmer que mon intuition était correcte : l'informatique, les mathématiques et la pédagogie sont intervenues durant tout mon travail. La programmation sur micro était elle aussi intéressante, pour moi qui n'avait jamais travaillé sur ce type d'ordinateurs. En bref, le choix du début de l'année fut bon et le travail réalisé m'aidera beaucoup dans ma future vie professionnelle.

BIBLIOGRAPHIE

[1] ALBERTINI J-M. : Vers une informatique de formation , Education permanente, 70 - 71, (p 51 à 72), 1983.

[2] DUCHATEAU Ch. : L'informatique : technique nouvelle ou nouvel humanisme , FNDP Namur, 1983.

[3] GEM : L'Archipel des Isométries , GEM Louvain-la-Neuve, Dossier no 3, 1982.

[4] MEURRENS M. : Langages d'auteurs en EAO : Quels langages pour quels auteurs ? , Deuxième journée de réflexion sur l'informatique, JRI.2 (Contributions), Namur, 30 - 31 août, 1er septembre 1984.

[5] PASCALISSIME, Juillet 1983.

[6] PERRIAULT J. : Vingt ans d'EAO : usages, oublis, diversifications. , Education permanente, 70 - 71, (p 7 à 15), 1983.

[7] SCHWARTZ B. : L'informatique et l'éducation , La Documentation Française-Paris, 1981.

ANNEXES

1. Seconde page graphique
2. Manuel utilisateur
3. Spécification des modules
4. Programme

SECONDE PAGE GRAPHIQUE

LA SECONDE PAGE GRAPHIQUE

1. Affichage de la seconde page graphique

L'apple II peut utiliser quatre pages: 2 pages textes et deux pages graphiques. L'ordinateur n'a qu'un seul écran, il faut donc lui indiquer quelle page il doit afficher.

Pour afficher quelque chose à l'écran, il suffit

- de placer ce quelque chose dans la zone mémoire correspondant à l'une des quatres pages.
- d'indiquer à la machine qu'elle doit afficher cette zone.

Par défaut, en mode texte, la page texte numéro 1 est affichée. De même en mode graphique, la première page graphique est à l'écran.

La possibilité existe cependant de modifier ces affichages:

> Pour afficher les secondes pages (texte ou graphique)
> il faut forcer la valeur 0 à l'adresse -16299
> (POKE -16299,0)

L'instruction POKE -16300,0 permet le retour à l'affichage des premières pages.

La procédure suivante permet ce passage de l'une à l'autre :

```
procedure  AFFPAGE (N: integer);

begin
if N=1  then POKE (-16300,0)
        else POKE (-16299,0)
end;
```

avec

```
procedure  POKE (LOC,CONT:integer);

var X: LOCBYTE;

begin
X.INT:=LOC;
X.PTR ^[0]:=CONT
end;
```

où

```
type    BYTE    = packed array [0..1] of 0..255;
        LOCBYTE = record
                      case boolean of
                           true: (INT:integer);
                           false: (PTR:^BYTE)
                           end;
```

2. Le tracé sur la page graphique 2

Les procédures de TURTTLEGRAPHICS manipulent normalement la page graphique 1 et il n'existe pas dans TURTTLEGRAPHICS de procédure permettant de changer de page
Toutefois ces primitives utilisent un paramètre qui indique quelle est la page utilisée. Par défaut, la valeur du paramètre est 1. Il suffit de lui donner la valeur 2 pour pouvoir écrire dans la seconde page graphique

CE PARAMETRE EST LE QUATORZIEME OCTET DE L'ENREGISTREMENT DONT L'ADRESSE EST MAINTENUE A L'ADRESSE 254 ($FE) DE LA PAGE 0

Pour demander à la tortue d'écrire sur l'une ou l'autre des pages, il suffit d'utiliser la procédure suivante :

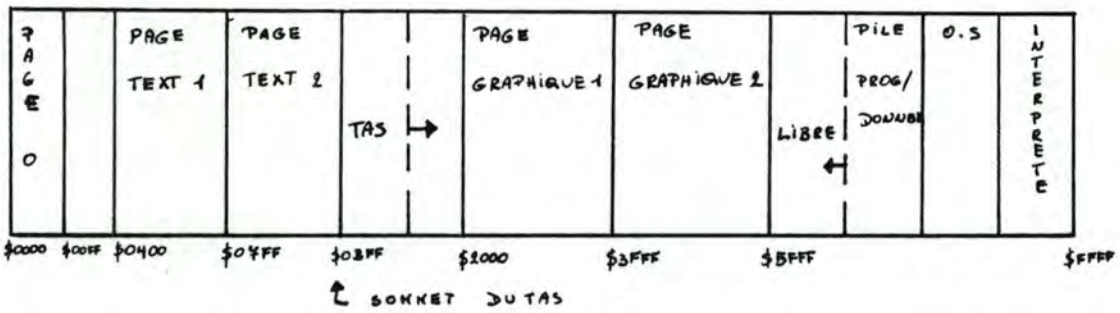
```
procedure TRACEPAGE (N:integer);

type  PAGE = record
             case integer of
                  1: (ADRPAGE : integer);
                  2: (PTRPAGE : ^integer)
                  end;

var TRPAGE : PAGE;

begin
TRPAGE . ADRPAGE := 254 ;
TRPAGE . ADRPAGE := TRPAGE . ADRPAGE^ + 14;
TRPAGE . PTRPAGE^:= N
end;
```

Remarque :
Il est possible d'écrire dans une page sans que celle-ci soit affichée.

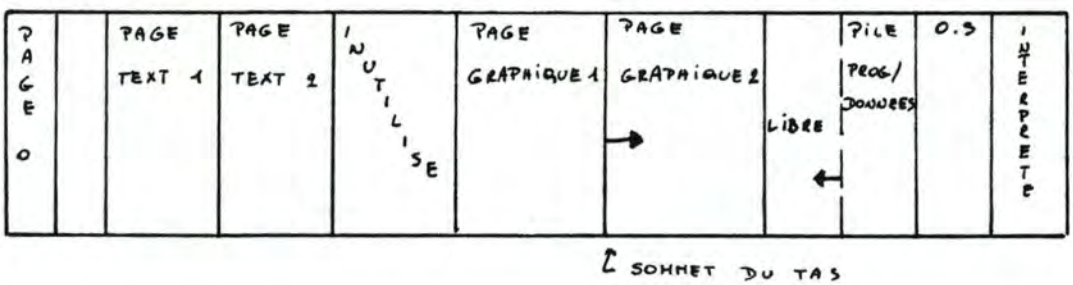
3. Modification du sommet du tas.

Si nous voulons utiliser la seconde page graphique, nous devons la protèger. En effet, le système maintient en permanence l'adresse du sommet du tas. Chaque fois qu'une nouvelle structure est crèée (NEW), il ajuste cette valeur. De plus le système place sur le tas la table des matières chaque fois qu'il a besoin d'accéder à une disquette.

Initialement, le sommet a pour valeur $OCF9 environ. Lorsque nous utilisons une page graphique, il y a risque de collision entre le tas et cette page.



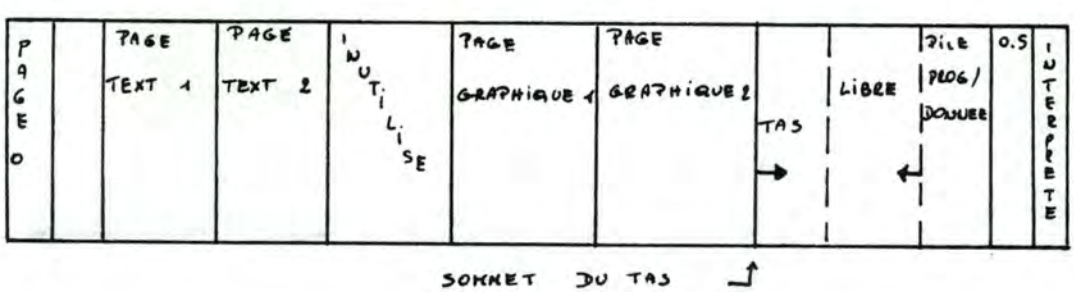
Pour èviter une collision, lorsque INITTURTLE est appellè le sommet du tas prend pour valeur le sommet de la page graphique 1.



Ainsi, si nous choisissons d'utiliser la page graphique 2 , il faut modifier l'adresse du tas. Pour cela, il nous suffit d'utiliser la fonction de gestion du tas :

RELEASE (SOMMET) où SOMMET est du type pointeur vers un entier

dont l'effet est de donner la valeur de SOMMET au sommet du tas (pointeur vers une adresse).



Par conséquent, si nous souhaitons modifier l'adresse du sommet du tas, nous pouvons utiliser la procédure suivante :

```
type PAGEGRAPH = record
                   case boolean of
                        false : (ADR : integer) ;
                        true  : (PADR: ^integer)
                        end ;
```

et

```
procédure RESERVE (MINIMUM : integer) ;

var TASRES : PAGEGRAPH ;

begin
TASRES.ADR := MINIMUM ;
RELEASE (TASRES . PADR)
end;
```

MANUEL UTILISATEUR

MANUEL UTILISATEUR

1. Comment démarrer ?

Après avoir chargé le système Pascal, vous introduisez la disquette dans le second drive. Le programme se trouve dans le fichier GREDIT.CODE. Pour le faire exécuter vous devez introduire la commande

eX)ecute #5:GREDIT.CODE

Il vous est alors demandé si vous utilisez un PADDLE. Si oui, vous devez le brancher.

L'écran graphique apparaît. Vous y voyez un cadre et une flèche. La flèche représente le curseur, le cadre en délimite les mouvements.

Voyons maintenant comment déplacer ce curseur et quels sont les outils offerts.

2. Les outils

Quatre outils sont offerts :

. : l'outil POINT

R : l'outil REGLE

K : l'outil COMPAS

G : l'outil GOMME

Si vous enfoncez l'une des quatres touches, vous verrez apparaître dans le coin supérieur droit, respectivement le mot-clé POINT, REGLE, COMPAS ou GOMME, ceci afin de vous rappeler à tout instant quel est l'outil utilisé !

2.1. Gestion du curseur

Si vous utilisez un PADDLE, celui-ci vous permet de déplacer la flèche sur l'écran, sinon les quatres touches

-> : à droite;

<- : à gauche;

A : en haut;

Z : en bas.

réalisent le mouvement.

La lettre C (changerpas) permet de modifier la longueur du déplacement du curseur.

Le curseur a de plus une autre propriété : il écrit au crayon ou à l'encre (nous verrons pourquoi dans l'outil GOMME). La touche '/' permet de passer d'un mode d'écriture a l'autre. Regardez dans le coin supérieur droit :

(C) indique que le mode d'écriture est 'crayon';

(E) qu'il est 'encre'.

2.2. L'outil GOMME

Les traits du dessin sont soit au crayon, soit à l'encre (selon le mode d'écriture choisi).

Les différentes possibilités sont :

C)lignoter : clignotement de ce qui est au crayon;

E)ffacer : efface ce qui est au crayon;

M)ise à l'encre :
tous les traits sont mis à l'encre.

Outils accessibles
.....

. : POINT

R : REGLE

K : COMPAS

<RET> : FIN DU PROGRAMME

2.3. L'outil POINT

Outre la gestion du curseur, l'outil POINT offre la particularité de dessiner le point identifié par le curseur : touche D)essinpoint.

En résumé, vous avez les possibilités suivantes :

Gestion du curseur
.....

<- : GAUCHE

-> : DROITE

A : HAUT

Z : BAS

C : change la valeur du déplacement du curseur (grand <-> petit)

/ : change le mode d'écriture (encre <-> crayon)

D : DESSINE le point identifié par le curseur

Outils accessibles
.....

R : REGLE

K : COMPAS

G : GOMME

<RET> : FIN DU PROGRAMME

2.4. L'outil REGLE

Quelques définitions

Appelons PTCOURANT le point déterminé par le curseur et PTCOUREG le dernier point soit qui a été dessiné, soit à partir duquel on a tracé une droite en mode règle.
PTMAREG est le dernier point marqué en mode REGLE.

Gestion du curseur

<- : GAUCHE

-> : DROITE

A : HAUT

Z : BAS

C : change la valeur du déplacement du curseur (grand <-> petit)

/ : change le mode d'écriture (encre <-> crayon)

Autres possibilités

M : marque PTCOURANT

* : trace le segment de droite joignant PTCOURANT à PTMAREG

S : trace le segment de droite joignant PTCOURANT à PTCOUREG

Et si la configuration le permet
.....

D : dessine PTCOURANT

P : trace à l'aide de <- et ->, la demi-droite gauche, la demi-droite droite, ou la droite passant par PTCOURANT, parallèle à la dernière droite tracée

O : trace à l'aide de <- et ->, la demi-droite gauche, la demi-droite droite, ou la droite passant par PTCOURANT, perpendiculaire à la dernière droite tracée

Outils accessibles
.....

. : POINT

K : COMPAS

G : GOMME

<RET> : FIN DU PROGRAMME

2.5. L'outil COMPAS

Quelques définitions

Appelons PTCOURANT le point déterminé par le curseur.
ECART et ANGLE représentent respectivement l'écartement et l'angle mémorisés par le compas. PTMARCOMP est le dernier point marqué en mode COMPAS.

Gestion du curseur

<- : GAUCHE

-> : DROITE

A : HAUT

Z : BAS

C : change la valeur du déplacement du curseur (grand <-> petit)

/ : change le mode d'écriture (encre <-> crayon)

Autres possibilités

M : marque PTCOURANT

E : permet d'introduire, au terminal, une valeur numérique positive pour ECART

L : permet d'introduire, au terminal, une valeur numérique signée pour ANGLE

D : donne comme valeur à ECART la distance entre PTCOURANT et PTMARCOMP

O : trace le cercle de centre PTMARCOMP et de rayon ECART

Et si la configuration le permet
.....

P : renvoit, en inverse, les points de la droite passant par PTCOURANT de direction égale à celle de la règle, situés à une distance de PTCOURANT égale à ECART

2 : trace les deux demi-droites formant, en PTMARCOMP, un angle égal en valeur absolue à ANGLE, avec la droite joignant PTMARCOMP à PTCOURANT

1 : trace la demi-droite formant, en PTMARCOMP, un angle égal à ANGLE, avec la droite joignant PTMARCOMP à PTCOURANT

B : renvoit, en inverse, le point milieu du segment [PTCOURANT,PTMARCOMP]

Outils accessibles
.....

. : POINT

R : REGLE

G : GOMME

<RET> : FIN DU PROGRAMME

3. Configuration

L'utilisateur privilègiè peut autoriser ou non l'usage des primitives suivantes :

en mode REGLE

- tracer la droite passant par PTCOURANT de direction égale à celle de la règle
- tracer la droite passant par PTCOURANT de direction orthogonale à celle de la règle
- dessiner PTCOURANT

en mode COMPAS

- tracer les deux demi-droites formant, en PTMARCOMP, un angle égal en valeur absolue à ANGLE, avec la droite joignant PTMARCOMP à PTCOURANT
- tracer la demi-droite formant, en PTMARCOMP, un angle égal à ANGLE, avec la droite joignant PTMARCOMP à PTCOURANT
- dèterminer le point milieu du segment [PTCOURANT,PTMARCOMP].

Le module CONFIGURATION est accessible à partir de chaque outil. Il suffit d'enfoncer la touche '$'.

Dans un premier temps, le mot de passe est demandè à l'utilisateur. Si celui-ci n'introduit pas le mot de passe correctement, il ne peut continuer. Il doit alors, soit sèlectionner un des outils disponibles, soit terminer le programme.

Par contre, si l'identification est correcte, le menu suivant est affichè :

M) changer le mot de passe

C) changer la configuration

.) POINT R) REGLE

K) COMPAS G) GOMME

<RET> fin du programme.

Les possibilités offertes sont le changement du mot de passe, la modification de la configuration, la sélection d'un des outils ou la fin du programme.

Une seule remarque est à faire. Elle concerne le changement de configuration . Vous voyez apparaître, en lieu et place du libellé complet de la primitive, un mot-clé. L'ordre d'apparition de ceux-ci est celui du rappel ci-dessus. Les mots-clé associés sont :

PARALPERMIS

PERPENDPERMIS

OKPOINTREGLE

DRDEUXROT

DRROTPERMIS

MILIEUPERMIS

Pour chaque primitive, il suffit de répondre O (oui) ou N (non) selon que vous autorisez ou non l'usage de la primitive concernée. L'écran reste affiché jusqu'à ce que vous enfonciez une touche et ce afin de vous permettre d'examiner vos réponses.

Cette nouvelle configuration est enregistrée sur fichier en fin de programme et initialisera la prochaine session.

Initialement, toutes les primitives sont permises et le mot de passe est SIMON.

SPECIFICATION DES MODULES

procedure ADDANGLE

arguments : - COSA, SINA : réel
- B : entier

préconditions : -1 <= COSA <= 1
-1 <= SINA <= 1

résultats : COAB, SIAB : réels

postconditions : Soit A l'angle dont le cosinus est COSA et le sinus SINA. Alors COAB = cos(A+B) et SIAB = sin(A+B)

appels : /

procedure AFFMENU

arguments : /

préconditions : /

résultats : écran

postconditions : affiche à l'écran en mode texte les diverses possibilités offertes par l'outil CONFIGURATION

appels : /

procedure AFFPAGE

arguments : N : entier

préconditions : N appartient à {1,2}

résultats : écran

postconditions : affiche à l'écran la Nème page graphique

appels : /

Rem : cfr. annexes pour plus de renseignements
===

procedure BAS

arguments : I : entier

préconditions : I appartient à [0 , min (LARGFEN,HAUTFEN)]

résultats : - XCOURANT,YCOURANT : entier
COPIE : curseur

postconditions : - déplace le curseur et la tortue de I points vers le bas, en restaurant l'écran sous le curseur et en sauvant la portion de l'écran de la nouvelle position du curseur
- mise à jour de (XCOURANT, YCOURANT)
- si la nouvelle position déborde la fenêtre en bas, le curseur réapparaît en haut

appels : DESSINCURSEUR
MODULO

procedure CALCULPOINT

arguments : A,B,C : entier

préconditions : A <> 0
0 <= C <= LARGFEN

résultats : CALCULPOINT : entier

postconditions : CALCULPOINT est l'ordonnée du point d'abscisse C, appartenant à la droite passant par le point courant et de paramètres directeurs (A,B)

appels : /

fonction CARON

arguments : C : caractère

préconditions : /

résultats : CARON : booléen

postconditions : CARON SSI (C=o ou C=O ou C=n ou C=N)

appels : /

procedure CERCLE

arguments : /

prêconditions : /

résultats : COPIE : curseur
pages graphiques 1 et 2

postconditions : - si ECART<>0, trace, dans la page graphique 1
si PLUME=1, dans les deux si PLUME=2,
le cercle de centre
(XMARCOMP,YMARCOMP) et de rayon égal a ECART
- dessine le centre
- la tortue est au point courant
- mise à jour de COPIE

appels : TRACERCERCLE
SAUVER
TRACEPAGE

procedure CHANGECONFIGURATION

arguments : /

prêconditions : /

résultats : - PARALPERMIS, PERPENDPERMIS, OKPOINTTREGLE, DRDEUXROT,
DRROTPERMIS et MILIEUPERMIS : booléens

postconditions : - demande à l'utilisateur, pour chaque primitive
optionnelle, s'il autorise ou non son usage
- mise à jour des booléens cités en fonction
de la nouvelle configuration

appels : CARON

procedure CHANGEMOTPASSE

arguments : /

préconditions : /

résultats : MOTCOMP : PASSWD

postconditions : - Si on dispose de deux introductions identiques, MOTCOMP prend la valeur introduite. Sinon, le message 'ERREUR DANS L'INTRODUCTION DU MOT DE PASSE' est affiché le temps nécessaire à la lecture
- mise à blanc de l'écran

appels : ECRIREMOTPASSE

procedure CHANGEPAS

arguments : /

préconditions : /

résultats : PAS : entier

postconditions : change la grandeur du déplacement PAS (GRANDPAS <-> PETITPAS)

appels : /

procedure CLIGNOTER

arguments : /

préconditions : /

résultats : écran

postconditions : affichage en alternance de la page graphique 1 et de la page graphique 2 jusqu'à ce que l'utilisateur enfonce une touche.

appels : AFFPAGE

procedure COMPAS

arguments : /

préconditions : /

résultats : XCOURANT, YCOURANT, XMAREG, YMAREG : entier
ECART, ANGLE : entier
COPIE : curseur
ACTION : caractère
pages graphiques 1 et 2

postconditions : - écriture dans le coin supérieur droit, à partir du point de coordonnées (200,HAUTFEN+2), du mot-clé COMPAS
- offre les possibilités suivantes :
gestion du curseur
marquer PTCOURANT
introduire au terminal une valeur pour ECART
introduire au terminal une valeur pour ANGLE
donner comme valeur à ECART la distance entre PTCOURANT et PTMARCOMP
tracer le cercle de centre PTMARCOMP et de rayon ECART
renvoyer en inverse, les points de la droite passant par PTCOURANT de direction égale à celle de la règle, situés à une distance de PTCOURANT égale à ECART
tracer les deux demi-droites formant en PTMARCOMP un angle égal en valeur absolue à ANGLE, avec la droite joignant PTMARCOMP à PTCOURANT
tracer la demi-droite formant en PTMARCOMP, un angle égal à ANGLE, avec la droite joignant PTMARCOMP à PTCOURANT
renvoyer en inverse, le point milieu du segment [PTMARCOMP,PTCOURANT]
- ACTION appartient à {<RET>,.,R,G,$}

appels :

ECRITMODE	ENCRECRAYON	CERCLE
GAUCHE	MARQUERCENTRE	POINTDISTANCEECART
DROITE	ECARTENTIER	DEUXDRROTATION
HAUT	LECTUREANGLE	DROITEROTATION
BAS	ECARTDISTANCE	MILIEU
CHANGEPAS		

procedure CONFIGURATION

arguments : /

préconditions : /

résultats : ACTION : caractère

postconditions : - si l'identification est correcte,
affichage du menu
et/ou changement du mot de passe
et/ou changement de la configuration

jusqu'à ce que l'utilisateur appuie sur l'une des touches suivantes : {R,K,.,G,<RET>} = A.
ACTION prend cette valeur.

sinon, le message 'Quel mode voulez vous ?' est affiché en mode texte et ACTION prend la valeur entrée par l'utilisateur, pour autant que cette lettre appartienne à A
- mise à blanc de l'écran texte
- retour au mode graphique

appels : IDENTIF
AFFMENU
CHANGEMOTPASSE
CHANGECONFIGURATION

procedure COORDPOLAIRE

arguments : /

préconditions : /

résultats : DIST, COANGLE, SIANGLE : réels

postconditions : soit (DIST,O) les coordonnées polaires de PTCOURANT par rapport à PRMARCOMP, alors
COANGLE = cos(O)
SIANGLE = sin(O)

appels : /

procedure DESSINCURSEUR

arguments : X,Y : entier

préconditions : 0 <= X <= LARGFEN
0 <= Y <= HAUTFEN

résultats : XCOURANT, YCOURANT : entier
COPIE : curseur

postconditions : - restaure l'écran sous le curseur
- sauve dans COPIE l'état de l'écran à la nouvelle position du curseur
- déplace le curseur et la tortue jusqu'au point de coordonnées (X,Y)
- mise à jour de XCOURANT, YCOURANT

appels : SAUVER

procedure DESSINPOINT

arguments : /

préconditions : /

résultats : XCOUREG, YCOUREG : entier
COPIE : curseur
pages graphiques 1 et 2

postconditions : - si la configuration le permet dessine le point courant
dans la page graphique 1 si PLUME=1,
dans les deux si PLUME=2
et met à jour XCOUREG,YCOUREG et COPIE
sinon affichage du message 'PRIMITIVE NON PERMISE'

appels : POINTDES
ERREUR

procedure DEUXDRROTATION

arguments : ANG : entier

préconditions : /

résultats : COPIE : curseur
pages graphiques 1 et 2

postconditions : si la configuration le permet
trace, si PTMARCOMP <> PTCOURANT,
dans la page graphique 1 si PLUME=1,
dans les deux si PLUME=2
les deux demi-droites formant en PTMARCOMP, un angle
égal en valeur absolue à ANG, avec la demi-droite
déterminée par PTMARCOMP et PTCOURANT

mise à jour de PTCOUREG
sinon le message 'PRIMITIVE PAS PERMISE' apparaît
à l'écran le temps nécessaire à la lecture

appels : ERREUR
COORDPOLAIRE
ADDANGLE
SAUVER
TRACEPAGE

procedure DROITE

arguments : I : entier

préconditions : I appartient à [0 , min (LARGFEN,HAUTFEN)]

résultats : - XCOURANT,YCOURANT : entier
COPIE : curseur

postconditions : - déplace le curseur et la tortue de I points
vers la droite, en restaurant l'écran sous
le curseur et en sauvant la portion de l'écran
de la nouvelle position du curseur
- mise à jour de (XCOURANT, YCOURANT)
- si la nouvelle position déborde la fenêtre à
droite le curseur réapparaît à gauche

appels : DESSINCURSEUR
MODULO

procedure DROITEROTATION

arguments : ANG : entier

préconditions : /

résultats : COPIE : curseur
pages graphiques 1 et 2

postconditions : - si la configuration le permet
trace, si PTMARCOMP <> PTCOURANT,
dans la page graphique 1 si PLUME=1,
dans les deux si PLUME=2
la demi-droite formant en PTMARCOMP, un angle
égal à ANG, avec la demi-droite
déterminée par PTMARCOMP et PTCOURANT
mise à jour de PTCOUREG
- sinon le message 'PRIMITIVE PAS PERMISE' apparaît
à l'écran le temps nécessaire à la lecture

appels : ERREUR
COORDPOLAIRE
ADDANGLE
SAUVER
TRACEPAGE

procedure ECARTDISTANCE

arguments : /

préconditions : /

résultats : ECART : entier

postconditions : ECART est égal à la distance entre PTCOURANT
et PTMARCOMP

appels : /

procedure ECARTENTIER

arguments : /

préconditions : /

résultats : ECART : entier

postconditions : - écriture du message 'ENTREZ UNE DISTANCE (>=0)'
- ECART est un entier positif, introduit au clavier en mode graphique, appartenant à [0,LARGFEN]

appels : ECRITMODE
LECTUREENTIER
MODULO

procedure ECRIREMOTPASSE

arguments : - NOMFICH : string
- MOT : PASSWD

préconditions : /

résultats : NOMFICH : string

postconditions : le fichier NOMFICH est crée et contient le mot de passe MOT codé

appels : /

procedure ECRITMODE

arguments : - X : entier
- S : string

préconditions : 0 <= X <= LARGFEN - 7*LENGHT(S)

résultats : écran

postconditions : écriture à l'écran, en mode graphique de la valeur de S à partir du point de coordonnées (X,HAUTFEN+2)

appels : /

procedure ECRSAUVER

arguments : A,B : entiers

préconditions : 0 <= A <= LARGFEN
0 <= B <= HAUTFEN

résultats : COPIE : curseur
pages graphiques 1 et 2

postconditions : - tracé du segment joignant le point courant au point de coordonnées (A,B), dans la page graphique 1, si PLUME=1
(i.e. le mode d'écriture choisi est "crayon")
dans les deux pages graphiques si PLUME=2
(i.e le mode d'écriture choisi est "encre"),
avec modification de COPIE
- la tortue est au point courant

appels : TRACEPAGE
SAUVER

procedure EFFCRAYON

arguments : /

préconditions : /

résultats : page graphique 1

postconditions : la page graphique 2 est copiée dans la page graphique 1

appels : ECRITMODE

procedure ENCRECRAYON

arguments : /

préconditions : /

résultats : PLUME : entier

postconditions : - si PLUME = 1 alors
PLUME = 2
écriture de ' (E)' dans le coin
supérieur droit à partir du point
de coordonnées (242,HAUTFEN+2)
sinon
PLUME = 1
écriture de ' (C)' dans le coin
supérieur droit à partir du point
de coordonnées (242,HAUTFEN+2)

appels : ECRITMODE

procedure ERREUR

arguments : /

préconditions : /

résultats : écran

postconditions : - écrit dans le coin supérieur gauche de la page
graphique 1 le message 'PRIMITIVE PAS PERMISE' à
partir du point de coordonnées (0,HAUTFEN+2),
et ce durant le temps nécessaire à la lecture
- efface le message

appels : ECRITMODE

procedure FLECHEDROITE
---

arguments : A,B : entier

préconditions : /

résultats : COPIE : curseur
pages graphiques 1 et 2

postconditions : - tracé dans la page graphique 1 si PLUME=1, et dans les deux pages si PLUME=2, de la partie droite de la droite passant par le point courant et de paramètres directeurs (A,B).
convention : si la droite est verticale, on trace vers le bas
- mise à jour de COPIE
- la tortue est au point courant

appels : ECRSAUVER
CALCULPOINT

procedure FLECHEGAUCHE
---

arguments : A,B : entier

préconditions : /

résultats : COPIE : curseur
pages graphiques 1 et 2

postconditions : - tracé dans la page graphique 1 si PLUME=1, et dans les deux pages si PLUME=2 de la partie gauche de la droite passant par le point courant et de paramètres directeurs (A,B).
convention : si la droite est verticale, on trace vers le haut
- mise à jour de COPIE
- la tortue est au point courant

appels : ECRSAUVER
CALCULPOINT

procedure GAUCHE

arguments : I : entier

préconditions : I appartient à [0 , min (LARGFEN,HAUTFEN)]

résultats : - XCOURANT,YCOURANT : entier
COPIE : curseur

postconditions : - déplace le curseur et la tortue de I points vers la gauche, en restaurant l'écran sous le curseur et en sauvant la portion de l'écran de la nouvelle position du curseur
- mise à jour de (XCOURANT, YCOURANT)
- si la nouvelle position déborde la fenêtre à gauche le curseur réapparaît à droite

appels : DESSINCURSEUR
MODULO

procedure GOMME

arguments : /

préconditions : /

résultats : ACTION : caractère

postconditions : - écriture dans le coin supérieur droit, à partir du point de coordonnées (200,HAUTFEN+2), du mot-clé GOMME
- permet d'effacer les traits au crayon et/ou le clignotement des traits au crayon et/ou de mettre tous les traits à l'encre
- ACTION appartient à {<RET>,R,.,K,$}

appels : ECRITMODE
EFFCRAYON
CLIGNOTER
METTREENCRE

procedure HAUT

arguments : I : entier

préconditions : I appartient à [0 , min (LARGPEN,HAUTPEN)]

résultats : - XCOURANT,YCOURANT : entier
COPIE : curseur

postconditions : - déplace le curseur et la tortue de I points vers le haut, en restaurant l'écran sous le curseur et en sauvant la portion de l'écran de la nouvelle position du curseur
- mise à jour de (XCOURANT, YCOURANT)
- si la nouvelle position déborde la fenêtre en haut le curseur réapparaît en bas

appels : DESSINCURSEUR
MODULO

fonction IDENTIF

arguments : /

préconditions : /

résultats : IDENTIF : booléen

postconditions : - si le mot de passe introduit par l'utilisateur est identique à MOTCOMP, IDENTIF = TRUE sinon IDENTIF = FALSE et le message 'IDENTIFICATION INCORRECTE' est affiché à l'écran le temps nécessaire à la lecture
- mise à blanc de l'écran

appels : LECTMOTPASSE

fonction IMMOBJOY
---

arguments : /

préconditions : /

résultats : XPAD, YPAD : entier

postconditions : - XPAD et YPAD sont les valeurs données
par le PADDLE
- IMMOBJOY SSI la manette est déplacée
de sa position centrale
d'une valeur supérieure à SAUT

appels : /

procedure INIT
---

arguments : /

préconditions : /

résultats :

postconditions : - lecture du mot de passe et de la configuration
stockés sur fichiers
- initialise l'écran, le curseur, le PADDLE et
les outils POINT, REGLE, COMPAS et GOMME
- dessine le curseur au point (90,140)

appels : INITPADDLE
LIRECONF
INITPOINTECRAN
INITGOMME
INITCOMPAS
DESSINCURSEUR

procedure INITAUX

arguments : /

préconditions : /

résultats : AUX : curseur

postconditions : tous les éléments de AUX sont à zéro

appels : /

procedure INITCOMPAS

arguments : /

préconditions : /

résultats : XMAREG, YMAREG, XCOUREG, YCOUREG, XDIR, YDIR : entier
XMARCOMP, YMARCOMP, ANGLE, ECART : entier
TAB : array [1..31] of real

postconditions : - initialise PTMAREG, PTCOUREG et PTMARCOMP
au point courant initial (140,90)
- initialise (XDIR,YDIR) à (0,1)
- initialise ANGLE et ECART à 0
- initialisation de TAB aux valeurs du sinus

appels : /

procedure INITCURSEUR

arguments : /

préconditions : /

résultats : AUX : curseur

postconditions : AUX est initialisé au dessin voulu pour le curseur.
Dans la version actuelle, le curseur est une flèche

appels : /

procedure INITGOMME

arguments : /

préconditions : /

résultats : page graphique 2

postconditions : - réserve la place nécessaire pour le seconde page graphique
- la seconde page contient le cadre déterminé par les constantes LARGFEN et HAUTFEN, le mot-clé GOMME (C) dans le coin supérieur droit, à partir du point de coordonnées (200,HAUTFEN+2)

appels : TRACEPAGE
ECRITMODE
RESERVE

procedure INITPADDLE

arguments : /

préconditions : /

résultats : XCENJOY, YCENJOY : entier

postconditions : - demande à l'utilisateur s'il utilise un PADDLE
- si oui, demande de le connecter
- XCENJOY, YCENJOY donnent la position centrale de la manette

appels : /

procedure INITPOINTECRAN

arguments : /

préconditions : /

résultats : FLECHE, COPIE : curseur
PAS, XCOURANT, YCOURANT, PLUME : entier
DESPOINT : booléen
page graphique 1

postconditions : - FLECHE contient le dessin choisi pour le curseur
- mise à zéro des éléments de COPIE
- PAS vaut GRANDPAS
- (XCOURANT,YCOURANT) = (140,90) et la tortue est à cette position
- dessin du cadre déterminé par LARGFEN et HAUTFEN

appels : INITCURSEUR
INITAUX
ECRITMODE

procedure JOYCURS

arguments : /

préconditions : /

résultats : XCOURANT, YCOURANT : entier

postconditions : - déplacement du curseur et de la tortue en fonction du déplacement du PADDLE, avec restauration de l'écran sous le curseur et sauvetage de l'écran sous la nouvelle position
- mise à jour de XCOURANT, YCOURANT

appels : IMMOBJOY

procedure LATTE

arguments : /

preconditions : /

résultats : XCOURANT, YCOURANT, XCOUREG, YCOUREG : entier
XMAREG, YMAREG, XDIR, YDIR, : entier
COPIE : curseur
ACTION : caractère
pages graphiques 1 et 2

postconditions : - écriture dans le coin supérieur droit, à partir du point de coordonnées (200,HAUTFEN+2), du mot-clé REGLE
- offre les possibilités suivantes
gestion du curseur
marquer PTCOURANT
tracé du segment joignant PTCOURANT à PTCOUREG
tracé du segment joignant PTCOURANT à PTMAREG
tracé d'une droite passant par PTCOURANT et de direction égale à celle de la règle
tracé d'une droite passant par PTCOURANT et de direction orthogonale à celle de la règle
dessin du point courant
- ACTION appartient à {<RET>,.,K,G,$}

appels : ECRITMODE, GAUCHE, DROITE, HAUT, BAS, CHANGEPAS, ENCRECRAYON, MARQUERPOINT, DESSINPOINT, TRACERDROITE, PARALLELE, PERPENDICULAIRE, IMMOBJOY, JOYCURS

procedure LECTMOTPASSE

arguments : NOMFICH : string

préconditions : NOMFICH est un fichier de la disquette utilisée, contenant un mot de passe codé

résultats : MOT : PASSWD

postconditions : MOT est le mot de passe (décodé) mémorisé dans le fichier NOMFICH

appels : /

PROCEDURE LECTUREANGLE

arguments : /

préconditions : /

résultats : ANGLE : entier

postconditions : - écriture du message 'VALEUR ANGLE (ENTIER SIGNE)'
- ANGLE est un entier signé introduit au clavier en mode graphique, modulo 360

appels : ECRITMODE
LECTUREENTIER
MODULO

procedure LECTUREENTIER

arguments : MODE : entier

préconditions : MODE appartient à [0,1]

résultats : X : entier

postconditions : X est un entier introduit au clavier en mode graphique, positif si MODE=1, négatif si MODE=0

appels : /

procedure LIRECONF

arguments : /

préconditions : /

résultats : PARALPERMIS, PERPENDPERMIS, OKPOINTREGLE, DRDEUXROT,
DRROTPERMIS, MILIEUPERMIS : booléen
MOTCOMP : PASSWD

postconditions : - lecture de la configuration mémorisée dans
le fichier VOL:CONF.TEXT et mise à jour des
variables booléennes
(variable à TRUE si la primitive concernée
est utilisable)
- s'il y a erreur dans la lecture du fichier, le
message 'ERREUR DE CONFIGURATION' est affiché
- MOTCOMP est le mot de passe mémorisé dans
le fichier VOL:BOF.TEXT

appels : LECTMOTPASSE

procedure MARQUERCENTRE

arguments : /

préconditions : /

résultats : XMARCOMP, YMARCOMP : entier
COPIE : curseur
pages graphiques 1 et 2

postconditions : - le point courant est marqué. Il est dessiné
dans la page graphique 1 si PLUME=1,
dans les deux si PLUME=2
- mise à jour de COPIE et de XMARCOMP, YMARCOMP

appels : POINTDES

procedure MARQUERPOINT

arguments : /

préconditions : /

résultats : XMAREG, YMAREG : entier
COPIE : curseur
pages graphiques 1 et 2

postconditions : - le point courant est marqué. Il est dessiné
dans la page graphique 1 si PLUME=1,
dans les deux si PLUME=2
- mise à jour de COPIE et de XMAREG, YMAREG

appels : POINTDES

procedure METTREENCRE

arguments : /

préconditions : /

résultats : page graphique 2

postconditions : copie de la page graphique 1 dans la page graphique 2

appels : /

procedure MILIEU

arguments : /

préconditions : /

résultats : pages graphiques 1 et 2

postconditions : si la configuration le permet dessine en
inverse dans la page graphique 1 si PLUME=1,
dans les deux si PLUME=2, le point milieu
du segment PTCOURANT,PTMARCOMP;
sinon le message 'PRIMITIVE PAS PERMISE'
est affiché à l'écran le temps
nécessaire à la lecture

appels : POINTDES
ERREUR

fonction MODULO

arguments : I, J : entiers

préconditions : J > 0

résultats : MODULO : entier

postconditions : MODULO = I MOD J

appels : /

procedure PARALLELE

arguments : /

préconditions : /

résultats : XCOUREG, YCOUREG : entier
COPIE : curseur
pages graphiques 1 et 2

postconditions : si la configuration le permet, trace moyennant l'usage de <- et -> la demi-droite gauche, la demi-droite droite ou les deux, passant par le point courant et de paramètres directeurs (XDIR,YDIR)
dans la page graphique 1 si PLUME=1, dans les deux si PLUME=2
- mise à jour de PTCOUREG et COPIE
- la tortue est au point courant
sinon le message 'PRIMITIVE PAS PERMISE' est affiché à l'écran

appels : PLECHEGAUCHE
PLECHEDROITE
ERREUR

procedure PERPENDICULAIRE

arguments : /

préconditions : /

résultats : XCOUREG, YCOUREG : entier
COPIE : curseur
pages graphiques 1 et 2

postconditions : si la configuration le permet, trace moyennant l'usage de <- et -> la demi-droite gauche, la demi-droite droite ou les deux, passant par le point courant et de direction orthogonale à celle de la règle dans la page graphique 1 si PLUME=1, dans les deux si PLUME=2
- mise à jour de PTCOUREG et COPIE
- la tortue est au point courant
sinon le message 'PRIMITIVE PAS PERMISE' est affiché à l'écran

appels : FLECHEGAUCHE
FLECHEDROITE
ERREUR

procedure POINTDES

arguments : X,Y : entier
MODE : entier

préconditions : 0 <= X <= LARGFEN
0 <= Y <= HAUTFEN
MODE appartient à {3,14}

résultats : COPIE : curseur
pages graphiques 1 et 2

postconditions : - dessine, en blanc si MODE=14, en inverse si MODE=3, le point courant, dans la page graphique 1 si PLUME=1, dans les deux si PLUME=2
- mise à jour de COPIE

appels : SAUVER
TRACEPAGE

procedure POINTDISTANCEECART

arguments : /

préconditions : /

résultats : COPIE : curseur
pages graphiques 1 et 2

postconditions : - dessin en inverse, dans la page graphique 1 si PLUME=1, dans les deux si PLUME=2, des deux points situés sur la droite passant par PTCOURANT, de direction égale à celle de la règle, à une distance de PTCOURANT égale à ECART
- mise à jour de COPIE

appels : POINTDES

procedure POINTMODE

arguments : /

préconditions : /

résultats : ACTION : caractère

postconditions : - écriture dans le coin supérieur droit à partir du point de coordonnées (200,HAUTFEN), du mot-clé POINT
- permet la gestion du curseur, le dessin de points
- ACTION appartient à {R,K,G,.,$,<RET>}

appels : GAUCHE
DROITE
HAUT
BAS
CHANGEPAS
ENCRECRAYON
POINTDES
IMMOBJOY
JOYCURS

procedure RESERVE

arguments : MINIMUM : entier

préconditions : /

résultats : pointeur sur le sommet du tas

postconditions : le sommet du tas est à l'adresse MINIMUM

appels : /

procedure SAUVER

arguments : X,Y : entier

préconditions : 0 <= X <= LARGFEN
0 <= Y <= HAUTFEN

résultats : AUX : curseur

postconditions : copie dans AUX la portion de l'écran égale
à celle du curseur et identifiée
par le point de coordonnées (X,Y)
(i.e. (X,Y) sont les coordonnées du coin
inférieur gauche d'un rectangle de dimensions
LARGCUR et HAUTCUR

appels : /

procedure TERMINAISON

arguments : /

préconditions : /

résultats : VOL:BOF.TEXT, VOL:CONF.TEXT : fichiers

postconditions : - enregistrement dans BOF.TEXT de MOTCOMP codé
- enregistrement dans CONF.TEXT
de la configuration

appels : ECRIREMOTPASSE

procedure TRACEPAGE

arguments : N : entier

préconditions : N appartient à [1,2]

résultats : /

postconditions : modification du paramètre utilisé
par TURTTLEGRAPHICS
pour savoir dans quelle page écrire
(Rem : cfr. annexes pour plus de détails)

appels : /

procedure TRACERCERCLE

arguments : XCEN, YCEN, RAYON : entier

préconditions : 0 <= XCEN <= LARGFEN
0 <= YCEN <= HAUTFEN
0 < RAYON

résultats : page graphique sélectionnée

postconditions : - tracé du cercle de centre (XCEN,YCEN) et de rayon égal à RAYON, dans la page graphique sélectionnée
- la tortue est au point courant

appels : /

procedure TRACERDROITE

arguments : A,B : entier

préconditions : 0 <= A <= LARGFEN
0 <= B <= HAUTFEN

résultats : XDIR, YDIR, XCOUREG, YCOUREG : entier
COPIE : curseur
pages graphiques 1 et 2

postconditions : - tracé du segment joignant le point courant au point de coordonnées (A,B) dans la page graphique 1 si PLUME=1, dans les deux si PLUME=2
- mise à jour de la direction de la règle
- la tortue est au point courant
- modification de COPIE

appels : ECRSAUVER

PROGRAMME

```
(*$S+*)

unit OUTILS;


interface

uses turtlegraphics,applestuff;

const largcur = 5;
      hautcur = 5;
      (*DIMENSIONS DU CURSEUR*)

      largfen = 278;
      hautfen = 179;
      (*DIMENSIONS DU CADRE*)

      grandpas = 5;
      petitpas = 1;
      (*DEPLACEMENTS POSSIBLES DU CURSEUR*)

      vol = 'SIMPAS3:';
      saut = 10;

      cache = 12;
      longpass = 5;
      (*LONGUEUR DU MOT DE PASSE*)


type curseur = packed array [1..hautcur,1..largcur] of boolean;

     pagegraph = record
                   case boolean of
                        false : (adr : integer);
                        true  : (padr : ^integer);
                        end;

     passwd = string[5];


var page1,page2 : pagegraph;
    f : text;

    action : char;

    xcourant, ycourant : integer;
    (*POSITION DE LA FLECHE,POSITION DU LA TORTUE*)

    xmareg, ymareg : integer;
    (*POSITION DU DERNIER POINT MARQUE DE LA REGLE*)

    xcoureg, ycoureg : integer;
    (*DERNIER POINT COURANT DE LA REGLE*)

    xdir,ydir : integer;
    (*DIRECTION DE LA REGLE*)

    xmarcomp,ymarcomp : integer;
    (*DERNIER POINT MARQUE DU COMPAS*)
```

```
    xcenjoy,ycenjoy : integer;
    xpad,ypad : integer;

    plume : 1..2;
    (*1 = MODE CRAYON, 2 = MODE ENCRE*)


    ecart : integer;
    angle : integer;
    tab : array [1..31] of real;

    copie, fleche : curseur;
    (*ETAT DE L'ECRAN SOUS LE CURSEUR*)
    (*DESSIN CHOISI POUR LE CURSEUR*)

    pas : integer;
    despoint : boolean;
    paralpermis,perpendpermis,okpointregle : boolean;
    drdeuxrot,drrotpermis,milieupermis : boolean;
    motcomp : passwd;

procedure ECRITMODE(x:integer;s:string);
procedure LECTUREENTIER(mode:integer;var x :integer);
procedure TRACEPAGE(n:integer);
procedure SAUVER(x,y:integer;var aux:curseur);
procedure DESSINCURSEUR(x,y:integer);
procedure POINTDES(x,y,mode : integer);
function  MODULO(i,j:integer):integer;
procedure GAUCHE(i:integer);
procedure DROITE(i:integer);
procedure HAUT(i:integer);
procedure BAS(i:integer);
procedure CHANGEPAS;
procedure ENCRECRAYON;
procedure ERREUR;
function  IMMOBJOY:boolean;
procedure JOYCURS;

implementation

(*************************************)

procedure ECRITMODE;
(*ECRIT LA VALEUR DU STRING A PARTIR DU POINT*)
(*DE COORDONNEE (X,HAUTFEN+2)*)


begin
viewport(0,279,0,189);
pencolor(none);
moveto(x,hautfen+2);
wstring(s);
moveto(xcourant,ycourant);
pencolor(white);
viewport(0,largfen,0,hautfen);
end;


(*****************************)
```

```
procedure LECTUREENTIER;
(*TRAITE, LIT ET ECRIT, EN MODE GRAPHIQUE*)
(*UN ENTIER INTRODUIT AU CLAVIER,*)
(* POSITIF SI MODE=0, NEGATIF SI MODE=1*)
(*AVEC GESTION DE LA <-, POUR EFFACEMENT*)

var negatif : boolean;
    c : char;
    compt,pos : integer;
    (*POS EST ABS RELATIVE DE LA POSIT DU CURSEUR*)
    (*COMPT EST LE NOMBRE DE CHIFFRES INTRODUITS*)

(**************)

function CAROK(car : char):boolean;
(*RENVOIT LA VALEUR TRUE SI CAR EST UN ENTIER*)
(*OU RETURN OU '-'*)

begin
carok:=(((47<ord(car)) and (ord(car)<58))
          or (ord(car) = 32) or (ord(c) = 45))
end;

(**************)

procedure ECRITCHAR(x:integer; s:char);
(*ECRIT LE CHAR A PARTIR DU POINT DE  *)
(*COORDONNEE (X,HAUTFEN+2)*)

begin
viewport(0,279,0,189);
pencolor(none);
moveto(x,hautfen+2);
pencolor(white);
wchar(s);
viewport(0,largfen,0,hautfen);
end;

(*************)

procedure TRAITSIGNE;

begin
if (pos=14)
              then begin
                   ecritchar(pos,c);
                   negatif:=true;
                   pos:=pos+7;
                   end
end;

(***************)

procedure TRAITFLECHE;

begin
if pos >= 21 then begin
             x:=x div 10;
             pos:=pos-7;
             ecritchar(pos,' ');
             compt:=compt-1;
                  end;
```

```
if pos=14 then begin
               if negatif then compt:=compt+1;
               negatif:=false
               end
end;

(*****************)

procedure ECRITCHIFFRE;

begin
x:=x*10+ord(c)-48;
ecritchar(pos,c);
pos:=pos+7;
compt:=compt+1;
end;

(******************)

begin (*CORPS DE LA PROCEDURE LECTUREENTIER*)
x:=0;
negatif:=false;
compt:=0;
repeat read(keyboard,c) until carok(c);

ecritmode(0,'                                  ');
ecritmode(0,'->');
pos:=14;

while ((ord(c)<>32) and (compt<4)) do
         begin
         if (ord(c) = 45) then begin
                    if mode=1 then traitsigne;
                               end;
         if (ord(c) = 8) then traitfleche;
         if (ord(c) in [48..57]) then ecritchiffre;
         repeat read(keyboard,c) until (carok(c) or (ord(c)=8))
         end;
if negatif then x:=-x;
pencolor(none);
moveto(xcourant,ycourant);
pencolor(white);
ecritmode(0,'                                  ');
end;

(*****************************)

procedure TRACEPAGE;
(*MODIFIE LA VALEUR DU PARAMETRE INDIQUANT*)
(*DANS QUELLE PAGE TURTLEGRAPHICS ECRIT*)


var trpage : pagegraph;

begin
trpage.adr:=254;
trpage.adr:=trpage.padr^ +14;
trpage.padr^:=n
end;
```

```
(****************************************)
(****************************************)

procedure SAUVER;
(*SAUVE L'ETAT DE L'ECRAN SUR UNE SURFACE*)
(*EGALE A CELLE DU CURSEUR*)
(*X,Y COIN INFERIEUR GAUCHE DE LA*)
(*PARTIE A SAUVER*)

var itx,ity : integer;

begin
for itx:=1 to hautcur do
            begin
            for ity:=1 to largcur do
                        begin
                        aux[itx,ity]:=screenbit(x+ity-1,y+itx-1);
                        end
            end
end;

(****************************************)

procedure DESSINCURSEUR;
(*DEPLACE LE CURSEUR ET LA TORTUE DE SA*)
(*POSITION COURANTE VERS X,Y ==> *)
(*M.A.J. DE PTCOURANT*)
(*RESTITUTION DE L'ETAT DE L'ECRAN SOUS LE CURSEUR*)
(*SAUVETAGE DE LA POSITION PROCHAINE DU CURSEUR*)


begin
drawblock(copie,2,0,0,largcur,hautcur,xcourant,ycourant,10);
sauver(x,y,copie);
drawblock(fleche,2,0,0,largcur,hautcur,x,y,14);
pencolor(none);
moveto(x,y);
pencolor(white);
xcourant:=x;
ycourant:=y;
end;

(****************************************)

procedure POINTDES;
(*DESSINE LE POINT COURANT DANS LA PREMIERE*)
(*PAGE GRAPHIQUE ET DANS LA SECONDE, SI PLUME=2*)
(*MISE A JOUR DE LA COPIE DE L'ECRAN*)

begin
drawblock(copie,2,0,0,largcur,hautcur,xcourant,ycourant,10);
drawblock(despoint,1,0,0,1,1,x,y,mode);
sauver(xcourant,ycourant,copie);
drawblock(fleche,2,0,0,largcur,hautcur,xcourant,ycourant,14);
if plume=2 then begin
            tracepage(2);
            drawblock(despoint,1,0,0,1,1,x,y,mode);
            tracepage(1)
                end;
end;
```

```
(*****************************************)

function MODULO;

var k : integer;

begin
if (i>=0) then modulo:=i mod j
          else begin
               k:=i;
               while k<0 do k:=k+j;
               modulo:=k
               end
end;


(*****************************************)

procedure GAUCHE;
(*DEPLACE LE CURSEUR ET LA TORTUE DE I*)
(*POINTS VERS LA GAUCHE. MAJ DU PT COURANT*)

begin
dessincurseur(modulo(xcourant-i,largfen),ycourant);
end;

(*****************************************)

procedure DROITE;

begin
dessincurseur(modulo(xcourant+i,largfen),ycourant);
end;

(*****************************************)

procedure HAUT;

begin
dessincurseur(xcourant,modulo(ycourant+i,hautfen));
end;

(*****************************************)

procedure BAS;

begin
dessincurseur(xcourant,modulo(ycourant-i,hautfen));
end;

(*****************************************)

procedure CHANGEPAS;

begin
if pas=grandpas then pas:=petitpas
                else pas:=grandpas
end;

(*****************************************)
```

```
procedure ENCRECRAYON;

begin
if plume = 1 then begin
                  plume:=2;
                  ecritmode(242,' (E)')
                  end
             else begin
                  plume:=1;
                  ecritmode(242,' (C)')
                  end;
end;

(*****************************)

procedure ERREUR;

var attente : integer;

begin
ecritmode(0,'PRIMITIVE PAS PERMISE');
write(chr(7));
for attente:=1 to 150 do;
ecritmode(0,'                      ')
end;

(*****************************)


function IMMOBJOY;
(*MET A JOUR XPAD ET YPAD*)

var temps : integer;

begin
xpad:=paddle(4);
for temps:=1 to 3 do;
ypad:=paddle(1);
immobjoy:=(xcenjoy-saut<xpad) and (xpad<xcenjoy+saut)
      and (ycenjoy-saut<ypad) and (ypad<ycenjoy+saut);
end;


procedure JOYCURS;

var xdep,ydep : integer;

begin
xdep:= (xpad - xcenjoy) div saut;
ydep:= (ycenjoy - ypad) div saut;
dessincurseur(modulo(xcourant+xdep,largfen),
                  modulo(ycourant+ydep,hautfen));
end;


(*************************************)
(*************************************)


begin
end.
```

```
(*$N+*)
(*$S+*)

unit REGLE;


interface

uses turtlegraphics,applestuff,
     (*$U #5:OUTI.CODE*)outils;

procedure LATTE;
procedure POINTMODE;

implementation


procedure LATTE;

var actreg : char;
    caractok : boolean;
    i : integer;

(*****************************)

procedure MARQUERPOINT;
(*LE PT CRT EN MODE REGLE EST MARQUE ET DESSINE*)

begin
xmareg:=xcourant;
ymareg:=ycourant;
pointdes(xcourant,ycourant,14);
end;

(*****************************)

procedure ECRSAUVER(c,d:integer);
(*REALISE MOVETO(C,D) EN FAISANT LES MODIF*)
(*EVENTUELLES NEC DANS COPIE*)
(*RETOUR AU POINT COURANT*)

begin
drawblock(copie,2,0,0,largcur,hautcur,xcourant,ycourant,10);
moveto(c,d);

if plume=2 then begin
                tracepage(2);
                moveto(xcourant,ycourant);
                tracepage(1)
                end
           else moveto(xcourant,ycourant);

sauver(xcourant,ycourant,copie);
drawblock(fleche,2,0,0,largcur,hautcur,xcourant,ycourant,14);
end;

(*****************************)
```

```
procedure TRACERDROITE(a,b:integer);

(*TRACE LE SGT JOIGNANT LE PT COURANT AU*)
(*POINT DE COORD (A,B) - M.A.J. DE PTMAREG*)
(*M.A.J. DE LA DIR DE LA REGLE*)

begin
ecrsauver(a,b);
xdir:=a - xcourant;
ydir:=b - ycourant;
xcoureg:=xcourant;
ycoureg:=ycourant
end;

(*****************************)

function CALCULPOINT(a,b,c : integer):integer;

(*CALCULE L'ORDONNEE DU POINT D'ABSCISSE C,*)
(*APPARTENANT A LA DROITE PASSANT PAR LE POINT*)
(*COURANT, DE VECT DIRECTEUR (A,B), SI CETTE*)
(*DIRECTION N'EST PAS VERTICALE, C-A-D A<>0*)

begin
calculpoint:=round((b*c - b*xcourant + a*ycourant) / a)
end;

(*****************************)

procedure FLECHEGAUCHE(a,b : integer);

(*TRACE LA PARTIE GAUCHE DE LA DROITE PASSANT*)
(*PAR LE POINT COURANT ET DE VECT DIRECTEUR*)
(*(A,B). ET RETOUR AU PTCOURANT*)
(*CONVENTION: SI LA DROITE EST VERTICALE*)
             (*ON TRACE VERS LE HAUT*)
(*MODIFIE COPIE*)

begin
if a=0 then ecrsauver(xcourant,hautfen)
       else ecrsauver(0,calculpoint(a,b,0));
end;

(*****************************)

procedure FLECHEDROITE(a,b : integer);

(*TRACE LA PARTIE DROITE DE LA DROITE PASSANT*)
(*PAR LE POINT COURANT ET DE VECT DIRECTEUR*)
(*(A,B). ET RETOUR AU PTCOURANT*)
(*CONVENTION: SI LA DROITE EST VERTICALE*)
             (*ON TRACE VERS LE BAS *)
(*MODIFIE COPIE*)

begin
if a=0 then ecrsauver(xcourant,0)
       else ecrsauver(largfen,calculpoint(a,b,largfen));
end;

(*****************************)
```

```
procedure PARALLELE;

(*TRACE, MOYENNANT L'USAGE DE --> ET <--, LA*)
(*DROITE PASSANT PAR LE POINT COURANT DE DIR*)
(*EGALE A CELLE DE LA REGLE*)

var c : char;
    compt : integer;

begin
if paralpermis then begin
                read(keyboard,c);
                compt:=0;
                while (ord(c) in [8,21]) and (compt<2) do begin
                            compt:=compt+1;
                            if ord(c) = 8 then flechegauche(xdir,ydir)
                                          else flechedroite(xdir,ydir);
                            read(keyboard,c)
                                          end;
                actreg:=c;
                xcoureg:=xcourant;
                ycoureg:=ycourant
                     end
                else begin
                     erreur;
                     read(keyboard,actreg)
                     end;

end;

(*****************************)


procedure PERPENDICULAIRE;

(*TRACE, MOYENNANT L'USAGE DE --> ET <--, LA*)
(*DROITE PASSANT PAR LE POINT COURANT DE DIR*)
(*ORTHOGONALE A CELLE DE LA REGLE*)

var c : char;
    compt : integer;

begin
if perpendpermis then begin
                  read(keyboard,c);
                  compt:=0;
                  while (ord(c) in [8,21]) and (compt<2) do begin
                              compt:=compt+1;
                              if ord(c) = 8 then flechegauche(-ydir,xdir)
                                            else flechedroite(-ydir,xdir);
                              read(keyboard,c)
                                            end;
                  actreg:=c;
                  xcoureg:=xcourant;
                  ycoureg:=ycourant
                       end
                  else begin
                       erreur;
                       read(keyboard,actreg)
                       end;
end;
```

```
(*****************************)

procedure DESSINPOINT;

begin
if okpointregle then begin
                pointdes(xcourant,ycourant,14);
                xcoureg:=xcourant;
                ycoureg:=ycourant
                     end
                else erreur
end;

(*****************************)

begin
(*CORPS DE LA PROCEDURE REGLE*)

ecritmode(200,'REGLE ');
read(keyboard,actreg);
while (ord(actreg)<>32) and (actreg<>'K') and
    (actreg<>'G') and (ord(actreg)<>36) and (actreg <> '.')
        do begin

        caractok:=false;
        case ord(actreg) of
             8  : gauche(pas); (*<-*)
             21 : droite(pas); (*->*)
           65,97 : haut(pas); (*A*)
           90,122: bas(pas); (*Z*)
           67,99 : changepas; (*C*)
             47 : encrecrayon; (*/*)
          77,109 : marquerpoint; (*M*)
          68,100 : dessinpoint; (*D*)
          83,115 : tracerdroite(xcoureg,ycoureg); (*S*)
             42 : tracerdroite(xmareg,ymareg); (***)
          80,112 : begin
                   caractok:=true; (*P*)
                   parallele
                   end;
          79,111 : begin
                   caractok:=true; (*O*)
                   perpendiculaire;
                   end;
                      end;
      if not caractok then
                      begin
                      repeat begin
                            while not immobjoy
                                do begin
                                   joycurs;
                                   for i:=1 to 100 do;
                                   end;
                            end
                         until keypress;
                      read(keyboard,actreg)
                      end;

                        end;
action:=actreg
end;
```

```
(*************************************)

procedure POINTMODE;

var actpoint : char;
    i : integer;

begin

ecritmode(200,'POINT ');
read(keyboard,actpoint);
while (ord(actpoint) <> 32) and (actpoint <> 'K')
             and (actpoint <> 'R') and (actpoint <> 'G')
                    and (ord(actpoint) <> 36)
        do begin

        case ord(actpoint) of
             8  : gauche(pas); (*<-*)
             21 : droite(pas); (*->*)
          65,97 : haut(pas); (*A*)
          90,122: bas(pas); (*Z*)
          67,99 : changepas; (*C*)
             47 : encrecrayon; (*/*)
         68,100 : pointdes(xcourant,ycourant,14); (*D*)
             end;
        repeat begin
             while not immobjoy do
                        begin
                        joycurs;
                        for i:=1 to 100 do;
                        end;
               end
        until keypress;

        read(keyboard,actpoint)
           end;
action:=actpoint
end;


begin
end.
```

```
(*$N+*)
(*$S+*)

unit ROTA;

interface

uses turtlegraphics,applestuff,transcend,
     (*$U #5:OUTI.CODE*)outils;

procedure COORDPOLAIRE(var dist,coangle,siangle : real);
procedure ADDANGLE(cosa,sina:real;b:integer;var coab,siab : real);
procedure DEUXDRROTATION(ang : integer);
procedure DROITEROTATION(ang : integer);
procedure POINTDISTANCEECART;

implementation


procedure COORDPOLAIRE;
(*CALCUL DES COORD. POLAIRES DE PTCOURANT*)
(*PAR RAPPORT A PTMARCOMP*)

begin
dist:=sqrt(((xcourant-xmarcomp)
                    *(xcourant-xmarcomp))
                    +((ycourant-ymarcomp)
                       *(ycourant-ymarcomp)));
if dist <> 0 then begin
      coangle:=((xcourant-xmarcomp) / dist);
      siangle:=((ycourant-ymarcomp) / dist)
                  end;
end;


(*****************************)

procedure ADDANGLE;
(*SI A EST L'ANGLE DETERMINE PAR COSA ET SINA*)
(*ALORS COAB = COS(A+B) ET SIAB = SIN(A+B)*)

var brad : real;

begin
brad:=(b / 180) * 3.1416;
coab:=cosa*cos(brad) - sina*sin(brad);
siab:=sina*cos(brad) + cosa*sin(brad);
end;

(*****************************)

procedure DEUXDRROTATION;

var dist,coangle,siangle,coadd,siadd : real;

begin
if drdeuxrot then begin
   if (xcourant<>xmarcomp) or (ycourant<>ymarcomp)
   then begin
        coordpolaire(dist,coangle,siangle);
        addangle(coangle,siangle,ang,coadd,siadd);
```

```
        pencolor(none);
        moveto(xmarcomp,ymarcomp);
        pencolor(white);
        drawblock(copie,2,0,0,largcur,hautcur,xcourant,ycourant,10);
        moveto(round(2*dist*coadd) + xmarcomp,
                        round(2*dist*siadd) + ymarcomp);
        if plume=2 then begin
                        tracepage(2);
                        moveto(xmarcomp,ymarcomp);
                        tracepage(1)
                        end
                   else moveto(xmarcomp,ymarcomp);
        addangle(coangle,siangle,-ang,coadd,siadd);
        moveto(round(2*dist*coadd) + xmarcomp,
                        round(2*dist*siadd) + ymarcomp);
        if plume=2 then begin
                        tracepage(2);
                        moveto(xmarcomp,ymarcomp);
                        tracepage(1)
                        end;
        pencolor(none);
        moveto(xcourant,ycourant);
        pencolor(white);
        sauver(xcourant,ycourant,copie);
        drawblock(fleche,2,0,0,largcur,hautcur,xcourant,ycourant,14);
        end
                end
else erreur;
end;

(*****************************)

procedure DROITEROTATION;

var dist,coangle,siangle,coadd,siadd : real;


begin
if drrotpermis then begin
     if (xcourant<>xmarcomp) or (ycourant<>ymarcomp)
     then begin
          coordpolaire(dist,coangle,siangle);
          addangle(coangle,siangle,ang,coadd,siadd);
          pencolor(none);
          moveto(xmarcomp,ymarcomp);
          pencolor(white);
          drawblock(copie,2,0,0,largcur,hautcur,xcourant,ycourant,10);
          moveto(round(2*dist*coadd) + xmarcomp,
                          round(2*dist*siadd) + ymarcomp);
          if plume=2 then begin
                          tracepage(2);
                          moveto(xmarcomp,ymarcomp);
                          tracepage(1)
                          end;


          pencolor(none);
          moveto(xcourant,ycourant);
          pencolor(white);
          sauver(xcourant,ycourant,copie);
          drawblock(fleche,2,0,0,largcur,hautcur,xcourant,ycourant,14)
          end
```

```
                    end
                else erreur
end;

(*****************************)

procedure POINTDISTANCEECART;
(*RENVOIT EN INVERSE LES POINTS SITUES, SUR LA*)
(*DROITE PASSANT PAR PTCOURANT, DE DIRECTION*)
(*EGALE A CELLE DE LA REGLE, A UNE DISTANCE*)
(*EGALE A ECART*)

var xint,yint : integer;

begin
if xdir <> 0 then begin
   xint:=round(sqrt((ecart*ecart)/
              (1+(ydir/xdir)*(ydir/xdir))));
   yint:=round((ydir/xdir)*xint);
                end
            else begin
                xint:=0;
                yint:=ecart;
                end;
pointdes(xcourant + xint, ycourant + yint, 3);
pointdes(xcourant - xint, ycourant - yint, 3);
end;


(*************************************)
(*************************************)

begin
end.
```

```
(*$N+*)
(*$S+*)

unit COMP;

interface


uses turtlegraphics,applestuff,transcend,
      (*$U #5:OUTI.CODE*)outils,
      (*$U #5:DROIT.CODE*)rota;

procedure COMPAS;


implementation

(*****************************)

procedure COMPAS;

var actcomp : char;
    i : integer;

(*****************************)

procedure MARQUERCENTRE;
(*MARQUE ET DESSINE LE POINT COURANT*)


begin
xmarcomp:=xcourant;
ymarcomp:=ycourant;
pointdes(xcourant,ycourant,14);
end;

(*****************************)

procedure ECARTENTIER;
(*LIT A L'ECRAN UNE DISTANCE ET DONNE CETTE VALEUR A ECART*)

begin
ecritmode(0,'ENTREZ UNE DISTANCE (>=0)');
lectureentier(0,ecart);
ecart:=modulo(ecart,largfen);
end;

(*****************************)

procedure ECARTDISTANCE;
(*ECART:=D(POINT COURANT,DERNIER POINT MARQUE)*)

begin
ecart:=round(sqrt(((xcourant-xmarcomp)
                   *(xcourant-xmarcomp))
                   +((ycourant-ymarcomp)
                      *(ycourant-ymarcomp))));
end;

(*****************************)
```

```
procedure LECTUREANGLE;
(*LECTURE A L'ECRAN D'UN ANGLE SIGNE*)

begin
ecritmode(0,'VALEUR ANGLE (ENTIER SIGNE)');
lectureentier(1,angle);
angle:=modulo(angle,360);
end;

(*****************************)

procedure TRACERCERCLE(xcen,ycen,rayon : integer);
(*TRACE LE CERCLE DE CENTRE (XCENTRE,YCENTRE)*)
(*ET DE RAYON EGAL A RAYON*)
(*RETOUR DE LA TORTUE AU PTCOURANT*)

var i,e1,e2 : integer;

begin
pencolor(none);
moveto(xcen+rayon,ycen);
pencolor(white);

for i:=2 to 31 do
begin
    e1:=round(rayon*tab[32-i]);
    e2:=round(rayon*tab[i]);
    moveto(xcen+e1,ycen+e2);
end;

for i:=2 to 31 do
begin
    e1:=round(-rayon*tab[i]);
    e2:=round(rayon*tab[32-i]);
    moveto(xcen+e1,ycen+e2);
end;

for i:=2 to 31 do
begin
    e1:=round(-rayon*tab[32-i]);
    e2:=round(-rayon*tab[i]);
    moveto(xcen+e1,ycen+e2);
end;

for i:=2 to 31 do
begin
    e1:=round(rayon*tab[i]);
    e2:=round(-rayon*tab[32-i]);
    moveto(xcen+e1,ycen+e2);
end;

pencolor(none);
moveto(xcourant,ycourant);
pencolor(white);
end;

(*****************************)
```

```
    procedure CERCLE;
    (* TRACE LE CERCLE DE CENTRE (XMARCOMP,YMARCOMP)*)
    (*ET DE RAYON EGAL A ECART*)

    begin
    if ecart <> 0 then begin
          drawblock(copie,2,0,0,largcur,hautcur,xcourant,ycourant,10);
          tracercercle(xmarcomp,ymarcomp,ecart);
          drawblock(despoint,1,0,0,1,1,xmarcomp,ymarcomp,14);
          sauver(xcourant,ycourant,copie);
          drawblock(fleche,2,0,0,largcur,hautcur,xcourant,ycourant,14);
          if plume=2 then begin
                     tracepage(2);
                     tracercercle(xmarcomp,ymarcomp,ecart);
                     drawblock(despoint,1,0,0,1,1,xmarcomp,ymarcomp,14);
                     tracepage(1)
                     end;
          end;
    end;

    (*****************************)

    procedure MILIEU;
    (*RENVOIT LE POINT MILIEU DU SEGMENT*)
    (*[DERNIER POINT MARQUE, POINT COURANT]*)

    var xmil,ymil : integer;

    begin
    if milieupermis then begin
                     if (xmarcomp > xcourant)
                                then xmil:=round((xmarcomp-xcourant)/2) +
xcourant
                                else xmil:=round((xcourant-xmarcomp)/2) +
xmarcomp;
                     if (ymarcomp > ycourant)
                                then ymil:=round((ymarcomp-ycourant)/2) +
ycourant
                                else ymil:=round((ycourant-ymarcomp)/2) +
ymarcomp;

                     pointdes(xmil,ymil,3);
                     end
                     else erreur
    end;

    (*****************************)

    begin
    (*CORPS DE LA PROCEDURE COMPAS*)

    ecritmode(200,'COMPAS');
    read(keyboard,actcomp);
    while (ord(actcomp) <> 32) and (actcomp <> 'R') and (actcomp<>'G')
                     and (ord(actcomp)<>36) and (actcomp <> '.')
                               do begin
              case ord(actcomp) of
                   8  : gauche(pas); (*<-*)
                   21 : droite(pas); (*->*)
                65,97 : haut(pas); (*A*)
                90,122: bas(pas); (*Z*)
                67,99 : changepas; (*C*)
```

```
            47 : encrecrayon; (*/*)
         77,109 : marquercentre; (*M*)
         69,101 : ecartentier; (*E*)
         76,108 : lectureangle; (*L*)
         68,100 : ecartdistance; (*D*)
         79,111 : cercle; (*O*)
         80,112 : pointdistanceecart; (*P*)
            50 : deuxdrrotation(angle); (*2*)
            49 : droiterotation(angle); (*1*)
          66,98 : milieu; (*B*)
                  end;
        repeat begin
               while not immobjoy do
                           begin
                           joycurs;
                           for i:=1 to 100 do;
                           end;
               end
        until keypress;
        read(keyboard,actcomp)
                  end;
action:=actcomp
end;

(*************************************)
(*************************************)

begin
end.
```

```
(*$N+*)
(*$S+*)

unit GOM;

interface


uses turtlegraphics,applestuff,
      (*$U #5:OUTI.CODE*)outils;

procedure GOMME;

implementation

(*****************************)

procedure GOMME;

var actgom : char;

(******************)

procedure POKE(loc,cont:integer);

type byte = packed array[0..1] of 0..255;

     locbyte = record
                 case boolean of
                      true : (int:integer);
                      false: (ptr:^byte)
                      end;

var x:locbyte;

begin
x.int:=loc;
x.ptr^[0]:=cont
end;

(******************)

procedure AFFPAGE(n:integer);
(*AFFICHE LA N-EME PAGE*)

begin
if n=1 then poke(-16300,0)
       else poke(-16299,0)
end;

(******************)

procedure EFFCRAYON;
(*COPIE LA PAGE GRAPHIQUE 2 DANS LA PAGE GRAPHIQUE 1*)

begin
drawblock(copie,2,0,0,largcur,hautcur,xcourant,ycourant,10);
moveleft(page2.padr^,page1.padr^,8192);
sauver(xcourant,ycourant,copie);
drawblock(fleche,2,0,0,largcur,hautcur,xcourant,ycourant,14);
if plume = 1 then ecritmode(242,' (C)')
```

```
            else ecritmode(242,' (E)')
end;

(******************)

procedure CLIGNOTER;
(*AFFICHE EN ALTERNANCE LA PAGE GRAPHIQUE1*)
(*ET LA PAGE GRAPHIQUE 2*)

var inc : integer;

begin
repeat begin
       affpage(2);
       for inc:=1 to 400 do;
       affpage(1);
       for inc:=1 to 400 do;
       end
until keypress
end;

(******************)

procedure METTREENCRE;
(*COPIE LA PAGE GRAPHIQUE 1 DANS LA *)
(*PAGE GRAPHIQUE 2*)

begin
drawblock(copie,2,0,0,largcur,hautcur,xcourant,ycourant,10);
(*AFIN DE NE PAS SAUVER LA FLECHE*)
moveleft(page1.padr^,page2.padr^,8192);
drawblock(fleche,2,0,0,largcur,hautcur,xcourant,ycourant,14);
end;

(******************)

begin (*CORPS DE GOMME*)
ecritmode(200,'GOMME ');
read (keyboard,actgom);
while (ord(actgom)<>32) and (actgom<>'K') and (actgom<>'R')
      and (ord(actgom)<>36) and (actgom<>',')

      do begin
         case ord(actgom) of

        69,101 : effcrayon; (*E*)
         67,99 : clignoter; (*C*)
        77,109 : mettreencre; (*M*)
                 end;

         read(keyboard,actgom)
         end;
action:=actgom
end;

(*******************************)


begin
end.
```

```
(*$S++*)
program PRINCIPAL;

uses turtlegraphics,applestuff,transcend,
     (*$U #5:OUTI.CODE*)outils,
     (*$U #5:DROIT.CODE*)rota,
     (*$U #5:REG.CODE*)regle,
     (*$U #5:COMP.CODE*)comp,
     (*$U #5:GOM.CODE*)gom;


segment procedure LECTMOTPASSE(nomfich:string;var mot:passwd);
(*EN SORTIE, MOT EST LE MOT DE PASSE *)
(*STOCKE SUR LE FICHIER VOL:NOMFICH.TEXT*)

var f : text;
    i : integer;


begin
reset(f,nomfich);
readln(f,mot);
for i:=1 to longpass do
      mot[i]:=chr(ord(mot[i]) + cache);
close(f);
end;

(************)


segment procedure INIT;


segment procedure LIRECONF;
(*LIT LA CONFIGURATION ET LE MOT DE PASSE*)
(*STOCKES SUR FICHIERS*)


var f : text;
    ok : integer;
    c : char;


begin
reset(f,concat(vol,'CONF.TEXT'));

page(output);
readln(f,ok);
if eof(f) then writeln('ERREUR DE CONFIGURATION');
if ok = 1 then paralpermis:=true
          else paralpermis:=false;

readln(f,ok);
if eof(f) then writeln('ERREUR DE CONFIGURATION');
if ok = 1 then perpendpermis:=true
          else perpendpermis:=false;

readln(f,ok);
if eof(f) then writeln('ERREUR DE CONFIGURATION');
```

```
if ok = 1 then okpointregle:=true
          else okpointregle:=false;

readln(f,ok);
if eof(f) then writeln('ERREUR DE CONFIGURATION');
if ok = 1 then drdeuxrot:=true
          else drdeuxrot:=false;

readln(f,ok);
if eof(f) then writeln('ERREUR DE CONFIGURATION');
if ok = 1 then drrotpermis:=true
          else drrotpermis:=false;

read(f,ok);
if eof(f) then writeln('ERREUR DE CONFIGURATION');
if ok = 1 then milieupermis:=true
          else milieupermis:=false;

readln(f,ok);
if not eof(f) then writeln('ERREUR DE CONFIGURATION');
lectmotpasse(concat(vol,'BOF.TEXT'),motcomp);
end;


(*****************************)

segment procedure INITPOINTECRAN;
(*INITIALISE L'ECRAN ET LE CURSEUR*)


procedure INITCURSEUR(var aux : curseur);
(*INITIALISE LE CURSEUR*)

var i,j : integer;

begin
   for i:=1 to hautcur do begin
                    for j := 1 to largcur do begin
                              aux[i,j]:=false
                                   end;
                    end;

   aux[1,1]:=true;aux[2,2]:=true;
   aux[3,3]:=true;aux[4,4]:=true;
   aux[5,5]:=true;aux[1,2]:=true;
   aux[1,3]:=true;aux[2,1]:=true;
   aux[3,1]:=true;
end;

(*****************************)

procedure INITAUX(var aux : curseur);

var i, j : integer;

begin
for i:=1 to hautcur do begin
           for j:=1 to largcur do begin
                     aux[i,j]:=false
                          end
                end
end;
```

```
begin (*CORPS DE INITPOINTECRAN*)
initturtle;
plume:=1;
initcurseur(fleche);
initaux(copie);
pas:=grandpas;
pencolor(none);
moveto(0,0);
pencolor(white);
moveto(0,hautfen);
moveto(largfen,hautfen);
moveto(largfen,0);
moveto(0,0);
xcourant:=140;
ycourant:=90;
pencolor(none);
moveto(140,90);
pencolor(white);
despoint:=true;
ecritmode(242,' (C)');
end;

(*****************************)

procedure INITGOMME;
(*INIT DE LA DECONDE PAGE GRAPHIQUE*)
(*ET RESERVATION DU TAS*)

procedure RESERVE(minimum:integer);

var tasres : pagegraph;

begin
tasres.adr:=minimum;
release(tasres.padr)
end;

(************)

begin
reserve(24576);
page1.adr:=8192;
page2.adr:=16384;
moveleft(page1.padr^,page2.padr^,8192);
tracepage(2);
ecritmode(200,'GOMME  (C)');
tracepage(1);
end;

(*****************************)

procedure INITPADDLE;

var i : integer;
    pad : char;

begin
page(output);
```

```
write('UTILISEZ-VOUS UN PADDLE ?  ');
repeat read(keyboard,pad) until
       ord(pad) in [78,79,110,111];
writeln(pad);
if ord(pad) in [79,111]
   then begin
        writeln('CONNECTEZ LE PADDLE');
        writeln('EST-CE FAIT ?');
        repeat read(keyboard,pad) until
               ord(pad) in [79,111];
        end;
xcenjoy:=paddle(4);
for i:=1 to 3 do;
ycenjoy:=paddle(1);
end;

(*****************************)

procedure INITCOMPAS;
(*INIT DE REGLE ET DE COMPAS*)


begin
xmareg:=xcourant;
ymareg:=ycourant;
xcoureg:=xcourant;
ycoureg:=ycourant;
xdir:=0;
ydir:=1;
xmarcomp:=xcourant;
ymarcomp:=ycourant;
angle:=0;
ecart:=0;
tab[1] :=0;       tab[2] :=0.0523; tab[3] :=0.1045;
tab[4] :=0.1564; tab[5] :=0.2079; tab[6] :=0.2588;
tab[7] :=0.3090; tab[8] :=0.3583; tab[9] :=0.4067;
tab[10]:=0.4539; tab[11]:=0.5   ; tab[12]:=0.5446;
tab[13]:=0.5877; tab[14]:=0.6293; tab[15]:=0.6691;
tab[16]:=0.7071; tab[17]:=0.7431; tab[18]:=0.7771;
tab[19]:=0.8090; tab[20]:=0.8386; tab[21]:=0.8660;
tab[22]:=0.8910; tab[23]:=0.9135; tab[24]:=0.9335;
tab[25]:=0.9510; tab[26]:=0.9659; tab[27]:=0.9781;
tab[28]:=0.9876; tab[29]:=0.9945; tab[30]:=0.9986;
tab[31]:=1

end;

(*****************************)

begin (*CORPS DE INIT*)
initpaddle;
lireconf;
initpointecran;
initgomme;
initcompas;
dessincurseur(xcourant,ycourant);
end;


(*****************************)
```

```
segment procedure CONFIGURATION;

var choix :char;

(*************)

segment procedure CHANGEMOTPASSE;

var nouv1passe,nouv2passe : passwd;
    temps : integer;

begin
page(output);
writeln('QUEL EST LE NOUVEAU MOT DE PASSE ?');
writeln('(5 CARACTERES)  <RET> ');
readln(keyboard,nouv1passe);
writeln('ENCORE UNE FOIS');
readln(keyboard,nouv2passe);
if nouv1passe=nouv2passe
   then motcomp:=nouv1passe
   else begin
        writeln('ERREUR DANS L''INTRODUCTION DU MOT DE PASSE');
        for temps:=1 to 750 do;
        end;
page(output);
end;

(*************)

segment procedure CHANGECONFIGURATION;

var choixconf : char;


function CARON(c:char):boolean;

begin
if ord(c) in [78,79,110,111] then caron:=true
                             else caron:=false;
end;

(*************)

begin
page(output);
writeln('EST-IL PERMIS DE : ');
writeln;


write('PARALPERMIS ');
repeat read(keyboard,choixconf) until caron(choixconf);
writeln(choixconf);
writeln;
if ord(choixconf) in [79,111]
        then paralpermis:=true
        else paralpermis:=false;
write('PERPENDPERMIS ');
repeat read(keyboard,choixconf) until caron(choixconf);
writeln(choixconf);
writeln;
```

```
if ord(choixconf) in [79,111]
        then perpendpermis:=true
        else perpendpermis:=false;
write('OKPOINTREGLE ');
repeat read(keyboard,choixconf) until caron(choixconf);
writeln(choixconf);
writeln;
if ord(choixconf) in [79,111]
        then okpointregle:=true
        else okpointregle:=false;
write('DRDEUXROT ');
repeat read(keyboard,choixconf) until caron(choixconf);
writeln(choixconf);
writeln;
if ord(choixconf) in [79,111]
        then drdeuxrot  :=true
        else drdeuxrot  :=false;
write('DRROTPERMIS ');
repeat read(keyboard,choixconf) until caron(choixconf);
writeln(choixconf);
writeln;
if ord(choixconf) in [79,111]
        then drrotpermis:=true
        else drrotpermis:=false;
write('MILIEUPERMIS ');
repeat read(keyboard,choixconf) until caron(choixconf);
writeln(choixconf);
writeln;
if ord(choixconf) in [79,111]
        then milieupermis:=true
        else milieupermis:=false;
read(keyboard,choixconf);
end;


(*************)

function IDENTIF:boolean;

var f : text;
    motint : passwd;
    i : integer;

begin
page(output);
writeln('QUEL EST VOTRE MOT DE PASSE ?');
readln(keyboard,motint);
if motcomp=motint then identif:=true
                  else begin
                       identif:=false;
                       writeln('IDENTIFICATION INCORRECTE');
                       for i:=1 to 750 do;
                       end;
page(output);
end;

(*************)
```

```
procedure AFFMENU;


begin
page(output);
writeln('POSSIBILITES OFFERTES : ');
writeln('^^^^^^^^^^^^^^^^^^^^^^^^');
writeln; writeln;
writeln('M) CHANGER LE MOT DE PASSE');
writeln;
writeln('C) CHANGER LA CONFIGURATION');
writeln;
writeln('.) POINT       R) REGLE      ');
writeln;
writeln('K) COMPAS      G) GOMME      ');
writeln;
writeln('<RET>  FIN DU PROGRAMME');
writeln;writeln;writeln;
end;

(****************)

begin
textmode;
page(output);

if identif then begin
        affmenu;
        write('VOTRE CHOIX : ');
        read(choix);
        writeln;
        while (ord(choix)<>32) and (choix <> 'K')
       and (choix <> 'R') and (choix <> '.')
       and (choix <> 'G') do
          begin
          case ord(choix) of
               77,109 : changemotpasse; (*M*)
                67,99 : changeconfiguration; (*C*)
                        end;
          affmenu;
          write('VOTRE CHOIX : ');
          read(choix);
          writeln;
          end;
                end
          else begin
         writeln('QUEL MODE VOULEZ VOUS ?');
         repeat read(keyboard,choix) until
                  ord(choix) in [32,75,107,82,114,46,71,103,63];
                end;

page(output);
action:=choix;
grafmode;
end;

(*********************)
```

```
procedure ECRIREMOTPASSE(nomfich:string; mot:passwd);

var f : text;
    i : integer;

begin
rewrite(f,nomfich);
for i:=1 to longpass do
      mot[i]:=chr(ord(mot[i]) - cache);
writeln(f,mot);
close(f,lock);
end;

(************)

procedure TERMINAISON;

const oui=1;
      non=0;

var f : text;

begin
ecriremotpasse(concat(vol,'BOF.TEXT'),motcomp);
rewrite(f,concat(vol,'CONF.TEXT'));
if paralpermis then writeln(f,oui)
   else writeln(f,non);
if perpendpermis then writeln(f,oui)
   else writeln(f,non);
if okpointregle then writeln(f,oui)
   else writeln(f,non);
if drdeuxrot then writeln(f,oui)
   else writeln(f,non);
if drrotpermis then writeln(f,oui)
   else writeln(f,non);
if milieupermis then writeln(f,oui)
   else writeln(f,non);
close(f,lock);
end;


(*************************************)


begin
(*PROGRAMME PRINCIPAL*)

init;
repeat read(keyboard,action) until
    ord(action) in [32,75,107,82,114,46,71,103,36];
                   (*RET,K,R,.,G,$,? *)
while ord(action) <> 32 do

      begin
      case ord(action) of
              46 : pointmode;
         82,114 : latte;
         75,107 : compas;
         71,103 : gomme;
             36 : configuration;
                end;
      end;
```

```
terminaison;
end.
```